რ. სამხარაძე

ტესტების კრებული საგანში

"ობიექტზე ორიენტირებული დაპროგრამება. C#"

“ტექნიკური უნივერსიტეტი”

საქართველოს ტექნიკური უნივერსიტეტი

რ. სამხარაძე

ტესტების კრებული საგანში

"ობიექტზე ორიენტირებული დაპროგრამება. C#"

რეგისტრირებულია სტუ-ის
სარედაქციო-საგამომცემლო საბჭოს
მიერ. 15.12.2010, ოქმი №4

თბილისი
2012

კრებულში წარმოდგენილია C# ენისთვის შემუშავებულია ტესტები. ტესტებში ასახულია C# ენის საფუძვლები, მმართველი ოპერატორები, მასივები, კლასები, ინკაფსულაცია, პოლიმორფიზმი და მემკვიდრეობითობა, აბსტრაქტული კლასები და მეთოდები, კონსტრუქტორები და მემკვიდრეობითობა, თვისებები, ერთგანზომილებიანი ინდექსატორები, განსაკუთრებული სიტუაციების დამუშავების ხერხები, დელეგატები, სახელების სივრცე და ინფორმაციის შეტანა-გამოტანის საკითხები, აგრეთვე მონაცემთა ბაზებთან მუშაობის პრინციპები. წარმოდგენილი ტესტების საშუალებით შესაძლებელია სტუდენტების ტესტირების ჩატარება.

განკუთვნილია კომპიუტერული სისტემებისა და ქსელების სპეციალობის სტუდენტებისათვის (შიფრი - 2201).

რეცენზენტები: ტექნიკის მეცნიერებათა კანდიდატი, ასოცირებული პროფესორი მ. ანდღულაძე
ტექნიკის მეცნიერებათა კანდიდატი, ასოცირებული პროფესორი გ. ჭიკაძე

ISBN 978-9941-20-029-8

http://gtu.ge/publishinghouse/

სარჩევი

I ნაწილი. C# ენა
თავი 2. C# ენის საფუძვლები

საწყისი ცნებები

2.1.1. C# ენაზე შედგენილი პროგრამა იწყება და მთავრდება:
ა. კვადრატული ფრჩხილებით
ბ. ფიგურული ფრჩხილებით
გ. მრგვალი ფრჩხილებით

2.1.2. // სიმბოლოებით იწყება:
ა. მრავალსტრიქონიანი კომენტარი
ბ. კომენტარი ამ სიმბოლოებით არ იწყება
გ. ერთსტრიქონიანი კომენტარი

2.1.3. /* და */ სიმბოლოებს შორის მოთავსებულია:
ა. მრავალსტრიქონიანი კომენტარი
ბ. ერთსტრიქონიანი კომენტარი
გ. კომენტარი ამ სიმბოლოებით არ იწყება

2.1.4. პროგრამის შესრულების დროს ხდება:
ა. კომენტარების გამოტოვება
ბ. კომენტარების შესრულება
გ. კომენტარების გარდაქმნა მთელ ტიპად

2.1.5. ცვლადების გამოცხადებას ამთავრებს:
ა. წერტილ-მძიმე ";"
ბ. მძიმე ","
გ. ორი წერტილი ":"

2.1.6. ოპერატორები ერთმანეთისაგან გამოიყოფა:
ა. მძიმით ","
ბ. წერტილ-მძიმით ";"
გ. ორი წერტილით ":"

2.1.7. int სიტყვა იწყებს:
ა. წილადის გამოცხადებას
ბ. ლოგიკური ცვლადის გამოცხადებას
გ. მთელი რიცხვის გამოცხადებას

2.1.8. double სიტყვა იწყებს:
ა. მთელი რიცხვის გამოცხადებას
ბ. ორმაგი სიზუსტის წილადის გამოცხადებას
გ. ლოგიკური ცვლადის გამოცხადებას

2.1.9. bool სიტყვა იწყებს:
ა. ლოგიკური ცვლადის გამოცხადებას

ბ. მთელი რიცხვის გამოცხადებას
გ. წილადის გამოცხადებას

2.1.10. ცვლადების სახელები აუცილებლად უნდა იწყებოდეს:
ა. რიცხვით
ბ. სიმბოლოთი
გ. სპეციალური სიმბოლოთი

2.1.11. ცვლადი გამოცხადებული უნდა იყოს:
ა. მის გამოყენებამდე
ბ. მისი გამოყენების შემდეგ
გ. არასოდეს არ უნდა იყოს გამოცხადებული

2.1.12. შეიძლება თუ არა პროგრამის ერთ სტრიქონში ორი ან მეტი ოპერატორის მოთავსება:
ა. ნაწილობრივ შეიძლება
ბ. არ შეიძლება
გ. შეიძლება

2.1.13. მონაცემების შესატანად გამოიყენება:
ა. label კომპონენტი
ბ. textBox კომპონენტი
გ. button კომპონენტი

2.1.14. მონაცემების გამოსატანად გამოიყენება:
ა. label კომპონენტი
ბ. textBox კომპონენტი
გ. button კომპონენტი

2.1.15. მეთოდის შესასრულებლად გამოიყენება:
ა. button კომპონენტი
ბ. label კომპონენტი
გ. textBox კომპონენტი

2.1.16. textBox კომპონენტის Text თვისებას აქვს:
ა. მთელი ტიპი
ბ. წილადი ტიპი
გ. სტრიქონული ტიპი

2.1.17. textBox კომპონენტში შეტანილი მონაცემები ენიჭება ამავე კომპონენტის:
ა. Text თვისებას
ბ. Font თვისებას
გ. Enabled თვისებას

2.1.18. Convert.ToInt32 მეთოდი თავის არგუმენტს გარდაქმნის:
ა. წილად რიცხვად
ბ. სტრიქონად
გ. მთელ რიცხვად

2.1.19. მინიჭების ოპერატორის მარცხნივ მოთავსებულ ცვლადს და მარჯვნივ მოთავსებულ გამოსახულებას უნდა ჰქონდეს:

ა. ერთნაირი ტიპი

ბ. სხვადასხვა ტიპი

გ. მხოლოდ მთელი ტიპი

2.1.20. ToString() მეთოდი ნებისმიერი ტიპის ცვლადს გარდაქმნის:

ა. მთელ რიცხვად

ბ. სტრიქონად

გ. სიმბოლოდ

2.1.21. C# ენაში ზედა და ქვედა რეგისტრის სიმბოლოები:

ა. არ არის განსხვავებული

ბ. წილადი ტიპისაა

გ. განსხვავებულია

2.1.22. კოდის რომელი ფრაგმენტი შეიცავს შეცდომას:

ა.
```
{
        int Ricxvebi;
        ricxvebi = 51;
}
```
ბ.
```
{
        int ricxvebi;
        ricxvebi = 51;
}
```
გ.
```
{
        int Ricxvebi;
        Ricxvebi = 51;
}
```

2.1.23. C# პროგრამის ფაილებს აქვს:

ა. .exe გაფართოება

ბ. .cs გაფართოება

გ. .cpp გაფართოება

C# ენის საბაზო ტიპები

2.2.1. C# ენაში არსებობს:

ა. ჩვეულებრივი და მიმართვითი ტიპები

ბ. მხოლოდ ჩვეულებრივი ტიპები

გ. მხოლოდ მიმართვითი ტიპები

2.2.2. ჩვეულებრივი ტიპის ცვლადები შეიცავს:

ა. ობიექტების მისამართებს

ბ. კონკრეტულ მნიშვნელობებს

გ. მხოლოდ სტრიქონებს

2.2.3. მიმართვითი ტიპის ცვლადები შეიცავს:
ა. ობიექტების მისამართებს
ბ. კონკრეტულ მნიშვნელობებს
გ. მხოლოდ სტრიქონებს

2.2.4. byte ტიპი აღწერს:
ა. წილადს
ბ. ლოგიკურ ცვლადს
გ. 8 ბიტიანი უნიშნო მთელი რიცხვს

2.2.5. char ტიპი აღწერს:
ა. მთელ რიცხვს
ბ. სიმბოლოს
გ. ლოგიკურ ცვლადს

2.2.6. decimal ტიპი აღწერს:
ა. ციფრული ტიპის საფინანსო ცვლადს
ბ. მთელ რიცხვს
გ. წილადს

2.2.7. float ტიპი აღწერს:
ა. ორმაგი სიზუსტის წილადს
ბ. ლოგიკურ ცვლადს
გ. ერთმაგი სიზუსტის წილადს

2.2.8. long ტიპი აღწერს:
ა. გრძელ მთელ რიცხვს
ბ. წილადს
გ. ლოგიკურ ცვლადს

2.2.9. sbyte ტიპი აღწერს:
ა. 8 ბიტიან ნიშნიან მთელ რიცხვს
ბ. მთელ რიცხვს
გ. წილადს

2.2.10. short ტიპი აღწერს:
ა. წილადს
ბ. ლოგიკურ ცვლადს
გ. მოკლე მთელ რიცხვს

2.2.11. uint ტიპი აღწერს:
ა. მთელ რიცხვს
ბ. უნიშნო მთელ რიცხვს
გ. ლოგიკურ ცვლადს

2.2.12. ulong ტიპი აღწერს:
ა. უნიშნო გრძელ მთელ რიცხვს

ბ. მთელ რიცხვს
გ. წილადს

2.2.13. ushort ტიპი აღწერს:
ა. წილადს
ბ. ლოგიკურ ცვლადს
გ. უნიშნო მოკლე მთელ რიცხვს

2.2.14. byte ტიპის ცვლადი მნიშვნელობებს იღებს დიაპაზონში:
ა. 0÷65537
ბ. -32768÷32767
გ. 0÷255

2.2.15. bool ტიპის ცვლადი იღებს:
ა. მხოლოდ true მნიშვნელობებს
ბ. მხოლოდ false მნიშვნელობებს
გ. true და false მნიშვნელობებს

2.2.16. int ტიპის ცვლადი იკავებს:
ა. 8 ბაიტს
ბ. 4 ბაიტს
გ. 1 ბაიტს

2.2.17. long ტიპის ცვლადი იკავებს:
ა. 1 ბაიტს
ბ. 8 ბაიტს
გ. 4 ბაიტს

2.2.18. byte ტიპის ცვლადი იკავებს:
ა. 1 ბაიტს
ბ. 8 ბაიტს
გ. 4 ბაიტს

2.2.19. char ტიპის ცვლადი იკავებს:
ა. 2 ბაიტს
ბ. 1 ბაიტს
გ. 4 ბაიტს

2.2.20. decimal ტიპის ცვლადი იკავებს:
ა. 8 ბაიტს
ბ. 16 ბაიტს
გ. 4 ბაიტს

2.2.21. double ტიპის ცვლადი იკავებს:
ა. 4 ბაიტს
ბ. 16 ბაიტს
გ. 8 ბაიტს

2.2.22. float ტიპის ცვლადი იკავებს:
ა. 4 ბაიტს
ბ. 16 ბაიტს
გ. 8 ბაიტს

2.2.23. sbyte ტიპის ცვლადი იკავებს:
ა. 1 ბაიტს
ბ. 2 ბაიტს
გ. 8 ბაიტს

2.2.24. short ტიპის ცვლადი იკავებს:
ა. 4 ბაიტს
ბ. 1 ბაიტს
გ. 2 ბაიტს

2.2.25. uint ტიპის ცვლადი იკავებს:
ა. 4 ბაიტს
ბ. 1 ბაიტს
გ. 8 ბაიტს

2.2.26. ulong ტიპის ცვლადი იკავებს:
ა. 4 ბაიტს
ბ. 1 ბაიტს
გ. 8 ბაიტს

2.2.27. ushort ტიპის ცვლადი იკავებს:
ა. 4 ბაიტს
ბ. 2 ბაიტს
გ. 8 ბაიტს

2.2.28. მთელრიცხვა ტიპებია:
ა. char, byte, short, int
ბ. char, string, ulong, uint
გ. sbyte, short, double, long

2.2.29. ნიშნიან რიცხვებში უფროსი ბიტი:
ა. შედის რიცხვის შემადგენლობაში
ბ. გამოიყენება ნიშნის მისათითებლად
გ. არ გამოიყენება ნიშნის მისათითებლად

2.2.30. უნიშნო რიცხვებში უფროსი ბიტი:
ა. გამოიყენება ნიშნის მისათითებლად
ბ. შედის რიცხვის შემადგენლობაში
გ. არ შედის რიცხვის შემადგენლობაში

2.2.31. უნიშნო რიცხვების აბსოლუტური მნიშვნელობა ნიშნიანი რიცხვების მნიშვნელობას აღემატება:
ა. ოთხჯერ

ბ. სამჯერ
გ. ორჯერ

2.2.32. მცურავწერტილიანი ტიპებია:
ა. float და double
ბ. int და char
გ. char და string

2.2.33. C# ენაში სიმბოლოების წარმოსადგენად გამოიყენება:
ა. სიმბოლოების არც ერთი ნაკრები არ გამოიყენება
ბ. ASCII სიმბოლოების ნაკრები
გ. Unicode სიმბოლოების ნაკრები

2.2.34. Unicode სიმბოლო იკავებს:
ა. 16 ბიტს
ბ. 8 ბიტს
გ. 4 ბიტს

2.2.35. ლიტერალი არის:
ა. ცვლადი
ბ. მუდმივა
გ. ცვლადი და მუდმივა

2.2.36. ლიტერალი შეიძლება იყოს:
ა. მხოლოდ სტრიქონი
ბ. მხოლოდ წილადი
გ. ნებისმიერი ტიპის მნიშვნელობა

2.2.37. u ან U სუფიქსი გამოიყენება:
ა. უნიშნო მთელრიცხვა ლიტერალის აღსანიშნავად
ბ. უნიშნო წილადი ლიტერალის აღსანიშნავად
გ. სტრიქონის აღსანიშნავად

2.2.38. f ან F სუფიქსი გამოიყენება:
ა. მთელრიცხვა ლიტერალის აღსანიშნავად
ბ. წილადი ლიტერალის აღსანიშნავად
გ. სტრიქონის აღსანიშნავად

2.2.39. m ან M სუფიქსი გამოიყენება:
ა. მთელრიცხვა ლიტერალის აღსანიშნავად
ბ. decimal ტიპის ლიტერალის აღსანიშნავად
გ. სიმბოლოს აღსანიშნავად

2.2.40. ul ან UL სუფიქსი გამოიყენება:
ა. სტრიქონის აღსანიშნავად
ბ. გრძელი უნიშნო წილადი ლიტერალის აღსანიშნავად
გ. გრძელი უნიშნო მთელრიცხვა ლიტერალის აღსანიშნავად

2.2.41. ToDecimal() მეთოდი მონაცემებს გარდაქმნის:
ა. მთელი ტიპის მონაცემად
ბ. სიმბოლოდ
გ. საფინანსო ტიპის მონაცემად

2.2.42. ToBoolean() მეთოდი მონაცემებს გარდაქმნის:
ა. ლოგიკური ტიპის მონაცემად
ბ. სიმბოლოდ
გ. საფინანსო ტიპის მონაცემად

2.2.43. ToDouble() მეთოდი მონაცემებს გარდაქმნის:
ა. წილადად
ბ. სიმბოლოდ
გ. საფინანსო ტიპის მონაცემად

2.2.44. ToChar() მეთოდი მონაცემებს გარდაქმნის:
ა. ლოგიკური ტიპის მონაცემად
ბ. სიმბოლოდ
გ. საფინანსო ტიპის მონაცემად

2.2.45. რა მნიშვნელობებს იღებს bool ტიპის მნიშვნელობა:
ა. -1 და 0
ბ. -1 და 1
გ. true და false

2.2.46. მთელრიცხვა ტიპებია:
ა. char, byte, sbyte, short, ushort, int, uint, long, ulong
ბ. string, double
გ. bool, char, int, float

2.2.47. ნიშნიანი მთელრიცხვა ტიპებია:
ა. uint, bool
ბ. sbyte, short, int, long
გ. ushort, char

2.2.48. უნიშნო მთელრიცხვა ტიპებია:
ა. byte, uint, ulong, ushort
ბ. sbyte, short
გ. uint, bool

2.2.49. რაში მდგომარეობს განსხვავება ნიშნიან და უნიშნო მთელ რიცხვებს შორის:
ა. ნიშნიან რიცხვებში უფროსი თანრიგი ეთმობა რიცხვის ნიშანს
ბ. უნიშნო რიცხვებში უფროსი თანრიგი ეთმობა რიცხვის ნიშანს
გ. ნიშნიან რიცხვებში უმცროსი თანრიგი ეთმობა რიცხვის ნიშანს

ცვლადები და მათი ინიციალიზება

2.3.1. ცვლადი არის:
ა. მეხსიერების სახელდებული უბანი, რომელსაც მნიშვნელობა ენიჭება
ბ. მუდმივა, რომელსაც სახელი აქვს
გ. მხოლოდ წილადი ტიპის მონაცემი

2.3.2. ცვლადის ინიციალიზება არის:
ა. ცვლადისათვის მნიშვნელობის მინიჭება მისი გამოცხადების გარეშე
ბ. ცვლადისათვის მნიშვნელობის მინიჭება მისი გამოცხადების დროს
გ. ცვლადისათვის მნიშვნელობის მინიჭება მისი გამოცხადების შემდეგ

2.3.3. ცვლადის დინამიკური ინიციალიზება სრულდება:
ა. პროგრამის ნებისმიერ ადგილში
ბ. პროგრამის დასაწყისში
გ. მხოლოდ მასივის გამოცხადების წინ

2.3.4. როდის სრულდება ცვლადის დინამიკური ინიციალიზება:
ა. პროგრამის დამთავრების შემდეგ
ბ. პროგრამის შესრულების დროს
გ. პროგრამის დაწყებამდე

2.3.5. როგორია ცვლადის გამოცხადების სინტაქსი:
ა. ტიპი
ბ. ცვლადის_სახელი
გ. ტიპი ცვლადის_სახელი

2.3.6. როგორია ცვლადის ინიციალიზაციის ოპერატორის სინტაქსი:
ა. ტიპი ცვლადის_სახელი = მნიშვნელობა;
ბ. ტიპი = მნიშვნელობა;
გ. ტიპი = ცვლადის_სახელი;

ოპერატორები

არითმეტიკის ოპერატორები

2.4.1. ოპერატორების რამდენი ძირითადი კლასი არსებობს C# ენაში:
ა. 4
ბ. 3
გ. 5

2.4.2. ოპერატორების რომელი ძირითადი კლასები არსებობს:
ა. მხოლოდ არითმეტიკის და შედარების
ბ. არითმეტიკის, ლოგიკის, შედარებისა და ბიტობრივი
გ. მხოლოდ ლოგიკის და ბიტობრივი

2.4.3. C# ენაში არსებობს შემდეგი ოპერატორები:
ა. მხოლოდ ლოგიკის და ბიტობრივი
ბ. მხოლოდ არითმეტიკის და შედარების
გ. არითმეტიკის, ლოგიკის, შედარებისა და ბიტობრივი

2.4.4. რა არის ოპერატორი:
ა. ოპერატორი არის მასივი, რომელიც კომპილატორს ატყობინებს, თუ რომელი ოპერაცია უნდა შესრულდეს
ბ. ოპერატორი არის სიმბოლო, რომელიც კომპილატორს ატყობინებს, თუ რომელი ოპერაცია უნდა შესრულდეს
გ. ოპერატორი არის ობიექტი, რომელიც კომპილატორს ატყობინებს, თუ რომელი ოპერაცია უნდა შესრულდეს

2.4.5. არითმეტიკის ოპერატორები გამოიყენება:
ა. სიმბოლოების მიმართ
ბ. სტრიქონების მიმართ
გ. მთელი და წილადი რიცხვების მიმართ

2.4.6. როცა გაყოფის ოპერატორს ვიყენებთ მთელი რიცხვების მიმართ, მაშინ შედეგი იქნება:
ა. წილადი
ბ. მთელი რიცხვი
გ. სიმბოლო

2.4.7. როცა გაყოფის ოპერატორს ვიყენებთ წილადი რიცხვების მიმართ, მაშინ შედეგი იქნება:
ა. წილადი
ბ. მთელი რიცხვი
გ. სტრიქონი

2.4.8. როცა წილადი იყოფა მთელ რიცხვზე, მაშინ შედეგი იქნება:
ა. მთელი რიცხვი
ბ. წილადი
გ. სიმბოლო

2.4.9. როცა მთელი რიცხვი იყოფა წილადზე, მაშინ შედეგი იქნება:
ა. მთელი რიცხვი
ბ. ლოგიკური მნიშვნელობა
გ. წილადი

2.4.10. რომელია არითმეტიკის ოპერატორები:
ა. <, >, !=
ბ. +, -, *, /, %, ++, --
გ. =, >, !=

2.4.11. რომელი ოპერატორი გამოიყენება ნაშთის მისაღებად:
ა. %
ბ. /
გ. *

2.4.12. (2 + 3) % 2 გამოსახულების გამოთვლის შედეგი იქნება:
ა. 0
ბ. 1
გ. 2

ინკრემენტისა და დეკრემენტის ოპერატორები

2.4.13. ინკრემენტის ოპერატორი:
ა. ორით ზრდის თავისი ოპერანდის მნიშვნელობას
ბ. ერთით ამცირებს თავისი ოპერანდის მნიშვნელობას
გ. ერთით ზრდის თავისი ოპერანდის მნიშვნელობას

2.4.14. დეკრემენტის ოპერატორი:
ა. ორით ზრდის თავისი ოპერანდის მნიშვნელობას
ბ. ერთით ამცირებს თავისი ოპერანდის მნიშვნელობას
გ. ერთით ზრდის თავისი ოპერანდის მნიშვნელობას

2.4.15. რა ფუნქციას ასრულებს ++ ოპერატორი:
ა. ერთით ზრდის თავისი ოპერანდის მნიშვნელობას
ბ. ერთით ამცირებს თავისი ოპერანდის მნიშვნელობას
გ. ორით ზრდის თავისი ოპერანდის მნიშვნელობას

2.4.16. რა ფუნქციას ასრულებს -- ოპერატორი:
ა. ერთით ზრდის თავისი ოპერანდის მნიშვნელობას
ბ. ერთით ამცირებს თავისი ოპერანდის მნიშვნელობას
გ. ორით ზრდის თავისი ოპერანდის მნიშვნელობას

2.4.17. როცა ინკრემენტის ოპერატორი მითითებულია ოპერანდის წინ, მაშინ გვაქვს:
ა. პოსტფიქსური ინკრემენტი
ბ. პრეფიქსური დეკრემენტი
გ. პრეფიქსური ინკრემენტი

2.4.18. როცა ინკრემენტის ოპერატორი მითითებულია ოპერანდის შემდეგ, მაშინ გვაქვს:
ა. პოსტფიქსური ინკრემენტი
ბ. პოსტფიქსური დეკრემენტი
გ. პრეფიქსური ინკრემენტი

2.4.19. როცა დეკრემენტის ოპერატორი მითითებულია ოპერანდის წინ, მაშინ გვაქვს:
ა. პოსტფიქსური ინკრემენტი
ბ. პრეფიქსური დეკრემენტი
გ. პრეფიქსური ინკრემენტი

2.4.20. როცა დეკრემენტის ოპერატორი მითითებულია ოპერანდის შემდეგ, მაშინ გვაქვს:
ა. პოსტფიქსური ინკრემენტი
ბ. პოსტფიქსური დეკრემენტი
გ. პრეფიქსური ინკრემენტი

2.4.21. პრეფიქსური ინკრემენტის შემთხვევაში:

ა. ჯერ შესრულდება ოპერანდის მნიშვნელობის გაზრდა, შემდეგ კი მისი გამოყენება გამოსახულებაში

ბ. ჯერ შესრულდება ოპერანდის გამოყენება გამოსახულებაში, შემდეგ კი მისი მნიშვნელობის გაზრდა

გ. ჯერ შესრულდება ოპერანდის მნიშვნელობის შემცირება, შემდეგ კი მისი გამოყენება გამოსახულებაში

2.4.22. პოსტფიქსური ინკრემენტის შემთხვევაში:

ა. ჯერ შესრულდება ოპერანდის მნიშვნელობის გაზრდა, შემდეგ კი მისი გამოყენება გამოსახულებაში

ბ. ჯერ შესრულდება ოპერანდის გამოყენება გამოსახულებაში, შემდეგ კი მისი მნიშვნელობის გაზრდა

გ. ჯერ შესრულდება ოპერანდის მნიშვნელობის შემცირება, შემდეგ კი მისი გამოყენება გამოსახულებაში

2.4.23. პრეფიქსური დეკრემენტის შემთხვევაში:

ა. ჯერ შესრულდება ოპერანდის მნიშვნელობის გაზრდა, შემდეგ კი მისი გამოყენება გამოსახულებაში

ბ. ჯერ შესრულდება ოპერანდის გამოყენება გამოსახულებაში, შემდეგ კი მისი მნიშვნელობის გაზრდა

გ. ჯერ შესრულდება ოპერანდის მნიშვნელობის შემცირება, შემდეგ კი მისი გამოყენება გამოსახულებაში

2.4.24. პოსტფიქსური დეკრემენტის შემთხვევაში:

ა. ჯერ შესრულდება ოპერანდის მნიშვნელობის გაზრდა, შემდეგ კი მისი გამოყენება გამოსახულებაში

ბ. ჯერ შესრულდება ოპერანდის გამოყენება გამოსახულებაში, შემდეგ კი მისი მნიშვნელობის შემცირება

გ. ჯერ შესრულდება ოპერანდის მნიშვნელობის შემცირება, შემდეგ კი მისი გამოყენება გამოსახულებაში

2.4.25.
```
{
        int ricxvi = 5, shedegi;
        shedegi = ++ricxvi;
}
```
ამ კოდის შესრულების შედეგად shedegi ცვლადი მიიღებს მნიშვნელობას:

ა. 4

ბ. 5

გ. 6

2.4.26.
```
{
        int ricxvi = 5, shedegi;
        shedegi = ricxvi++;
}
```
ამ კოდის შესრულების შედეგად shedegi ცვლადი მიიღებს მნიშვნელობას:

ა. 4

ბ. 5
გ. 6

2.4.27. {
int ricxvi = 5, shedegi;
shedegi = --ricxvi;
}
ამ კოდის შესრულების შედეგად shedegi ცვლადი მიიღებს მნიშვნელობას:
ა. 5
ბ. 4
გ. 6

2.4.28. {
int ricxvi = 5, shedegi;
shedegi = ricxvi--;
}
ამ კოდის შესრულების შედეგად shedegi ცვლადი მიიღებს მნიშვნელობას:
ა. 4
ბ. 5
გ. 6

2.4.29. {
int ricxvi = 5, shedegi;
shedegi = ++ricxvi;
}
ამ კოდის შესრულების შედეგად ricxvi ცვლადი მიიღებს მნიშვნელობას:
ა. 4
ბ. 5
გ. 6

2.4.30. {
int ricxvi = 5, shedegi;
shedegi = ricxvi++;
}
ამ კოდის შესრულების შედეგად ricxvi ცვლადი მიიღებს მნიშვნელობას:
ა. 4
ბ. 5
გ. 6

2.4.31. {
int ricxvi = 5, shedegi;
shedegi = --ricxvi;
}
ამ კოდის შესრულების შედეგად ricxvi ცვლადი მიიღებს მნიშვნელობას:
ა. 5
ბ. 4
გ. 6

2.4.32. {
```
int ricxvi = 5, shedegi;
shedegi = ricxvi--;
```
}
ამ კოდის შესრულების შედეგად ricxvi ცვლადი მიიღებს მნიშვნელობას:

ა. 4

ბ. 5

გ. 6

შედარებისა და ლოგიკის ოპერატორები

2.5.1. შედარების ოპერატორები გასცემენ:

ა. char ტიპის მნიშვნელობებს

ბ. bool ტიპის მნიშვნელობებს

გ. int ტიპის მნიშვნელობებს

2.5.2. bool ტიპის მნიშვნელობების შედარება შეიძლება:

ა. მეტობაზე ან ნაკლებობაზე

ბ. ტოლობაზე ან მეტობაზე

გ. ტოლობაზე ან უტოლობაზე

2.5.3. მონაცემთა როგორი ტიპების მიმართ გამოიყენება შედარების ოპერატორები >, <, >=, <= :

ა. ჩამოთვლადი ტიპების მიმართ

ბ. არაჩამოთვლადი ტიპების მიმართ

გ. მხოლოდ სტრიქონების მიმართ

2.5.4. {
```
int ricxvi1 = 5, ricxvi2 = 10;
bool shedegi;
shedegi = ricxvi1 > ricxvi2;
```
}
ამ კოდის შესრულების შედეგად shedegi ცვლადი მიიღებს მნიშვნელობას:

ა. false

ბ. true

გ. 5

2.5.5. {
```
int ricxvi1 = 5, ricxvi2 = 10;
bool shedegi;
shedegi = ricxvi1 < ricxvi2;
```
}
ამ კოდის შესრულების შედეგად shedegi ცვლადი მიიღებს მნიშვნელობას:

ა. false

ბ. true

გ. 5

2.5.6. {
```
int ricxvi1 = 5, ricxvi2 = 10;
```

```
        bool shedegi;
        shedegi = ricxvi1 == ricxvi2;
    }
```
ამ კოდის შესრულების შედეგად shedegi ცვლადი მიიღებს მნიშვნელობას:

ა. 5
ბ. true
გ. false

2.5.7.
```
    {
        int ricxvi1 = 5, ricxvi2 = 10;
        bool shedegi;
        shedegi = ricxvi1 != ricxvi2;
    }
```
ამ კოდის შესრულების შედეგად shedegi ცვლადი მიიღებს მნიშვნელობას:

ა. 5
ბ. true
გ. false

2.5.8.
```
    {
        int ricxvi1 = 5, ricxvi2 = 10;
        bool shedegi;
        shedegi = ricxvi1 >= ricxvi2;
    }
```
ამ კოდის შესრულების შედეგად shedegi ცვლადი მიიღებს მნიშვნელობას:

ა. 5
ბ. true
გ. false

2.5.9.
```
    {
        int ricxvi1 = 5, ricxvi2 = 10;
        bool shedegi;
        shedegi = ricxvi1 <= ricxvi2;
    }
```
ამ კოდის შესრულების შედეგად shedegi ცვლადი მიიღებს მნიშვნელობას:

ა. 5
ბ. true
გ. false

2.5.10. რა არის შედარების ოპერატორების შესრულების შედეგი:

ა. true და false
ბ. -1 და 0
გ. -1 და 1

2.5.11. როგორი ტიპი უნდა ჰქონდეს ლოგიკის ოპერატორებს:

ა. int
ბ. bool
გ. short

2.5.12. რომელია შედარების ოპერატორები:
ა. +, -, *
ბ. ==, !=, >, <, <=, >=
გ. /, %, ++

2.5.13. რომელია ლოგიკის ოპერატორები:
ა. ==, !=, >
ბ. -, /, *
გ. &, |, ^, &&, ||, !

2.5.14. & ოპერატორი გასცემს true მნიშვნელობას მაშინ, როცა:
ა. მისი ერთ-ერთი ოპერანდის მნიშვნელობაა false
ბ. მისი ორივე ოპერანდის მნიშვნელობაა false
გ. მისი ორივე ოპერანდის მნიშვნელობაა true

2.5.15. | ოპერატორი გასცემს true მნიშვნელობას მაშინ, როცა:
ა. მისი ერთ-ერთი ოპერანდის მნიშვნელობაა true
ბ. მისი ორივე ოპერანდის მნიშვნელობაა false
გ. მისი ერთ-ერთი ოპერანდის მნიშვნელობაა 0

2.5.16. ! ოპერატორი გასცემს true მნიშვნელობას მაშინ, როცა:
ა. მისი ოპერანდის მნიშვნელობაა false
ბ. მისი ორივე ოპერანდის მნიშვნელობაა false
გ. მისი ოპერანდის მნიშვნელობაა true

2.5.17. ^ ოპერატორი გასცემს true მნიშვნელობას მაშინ, როცა:
ა. მის ორივე ოპერანდს ერთნაირი მნიშვნელობები აქვს
ბ. მის ორივე ოპერანდს განსხვავებული მნიშვნელობები აქვს
გ. მისი ერთ-ერთი ოპერანდის მნიშვნელობაა true

2.5.18. & ოპერატორი გასცემს false მნიშვნელობას მაშინ, როცა:
ა. მისი ერთ-ერთი ოპერანდის მნიშვნელობაა false
ბ. მისი ორივე ოპერანდის მნიშვნელობაა true
გ. მისი ერთ-ერთი ოპერანდის მნიშვნელობაა true

2.5.19. | ოპერატორი გასცემს false მნიშვნელობას მაშინ, როცა:
ა. მისი ერთ-ერთი ოპერანდის მნიშვნელობაა true
ბ. მისი ორივე ოპერანდის მნიშვნელობაა false
გ. მისი ერთ-ერთი ოპერანდის მნიშვნელობაა false

2.5.20. ! ოპერატორი გასცემს false მნიშვნელობას მაშინ, როცა:
ა. მისი ოპერანდის მნიშვნელობაა false
ბ. მისი ორივე ოპერანდის მნიშვნელობაა false
გ. მისი ოპერანდის მნიშვნელობაა true

2.5.21. ^ ოპერატორი გასცემს false მნიშვნელობას მაშინ, როცა:
ა. მის ორივე ოპერანდს ერთნაირი მნიშვნელობები აქვს

ბ. მის ორივე ოპერანდს განსხვავებული მნიშვნელობები აქვს
გ. მისი ერთ-ერთი ოპერანდის მნიშვნელობაა true

2.5.22. {
```
bool b1 = true, b2 = false, shedegi;
shedegi = b1 & b2;
```
}
ამ კოდის შესრულების შედეგად shedegi ცვლადი მიიღებს მნიშვნელობას:
ა. true
ბ. false
გ. 5

2.5.23. {
```
bool b1 = true, b2 = false, shedegi;
shedegi = b1 | b2;
```
}
ამ კოდის შესრულების შედეგად shedegi ცვლადი მიიღებს მნიშვნელობას:
ა. true
ბ. false
გ. 5

2.5.24. {
```
bool b1 = true, b2 = false, shedegi;
shedegi = b1 ^ b2;
```
}
ამ კოდის შესრულების შედეგად shedegi ცვლადი მიიღებს მნიშვნელობას:
ა. true
ბ. false
გ. 5

2.5.25. {
```
bool b1 = true, shedegi;
shedegi = !b1;
```
}
ამ კოდის შესრულების შედეგად shedegi ცვლადი მიიღებს მნიშვნელობას:
ა. true
ბ. 5
გ. false

2.5.26. {
```
bool b1 = true, b2 = false, b3 = true, b4 = false, shedegi;
shedegi = !b1 || b2 && !b3 || b4;
```
}
ამ კოდის შესრულების შედეგად shedegi ცვლადი მიიღებს მნიშვნელობას:
ა. true
ბ. 5
გ. false

2.5.27. {

```
bool b1 = true, b2 = false, b3 = true, shedegi;
shedegi = b1 || b2 && b3;
```

}

ამ კოდის შესრულების შედეგად shedegi ცვლადი მიიღებს მნიშვნელობას:

ა. true

ბ. 5

გ. false

2.5.28. რომელია ლოგიკის სწრაფი ოპერატორი:

ა. &&

ბ. &

გ. ||

2.5.29. ლოგიკის სწრაფი ოპერატორია:

ა. &

ბ. !

გ. ||

2.5.30. თუ && ოპერატორის შესრულებისას პირველმა ოპერანდმა მიიღო false მნიშვნელობა, მაშინ შედეგი იქნება:

ა. true

ბ. false

გ. 5

2.5.31. თუ || ოპერატორის შესრულებისას პირველმა ოპერანდმა მიიღო true მნიშვნელობა, მაშინ შედეგი იქნება:

ა. true

ბ. false

გ. 5

მინიჭების ოპერატორი. შედგენილი ოპერატორები

2.6.1. როგორია მინიჭების ოპერატორის სინტაქსი:

ა. ცვლადი = გამოსახულება;

ბ. გამოსახულება = ცვლადი;

გ. ცვლადი + გამოსახულება;

2.6.2. შეიძლება თუ არა, რომ ცვლადსა და გამოსახულებას სხვადასხვა ტიპი ჰქონდეს:

ა. კი

ბ. არა

გ. ნაწილობრივ

2.6.3. ricxvi1 = ricxvi2 = ricxvi3 = 25; ოპერატორის შესრულების შედეგად როგორი იქნება ricxvi2 ცვლადის მნიშვნელობა:

ა. 25

ბ. 52

გ. 0

2.6.4. შეიძლება თუ არა, რომ მინიჭების ოპერატორის საშუალებით რამდენიმე ცვლადს ერთდროულად მივანიჭოთ მნიშვნელობა:

ა. ნაწილობრივ

ბ. არა

გ. კი

2.6.5. რომელია შედგენილი ოპერატორები:

ა. +=, -+, *=, /=, %=, &=, |=, ^=

ბ. !==, <==

გ. ==, >==

2.6.6. როგორია შედგენილი ოპერატორის სინტაქსი:

ა. ცვლადი ოპერაცია გამოსახულება;

ბ. ცვლადი =ოპერაცია გამოსახულება;

გ. ცვლადი ოპერაცია= გამოსახულება;

2.6.7. რა უპირატესობა აქვს მინიჭების შედგენილი ოპერატორის გამოყენებას მინიჭების ჩვეულებრივი ოპერატორის გამოყენებასთან შედარებით:

ა. მინიჭების შედგენილი ოპერატორი უფრო კომპაქტურია და მისი გამოყენებისას ჩქარდება კოდი კომპილირება

ბ. მინიჭების შედგენილი ოპერატორი ნაკლებად კომპაქტურია და მისი გამოყენებისას არ ჩქარდება კოდი კომპილირება

გ. არავითარი უპირატესობა არ აქვს

2.6.8. jami += 25; მინიჭების ოპერატორი ქვემოთ მოყვანილი რომელი ოპერატორის ანალოგიურია:

ა. jami = jami + jami + 25;

ბ. jami = jami - 25;

გ. jami = jami + 25;

2.6.9. jami -= 25; მინიჭების ოპერატორი ქვემოთ მოყვანილი რომელი ოპერატორის ანალოგიურია:

ა. jami = jami - jami - 25;

ბ. jami = jami - 25;

გ. jami = 25 - jami;

2.6.10. jami *= 25; მინიჭების ოპერატორი ქვემოთ მოყვანილი რომელი ოპერატორის ანალოგიურია:

ა. jami = jami * jami * 25;

ბ. jami = jami - 25;

გ. jami = jami * 25;

2.6.11. jami /= 25; მინიჭების ოპერატორი ქვემოთ მოყვანილი რომელი ოპერატორის ანალოგიურია:

ა. jami = jami / jami / 25;

ბ. jami = jami / 25;
გ. jami = jami % 25;

2.6.12. jami %= 25; მინიჭების ოპერატორი ქვემოთ მოყვანილი რომელი ოპერატორის ანალოგიურია:
ა. jami = jami % 25;
ბ. jami = jami % jami % 25;
გ. jami = jami / 25;

ოპერატორების პრიორიტეტები

2.7.1. * და + ოპერატორებს შორის რომელს აქვს მაღალი პრიორიტეტი:
ა. *
ბ. +
გ. მათ თანაბარი პრიორიტეტი აქვს

2.7.2. - და / ოპერატორებს შორის რომელს აქვს მაღალი პრიორიტეტი:
ა. -
ბ. /
გ. მათ თანაბარი პრიორიტეტი აქვს

2.7.3. {
int ricxvi1 = 5, ricxvi2 = 3, ricxvi3 = 2, ricxvi4 = 4, ricxvi5 = 1,shedegi;
shedegi = ricxvi1 - ricxvi2 / ricxvi3 + ricxvi4 * ricxvi5;
}
ამ კოდის შესრულების შედეგად shedegi ცვლადი მიიღებს მნიშვნელობას:
ა. 7.5
ბ. 5.7
გ. 3

2.7.4. {
int ricxvi1 = 1, ricxvi2 = 3, ricxvi3 = 2, ricxvi4 = 5, ricxvi5 = 4, ricxvi6 = 6;
double shedegi = (ricxvi1 + ricxvi2) / ricxvi3 +
(ricxvi4 - ricxvi5) * ricxvi6;
}
ამ კოდის შესრულების შედეგად shedegi ცვლადი მიიღებს მნიშვნელობას:
ა. 5
ბ. 8
გ. 3

2.7.5. რომელი მსჯელობაა სწორი:
ა. / ოპერატორს აქვს - ოპერატორზე მაღალი პრიორიტეტი
ბ. + ოპერატორს აქვს % ოპერატორზე მაღალი პრიორიტეტი
გ. - ოპერატორს აქვს + ოპერატორზე მაღალი პრიორიტეტი

გამოსახულებები. მათემატიკის მეთოდები

2.8.1. რისთვის გამოიყენება ტაბულაციის სიმბოლოები და ინტერვალები:
ა. პროგრამის კომპილირების დასაჩქარებლად
ბ. პროგრამის შესრულების დასაჩქარებლად
გ. პროგრამის კითხვადობის გასაუმჯობესებლად

2.8.2. რომელ სტანდარტულ კლასშია აღწერილი მათემატიკური ფუნქციები:
ა. System.Math
ბ. System.Object
გ. System.Exception

2.8.3. რომელ ტიპს იყენებს მათემატიკური ფუნქციების უმრავლესობა:
ა. long
ბ. double
გ. uint

2.8.4. Floor(x) მეთოდი თავის არგუმენტს ამრგვალებს:
ა. ნაკლებობით და მეტობით
ბ. მეტობით
გ. ნაკლებობით

2.8.5. Ceiling(x) მეთოდი თავის არგუმენტს ამრგვალებს:
ა. ნაკლებობით და მეტობით
ბ. მეტობით
გ. ნაკლებობით

2.8.6. Sqrt(x) მეთოდი:
ა. იღებს კვადრატულ ფესვს თავისი არგუმენტიდან
ბ. აჰყავს კვადრატში თავისი არგუმენტი
გ. ამრგვალებს თავის არგუმენტს

2.8.7. Round(x,y) მეთოდი:
ა. გასცემს პირველ არგუმენტს ისეთი სახით, რომ მას წილად ნაწილში ჰქონდეს მეორე არგუმენტით მითითებული თანრიგების რაოდენობა
ბ. გასცემს მის ორ არგუმენტს შორის მაქსიმალურს
გ. გასცემს მის ორ არგუმენტს შორის მინიმალურს

2.8.8. Max(x,y) მეთოდი:
ა. გასცემს მის ორ არგუმენტს შორის მინიმალურს
ბ. გასცემს მისი პირველი არგუმენტის კვადრატს
გ. გასცემს მის ორ არგუმენტს შორის მაქსიმალურს

2.8.9. Min(x,y) მეთოდი:
ა. გასცემს მის ორ არგუმენტს შორის მაქსიმალურს
ბ. გასცემს მის ორ არგუმენტს შორის მინიმალურს
გ. გასცემს მისი მეორე არგუმენტის კვადრატს

2.8.10. Abs(x) მეთოდი:

ა. გასცემს მისი არგუმენტის უარყოფით მნიშვნელობას

ბ. გასცემს მისი არგუმენტის აბსოლუტურ მნიშვნელობას

გ. გასცემს მისი არგუმენტის კვადრატს

2.8.11. Pow(x,y) მეთოდი:

ა. პირველი არგუმენტი აჰყავს მეორე არგუმენტით მითითებულ ხარისხში

ბ. მეორე არგუმენტი აჰყავს პირველი არგუმენტით მითითებულ ხარისხში

გ. გასცემს მისი არგუმენტის კვადრატს

2.8.12. Log(x) მეთოდი:

ა. გასცემს მისი არგუმენტის კვადრატს

ბ. გასცემს მისი არგუმენტის უარყოფით მნიშვნელობას

გ. გასცემს მისი არგუმენტის ლოგარითმს

2.8.13. Sin(x) მეთოდი:

ა. გასცემს მისი არგუმენტის ლოგარითმს

ბ. გასცემს მისი არგუმენტის კოსინუსს

გ. გასცემს მისი არგუმენტის სინუსს

2.8.14. Cos(x) მეთოდი:

ა. გასცემს მისი არგუმენტის სინუსს

ბ. გასცემს მისი არგუმენტის კოსინუსს

გ. გასცემს მისი არგუმენტის ლოგარითმს

2.8.15. Tan(x) მეთოდი:

ა. გასცემს მისი არგუმენტის ტანგენსს

ბ. გასცემს მისი არგუმენტის სინუსს

გ. გასცემს მისი არგუმენტის კოსინუსს

2.8.16. Exp(x) მეთოდი:

ა. გასცემს მისი არგუმენტის კვადრატს

ბ. e რიცხვი აჰყავს მითითებულ ხარისხში

გ. გასცემს მისი არგუმენტის კოსინუსს

2.8.17. რომელი მინიჭების ოპერატორი გამოთვლის მოცემული გამოსახულების მნიშვნელობას:

$$y = \frac{\sqrt{x^3 + e^x}}{\sin x}$$

ა. y = Math.Sqrt(Math.Pow(x,3) + Math.Exp(x)) / Math.Sin(x);

ბ. y = Math.Sqrt((Math.Pow(x,3) + Math.Exp(x)) / Math.Sin(x));

გ. y = (Math.Pow(x,3) + Math.Exp(x)) / Math.Sin(x);

ტიპების გარდაქმნა

2.9.1. ჩვეულებრივ შემთხვევაში ერთი ტიპის მქონე ცვლადს:
ა. ნაწილობრივ უნდა მივანიჭოთ ამავე ტიპის მნიშვნელობა
ბ. არ უნდა მივანიჭოთ ამავე ტიპის მნიშვნელობა
გ. უნდა მივანიჭოთ ამავე ტიპის მნიშვნელობა

2.9.2. ჩვეულებრივ შემთხვევაში მინიჭების ოპერატორის მარცხნივ მოთავსებულ ცვლადს და მარჯვნივ მოთავსებულ გამოსახულებას:
ა. ერთნაირი ტიპები არ უნდა ჰქონდეს
ბ. ერთნაირი ტიპები უნდა ჰქონდეს
გ. ერთნაირი ტიპები ნაწილობრივ უნდა ჰქონდეს

2.9.3. შეიძლება თუ არა, რომ double ტიპის ცვლადს მივანიჭოთ int ტიპის მნიშვნელობა:
ა. კი
ბ. არა
გ. ნაწილობრივ

2.9.4. თუ მინიჭების ოპერატორში გამოიყენება თავსებადი ტიპები, მაშინ:
ა. მინიჭების ოპერატორის მარჯვნივ მოთავსებული გამოსახულების ტიპი ავტომატურად გარდაიქმნება მინიჭების ოპერატორის მარცხნივ მოთავსებული ცვლადის ტიპად
ბ. მინიჭების ოპერატორის მარჯვნივ მოთავსებული გამოსახულების ტიპი ავტომატურად არ გარდაიქმნება მინიჭების ოპერატორის მარცხნივ მოთავსებული ცვლადის ტიპად
გ. მინიჭების ოპერატორის მარცხნივ მოთავსებული გამოსახულების ტიპი ავტომატურად გარდაიქმნება მინიჭების ოპერატორის მარჯვნივ მოთავსებული ცვლადის ტიპად

2.9.5. თავსებადია თუ არა int და bool ტიპები:
ა. ნაწილობრივ
ბ. კი
გ. არა

2.9.6. შესრულდება თუ არა ტიპების ავტომატური გარდაქმნა decimal და double ტიპებს შორის:
ა. არა
ბ. კი
გ. ნაწილობრივ

2.9.7. შესრულდება თუ არა ტიპების ავტომატური გარდაქმნა decimal და float ტიპებს შორის:
ა. არა
ბ. კი
გ. ნაწილობრივ

2.9.8. ტიპების ავტომატური გარდაქმნა არ სრულდება:
ა. double და float ტიპებს შორის
ბ. long და int ტიპებს შორის
გ. decimal და float ტიპებს შორის

2.9.9. ტიპების ავტომატური გარდაქმნა არ სრულდება:
ა. long და int ტიპებს შორის

ბ. short და int ტიპებს შორის
გ. decimal და double ტიპებს შორის

2.9.10. ტიპების ავტომატური გარდაქმნა არ სრულდება:
ა. char და bool ტიპებს შორის
ბ. long და int ტიპებს შორის
გ. double და float ტიპებს შორის

2.9.11. ტიპების ავტომატური გარდაქმნა არ სრულდება:
ა. short და int ტიპებს შორის
ბ. int და char ტიპებს შორის
გ. double და float ტიპებს შორის

2.9.12. ტიპების ავტომატური გარდაქმნა არ სრულდება:
ა. int და bool ტიპებს შორის
ბ. double და float ტიპებს შორის
გ. short და int ტიპებს შორის

2.9.13. როდის სრულდება ტიპების ავტომატური გარდაქმნა:
ა. როცა ორივე ტიპი თავსებადია და საბოლოო ტიპი მეტია საწყის ტიპზე
ბ. როცა ორივე ტიპი თავსებადია და საბოლოო ტიპი ნაკლებია საწყის ტიპზე
გ. როცა ორივე ტიპი არათავსებადია და საბოლოო ტიპი მეტია საწყის ტიპზე

2.9.14. როდის სრულდება ტიპების გაფართოებადი გარდაქმნა:
ა. როცა ორივე ტიპი არათავსებადია და საბოლოო ტიპი მეტია საწყის ტიპზე
ბ. როცა ორივე ტიპი თავსებადია და საბოლოო ტიპი ნაკლებია საწყის ტიპზე
გ. როცა ორივე ტიპი თავსებადია და საბოლოო ტიპი მეტია საწყის ტიპზე

2.9.15. საბოლოო ტიპი მეტია საწყის ტიპზე - ეს იმას ნიშნავს, რომ:
ა. საბოლოო ტიპის მქონე ცვლადის მიერ დაკავებული თანრიგების რაოდენობა ნაკლებია საწყისი ტიპის მქონე ცვლადის მიერ დაკავებული თანრიგების რაოდენობაზე
ბ. საბოლოო ტიპის მქონე ცვლადის მიერ დაკავებული თანრიგების რაოდენობა მეტია საწყისი ტიპის მქონე ცვლადის მიერ დაკავებული თანრიგების რაოდენობაზე
გ. საბოლოო ტიპის მქონე ცვლადის მიერ დაკავებული თანრიგების რაოდენობა ტოლია საწყისი ტიპის მქონე ცვლადის მიერ დაკავებული თანრიგების რაოდენობაზე

2.9.16. ტიპების გაფართოებადი გარდაქმნის შესასრულებლად:
ა. ყველა რიცხვითი ტიპი მთელრიცხვა და მცურავწერტილიანი ტიპების ჩათვლით თავსებადი უნდა იყოს
ბ. ყველა რიცხვითი ტიპი მთელრიცხვა და მცურავწერტილიანი ტიპების ჩათვლით თავსებადი არ უნდა იყოს
გ. ყველა რიცხვითი ტიპი მთელრიცხვა და ლოგიკური ტიპების ჩათვლით თავსებადი უნდა იყოს

2.9.17. შეიძლება თუ არა long ტიპის ცვლადი გარდაიქმნას double ტიპის ცვლადად:
ა. ნაწილობრივ
ბ. არა
გ. კი

2.9.18. შეიძლება თუ არა double ტიპის ცვლადის ავტომატური გარდაქმნა long ტიპის ცვლადად:
ა. კი
ბ. არა
გ. ნაწილობრივ

2.9.19. რატომ არ შეიძლება double ტიპის ცვლადის ავტომატური გარდაქმნა long ტიპის ცვლადად:
ა. იმიტომ, რომ ასეთი გარდაქმნა არის გაფართოებადი
ბ. იმიტომ, რომ ასეთი გარდაქმნა არ არის გაფართოებადი
გ. იმიტომ, რომ ასეთი გარდაქმნა ნაწილობრივ არის გაფართოებადი

2.9.20. რატომ არ შეიძლება long ტიპის ცვლადის ავტომატური გარდაქმნა double ტიპის ცვლადად:
ა. იმიტომ, რომ ასეთი გარდაქმნა არის გაფართოებადი
ბ. იმიტომ, რომ ასეთი გარდაქმნა არ არის გაფართოებადი
გ. იმიტომ, რომ ასეთი გარდაქმნა ნაწილობრივ არის გაფართოებადი

2.9.21. რა არის ტიპის დაყვანის ოპერაცია:
ა. ეს არის bool ტიპის string ტიპად გარდაქმნის ოპერაცია
ბ. ეს არ არის ერთი ტიპის მეორე ტიპად გარდაქმნის ოპერაცია
გ. ეს არის ერთი ტიპის მეორე ტიპად გარდაქმნის ოპერაცია

2.9.22. როგორია ტიპის დაყვანის ოპერაციის სინტაქსი:
ა. გამოსახულება (საბოლოო_ტიპი);
ბ. (საბოლოო_ტიპი) გამოსახულება;
გ. (საწყისი_ტიპი) გამოსახულება;

2.9.23. ტიპის დაყვანას სხვანაირად ეწოდება:
ა. ცხადი გარდაქმნა
ბ. არაცხადი გარდაქმნა
გ. ნაწილობრივი გარდაქმნა

2.9.24. {
```
        int ricxvi1 = 60;
        byte ricxvi2;
        ricxvi2 = ( byte ) ricxvi1;
}
```
კოდის შესრულების შედეგად:
ა. არ მოხდება ინფორმაციის დაკარგვა
ბ. მოხდება ინფორმაციის დაკარგვა
გ. ნაწილობრივ მოხდება ინფორმაციის დაკარგვა

2.9.25. {
```
        char simbolo;
        byte baiti = 10;
        simbolo = ( char ) baiti;
}
```

კოდის შესრულების შედეგად:

ა. ტიპების დაყვანის ოპერაცია არ სრულდება საერთოდ

ბ. ტიპების დაყვანის ოპერაცია სრულდება თავსებად ტიპებს შორის

გ. ტიპების დაყვანის ოპერაცია სრულდება შეუთავსებად ტიპებს შორის

2.9.26.
```
{
        double wiladi1 = 10.93, wiladi2 = 3.21;
        int mteli;
        mteli = ( int ) ( wiladi1 / wiladi2 );
}
```
კოდის შესრულების შედეგად:

ა. int ტიპის გამოსახულება გარდაიქმნება double ტიპად

ბ. double ტიპის გამოსახულება არ გარდაიქმნება int ტიპად

გ. double ტიპის გამოსახულება გარდაიქმნება int ტიპად

2.9.27. ტიპების დაყვანის ოპერაცია სრულდება მაშინ, როცა:

ა. ერთი ტიპი ნაწილობრივ გარდაიქმნება მეორე ტიპად

ბ. ერთი ტიპი ავტომატურად გარდაიქმნება მეორე ტიპად

გ. ერთი ტიპი ავტომატურად არ გარდაიქმნება მეორე ტიპად

2.9.28. როცა ტიპებს შორის სრულდება კუმშვადი გარდაქმნა, მაშინ:

ა. ინფორმაცია შეიძლება დაიკარგოს

ბ. ინფორმაცია შეიძლება არ დაიკარგოს

გ. ინფორმაცია შეიძლება ნაწილობრივ დაიკარგოს

2.9.29. კუმშვადი გარდაქმნის დროს შეიძლება თუ არა ინფორმაციის დაკარგვა:

ა. კი

ბ. არა

გ. არასოდეს

2.9.30. მცურავწერტილიანი მნიშვნელობის გარდაქმნისას მთელ მნიშვნელობად ხდება თუ არა წილადი ნაწილის დაკარგვა:

ა. არა

ბ. კი

გ. არასოდეს

2.9.31. თუ long ტიპის დაყვანისას int ტიპზე, long ტიპის მქონე ცვლადის მნიშვნელობა არ ეტევა int ტიპის დიაპაზონის ფარგლებში, მაშინ:

ა. უმცროსი თანრიგის ბიტები იკარგება

ბ. უფროსი თანრიგის ბიტები იკარგება

გ. უფროსი თანრიგის ბიტები არ იკარგება

2.9.32. თუ long ტიპის დაყვანისას int ტიპზე, long ტიპის მქონე ცვლადის მნიშვნელობა ეტევა int ტიპის დიაპაზონის ფარგლებში, მაშინ:

ა. უფროსი თანრიგის ბიტები არ იკარგება

ბ. უფროსი თანრიგის ბიტები იკარგება

გ. უმცროსი თანრიგის ბიტები იკარგება

2.9.33. როცა ხდება მცურავწერტილიანი მნიშვნელობის დაყვანა მთელ მნიშვნელობაზე, მაშინ:

ა. წილადი ნაწილი არ იკარგება

ბ. წილადი ნაწილი იკარგება

გ. წილადი ნაწილი ნაწილობრივ იკარგება

2.9.34. {

```
double wiladi = 5.09;
int mteli;
mteli = ( int ) wiladi1;
```

}

კოდის შესრულების შედეგად mteli ცვლადი მიიღებს მნიშვნელობას:

ა. 5.09

ბ. 09

გ. 5

2.9.35. byte ტიპის გარდაქმნისას char ტიპად ინფორმაცია:

ა. ნაწილობრივ იკარგება

ბ. იკარგება

გ. არ იკარგება

2.9.36. არითმეტიკულ გამოსახულებაში თუ ერთ ოპერანდს აქვს decimal ტიპი, მაშინ მეორე ოპერანდი:

ა. ავტომატურად გარდაიქმნება decimal ტიპად, თუ მას არ აქვს double და float ტიპი

ბ. ავტომატურად გარდაიქმნება decimal ტიპად, თუ მას აქვს double და float ტიპი

გ. ავტომატურად არ გარდაიქმნება decimal ტიპად, თუ მას არ აქვს double და float ტიპი

2.9.37. არითმეტიკულ გამოსახულებაში თუ ერთ ოპერანდს აქვს double ტიპი, მაშინ მეორე ოპერანდი:

ა. ავტომატურად არ გარდაიქმნება ამ ტიპად

ბ. ავტომატურად გარდაიქმნება ამ ტიპად

გ. ნაწილობრივ გარდაიქმნება ამ ტიპად

2.9.38. არითმეტიკულ გამოსახულებაში თუ ერთ ოპერანდს აქვს float ტიპი, მაშინ მეორე ოპერანდი:

ა. ავტომატურად გარდაიქმნება ამ ტიპად

ბ. ავტომატურად არ გარდაიქმნება ამ ტიპად

გ. ნაწილობრივ გარდაიქმნება ამ ტიპად

2.9.39. არითმეტიკულ გამოსახულებაში თუ ერთ ოპერანდს აქვს ulong ტიპი, მაშინ პირველი ოპერანდი:

ა. ავტომატურად გარდაიქმნება ამ ტიპად თუ მეორე ოპერანდს არ აქვს sbyte, short, int და long ტიპი

ბ. ავტომატურად არ გარდაიქმნება ამ ტიპად თუ მეორე ოპერანდს არ აქვს sbyte, short, int და long ტიპი

გ. ნაწილობრივ გარდაიქმნება ამ ტიპად თუ მეორე ოპერანდს არ აქვს sbyte, short, int და long ტიპი

2.9.40. არითმეტიკულ გამოსახულებაში თუ ერთ ოპერანდს აქვს uint ტიპი და მეორეს sbyte, short ან int ტიპი, მაშინ:
ა. ორივე ოპერანდი ავტომატურად გარდაიქმნება uint ტიპად
ბ. ორივე ოპერანდი ავტომატურად გარდაიქმნება long ტიპად
გ. ორივე ოპერანდი ავტომატურად გარდაიქმნება int ტიპად

2.9.41. არითმეტიკულ გამოსახულებაში თუ ერთ ოპერანდს აქვს uint ტიპი, მაშინ მეორე ოპერანდი:
ა. ნაწილობრივ გარდაიქმნება ამ ტიპად
ბ. ავტომატურად არ გარდაიქმნება ამ ტიპად
გ. ავტომატურად გარდაიქმნება ამ ტიპად

2.9.42. არითმეტიკულ გამოსახულებაში თუ არ იყო გამოყენებული ტიპების ავტომატური გარდაქმნის არც ერთი წესი, მაშინ:
ა. ორივე ოპერანდი double ტიპად გარდაიქმნება
ბ. ორივე ოპერანდი long ტიპად გარდაიქმნება
გ. ორივე ოპერანდი int ტიპად გარდაიქმნება

2.9.43. არითმეტიკულ გამოსახულებაში:
ა. შეუძლებელია ყველა ტიპის ავტომატურად შეთავსება
ბ. შესაძლებელია ყველა ტიპის ავტომატურად შეთავსება
გ. შესაძლებელია ყველა ტიპის ნაწილობრივ შეთავსება

2.9.44. არითმეტიკულ გამოსახულებაში float ან double ტიპის ავტომატურად გარდაქმნა decimal ტიპად:
ა. შესაძლებელია
ბ. შეუძლებელია
გ. შესაძლებელია ნაწილობრივ

2.9.45. არითმეტიკულ გამოსახულებაში ulong ტიპის ავტომატურად გარდაქმნა სხვა ნიშნიან მთელრიცხვა ტიპთან:
ა. შეუძლებელია
ბ. შესაძლებელია
გ. შესაძლებელია ნაწილობრივ

2.9.46. არითმეტიკულ გამოსახულებაში float ან double ტიპის ავტომატურად გარდაქმნა decimal ტიპად:
ა. ნაწილობრივ შესაძლებელია ტიპის აშკარა გარდაქმნის ოპერაციის გამოყენებით
ბ. შეუძლებელია ტიპის აშკარა გარდაქმნის ოპერაციის გამოყენებით
გ. შესაძლებელია ტიპის აშკარა გარდაქმნის ოპერაციის გამოყენებით

2.9.47. არითმეტიკულ გამოსახულებაში ulong ტიპის ავტომატურად გარდაქმნა სხვა ნიშნიან მთელრიცხვა ტიპთან:
ა. შეუძლებელია ტიპის აშკარა გარდაქმნის ოპერაციის გამოყენებით
ბ. შესაძლებელია ტიპის აშკარა გარდაქმნის ოპერაციის გამოყენებით
გ. ნაწილობრივ შესაძლებელია ტიპის აშკარა გარდაქმნის ოპერაციის გამოყენებით

2.9.48. თუ არითმეტიკულ გამოსახულების გამოთვლისას არ იყო გამოყენებული ტიპების ავტომატური გარდაქმნის წესები, მაშინ:

ა. char, sbyte, byte, ushort და short ტიპები ნაწილობრივ გარდაიქმნება int ტიპად

ბ. char, sbyte, byte, ushort და short ტიპები არ გარდაიქმნება int ტიპად

გ. char, sbyte, byte, ushort და short ტიპები გარდაიქმნება int ტიპად

2.9.49. როცა არითმეტიკულ ოპერაციას ვასრულებთ byte ტიპის ცვლადებზე, მაშინ:

ა. შედეგს ექნება double ტიპი

ბ. შედეგს ექნება int ტიპი

გ. შედეგს ექნება byte ტიპი

2.9.50. როცა არითმეტიკულ ოპერაციას ვასრულებთ char ტიპის ცვლადებზე, მაშინ:

ა. შედეგს ექნება int ტიპი

ბ. შედეგს ექნება byte ტიპი

გ. შედეგს ექნება char ტიპი

თავი 3. მმართველი ოპერატორები

ამორჩევის ოპერატორები. if ოპერატორი

3.1.1. რომელი კატეგორიის მმართველი ოპერატორები არსებობს:
ა. შედარების, არითმეტიკული
ბ. არითმეტიკული, ლოგიკური
გ. ამორჩევის, იტერაციული, გადასვლის

3.1.2. რა მნიშვნელობებს იღებს პირობა if გამოსახულებაში:
ა. true და false
ბ. bool და 0
გ. -1 და 1

3.1.3. როგორია if ოპერატორის სინტაქსი:
ა. if (პირობა) ოპერატორი_1;
else ოპერატორი_2;
ბ. if else ოპერატორი_1;
(პირობა) ოპერატორი_2;
გ. else (პირობა) ოპერატორი_1;
if ოპერატორი_2;

3.1.4. როდის გამოიყენება ჩადგმული if ოპერატორი:
ა. ჩადგმული if ოპერატორი გამოიყენება მასივების დამატებით შესამოწმებლად
ბ. ჩადგმული if ოპერატორი გამოიყენება კლასების დამატებით შესამოწმებლად
გ. ჩადგმული if ოპერატორი გამოიყენება მონაცემების დამატებით შესამოწმებლად მას შემდეგ, რაც წინა if ოპერატორის პირობამ მიიღო true ან false მნიშვნელობა

3.1.5.
```
{
        int ricxvi = 5, shedegi;
        if ( ricxvi == 5 ) shedegi = 10;
                else       shedegi = 25;
}
```
ამ კოდის შესრულების შედეგად shedegi ცვლადი მიიღებს მნიშვნელობას:
ა. 25
ბ. 10
გ. 15

3.1.6. if ოპერატორის პირობა არის გამოსახულება, რომელსაც აქვს:
ა. სიმბოლური ტიპი
ბ. მთელი ტიპი
გ. ლოგიკური ტიპი

3.1.7. ჩადგმულია if ოპერატორი, რომელიც:
ა. მოთავსებულია სხვა if ოპერატორში
ბ. არაა მოთავსებული სხვა if ოპერატორში
გ. მოთავსებულია სხვა for ოპერატორში

3.1.8. ჩადგმულ if ოპერატორში else ნაწილი:
ა. ეკუთვნის უახლოეს for ოპერატორს
ბ. არ ეკუთვნის უახლოეს if ოპერატორს
გ. ეკუთვნის უახლოეს if ოპერატორს

3.1.9. {
```
int ricxvi1 = 5, ricxvi2 = 10, shedegi;
if ( ricxvi1 > 5 ) shedegi = 50;
    else if ( ricxvi2 <= 10 ) shedegi = 100;
}
```
ამ კოდის შესრულების შედეგად shedegi ცვლადი მიიღებს მნიშვნელობას:
ა. 50
ბ. 100
გ. 150

ხილვადობის უბანი

3.2.1. ლოკალურია ცვლადი, რომელიც გამოცხადებულია:
ა. მეთოდის შიგნით
ბ. მეთოდის გარეთ
გ. მხოლოდ if ოპერატორში

3.2.2. გლობალურია ცვლადი, რომელიც გამოცხადებულია:
ა. მეთოდის შიგნით
ბ. მეთოდის გარეთ
გ. for ოპერატორში

3.2.3. ხილვადობის უბანში გამოცხადებული ცვლადი:
ა. მიუწვდომელია ამ უბნის გარეთ განსაზღვრული კოდისთვის
ბ. მისაწვდომია ამ უბნის გარეთ განსაზღვრული კოდისთვის
გ. მისაწვდომია მხოლოდ if ოპერატორში

3.2.4. პროგრამის ბლოკი მოთავსებულია:
ა. ფიგურულ ფრჩხილებში
ბ. მრგვალ ფრჩხილებში
გ. კვადრატულ ფრჩხილებში

3.2.5. ერთ ბლოკში რამდენიმე ოპერატორი იმ შემთხვევაში უნდა მოვათავსოთ, როცა:
ა. საჭიროა მათ შორის ციფრული კავშირის დამყარება
ბ. საჭიროა მათ შორის ფიზიკური კავშირის დამყარება
გ. საჭიროა მათ შორის ლოგიკური კავშირის დამყარება

3.2.6. თუ ცვლადი გამოცხადებულია მეთოდის დასაწყისში, მაშინ:
ა. მასთან მიმართვა შესაძლებელი იქნება მის გამოცხადებამდე
ბ. მასთან მიმართვა შესაძლებელი იქნება მხოლოდ if ოპერატორში
გ. მასთან მიმართვა შესაძლებელი იქნება მთელი კოდის ფარგლებში

3.2.7. ხილვადობის უბანში განსაზღვრული ცვლადი ხილვადობის უბნიდან გამოსვლისას:
ა. კარგავს თავის მნიშვნელობას
ბ. ინახავს თავის მნიშვნელობას
გ. ორჯერ ზრდის თავის მნიშვნელობას

3.2.8. მეთოდის შიგნით გამოცხადებული ცვლადი:
ა. ინახავს თავის მნიშვნელობას ამ მეთოდის გამოძახებებს შორის
ბ. არ ინახავს თავის მნიშვნელობას ამ მეთოდის გამოძახებებს შორის
გ. ორჯერ ზრდის თავის მნიშვნელობას

3.2.9. ცვლადის სიცოცხლის დრო:
ა. შეზღუდულია მასივის საზღვრებით
ბ. არ არის შეზღუდული მისი ხილვადობის უბნით
გ. შეზღუდულია მისი ხილვადობის უბნით

3.2.10. ხილვადობის უბანი:
ა. არ შეიძლება იყოს ჩადგმული
ბ. შეიძლება იყოს ჩადგმული
გ. შეზღუდულია მასივის საზღვრებით

3.2.11. გარე ხილვადობის უბანში გამოცხადებული ობიექტები და ცვლადები:
ა. ხილული იქნება შიდა ხილვადობის უბანში
ბ. ხილული არ იქნება შიდა ხილვადობის უბანში
გ. შეზღუდულია მასივის საზღვრებით

3.2.12. შიდა ხილვადობის უბანში გამოცხადებული ობიექტები და ცვლადები:
ა. ხილული არ იქნება გარე ხილვადობის უბანში
ბ. ხილული იქნება გარე ხილვადობის უბანში
გ. შეზღუდულია მასივის საზღვრებით

3.2.13. გარე და ჩადგმულ ბლოკებში გამოცხადებულ ცვლადებს:
ა. შეიძლება ჰქონდეს ერთნაირი სახელები
ბ. არ შეიძლება ჰქონდეს ერთნაირი სახელები
გ. აქვს შეზღუდული ზომები

3.2.14. გლობალური ცვლადები:
ა. შეზღუდული ზომების მქონეა
ბ. არ მოქმედებს ყველა მეთოდის შიგნით
გ. მოქმედებს ყველა მეთოდის შიგნით

3.2.15. გლობალური ცვლადები:
ა. არ კარგავენ თავიანთ მნიშვნელობებს მეთოდიდან გამოსვლისას
ბ. კარგავენ თავიანთ მნიშვნელობებს მეთოდიდან გამოსვლისას
გ. ფორმდება როგორც მასივი

3.2.16. მეთოდის პარამეტრები ჩართული არის თუ არა ამავე მეთოდის ხილვადობის უბანში:
ა. არა

ბ. კი
გ. ნაწილობრივ

3.2.17. ხილვადობის უბანში გამოცხადებული ცვლადი ხილულია თუ უხილავი ამ უბნის გარეთ განსაზღვრული კოდისთვის:
ა. უხილავია
ბ. ხილულია
გ. ნაწილობრივ არის ხილული

3.2.18. პროგრამული კოდის ბლოკის შექმნისას იქმნება თუ არა ხილვადობის ახალი უბანი:
ა. კი
ბ. არა
გ. ნაწილობრივ

3.2.19. შეიძლება თუ არა პროგრამული ბლოკის შიგნით ცვლადების გამოცხადება ნებისმიერ ადგილას:
ა. კი
ბ. არა
გ. ხანდახან

3.2.20. შეიძლება თუ არა ცვლადთან მიმართვა მის გამოცხადებამდე:
ა. კი
ბ. არა
გ. ზოგჯერ

3.2.21. ინახავს თუ არა ცვლადი თავის მნიშვნელობას ხილვადობის უბნიდან გამოსვლისას:
ა. არა
ბ. კი
გ. ზოგჯერ

3.2.22. ინახავს თუ არა მეთოდის შიგნით გამოცხადებული ცვლადი თავის მნიშვნელობას ამ მეთოდის გამოძახებებს შორის:
ა. ნაწილობრივ
ბ. კი
გ. არა

3.2.23. არის თუ არა ცვლადის სიცოცხლის დრო შეზღუდული მისი ხილვადობის უბნით:
ა. ნაწილობრივ
ბ. არა
გ. კი

3.2.24. შეიძლება თუ არა ჰქონდეს ერთნაირი სახელები გარე და ჩადგმულ ბლოკებში გამოცხადებულ ცვლადებს:
ა. არა
ბ. კი
გ. ზოგჯერ

ამორჩევის ოპერატორები. switch ოპერატორი

3.3.1. შიდა და გარე switch ოპერატორების case განშტოებების მუდმივებს:
ა. შეიძლება ერთნაირი მნიშვნელობები ჰქონდეს
ბ. არ შეიძლება ერთნაირი მნიშვნელობები ჰქონდეს
გ. მხოლოდ ლოგიკური მნიშვნელობები უნდა ჰქონდეს

3.3.2. როგორი ტიპი შეიძლება ჰქონდეს switch ოპერატორის გამოსახულებას:
ა. char, short, double
ბ. char, byte, int
გ. float, byte, int

3.3.3. switch ოპერატორის case განშტოებების მუდმივებს:
ა. უნდა ჰქონდეს გამოსახულების ტიპთან თავსებადი ტიპი
ბ. არ უნდა ჰქონდეს გამოსახულების ტიპთან თავსებადი ტიპი
გ. ხან უნდა ჰქონდეს გამოსახულების ტიპთან თავსებადი ტიპი, ხან არა

3.3.4. switch ოპერატორის case განშტოებების მუდმივებს შეიძლება თუ არა, რომ ერთნაირი მნიშვნელობები ჰქონდეს :
ა. ხან შეიძლება, ხან არა
ბ. შეიძლება
გ. არ შეიძლება

3.3.5. როდის შესრულდება switch ოპერატორის default განშტოების ოპერატორები:
ა. როცა case განშტოების არც ერთი მუდმივა არ შეესაბამება გამოსახულების მნიშვნელობას
ბ. როცა case განშტოების ერთ-ერთი მუდმივა შეესაბამება გამოსახულების მნიშვნელობას
გ. როცა case განშტოების ყველა მუდმივა შეესაბამება გამოსახულების მნიშვნელობას

3.3.6. break ოპერატორი მართვას გადასცემს:
ა. მომდევნო case განშტოებას
ბ. switch ოპერატორის შემდეგ მოთავსებულ ოპერატორს
გ. default განშტოებას

3.3.7. switch ოპერატორში აუცილებელია თუ არა default განშტოების მითითება:
ა. არა
ბ. კი
გ. ნაწილობრივ

3.3.8. switch ოპერატორში რა ფუნქციას ასრულებს break ოპერატორი:
ა. იგი მართვას გადასცემს switch ოპერატორის წინ მოთავსებულ ოპერატორს
ბ. იგი მართვას გადასცემს switch ოპერატორის შუაში მოთავსებულ ოპერატორს
გ. იგი მართვას გადასცემს switch ოპერატორის შემდეგ მოთავსებულ ოპერატორს

3.3.9. შეიძლება თუ არა მართვა გადაეცეს ერთი case განშტოებიდან მეორე case განშტოებას:
ა. არა
ბ. კი
გ. ნაწილობრივ

3.3.10. ერთ case განშტოებასთან დაკავშირებულმა ოპერატორების მიმდევრობამ:
ა. მართვა უნდა გადასცეს მომდევნო case განშტოების ოპერატორებს
ბ. მართვა შეიძლება გადასცეს მომდევნო case განშტოების ოპერატორებს
გ. მართვა არ უნდა გადასცეს მომდევნო case განშტოების ოპერატორებს

3.3.11. შეიძლება თუ არა, რომ switch ოპერატორის ერთი ან მეტი განშტოება დაკავშირებული იყოს ოპერატორების ერთსა და იმავე მიმდევრობასთან:
ა. არა
ბ. კი
გ. ნაწილობრივ

3.3.12. switch ოპერატორში ჩავარდნა არის შემთხვევა, როცა:
ა. switch ოპერატორის არც ერთი განშტოება არაა დაკავშირებული ოპერატორების ერთსა და იმავე მიმდევრობასთან
ბ. switch ოპერატორის მხოლოდ ერთი განშტოებაა დაკავშირებული ოპერატორების ერთსა და იმავე მიმდევრობასთან
გ. switch ოპერატორის ორი ან მეტი განშტოება დაკავშირებული ოპერატორების ერთსა და იმავე მიმდევრობასთან

3.3.13. შეიძლება თუ არა, რომ switch ოპერატორი იყოს გარე switch ოპერატორის ნაწილი:
ა. არა
ბ. კი
გ. ნაწილობრივ

იტერაციის ოპერატორები

for ოპერატორი

3.4.1. რისთვის გამოიყენება for ოპერატორი:
ა. პროგრამის შესრულების შესაწყვეტად
ბ. განშტოების შესასრულებლად
გ. ერთი ოპერატორის ან ოპერატორების ბლოკის მრავალჯერ შესასრულებლად

3.4.2. როგორია for ოპერატორის სინტაქსი:
ა. for (ციკლის ტანი; პირობა; იტერაცია) { ინიციალიზება }
ბ. for (ინიციალიზება; პირობა; იტერაცია) { ციკლის ტანი }
გ. for (იტერაცია; პირობა; ციკლის ტანი) { ინიციალიზება }

3.4.3. როგორია for ოპერატორის "ინიციალიზება" შემადგენლის დანიშნულება:
ა. მასში მმართველ ცვლადს საწყისი მნიშვნელობა ენიჭება
ბ. მასში მმართველ ცვლადს საწყისი მნიშვნელობა არ ენიჭება
გ. მოწმდება "პირობა" შემადგენელი

3.4.4. როგორია for ოპერატორის "პირობა" შემადგენლის დანიშნულება:
ა. მოწმდება მასში მითითებული გამოსახულება. თუ იგი იღებს false მნიშვნელობას, მაშინ ციკლის ტანი მეორდება, წინააღმდეგ შემთხვევაში ციკლი მთავრდება

ბ. მოწმდება მასში მითითებული გამოსახულება. თუ იგი იღებს true მნიშვნელობას, მაშინ ციკლის ტანი მეორდება, წინააღმდეგ შემთხვევაში ციკლი მთავრდება
გ. მასში მმართველ ცვლადს საწყისი მნიშვნელობა ენიჭება

3.4.5. როგორია for ოპერატორის "იტერაცია" შემადგენლის დანიშნულება:
ა. მასში მმართველ ცვლადს საწყისი მნიშვნელობა ენიჭება
ბ. მოწმდება მასში მითითებული გამოსახულება. თუ იგი იღებს true მნიშვნელობას, მაშინ ციკლის ტანი მეორდება, წინააღმდეგ შემთხვევაში ციკლი მთავრდება
გ. ესაა გამოსახულება, რომელიც განსაზღვრავს სიდიდეს (ბიჯს), რომლითაც იცვლება მმართველი ცვლადის მნიშვნელობა ციკლის ყოველი გამეორების წინ

3.4.6. for ოპერატორის ინიციალიზატორი არის გამოსახულება, რომელიც:
ა. ციკლის დაწყების წინ გამოითვლება
ბ. ციკლის დამთავრებისას გამოითვლება
გ. არასოდეს არ გამოითვლება

3.4.7. for ოპერატორის ინიციალიზატორში სრულდება:
ა. გლობალური ცვლადის გამოცხადება
ბ. ციკლის მმართველი ცვლადის ინიციალიზება
გ. სტატიკური ცვლადის ინიციალიზება

3.4.8. for ოპერატორის პირობა არის ლოგიკური გამოსახულება, რომელიც იღებს:
ა. true ან int მნიშვნელობას
ბ. true ან false მნიშვნელობას
გ. double ან false მნიშვნელობას

3.4.9. for ოპერატორის პირობის პირობა გამოითვლება:
ა. პროგრამის დასაწყისში
ბ. ციკლის ყოველი იტერაციის შემდეგ
გ. ციკლის ყოველი იტერაციის წინ

3.4.10. for ციკლის გამეორება ხდება მანამ, სანამ პირობა იღებს:
ა. false მნიშვნელობას
ბ. int მნიშვნელობას
გ. true მნიშვნელობას

3.4.11. for ციკლის დამთავრება ხდება მაშინ, როცა პირობა იღებს:
ა. false მნიშვნელობას
ბ. int მნიშვნელობას
გ. true მნიშვნელობას

3.4.12. for ციკლის იტერატორი არის გამოსახულება, რომელიც:
ა. არ ცვლის მმართველი ცვლადის მნიშვნელობას
ბ. ზრდის ან ამცირებს მმართველი ცვლადის მნიშვნელობას
გ. მხოლოდ ამცირებს მმართველი ცვლადის მნიშვნელობას

3.4.13. ციკლის მმართველი ცვლადი შეიძლება თუ არა ამცირებდეს თავის მნიშვნელობას:
ა. კი

ბ. არა
გ. ნაწილობრივ

3.4.14. for ციკლის ტანი:
ა. აუცილებლად უნდა შესრულდეს
ბ. შეიძლება არ შესრულდეს
გ. აუცილებლად უნდა შესრულდეს ორჯერ

3.4.15. for ოპერატორში ინიციალიზატორის მითითება:
ა. აუცილებელია მასივის გამოცხადებისას
ბ. აუცილებელია
გ. არ არის აუცილებელი

3.4.16. for ოპერატორში იტერატორის მითითება:
ა. აუცილებელია მასივის გამოცხადებისას
ბ. აუცილებელია
გ. არ არის აუცილებელი

3.4.17. for ოპერატორში პირობის მითითება:
ა. აუცილებელია მასივის გამოცხადებისას
ბ. აუცილებელია
გ. არ არის აუცილებელი

3.4.18. for ოპერატორში ტანის მითითება:
ა. აუცილებელია მასივის გამოცხადებისას
ბ. აუცილებელია
გ. არ არის აუცილებელი

3.4.19. for ოპერატორში დასაშვებია:
ა. რამდენიმე მმართველი ცვლადის მითითება
ბ. მხოლოდ ერთი მმართველი ცვლადის მითითება
გ. მხოლოდ ორი მმართველი ცვლადის მითითება

3.4.20. რომელი მსჯელობაა სწორი:
ა. for ციკლის შესაწყვეტად დაუშვებელია break ოპერატორის გამოყენება
ბ. for ციკლის შესაწყვეტად შეგვიძლია break ოპერატორის გამოყენება
გ. for ციკლის შესაწყვეტად შეგვიძლია case ოპერატორის გამოყენება

3.4.21. დასაშვებია თუ არა for ოპერატორის საინიციალიზაციო ნაწილში მმართველი ცვლადის გამოცხადება:
ა. ნაწილობრივ
ბ. არა
გ. კი

3.4.22. for ოპერატორის ინიციალიზატორში გამოცხადებული ცვლადის გამოყენება შეიძლება:
ა. მხოლოდ ამ ციკლის ტანში
ბ. ამ ციკლის ტანის გარეთ
გ. ნებისმიერი მეთოდის შიგნით

3.4.23. თუ ციკლის ტანში ერთი ოპერატორია მითითებული, მაშინ:
ა. მრგვალი ფრჩხილების მითითება აუცილებელია
ბ. ფიგურული ფრჩხილების მითითება აუცილებელია
გ. ფიგურული ფრჩხილების მითითება არ არის აუცილებელი

while ოპერატორი

3.5.1. როგორია while ოპერატორის სინტაქსი:
ა. while (პირობა) { ციკლის კოდი }
ბ. while (ციკლის კოდი) { პირობა }
გ. while ( ) { ციკლის კოდი }

3.5.2. while ოპერატორის პირობა არის ლოგიკური გამოსახულება, რომელიც იღებს:
ა. true ან int მნიშვნელობას
ბ. true ან false მნიშვნელობას
გ. double ან false მნიშვნელობას

3.5.3. while ციკლის გამეორება ხდება მანამ, სანამ პირობა იღებს:
ა. false მნიშვნელობას
ბ. int მნიშვნელობას
გ. true მნიშვნელობას

3.5.4. while ციკლის დამთავრება ხდება მაშინ, როცა პირობა იღებს:
ა. false მნიშვნელობას
ბ. int მნიშვნელობას
გ. true მნიშვნელობას

3.5.5. while ციკლის ტანი:
ა. აუცილებლად უნდა შესრულდეს
ბ. შეიძლება არ შესრულდეს
გ. აუცილებლად უნდა შესრულდეს ერთხელ

do-while ოპერატორი

3.6.1. როგორია do-while ოპერატორის სინტაქსი:
ა. do { ციკლის კოდი } while (პირობა)
ბ. do { პირობა } while (ციკლის კოდი)
გ. do { } while { ციკლის კოდი }

3.6.2. do-while ოპერატორის პირობა არის ლოგიკური გამოსახულება, რომელიც იღებს:
ა. true ან int მნიშვნელობას
ბ. true ან false მნიშვნელობას
გ. double ან false მნიშვნელობას

3.6.3. do-while ციკლის გამეორება ხდება მანამ, სანამ პირობა იღებს:
ა. false მნიშვნელობას

ბ. int მნიშვნელობას
გ. true მნიშვნელობას

3.6.4. do-while ციკლის დამთავრება ხდება მაშინ, როცა პირობა იღებს:
ა. false მნიშვნელობას
ბ. int მნიშვნელობას
გ. true მნიშვნელობას

3.6.5. do-while ოპერატორის ტანი:
ა. შეიძლება არ შესრულდეს
ბ. აუცილებლად უნდა შესრულდეს
გ. არასოდეს არ შესრულდება

გადასვლის ოპერატორები.

break ოპერატორი

3.7.1. რისთვის გამოიყენება break ოპერატორი:
ა. იგი გვაძლევს ციკლის იძულებით შეწყვეტისა და ციკლიდან გამოსვლის საშუალებას
ბ. იგი გვაძლევს ციკლის შესრულების გაგრძელებისა და ციკლიდან გამოსვლის საშუალებას
გ. იგი არ გვაძლევს ციკლის იძულებით შეწყვეტისა და ციკლიდან გამოსვლის საშუალებას

3.7.2. შეიძლება თუ არა break ოპერატორის გამოყენება while ციკლის შიგნით:
ა. არა
ბ. კი
გ. ნაწილობრივ

3.7.3. შეიძლება თუ არა break ოპერატორის გამოყენება do-while ციკლის შიგნით:
ა. ნაწილობრივ
ბ. არა
გ. კი

3.7.4. break ოპერატორის გამოყენება შეიძლება:
ა. მხოლოდ for ციკლის შიგნით
ბ. ნებისმიერი ციკლის შიგნით
გ. მხოლოდ while ციკლის შიგნით

3.7.5. break ოპერატორი გამოიყენება:
ა. პროგრამის დასაწყისში
ბ. ციკლის შესრულების გასაგრძელებლად
გ. ციკლის შესრულების შესაწყვეტად

3.7.6. break ოპერატორის შესრულების შედეგად:
ა. ციკლის დანარჩენი ოპერატორები არ შესრულდება
ბ. ციკლის დანარჩენი ოპერატორები მაინც შესრულდება
გ. ციკლის დანარჩენი ოპერატორები ხან შესრულდება, ხან არა

3.7.7. თუ break ოპერატორი გამოიყენება ჩადგმული ციკლის შიგნით, მაშინ იგი წყვეტს:
ა. ორივე ციკლის შესრულებას
ბ. გარე ციკლის შესრულებას
გ. შიდა ციკლის შესრულებას

continue ოპერატორი

3.8.1. რისთვის გამოიყენება continue ოპერატორი:
ა. იგი გამოიყენება ციკლის მომდევნო იტერაციაზე იძულებით გადასასვლელად
ბ. იგი გამოიყენება ციკლიდან გამოსასვლელად
გ. იგი გამოიყენება ციკლის შესაწყვეტად

3.8.2. შეიძლება თუ არა continue ოპერატორის გამოყენება while ოპერატორში:
ა. არა
ბ. კი
გ. ნაწილობრივ

3.8.3. შეიძლება თუ არა continue ოპერატორის გამოყენება do-while ოპერატორში:
ა. ნაწილობრივ
ბ. არა
გ. კი

3.8.4. continue ოპერატორის შესრულების შედეგად:
ა. ციკლის დანარჩენი ოპერატორები მაინც შესრულდება
ბ. ციკლის დანარჩენი ოპერატორები არ შესრულდება
გ. ციკლის დანარჩენი ოპერატორები ხან შესრულდება, ხან არა

3.8.5. continue ოპერატორი ახდენს:
ა. ციკლის იძულებით შეწყვეტას
ბ. მომდევნო იტერაციაზე იძულებით გადასვლას
გ. მომდევნო ოპერატორზე გადასვლას

3.8.6. continue ოპერატორის გამოყენება შეიძლება:
ა. მხოლოდ while ციკლის შიგნით
ბ. მხოლოდ for ციკლის შიგნით
გ. ნებისმიერი ციკლის შიგნით

goto ოპერატორი

3.9.1. goto ოპერატორი არის:
ა. უპირობო გადასვლის ოპერატორი
ბ. პირობითი გადასვლის ოპერატორი
გ. მრავალგანშტოებიანი ოპერატორი

3.9.2. ჭდე არის:
ა. წილადი რიცხვი
ბ. იდენტიფიკატორი, რომელსაც ორი წერტილი მოსდევს
გ. ლოგიკური გამოსახულება

3.9.3. როგორია goto ოპერატორის სინტაქსი:
ა. goto კლასი;
ბ. goto მასივი;
გ. goto ჭდე;

ტერნარული ოპერატორი

3.10.1. რამდენი ოპერანდი აქვს ტერნარულ ოპერატორს:
ა. ერთი
ბ. სამი
გ. ორი

3.10.2. ტერნარული ოპერატორის პირველი ოპერანდი არის:
ა. ლოგიკური გამოსახულება
ბ. მთელი ტიპის გამოსახულება
გ. სტრიქონი

3.10.3. თუ ტერნარული ოპერატორის პირობის გამოსახულების მნიშვნელობაა true, მაშინ:
ა. გაიცემა მესამე ოპერანდის მნიშვნელობა
ბ. გაიცემა მეორე ოპერანდის მნიშვნელობა
გ. არაფერი არ გაიცემა

3.10.4. ტერნარული ოპერატორის მეორე და მესამე ოპერანდების ტიპები:
ა. სულერთია როგორი იქნება
ბ. სხვადასხვა უნდა იყოს
გ. ერთნაირი უნდა იყოს

3.10.5. თუ ტერნარული ოპერატორის პირობის გამოსახულების მნიშვნელობაა false, მაშინ:
ა. გაიცემა მესამე ოპერანდის მნიშვნელობა
ბ. გაიცემა მეორე ოპერანდის მნიშვნელობა
გ. არაფერი არ გაიცემა

3.10.6. როგორია ტერნარული ოპრატორის სინტაქსი:
ა. გამოსახულება1 : გამოსახულება2 ? გამოსახულება3;
ბ. გამოსახულება1 ? გამოსახულება2 : გამოსახულება3;
გ. გამოსახულება1 ! გამოსახულება2 ? გამოსახულება3;

თავი 4. მასივები, ბიტობრივი ოპერაციები და ჩამოთვლები

მასივები

ერთგანზომილებიანი მასივები

4.1.1. მასივი არის:
ა. ობიექტების ერთობლიობა
ბ. მუდმივა
გ. სიმბოლო

4.1.2. მასივი გამოიყენება:
ა. ერთი ცვლადის ან ობიექტის შესანახად
ბ. რამდენიმე ცვლადის ან ობიექტის შესანახად
გ. არაფრის შესანახად არ გამოიყენება

4.1.3. მასივის ელემენტებთან მიმართვისათვის უნდა მივუთითოთ:
ა. მასივის სახელი
ბ. ელემენტის ინდექსი
გ. მასივის სახელი და ელემენტის ინდექსი

4.1.4. რა არის მასივი:
ა. მასივი არის სხვადასხვა ტიპის მქონე ობიექტების ერთობლიობა
ბ. მასივი არის ერთი ტიპის მქონე ობიექტების ერთობლიობა
გ. მასივი არის მხოლოდ სტრიქონების ერთობლიობა

4.1.5. როგორია ერთგანზომილებიანი მასივის სინტაქსი:
ა. ტიპი[ ] მასივის_სახელი = new ტიპი [ზომა];
ბ. [ ] მასივის_სახელი = new ტიპი [ზომა];
გ. ტიპი[ზომა] მასივის_სახელი = new ტიპი [];

4.1.6. რომელი ეტაპებისგან შედგება მასივის შექმნის პროცესი:
ა. მასივის სახელის გამოცხადება
ბ. პირველ ეტაპზე ხდება მასივის სახელის გამოცხადება. მეორე ეტაპზე მასივისათვის მეხსიერება გამოიყოფა და მასივის ცვლადს მიენიჭება მიმართვა მეხსიერების ამ უბანზე
გ. მასივისათვის მეხსიერების გამოყოფა და მასივის ცვლადისთვის მიმართვის მიენიჭება მეხსიერების ამ უბანზე

4.1.7. როგორია ერთგანზომილებიანი მასივის ინიციალიზაციის სინტაქსი:
ა. მასივის_სახელი = new ტიპი [] { სიდიდე1, სიდიდე2, ..., სიდიდეn };
ბ. ტიპი[n] მასივის_სახელი = new ტიპი [] { სიდიდე1, სიდიდე2, ..., სიდიდეn };
გ. ტიპი[ ] მასივის_სახელი = new ტიპი [] { სიდიდე1, სიდიდე2, ..., სიდიდეn };

4.1.8. ერთგანზომილებიან მასივში კონკრეტული ელემენტის მდებარეობა განისაზღვრება:
ა. ერთი ინდექსით
ბ. ორი ინდექსით
გ. სამი ინდექსით

4.1.9. მასივს მეხსიერება new ოპერატორის საშუალებით გამოეყოფა:
ა. დინამიკურად
ბ. სტატიკურად
გ. არ გამოეყოფა

4.1.10. რამდენ ბაიტს იკავებს მასივი int [ ] masivi = new int[15]; :
ა. 15
ბ. 60
გ. 30

4.1.11. რამდენ ბაიტს იკავებს მასივი double [ ] masivi = new double[10]; :
ა. 10
ბ. 80
გ. 8

4.1.12. მასივის პირველი ელემენტის ინდექსია:
ა. -1
ბ. 1
გ. 0

4.1.13. მასივის მესამე ელემენტის ინდექსია:
ა. 2
ბ. 3
გ. -2

4.1.14. თუ მასივი 5 ელემენტისაგან შედგება, მაშინ მისი ინდექსები მოთავსებული იქნება დიაპაზონში:
ა. 1÷5
ბ. 0÷4
გ. 0÷5

4.1.15. masivi[1] აღნიშნავს მასივის რიგით:
ა. მესამე ელემენტს
ბ. პირველ ელემენტს
გ. მეორე ელემენტს

4.1.16. double [ ] masivi = new double[3]; მასივის ელემენტებს აქვს:
ა. ლოგიკური ტიპი
ბ. მთელი ტიპი
გ. წილადი ტიპი

4.1.17. თუ მასივის ინდექსის მნიშვნელობა გასცდა დასაშვებ საზღვრებს, მაშინ:
ა. პროგრამა ავარიულად დაამთავრებს მუშაობას
ბ. პროგრამა გააგრძელებს მუშაობას
გ. პროგრამა არც ავარიულად დაამთავრდება და არ გააგრძელებს მუშაობას

ორგანზომილებიანი მასივები

4.2.1. რამდენ ბაიტს იკავებს მასივი int[,] cxrili = new int[2,3]; :
ა. 6
ბ. 24
გ. 4

4.2.2. რამდენ ბაიტს იკავებს მასივი double[,] cxrili = new double[2,3]; :
ა. 6
ბ. 48
გ. 8

4.2.3. ორგანზომილებიან მასივში პირველი ინდექსი მიუთითებს:
ა. სტრიქონების რაოდენობას
ბ. სვეტების რაოდენობას
გ. სტრიქონებისა და სვეტების საერთო რაოდენობას

4.2.4. ორგანზომილებიან მასივში მეორე ინდექსი მიუთითებს:
ა. სტრიქონების რაოდენობას
ბ. სვეტების რაოდენობას
გ. სტრიქონებისა და სვეტების საერთო რაოდენობას

4.2.5. ორგანზომილებიანი cxrili მასივის მეორე სტრიქონისა და მესამე სვეტის გადაკვეთაზე მყოფი ელემენტი შემდეგნაირად აღინიშნება:
ა. cxrili[2,3]
ბ. cxrili[3,2]
გ. cxrili[1,2]

4.2.6. x = cxrili[2,4]; მინიჭების ოპერატორის შესრულების შედეგად x ცვლადს მიენიჭება:
ა. cxrili მასივის მეორე სვეტისა და მეოთხე სტრიქონის გადაკვეთაზე მყოფი ელემენტი
ბ. cxrili მასივის მესამე სვეტისა და მეხუთე სტრიქონის გადაკვეთაზე მყოფი ელემენტი
გ. cxrili მასივის მეოთხე სვეტისა და მეორე სტრიქონის გადაკვეთაზე მყოფი ელემენტი

4.2.7. ორგანზომილებიან მასივში სტრიქონის პირველი ელემენტის ინდექსია:
ა. -1
ბ. 1
გ. 0

4.2.8. ორგანზომილებიან მასივში სვეტის პირველი ელემენტის ინდექსია:
ა. 0
ბ. 1
გ. -1

4.2.9. ორგანზომილებიან მასივში კონკრეტული ელემენტის მდებარეობა განისაზღვრება:
ა. სამი ინდექსით
ბ. ერთი ინდექსით
გ. ორი ინდექსით

გაუსწორებელი მასივები

4.3.1. სწორკუთხაა მასივი, რომელშიც:
ა. სტრიქონების სიგრძე ერთნაირია
ბ. სტრიქონების სიგრძე სხვადასხვაა
გ. სტრიქონების სიგრძე ნულის ტოლია

4.3.2. გაუსწორებელია მასივი, რომელშიც:
ა. სტრიქონების სიგრძე ნულის ტოლია
ბ. სტრიქონების სიგრძე ერთნაირია
გ. სტრიქონების სიგრძე სხვადასხვაა

4.3.3. რა არის გაუსწორებელი მასივი:
ა. გაუსწორებელი მასივი არის შიდა მასივი, რომელიც შედგება სხვადასხვა სიგრძის გარე მასივებისაგან
ბ. გაუსწორებელი მასივი არის გარე მასივი, რომელიც შედგება სხვადასხვა სიგრძის შიდა მასივებისაგან
გ. გაუსწორებელი მასივი არის მხოლოდ გარე მასივი

Length თვისება

4.4.1. Length თვისება აქვს:
ა. მხოლოდ მთელრიცხვა მასივს
ბ. ნებისმიერ მასივს
გ. მხოლოდ სტრიქონებს

4.4.2. Length თვისება ინახავს ინფორმაციას:
ა. მასივში ელემენტების საერთო რაოდენობის შესახებ
ბ. მასივის თითოეული ელემენტის სიგრძის შესახებ
გ. მასივში მთელრიცხვა ელემენტების რაოდენობის შესახებ

4.4.3. გვაქვს 7-ელემენტიანი მასივი და შევსებულია მისი პირველი 3 ელემენტი. Length თვისების მნიშვნელობა იქნება:
ა. 3
ბ. 7
გ. 10

4.4.4.
```
{
        int[ ][ ] masivi = new int[3][ ];
        masivi[0] = new int[4];
        masivi[1] = new int[1];
        masivi[2] = new int[2];
        int raodenoba = masivi.Length;
}
```
ამ კოდის შესრულების შედეგად raodenoba ცვლადში ჩაიწერება:
ა. 7

ბ. 4
გ. 3

4.4.5. {
```
int[ ][ ] masivi = new int[3][ ];
masivi[0] = new int[4];
masivi[1] = new int[1];
masivi[2] = new int[2];
int raodenoba = masivi.Length;
```
}
ამ კოდის შესრულების შედეგად raodenoba ცვლადში ჩაიწერება:
ა. შიდა მასივების რაოდენობა
ბ. სვეტების რაოდენობა
გ. პირველ სტრიქონში ელემენტების რაოდენობა

4.4.6. {
```
int[ ][ ] masivi = new int[3][ ];
masivi[0] = new int[4];
masivi[1] = new int[1];
masivi[2] = new int[2];
int raodenoba = masivi[0].Length;
```
}
ამ კოდის შესრულების შედეგად raodenoba ცვლადში ჩაიწერება:
ა. masivi მასივის სვეტების რაოდენობა
ბ. masivi მასივის სტრიქონების რაოდენობა
გ. masivi მასივის პირველი სტრიქონის ელემენტების რაოდენობა

4.4.7. {
```
int[ ][ ] masivi = new int[3][ ];
masivi[0] = new int[4];
masivi[1] = new int[1];
masivi[2] = new int[2];
int raodenoba = masivi[0].Length;
```
}
ამ კოდის შესრულების შედეგად raodenoba ცვლადში ჩაიწერება:
ა. 4
ბ. 7
გ. 3

foreach ოპერატორი

4.5.1. როგორია foreach ოპერატორის სინტაქსი:
ა. foreach (ტიპი ცვლადის_სახელი in კოლექცია) { ციკლის ტანი }
ბ. foreach (ციკლის ტანი ცვლადის_სახელი in კოლექცია) { ტიპი }
გ. foreach (ტიპი ციკლის ტანი) { ცვლადის_სახელი in კოლექცია }

4.5.2. foreach ოპერატორში იტერაციული ცვლადის ტიპი:
ა. ხანდახან უნდა ემთხვეოდეს კოლექციის ტიპს
ბ. არ უნდა ემთხვეოდეს კოლექციის ტიპს
გ. უნდა ემთხვეოდეს კოლექციის ტიპს

4.5.3. foreach ოპერატორში იტერაციულ ცვლადს:
ა. კოლექციის მნიშვნელობები ენიჭება უკუმიმდევრობით
ბ. მიმდევრობით ენიჭება კოლექციის მნიშვნელობები დაწყებული პირველი ელემენტიდან
გ. არ ენიჭება კოლექციის მნიშვნელობები

4.5.4. foreach ციკლში:
ა. დაუშვებელია იტერაციული ცვლადის შეცვლა
ბ. დასაშვებია იტერაციული ცვლადის შეცვლა
გ. იტერაციულ ცვლადს ენიჭება მხოლოდ მუდმივა

ბიტობრივი ოპერატორები

4.6.1. ბიტობრივი ოპერატორების ოპერანდებია:
ა. სტრიქონები
ბ. მთელი რიცხვები
გ. წილადი რიცხვები

4.6.2. როგორ ოპერაციებს ეწოდება ბიტობრივი:
ა. ბიტობრივი ეწოდება ისეთ ოპერაციებს, რომლებიც გამოიყენება მთელი რიცხვების ბიტების შესამოწმებლად, დასაყენებლად და ძვრისათვის
ბ. ბიტობრივი ეწოდება ისეთ ოპერაციებს, რომლებიც გამოიყენება მთელი რიცხვების შესაკრებად
გ. ბიტობრივი ეწოდება ისეთ ოპერაციებს, რომლებიც გამოიყენება მთელი რიცხვების გამოსაკლებად

4.6.3. რომელია ბიტობრივი ოპერატორები:
ა. +, -, *
ბ. &, |, ^, ~
გ. &, |, %, /

4.6.4. {
```
int b1 = 10, b2 = 12, shedegi;
shedegi = b1 & b2;
```
}
ამ კოდის შესრულების შედეგად shedegi ცვლადს მიენიჭება:
ა. -2
ბ. 22
გ. 8

4.6.5. თუ & ბიტობრივი ოპერატორის ორივე ოპერანდის მნიშვნელობაა 1, მაშინ შედეგი იქნება:
ა. -1

ბ. 0
გ. 1

4.6.6. თუ | ბიტობრივი ოპერატორის ორივე ოპერანდის მნიშვნელობაა 1, მაშინ შედეგი იქნება:
ა. 0
ბ. 1
გ. -1

4.6.7. თუ & ბიტობრივი ოპერატორის ერთ-ერთი ოპერანდია 0, მაშინ შედეგი იქნება:
ა. -1
ბ. 0
გ. 1

4.6.8. თუ | ბიტობრივი ოპერატორის ერთ-ერთი ოპერანდია 1, მაშინ შედეგი იქნება:
ა. -1
ბ. 0
გ. 1

4.6.9. თუ ~ ბიტობრივი ოპერატორის ოპერანდია 0, მაშინ შედეგი იქნება:
ა. -1
ბ. 0
გ. 1

4.6.10. თუ ~ ბიტობრივი ოპერატორის ოპერანდია 1, მაშინ შედეგი იქნება:
ა. -1
ბ. 0
გ. 1

4.6.11. თუ ^ ოპერატორის ორივე ოპერანდი ერთნაირია, მაშინ შედეგი იქნება:
ა. 0
ბ 1
გ. -1

4.6.12. თუ ^ ოპერატორის ოპერანდები განსხვავებულია, მაშინ შედეგი იქნება:
ა. 0
ბ 1
გ. -1

4.6.13. რიცხვის მარცხნივ ძვრის დროს:
ა. უფროსი თანრიგები იკარგება
ბ. უმცროსი თანრიგები იკარგება
გ. არც ერთი თანრიგი არ იკარგება

4.6.14. რიცხვის მარჯვნივ ძვრის დროს:
ა. უფროსი თანრიგები იკარგება
ბ. უმცროსი თანრიგები იკარგება
გ. არც ერთი თანრიგი არ იკარგება

4.6.15. რიცხვის მარჯვნივ ძვრის დროს:
ა. უმცროსი თანრიგები ნულებით ივსება
ბ. უფროსი თანრიგები ერთებით ივსება
გ. უფროსი თანრიგები ნულებით ივსება

4.6.16. რიცხვის მარცხნივ ძვრის დროს:
ა. უმცროსი თანრიგები ნულებით ივსება
ბ. უმცროსი თანრიგები ერთებით ივსება
გ. უფროსი თანრიგები ნულებით ივსება

4.6.17. ნიშნიანი რიცხვის მარჯვნივ ძვრის დროს:
ა. ნიშნის თანრიგი ინახება
ბ. ნიშნის თანრიგი ერთით ივსება
გ. ნიშნის თანრიგი ნულით ივსება

ჩამოთვლები

4.7.1. ჩამოთვლადი ტიპის განსაზღვრისათვის გამოიყენება:
ა. enum სიტყვა
ბ. public სიტყვა
გ. static სიტყვა

4.7.2. რომელი მსჯელობაა სწორი:
ა. ჩამოთვლები წარმოადგენს სახელდებული წილადი მუდმივების ნაკრებს, რომლებიც განსაზღვრავენ ცვლადის ყველა დასაშვებ მნიშვნელობას მოცემული ტიპისათვის
ბ. ჩამოთვლები წარმოადგენს სახელდებული მთელრიცხვა მუდმივების ნაკრებს, რომლებიც განსაზღვრავენ ცვლადის ყველა დასაშვებ მნიშვნელობას მოცემული ტიპისათვის
გ. ჩამოთვლები წარმოადგენს სახელდებული სტრიქონების მუდმივების ნაკრებს, რომლებიც განსაზღვრავენ ცვლადის ყველა დასაშვებ მნიშვნელობას მოცემული ტიპისათვის

4.7.3. ჩამოთვლაში თითოეულ მუდმივას შეესაბამება:
ა. სტრიქონული მნიშვნელობა
ბ. წილადი მნიშვნელობა
გ. მთელრიცხვა მნიშვნელობა

4.7.4. ჩამოთვლაში მთელრიცხვა მნიშვნელობა თითოეული მუდმივასათვის:
ა. მეტია წინა მუდმივას მნიშვნელობაზე
ბ. ნაკლებია წინა მუდმივას მნიშვნელობაზე
გ. ტოლია წინა მუდმივას მნიშვნელობის

4.7.5. ჩამოთვლაში მთელრიცხვა მნიშვნელობა თითოეული მუდმივასათვის:
ა. ტოლია უკანა მუდმივას მნიშვნელობის
ბ. მეტია უკანა მუდმივას მნიშვნელობაზე
გ. ნაკლებია უკანა მუდმივას მნიშვნელობაზე

4.7.6. ჩამოთვლაში პირველი მუდმივას მნიშვნელობა არის:
ა. -1
ბ. 0
გ. 1

4.7.7. ჩამოთვლის ელემენტებთან მიმართვისათვის:
ა. უნდა მივუთითოთ ჩამოთვლის სახელი, წერტილი და ელემენტის სახელი
ბ. მხოლოდ ჩამოთვლის სახელი
გ. მხოლოდ ჩამოთვლის ელემენტის სახელი

4.7.8. ჩამოთვლებში ინიციალიზატორის საშუალებით:
ა. ნაწილობრივ შეიძლება მივუთითოთ მუდმივას მნიშვნელობა
ბ. არ შეიძლება მივუთითოთ მუდმივას მნიშვნელობა
გ. შეიძლება მივუთითოთ მუდმივას მნიშვნელობა

4.7.9. ჩამოთვლებში ინიციალიზატორის მიერ განსაზღვრული მნიშვნელობა:
ა. ნაკლები უნდა იყოს წინა მუდმივას მნიშვნელობაზე
ბ. მეტი უნდა იყოს წინა მუდმივას მნიშვნელობაზე
გ. ტოლი უნდა იყოს წინა მუდმივას მნიშვნელობის

4.7.10. ჩამოთვლებში ინიციალიზატორის მიერ განსაზღვრული მნიშვნელობა:
ა. ნაკლები უნდა იყოს უკანა მუდმივას მნიშვნელობაზე
ბ. მეტი უნდა იყოს უკანა მუდმივას მნიშვნელობაზე
გ. ტოლი უნდა იყოს უკანა მუდმივას მნიშვნელობის

თავი 5. კლასები, ინკაფსულაცია, მეთოდები

კლასების გამოცხადება

5.1.1. რა არის კლასი:
ა. კლასი არის ობიექტების მასივი
ბ. კლასი არის ობიექტზე ორიენტირებული პროგრამირების ძირითადი სტრუქტურა, რომელიც გამოიყენება მასივების შესაქმნელად
გ. კლასი არის ობიექტზე ორიენტირებული პროგრამირების ძირითადი სტრუქტურა, რომელიც გამოიყენება ობიექტების შესაქმნელად

5.1.2. ობიექტზე ორიენტირებული პროგრამირების პრინციპებია:
ა. ინკაფსულაცია, პოლიმორფიზმი, მემკვიდრეობითობა
ბ. მხოლოდ ინკაფსულაცია
გ. მხოლოდ პოლიმორფიზმი

5.1.3. ინკაფსულაცია არის:
ა. კლასი, რომელიც ობიექტს აღწერს
ბ. დაპროგრამების მექანიზმი, რომელიც აერთიანებს პროგრამის კოდსა და მონაცემებს, აგრეთვე, გამიჯნავს მათ სხვა პროგრამების მხრიდან მიმართვებისაგან
გ. ობიექტი, რომელიც წარმოადგენს კლასის ეგზემპლარს

5.1.4. ობიექტი არის:
ა. მასივი
ბ. მუდმივა
გ. კლასის კონკრეტული რეალიზება კომპიუტერის მეხსიერებაში

5.1.5. ობიექტი წარმოადგენს:
ა. სტრიქონების მასივს
ბ. კლასის ეგზემპლარს
გ. სიმბოლოების მასივს

5.1.6. ინკაფსულაცია არის დაპროგრამების მექანიზმი, რომელიც:
ა. მუშაობს მხოლოდ მთელ რიცხვებთან
ბ. კლასის წევრებს იცავს არასწორი გამოყენებისაგან
გ. მუშაობს მხოლოდ სტრიქონებთან

5.1.7. დახურული ცვლადი აღიწერება სიტყვით:
ა. private
ბ. public
გ. static

5.1.8. ღია ცვლადი აღიწერება სიტყვით:
ა. public
ბ. private
გ. static

5.1.9. რომელი მოდიფიკატორები გამოიყენება კლასის წევრებთან მიმართვისათვის:
ა. params და static
ბ. out და ref
გ. public, private და protected

5.1.10. როგორია private მოდიფიკატორის დანიშნულება:
ა. იგი მიუთითებს, რომ კლასის წევრებთან მიმართვა შეუძლია მხოლოდ ამავე კლასის წევრებს
ბ. იგი მიუთითებს, რომ კლასის წევრებთან მიმართვა შეუძლია მხოლოდ სხვა კლასის წევრებს
გ. იგი მიუთითებს, რომ კლასის წევრებთან მიმართვა არ შეუძლია არც ერთი კლასის წევრებს

5.1.11. როგორია public მოდიფიკატორის დანიშნულება:
ა. იგი მიუთითებს, რომ კლასის წევრებთან მიმართვა შეუძლია მხოლოდ ამავე კლასის წევრებს
ბ. იგი მიუთითებს, რომ კლასის წევრებთან მიმართვა შეუძლია სხვა კლასის წევრებს
გ. იგი მიუთითებს, რომ კლასის წევრებთან მიმართვა არ შეუძლია არც ერთი კლასის წევრებს

5.1.12. კლასის პრივატულ წევრთან მიმართვა შეუძლია:
ა. სხვა კლასში განსაზღვრულ მეთოდებს
ბ. ამავე კლასში განსაზღვრულ მეთოდებს
გ. არც ერთ კლასში განსაზღვრულ მეთოდებს

5.1.13. კლასის საერთო წვდომის წევრთან მიმართვა შეუძლია:
ა. ნებისმიერ კლასში განსაზღვრულ მეთოდებს
ბ. ამავე კლასში განსაზღვრულ მეთოდებს
გ. არც ერთ კლასში განსაზღვრულ მეთოდებს

5.1.14. კლასი გამოიყენება:
ა. სტრიქონების აღსაწერად
ბ. ლოგიკური მონაცემების შესაქმნელად
გ. ობიექტების აღსაწერად და შესაქმნელად

5.1.15. კლასში შემავალ მეთოდებსა და ცვლადებს:
ა. მთელრიცხვა მასივი ეწოდება
ბ. სტრიქონები ეწოდება
გ. კლასის წევრები ეწოდება

5.1.16. კლასის გამოცხადება იწყება სიტყვით:
ა. static
ბ. private
გ. class

5.1.17. იქმნება თუ არა ობიექტი class სიტყვის გამოყენებით კლასის გამოცხადებისას:
ა. კი
ბ. არა
გ. ხანდახან

ინკაფსულაცია

5.2.1. კლასი ორ ძირითად ფუნქციას ასრულებს:
ა. მონაცემებს აკავშირებს კოდთან და განსაზღვრავს მხოლოდ სტრიქონებს
ბ. მონაცემებს აკავშირებს კოდთან და უზრუნველყოფს კლასის წევრებთან მიმართვის საშუალებებს
გ. უზრუნველყოფს მხოლოდ კლასის წევრებთან მიმართვის საშუალებებს

5.2.2. კლასის წევრებთან მიმართვა სრულდება შემდეგი მოდიფიკატორის გამოყენებით:
ა. private, public string
ბ. public, private, protected, internal
გ. protected, static, public

5.2.3. თუ კლასის წევრთან მიმართვის მოდიფიკატორი არ არის მითითებული, მაშინ იგულისხმება:
ა. private
ბ. public
გ. protected

5.2.4. class სიტყვის გამოყენებით კლასის გამოცხადებისას ხდება:
ა. სტრიქონის შექმნა
ბ. ობიექტის შექმნა
გ. კლასის აღწერა

5.2.5. ობიექტის შესაქმნელად გამოიყენება:
ა. class სიტყვა
ბ. new სიტყვა
გ. protected მოდიფიკატორი

5.2.6. ობიექტის ცვლადებთან მიმართვისათვის გამოიყენება:
ა. private მოდიფიკატორი
ბ. new ოპერატორი
გ. . (წერტილი) ოპერატორი

5.2.7. ობიექტის მეთოდთან მიმართვისათვის გამოიყენება:
ა. . (წერტილი) ოპერატორი
ბ. protected მოდიფიკატორი
გ. private მოდიფიკატორი

5.2.8. რომელი ოპერატორი გამოიყენება ობიექტის ცვლადებთან მიმართვისათვის:
ა. წერტილი (.)
ბ. გამრავლება (*)
გ. შეკრება (+)

5.2.9. შეიძლება თუ არა წერტილის გამოყენება ობიექტის მეთოდებთან მიმართვისათვის:

ა. არა

ბ. კი

გ. ხანდახან

5.2.10. რა არის მიმართვა:

ა. მიმართვა არის მეხსიერების იმ უბნის მისამართი, რომელიც ობიექტს არ გამოეყო new ოპერატორის მიერ

ბ. მიმართვა არ არის მეხსიერების იმ უბნის მისამართი, რომელიც ობიექტს გამოეყო new ოპერატორის მიერ

გ. მიმართვა არის მეხსიერების იმ უბნის მისამართი, რომელიც ობიექტს გამოეყო new ოპერატორის მიერ

5.2.11. თუ ორ ობიექტს ერთნაირი ტიპი აქვს, მაშინ მათი ცვლადები:

ა. ერთმანეთზე დამოკიდებულია

ბ. ერთმანეთისაგან დამოუკიდებელია

გ. მხოლოდ int ტიპისაა

5.2.12. ChemiKlasi obieqti = new ChemiKlasi();
ოპერატორი როგორ შეიძლება ჩავწეროთ ორი ოპერატორის სახით:

ა. ChemiKlasi obieqti;
obieqti = new ChemiKlasi();

ბ. obieqti = new ChemiKlasi();
ChemiKlasi obieqti;

გ. ChemiKlasi obieqti;
ChemiKlasi = new ChemiKlasi();

5.2.13. რა მოქმედებები შესრულდება
ChemiKlasi obieqti = new ChemiKlasi();
ოპერატორის შესრულების დროს:

ა. შესრულდება ერთი მოქმედება - ხდება ChemiKlasi ტიპის obieqti მიმართვითი ცვლადის გამოცხადება

ბ. შესრულდება ორი მოქმედება: 1. ხდება ChemiKlasi ტიპის obieqti მიმართვითი ცვლადის გამოცხადება; 2. new ოპერატორი ქმნის ობიექტს. მასზე მიმართვა მიენიჭება obieqti ცვლადს

გ. შესრულდება ერთი მოქმედება - new ოპერატორი ქმნის ობიექტს. მასზე მიმართვა მიენიჭება obieqti ცვლადს

5.2.14. მიმართვითი ტიპის ცვლადი ინახავს:

ა. ცვლადის მისამართს

ბ. ცვლადის მნიშვნელობას

გ. ობიექტის მისამართს

5.2.15. ჩვეულებრივი ტიპის ცვლადი ინახავს:

ა. ობიექტის მისამართს

ბ. ცვლადის მნიშვნელობას

გ. ცვლადის მისამართს

5.2.16. როგორ სრულდება ობიექტთან მიმართვა:
ა. მიმართვის მიხედვით
ბ. მნიშვნელობის მიხედვით
გ. მასივის საშუალებით

5.2.17. როგორ ტიპებს ეკუთვნის კლასები:
ა. ჩვეულებრივს
ბ. მიმართვითს
გ. ლოგიკურს

5.2.18. კლასები გამოიყენება:
ა. მიმართვითი ტიპის მქონე ობიექტების გამოცხადებისათვის
ბ. ჩვეულებრივი ტიპის მქონე ცვლადების გამოცხადებისათვის
გ. მთელრიცხვა მასივების გამოცხადებისათვის

5.2.19. {
```
klasi1 obieqti1 = new klasi1();
klasi2 obieqti2 = new klasi2();
obieqti1 = obieqti2;
```
}
ამ კოდის შესრულების შედეგად:
ა. obieqti1 ცვლადს ენიჭება obieqti2 კლასზე მიმართვა
ბ. obieqti2 ცვლადს ენიჭება იმ ობიექტზე მიმართვა, რომელსაც obieqti1 ცვლადი მიმართავს
გ. obieqti1 ცვლადს ენიჭება იმ ობიექტზე მიმართვა, რომელსაც obieqti2 ცვლადი მიმართავს

მეთოდები

5.3.1. მეთოდი არის:
ა. პროგრამა
ბ. სტრიქონი
გ. კლასის ობიექტი

5.3.2. ერთი მეთოდის გამოძახება შეიძლება:
ა. ერთხელ
ბ. ბევრჯერ
გ. ორჯერ

5.3.3. შეიძლება თუ არა გამოვიყენოთ დარეზერვებული სიტყვები მეთოდების სახელებად:
ა. ზოგჯერ
ბ. კი
გ. არა

5.3.4. რა შემთხვევაში მიეთითება მეთოდისათვის void ტიპი:
ა. როცა მეთოდი მნიშვნელობას გასცემს
ბ. როცა მეთოდი მნიშვნელობას არ გასცემს
გ. როცა მეთოდი მთელ რიცხვს გასცემს

5.3.5. როგორია მეთოდის სინტაქსი:

ა. მიმართვის_მოდიფიკატორი ტიპი მეთოდის_სახელი (მეთოდის კოდი)
{ პარამეტრების_სია }

ბ. მიმართვის_მოდიფიკატორი ტიპი მეთოდის_სახელი (პარამეტრების_სია)
{ მეთოდის კოდი }

გ. მეთოდის_სახელი (პარამეტრების_სია) { მეთოდის კოდი }

5.3.6. რომელი მსჯელობაა არასწორი:

ა. მეთოდმა შეიძლება დააბრუნოს მხოლოდ სტრიქონი

ბ. მეთოდმა შეიძლება არ დააბრუნოს მნიშვნელობა

გ. მეთოდმა შეიძლება დააბრუნოს მნიშვნელობა

5.3.7. რომელი მსჯელობაა არასწორი:

ა. მეთოდს შეიძლება ჰქონდეს რამდენიმე პარამეტრი

ბ. მეთოდს შეიძლება არ ჰქონდეს პარამეტრები

გ. მეთოდს შეიძლება ჰქონდეს მხოლოდ მთელი ტიპის პარამეტრები

5.3.8. void სიტყვა მიუთითებს იმას, რომ მეთოდი:

ა. აბრუნებს მთელ რიცხვს

ბ. აბრუნებს წილადს

გ. არ აბრუნებს შედეგს

5.3.9. როცა მეთოდი მუშაობას ამთავრებს, მაშინ მართვა გადაეცემა გამომძახებელ პროგრამის:

ა. იმ სტრიქონს, საიდანაც მოხდა მეთოდის გამოძახება

ბ. დასაწყისს

გ. დასასრულს

5.3.10. როცა მეთოდი მუშაობას ამთავრებს, მაშინ გამომძახებელი პროგრამის შესრულება გაგრძელდება:

ა. პროგრამის დასაწყისიდან

ბ. იმ სტრიქონიდან, რომელიც მოსდევს მეთოდის გამომძახებელ სტრიქონს

გ. პროგრამის დასასრულიდან

5.3.11. მეთოდიდან მართვის დაბრუნება ხდება მაშინ, როცა გვხვდება:

ა. static სიტყვა

ბ. მეთოდის დამხურავი ფიგურული ფრჩხილი ან return ოპერატორი

გ. public მოდიფიკატორი

5.3.12. რომელი მსჯელობაა მცდარი:

ა. როცა მეთოდი მნიშვნელობას არ აბრუნებს, მაშინ მიეთითება return 7;

ბ. როცა მეთოდი მნიშვნელობას არ აბრუნებს, მაშინ მიეთითება return ოპერატორი მნიშვნელობის გარეშე

გ. როცა მეთოდი მნიშვნელობას არ აბრუნებს, მაშინ არ მიეთითება return ოპერატორი

5.3.13. მეთოდისათვის გადასაცემ მნიშვნელობას ეწოდება:

ა. წილადი

ბ. პარამეტრი
გ. არგუმენტი

5.3.14. ცვლადს მეთოდის შიგნით, რომელიც იღებს არგუმენტის მნიშვნელობას ეწოდება:
ა. სიმბოლო
ბ. არგუმენტი
გ. პარამეტრი

5.3.15. პარამეტრი მოქმედებს:
ა. მხოლოდ თავისი მეთოდის ხილვადობის უბანში
ბ. სხვა მეთოდის ხილვადობის უბანში
გ. მთელ პროგრამაში

5.3.16. თუ მეთოდი არ აბრუნებს შედეგს, მაშინ რომელი ტიპი უნდა მივუთითოთ:
ა. int
ბ. void
გ. bool

5.3.17. შეიძლება თუ არა მეთოდის გამოცხადება პარამეტრების გარეშე:
ა. ნაწილობრივ
ბ. არა
გ. კი

5.3.18. რომელი მსჯელობაა სწორი:
ა. მეთოდიდან მართვის დაბრუნება ხდება მაშინ, როცა არ გვხვდება მეთოდის დამხურავი ფიგურული ფრჩხილი ან return ოპერატორი
ბ. მეთოდიდან მართვის დაბრუნება ხდება მაშინ, როცა გვხვდება მეთოდის დამხურავი ფიგურული ფრჩხილი ან return ოპერატორი
გ. მეთოდიდან მართვის დაბრუნება ხდება მაშინ, როცა გვხვდება მეთოდის გამხსნელი ფიგურული ფრჩხილი ან continue ოპერატორი

5.3.19. return ოპერატორის რა ფორმები არსებობს:
ა. არსებობს ორი ფორმა: პირველი ფორმა გამოიყენება მეთოდებში, რომლებიც მნიშვნელობებს არ აბრუნებენ, მეორე კი მეთოდებში, რომლებიც მნიშვნელობებს აბრუნებენ
ბ. არსებობს ერთი ფორმა, რომელიც გამოიყენება მეთოდებში, რომლებიც მნიშვნელობებს არ აბრუნებენ
გ. არსებობს ერთი ფორმა, რომელიც გამოიყენება მეთოდებში, რომლებიც მნიშვნელობებს აბრუნებენ

5.3.20. შესრულდება თუ არა return ოპერატორის შემდეგ მოთავსებული ოპერატორები:
ა. კი
ბ. არა
გ. ხანდახან

5.3.21. იმყოფება თუ არა პარამეტრი თავისი მეთოდის ხილვადობის უბანში:
ა. კი

ბ. არა
გ. ზოგჯერ

კონსტრუქტორი

5.4.1. კონსტრუქტორი არის მეთოდი, რომლის დანიშნულებაა:
ა. ობიექტის ცვლადების ინიციალიზება
ბ. მასივის შევსება რიცხვებით
გ. სტრიქონების ფორმირება

5.4.2. რა არის კონსტრუქტორი:
ა. კონსტრუქტორი არის მეთოდი, რომელიც ახდენს ობიექტის ინიციალიზებას მისი შექმნისას. მისი სახელი ემთხვევა კლასის სახელს
ბ. კონსტრუქტორი არის მეთოდი, რომელიც ახდენს ობიექტის ინიციალიზებას მისი შექმნისას. მისი სახელი არ ემთხვევა კლასის სახელს
გ. კონსტრუქტორი არის მეთოდი, რომელიც არ ახდენს ობიექტის ინიციალიზებას მისი შექმნისას. მისი სახელი არ ემთხვევა კლასის სახელს

5.4.3. როგორია კონსტრუქტორის სინტაქსი:
ა. კლასის_სახელი {კონსტრუქტორის კოდი}
ბ. კონსტრუქტორის კოდი () { კლასის_სახელი }
გ. კლასის_სახელი() {კონსტრუქტორის კოდი}

5.4.4. აბრუნებს თუ არა კონსტრუქტორი შედეგს:
ა. არა
ბ. კი
გ. ხანდახან

5.4.5. აქვს თუ არა ყველა კლასს კონსტრუქტორი:
ა. არა
ბ. კი
გ. ხანდახან

5.4.6. რისთვის გამოიყენება კონსტრუქტორი:
ა. ობიექტის ცვლადების შესაცვლელად
ბ. ობიექტის ცვლადების წასაშლელად
გ. ობიექტის ცვლადებისათვის საწყისი მნიშვნელობების მისანიჭებლად

5.4.7. როდის გამოიძახება ავტომატური კონსტრუქტორი:
ა. როცა კლასში კონსტრუქტორი არ არის განსაზღვრული
ბ. როცა კლასში კონსტრუქტორი არის განსაზღვრული
გ. როცა კლასში მეთოდები არ არის განსაზღვრული

5.4.8. კონსტრუქტორის მიერ ობიექტის ინიციალიზება სრულდება:
ა. მეხსიერებიდან ობიექტის წაშლისას

ბ. ობიექტის შექმნისას
გ. ობიექტის შეცვლისას სხვა ობიექტით

5.4.9. კონსტრუქტორის სახელი ემთხვევა:
ა. ცვლადის სახელს
ბ. მასივის სახელს
გ. მისი კლასის სახელს

5.4.10. კონსტრუქტორში დასაბრუნებელი მნიშვნელობის ტიპი:
ა. არ მიეთითება
ბ. მიეთითება
გ. მიეთითება ლოგიკური ტიპი

5.4.11. ყველა კლასს:
ა. აქვს კონსტრუქტორი იმ შემთხვევაში, თუ ის განსაზღვრულია კლასში
ბ. არ აქვს კონსტრუქიტორი
გ. აქვს კონსტრუქტორი

5.4.12. ავტომატური კონსტრუქტორი:
ა. ყველა ცვლადს ერთეულოვან მნიშვნელობას ანიჭებს
ბ. ყველა ცვლადს ნულოვან მნიშვნელობას ანიჭებს
გ. ცვლადებს არანაირ მნიშვნელობას არ ანიჭებს

5.4.13. ავტომატური კონსტრუქტორი იმ შემთხვევაში გამოიძახება, როცა:
ა. კლასში კონსტრუქტორი არის განსაზღვრული
ბ. კლასში კონსტრუქტორი არ არის განსაზღვრული
გ. კლასში ორი კონსტრუქტორია განსაზღვრული

5.4.14. თუ კლასში კონსტრუქტორი აშკარადაა განსაზღვრული, მაშინ შესრულდება:
ა. ორივე კონსტრუქტორის გამოძახება
ბ. ავტომატური კონსტრუქტორის გამოძახება
გ. ამ კონსტრუქტორის გამოძახება

დესტრუქტორი

5.5.1. "ნაგვის შეგროვება" არის ოპერაცია, როცა ხდება:
ა. ობიექტის მიერ დაკავებული მეხსიერების გათავისუფლება
ბ. მეხსიერების დაკავება ობიექტის მიერ
გ. მეხსიერების დაკავება სტრიქონის მიერ

5.5.2. "ნაგვის შეგროვება" სრულდება:
ა. პროგრამის მუშაობის დასაწყისში
ბ. პროგრამის მუშაობის დროს
გ. მაშინ, როცა არ არსებობს მეხსიერებიდან წასაშლელი ობიექტები

5.5.3. დესტრუქტორი არის მეთოდი, რომელიც გამოიყენება:
ა. ობიექტის ცვლადების ინიციალიზებისათვის

ბ. მეხსიერებაში ობიექტის მოსათავსებლად
გ. მეხსიერებიდან ობიექტის წასაშლელად

5.5.4. ~ (ტილდა) სიმბოლო მიეთითება:
ა. დესტრუქტორის სახელის წინ
ბ. კონსტრუქტორის სახელის წინ
გ. კონსტრუქტორის სახელის შემდეგ

5.5.5. როგორია დესტრუქტორის სინტაქსი:
ა. ~კლასის_სახელი {დესტრუქტორის კოდი}
ბ. ~კლასის_სახელი() {დესტრუქტორის კოდი}
გ. ~ (დესტრუქტორის კოდი) { კლასის_სახელი }

5.5.6. აბრუნებს თუ არა დესტრუქტორი შედეგს:
ა. კი
ბ. არა
გ. ხანდახან

5.5.7. როდის გამოიძახება დესტრუქტორი:
ა. "ნაგვის შეგროვების" შემდეგ
ბ. "ნაგვის შეგროვების" შუაში
გ. უშუალოდ "ნაგვის შეგროვების" წინ

this სიტყვა

5.6.1. მეთოდს გამოძახებისას გადაეცემა:
ა. ობიექტის სახელი
ბ. გამომძახებელ ობიექტზე მიმართვა
გ. ლოგიკური მნიშვნელობა

5.6.2. this მიმართვა მიუთითებს:
ა. ობიექტის ცვლადზე
ბ. გამომძახებელ ობიექტზე
გ. სტრიქონზე

5.6.3. როცა პარამეტრის სახელი ემთხვევა ლოგიკური ცვლადის სახელს, მაშინ:
ა. ლოკალური ცვლადი მალავს ლოგიკურ ცვლადს
ბ. ლოკალური ცვლადი მალავს პარამეტრს
გ. პარამეტრი მალავს ლოკალურ ცვლადს

5.6.4. როცა პარამეტრი მალავს ლოკალურ ცვლადს, მაშინ:
ა. ლოკალურ ცვლადთან მიმართვა შესაძლებელია this მიმართვის გარეშე
ბ. პარამეტრთან მიმართვა შესაძლებელია this მიმართვის გამოყენებით
გ. ლოკალურ ცვლადთან მიმართვა შესაძლებელია this მიმართვის გამოყენებით

5.6.5. this არის:
ა. მიმართვა

ბ. მასივი
გ. ცვლადი

5.6.6. მეთოდში ოპერატორის ჩაწერა this სიტყვის გარეშე არის:
ა. ჩაწერის შემოკლებული ფორმა
ბ. ჩაწერის განვრცობილი ფორმა
გ. ობიექტის გამოცხადების სახესხვაობა

5.6.7. C# ენის სინტაქსი:
ა. არ იძლევა ერთნაირი სახელების გამოყენების საშუალებას პარამეტრებისა და ლოკალური ცვლადებისათვის
ბ. იძლევა ერთნაირი სახელების გამოყენების საშუალებას პარამეტრებისა და ლოკალური ცვლადებისათვის
გ. იძლევა ერთნაირი სახელების გამოყენების საშუალებას ორი სხვადასხვა განზომილების მქონე მასივისათვის

5.6.8. this.xarisxi = xarisxi; მინიჭების ოპერატორში:
ა. this.xarisxi არის მეთოდის პარამეტრი
ბ. this.xarisxi არ არის კლასში განსაზღვრული ცვლადი
გ. this.xarisxi არის კლასში განსაზღვრული ცვლადი

5.6.9. this.xarisxi = xarisxi; მინიჭების ოპერატორში:
ა. xarisxi არის მეთოდის პარამეტრი
ბ. this.xarisxi არის მეთოდის პარამეტრი
გ. xarisxi არ არის მეთოდის პარამეტრი

რთული კლასები

5.7.1. ერთი ობიექტი მეორე ობიექტს შეიძლება:
ა. მოიცავდეს
ბ. არ მოიცავდეს

5.7.2. კლასის ერთ წევრს:
ა. არ შეიძლება ჰქონდეს მეორე კლასის ტიპი
ბ. შეიძლება ჰქონდეს მეორე კლასის ტიპი

5.7.3. კლასის წევრის გამოცხადება შეიძლება:
ა. არ შეიძლება სხვა კლასის გამოყენებით
ბ. შეიძლება სხვა კლასის გამოყენებით

5.7.4. რთული ეწოდება კლასს, რომლის:
ა. წევრს აქვს სხვა კლასის ტიპი
ბ. წევრს არ აქვს სხვა კლასის ტიპი

ჩადგმული კლასები

5.8.1. ერთი კლასის შიგნით:
ა. დაუშვებელია მეორე კლასის გამოცხადება
ბ. დასაშვებია მეორე კლასის გამოცხადება

5.8.2. ჩადგმული შეიძლება იყოს:
ა. რამდენიმე კლასი
ბ. მხოლოდ ერთი კლასი
გ. მხოლოდ ორი კლასი

5.8.3. ჩადგმულ კლასს ეწოდება:
ა. პრივატული კლასი
ბ. გარე კლასი
გ. შიდა კლასი

5.8.4. ჩადგმული კლასის შემცველ კლასს ეწოდება:
ა. დაცული კლასი
ბ. შიდა კლასი
გ. გარე კლასი

5.8.5. შიდა კლასი შეიძლება იყოს:
ა. როგორ პრივატული, ისე ღია
ბ. მხოლოდ პრივატული
გ. მხოლოდ ღია

სტრუქტურები

5.9.1. სტრუქტურა არის კლასის:
ა. გართულებული ვარიანტი
ბ. შედგენილი ვარიანტი
გ. გამარტივებული ვარიანტი

5.9.2. სტრუქტურებთან მუშაობა სრულდება ისე, როგორც:
ა. მნიშვნელობის მქონე ტიპებთან
ბ. კლასებთან
გ. მასივის ტიპებთან

5.9.3. სტრუქტურის ეგზემპლარი ინახება:
ა. ორგანზომილებიან მასივში
ბ. გროვაში
გ. სტეკში

5.9.4. სტრუქტურის ეგზემპლარი მეხსიერებიდან იშლება მაშინ, როცა ეს ეგზემპლარი:
ა. რჩება ხილვადობის უბანში
ბ. გამოდის ხილვადობის უბნიდან
გ. იქმნება

5.9.5. სტრუქტურა უმჯობესია იყოს:
ა. დიდი ზომის
ბ. მცირე ზომის
გ. საშუალო ზომის

5.9.6. სტრუქტურის გამოყენება უმჯობესია მაშინ, როცა გვაქვს:
ა. მცირე რაოდენობის ცვლადები და მეთოდები
ბ. დიდი რაოდენობის ცვლადები და მეთოდები

5.9.7. სტრუქტურა შეიძლება შეიცავდეს:
ა. მეთოდებს, დესტრუქტორებსა და ცვლადებს
ბ. დესტრუქტორებს, კონსტრუქტორებსა და თვისებებს
გ. მოვლენებს, კონსტრუქტორებსა და ინდექსატორებს

5.9.8. სტრუქტურის გამოცხადებისათვის გამოიყენება:
ა. this საკვანძო სიტყვა
ბ. struct საკვანძო სიტყვა
გ. static საკვანძო სიტყვა

5.9.9. კლასისაგან განსხვავებით სტრუქტურა:
ა. უზრუნველყოფს მემკვიდრეობითობას
ბ. არ უზრუნველყოფს მემკვიდრეობითობას

5.9.10. კლასისაგან განსხვავებით სტრუქტურისათვის:
ა. არ შეიძლება ავტომატური კონსტრუქტორის განსაზღვრა
ბ. შეიძლება ავტომატური კონსტრუქტორის განსაზღვრა
გ. შეიძლება დესტრუქტორის განსაზღვრა

5.9.11. კლასისაგან განსხვავებით სტრუქტურისათვის:
ა. შეიძლება დესტრუქტორის განსაზღვრა
ბ. არ შეიძლება დესტრუქტორის განსაზღვრა
გ. შეიძლება ავტომატური კონსტრუქტორის განსაზღვრა

5.9.12. კლასისაგან განსხვავებით სტრუქტურისათვის:
ა. არ შეიძლება ინიციალიზატორის გამოყენება ცვლადებისათვის მნიშვნელობების მისანიჭებლად
ბ. შეიძლება ინიციალიზატორის გამოყენება ცვლადებისათვის მნიშვნელობების მისანიჭებლად
გ. შეიძლება ავტომატური კონსტრუქტორის განსაზღვრა

შემთხვევითი რიცხვების გენერატორი

5.10.1. შემთხვევით რიცხვებთან სამუშაოდ გამოიყენება:
ა. System.Random კლასი
ბ. System.IO კლასი
გ. System.Data კლასი

5.10.2. int Random.Next(int ცვლადი) მეთოდი:
ა. გასცემს უარყოფით შემთხვევით მთელ რიცხვს, რომელიც ნაკლებია მითითებული ცვლადის მნიშვნელობაზე
ბ. გასცემს არაუარყოფით შემთხვევით მთელ რიცხვს, რომელიც ნაკლებია მითითებული ცვლადის მნიშვნელობაზე
გ. გასცემს არაუარყოფით შემთხვევით წილად რიცხვს, რომელიც ნაკლებია მითითებული ცვლადის მნიშვნელობაზე

5.10.3. double Random.NextDouble() მეთოდი:
ა. გასცემს შემთხვევით წილად რიცხვს, რომლის მნიშვნელობა მოთავსებულია -1.0-სა და 1.0-ს შორის
ბ. გასცემს შემთხვევით მთელ რიცხვს, რომლის მნიშვნელობა მოთავსებულია 0.0-სა და 1.0-ს შორის
გ. გასცემს შემთხვევით წილად რიცხვს, რომლის მნიშვნელობა მოთავსებულია 0.0-სა და 1.0-ს შორის

5.10.4. void Random.NextBytes(byte[ ] მასივი) მეთოდი:
ა. მითითებულ მასივს შეავსებს შემთხვევითი რიცხვებით, რომელთა ტიპია int
ბ. მითითებულ მასივს შეავსებს შემთხვევითი რიცხვებით, რომელთა ტიპია byte
გ. მითითებულ მასივს შეავსებს შემთხვევითი რიცხვებით, რომელთა ტიპია float

5.10.5. ricxvi = obieqti.Next(5); ოპერატორის შესრულების შედეგად ricxvi ცვლადს მიენიჭება:
ა. 5-ზე მეტი მნიშვნელობა
ბ. 5-ზე ნაკლები მნიშვნელობა
გ. 5-ის ტოლი მნიშვნელობა

თავი 6. მეთოდები უფრო დაწვრილებით. პოლიმორფიზმი

მეთოდისთვის ობიექტის გადაცემა

6.1.1. მეთოდს პარამეტრებად შეგვიძლია გადავცეთ:
ა. მხოლოდ ობიექტები
ბ. ჩვეულებრივი ტიპის მნიშვნელობები და ობიექტები
გ. მხოლოდ ჩვეულებრივი ტიპის მნიშვნელობები

6.1.2. obieqti1.Metodi(obieqti2); გამოსახულებაში გამომძახებელი ობიექტია:
ა. obieqti1
ბ. obieqti2
გ. Metodi

6.1.3. რომელ მეთოდს გადაეცემა ჩვენ მიერ შექმნილი Klasi კლასის ტიპის ობიექტი:
ა. public int Metodi3(double obj3);
ბ. public int Metodi2(int par2);
გ public int Metodi1(Klasi par1);

6.1.4. რომელ მეთოდს არ გადაეცემა ჩვენ მიერ შექმნილი Klasi1 კლასის ტიპის ობიექტი:
ა. public int Metodi3(double obj3);
ბ. public int Metodi2(Klasi1 par2);
გ public int Metodi1(Klasi1 obj1);

მეთოდის მიერ ობიექტის დაბრუნება

6.2.1. მეთოდს შეუძლია დააბრუნოს:
ა. როგორც ნებისმიერი ტიპის მონაცემი, ისე კლასის ობიექტიც
ბ. მხოლოდ მთელი და წილადი რიცხვები
გ. მხოლოდ კლასის ობიექტი

6.2.2. ობიექტს აბრუნებს შემდეგი ოპერატორი:
ა. return new Konstruqtori(cvladi1, cvladi2);
ბ. return Konstruqtori(cvladi1, cvladi2);
გ. return cvladi1;

6.2.3. რომელი ოპერატორი შეიცავს შეცდომას:
ა. return cvladi1;
ბ. return new Konstruqtori(cvladi1, cvladi2);
გ. return Konstruqtori(cvladi1, cvladi2);

მეთოდისთვის არგუმენტების გადაცემის ხერხები

6.3.1. მეთოდს არგუმენტები შეგვიძლია გადავცეთ:
ა. მნიშვნელობის მიხედვით ან მიმართვის მიხედვით

ბ. მხოლოდ მნიშვნელობის მიხედვით
გ. მხოლოდ მიმართვის მიხედვით

6.3.2. როცა მეთოდს არგუმენტი გადაეცემა მნიშვნელობის მიხედვით, მაშინ:
ა. არგუმენტის მისამართი ენიჭება პარამეტრს
ბ. არგუმენტის მნიშვნელობის ასლი ენიჭება მეთოდის პარამეტრს
გ. პარამეტრს ენიჭება არგუმენტის მნიშვნელობაც და მისამართიც

6.3.3. როცა მეთოდს არგუმენტი გადაეცემა მიმართვის მიხედვით, მაშინ:
ა. არგუმენტის მნიშვნელობა ენიჭება მეთოდის პარამეტრს
ბ. მეთოდის პარამეტრს გადაეცემა არგუმენტის მისამართი
გ. მეთოდის პარამეტრს გადაეცემა არგუმენტის მნიშვნელობაც და მისამართიც

6.3.4. როცა მეთოდს არგუმენტი გადაეცემა მიმართვის მიხედვით, მაშინ:
ა. პარამეტრის მნიშვნელობის შეცვლისას იცვლება არგუმენტის მისამართი
ბ. პარამეტრის მნიშვნელობის შეცვლისას არ იცვლება არგუმენტის მნიშვნელობა
გ. პარამეტრის მნიშვნელობის შეცვლისას იცვლება არგუმენტის მნიშვნელობა

6.3.5. მეთოდის პარამეტრის მნიშვნელობის შეცვლისას იცვლება არგუმენტის მნიშვნელობა იმიტომ, რომ:
ა. პარამეტრიც და არგუმენტიც მეხსიერების ერთსა და იმავე უბანს მიმართავენ
ბ. პარამეტრიც და არგუმენტიც მეხსიერების სხვადასხვა უბანს მიმართავენ
გ. პარამეტრიც და არგუმენტიც ლოგიკურ მონაცემებს მიმართავენ

6.3.6. მეთოდისთვის ჩვეულებრივი ტიპის მქონე მნიშვნელობის გადაცემისას გამოიყენება:
ა. გადაცემა მნიშვნელობის მიხედვით
ბ. გადაცემა მიმართვის მიხედვით
გ. ლოგიკური მონაცემების გადაცემა

6.3.7. როცა მეთოდს არგუმენტი გადაეცემა მნიშვნელობის მიხედვით, მაშინ:
ა. პარამეტრის მნიშვნელობის შეცვლისას იცვლება არგუმენტის მნიშვნელობა
ბ. პარამეტრის მნიშვნელობის შეცვლისას არ იცვლება არგუმენტის მნიშვნელობა
გ. პარამეტრის მნიშვნელობის შეცვლისას იცვლება მხოლოდ სტრიქონები

ref და out მოდიფიკატორები

ref მოდიფიკატორი

6.4.1. ref მოდიფიკატორის გამოყენება იწვევს:
ა. არგუმენტისათვის პარამეტრის მინიჭებას
ბ. პარამეტრისათვის არგუმენტის მნიშვნელობის მინიჭებას
გ. პარამეტრისათვის არგუმენტის მისამართის მინიჭებას

6.4.2. ref მოდიფიკატორი მიეთითება:
ა. მეთოდის გამოცხადებისას და მეთოდის გამოძახებისას
ბ. მხოლოდ მეთოდის გამოცხადებისას
გ. მხოლოდ მეთოდის გამოძახებისას

out მოდიფიკატორი

6.5.3. out მოდიფიკატორის გამოყენებით შესაძლებელია:
ა. მხოლოდ ერთი მნიშვნელობის დაბრუნება
ბ. რამდენიმე მნიშვნელობის დაბრუნება
გ. მხოლოდ ორი მნიშვნელობის დაბრუნება

6.5.4. out მოდიფიკატორი მიეთითება:
ა. იმ პარამეტრის წინ, რომლის დაბრუნებაც გვინდა
ბ. იმ პარამეტრის წინ, რომლის დაბრუნებაც არ გვინდა
გ. პროგრამის პირველ სტრიქონში გამოცხადებული ცვლადის წინ

6.5.5. out მოდიფიკატორიანი პარამეტრი გამოიყენება:
ა. მეთოდისთვის ინფორმაციის გადასაცემად
ბ. მეთოდიდან ინფორმაციის დასაბრუნებლად
გ. მეთოდისთვის როგორც ინფორმაციის გადასაცემად, ისე მეთოდიდან ინფორმაციის დასაბრუნებლად

6.5.6. მეთოდმა out მოდიფიკატორიან პარამეტრს:
ა. შეიძლება არ მიანიჭოს მნიშვნელობა
ბ. არ უნდა მიანიჭოს მნიშვნელობა
გ. აუცილებლად უნდა მიანიჭოს მნიშვნელობა

params მოდიფიკატორი

6.6.1. მეთოდისთვის ცვლადი რაოდენობის არგუმენტების გადასაცემად უნდა გამოვიყენოთ:
ა. out მოდიფიკატორი
ბ. public მოდიფიკატორი
გ. params მოდიფიკატორი

6.6.2. params მოდიფიკატორი საშუალებას გვაძლევს მეთოდს გადავცეთ:
ა. არც ერთი არგუმენტი
ბ. მხოლოდ ორი არგუმენტი
გ. არგუმენტების ნებისმიერი რაოდენობა

6.6.3. როცა მეთოდში მითითებულია ჩვეულებრივი და ცვლადი სიგრძის პარამეტრები, მაშინ:
ა. ცვლადი სიგრძის პარამეტრი პარამეტრების სიაში უნდა იყოს უკანასკნელი
ბ. ცვლადი სიგრძის პარამეტრი პარამეტრების სიაში უნდა იყოს პირველი
გ. ცვლადი სიგრძის პარამეტრი პარამეტრების სიაში უნდა იყოს რიგით მეორე

მეთოდის გადატვირთვა. პოლიმორფიზმი

6.7.1. როცა კლასში ორ ან მეტ მეთოდს ერთნაირი სახელი აქვს, მაშინ:
ა. ამ მეთოდების პარამეტრების რაოდენობა უნდა იყოს სხვადასხვა ან პარამეტრებს სხვადასხვა ტიპი უნდა ჰქონდეთ

ბ. ეს მეთოდები უნდა განსხვავდებოდეს მხოლოდ პარამეტრების რაოდენობის მიხედვით
გ. ეს მეთოდები უნდა განსხვავდებოდეს მხოლოდ პარამეტრების ტიპის მიხედვით

6.7.2. როცა კლასში რამდენიმე მეთოდს ერთნაირი სახელი აქვს, მაშინ
ა. ისინი არ არის გადატვირთვადი მეთოდები
ბ. მათ გადატვირთვადი ეწოდება
გ. ადგილი აქვს ინკაფსულაციას

6.7.3. კლასში ერთნაირი სახელის მქონე მეთოდების განსაზღვრის პროცესს ეწოდება:
ა. მემკვიდრეობითობა
ბ. ინკაფსულაცია
გ. მეთოდის გადატვირთვა

6.7.4. პოლიმორფიზმი საშუალებას გვაძლევს:
ა. მეთოდის რამდენიმე სახელის ნაცვლად გამოვიყენოთ ერთი
ბ. მეთოდის ერთი სახელის ნაცვლად გამოვიყენოთ რამდენიმე
გ. მეთოდის სახელები მემკვიდრეობით გადავცეთ

6.7.5. პოლიმორფიზმის უზრუნველყოფა ხდება:
ა. ინკაფსულაციის საშუალებით
ბ. მეთოდების გადატვირთვის საშუალებით
გ. ერთი მეთოდისათვის რამდენიმე სახელის დარქმევის საშუალებით

6.7.6. გადატვირთვადი მეთოდის გამოძახებისას სრულდება ის მეთოდი, რომლის:
ა. პარამეტრები წილადი ტიპისაა
ბ. პარამეტრები არ ემთხვევა არგუმენტებს
გ. პარამეტრები ემთხვევა არგუმენტებს

6.7.7. გადატვირთვად მეთოდებში თუ პარამეტრების რაოდენობა თანაბარია, მაშინ
ა. პარამეტრს უნდა ჰქონდეს მთელი ტიპი
ბ. არ უნდა განსხვავდებოდეს მათი ტიპები
გ. უნდა განსხვავდებოდეს მათი ტიპები

6.7.8. მეთოდების გადატვირთვისას:
ა. შეიძლება ref და out მოდიფიკატორების გამოყენება
ბ. არ შეიძლება ref და out მოდიფიკატორების გამოყენება
გ. შეიძლება მხოლოდ out მოდიფიკატორების გამოყენება

6.7.9. თუ გადატვირთვად მეთოდებში პარამეტრების რაოდენობა თანაბარია და არგუმენტებსა და პარამეტრებს განსხვავებული ტიპები აქვს, მაშინ:
ა. არ სრულდება ტიპების ავტომატური გარდაქმნა
ბ. სრულდება ტიპების ავტომატური გარდაქმნა
გ. ყველა მონაცემი გარდაიქმნება მთელი ტიპის მონაცემად

6.7.10. C# ენაში ერთი კლასის შიგნით ორ ან მეტ მეთოდს:
ა. არ შეიძლება ერთნაირი სახელი ჰქონდეს
ბ. აუცილებლად უნდა ჰქონდეს ერთნაირი სახელი
გ. შეიძლება ერთნაირი სახელი ჰქონდეს

კონსტრუქტორების გადატვირთვა

6.8.1. რომელი მსჯელობაა სწორი:
ა. შეიძლება კონსტრუქტორის გადატვირთვა
ბ. არ შეიძლება კონსტრუქტორის გადატვირთვა
გ. კონსტრუქტორის გადატვირთვა ხანდახან შეიძლება

6.8.2. ერთი მეთოდის ინიციალიზებისათვის კონსტრუქტორმა:
ა. შეიძლება გამოიყენოს მხოლოდ ლოგიკური ობიექტები
ბ. არ შეიძლება მეორე ობიექტი გამოიყენოს
გ. შეიძლება მეორე ობიექტი გამოიყენოს

6.8.3. კლასის შიგნით ერთი გადატვირთვადი კონსტრუქტორის მიერ მეორის გამოძახება შესაძლებელია:
ა. public მოდიფიკატორის გამოყენებით
ბ. this სიტყვის გამოყენებით
გ. static მოდიფიკატორის გამოყენებით

6.8.4. გამომძახებელი კონსტრუქტორის შესრულებისას:
ა. ჯერ გამოიძახება მითითებული გადატვირთვადი კონსტრუქტორი, შემდეგ კი შესრულდება გამომძახებელი კონსტრუქტორის დანარჩენი ოპერატორები
ბ. ჯერ შესრულდება გამომძახებელი კონსტრუქტორის ოპერატორები, შემდეგ კი გამოიძახება მითითებული გადატვირთვადი კონსტრუქტორი
გ. არ გამოიძახება გადატვირთვადი კონსტრუქტორი

გადატვირთვადი კონსტრუქტორის გამოძახება this სიტყვის გამოყენებით

6.9.1. კლასის შიგნით ერთი გადატვირთვადი კონსტრუქტორის მიერ მეორის გამოძახება შესაძლებელია:
ა. public მოდიფიკატორის გამოყენებით
ბ. this სიტყვის გამოყენებით
გ. static მოდიფიკატორის გამოყენებით

6.9.2. გამომძახებელი კონსტრუქტორის შესრულებისას:
ა. ჯერ გამოიძახება მითითებული გადატვირთვადი კონსტრუქტორი, შემდეგ კი შესრულდება გამომძახებელი კონსტრუქტორის დანარჩენი ოპერატორები
ბ. ჯერ შესრულდება გამომძახებელი კონსტრუქტორის ოპერატორები, შემდეგ კი გამოიძახება მითითებული გადატვირთვადი კონსტრუქტორი
გ. არ გამოიძახება გადატვირთვადი კონსტრუქტორი

6.9.3. გადატვირთვადი კონსტრუქტორის გამოძახებისას :
ა. შეუძლებელია გამოვიყენოთ ერთი კონსტრუქტორის მიერ მეორის გამოძახება
ბ. აუცილებელია გამოვიყენოთ ერთი კონსტრუქტორის მიერ მეორის გამოძახება
გ. შესაძლებელია გამოვიყენოთ ერთი კონსტრუქტორის მიერ მეორის გამოძახება

6.9.4. გადატვირთული კონსტრუქტორის გამოძახებისას გამოიძახება ის კონსტრუქტორი, რომელშიც:

ა. პარამეტრების სია ემთხვევა არგუმენტების სიას

ბ. პარამეტრების სია არ ემთხვევა არგუმენტების სიას

გ. პარამეტრების სია ნაწილობრივ ემთხვევა არგუმენტების სიას

6.9.5. გადატვირთვადი კონსტრუქტორის სინტაქსია:

ა. this.კონსტრუქტორის_სახელი(პარამეტრების_სია) : (არგუმენტების_სია)
{ კონსტრუქტორის კოდი }

ბ. კონსტრუქტორის_სახელი(პარამეტრების_სია) : this(არგუმენტების_სია)
{ კონსტრუქტორის კოდი }

გ. this : კონსტრუქტორის_სახელი(პარამეტრების_სია) : this(არგუმენტების_სია)
{ კონსტრუქტორის კოდი }

რეკურსია

6.10.1. რომელი მსჯელობაა სწორი:

ა. მეთოდს შეუძლია თავისი თავის გამოძახება

ბ. მეთოდს არ შეუძლია თავისი თავის გამოძახება

გ. მეთოდს შეუძლია მხოლოდ სხვა მეთოდის გამოძახება

6.10.2. რეკურსია არის პროცესი, როცა მეთოდი:

ა. სხვა მეთოდს იძახებს

ბ. თავის თავს იძახებს

გ. სხვა პროგრამას იძახებს

6.10.3. რეკურსიული არის მეთოდი, რომელიც:

ა. სხვა მეთოდს იძახებს

ბ. თავის თავს იძახებს

გ. სხვა პროგრამას იძახებს

სტატიკური კომპონენტები

static მოდიფიკატორი

6.11.1. კლასის სტატიკური წევრის გამოცხადება იწყება სიტყვით:

ა. static

ბ. public

გ. private

6.11.2. კლასის სტატიკურ წევრთან მუშაობა შეიძლება:

ა. ამ კლასის ობიექტის შექმნის შემდეგ

ბ. ამ კლასის ობიექტის შექმნამდე

გ. მასივის საშუალებით

6.11.3. კლასის სტატიკურ წევრთან მიმართვისათვის უნდა მივუთითოთ:
ა. კლასის სახელი და წევრის სახელი წერტილის გარეშე
ბ. ობიექტის სახელი, წერტილი და წევრის სახელი
გ. კლასის სახელი, წერტილი და წევრის სახელი

6.11.4. თუ სტატიკური ცვლადი რიცხვითი ტიპისაა და არ მიენიჭა საწყისი მნიშვნელობა, მაშინ მას ავტომატურად მიენიჭება:
ა. null
ბ. 0
გ. false

6.11.5. თუ სტატიკური ცვლადი მიმართვითი ტიპისაა და არ მიენიჭა საწყისი მნიშვნელობა, მაშინ მას ავტომატურად მიენიჭება:
ა. null
ბ. 0
გ. true

6.11.6. თუ სტატიკური ცვლადი ლოგიკური ტიპისაა და არ მიენიჭა საწყისი მნიშვნელობა, მაშინ მას ავტომატურად მიენიჭება:
ა. null
ბ. 0
გ. false

6.11.7. სტატიკურ მეთოდს:
ა. აქვს this მიმართვა
ბ. არ აქვს this მიმართვა
გ. ხანდახან აქვს this მიმართვა

6.11.8. სტატიკურ მეთოდს შეუძლია:
ა. პირდაპირ მიმართოს მხოლოდ სტატიკურ მონაცემებს
ბ. პირდაპირ მიმართოს მხოლოდ არასტატიკურ მონაცემებს
გ. პირდაპირ მიმართოს მხოლოდ სტრიქონებს

6.11.9. სტატიკურ მეთოდს შეუძლია:
ა. უშუალოდ გამოიძახოს სტატიკური მეთოდი
ბ. უშუალოდ გამოიძახოს არასტატიკური მეთოდი
გ. ირიბად გამოიძახოს სტატიკური მეთოდი

6.11.10. სტატიკურ მეთოდს შეუძლია მიმართოს თავისი კლასის ობიექტის მეთოდებსა და ცვლადებს მხოლოდ:
ა. სხვა კლასის ობიექტის გამოყენებით
ბ. თავისი კლასის ობიექტის გამოყენებით
გ. ლოგიკური ცვლადების გამოყენებით

მუდმივები

6.12.1. მუდმივას გამოცხადებისათვის გამოიყენება:
ა. this საკვანძო სიტყვა

ბ. static საკვანძო სიტყვა
გ. const საკვანძო სიტყვა

6.12.2. მუდმივას მნიშვნელობა:
ა. აუცილებლად უნდა მივანიჭოთ
ბ. არ უნდა მივანიჭოთ
გ. ხანდახან უნდა მივანიჭოთ

6.12.3. კლასი შეიძლება შეიცავდეს მუდმივ ცვლადებს, რომლებიც ავტომატურად:
ა. არ არის სტატიკური
ბ. არის სტატიკური

მხოლოდწაკითხვადი ველები

6.13.1. მხოლოდ წაკითხვადი ცვლადი:
ა. შეიძლება ეკუთვნოდეს კლასს ან მის ეგზემპლარს
ბ. არ შეიძლება ეკუთვნოდეს კლასს ან მის ეგზემპლარს

6.13.2. თუ მხოლოდ წაკითხვადი ცვლადი ეკუთვნის კლასს, მაშინ ის:
ა. უნდა გამოვაცხადოთ, როგორც პრივატული
ბ. არ უნდა გამოვაცხადოთ, როგორც სტატიკური
გ. უნდა გამოვაცხადოთ, როგორც სტატიკური

6.13.3. თუ მხოლოდ წაკითხვადი ცვლადი ეკუთვნის კლასის ეგზემპლარს, მაშინ მას მნიშვნელობებს:
ა. დესტრუქტორი ანიჭებს
ბ. კონსტრუქტორი ანიჭებს
გ. ინდექსატორი ანიჭებს

თავი 7. მემკვიდრეობითობა. ვირტუალური მეთოდები

მემკვიდრეობითობის საფუძვლები

7.1.1. წინაპარი არის კლასი, რომლის:
ა. ცვლადები და მეთოდები ავტომატურად არ ხდება სხვა ახალი კლასის წევრები
ბ. მხოლოდ ცვლადები ხდება სხვა ახალი კლასის წევრები
გ. ცვლადები და მეთოდები ავტომატურად ხდება სხვა ახალი კლასის წევრები

7.1.2. მემკვიდრე არის კლასი, რომელიც:
ა. წინაპრის კლასის წევრებს ახალ წევრებს არ უმატებს
ბ. წინაპრის კლასის წევრებს მხოლოდ მეთოდებს უმატებს
გ. წინაპრის კლასის წევრებს ახალ წევრებს უმატებს

7.1.3. მემკვიდრე კლასი წინაპარი კლასისაგან:
ა. მემკვიდრეობით იღებს ყველა წევრს
ბ. მემკვიდრეობით არ იღებს ყველა წევრს
გ. მემკვიდრეობით იღებს მხოლოდ ინდექსატორებს

7.1.4. მემკვიდრე კლასისათვის შეგვიძლია მივუთითოთ:
ა. მხოლოდ ორი წინაპარი კლასი
ბ. მხოლოდ ერთი წინაპარი კლასი
გ. რამდენიმე წინაპარი კლასი

7.1.5. მემკვიდრე კლასი:
ა. შეიძლება გახდეს წინაპარი სხვა კლასისათვის
ბ. არ შეიძლება გახდეს წინაპარი სხვა კლასისათვის
გ. შეიძლება გახდეს წინაპარი სტატიკური კლასისათვის

7.1.6. რომელი მსჯელობაა სწორი:
ა. კლასი შეიძლება იყოს თავისი თავის წინაპარი
ბ. მემკვიდრე კლასი შეიძლება გახდეს წინაპარი სტატიკური კლასისათვის
გ. კლასი არ შეიძლება იყოს თავისი თავის წინაპარი

7.1.7. მემკვიდრე კლასის გამოცხადების სინტაქსია:
ა. class წინაპარი_კლასის_სახელი : მემკვიდრე_კლასის_სახელი
{ კლასის კოდი }
ბ. class მემკვიდრე_კლასის_სახელი : წინაპარი_კლასის_სახელი
{ კლასის კოდი }
გ. class წინაპარი_კლასის_სახელი : წინაპარი_კლასის_სახელი
{ კლასის კოდი }

7.1.8. რომელი კოდია სწორი:
ა. class klasi1 { public int cvladi1; }
class klasi2 { public int cvladi2; }
class klasi3 : a1 { public int cvladi3; }
ბ. class klasi1 { public int cvladi1; }

class klasi2 { public int cvladi2; }
class klasi3 : a1, a2 { public int cvladi3; }

გ. class klasi1 { public int cvladi1; }
class klasi2 { public int cvladi2; }
class klasi3 : a2, a1 { public int cvladi3; }

7.1.9. რომელი კოდშია შეცდომა:

ა. class klasi1 { public int cvladi1; }
class klasi2 { public int cvladi2; }
class klasi3 : a1 { public int cvladi3; }

ბ. class klasi1 { public int cvladi1; }
class klasi3 : a1 { public int cvladi3; }

გ. class klasi1 { public int cvladi1; }
class klasi2 { public int cvladi2; }
class klasi3 : a2, a1 { public int cvladi3; }

protected მოდიფიკატორი

7.2.1. მემკვიდრეობითობის დროს:

ა. ძალაში არ არის დახურულ წევრებთან მიმართვისას არსებული შეზღუდვები

ბ. ძალაშია დახურულ წევრებთან მიმართვისას არსებული შეზღუდვები

გ. ძალაშია მხოლოდ დახურულ თვისებებთან მიმართვისას არსებული შეზღუდვები

7.2.2. მემკვიდრე კლასის მეთოდს:

ა. შეუძლია მიმართოს მემკვიდრე კლასის დახურულ წევრებს

ბ. შეუძლია მიმართოს წინაპარი კლასის დახურულ წევრებს

გ. არ შეუძლია მიმართოს წინაპარი კლასის დახურულ წევრებს

7.2.3. კლასის დაცული წევრი:

ა. გახსნილია მხოლოდ მემკვიდრე კლასებისათვის და დახურულია სხვა კლასებისათვის

ბ. დახურულია მხოლოდ მემკვიდრე კლასებისათვის და გახსნილია სხვა კლასებისათვის

გ. დახურულია როგორც მემკვიდრე, ისე სხვა კლასებისათვის

7.2.4. კლასის დაცული წევრი იქმნება:

ა. protected მოდიფიკატორის საშუალებით

ბ. public მოდიფიკატორის საშუალებით

გ. private მოდიფიკატორის საშუალებით

7.2.5. class klasi1 { protected int cvladi1; }
class klasi2 : a1 { public int cvladi2; }
კოდში:

ა. klasi1 კლასის cvladi1 ცვლადი უხილავია klasi2 კლასში

ბ. klasi1 კლასის cvladi1 ცვლადი ხილულია klasi2 კლასში

გ. klasi2 კლასის cvladi2 ცვლადი ხილულია klasi1 კლასში

7.2.6. class klasi1 { private int cvladi1; }
class klasi2 : klasi1 { public int cvladi2; }
კოდში:
ა. klasi1 კლასის cvladi1 ცვლადი ხილულია klasi2 კლასში
ბ. klasi2 კლასის cvladi2 ცვლადი ხილულია klasi1 კლასში
გ. klasi1 კლასის cvladi1 ცვლადი უხილავია klasi2 კლასში

7.2.7. class klasi1 { public int cvladi1; }
class klasi2 : klasi1 { private int cvladi2; }
კოდში:
ა. klasi1 კლასის cvladi1 ცვლადი ხილულია klasi2 კლასში
ბ. klasi2 კლასის cvladi2 ცვლადი ხილულია klasi1 კლასში
გ. klasi1 კლასის cvladi1 ცვლადი უხილავია klasi2 კლასში

7.2.8. class klasi1 { int cvladi1; }
class klasi2 : klasi1 { public int cvladi2; }
კოდში:
ა. klasi1 კლასის cvladi1 ცვლადი ხილულია klasi2 კლასში
ბ. klasi2 კლასის cvladi2 ცვლადი ხილულია klasi1 კლასში
გ. klasi1 კლასის cvladi1 ცვლადი უხილავია klasi2 კლასში

კონსტრუქტორები და მემკვიდრეობითობა. base საკვანძო სიტყვა

7.3.1. მემკვიდრე კლასის ობიექტი:
ა. მთლიანად იქმნება
ბ. ნაწილ-ნაწილ იქმნება
გ. არ იქმნება

7.3.2. თუ კონსტრუქტორი არ არის განსაზღვრული წინაპარ კლასში და განსაზღვრულია მემკვიდრე კლასში, მაშინ:
ა. მემკვიდრე კლასის ობიექტი არ იქმნება
ბ. წინაპარი კლასის ობიექტის კონსტრუირება სრულდება მისი კონსტრუქტორის მიერ, ხოლო მემკვიდრე კლასში განსაზღვრული ობიექტის ნაწილი კი იქმნება მისი ავტომატური კონსტრუქტორის მიერ
გ. მემკვიდრე კლასის ობიექტის კონსტრუირება სრულდება მისი კონსტრუქტორის მიერ, ხოლო წინაპარ კლასში განსაზღვრული ობიექტის ნაწილი კი იქმნება მისი ავტომატური კონსტრუქტორის მიერ

7.3.3. თუ კონსტრუქტორები განსაზღვრულია როგორც წინაპარ, ისე მემკვიდრე კლასებში, მაშინ:
ა. წინაპარი კლასის კონსტრუქტორის გამოსაძახებლად არ გამოიყენება base სიტყვა
ბ. მემკვიდრე კლასის კონსტრუქტორის გამოსაძახებლად გამოიყენება base სიტყვა
გ. წინაპარი კლასის კონსტრუქტორის გამოსაძახებლად გამოიყენება base სიტყვა

7.3.4. base სიტყვის გამოყენებით გამოიძახება ის კონსტრუქტორი, რომლის:
ა. პარამეტრების სია არ შეესაბამება არგუმენტების სიას

ბ. პარამეტრების სია შეესაბამება არგუმენტების სიას
გ. პარამეტრები სტატიკურია

7.3.5. მემკვიდრეობითობის დროს base სიტყვა ყოველთვის მიმართავს:
ა. უახლოესი წინაპარი კლასის კონსტრუქტორს
ბ. ბოლოს წინა კლასის კონსტრუქტორს
გ. მომდევნო მემკვიდრე კლასის კონსტრუქტორს

7.3.6. თუ მემკვიდრე კლასის კონსტრუქტორში არ არის მითითებული base სიტყვა, მაშინ:
ა. არ გამოიძახება წინაპარი კლასის ავტომატური კონსტრუქტორი
ბ. გამოიძახება წინაპარი კლასის ავტომატური კონსტრუქტორი
გ. არ გამოიძახება არც ერთი კონსტრუქტორი

7.3.7. წინაპარ და მემკვიდრე კლასებს:
ა. აუცილებლად უნდა ჰქონდეს საკუთარი კონსტრუქტორები
ბ. არ შეიძლება ჰქონდეს საკუთარი კონსტრუქტორები
გ. შეიძლება ჰქონდეს საკუთარი კონსტრუქტორები

7.3.8. წინაპარი და მემკვიდრე კლასის კონსტრუქტორები:
ა. ქმნის ობიექტის თავის ნაწილებს
ბ. არ ქმნის ობიექტის თავის ნაწილებს
გ. ქმნის ობიექტის ერთმანეთის ნაწილებს

7.3.9. წინაპარი კლასის კონსტრუქტორს:
ა. შეუძლია მიმართოს მემკვიდრე კლასის წევრებს და მოახდინოს მათი ინიციალიზება
ბ. არ შეუძლია მიმართოს მემკვიდრე კლასის წევრებს და მოახდინოს მათი ინიციალიზება
გ. არ შეუძლია მიმართოს მემკვიდრე კლასის წევრებს, მაგრამ შეუძლია მათი ინიციალიზება

7.3.10. თითოეული კონსტრუქტორი ქმნის ობიექტის თავის ნაწილს, რადგან:
ა. წინაპარი კლასის კონსტრუქტორს არ შეუძლია მიმართოს მემკვიდრე კლასის წევრებს, მაგრამ შეუძლია მათი ინიციალიზება
ბ. წინაპარი კლასის კონსტრუქტორს შეუძლია მიმართოს მემკვიდრე კლასის წევრებს და მოახდინოს მათი ინიციალიზება
გ. წინაპარი კლასის კონსტრუქტორს არ შეუძლია მიმართოს მემკვიდრე კლასის წევრებს და მოახდინოს მათი ინიციალიზება

ცვლადების დამალვა მემკვიდრეობითობის დროს

7.4.1. რომელი მსჯელობაა სწორი:
ა. წინაპარი და მემკვიდრე კლასების წევრებს აუცილებლად უნდა ჰქონდეს ერთი და იგივე სახელი
ბ. წინაპარი და მემკვიდრე კლასების წევრებს არ შეიძლება ერთი და იგივე სახელი ჰქონდეს
გ. წინაპარი და მემკვიდრე კლასების წევრებს შეიძლება ერთი და იგივე სახელი ჰქონდეს

7.4.2. თუ წინაპარი და მემკვიდრე კლასების წევრებს ერთნაირი სახელები აქვს, მაშინ:
ა. წინაპარი კლასის წევრი დამალულია წინაპარი კლასის წევრებისათვის

ბ. წინაპარი კლასის წევრი არ არის დამალული მემკვიდრე კლასის წევრებისათვის
გ. წინაპარი კლასის წევრი დამალულია მემკვიდრე კლასის წევრებისათვის

7.4.3. წინაპარი კლასის წევრი დამალულია მემკვიდრე კლასის წევრებისათვის მაშინ, როცა:
ა. წინაპარი და მემკვიდრე კლასების წევრებს ერთნაირი სახელები აქვს
ბ. წინაპარი და მემკვიდრე კლასების წევრებს სხვადასხვა სახელები აქვს
გ. ყველა მემკვიდრე კლასის წევრებს ერთნაირი სახელები აქვს

7.4.4. თუ განზრახ ვმალავთ წინაპარი კლასის წევრს, მაშინ:
ა. მემკვიდრე კლასის შესაბამისი წევრის წინ არ უნდა მივუთითოთ new ოპერატორი
ბ. მემკვიდრე კლასის შესაბამისი წევრის წინ უნდა მივუთითოთ new ოპერატორი
გ. მემკვიდრე კლასის შესაბამისი წევრის წინ უნდა მივუთითოთ static ოპერატორი

7.4.5. როცა განზრახ ვმალავთ წინაპარი კლასის წევრს, მაშინ მემკვიდრე კლასის შესაბამისი წევრის წინ მოთავსებული new ოპერატორი:
ა. ახდენს ობიექტის შექმნას
ბ. არ ახდენს ობიექტის შექმნას
გ. ახდენს კლასის შექმნას

7.4.6. წინაპარი კლასის დამალულ წევრთან მიმართვისათვის გამოიყენება:
ა. protected სიტყვა
ბ. public სიტყვა
გ. base სიტყვა

კლასების მრავალდონიანი იერარქიის შექმნა

7.5.1. რომელი მსჯელობაა სწორი:
ა. მემკვიდრე კლასი შეიძლება იყოს წინაპარი სხვა კლასებისათვის
ბ. მემკვიდრე კლასი არ შეიძლება იყოს წინაპარი სხვა კლასებისათვის
გ. წინაპარი კლასი არ შეიძლება იყოს მემკვიდრე სხვა კლასებისათვის

7.5.2. კლასების იერარქიაში კონსტრუქტორების გამოძახების რიგითობაა:
ა. წინაპარი კლასიდან მემკვიდრე კლასისაკენ
ბ. მემკვიდრე კლასიდან წინაპარი კლასისაკენ
გ. წინაპარი კლასიდან წინაპარი კლასისაკენ

7.5.3. კლასების იერარქიაში თუ არ გამოიყენება base სიტყვა, მაშინ:
ა. არ გამოიძახება თითოეული წინაპარი კლასის ავტომატური კონსტრუქტორი
ბ. გამოიძახება თითოეული წინაპარი კლასის ავტომატური კონსტრუქტორი
გ. გამოიძახება თითოეული მემკვიდრე კლასის ავტომატური კონსტრუქტორი

მიმართვები წინაპარი და მემკვიდრე კლასის ობიექტებზე

7.6.1. ერთი კლასის მიმართვით ცვლადს:
ა. შეიძლება მიენიჭოს მხოლოდ წინაპარი კლასის ობიექტზე მიმართვა

ბ. შეიძლება მიენიჭოს სხვა კლასის ობიექტზე მიმართვა
გ. არ შეიძლება მიენიჭოს სხვა კლასის ობიექტზე მიმართვა

7.6.2. წინაპარი კლასის მიმართვით ცვლადს:
ა. შეიძლება მიენიჭოს მემკვიდრე კლასის ობიექტზე მიმართვა
ბ. არ შეიძლება მიენიჭოს მემკვიდრე კლასის ობიექტზე მიმართვა
გ. შეიძლება მიენიჭოს წინაპარი კლასის ობიექტზე მიმართვა

7.6.3. კლასის წევრთან მიმართვის შესაძლებლობა დამოკიდებულია:
ა. მიმართვითი ცვლადის ტიპზე
ბ. ობიექტის ტიპზე
გ. მასივის ტიპზე

7.6.4. თუ მიმართვით ცვლადს, რომელსაც აქვს წინაპარი კლასის ტიპი, ენიჭება მემკვიდრე კლასის ობიექტზე მიმართვა, მაშინ:
ა. ამ ცვლადის საშუალებით შეგვეძლება მხოლოდ იმ წევრებთან მიმართვა, რომლებიც განსაზღვრული იყო მემკვიდრე კლასში
ბ. ამ ცვლადის საშუალებით შეგვეძლება მხოლოდ იმ წევრებთან მიმართვა, რომლებიც განსაზღვრული იყო წინაპარ კლასში
გ. ამ ცვლადის საშუალებით შეგვეძლება მიმართვა მხოლოდ მასივებთან

ვირტუალური მეთოდები

7.7.1. პროგრამის შესრულების დროს ვირტუალური მეთოდები უზრუნველყოფს:
ა. მასივებთან მუშაობას
ბ. ინკაფსულაციას
გ. პოლიმორფიზმს

7.7.2. ვირტუალური მეთოდის კოდი მემკვიდრე კლასებში:
ა. შეიძლება შეიცვალოს
ბ. არ შეიძლება შეიცვალოს
გ. შეიძლება შეიცვალოს სტატიკური მეთოდებისათვის

7.7.3. virtual საკვანძო სიტყვა გამოიყენება:
ა. სტატიკური მეთოდის გამოცხადებისათვის
ბ. აბსტრაქტული მეთოდის გამოცხადებისათვის
გ. ვირტუალური მეთოდის გამოცხადებისათვის

7.7.4. ვირტუალური მეთოდის ვერსიის არჩევა ხორციელდება:
ა. მიმართვითი ცვლადის ტიპის მიხედვით
ბ. იმ ობიექტის ტიპის შესაბამისად, რომელსაც მიმართვითი ცვლადი მიმართავს
გ. სტატიკური მეთოდის ტიპის მიხედვით

7.7.5. ვირტუალური მეთოდის არჩევა ხორციელდება:
ა. პროგრამის შესრულების დროს
ბ. პროგრამის შესრულების შემდეგ
გ. პროგრამის შესრულებამდე

7.7.6. რომელი მსჯელობაა სწორი:
ა. ტიპი განსაზღვრავს გამოსაძახებელი სტატიკური მეთოდის ვერსიას
ბ. ობიექტის ტიპი განსაზღვრავს გამოსაძახებელი ვირტუალური მეთოდის ვერსიას
გ. მიმართვითი ცვლადის ტიპი განსაზღვრავს გამოსაძახებელი ვირტუალური მეთოდის ვერსიას

7.7.7. მემკვიდრე კლასში ვირტუალური მეთოდის ხელახალი განსაზღვრისას გამოიყენება:
ა. override სიტყვა
ბ. virtual სიტყვა
გ. public სიტყვა

7.7.8. წინაპარ კლასში ვირტუალური მეთოდის განსაზღვრისას გამოიყენება:
ა. override სიტყვა
ბ. public სიტყვა
გ. virtual სიტყვა

7.7.9. ვირტუალური მეთოდი:
ა. შეიძლება განისაზღვროს params მოდიფიკატორის გამოყენებით
ბ. არ შეიძლება განისაზღვროს static მოდიფიკატორის გამოყენებით
გ. შეიძლება განისაზღვროს static მოდიფიკატორის გამოყენებით

7.7.10. მეთოდის ხელახალი განსაზღვრა ეწოდება ვირტუალური მეთოდის განსაზღვრის პროცესს, როცა:
ა. ნაწილობრივ ან მთლიანად იცვლება მეთოდის ტანი, ხოლო მეთოდის სახელი, პარამეტრები და მათი ტიპები იგივე რჩება
ბ. იცვლება მეთოდის ტანი და სახელი
გ. იცვლება მეთოდის სახელი და პარამეტრები

7.7.11. მეთოდის ხელახალი განსაზღვრა საფუძვლად უდევს:
ა. გამოსაძახებელი მეთოდის სტატიკური არჩევის კონცეფციას
ბ. გამოსაძახებელი მეთოდის დინამიკური არჩევის კონცეფციას
გ. მასივების განსაზღვრის კონცეფციას

7.7.12. ვირტუალური მეთოდის გადატვირთვა:
ა. აუცილებელია მხოლოდ მასივებისათვის
ბ. არ არის აუცილებელი
გ. აუცილებელია

7.7.13. თუ მემკვიდრე კლასს არ აქვს ვირტუალური მეთოდის საკუთარი ვერსია, მაშინ:
ა. არ გამოიყენება წინაპარი კლასის მეთოდი
ბ. გამოიყენება წინაპარი კლასის მეთოდი
გ. გამოიყენება მემკვიდრე კლასის მეთოდი

აბსტრაქტული კლასები

7.8.1. მეთოდის ხელმოწერა არის:
ა. მხოლოდ მეთოდის სახელი
ბ. დასაბრუნებელი მნიშვნელობის ტიპი, მეთოდის სახელი და პარამეტრების სია
გ. მხოლოდ პარამეტრების სია

7.8.2. აბსტრაქტული მეთოდების რეალიზება უნდა მოხდეს:
ა. წინაპარ კლასში
ბ. მემკვიდრე კლასებში
გ. მასივებში

7.8.3. აბსტრაქტულ მეთოდებს:
ა. ტანი არ აქვს
ბ. ტანი აქვს
გ. ტანი აქვს წინაპარ კლასში

7.8.4. აბსტრაქტული მეთოდების გამოცხადებისათვის გამოიყენება:
ა. params მოდიფიკატორი
ბ. public მოდიფიკატორი
გ. abstract მოდიფიკატორი

7.8.5. აბსტრაქტული მეთოდი:
ა. არ არის ვირტუალური მეთოდი
ბ. არის ვირტუალური მეთოდი
გ. არის სტატიკური მეთოდი

7.8.6. virtual და abstract მოდიფიკატორების ერთობლივი გამოყენება:
ა. დაუშვებელია
ბ. დასაშვებია
გ. დასაშვებია მასივებთან ერთად

7.8.7. abstract მოდიფიკატორი:
ა. შეიძლება გამოვიყენოთ მასივების მიმართ
ბ. შეიძლება გამოვიყენოთ სტატიკური მეთოდების მიმართ
გ. არ შეიძლება გამოვიყენოთ სტატიკური მეთოდების მიმართ

7.8.8. კლასი, რომელიც აბსტრაქტულ მეთოდებს შეიცავს, გამოცხადებული უნდა იყოს როგორც:
ა. აბსტრაქტული
ბ. ვირტუალური
გ. სტატიკური

7.8.9. აბსტრაქტული კლასის ობიექტის შექმნა:
ა. შესაძლებელია
ბ. შეუძლებელია
გ. შესაძლებელია ლოგიკურ მონაცემებთან ერთად

7.8.10. თუ მემკვიდრე კლასი არ ახდენს წინაპარი კლასის აბსტრაქტული მეთოდების რეალიზებას, მაშინ ეს მემკვიდრე კლასი გამოცხადებული უნდა იყოს როგორც:

ა. ვირტუალური
ბ. აბსტრაქტული
გ. სტატიკური

მემკვიდრეობითობის აკრძალვა

7.9.1. მემკვიდრეობითობის აკრძალვისათვის გამოიყენება:

ა. sealed მოდიფიკატორი
ბ. public მოდიფიკატორი
გ. protected მოდიფიკატორი

7.9.2.
sealed class Klasi1 { }
class Klasi2 : Klasi1 { } კოდი:

ა. არ შეიცავს შეცდომას
ბ. შეიცავს შეცდომას
გ. ხანდახან შეიცავს შეცდომას

პოლიმორფიზმის აკრძალვა

7.10.1. sealed საკვანძო სიტყვა:

ა შეგვიძლია გამოვიყენოთ მეთოდის მიმართ მემკვიდრე კლასებში მათი ხელახალი განსაზღვრის აკრძალვის მიზნით
ბ. არ შეგვიძლია გამოვიყენოთ მეთოდის მიმართ მემკვიდრე კლასებში მათი ხელახალი განსაზღვრის აკრძალვის მიზნით
გ. შეგვიძლია გამოვიყენოთ მეთოდის მიმართ მემკვიდრე კლასებში მათი ხელახალი განსაზღვრის ნებართვის მიზნით

7.10.2. მეთოდს, რომლის ხელახალი განსაზღვრა აკრძალულია მეკვიდრე კლასში, ეწოდება:

ა. დაულუქავი მეთოდი
ბ. დალუქული მეთოდი
გ. ღია მეთოდი

7.10.3.
class Sabazo { public virtual void Metodi() { } }
class Memkvidre : Sabazo { sealed public override void Metodi() { } } კოდი:

ა. ხანდახან შეიცავს შეცდომას
ბ. შეიცავს შეცდომას
გ. არ შეიცავს შეცდომას

7.10.4.
class Sabazo { public virtual void Metodi() { } }
class Memkvidre : Sabazo { sealed public override void Metodi() { } } კოდი:
class Klasi : Memkvidre { public override void Metodi() { } }

ა. ხანდახან შეიცავს შეცდომას

ბ. შეიცავს შეცდომას
გ. არ შეიცავს შეცდომას

ობიექტების ტიპების დაყვანა

7.11.1. თუ ხდება მემკვიდრე კლასის ობიექტის ტიპის დაყვანა საბაზო კლასის ტიპზე, მაშინ:
ა. სრულდება დაყვანა ზევით
ბ. სრულდება დაყვანა ქვევით
გ. სრულდება დაყვანა ობიექტის ტიპზე

7.11.2. თუ ხდება მემკვიდრე კლასის ობიექტის ტიპის დაყვანა მემკვიდრე კლასის ტიპზე, მაშინ:
ა. სრულდება დაყვანა ზევით
ბ. სრულდება დაყვანა ობიექტის ტიპზე
გ. სრულდება დაყვანა ქვევით

7.11.3. ერთი კლასის ობიექტები შეიძლება გარდავქმნათ მეორე კლასის ობიექტების ტიპად იმ შემთხვევაში, თუ:
ა. ეს კლასები არათავსებადია
ბ. ეს კლასები თავსებადია
გ. ორივე კლასი მემკვიდრეა

7.11.4. მემკვიდრე კლასი:
ა. არათავსებადია საბაზო კლასთან
ბ. თავსებადია საბაზო კლასთან
გ. ხან თავსებადია, ხან არა

7.11.5. ობიექტის ტიპის დაყვანის შემთხვევაში, მიმართვა შეგვეძლება მხოლოდ იმ წევრებთან, რომლებიც:
ა. განსაზღვრულია იმ კლასში, რომლის ტიპზეც არ ხდება დაყვანა
ბ. არაა განსაზღვრული იმ კლასში, რომლის ტიპზეც ხდება დაყვანა
გ. განსაზღვრულია იმ კლასში, რომლის ტიპზეც ხდება დაყვანა

7.11.6. კლასები არათავსებადია, თუ:
ა. ერთი კლასი არ არის მეორე კლასის მემკვიდრე
ბ. ერთი კლასი არის მეორე კლასის მემკვიდრე
გ. თუ ერთ-ერთი კლასი პრივატულია

თავი 8. თვისებები, ინდექსატორები და ოპერატორების გადატვირთვა

თვისებები

8.1.1. თვისება არის კლასის წევრი, რომელიც საშუალებას გვაძლევს:
ა. ვიმუშაოთ მხოლოდ მასივებთან
ბ. მეთოდების საშუალებით მივიღოთ ან შევცვალოთ ცვლადების მნიშვნელობები
გ. ვიმუშაოთ მხოლოდ წილადებთან

8.1.2. თვისება შედგება:
ა. მხოლოდ public მოდიფიკატორიანი მეთოდებისაგან
ბ. მხოლოდ სტატიკური ცვლადებისაგან
გ. მიმართვის get და set მეთოდებისაგან

8.1.3. თვისების get მეთოდი გამოიყენება:
ა. ცვლადის მნიშვნელობის მისაღებად
ბ. ცვლადის მნიშვნელობის შესაცვლელად
გ. ცვლადის მნიშვნელობისათვის წილადის მისანიჭებლად

8.1.4. თვისების set მეთოდი გამოიყენება:
ა. მხოლოდ წილადი მნიშვნელობის გასაცემად
ბ. ცვლადის მნიშვნელობის გადასაცემად
გ. ცვლადისათვის მნიშვნელობის მისანიჭებლად

8.1.5. თვისების set მეთოდს ავტომატურად გადაეცემა:
ა. value არაცხადი პარამეტრი
ბ. value ცხადი პარამეტრი
გ. კლასის მისამართი

8.1.6. თვისების set მეთოდისთვის ავტომატურად გადაცემული value პარამეტრი:
ა. არ შეიცავს ცვლადისათვის მისანიჭებელ მნიშვნელობას
ბ. შეიცავს ცვლადისათვის მისანიჭებელ მნიშვნელობას
გ. შეიცავს მხოლოდ მთელ რიცხვს

8.1.7. რომელი მსჯელობაა სწორი:
ა. თვისების სახელი არ შეიძლება გამოვიყენოთ გამოსახულებაში
ბ. თვისების სახელი შეიძლება გამოვიყენოთ გამოსახულებაში
გ. თვისების სახელი შეიძლება გამოვიყენოთ მხოლოდ მთელი რიცხვების მიმართ

8.1.8. რომელი მსჯელობაა სწორი:
ა. ჩვეულებრივ, თვისებაში არ ხდება ცვლადის გამოცხადება
ბ. თვისებაში აუცილებლად ხდება ცვლადის გამოცხადება
გ. თვისებაში ხდება მხოლოდ წილადი რიცხვების გამოცხადება

8.1.9. თუ თვისების სახელი მითითებულია მინიჭების ოპერატორის მარცხნივ, მაშინ:
ა. გამოიძახება get მეთოდი

ბ. არ გამოიძახება არც ერთი მეთოდი
გ. გამოიძახება set მეთოდი

8.1.10. თუ თვისების სახელი მითითებულია მინიჭების ოპერატორის მარჯვნივ, მაშინ:
ა. გამოიძახება get მეთოდი
ბ. გამოიძახება set მეთოდი
გ. გამოიძახება ორივე მეთოდი

8.1.11. რომელი მსჯელობაა სწორი:
ა. თვისება შეიძლება გადაეცეს მეთოდს, როგორც ref ან out პარამეტრი
ბ. თვისება შეიძლება გადაეცეს მეთოდს, მხოლოდ როგორც ref პარამეტრი
გ. თვისება არ შეიძლება გადაეცეს მეთოდს, როგორც ref ან out პარამეტრი

8.1.12. რომელი მსჯელობაა სწორი:
ა. თვისება შეიძლება იყოს გადატვირთული
ბ. თვისება არ შეიძლება იყოს გადატვირთული
გ. თვისება ნაწილობრივ შეიძლება იყოს გადატვირთული

8.1.13. რომელი მსჯელობაა სწორი:
ა. ჩვეულებრივ, თვისებამ არ უნდა შეცვალოს ცვლადების მნიშვნელობა get მეთოდის გამოყენებით
ბ. თვისებამ აუცილებლად უნდა შეცვალოს ცვლადების მნიშვნელობა get მეთოდის გამოყენებით
გ. თვისებამ აუცილებლად უნდა შეცვალოს მხოლოდ სტრიქონი get მეთოდის გამოყენებით

ინდექსატორები

ერთგანზომილებიანი ინდექსატორები

8.2.1. ინდექსატორი საშუალებას გვაძლევს ობიექტის ცვლადებს მივმართოთ:
ა. ობიექტის სახელისა და ინდექსის საშუალებით
ბ. მხოლოდ ობიექტის სახელის საშუალებით
გ. მხოლოდ ინდექსის საშუალებით

8.2.2. ინდექსატორს შეიძლება ჰქონდეს:
ა. ერთი ან მეტი განზომილება
ბ. მხოლოდ ერთი განზომილება
გ. მხოლოდ ორი განზომილება

8.2.3. თუ ინდექსატორი მითითებულია მინიჭების ოპერატორის მარცხნივ, მაშინ:
ა. არც ერთი მეთოდი არ გამოიძახება
ბ. გამოიძახება get მეთოდი
გ. გამოიძახება set მეთოდი

8.2.4. თუ ინდექსატორი მითითებულია მინიჭების ოპერატორის მარჯვნივ, მაშინ:
ა. გამოიძახება set მეთოდი
ბ. გამოიძახება get მეთოდი
გ. ორივე მეთოდი გამოიძახება

8.2.5. რომელი მსჯელობაა მცდარი:
ა. ინდექსატორში შეიძლება განსაზღვრული იყოს set და get მეთოდები
ბ. ინდექსატორში შეიძლება არ იყოს განსაზღვრული set და get მეთოდები
გ. ინდექსატორში შეიძლება განსაზღვრული იყოს set ან get მეთოდი

8.2.6. რომელი მსჯელობაა სწორი:
ა. ინდექსატორისათვის არ შეიძლება ref ან out მოდიფიკატორის გამოყენება
ბ. ინდექსატორისათვის შეიძლება ref ან out მოდიფიკატორის გამოყენება
გ. ინდექსატორისათვის შეიძლება მხოლოდ ref მოდიფიკატორის გამოყენება

ოპერატორების გადატვირთვა

8.3.1. ოპერატორის გადატვირთვა არის მოქმედებების ერთობლიობა:
ა. რომლებსაც ოპერატორი შეასრულებს ჩვენ მიერ შექმნილი კლასის მიმართ
ბ. როცა იცვლება ოპერატორის საწყისი დანიშნულება
გ. როცა ხდება კლასის გამოცხადება

8.3.2. ოპერატორის გადატვირთვისას მისი:
ა. გამოყენება ხდება მასივებში
ბ. საწყისი დანიშნულება იცვლება
გ. საწყისი დანიშნულება არ იცვლება

8.3.3. ოპერატორის გადატვირთვისას:
ა. მის არსენალს ახალი ფუნქციები არ ემატება
ბ. მის არსენალს ახალი ფუნქციები ემატება
გ. ხდება ინდექსატორის გამოცხადება

8.3.4. ოპერატორის გადატვირთვისათვის გამოიყენება:
ა. public სიტყვა
ბ. operator სიტყვა
გ. static სიტყვა

8.3.5. operator სიტყვა განსაზღვრავს:
ა. ოპერატორის მეთოდს
ბ. გადატვირთვად მეთოდს
გ. სტატიკურ მეთოდს

8.3.6. უნარულია ოპერატორი, რომელსაც აქვს:
ა. სამი ოპერანდი
ბ. ორი ოპერანდი
გ. ერთი ოპერანდი

8.3.7. ბინარულია ოპერატორი, რომელსაც აქვს:
ა. სამი ოპერანდი
ბ. ერთი ოპერანდი
გ. ორი ოპერანდი

8.3.8. უნარული ოპერატორის გადატვირთვისას:
ა. ოპერანდს უნდა ჰქონდეს მთელი ტიპი
ბ. ოპერანდი უნდა ეკუთვნოდეს იმ კლასის ტიპს, რომლისათვისაც განისაზღვრა ოპერატორი
გ. ოპერანდს უნდა ჰქონდეს ლოგიკური ტიპი

8.3.9. ბინარული ოპერატორების გადატვირთვისას:
ა. ერთ-ერთი ოპერანდის ტიპი უნდა ემთხვეოდეს იმ კლასის ტიპს, რომლისთვისაც ხდება ოპერატორის ხელახალი განსაზღვრა
ბ. ერთ-ერთი ოპერანდი უნდა იყოს მთელი ტიპის
გ. ერთ-ერთი ოპერანდი უნდა იყოს წილადი ტიპის

8.3.10. არ შეიძლება + ოპერატორის გადატვირთვა:
ა. არარსებული ტიპებისათვის
ბ. უკვე არსებული ტიპებისათვის
გ. ნებისმიერი ტიპისათვის

8.3.11. - ოპერატორის გადატვირთვისას მნიშვნელობა:
ა. აქვს მხოლოდ მთელი რიცხვების მიმდევრობას
ბ. არ აქვს ოპერანდების მიმდევრობას
გ. აქვს ოპერანდების მიმდევრობას

8.3.12. არაკომუტაციური ოპერატორების გადატვირთვისას:
ა. არ უნდა გავითვალისწინოთ ოპერანდების ურთიერთგანლაგება
ბ. უნდა გავითვალისწინოთ მხოლოდ მთელი რიცხვების ურთიერთგანლაგება
გ. უნდა გავითვალისწინოთ ოპერანდების ურთიერთგანლაგება

8.3.13. კომუტაციური ოპერატორების გადატვირთვისას:
ა. არ უნდა გავითვალისწინოთ ოპერანდების ურთიერთგანლაგება
ბ. უნდა გავითვალისწინოთ ოპერანდების ურთიერთგანლაგება
გ. უნდა გავითვალისწინოთ ლოგიკური მონაცემების ურთიერთგანლაგება

8.3.14. შედარების გადატვირთული ოპერატორები აბრუნებენ:
ა. true ან false მნიშვნელობას
ბ. true ან int ტიპის მნიშვნელობას
გ. false ან double ტიპის მნიშვნელობას

8.3.15. შედარების ოპერატორების გადატვირთვა:
ა. უნდა შესრულდეს წყვილობრივად
ბ. არ უნდა შესრულდეს წყვილობრივად
გ. უნდა შესრულდეს მხოლოდ მთელი რიცხვებისათვის

8.3.16. C# ენაში არსებობს ოპერატორების შემდეგი წყვილები:
ა. < და <=, > და >=, <= და >=
ბ. == და >=, <= და !=, < და >
გ. == და !=, > და <, <= და >=

8.3.17. ოპერატორების გადატვირთვისას:

ა. შეიძლება ოპერატორების პრიორიტეტების შეცვლა

ბ. შეუძლებელია ოპერატორების პრიორიტეტების შეცვლა

გ. შეიძლება ოპერატორების პრიორიტეტების შეცვლა მხოლოდ მთელი რიცხვებისათვის

8.3.18. ოპერატორების გადატვირთვისას:

ა. შეიძლება ოპერატორის ოპერანდების რაოდენობის შეცვლა

ბ. შეიძლება ოპერატორის ოპერანდების რაოდენობის შეცვლა მხოლოდ წილადებისათვის

გ. არ შეიძლება ოპერატორის ოპერანდების რაოდენობის შეცვლა

8.3.19. მინიჭების ოპერატორის გადატვირთვა:

ა. შეიძლება მხოლოდ მთელი რიცხვებისათვის

ბ. შეიძლება

გ. არ შეიძლება

8.3.20. რომელი ოპერატორების გადატვირთვა არ შეიძლება:

ა. + , <

ბ. && , ||

გ. * , >=

8.3.21. რომელი ოპერატორების გადატვირთვა შეიძლება:

ა. / , %

ბ. ? , =

გ. new , >

თავი 9. დელეგატები, მოვლენები, ინტერფეისები და სახელების სივრცე

დელეგატები

9.1.1. დელეგატი არის ობიექტი, რომელიც:
ა. მეთოდს მიმართავს
ბ. მეთოდს არ მიმართავს
გ. ობიექტს მიმართავს

9.1.2. რომელი მსჯელობაა სწორი:
ა. დელეგატი არის ობიექტი
ბ. დელეგატი არ არის ობიექტი
გ. დელეგატი არის მეთოდი

9.1.3. დელეგატების გამოყენებით შესასრულებელი მეთოდის არჩევა ხდება:
ა. სტატიკურად
ბ. დინამიკურად
გ. არადინამიკურად

9.1.4. დელეგატები უზრუნველყოფს:
ა. კლასებს
ბ. მასივებს
გ. მოვლენებს

9.1.5. რომელი მსჯელობაა სწორი:
ა. დელეგატი შეიცავს მეთოდის სიგნატურას
ბ. დელეგატი არ შეიცავს მეთოდის სიგნატურას
გ. დელეგატი შეიცავს მასივს

9.1.6. მეთოდის სიგნატურა არის:
ა. მეთოდის სახელი და პარამეტრების სია
ბ. მეთოდის მიერ დაბრუნებული მნიშვნელობის ტიპი
გ. მეთოდის სახელი, მის მიერ დაბრუნებული მნიშვნელობის ტიპი და პარამეტრების სია

9.1.7. რომელი მსჯელობაა სწორი:
ა. დელეგატი არ ინახავს მეთოდის მისამართს
ბ. დელეგატი ინახავს მეთოდის მისამართს
გ. დელეგატი ინახავს კლასის მისამართს

9.1.8. ერთი დელეგატი:
ა. შეგვიძლია გამოვიყენოთ რამდენიმე მეთოდის გამოძახებისათვის
ბ. არ შეგვიძლია გამოვიყენოთ რამდენიმე მეთოდის გამოძახებისათვის
გ. შეგვიძლია გამოვიყენოთ მხოლოდ ორი მეთოდის გამოძახებისათვის

9.1.9. ერთი დელეგატი:
ა. შეიძლება ინახავდეს რამდენიმე მეთოდის მისამართს

ბ. არ შეიძლება ინახავდეს რამდენიმე მეთოდის მისამართს
გ. შეიძლება ინახავდეს მხოლოდ ერთი მეთოდის მისამართს

9.1.10. დელეგატის გამოცხადებისათვის გამოიყენება:
ა. public სიტყვა
ბ. delegate სიტყვა
გ. abstract სიტყვა

9.1.11. დელეგატს შეუძლია გამოიძახოს მხოლოდ ის მეთოდი, რომლის:
ა. ტიპია int
ბ. სიგნატურა არ შეესაბამება დელეგატის სიგნატურას
გ. სიგნატურა შეესაბამება დელეგატის სიგნატურას

9.1.12. დელეგატის საშუალებით შეიძლება გამოვიძახოთ:
ა. მხოლოდ ობიექტის მეთოდი
ბ. ობიექტის მეთოდი ან კლასის სტატიკური მეთოდი
გ. მხოლოდ სტატიკური მეთოდი

დელეგატების მრავალმისამართიანობა

9.1.13. მისამართების მიმდევრობით ინიციალიზების გზით დელეგატს:
ა. შეუძლია ერთმანეთის მიყოლებით გამოიძახოს შესაბამისი მეთოდები
ბ. არ შეუძლია ერთმანეთის მიყოლებით გამოიძახოს შესაბამისი მეთოდები
გ. შეუძლია ერთდროულად გამოიძახოს ყველა მეთოდი

9.1.14. დელეგატის მრავალმისამართიანობა არის მისი თვისება:
ა. არ შეინახოს რამდენიმე მისამართი
ბ. შეინახოს რამდენიმე მისამართი
გ. შეინახოს მხოლოდ 1 მისამართი

9.1.15. მეთოდებზე მიმართვების მწკრივი არის:
ა. გამოსაძახებელი მეთოდების მისამართების მიმდევრობა
ბ. მხოლოდ სტატიკური მეთოდების მისამართების მიმდევრობა
გ. არაგამოსაძახებელი მეთოდების მისამართების მიმდევრობა

9.1.16. მეთოდებზე მიმართვების მწკრივისათვის მისამართის დასამატებლად გამოიყენება:
ა. -= ოპერატორი
ბ. += ოპერატორი
გ. + ოპერატორი

9.1.17. მეთოდებზე მიმართვების მწკრივიდან მისამართის წასაშლელად გამოიყენება:
ა. += ოპერატორი
ბ. -= ოპერატორი
გ. - ოპერატორი

9.1.18. დელეგატს, რომელიც რამდენიმე მიმართვას ინახავს, უნდა ჰქონდეს დასაბრუნებელი მნიშვნელობის ტიპი:
ა. double

ბ. int
გ. void

მოვლენები

9.2.1. რომელი მსჯელობაა სწორი:
ა. მოვლენა არის სტატიკური მეთოდის რეალიზება
ბ. მოვლენა არ არის ავტომატური უწყება რაიმე მომხდარ მოქმედებაზე
გ. მოვლენა არის ავტომატური უწყება რაიმე მომხდარ მოქმედებაზე

9.2.2. მოვლენის მაგალითებია:
ა. კლავიატურის კლავიშზე თითის დაჭერა
ბ. მასივის განულება
გ. ცვლადის ინიციალიზება

9.2.3. მოვლენის დამამუშავებელი იქმნება:
ა. მოვლენის საფუძველზე
ბ. დელეგატის საფუძველზე
გ. მასივის საფუძველზე

9.2.4. მოვლენის გამოცხადებისათვის გამოიყენება:
ა. delegate სიტყვა
ბ. event სიტყვა
გ. string სიტყვა

9.2.5. მოვლენის დამამუშავებელი აქტიურდება:
ა. დელეგატის საშუალებით
ბ. მოვლენის საშუალებით
გ. კლასის საშუალებით

9.2.6. რომელი მსჯელობაა სწორი:
ა. მოვლენა არ შეიძლება იყოს მრავალმისამართიანი
ბ. მოვლენა შეიძლება იყოს მრავალმისამართიანი
გ. მოვლენა შეიძლება იყოს მხოლოდ ერთმისამართიანი

9.2.7. მოვლენის დამამუშავებლისათვის უნდა მიეთითოს დასაბრუნებელი მნიშვნელობის ტიპი:
ა. bool
ბ. int
გ. void

ფართოსამაუწყებლო მოვლენა

9.2.8. ერთ მოვლენას შეიძლება ჰქონდეს:
ა. მხოლოდ 1 დამამუშავებელი
ბ. რამდენიმე დამამუშავებელი
გ. მხოლოდ 2 დამამუშავებელი

9.2.9. მოვლენას, რომლის დამამუშავებლების ნაწილი შეიძლება განსაზღვრული იყოს სხვადასხვა ობიექტებში, ეწოდება:
ა. ფართოსამაუწყებლო
ბ. სტატიკური
გ. ვირტუალური

9.2.10. ფართოსამაუწყებლო მოვლენა ბევრ ობიექტს საშუალებას:
ა. აძლევს რეაგირება არ მოახდინოს მოვლენის შესახებ უწყებაზე
ბ. არ აძლევს რეაგირება მოახდინოს მოვლენის შესახებ უწყებაზე
გ. აძლევს რეაგირება მოახდინოს მოვლენის შესახებ უწყებაზე

9.2.11. მოვლენის შესახებ უწყების მისაღებად წინასწარ:
ა. არ უნდა იყოს რეგისტრირებული ამ მოვლენის დამამუშავებელი თითოეული ობიექტისათვის
ბ. უნდა იყოს რეგისტრირებული ამ მოვლენის დამამუშავებელი ობიექტის გარეშე
გ. უნდა იყოს რეგისტრირებული ამ მოვლენის დამამუშავებელი თითოეული ობიექტისათვის

9.2.12. რადგან მოვლენის დამამუშავებელი ცალ-ცალკე რეგისტრირდება თითოეული ობიექტისათვის, ამიტომ:
ა. ყველა ობიექტი ერთდროულად იღებს უწყებას მომხდარი მოვლენის შესახებ
ბ. თითოეული ობიექტი ცალკე იღებს უწყებას მომხდარი მოვლენის შესახებ
გ. არც ერთი ობიექტი არ იღებს უწყებას მომხდარი მოვლენის შესახებ

ინტერფეისები

ინტერფეისების რეალიზება

9.3.1. ინტერფეისში შეიძლება განისაზღვროს:
ა. ინდექსატორები და მასივები
ბ. მეთოდები და სტრიქონები
გ. მეთოდები, თვისებები, ინდექსატორები და მოვლენები

9.3.2. კლასის ინტერფეისი ამ კლასის რეალიზაციისაგან შეიძლება:
ა. მთლიანად განვაცალკევოთ
ბ. ნაწილობრივ განვაცალკევოთ
გ. ვერ განვაცალკევოთ

9.3.3. ინტერფეისის რეალიზება:
ა. შეუძლებელია რამდენიმე კლასის საშუალებით
ბ. შესაძლებელია რამდენიმე კლასის საშუალებით
გ. შესაძლებელია მხოლოდ 1 კლასის საშუალებით

9.3.4. კლასი, რომელიც ახდენს ინტერფეისის რეალიზებას:
ა. არ შეიძლება შეიცავდეს დამატებით ელემენტებს
ბ. შეიძლება შეიცავდეს დამატებით ელემენტებს
გ. ნაწილობრივ შეიძლება შეიცავდეს დამატებით ელემენტებს

9.3.5. ინტერფეისში არ შეიძლება გამოვაცხადოთ:
ა. ველები (ცვლადები)
ბ. მეთოდები
გ. ინტერფეისები

9.3.6. ინტერფეისში მეთოდების გამოცხადება სრულდება:
ა. კლასის საშუალებით
ბ. მასივის საშუალებით
გ. მათი სიგნატურისა და დასაბრუნებელი მნიშვნელობის ტიპის მითითების გზით

9.3.7. ინტერფეისში გამოცხადებული მეთოდებისათვის იგულისხმება:
ა. protected მოდიფიკატორი
ბ. private მოდიფიკატორი
გ. public მოდიფიკატორი

9.3.8. ინტერფეისში არ შეიძლება გამოვაცხადოთ:
ა. ინდექსატორები
ბ. სტატიკური წევრები
გ. მოვლენები

9.3.9. ინტერფეისში არ შეიძლება გამოვაცხადოთ:
ა. თვისებები
ბ. მოვლენები
გ. კონსტრუქტორები

9.3.10. ინტერფეისში არ შეიძლება გამოვაცხადოთ:
ა. ინდექსატორები
ბ. დესტრუქტორები
გ. მოვლენები

9.3.11. კლასმა ინტერფეისის რეალიზება უნდა მოახდინოს:
ა. მთლიანად
ბ. ნაწილობრივ
გ. მიმდევრობით

9.3.12. ერთ კლასს შეუძლია:
ა. მხოლოდ ერთი ინტერფეისის რეალიზება
ბ. ერთზე მეტი ინტერფეისის რეალიზება
გ. მხოლოდ ორი ინტერფეისის რეალიზება

კლასების მემკვიდრეობითობა და ინტერფეისის რეალიზება

9.4.1. როდესაც კლასი ახდენს ერთზე მეტი ინტერფეისის რეალიზებას, მაშინ ინტერფეისების სახელები ერთმანეთისაგან:
ა. მძიმეებით უნდა გამოიყოს
ბ. წერტილებით უნდა გამოიყოს
გ. წერტილ-მძიმეებით უნდა გამოიყოს

9.4.2. კლასი შეიძლება იყოს საბაზო კლასის მემკვიდრე და:
ა. აუცილებლად უნდა ახდენდეს ერთი ინტერფეისის რეალიზებას
ბ. ახდენდეს ერთი ან მეტი ინტერფეისის რეალიზებას
გ. აუცილებლად უნდა ახდენდეს ორი ინტერფეისის რეალიზებას

9.4.3. მემკვიდრეობითობის შემთხვევაში ინტერფეისის რეალიზებისას:
ა. წინაპარი კლასის სახელი უნდა იყოს მესამე
ბ. წინაპარი კლასის სახელი უნდა იყოს მეორე
გ. წინაპარი კლასის სახელი უნდა იყოს პირველი

9.4.4.
public interface Interpeisi_A { ... }
public interface Interpeisi_B { ... }
public class ChemiKlasi { ... }
public class Manqana : ChemiKlasi, Interpeisi_A, Interpeisi_B { ... } კოდი:
ა. შეიცავს შეცდომას
ბ. არ შეიცავს შეცდომას
გ. შეიცავს შეცდომას და არ შეიცავს შეცდომას

9.4.5.
public interface Interpeisi_A { ... }
public interface Interpeisi_B { ... }
public class ChemiKlasi { ... }
public class Manqana : Interpeisi_A, ChemiKlasi, Interpeisi_B { ... } კოდი:
ა. შეიცავს შეცდომას
ბ. არ შეიცავს შეცდომას
გ. შეიცავს შეცდომას და არ შეიცავს შეცდომას

წარმოებული ინტერფეისები

9.5.1. კლასების მსგავსად, ინტერფეისები:
ა. შეიძლება იყოს წარმოებული (მიღებული) მემკვიდრეობით
ბ. არ შეიძლება იყოს წარმოებული (მიღებული) მემკვიდრეობით
გ. ხანდახან შეიძლება იყოს წარმოებული (მიღებული) მემკვიდრეობით

9.5.2. კლასებისაგან განსხვავებით, ინტერფეისი:
ა. არ შეიძლება მიღებული იყოს ერთზე მეტი ინტერფეისისაგან
ბ. შეიძლება მიღებული იყოს ერთზე მეტი ინტერფეისისაგან
გ. შეიძლება მიღებული იყოს მხოლოდ ერთი ინტერფეისისაგან

9.5.3.
public interface Interpeisi_A { ... }
public interface Interpeisi_B : Interfeisi_A { ... }
public class ChemiKlasi : Interfeisi_B { ... } კოდი:
ა. შეიცავს შეცდომას
ბ. არ შეიცავს შეცდომას
გ. შეიცავს შეცდომას და არ შეიცავს შეცდომას

9.5.4.
public interface Interpeisi_A { ... }
public interface Interpeisi_B { ... }
public interface Interfeisi_C : Interfeisi_A, Interfeisi_B { ... }
public class ChemiKlasi : Interfeisi_C { ... } კოდი:
ა. შეიცავს შეცდომას და არ შეიცავს შეცდომას
ბ. შეიცავს შეცდომას
გ. არ შეიცავს შეცდომას

ინტერფეისული მიმართვები

9.6.1. შეგვიძლია გამოვაცხადოთ მიმართვითი ცვლადი, რომელსაც:
ა. აქვს ინტერფეისული ტიპი
ბ. არ აქვს ინტერფეისული ტიპი
გ. ხანდახან აქვს ინტერფეისული ტიპი

9.6.2. ინტერფეისული მიმართვითი ცვლადი:
ა. არ შეგვიძლია შევქმნათ
ბ. შეგვიძლია შევქმნათ
გ. ხანდახან შეგვიძლია შევქმნათ

9.6.3. ინტერფეისული ცვლადი შეიძლება მიმართავდეს ნებისმიერ ობიექტს, რომელიც:
ა. ახდენს სხვა ინტერფეისის რეალიზებას
ბ. არ ახდენს მისი ინტერფეისის რეალიზებას
გ. ახდენს მისი ინტერფეისის რეალიზებას

9.6.4. ინტერფეისული მიმართვის საშუალებით ობიექტის მეთოდის გამოძახებისას:
ა. გამოიძახება ის მეთოდი, რომლის რეალიზებას მოცემული ობიექტი ახდენს
ბ. არ გამოიძახება ის მეთოდი, რომლის რეალიზებას მოცემული ობიექტი ახდენს
გ. გამოიძახება ის მეთოდი, რომლის რეალიზებას მოცემული ობიექტი არ ახდენს

9.6.5.
public interface ChemiInterpeisi { ... }
class Klasi1 : ChemiInterpeisi { ... }
class Klasi2 : ChemiInterpeisi { ... }
{
Klasi1 Samkutxedi = new Klasi1(gverdi1, gverdi2, gverdi3);
Klasi2 Otxkutxedi = new Klasi2(gverdi1, gverdi2);
ChemiInterpeisi obieqti;
obieqti = Samkutxedi;
obieqti = Otxkutxedi;
}
კოდში obieqti ობიექტი გამოცხადებულია როგორც:
ა. მიმართვა Samkutxedi ინტერფეისზე
ბ. მიმართვა ChemiInterpeisi ინტერფეისზე
გ. მიმართვა Otxkutxedi ინტერფეისზე

9.6.6.

```
public interface ChemiInterpeisi { ... }
class Klasi1 : ChemiInterpeisi { ... }
class Klasi2 : ChemiInterpeisi { ... }
{
Klasi1 Samkutxedi = new Klasi1(gverdi1, gverdi2, gverdi3);
Klasi2 Otxkutxedi = new Klasi2(gverdi1, gverdi2);
ChemiInterpeisi obieqti;
obieqti = Samkutxedi;
obieqti = Otxkutxedi;
}
```

რადგან obieqti ობიექტი გამოცხადებულია როგორც მიმართვა ChemiInterpeisi ინტერფეისზე, ამიტომ ის :

ა. შეიძლება გამოყენებულ იქნას იმ ობიექტზე მიმართვების შენახვის მიზნით, რომელიც არ ახდენს ChemiInterpeisi ინტერფეისის რეალიზებას

ბ. შეიძლება გამოყენებულ იქნას იმ ობიექტზე მიმართვების შენახვის მიზნით, რომელიც ახდენს ChemiInterpeisi ინტერფეისის რეალიზებას

გ. არ შეიძლება გამოყენებულ იქნას იმ ობიექტზე მიმართვების შენახვის მიზნით, რომელიც ახდენს ChemiInterpeisi ინტერფეისის რეალიზებას

9.6.7.

```
public interface ChemiInterpeisi { ... }
class Klasi1 : ChemiInterpeisi { ... }
class Klasi2 : ChemiInterpeisi { ... }
{
Klasi1 Samkutxedi = new Klasi1(gverdi1, gverdi2, gverdi3);
Klasi2 Otxkutxedi = new Klasi2(gverdi1, gverdi2);
ChemiInterpeisi obieqti;
obieqti = Samkutxedi;
obieqti = Otxkutxedi;
}
```

obieqti ინტერფეისულმა მიმართვამ იცის მხოლოდ იმ მეთოდების შესახებ, რომლებიც:

ა. გამოცხადებულია კლასში

ბ. არაა გამოცხადებული ინტერფეისში

გ. გამოცხადებულია ინტერფეისში

ინტერფეისის თვისებები

9.7.1. ინტერფეისში თვისების განსაზღვრისას:

ა. მიეთითება თვისების ტანი

ბ. არ მიეთითება თვისების ტანი

გ. მიეთითება ინდექსატორი

ცხადი რეალიზება

9.8.1. ცხადი რეალიზაციის დროს:

ა. ინტერფეისის ელემენტის წინ მიეთითება ინტერფეისის სახელი

ბ. ინტერფეისის ელემენტის წინ არ მიეთითება ინტერფეისის სახელი

გ. ინტერფეისის ელემენტის წინ მიეთითება კლასის სახელი

9.8.2. ცხადი რეალიზაციის გამოყენება საჭიროა მაშინ, როცა:
ა. პროგრამაში გამოცხადებულია ორი ინტერფეისი, რომლებიც არ შეიცავენ ერთნაირი სახელის მქონე მეთოდებს
ბ. პროგრამაში გამოცხადებულია ორი ინტერფეისი, რომლებიც შეიცავენ ერთნაირი სახელის მქონე მეთოდებს
გ. პროგრამაში გამოცხადებულია სამი ინტერფეისი, რომლებიც არ შეიცავენ ერთნაირი სახელის მქონე მეთოდებს

სახელების სივრცე

9.9.1. სახელების სივრცე კლასების სახელებს შორის კონფლიქტს:
ა. თავიდან გვაცილებს
ბ. თავიდან არ გვაცილებს
გ. იწვევს

9.9.2. სახელების სივრცეში სახელების ერთი ნაკრები:
ა. არ შეგვიძლია შევინახოთ სახელების სხვა ნაკრებებისაგან ცალკე
ბ. შეგვიძლია შევინახოთ სახელების სხვა ნაკრებებისაგან ცალკე
გ. შეგვიძლია შევინახოთ სახელების სხვა ნაკრებებთან ერთად

9.9.3. სახელების ერთ სივრცეში გამოცხადებული სახელები:
ა. არ არის კონფლიქტში სახელების სხვა სივრცეში გამოცხადებულ ასეთივე სახელებთან
ბ. არის კონფლიქტში სახელების სხვა სივრცეში გამოცხადებულ ასეთივე სახელებთან
გ. არ არის კონფლიქტში სახელების ამავე სივრცეში გამოცხადებულ ასეთივე სახელებთან

9.9.4. სახელების სივრცეში განსაზღვრული ყველა წევრი:
ა. ნაწილობრივ იმყოფება ამ სახელების სივრცის მხედველობის უბნის საზღვრებში
ბ. არ იმყოფება ამ სახელების სივრცის მხედველობის უბნის საზღვრებში
გ. იმყოფება ამ სახელების სივრცის მხედველობის უბნის საზღვრებში

9.9.5. სახელების სივრცეში შეიძლება განსაზღვრული იყოს:
ა. ჩამოთვლები და მასივები
ბ. კლასები და დელეგატები
გ. ინტერფეისები და ცვლადები

9.9.6. თუ ორი სახელების სივრცე შეიცავს ერთნაირი სახელის მქონე კლასს, მაშინ იმისათვის, რომ ეს ორი კლასი ერთმანეთისაგან განვასხვავოთ:
ა. საჭირო არ არის კლასის სახელის წინ სახელების სივრცის მითითება
ბ. საჭიროა კლასის სახელის წინ სახელების სივრცის მითითება
გ. საჭიროა მხოლოდ კლასის სახელის მითითება

using დირექტივა

9.10.7. სახელების სივრცე პროგრამაში ხილული ხდება:
ა. using დირექტივაში მისი მითითების შემდეგ

ბ. მასივში მისი გამოცხადების შემდეგ
გ. public მოდიფიკატორის მითითების შემდეგ

9.10.8. using დირექტივა მითითებული უნდა იყოს:
ა. პროგრამის შუაში
ბ. პროგრამის ბოლოში
გ. პროგრამის დასაწყისში

9.10.9. თუ პროგრამაში გამოცხადებულია ორი სახელების სივრცე, მაშინ:
ა. სახელების პირველი სივრცე არ მალავს სახელების მეორე სივრცეს
ბ. სახელების პირველი სივრცე მალავს სახელების მეორე სივრცეს
გ. სახელების მეორე სივრცე მალავს სახელების პირველი სივრცეს

9.10.10. სახელების სივრცის გამოცხადებისას იგი:
ა. თავის სახელებს არ უმატებს უკვე არსებულ სახელებს
ბ. თავის სახელებს უმატებს უკვე არსებულ სახელებს
გ. ფარავს უკვე არსებულ სახელებს

ჩადგმული სახელების სივრცე

9.11.11. სახელების ერთი სივრცე:
ა. შეიძლება ნაწილობრივ მოვათავსოთ სახელების მეორე სივრცის შიგნით
ბ. არ შეიძლება მოვათავსოთ სახელების მეორე სივრცის შიგნით
გ. შეიძლება მოვათავსოთ სახელების მეორე სივრცის შიგნით

9.11.12. სახელების სივრცის იერარქია:
ა. შეიძლება გადანაწილებული იყოს სხვადასხვა პროგრამაში
ბ. არ შეიძლება გადანაწილებული იყოს სხვადასხვა პროგრამაში
გ. ნაწილობრივ შეიძლება გადანაწილებული იყოს სხვადასხვა პროგრამაში

ავტომატურად განსაზღვრული სახელების სივრცე

9.12.13. თუ პროგრამაში არ არის გამოცხადებული სახელების სივრცე, მაშინ:
ა. არ გამოიყენება ავტომატურად განსაზღვრული სახელების სივრცე
ბ. გამოიყენება ავტომატურად განსაზღვრული სახელების სივრცე
გ. არც ერთი სახელების სივრცე არ გამოიყენება

9.12.14. ავტომატურად განსაზღვრული სახელების სივრცე:
ა. არასოდეს არ გამოიყენება
ბ. გამოიყენება დიდი და რთული პროგრამებისათვის
გ. გამოიყენება მოკლე და მარტივი პროგრამებისათვის

თავი 10. ინფორმაციის შეტანა-გამოტანა

შეტანა-გამოტანის ნაკადების კლასები

10.1.1. C#-პროგრამები მონაცემების შეტანა-გამოტანას ახდენს:
ა. ნაკადების საშუალებით
ბ. ინდექსატორების საშუალებით
გ. მოვლენების საშუალებით

10.1.2. მონაცემების ნაკადი არის:
ა. ფაილების მასივი
ბ. თვისებების მასივი
გ. მონაცემების მიმდევრობა

10.1.3. ნაკადი ფიზიკურ მოწყობილობას უკავშირდება:
ა. ვირტუალური მეთოდების გამოყენებით
ბ. სტატიკური მეთოდების გამოყენებით
გ. შეტანა-გამოტანის სისტემის საშუალებით

10.1.4. შეტანა-გამოტანის ოპერაციები ყველაზე დაბალ დონეზე ოპერირებენ:
ა. სტრიქონებთან
ბ. ბაიტებთან
გ. წილადებთან

10.1.5. შეტანა-გამოტანის ოპერაციები ყველაზე დაბალ დონეზე ბაიტებთან ოპერირებს, რადგან:
ა. შეტანა-გამოტანის მოწყობილობების უმრავლესობა არის ბაიტ-ორიენტირებული
ბ. შეტანა-გამოტანის მოწყობილობების უმრავლესობა არის სიმბოლოზე ორიენტირებული
გ. შეტანა-გამოტანის მოწყობილობების უმრავლესობა არის მთელ რიცხვებზე ორიენტირებული

10.1.6. ნაკადების კლასები განსაზღვრულია:
ა. System.Data კლასში
ბ. System.IO კლასში
გ. System.Drawing კლასში

10.1.7. რომელი მსჯელობაა არასწორი:
ა. არსებობს სიმბოლოების object ნაკრები
ბ. არსებობს სიმბოლოების ASCII ნაკრები
გ. არსებობს სიმბოლოების Unicode ნაკრები

10.1.8. ASCII სიმბოლოს ზომაა:
ა. 1 ბაიტი
ბ. 2 ბაიტი
გ. 4 ბაიტი

10.1.9. Unicode სიმბოლოს ზომაა:
ა. 1 ბაიტი

ბ. 2 ბაიტი
გ. 4 ბაიტი

ბაიტების ნაკადები

10.1.10. C# ენაში ნაკადები იქმნება:
ა. System.Net კლასის საშუალებით
ბ. System კლასის საშუალებით
გ. Stream კლასის საშუალებით

10.1.11. Stream კლასი არის:
ა. სტატიკური
ბ. ვირტუალური
გ. აბსტრაქტული

10.1.12. Stream კლასის Close() მეთოდი:
ა. ნაკადს ხურავს
ბ. ნაკადს ხსნის
გ. ნაკადს ცვლის

10.1.13. Stream კლასის Flush() მეთოდი:
ა. ნაკადს ფაილში წერს
ბ. ნაკადს ფაილიდან კითხულობს
გ. ნაკადს ხსნის

10.1.14. Stream კლასის ReadByte() მეთოდი:
ა. შესასვლელი ნაკადიდან კითხულობს 2 ბაიტს და გასცემს მის მთელრიცხვა მნიშვნელობას
ბ. შესასვლელი ნაკადიდან კითხულობს 1 ბაიტს და გასცემს მის მთელრიცხვა მნიშვნელობას
გ. შესასვლელი ნაკადიდან კითხულობს 1 ბაიტს და გასცემს მის ლოგიკურ მნიშვნელობას

10.1.15. Stream კლასის Read(byte[] ბუფერი, int პოზიცია, int ბაიტების_რაოდენობა) მეთოდი:
ა. შესასვლელ ნაკადში წერს მითითებული რაოდენობის ბაიტს და მათ ათავსებს ბუფერში დაწყებული მითითებული პოზიციიდან
ბ. შესასვლელ ნაკადში ცვლის მითითებული რაოდენობის ბაიტს
გ. შესასვლელი ნაკადიდან კითხულობს მითითებული რაოდენობის ბაიტს და მათ ათავსებს ბუფერში დაწყებული მითითებული პოზიციიდან

10.1.16. Stream კლასის Seek() მეთოდი:
ა. გასცემს ნაკადის პირველი ბაიტის პოზიციას
ბ. გასცემს ნაკადის მიმდინარე ბაიტის პოზიციას
გ. გასცემს ნაკადის უკანასკნელი ბაიტის პოზიციას

10.1.17. Stream კლასის WriteByte(byte ცვლადი) მეთოდი:
ა. გამოსასვლელ ნაკადში 1 ბაიტს წერს
ბ. გამოსასვლელ ნაკადში 2 ბაიტს წერს
გ. შესასვლელი ნაკადიდან 1 ბაიტს კითხულობს

10.1.18. Stream კლასის Write(byte[ ] ბუფერი, int პოზიცია, int ბაიტების_რაოდენობა) მეთოდი:
ა. გამოსასვლელ ნაკადში წერს მითითებული რაოდენობის ბაიტს დაწყებული მითითებული პოზიციიდან
ბ. გამოსასვლელ ნაკადში წერს მხოლოდ წილად რიცხვებს
გ. შესასვლელი ნაკადიდან კითხულობს მითითებული რაოდენობის ბაიტს

10.1.19. Stream კლასის CanRead თვისება იღებს:
ა. true მნიშვნელობას, თუ შეუძლებელია ნაკადიდან წაკითხვა
ბ. false მნიშვნელობას, თუ შესაძლებელია ნაკადიდან წაკითხვა
გ. true მნიშვნელობას, თუ შესაძლებელია ნაკადიდან წაკითხვა

10.1.20. Stream კლასის CanSeek თვისება იღებს:
ა. false მნიშვნელობას, თუ შესაძლებელია ნაკადის ელემენტის მიმდინარე პოზიციის განსაზღვრა
ბ. true მნიშვნელობას, თუ შესაძლებელია ნაკადის ელემენტის მიმდინარე პოზიციის განსაზღვრა
გ. true მნიშვნელობას, თუ შეუძლებელია ნაკადის ელემენტის მიმდინარე პოზიციის განსაზღვრა

10.1.21. Stream კლასის CanWrite თვისება იღებს:
ა. true მნიშვნელობას, თუ შესაძლებელია ნაკადში ჩაწერა
ბ. false მნიშვნელობას, თუ შესაძლებელია ნაკადში ჩაწერა
გ. true მნიშვნელობას, თუ შეუძლებელია ნაკადში ჩაწერა

10.1.22. Stream კლასის Length თვისება განსაზღვრავს:
ა. ნაკადის სიგრძეს
ბ. ნაკადის მიმდინარე ელემენტის პოზიციას
გ. ნაკადში შეიძლება თუ არა მონაცემების ჩაწერა

10.1.23. Stream კლასის Position თვისება განსაზღვრავს:
ა. ნაკადში შეიძლება თუ არა მონაცემების წაკითხვა
ბ. ნაკადის მიმდინარე ელემენტის პოზიციას
გ. ნაკადის სიგრძეს

10.1.24. Stream კლასის მემკვიდრე FileStream კლასი განკუთვნილია:
ა. ფაილში მონაცემების მხოლოდ წასაკითხად
ბ. ფაილში მონაცემების მხოლოდ ჩასაწერად
გ. ფაილში შეტანა-გამოტანის ოპერაციების შესასრულებლად

სიმბოლოების ნაკადები

10.1.25. სიმბოლოების ნაკადების კლასებია:
ა. FileStream
ბ. TextReader და TextWriter
გ. BufferedStream

10.1.26. ბაიტების ნაკადების კლასებია:
ა. TextWriter და TextReader
ბ. FileStream და MemoryStream
გ. BufferedStream და TextWriter

10.1.27. TextReader კლასის Close() მეთოდი:
ა. ცვლის შესასვლელ ნაკადს
ბ. ხსნის შესასვლელ ნაკადს
გ. ხურავს შესასვლელ ნაკადს

10.1.28. TextReader კლასის Peek() მეთოდი:
ა. კითხულობს შესასვლელი ნაკადის მომდევნო სიმბოლოს და არ შლის მას
ბ. კითხულობს შესასვლელი ნაკადის მომდევნო სიმბოლოს და შლის მას
გ. წერს შესასვლელ ნაკადში მომდევნო სიმბოლოს და არ შლის მას

10.1.29. TextReader კლასის Peek() მეთოდი:
ა. გასცემს -2-ს, თუ არ არის მომდევნო წასაკითხი სიმბოლო
ბ. გასცემს -1-ს, თუ არ არის მომდევნო წასაკითხი სიმბოლო
გ. გასცემს -1-ს, თუ არის მომდევნო წასაკითხი სიმბოლო

10.1.30. TextReader კლასის Read() მეთოდი:
ა. წერს შესასვლელი ნაკადის მომდევნო სიმბოლოს და გასცემს მის მთელრიცხვა წარმოდგენას
ბ. კითხულობს შესასვლელი ნაკადის მომდევნო სიმბოლოს და გასცემს მის მთელრიცხვა წარმოდგენას
გ. კითხულობს შესასვლელი ნაკადის მომდევნო სიმბოლოს და გასცემს მის წილად წარმოდგენას

10.1.31. TextReader კლასის Read(char[ ] ბუფერი, int პოზიცია,
int სიმბოლოების_რაოდენობა) მეთოდი:
ა. შესასვლელი ნაკადიდან კითხულობს 10 სიმბოლოს და ათავსებს მათ ბუფერში დაწყებული პირველი პოზიციიდან
ბ. შესასვლელ ნაკადში წერს მითითებული რაოდენობის სიმბოლოს და ათავსებს მათ ბუფერში დაწყებული მითითებული პოზიციიდან
გ. შესასვლელი ნაკადიდან კითხულობს მითითებული რაოდენობის სიმბოლოს და ათავსებს მათ ბუფერში დაწყებული მითითებული პოზიციიდან

10.1.32. TextReader კლასის ReadBlock(char[ ] ბუფერი, int პოზიცია,
int სიმბოლოების_რაოდენობა) მეთოდი:
ა. შესასვლელი ნაკადიდან კითხულობს 10 სიმბოლოს და ათავსებს მათ ბუფერში დაწყებული პირველი პოზიციიდან
ბ. შესასვლელ ნაკადში წერს მითითებული რაოდენობის სიმბოლოს და ათავსებს მათ ბუფერში დაწყებული მითითებული პოზიციიდან
გ. შესასვლელი ნაკადიდან კითხულობს მითითებული რაოდენობის სიმბოლოს და ათავსებს მათ ბუფერში დაწყებული მითითებული პოზიციიდან

10.1.33. TextReader კლასის ReadLine() მეთოდი:
ა. კითხულობს ტექსტის სტრიქონს და გასცემს მას C#-სტრიქონის სახით

ბ. წერს ტექსტის სტრიქონს და გასცემს მას C#-სტრიქონის სახით
გ. კითხულობს ტექსტის სტრიქონს და გასცემს მის მთელრიცხვა წარმოდგენას

10.1.34. TextReader კლასის ReadToEnd() მეთოდი:
ა. კითხულობს ნაკადის დარჩენილ სიმბოლოებს და გასცემს მათ C#-სტრიქონის სახით
ბ. წერს ნაკადის დარჩენილ სიმბოლოებს და გასცემს მათ C#-სტრიქონის სახით
გ. კითხულობს ნაკადის დარჩენილ სიმბოლოებს და გასცემს მათ წილადის სახით

10.1.35. TextWriter კლასის Write(int ცვლადი) მეთოდი ახდენს:
ა. int ტიპის მნიშვნელობის წაკითხვას
ბ. double ტიპის მნიშვნელობის ჩაწერას
გ. int ტიპის მნიშვნელობის ჩაწერას

10.1.36. TextWriter კლასის Write(double ცვლადი) მეთოდი ახდენს:
ა. int ტიპის მნიშვნელობის ჩაწერას
ბ. double ტიპის მნიშვნელობის ჩაწერას
გ. double ტიპის მნიშვნელობის წაკითხვას

10.1.37. TextWriter კლასის Write(bool ცვლადი) მეთოდი ახდენს:
ა. int ტიპის მნიშვნელობის ჩაწერას
ბ. bool ტიპის მნიშვნელობის ჩაწერას
გ. bool ტიპის მნიშვნელობის წაკითხვას

10.1.38. TextWriter კლასის WriteLine(string ცვლადი) მეთოდი ახდენს:
ა. სტრიქონის ჩაწერას, რომელიც არ მთავრდება მომდევნო სტრიქონზე გადასვლის სიმბოლოთი
ბ. სტრიქონის ჩაწერას, რომელიც მთავრდება მომდევნო სტრიქონზე გადასვლის სიმბოლოთი
გ. სტრიქონის წაკითხვას, რომელიც მთავრდება მომდევნო სტრიქონზე გადასვლის სიმბოლოთი

10.1.39. TextWriter კლასის WriteLine(uint ცვლადი) მეთოდი ახდენს:
ა. uint ტიპის მნიშვნელობისა და მომდევნო სტრიქონზე გადასვლის სიმბოლოს ჩაწერას
ბ. uint ტიპის მნიშვნელობისა და მომდევნო სტრიქონზე გადასვლის სიმბოლოს წაკითხვას
გ. uint ტიპის მნიშვნელობის ჩაწერას

10.1.40. TextWriter კლასის WriteLine(char ცვლადი) მეთოდი ახდენს:
ა. სტრიქონის ჩაწერას, რომელსაც მოსდევს მომდევნო სტრიქონზე გადასვლის სიმბოლო
ბ. სიმბოლოს წაკითხვას, რომელსაც მოსდევს მომდევნო სტრიქონზე გადასვლის სიმბოლო
გ. სიმბოლოს ჩაწერას, რომელსაც მოსდევს მომდევნო სტრიქონზე გადასვლის სიმბოლო

10.1.41. TextWriter კლასის Close() მეთოდი:
ა. ცვლის ნაკადს
ბ. ხსნის ნაკადს
გ. ხურავს ნაკადს

10.1.42. TextWriter კლასის Flush() მეთოდი ახდენს:
ა. ბუფერში დარჩენილი მონაცემების ჩაწერას გამოსასვლელ ნაკადში

ბ. ბუფერში დარჩენილი მონაცემების წაკითხვას გამოსასვლელ ნაკადში
გ. ნაკადის დახურვას

ორობითი ნაკადები

10.1.43. ორობითი ნაკადების კლასებია:
ა. FileStream
ბ. BinaryReader და BinaryWriter
გ. TextReader და TextWriter

FileStream კლასი

ფაილის გახსნა და დახურვა

10.2.1. FileStream კლასი გამოიყენება:
ა. ფაილებში ბაიტების ჩაწერა-წაკითხვის ოპერაციების შესასრულებლად
ბ. ფაილებში სიმბოლოების ჩაწერა-წაკითხვის ოპერაციების შესასრულებლად
გ. ფაილში მხოლოდ სიმბოლოების ჩასაწერად

10.2.2. ფაილთან დაკავშირებული ბაიტების ნაკადის შესაქმნელად საჭიროა:
ა. MemoryStream ობიექტის ფორმირება
ბ. Object ობიექტის ფორმირება
გ. FileStream ობიექტის ფორმირება

10.2.3. FileMode რეჟიმი განსაზღვრავს:
ა. ფაილში ჩაწერის რეჟიმს
ბ. ფაილის გახსნის რეჟიმს
გ. ფაილიდან წაკითხვის რეჟიმს

10.2.4. FileMode რეჟიმის Append მნიშვნელობა მიუთითებს, რომ ხდება:
ა. მონაცემების წაკითხვა
ბ. გამოსასვლელი მონაცემების დამატება ფაილის ბოლოში
გ. მონაცემების ჩაწერა ფაილის დასაწყისში

10.2.5. FileMode რეჟიმის Create მნიშვნელობა მიუთითებს, რომ:
ა. იქმნება ახალი გამოსასვლელი ფაილი და ამავე სახელის მქონე არსებული ფაილი იშლება
ბ. იქმნება ახალი გამოსასვლელი ფაილი და ამავე სახელის მქონე არსებული ფაილი არ იშლება
გ. იხსნება არსებული ფაილი

10.2.6. FileMode რეჟიმის CreateNew მნიშვნელობა მიუთითებს, რომ:
ა. იქმნება ახალი გამოსასვლელი ფაილი
ბ. იშლება არსებული ფაილი
გ. იხსნება არსებული ფაილი

10.2.7. FileMode რეჟიმის Open მნიშვნელობა მიუთითებს, რომ:
ა. იხურება არსებული ფაილი

ბ. იშლება არსებული ფაილი
გ. იხსნება არსებული ფაილი

10.2.8. FileMode რეჟიმის OpenCreate მნიშვნელობა მიუთითებს, რომ:
ა. თუ ფაილი არსებობს, მაშინ ის გაიხსნება, თუ არ არსებობს, მაშინ ის შეიქმნება
ბ. თუ ფაილი არსებობს, მაშინ ის წაიშლება
გ. თუ ფაილი არსებობს, მაშინ ის შეიქმნება

10.2.9. FileMode რეჟიმის Truncate მნიშვნელობა მიუთითებს, რომ:
ა. იხსნება არსებული ფაილი, მაგრამ მისი სიგრძე არ ჩამოიჭრება ნულამდე
ბ. იქმნება ახალი ფაილი
გ. იხსნება არსებული ფაილი, მაგრამ მისი სიგრძე ჩამოიჭრება ნულამდე

10.2.10. FileStraem file_in = new FileStream("filetest.dat", FileMode.Open);
ოპერატორის შესრულების შედეგად:
ა. წაიშლება filetest.dat ფაილი
ბ. გაიხსნება filetest.dat ფაილი
გ. შეიცვლება filetest.dat ფაილი

10.2.11. FileAccess მიმართვის სახე იღებს მნიშვნელობას:
ა. Read, Write და ReadWrite
ბ. Read, Open
გ. Close, Open და Write

10.2.12. FileStraem file_in = new FileStream("filetest.dat", FileMode.Open,
FileAccess.Read); ოპერატორის შესრულების შედეგად:
ა. filetest.dat ფაილი გაიხსნება ჩაწერის რეჟიმში
ბ. filetest.dat ფაილი გაიხსნება დამატების რეჟიმში
გ. filetest.dat ფაილი გაიხსნება წაკითხვის რეჟიმში

ფაილიდან ბაიტების წაკითხვა

10.2.13. FileStraem კლასის ReadByte() მეთოდი:
ა. ფაილში წერს ერთ ბაიტს და გასცემს მის მთელრიცხვა მნიშვნელობას
ბ. ფაილიდან კითხულობს ერთ ბაიტს და გასცემს მის მთელრიცხვა მნიშვნელობას
გ. ფაილიდან კითხულობს ერთ ბაიტს და არ გასცემს მის მთელრიცხვა მნიშვნელობას

10.2.14. FileStraem კლასის Read(byte[ ] ბუფერი, int პოზიცია,
int ბაიტების_რაოდენობა) მეთოდი:
ა. ფაილიდან კითხულობს მითითებული რაოდენობის ბაიტს და ათავსებს მათ ბუფერში
ბ. ფაილში წერს მითითებული რაოდენობის ბაიტს და ათავსებს მათ ბუფერში
გ. ფაილიდან კითხულობს მითითებული რაოდენობის ბაიტს და არ ათავსებს მათ ბუფერში

10.2.15. FileStraem კლასის ReadByte() მეთოდი ფაილის დასასრულის სიმბოლოს წაკითხვის შემთხვევაში გასცემს:
ა. -1

ბ. 0
გ. 1

10.2.16. FileStraem კლასის Read(byte[ ] ბუფერი, int პოზიცია,
int ბაიტების_რაოდენობა) მეთოდის შესრულების შედეგად გაიცემა:
ა. წასაკითხი ბაიტების მითითებული რაოდენობა
ბ. რეალურად ჩაწერილი ბაიტების რაოდენობა
გ. რეალურად წაკითხული ბაიტების რაოდენობა

ფაილში ბაიტების ჩაწერა

10.2.17. FileStraem კლასის WriteByte() მეთოდი:
ა. ფაილში წერს სიმბოლოს
ბ. ფაილში წერს 1 ბაიტს
გ. ფაილში წერს 2 ბაიტს

10.2.18. FileStraem კლასის Write(byte[ ] ბუფერი, int პოზიცია,
int ბაიტების_რაოდენობა) მეთოდი:
ა. ბუფერიდან ფაილში წერს მხოლოდ ერთ ბაიტს
ბ. ბუფერიდან ფაილში კითხულობს მითითებული რაოდენობის ბაიტს
გ. ბუფერიდან ფაილში წერს მითითებული რაოდენობის ბაიტს

10.2.19. FileStraem კლასის Write(byte[ ] ბუფერი, int პოზიცია,
int ბაიტების_რაოდენობა) მეთოდის შესრულების შედეგად გაიცემა:
ა. რეალურად ჩაწერილი ბაიტების რაოდენობა
ბ. რეალურად წაკითხული ბაიტების რაოდენობა
გ. ჩასაწერი ბაიტების მითითებული რაოდენობა

10.2.20. ფაილში მონაცემები იწერება მაშინ, როცა ბუფერი:
ა. სავსეა
ბ. ცარიელია
გ. ნახევრად სავსეა

10.2.21. FileStraem კლასის Flush() მეთოდი ახდენს ფაილში ჩაწერას, როცა:
ა. ბუფერი ცარიელია
ბ. ბუფერი სავსეა
გ. ბუფერში მხოლოდ სიმბოლოებია

10.2.22. FileStraem კლასის Close() მეთოდი:
ა. ბუფერიდან მონაცემებს არ წერს ფაილში და ხურავს ფაილს
ბ. ბუფერიდან მონაცემებს წერს ფაილში და ხსნის ფაილს
გ. ბუფერიდან მონაცემებს წერს ფაილში და ხურავს ფაილს

ფაილში სიმბოლოების შეტანა-გამოტანა

10.3.1. სიმბოლურ ფაილებთან მუშაობისას FileStream ობიექტი უნდა ჩავრთოთ:
ა. object კლასის შემადგენლობაში

ბ. StreamReader ან StreamWriter კლასის შემადგენლობაში
გ. MemoryStream კლასის შემადგენლობაში

10.3.2. FileStream file_out = new FileStream("F1.txt", FileMode.Create);
StreamWriter file_strem_out = new StreamWriter(file_out);
ოპერატორების შესრულების შედეგად:
ა. იქმნება F1.txt ფაილი სიმბოლოების ჩასაწერად
ბ. იხსნება F1.txt ფაილი სიმბოლოების ჩასაწერად
გ. იშლება F1.txt ფაილი

10.3.3. FileStream file_in = new FileStream("F1.txt", FileMode.Open);
StreamReader file_strem_in = new StreamReader(file_in);
ოპერატორების შესრულების შედეგად:
ა. იქმნება F1.txt ფაილი სიმბოლოების წასაკითხად
ბ. იხსნება F1.txt ფაილი სიმბოლოების ჩასაწერად
გ. იხსნება F1.txt ფაილი სიმბოლოების წასაკითხად

ორობითი მონაცემების შეტანა-გამოტანა

BinaryWriter კლასი

10.4.1. ფაილში ორობითი მონაცემების ჩასაწერად გამოიყენება:
ა. TextWriter კლასი
ბ. BinaryReader კლასი
გ. BinaryWriter კლასი

10.4.2. ფაილში ორობითი მონაცემების წასაკითხად გამოიყენება:
ა. BinaryReader კლასი
ბ. BinaryWriter კლასი
გ. TextReader კლასი

10.4.3. BinaryWriter კლასის Write(sbyte ცვლადი) მეთოდი ახდენს ფაილში:
ა. ნიშნიანი ბაიტის წაკითხვას
ბ. ნიშნიანი ბაიტის ჩაწერას
გ. უნიშნო ბაიტის ჩაწერას

10.4.4. BinaryWriter კლასის Write(byte ცვლადი) მეთოდი ახდენს ფაილში:
ა. ნიშნიანი ბაიტის ჩაწერას
ბ. უნიშნო ბაიტის ჩაწერას
გ. უნიშნო ბაიტის წაკითხვას

10.4.5. BinaryWriter კლასის Write(byte[ ] ბუფერი) მეთოდი ახდენს ფაილში:
ა. მხოლოდ 1 ბაიტის ჩაწერას
ბ. ბაიტების მასივის წაკითხვას
გ. ბაიტების მასივის ჩაწერას

10.4.6. BinaryWriter კლასის Write(short ცვლადი) მეთოდი ახდენს ფაილში:
ა. მოკლე მთელი რიცხვის ჩაწერას

ბ. გრძელი მთელი რიცხვის ჩაწერას
გ. მოკლე მთელი რიცხვის წაკითხვას

10.4.7. BinaryWriter კლასის Write(ushort ცვლადი) მეთოდი ახდენს ფაილში:
ა. ნიშნიანი მთელი რიცხვის ჩაწერას
ბ. უნიშნო მთელი რიცხვის ჩაწერას
გ. უნიშნო მთელი რიცხვის წაკითხვას

10.4.8. BinaryWriter კლასის Write(int ცვლადი) მეთოდი ახდენს ფაილში:
ა. უნიშნო მთელი რიცხვის ჩაწერას
ბ. მთელი რიცხვის წაკითხვას
გ. მთელი რიცხვის ჩაწერას

10.4.9. BinaryWriter კლასის Write(uint ცვლადი) მეთოდი ახდენს ფაილში:
ა. უნიშნო მთელი რიცხვის წაკითხვას
ბ. უნიშნო მთელი რიცხვის ჩაწერას
გ. ნიშნიანი მთელი რიცხვის ჩაწერას

10.4.10. BinaryWriter კლასის Write(long ცვლადი) მეთოდი ახდენს ფაილში:
ა. გრძელი მთელი რიცხვის ჩაწერას
ბ. გრძელი მთელი რიცხვის წაკითხვას
გ. მოკლე მთელი რიცხვის ჩაწერას

10.4.11. BinaryWriter კლასის Write(ulong ცვლადი) მეთოდი ახდენს ფაილში:
ა. უნიშნო გრძელი მთელი რიცხვის ჩაწერას
ბ. უნიშნო გრძელი მთელი რიცხვის წაკითხვას
გ. ნიშნიანი გრძელი მთელი რიცხვის ჩაწერას

10.4.12. BinaryWriter კლასის Write(float ცვლადი) მეთოდი ახდენს ფაილში:
ა. ორმაგი სიზუსტის წილადის ჩაწერას
ბ. ერთმაგი სიზუსტის წილადის წაკითხვას
გ. ერთმაგი სიზუსტის წილადის ჩაწერას

10.4.13. BinaryWriter კლასის Write(double ცვლადი) მეთოდი ახდენს ფაილში:
ა. ერთმაგი სიზუსტის წილადის ჩაწერას
ბ. ორმაგი სიზუსტის წილადის ჩაწერას
გ. ორმაგი სიზუსტის წილადის წაკითხვას

10.4.14. BinaryWriter კლასის Write(char ცვლადი) მეთოდი ახდენს ფაილში:
ა. სიმბოლოს ჩაწერას
ბ. სიმბოლოს წაკითხვას
გ. სტრიქონის ჩაწერას

10.4.15. BinaryWriter კლასის Write(char[ ] მასივი) მეთოდი ახდენს ფაილში:
ა. სტრიქონების მასივის ჩაწერას
ბ. სიმბოლოების მასივის წაკითხვას
გ. სიმბოლოების მასივის ჩაწერას

10.4.16. BinaryWriter კლასის Write(string ცვლადი) მეთოდი ახდენს ფაილში:
ა. სტრიქონის წაკითხვას
ბ. სტრიქონის ჩაწერას
გ. სიმბოლოების მასივის წაკითხვას

BinaryReader ნაკადი

10.4.17. BinaryReader კლასის Read() მეთოდი ფაილიდან:
ა. კითხულობს შესასვლელი ნაკადის სიმბოლოს და გასცემს მის მთელრიცხვა მნიშვნელობას
ბ. კითხულობს შესასვლელი ნაკადის სიმბოლოს და არ გასცემს მის მთელრიცხვა მნიშვნელობას
გ. შესასვლელ ნაკადში წერს სიმბოლოს და გასცემს მის მთელრიცხვა მნიშვნელობას

10.4.18. BinaryReader კლასის Read(byte[ ] ბუფერი, int პოზიცია,
int ბაიტების_რაოდენობა) მეთოდი:
ა. ფაილიდან კითხულობს მხოლოდ 1 ბაიტს
ბ. ფაილში წერს მითითებული რაოდენობის ბაიტს ბუფერიდან
გ. ფაილიდან კითხულობს მითითებული რაოდენობის ბაიტს და ათავსებს მათ ბუფერში

10.4.19. BinaryReader კლასის Read(char[ ] ბუფერი, int პოზიცია,
int ბაიტების_რაოდენობა) მეთოდი:
ა. ფაილში წერს მითითებული რაოდენობის სიმბოლოს ბუფერიდან
ბ. ფაილიდან კითხულობს მითითებული რაოდენობის სიმბოლოს და ათავსებს მათ ბუფერში
გ. ფაილიდან კითხულობს მხოლოდ 1 სიმბოლოს

10.4.20. BinaryReader კლასის ReadBoolean() მეთოდი ახდენს:
ა. ლოგიკური მონაცემის წაკითხვას
ბ. ლოგიკური მონაცემის ჩაწერას
გ. მთელი რიცხვის წაკითხვას

10.4.21. BinaryReader კლასის ReadByte() მეთოდი ახდენს:
ა. ნიშნიანი ბაიტის წაკითხვას
ბ. უნიშნო ბაიტის წაკითხვას
გ. უნიშნო ბაიტის ჩაწერას

10.4.22. BinaryReader კლასის ReadSByte() მეთოდი ახდენს:
ა. ნიშნიანი ბაიტის წაკითხვას
ბ. უნიშნო ბაიტის წაკითხვას
გ. ნიშნიანი ბაიტის ჩაწერას

10.4.23. BinaryReader კლასის ReadBytes(int ბაიტების_რაოდენობა) მეთოდი ახდენს:
ა. მხოლოდ 1 ბაიტის წაკითხვას
ბ. მითითებული რაოდენობის ბაიტების ჩაწერას
გ. მითითებული რაოდენობის ბაიტების წაკითხვას

10.4.24. BinaryReader კლასის ReadChar() მეთოდი ახდენს:
ა. სიმბოლოს წაკითხვას
ბ. სიმბოლოს ჩაწერას
გ. ბაიტის წაკითხვას

10.4.25. BinaryReader კლასის ReadChars(int სიმბოლოების_რაოდენობა) მეთოდი ახდენს:
ა. მითითებული რაოდენობის სიმბოლოების ჩაწერას
ბ. მითითებული რაოდენობის სიმბოლოების წაკითხვას
გ. მხოლოდ 1 სიმბოლოების წაკითხვას

10.4.26. BinaryReader კლასის ReadDouble() მეთოდი ახდენს:
ა. ერთმაგი სიზუსტის წილადის წაკითხვას
ბ. ორმაგი სიზუსტის წილადის ჩაწერას
გ. ორმაგი სიზუსტის წილადის წაკითხვას

10.4.27. BinaryReader კლასის ReadSingle() მეთოდი ახდენს:
ა. ერთმაგი სიზუსტის წილადის წაკითხვას
ბ. ორმაგი სიზუსტის წილადის წაკითხვას
გ. ერთმაგი სიზუსტის წილადის ჩაწერას

10.4.28. BinaryReader კლასის ReadInt16() მეთოდი ახდენს:
ა. მოკლე მთელი რიცხვის წაკითხვას
ბ. მოკლე მთელი რიცხვის ჩაწერას
გ. გრძელი მთელი რიცხვის წაკითხვას

10.4.29. BinaryReader კლასის ReadInt32() მეთოდი ახდენს:
ა. მთელი რიცხვის წაკითხვას
ბ. მთელი რიცხვის ჩაწერას
გ. გრძელი მთელი რიცხვის წაკითხვას

10.4.30. BinaryReader კლასის ReadInt64() მეთოდი ახდენს:
ა. გრძელი მთელი რიცხვის წაკითხვას
ბ. გრძელი მთელი რიცხვის ჩაწერას
გ. მთელი რიცხვის წაკითხვას

10.4.31. BinaryReader კლასის ReadUInt16() მეთოდი ახდენს:
ა. უნიშნო მოკლე მთელი რიცხვის წაკითხვას
ბ. უნიშნო მოკლე მთელი რიცხვის ჩაწერას
გ. ნიშნიანი მოკლე მთელი რიცხვის წაკითხვას

10.4.32. BinaryReader კლასის ReadUInt32() მეთოდი ახდენს:
ა. უნიშნო მთელი რიცხვის ჩაწერას
ბ. უნიშნო მთელი რიცხვის წაკითხვას
გ. უნიშნო მოკლე მთელი რიცხვის წაკითხვას

10.4.33. BinaryReader კლასის ReadUInt64() მეთოდი ახდენს:
ა. უნიშნო გრძელი მთელი რიცხვის ჩაწერას

ბ. გრძელი მთელი რიცხვის წაკითხვას
გ. უნიშნო გრძელი მთელი რიცხვის წაკითხვას

10.4.34. BinaryReader კლასის ReadString() მეთოდი ახდენს:
ა. სტრიქონის ჩაწერას
ბ. სტრიქონის წაკითხვას
გ. სიმბოლოს ჩაწერას

10.4.35. ricxvi = file_in.ReadInt32(); ოპერატორის შესრულების შედეგად:
ა. ricxvi ცვლადს მიენიჭება მოკლე მთელი რიცხვი
ბ. ricxvi ცვლადს მიენიჭება წილადი
გ. ricxvi ცვლადს მიენიჭება მთელი რიცხვი

10.4.36. ricxvi = file_in.ReadInt16(); ოპერატორის შესრულების შედეგად:
ა. ricxvi ცვლადს მიენიჭება წილადი
ბ. ricxvi ცვლადს მიენიჭება მთელი რიცხვი
გ. ricxvi ცვლადს მიენიჭება მოკლე მთელი რიცხვი

10.4.37. ricxvi = file_in.ReadInt64(); ოპერატორის შესრულების შედეგად:
ა. ricxvi ცვლადს მიენიჭება გრძელი მთელი რიცხვი
ბ. ricxvi ცვლადს მიენიჭება მთელი რიცხვი
გ. ricxvi ცვლადს მიენიჭება მოკლე მთელი რიცხვი

10.4.38. ricxvi = file_in.ReadUInt64(); ოპერატორის შესრულების შედეგად:
ა. ricxvi ცვლადს მიენიჭება უნიშნო გრძელი მთელი რიცხვი
ბ. ricxvi ცვლადს მიენიჭება გრძელი მთელი რიცხვი
გ. ricxvi ცვლადს მიენიჭება მოკლე მთელი რიცხვი

10.4.39. ricxvi = file_in.ReadUInt32(); ოპერატორის შესრულების შედეგად:
ა. ricxvi ცვლადს მიენიჭება უნიშნო გრძელი მთელი რიცხვი
ბ. ricxvi ცვლადს მიენიჭება უნიშნო მთელი რიცხვი
გ. ricxvi ცვლადს მიენიჭება მოკლე მთელი რიცხვი

10.4.40. ricxvi = file_in.ReadUInt16(); ოპერატორის შესრულების შედეგად:
ა. ricxvi ცვლადს მიენიჭება უნიშნო გრძელი მთელი რიცხვი
ბ. ricxvi ცვლადს მიენიჭება გრძელი მთელი რიცხვი
გ. ricxvi ცვლადს მიენიჭება უნიშნო მოკლე მთელი რიცხვი

ფაილებთან პირდაპირი მიმართვა

10.5.1. რომელი მსჯელობაა არასწორი:
ა. ფაილებთან შესაძლებელია ლოგიკური მიმართვა
ბ. ფაილებთან შესაძლებელია მიმდევრობითი მიმართვა
გ. ფაილებთან შესაძლებელია პირდაპირი მიმართვა

10.5.2. ფაილთან მიმდევრობითი მიმართვის დროს:
ა. პირდაპირ სრულდება ფაილში მონაცემების ჩაწერა და წაკითხვა

ბ. მიმდევრობით სრულდება ფაილში მონაცემების ჩაწერა და წაკითხვა
გ. მიმდევრობით სრულდება მონაცემების ჩაწერა და პირდაპირ სრულდება წაკითხვა

10.5.3. ფაილთან პირდაპირი მიმართვის დროს:
ა. მიმდევრობით სრულდება მონაცემების ჩაწერა და პირდაპირ სრულდება წაკითხვა
ბ. მიმდევრობით სრულდება ფაილში მონაცემების ჩაწერა და წაკითხვა
გ. პირდაპირ სრულდება ფაილში მონაცემების ჩაწერა და წაკითხვა

10.5.4. ფაილთან პირდაპირი მიმართვის დროს გამოიყენება:
ა. ლოგიკური მიმთითებელი
ბ. საფაილო მიმთითებელი
გ. სიმბოლური მიმთითებელი

10.5.5. საფაილო მიმთითებელი არის:
ა. ფაილის მიმდინარე ელემენტის პოზიციის ნომერი
ბ. ფაილის სახელი
გ. ფაილის ტიპი

10.5.6. Seek() მეთოდი გამოიყენება:
ა. ფაილის ტიპის შესაცვლელად
ბ. ფაილის სახელის შესაცვლელად
გ. საფაილო ცვლადის მნიშვნელობის შესაცვლელად

10.5.7. საფაილო ცვლადის მნიშვნელობის შესაცვლელად გამოიყენება:
ა. object ობიექტი
ბ. FileStream კლასი
გ. Seek() მეთოდი

10.5.8. SeekOrigin.Begin მნიშვნელობა მიუთითებს იმაზე, რომ:
ა. გადათვლა უნდა შესრულდეს ფაილის დასაწყისიდან
ბ. გადათვლა უნდა შესრულდეს ფაილის ბოლოდან
გ. გადათვლა უნდა შესრულდეს ფაილის შუიდან

10.5.9. SeekOrigin.End მნიშვნელობა მიუთითებს იმაზე, რომ:
ა. გადათვლა უნდა შესრულდეს ფაილის დასაწყისიდან
ბ. გადათვლა უნდა შესრულდეს ფაილის მიმდინარე პოზიციიდან
გ. გადათვლა უნდა შესრულდეს ფაილის ბოლოდან

10.5.10. SeekOrigin.Current მნიშვნელობა მიუთითებს იმაზე, რომ:
ა. გადათვლა უნდა შესრულდეს ფაილის ბოლოდან
ბ. გადათვლა უნდა შესრულდეს ფაილის მიმდინარე პოზიციიდან
გ. გადათვლა უნდა შესრულდეს ფაილის დასაწყისიდან

10.5.11. file1.Seek(10, SeekOrigin.Begin); ოპერატორის შესრულების შედეგად:
ა. შესრულდება 10 ბაიტის გადათვლა ფაილის ბოლოდან
ბ. შესრულდება 10 ბაიტის გადათვლა ფაილის დასაწყისიდან
გ. შესრულდება 10 ბაიტის გადათვლა ფაილის მიმდინარე პოზიციიდან

10.5.12. file1.Seek(10, SeekOrigin.Current); ოპერატორის შესრულების შედეგად:
ა. შესრულდება 10 ბაიტის გადათვლა ფაილის დასაწყისიდან
ბ. შესრულდება 10 ბაიტის გადათვლა ფაილის ბოლოდან
გ. შესრულდება 10 ბაიტის გადათვლა ფაილის მიმდინარე პოზიციიდან

10.5.13. file1.Seek(10, SeekOrigin.End); ოპერატორის შესრულების შედეგად:
ა. შესრულდება 10 ბაიტის გადათვლა ფაილის ბოლოდან
ბ. შესრულდება 10 ბაიტის გადათვლა ფაილის მიმდინარე პოზიციიდან
გ. შესრულდება 10 ბაიტის გადათვლა ფაილის დასაწყისიდან

ფაილის შესახებ ინფორმაციის მიღება

10.6.1. File კლასში განსაზღვრულია:
ა. სტატიკური მეთოდები
ბ. როგორც სტატიკური, ისე ჩვეულებრივი მეთოდები
გ. ჩვეულებრივი მეთოდები

10.6.2. FileInfo კლასში განსაზღვრულია:
ა. სტატიკური მეთოდები
ბ. ჩვეულებრივი მეთოდები
გ. როგორც სტატიკური, ისე ჩვეულებრივი მეთოდები

10.6.3. რადგან File კლასის მეთოდები სტატიკურია, ამიტომ მათთან სამუშაოდ:
ა. საჭიროა ობიექტის შექმნა
ბ. აუცილებელია ობიექტის შექმნა
გ. არ არის საჭირო ობიექტის შექმნა

10.6.4. რადგან FileInfo კლასის მეთოდები ჩვეულებრივია, ამიტომ მათთან სამუშაოდ:
ა. საჭიროა ობიექტის შექმნა
ბ. არ არის საჭირო ობიექტის შექმნა
გ. არაა აუცილებელი ობიექტის შექმნა

10.6.5. FileInfo კლასის AppendText() მეთოდი:
ა. არსებულ ფაილს ტექსტს უმატებს
ბ. არსებულ ფაილს არაფერს არ უმატებს
გ. არსებულ ფაილს ტექსტს არ უმატებს

10.6.6. FileInfo კლასის CopyTo() მეთოდი:
ა. ახდენს ფაილის წაშლას
ბ. ახდენს ფაილის გადაწერას
გ. ახდენს ფაილის შექმნას

10.6.7. FileInfo კლასის Create() მეთოდი:
ა. ხურავს ფაილს
ბ. შლის ფაილს
გ. ქმნის ფაილს

10.6.8. FileInfo კლასის CreateText() მეთოდი:
ა. ქმნის ტექსტურ ფაილს
ბ. შლის ტექსტურ ფაილს
გ. ხურავს ტექსტურ ფაილს

10.6.9. FileInfo კლასის Delete() მეთოდი:
ა. ქმნის ფაილს
ბ. შლის ფაილს
გ. ხურავს ფაილს

10.6.10. FileInfo კლასის MoveTo() მეთოდი:
ა. ახდენს ფაილის გადატანას
ბ. ახდენს ფაილის გადაწერას
გ. ახდენს ფაილის წაშლას

10.6.11. FileInfo კლასის Open() მეთოდი:
ა. შლის ფაილს
ბ. ხურავს ფაილს
გ. ხსნის ფაილს

10.6.12. FileInfo კლასის OpenText() მეთოდი:
ა. ხსნის ფაილს ტექსტის წასაშლელად
ბ. ხსნის ფაილს ტექსტის გადასაწერად
გ. ხსნის ფაილს ტექსტის გადასატანად

10.6.13. FileInfo კლასის OpenWrite() მეთოდი:
ა. ხსნის ფაილს წაკითხვისათვის
ბ. ხსნის ფაილს ჩაწერისათვის
გ. ხსნის ფაილს დახურვისათვის

10.6.14. FileInfo კლასის Name თვისება:
ა. გასცემს ფაილის სახელს
ბ. გასცემს ფაილის გაფართოებას
გ. გასცემს ფაილის ზომას

10.6.15. FileInfo კლასის Length თვისება:
ა. გასცემს ფაილის სახელს
ბ. გასცემს ფაილის ზომას
გ. გასცემს ფაილის გაფართოებას

10.6.16. FileInfo კლასის FullName თვისება:
ა. გასცემს ფაილის ზომას
ბ. გასცემს ფაილის შექმნის თარიღს
გ. გასცემს ფაილისკენ გზას და ფაილის სახელს

10.6.17. FileInfo კლასის DirectoryName თვისება:
ა. გასცემს კატალოგის სახელს, რომელშიც ფაილია მოთავსებული

ბ. არ გასცემს კატალოგის სახელს, რომელშიც ფაილია მოთავსებული
გ. გასცემს კატალოგის სახელს, რომელშიც ფაილი არაა მოთავსებული

10.6.18. File კლასის Copy() მეთოდი:
ა. ახდენს ფაილის გადაწერას
ბ. ახდენს ფაილის გადატანას
გ. ახდენს ფაილის დახურვას

10.6.19. File კლასის Create() მეთოდი:
ა. შლის ფაილს
ბ. ქმნის ფაილს
გ. ხურავს ფაილს

10.6.20. File კლასის Delete() მეთოდი:
ა. ქმნის ფაილს
ბ. შლის ფაილს
გ. ხურავს ფაილს

10.6.21. File კლასის Open() მეთოდი:
ა. ხურავს ფაილს
ბ. შლის ფაილს
გ. ხსნის ფაილს

10.6.22. File კლასის OpenRead() მეთოდი:
ა. გასცემს ფაილის ატრიბუტებს
ბ. ხსნის ფაილს ჩაწერისათვის
გ. ხსნის ფაილს წაკითხვისათვის

10.6.23. File კლასის OpenWrite() მეთოდი:
ა. ხსნის ფაილს ჩაწერისათვის
ბ. ხსნის ფაილს დახურვისათვის
გ. გასცემს ფაილის ატრიბუტებს

10.6.24. File კლასის Exists() მთოდი:
ა. ამოწმებს ფაილის არსებობას
ბ. არ ამოწმებს ფაილის არსებობას
გ. ხსნის ფაილს წაკითხვისათვის

10.6.25. File კლასის GetAttributes() მეთოდი:
ა. ხსნის ფაილს წაკითხვისათვის
ბ. გასცემს ფაილის ატრიბუტებს
გ. ხსნის ფაილს წაკითხვისათვის

კატალოგებთან მუშაობა

10.7.1. DirectoryInfo კლასის Name თვისება:
ა. გასცემს კატალოგის სახელს

ბ. ქმნის ახალ კატალოგს
გ. გასცემს ან აყენებს კატალოგის შექმნის დროს

10.7.2. DirectoryInfo კლასის Parent თვისება:
ა. გასცემს კატალოგის სახელს
ბ. გასცემს მშობელი კატალოგის სახელს
გ. ქმნის ახალ კატალოგს

10.7.3. DirectoryInfo კლასის Root თვისება:
ა. ქმნის ახალ კატალოგს
ბ. გასცემს კატალოგის სახელს
გ. გასცემს ძირითადი კატალოგის სახელს მითითებული კატალოგისათვის

10.7.4. DirectoryInfo კლასის FullName თვისება:
ა. ქმნის ახალ კატალოგს
ბ. გასცემს გზას კატალოგისკენ და კატალოგის სახელს
გ. გასცემს კატალოგის სახელს

10.7.5. DirectoryInfo კლასის Exist თვისება:
ა. ამოწმებს კატალოგის არსებობას
ბ. გასცემს ან აყენებს კატალოგის შექმნის დროს
გ. ქმნის ახალ კატალოგს

10.7.6. DirectoryInfo კლასის CreationTime თვისება:
ა. გასცემს ან აყენებს კატალოგის შექმნის დროს
ბ. ქმნის ახალ კატალოგს
გ. გასცემს კატალოგის სახელს

10.7.7. DirectoryInfo კლასის Create() მეთოდი:
ა. ქმნის ახალ ქვეკატალოგს
ბ. გასცემს ან აყენებს კატალოგის შექმნის დროს
გ. ქმნის ახალ კატალოგს

10.7.8. DirectoryInfo კლასის CreateSubdirectory() მეთოდი:
ა. გასცემს ან აყენებს კატალოგის შექმნის დროს
ბ. ქმნის ახალ ქვეკატალოგს
გ. ქმნის ახალ კატალოგს

10.7.9. DirectoryInfo კლასის Delete() მეთოდი:
ა. შლის კატალოგს
ბ. გასცემს ან აყენებს კატალოგის შექმნის დროს
გ. ქმნის ახალ ქვეკატალოგს

10.7.10. DirectoryInfo კლასის GetDirectories() მეთოდი:
ა. გასცემს ან აყენებს კატალოგის შექმნის დროს
ბ. ქმნის ახალ ქვეკატალოგს
გ. გასცემს მიმდინარე კატალოგის ქვეკატალოგების სიას

10.7.11. DirectoryInfo კლასის GetFiles() მეთოდი:

ა. შლის კატალოგს მის შემცველობასთან ერთად

ბ. გასცემს მიმდინარე კატალოგის ფაილების სიას

გ. გასცემს მითითებულ კატალოგში მოთავსებული ყველა ფაილის სახელს

10.7.12. DirectoryInfo კლასის MoveTo() მეთოდი:

ა. ახდენს კატალოგის გადატანას

ბ. გასცემს მიმდინარე კატალოგის ფაილების სიას

გ. შლის კატალოგს მის შემცველობასთან ერთად

10.7.13. Directory კლასის CreateDirectory() მეთოდი:

ა. შლის კატალოგს მის შემცველობასთან ერთად

ბ. ქმნის ახალ კატალოგს

გ. გასცემს მიმდინარე კატალოგის ფაილების სიას

10.7.14. Directory კლასის Delete() მეთოდი:

ა. გასცემს მიმდინარე კატალოგის ფაილების სიას

ბ. გასცემს მითითებულ კატალოგში მოთავსებული ყველა ფაილის სახელს

გ. შლის კატალოგს მის შემცველობასთან ერთად

10.7.15. Directory კლასის Exists() მეთოდი:

ა. განსაზღვრავს არსებობს თუ არა კატალოგი

ბ. შლის კატალოგს მის შემცველობასთან ერთად

გ. გასცემს მითითებულ კატალოგში მოთავსებული ყველა ფაილის სახელს

10.7.16. Directory კლასის GetDirectories() მეთოდი:

ა. შლის კატალოგს მის შემცველობასთან ერთად

ბ. გასცემს მინდინარე კატალოგის ქვეკატალოგების სახელებს

გ. გასცემს მითითებულ კატალოგში მოთავსებული ყველა ფაილის სახელს

10.7.17. Directory კლასის GetFiles() მეთოდი:

ა. გასცემს მშობელი კატალოგის სახელს

ბ. შლის კატალოგს მის შემცველობასთან ერთად

გ. გასცემს მითითებულ კატალოგში მოთავსებული ყველა ფაილის სახელს

10.7.18. Directory კლასის GetParent() მეთოდი:

ა. გასცემს მშობელი კატალოგის სახელს

ბ. გასცემს მითითებულ კატალოგში მოთავსებული ყველა ფაილის სახელს

გ. ახდენს კატალოგის გადატანას

10.7.19. Directory კლასის Move() მეთოდი:

ა. გასცემს მშობელი კატალოგის სახელს

ბ. ახდენს კატალოგის გადატანას

გ. გასცემს მინდინარე კატალოგის ქვეკატალოგების სახელებს

NetworkStream კლასი

10.8.1. NetworkStream კლასი გამოცხადებულია;
ა. System.IO სახელების სივრცეში
ბ. System.Net.Sockets სახელების სივრცეში
გ. System.Data სახელების სივრცეში

10.8.2. ქსელური შეერთება ეფუძნება:
ა. ინდექსატორებს
ბ. სოკეტებს
გ. თვისებებს

10.8.3. სოკეტი შეიცავს:
ა. ჰოსთის მისამართს და პორტის ნომერს
ბ. მხოლოდ პორტის ნომერს
გ. მხოლოდ ჰოსთის მისამართს

10.8.4. ჰოსთის მისამართი შეიძლება იყოს:
ა. პროცესის სახელი
ბ. სოკეტის სახელი
გ. TCP/IP მისამართი ან ჰოსთის სახელი

10.8.5. პორტის ნომრები იყოფა:
ა. ორ ჯგუფად
ბ. სამ ჯგუფად
გ. ოთხ ჯგუფად

10.8.6. 0÷1023 პორტები დაკავებული აქვს:
ა. ცნობილ სამსახურებს
ბ. ნაკლებად ცნობილ სამსახურებს
გ. მომხმარებლების მიერ შემუშავებულ პროგრამებს

10.8.7. 1024÷49151 პორტები დაკავებული აქვს:
ა. მომხმარებლების მიერ შემუშავებულ პროგრამებს
ბ. ცნობილ სამსახურებს
გ. ნაკლებად ცნობილ სამსახურებს

10.8.8. 49152÷65536 პორტები დაკავებული აქვს:
ა. ცნობილ სამსახურებს
ბ. ნაკლებად ცნობილ სამსახურებს
გ. მომხმარებლების მიერ შემუშავებულ პროგრამებს

10.8.9. იმისათვის, რომ ორი პროგრამა ერთმანეთს დაუკავშირდეს ქსელის საშუალებით:
ა. თითოეულმა მათგანმა უნდა შექმნას სოკეტი
ბ. პირველმა მათგანმა უნდა შექმნას სოკეტი
გ. მეორე მათგანმა უნდა შექმნას სოკეტი

10.8.10. სოკეტის შესაქმნელად ვიყენებთ:
ა. TopClient კლასს
ბ. Socket კლასს
გ. NetworkStream კლასს

10.8.11. Socket კლასი განსაზღვრულია:
ა. System.Data სახელების სივრცეში
ბ. System.Net.Sockets სახელების სივრცეში
გ. System.IO სახელების სივრცეში

10.8.12. TopListener კლასი:
ა. ხსნის სოკეტს და ელოდება მასთან კლიენტის მიერთებას
ბ. ხსნის სოკეტს და უერთდება სამსახურს დაშორებულ კომპიუტერზე

10.8.13. TopClient კლასი:
ა. ხსნის სოკეტს და ელოდება მასთან კლიენტის მიერთებას
ბ. ხსნის სოკეტს და უერთდება სამსახურს დაშორებულ კომპიუტერზე

10.8.14. TopListener კლასის Start() მეთოდი:
ა. განაგრძობს პორტის მოსმენას ქსელში ახალი შეერთების მოლოდინში
ბ. ამთავრებს პორტის მოსმენას ქსელში ახალი შეერთების მოლოდინში
გ. იწყებს პორტის მოსმენას ქსელში ახალი შეერთების მოლოდინში

10.8.15. TopListener კლასის AcceptSocket() მეთოდი:
ა. არ ახდენს კლიენტის რეალურ შეერთებას, ანუ ქმნის ახალ სოკეტს შეერთებისათვის
ბ. ახდენს კლიენტის რეალურ შეერთებას, ანუ ქმნის ახალ სოკეტს შეერთებისათვის
გ. ახდენს კლიენტის რეალურ შეერთებას, ანუ არ ქმნის ახალ სოკეტს შეერთებისათვის

10.8.16. NetworkStream კლასის Write() მეთოდი გამოიყენება:
ა. სოკეტში მონაცემების გადასაცემად
ბ. სოკეტიდან მონაცემების მისაღებად

10.8.17. NetworkStream კლასის Read() მეთოდი გამოიყენება:
ა. სოკეტიდან მონაცემების მისაღებად
ბ. სოკეტში მონაცემების გადასაცემად

MemoryStream და BufferedStream კლასები

10.9.1. MemoryStream და BufferStream კლასების გამოყენება:
ა. ზრდის კოდის ეფექტურობას
ბ. ამცირებს კოდის ეფექტურობას

10.9.2. BufferStream კლასის გამოყენების შემთხვევაში მონაცემები წარმოდგენილი იქნება:
ა. ობიექტების სახით
ბ. მასივის სახით
გ. ბლოკების სახით

10.9.3. BufferStream კლასი პროგრამას გადასცემს მხოლოდ იმ მონაცემებს, რომლებიც:
ა. მასივშია მოთავსებული
ბ. მოთხოვნილი არ იყო
გ. მოთხოვნილი იყო

10.9.4. BufferStream კლასი ოპერატიულ მეხსიერებაში შეიძლება ინახავდეს ჩანაწერის:
ა. მხოლოდ ერთი ოპერაციის შედეგს
ბ. რამდენიმე ოპერაციის შედეგს

10.9.5. შიდა ბუფერირების გამო უნდა გამოვიყენოთ:
ა. Close() მეთოდი
ბ. Flush() მეთოდი
გ. Write() მეთოდი

10.9.6. MemoryStream კლასის ნაკადებისთვის მონაცემების სათავსოა:
ა. ოპერატიული მეხსიერება
ბ. მასივი
გ. ლოკალური დისკი

10.9.7. MemoryStream კლასის SetLength() მეთოდი ახდენს:
ა. ბუფერის ზომის გაზრდას
ბ. ბუფერის ზომის ძემცირებას

10.9.8. ბუფერთან პირდაპირი მიმართვისათვის გამოიყენება:
ა. Read() მეთოდი
ბ. GetBuffer() მეთოდი
გ. Close() მეთოდი

მონაცემების ასინქრონული შეტანა-გამოტანა

10.10.1. მონაცემების სინქრონული შეტანა-გამოტანის დროს:
ა. იწყება პროგრამის შესრულება
ბ. არ ჩერდება პროგრამის შესრულება
გ. ჩერდება პროგრამის შესრულება

10.10.2. მონაცემების ასინქრონული შეტანა-გამოტანის დროს შეტანა-გამოტანის ოპერაციების შესრულებისათვის:
ა. ცალკე ნაკადი იქმნება
ბ. ცალკე ნაკადი არ იქმნება

10.10.3. შეტანა-გამოტანის ოპერაციის დამთავრებისთანავე უკუგამოძახების ფუნქციის მიერ პროგრამა:
ა. არ იღებს შეტყობინებას
ბ. იღებს შეტყობინებას

10.10.4. უკუგამოძახების ფუნქცია გამოიძახება:
ა. თვისების მიერ

ბ. მოვლენის მიერ
გ. დელეგატის მიერ

10.10.5. თუ ფაილი გახსნილია სინქრონულ რეჟიმში სამუშაოდ, მაშინ ასინქრონული შეტანა-გამოტანის მეთოდები მასთან:
ა. სინქრონულ რეჟიმში იმუშავებენ
ბ. ასინქრონულ რეჟიმში იმუშავებენ
გ. საერთოდ არ იმუშავებენ

თავი 11. განსაკუთრებული სიტუაციები

განსაკუთრებული სიტუაციების დამუშავების საფუძვლები

11.1.1. განსაკუთრებული სიტუაცია არის სიტუაცია, რომელიც:
ა. არ იწვევს პროგრამის ავარიულად დამთავრებას
ბ. არ არის გამოწვეული პროგრამის შესრულების დროს აღძრული შეცდომის მიერ
გ. გამოწვეულია პროგრამის შესრულების დროს აღძრული შეცდომის მიერ

11.1.2. განსაკუთრებული სიტუაციებია:
ა. მასივის გამოცხადება და გადავსება
ბ. ნულზე გაყოფა და არარსებული ფაილის გახსნა
გ. კლასის გამოცხადება და მასივის ინდექსის გასვლა დიაპაზონის გარეთ

11.1.3. რა არის განსაკუთრებული სიტუაციების დამუშავების უპირატესობა:
ა. ის, რომ პროგრამას შეუძლია შეცდომის დამუშავება და შესრულების გაგრძელება
ბ. ის, რომ პროგრამას არ შეუძლია შეცდომის დამუშავება და შესრულების გაგრძელება
გ. ის, რომ პროგრამას შეუძლია შეცდომის დამუშავება, მაგრამ ავარიულად ამთავრებს მუშაობას

11.1.4. განსაკუთრებული სიტუაცია იწვევს:
ა. პროგრამის შესრულების ავარიულად დამთავრებას
ბ. პროგრამის შესრულების ნორმალურად დამთავრებას
გ. პროგრამის შესრულების გაგრძელებას

11.1.5. განსაკუთრებული სიტუაციების დამამუშავებლები იძლევიან:
ა. კლასების გამოცხადების საშუალებას
ბ. პროგრამის შესრულების ავარიულად დამთავრების საშუალებას
გ. პროგრამის შესრულების გაგრძელების საშუალებას

11.1.6. რომელია განსაკუთრებული სიტუაციების საბაზო კლასი:
ა. object კლასი
ბ. Exception კლასი
გ. FileStream კლასი

11.1.7. განსაკუთრებული სიტუაციების ყველა კლასი მემკვიდრეობით მიიღება:
ა. object ობიექტისაგან
ბ. Exception კლასისაგან
გ. FileStream კლასისაგან

11.1.8. Exception კლასი არის:
ა. System სახელების სივრცის ნაწილი
ბ. System.Data სახელების სივრცის ნაწილი
გ. System.Drawing სახელების სივრცის ნაწილი

11.1.9. განსაკუთრებული სიტუაციების რომელი კატეგორიები არსებობს:
ა. არსებობს ერთი კატეგორია: გენერირებული C# ენის შესრულების გარემოს მიერ

ბ. არსებობს ერთი კატეგორია: გენერირებული პროგრამა-დანართების მიერ
გ. არსებობს ორი კატეგორია: გენერირებული C# ენის შესრულების გარემოს მიერ და გენერირებული პროგრამა-დანართების მიერ

11.1.10. C# ენის შესრულების გარემოს (CLR) მიერ გენერირებული განსაკუთრებული სიტუაციები აღწერილია:
ა. object ობიექტში
ბ. SystemException კლასში
გ. ApplicationException კლასში

11.1.11. პროგრამა-დანართების მიერ გენერირებული განსაკუთრებული სიტუაციები აღწერილია:
ა. object ობიექტში
ბ. SystemException კლასში
გ. ApplicationException კლასში

11.1.12. რომელი განსაკუთრებული სიტუაცია აღიძვრება ნულზე გაყოფის დროს:
ა. ArrayTypeMismatchException
ბ. DivideByZeroException
გ. IndexOutOfRangeException

11.1.13. ArrayTypeMismatchException განსაკუთრებული სიტუაცია აღიძვრება მაშინ, როცა:
ა. მასივის ინდექსი გადის დიაპაზონის გარეთ
ბ. ადგილი აქვს ნულზე გაყოფას
გ. მნიშვნელობის ტიპი შეუთავსებელია მასივის ტიპთან

11.1.14. DivideByZeroException განსაკუთრებული სიტუაცია აღიძვრება მაშინ, როცა:
ა. ადგილი აქვს ნულზე გაყოფას
ბ. მნიშვნელობის ტიპი შეუთავსებელია მასივის ტიპთან
გ. მასივის ინდექსი გადის დიაპაზონის გარეთ

11.1.15. IndexOutOfRangeException განსაკუთრებული სიტუაცია აღიძვრება მაშინ, როცა:
ა. ადგილი აქვს არაკორექტულ გარდაქმნას პროგრამის შესრულების დროს
ბ. მასივის ინდექსი გადის დიაპაზონის გარეთ
გ. new ოპერატორის გამოძახება იყო არაკორექტული მეხსიერების უკმარისობის გამო

11.1.16. InvalidCastException განსაკუთრებული სიტუაცია აღიძვრება მაშინ, როცა:
ა. new ოპერატორის გამოძახება იყო არაკორექტული მეხსიერების უკმარისობის გამო
ბ. ადგილი აქვს არაკორექტულ გარდაქმნას პროგრამის შესრულების დროს
გ. ადგილი აქვს სტეკის გადავსებას

11.1.17. OutOfMemoryException განსაკუთრებული სიტუაცია აღიძვრება მაშინ, როცა:
ა. new ოპერატორის გამოძახება იყო არაკორექტული მეხსიერების უკმარისობის გამო
ბ. მასივის ინდექსი გადის დიაპაზონის გარეთ
გ. ადგილი აქვს არაკორექტულ გარდაქმნას პროგრამის შესრულების დროს

11.1.18. OverflowException განსაკუთრებული სიტუაცია აღიძვრება მაშინ, როცა:
ა. ადგილი აქვს ნულზე გაყოფას

ბ. ადგილი აქვს სტეკის გადავსებას
გ. ადგილი აქვს გადავსებას არითმეტიკული ოპერაციის შესრულების დროს

11.1.19. StackOverflowException განსაკუთრებული სიტუაცია აღიძვრება მაშინ, როცა:
ა. ადგილი აქვს სტეკის გადავსებას
ბ. ადგილი აქვს გადავსებას არითმეტიკული ოპერაციის შესრულების დროს
გ. ადგილი აქვს ნულზე გაყოფას

11.1.20. განსაკუთრებული სიტუაციების დამუშავება სრულდება შემდეგი ოპერატორების გამოყენებით:
ა. try, catch, params
ბ. try, catch, throw, finally
გ. throw, finally, object

11.1.21. try ბლოკში მოთავსებულია ის ოპერატორები, რომელთა:
ა. შესრულებასაც თვალყური არ უნდა ვადევნოთ
ბ. შესრულებასაც თვალყური უნდა ვადევნოთ
გ. შეცვლაც გვინდა

11.1.22. განსაკუთრებულ სიტუაციას იჭერს და ამუშავებს:
ა. try ბლოკი
ბ. finally ბლოკი
გ. catch ბლოკი

11.1.23. განსაკუთრებული სიტუაციის ხელოვნურად გენერირებას იწვევს:
ა. catch ოპერატორი
ბ. try ოპერატორი
გ. throw ოპერატორი

11.1.24. finally ბლოკში მოთავსებულია კოდი, რომელიც:
ა. ყოველთვის შესრულდება
ბ. ყოველთვის არ შესრულდება
გ. ხანდახან შესრულდება

try და catch ბლოკები

11.2.1. try ბლოკთან შეიძლება დაკავშირებული იყოს:
ა. რამდენიმე catch ოპერატორი
ბ. მხოლოდ 1 catch ოპერატორი
გ. მხოლოდ 2 catch ოპერატორი

11.2.2. catch ოპერატორი იჭერს იმ განსაკუთრებულ სიტუაციას, რომელიც:
ა. გამოცხადებულია FileStream კლასში
ბ. არ შეესაბამება catch ოპერატორში მითითებულ ტიპს
გ. შეესაბამება catch ოპერატორში მითითებულ ტიპს

11.2.3. თუ განსაკუთრებული სიტუაციის გენერირება არ ხდება, მაშინ:
ა. catch ბლოკი შესრულდება
ბ. catch ბლოკი არ შესრულდება
გ. შესრულდება უპარამეტრებო catch ოპერატორი

11.2.4. catch ოპერატორში პარამეტრის მითითება:
ა. არ არის აუცილებელი
ბ. აუცილებელია
გ. ნაწილობრივაა აუცილებელი

11.2.5. catch ოპერატორში პარამეტრის მითითება საჭიროა მაშინ, როცა:
ა. არ არის აუცილებელი განსაკუთრებული სიტუაციის ობიექტთან მიმართვა
ბ. აუცილებელია განსაკუთრებული სიტუაციის ობიექტთან მიმართვა
გ. გამოტოვებულია try ბლოკი

11.2.6. განსაკუთრებული სიტუაციის დაჭერა და დამუშავება თავიდან აგვაცილებს:
ა. try ბლოკის შესრულებას
ბ. პროგრამის ნორმალურად დამთავრებას
გ. პროგრამის ავარიულად დამთავრებას

11.2.7. თუ პროგრამა არ ახდენს განსაკუთრებული სიტუაციის დაჭერას, მაშინ განსაკუთრებულ სიტუაციას დაიჭერს:
ა. System სახელების სივრცე
ბ. Windows ოპერაციული სისტემა
გ. C#-ის შესრულების სისტემა

11.2.8. რომელი მსჯელობაა სწორი:
ა. თუ განსაკუთრებული სიტუაციის ტიპი შეესაბამება catch ოპერატორში მითითებულ ტიპს, მაშინ ეს catch ოპერატორი ვერ დაიჭერს ამ განსაკუთრებულ სიტუაციას
ბ. თუ განსაკუთრებული სიტუაციის ტიპი შეესაბამება catch ოპერატორში მითითებულ ტიპს, მაშინ ეს catch ოპერატორი დაიჭერს ამ განსაკუთრებულ სიტუაციას
გ. თუ განსაკუთრებული სიტუაციის ტიპი არ შეესაბამება catch ოპერატორში მითითებულ ტიპს, მაშინ ეს catch ოპერატორი დაიჭერს ამ განსაკუთრებულ სიტუაციას

11.2.9. ყველა განსაკუთრებული სიტუაციის დასაჭერად უნდა გამოვიყენოთ catch ოპერატორი:
ა. პარამეტრების გარეშე
ბ. ერთი პარამეტრით
გ. ორი პარამეტრით

11.2.10. catch ოპერატორები მოწმდება:
ა. იმ მიმდევრობით, როგორც არის პროგრამაში მითითებული
ბ. უკუმიმდევრობით
გ. დაწყებული შუა ოპერატორიდან

11.2.11. რისთვის გამოიყენება უპარამეტრებო catch ოპერატორი:
ა. ზოგიერთი განსაკუთრებული სიტუაციის დასაჭერად
ბ. ყველა განსაკუთრებული სიტუაციის დასაჭერად
გ. არ იჭერს არც ერთ განსაკუთრებულ სიტუაციას

ჩადგმული try ბლოკები

11.3.1. რომელი მსჯელობაა სწორი:
ა. try ბლოკში უნდა მოთავსდეს მხოლოდ finally ბლოკი
ბ. არაა შესაძლებელი try ბლოკების ერთმანეთში მოთავსება
გ. შესაძლებელია try ბლოკების ერთმანეთში მოთავსება

11.3.2. განსაკუთრებული სიტუაცია, რომელიც ვერ დაიჭირა შიდა try ბლოკის შესაბამისმა catch ოპერატორმა:
ა. ვრცელდება გარე try ბლოკზე
ბ. არ ვრცელდება გარე try ბლოკზე
გ. ვრცელდება მხოლოდ შიდა try ბლოკზე

throw ოპერატორი

11.4.1. რომელი მსჯელობაა სწორი:
ა. რადგან throw ოპერატორი ოპერირებს ობიექტებზე, ამიტომ მის გამოყენებამდე საჭიროა ობიექტის შექმნა
ბ. რადგან throw ოპერატორი არ ოპერირებს ობიექტებზე, ამიტომ მის გამოყენებამდე საჭიროა ობიექტის შექმნა
გ. რადგან throw ოპერატორი ოპერირებს ობიექტებზე, ამიტომ მის გამოყენებამდე არ არის საჭირო ობიექტის შექმნა

finally ბლოკი

11.5.1. რომელი მსჯელობაა სწორი:
ა. finally ბლოკი არასოდეს არ გამოიძახება
ბ. finally ბლოკი გამოიძახება იმისგან დამოუკიდებლად, გენერირდება თუ არა განსაკუთრებული სიტუაცია
გ. finally ბლოკი გამოიძახება მხოლოდ catch ბლოკში

11.5.2. finally ბლოკის კოდი:
ა. შესრულდება მხოლოდ catch ოპერატორში
ბ. არასოდეს არ შესრულდება
გ. ყოველთვის შესრულდება

checked და unchecked ოპერატორები

11.6.1. checked ოპერატორი გამოიყენება იმ:
ა. გამოსახულების მისათითებლად, რომლის შემოწმებაც უნდა მოხდეს გადავსების არსებობაზე
ბ. გამოსახულების მისათითებლად, რომლის შემოწმებაც არ უნდა მოხდეს გადავსების

არსებობაზე

გ. ოპერატორის ბლოკის მისათითებლად, რომლის შემოწმებაც არ უნდა მოხდეს გადავსების არსებობაზე

11.6.2. checked ოპერატორი გამოიყენება იმ:

ა. ოპერატორების ბლოკის მისათითებლად, რომლის შემოწმებაც არ უნდა მოხდეს გადავსების არსებობაზე

ბ. ოპერატორების ბლოკის მისათითებლად, რომლის შემოწმებაც უნდა მოხდეს გადავსების არსებობაზე

გ. გამოსახულების მისათითებლად, რომლის შემოწმებაც არ უნდა მოხდეს გადავსების არსებობაზე

11.6.3. როგორია checked ოპერატორის სინტაქსი, რომელიც გამოსახულებას ამოწმებს:

ა. checked

ბ. checked { შესამოწმებელი ოპერატორები }

გ. checked (გამოსახულება)

11.6.4. როგორია checked ოპერატორის სინტაქსი, რომელიც ოპერატორების ბლოკს ამოწმებს:

ა. checked

ბ. checked { შესამოწმებელი ოპერატორები }

გ. checked (გამოსახულება)

11.6.5. checked ოპერატორი იწვევს შემდეგი განსაკუთრებული სიტუაციის გენერირებას:

ა. OverflowException

ბ. IndexOutOfRangeException

გ. DivideByZeroException

11.6.6. unchecked ოპერატორი გამოიყენება იმ:

ა. ოპერატორების ბლოკის მიმართ, რომლისთვისაც არ უნდა შესრულდეს გადავსების იგნორირება

ბ. გამოსახულების მიმართ, რომლისთვისაც არ უნდა შესრულდეს გადავსების იგნორირება

გ. გამოსახულების მიმართ, რომლისთვისაც უნდა შესრულდეს გადავსების იგნორირება

11.6.7. unchecked ოპერატორი გამოიყენება იმ:

ა. ოპერატორების ბლოკის მიმართ, რომლისთვისაც არ უნდა შესრულდეს გადავსების იგნორირება

ბ. ოპერატორების ბლოკის მიმართ, რომლისთვისაც უნდა შესრულდეს გადავსების იგნორირება

გ. გამოსახულების მიმართ, რომლისთვისაც არ უნდა შესრულდეს გადავსების იგნორირება

11.6.8. თუ unchecked ოპერატორის გამოყენების შემთხვევაში ადგილი ექნა გადავსებას, მაშინ:

ა. შესრულდება შედეგის ჩამოჭრა

ბ. არ შესრულდება შედეგის ჩამოჭრა

გ. გენერირდება OverflowException განსაკუთრებული სიტუაცია

11.6.9. unchecked ოპერატორის გამოყენების შემთხვევაში შედეგის ჩამოჭრა ხდება იმ მიზნით, რომ:

ა. გამოსახულების ტიპი არ დაემთხვეს შედეგის ტიპს

ბ. გამოსახულების ტიპი დაემთხვეს შედეგის ტიპს
გ. შედეგის ტიპი გახდეს int

11.6.10.როგორია unchecked ოპერატორის სინტაქსი, რომელიც ახდენს გადავსების იგნორირებას გამოსახულებისათვის:
ა. unchecked
ბ. unchecked { შესამოწმებელი ოპერატორები }
გ. unchecked (გამოსახულება)

11.6.11.როგორია unchecked ოპერატორის სინტაქსი, რომელიც ახდენს გადავსების იგნორირებას ოპერატორების ბლოკისათვის:
ა. unchecked
ბ. unchecked { შესამოწმებელი ოპერატორები }
გ. unchecked (გამოსახულება)

თავი 12. სტრიქონები

სტრიქონები

12.1.1. სტრიქონები წარმოადგენს:
ა. მიმართვითი ტიპის მქონე ობიექტებს
ბ. ჩვეულებრივი ტიპის მქონე ობიექტებს
გ. მხოლოდ ლოგიკურ მონაცემებს

12.1.2. სტრიქონი არის:
ა. სიმბოლოების მასივი
ბ. მთელრიცხვა მასივი
გ. წილადების მასივი

12.1.3. სტრიქონებში პირველი ელემენტის ინდექსია:
ა. 1
ბ. 0
გ. -1

12.1.4. სტრიქონებს Length თვისება:
ა. არ აქვს
ბ. აქვს
გ. ხანდახან აქვს

12.1.5. სტრიქონის ცალკეული სიმბოლოს მისაღებად უნდა მივუთითოთ:
ა. სტრიქონის სახელი და ინდექსი
ბ. მხოლოდ ინდექსი
გ. მხოლოდ სახელი

12.1.6. Insert მეთოდი:
ა. გამომძახებელ სტრიქონში წაშლის მითითებულ სტრიქონს
ბ. გამომძახებელ სტრიქონში შეცვლის მითითებულ სტრიქონს
გ. გამომძახებელ სტრიქონში ჩასვამს მითითებულ სტრიქონს

12.1.7. Remove მეთოდი:
ა. გამომძახებელ სტრიქონში ჩასვამს მითითებული რაოდენობის სიმბოლოს
ბ. გამომძახებელ სტრიქონში შეცვლის მითითებული რაოდენობის სიმბოლოს
გ. გამომძახებელ სტრიქონში წაშლის მითითებული რაოდენობის სიმბოლოს

12.1.8. Replace მეთოდი:
ა. გამომძახებელ სტრიქონში ჩასვამს მითითებულ სტრიქონს
ბ. გამომძახებელ სტრიქონში წაშლის მითითებულ სტრიქონს
გ. გამომძახებელ სტრიქონში მითითებულ სტრიქონს ახალი სტრიქონით შეცვლის

12.1.9. + ოპერატორი გამოიყენება:
ა. ერთ სტრიქონში მეორე სტრიქონის ჩასასმელად

ბ. სტრიქონების კონკატენაციის ოპერაციის შესასრულებლად
გ. ერთი სტრიქონიდან ქვესტრიქონის წასაშლელად

12.1.10. LastIndexOf მეთოდი:
ა. გასცემს გამომძახებელი სტრიქონის ინდექსს, რომელშიც უკანასკნელად იყო ნაპოვნი მისი არგუმენტი
ბ. გასცემს გამომძახებელი სტრიქონის ინდექსს, რომელშიც პირველად იყო ნაპოვნი მისი არგუმენტი
გ. ადარებს გამომძახებელ სტრიქონსა და თავის არგუმენტს

12.1.11. IndexOf მეთოდი:
ა. გამომძახებელი სტრიქონის მითითებული პოზიციიდან გასცემს მითითებული რაოდენობის სიმბოლოსაგან შემდგარ ქვესტრიქონს
ბ. გასცემს გამომძახებელი სტრიქონის ინდექსს, რომელშიც პირველად იყო ნაპოვნი მისი არგუმენტი
გ. გასცემს გამომძახებელი სტრიქონის ინდექსს, რომელშიც უკანასკნელად იყო ნაპოვნი მისი არგუმენტი

12.1.12. CompareTo მეთოდი:
ა. გასცემს გამომძახებელი სტრიქონის ინდექსს, რომელშიც უკანასკნელად იყო ნაპოვნი მისი არგუმენტი
ბ. გასცემს გამომძახებელი სტრიქონის ინდექსს, რომელშიც პირველად იყო ნაპოვნი მისი არგუმენტი
გ. ადარებს გამომძახებელ სტრიქონსა და თავის არგუმენტს

12.1.13. Copy მეთოდი:
ა. გასცემს გამომძახებელი სტრიქონის ინდექსს, რომელშიც პირველად იყო ნაპოვნი მისი არგუმენტი
ბ. გამომძახებელი სტრიქონის მითითებული პოზიციიდან გასცემს მითითებული რაოდენობის სიმბოლოსაგან შემდგარ ქვესტრიქონს
გ. გამომძახებელ სტრიქონს ანიჭებს თავისი არგუმენტის ასლს

12.1.14. Substring მეთოდი:
ა. გამომძახებელი სტრიქონის მითითებული პოზიციიდან გასცემს მითითებული რაოდენობის სიმბოლოსაგან შემდგარ ქვესტრიქონს
ბ. ადარებს გამომძახებელ სტრიქონსა და თავის არგუმენტს
გ. გამომძახებელ სტრიქონს ანიჭებს თავისი არგუმენტის ასლს

სტრიქონების მასივები

12.1.15. სტრიქონების მასივის გამოცხადებისათვის string სიტყვის შემდეგ მიეთითება:
ა. მრგვალი ფრჩხილები
ბ. კვადრატული ფრჩხილები
გ. ფიგურული ფრჩხილები

სტრიქონების უცვლელობა

12.1.16. გვაქვს სტრიქონი string striqoni = "C# დაპროგრამების ენაა"; დასაშვებია თუ არა ასეთი სახის მინიჭება striqoni[1] = 'A'; :

ა. კი
ბ. არა
გ. ხანდახან

დინამიკური სტრიქონები

12.2.1. დინამიკურ სტრიქონებთან სამუშაოდ გამოიყენება:

ა. StringBuilder კლასი
ბ. Random კლასი
გ. String კლასი

12.2.2. StringBuilder კლასის მეთოდები მუშაობს უფრო სწრაფად, ვიდრე String კლასის მეთოდები, რადგან ისინი:

ა. ქმნის სტრიქონის ასლს და უშუალოდ სტრიქონთან მუშაობს
ბ. არ ქმნის სტრიქონის ასლს და უშუალოდ სტრიქონთან მუშაობს
გ. ქმნის სტრიქონის ასლს და უშუალოდ სტრიქონთან არ მუშაობს

12.2.3. დინამიკურ სტრიქონებში, String კლასის სტრიქონებისაგან განსხვავებით:

ა. შესაძლებელია სიმბოლოების ირიბად შეცვლა
ბ. შეუძლებელია სიმბოლოების პირდაპირ შეცვლა
გ. შესაძლებელია სიმბოლოების პირდაპირ შეცვლა

12.2.4. StringBuilder DinamiuriStriqoni1 = new StringBuilder(); კოდი ქმნის:

ა. ცარიელ დინამიკურ სტრიქონს
ბ. სავსე დინამიკურ სტრიქონს
გ. ნახევრად ცარიელ დინამიკურ სტრიქონს

12.2.5. შექმნის დროს დინამიკური სტრიქონის ტევადობაა:

ა. 1 სიმბოლო
ბ. 16 სიმბოლო
გ. 0 სიმბოლო

12.2.6. ახალი სიმბოლოს დამატების შემდეგ დინამიკური სტრიქონის ტევადობა:

ა. ნახევრად იზრდება
ბ. ავტომატურად მცირდება
გ. ავტომატურად იზრდება

12.2.7. ტევადობა შეგვიძლია მივუთითოთ დინამიკური სტრიქონის:

ა. შექმნის დროს
ბ. შექმნის შემდეგ
გ. წაშლისას

12.2.8. StringBuilder DinamiuriStriqoni2 = new StringBuilder(50); კოდის შესრულების შედეგად:

ა. დინამიკური სტრიქონის ტევადობა იქნება 5 სიმბოლო

ბ. დინამიკური სტრიქონის ტევადობა იქნება 50 სიმბოლო

გ. დინამიკური სტრიქონის ტევადობა იქნება 0 სიმბოლო

12.2.9. დინამიკური სტრიქონის შექმნისას კონსტრუქტორს:

ა. აუცილებლად უნდა გადავცეთ მაქსიმალური ტევადობა

ბ. არ შეგვიძლია გადავცეთ მაქსიმალური ტევადობა

გ. შეგვიძლია გადავცეთ მაქსიმალური ტევადობა

12.2.10. StringBuilder DinamiuriStriqoni3 = new StringBuilder(30, 100); კოდის შესრულების შედეგად:

ა. დინამიკური სტრიქონის ტევადობა იქნება 30 სიმბოლო, მაქსიმალური ტევადობა კი 100

ბ. დინამიკური სტრიქონის ტევადობა იქნება 100 სიმბოლო, მაქსიმალური ტევადობა კი 30

გ. დინამიკური სტრიქონის ტევადობა იქნება 30 სიმაბოლო, მაქსიმალური ტევადობა კი 30

12.2.11. შექმნის დროს დინამიკურ სტრიქონს ავტომატურად ენიჭება:

ა. 0-ის ტოლი მაქსიმალური ტევადობა

ბ. 2147483647-ის ტოლი მაქსიმალური ტევადობა

გ. 1-ის ტოლი მაქსიმალური ტევადობა

12.2.12. დინამიკური სტრიქონის შექმნისას კონსტრუქტორს:

ა. აუცილებლად უნდა გადავცეთ სტრიქონი, რომელიც დინამიკურ სტრიქონში უნდა ჩაიწეროს

ბ. არ შეგვიძლია გადავცეთ სტრიქონი, რომელიც დინამიკურ სტრიქონში უნდა ჩაიწეროს

გ. შეგვიძლია გადავცეთ სტრიქონი, რომელიც დინამიკურ სტრიქონში უნდა ჩაიწეროს

12.2.13. StringBuilder DinamiuriStriqoni4 = new StringBuilder(„კომპიუტერი“); კოდის შესრულების შედეგად:

ა. დინამიკურ სტრიქონში ჩაიწერება სტრიქონი „კომპიუტერი“

ბ. დინამიკურ სტრიქონში არ ჩაიწერება სტრიქონი „კომპიუტერი“

გ. დინამიკური სტრიქონიდან წაიშლება სტრიქონი „კომპიუტერი“

12.2.14. StringBuilder DinamiuriStriqoni6 = new StringBuilder(„კომპიუტერი“, 10, 5, 50); კოდის შესრულების შედეგად შეიქმნება დინამიკური სტრიქონი, რომელშიც:

ა. დაწყებული მე-5 პოზიციიდან ჩაიწერება მითითებული სტრიქონის 10 სიმბოლო „კომპიუტერი“

ბ. დაწყებული მე-10 პოზიციიდან ჩაიწერება მითითებული სტრიქონის 5 სიმბოლო „კომპი“

გ. დაწყებული 50-ე პოზიციიდან ჩაიწერება მითითებული სტრიქონის 10 სიმბოლო „კომპიუტერი“

12.2.15. StringBuilder კლასის Capacity თვისება:

ა. გასცემს ან აყენებს დინამიკურ სტრიქონში სიმბოლოების რაოდენობას

ბ. გასცემს დინამიკური სტრიქონის მაქსიმალურ ტევადობას

გ. გასცემს ან აყენებს დინამიკური სტრიქონის ტევადობას

12.2.16. StringBuilder კლასის Length თვისება:

ა. გასცემს ან აყენებს დინამიკურ სტრიქონში სიმბოლოების რაოდენობას

ბ. გასცემს დინამიკური სტრიქონის მაქსიმალურ ტევადობას
გ. გასცემს ან აყენებს დინამიკური სტრიქონის ტევადობას

12.2.17. StringBuilder კლასის MaxCapacity თვისება:
ა. გასცემს ან აყენებს დინამიკური სტრიქონის ტევადობას
ბ. გასცემს დინამიკური სტრიქონის მაქსიმალურ ტევადობას
გ. გასცემს ან აყენებს დინამიკურ სტრიქონში სიმბოლოების რაოდენობას

12.2.18. StringBuilder კლასის Append() მეთოდი:
ა. დინამიკურ სტრიქონს გარდაქმნის სტრიქონად
ბ. ამოწმებს, ტოლია თუ მეტი დინამიკური სტრიქონის მიმდინარე ტევადობა მითითებულ მნიშვნელობაზე
გ. ობიექტის სტრიქონულ წარმოდგენას ამატებს დინამიკური სტრიქონის ბოლოში

12.2.19. StringBuilder კლასის EnsureCapacity() მეთოდი:
ა. ამოწმებს, ტოლია თუ მეტი დინამიკური სტრიქონის მიმდინარე ტევადობა მითითებულ მნიშვნელობაზე
ბ. ობიექტის სტრიქონულ წარმოდგენას ამატებს დინამიკური სტრიქონის ბოლოში
გ. დინამიკურ სტრიქონს გარდაქმნის სტრიქონად

12.2.20. StringBuilder კლასის Equals() მეთოდი:
ა. ობიექტის სტრიქონულ წარმოდგენას ამატებს დინამიკური სტრიქონის ბოლოში
ბ. გასცემს true მნიშვნელობას, თუ დინამიკური სტრიქონი მითითებული ობიექტის ტოლია, წინააღმდეგ შემთხვევაში კი – false მნიშვნელობას
გ. ამოწმებს, ტოლია თუ მეტი დინამიკური სტრიქონის მიმდინარე ტევადობა მითითებულ მნიშვნელობაზე

12.2.21. StringBuilder კლასის Insert() მეთოდი:
ა. ობიექტის სტრიქონულ წარმოდგენას ამატებს დინამიკური სტრიქონის ბოლოში
ბ. ამოწმებს, ტოლია თუ მეტი დინამიკური სტრიქონის მიმდინარე ტევადობა მითითებულ მნიშვნელობაზე
გ. მითითებული ობიექტის სტრიქონულ წარმოდგენას ამატებს დინამიკურ სტრიქონში დაწყებული მითითებული პოზიციიდან

12.2.22. StringBuilder კლასის Remove() მეთოდი:
ა. დინამიკური სტრიქონიდან შლის მითითებული რაოდენობის სიმბოლოს დაწყებული მითითებული პოზიციიდან
ბ. ობიექტის სტრიქონულ წარმოდგენას ამატებს დინამიკური სტრიქონის ბოლოში
გ. ამოწმებს, ტოლია თუ მეტი დინამიკური სტრიქონის მიმდინარე ტევადობა მითითებულ მნიშვნელობაზე

12.2.23. StringBuilder კლასის Replace() მეთოდი:
ა. ობიექტის სტრიქონულ წარმოდგენას ამატებს დინამიკური სტრიქონის ბოლოში
ბ. მითითებულ სიმბოლოს ან ქვესტრიქონს დინამიკურ სტრიქონში ყველგან ცვლის მითითებული სიმბოლოთი ან ქვესტრიქონით
გ. დინამიკურ სტრიქონს გარდაქმნის სტრიქონად

12.2.24. StringBuilder კლასის ToString() მეთოდი:

ა. მითითებულ სიმბოლოს ან ქვესტრიქონს დინამიკურ სტრიქონში ყველგან ცვლის მითითებული სიმბოლოთი ან ქვესტრიქონით

ბ. ობიექტის სტრიქონულ წარმოდგენას ამატებს დინამიკური სტრიქონის ბოლოში

გ. დინამიკურ სტრიქონს გარდაქმნის სტრიქონად

თავი 13. თარიღი, დრო და დროის ინტერვალები

თარიღი და დრო

13.1.1. თარიღთან და დროსთან სამუშაოდ გამოიყენება:

ა. DateTime სტრუქტურა

ბ. DateTime კლასი

გ. StringBuilder კლასი

13.1.2. DateTime Tarigi = new DateTime(weli, tve, dge); კოდის შესრულებით იქმნება Tarigi სტრუქტურა, რომელიც შეიცავს:

ა. წელს, თვესა და საათს

ბ. წელს, თვესა და დღეს

გ. წელს, თვესა და წუთს

13.1.3. DateTime სტრუქტურის კონსტრუქტორს:

ა. აუცილებლად უნდა გადავცეთ საათი, წუთი, წამი და მილიწამი

ბ. არ შეიძლება გადავცეთ საათი, წუთი, წამი და მილიწამი

გ. შეიძლება გადავცეთ საათი, წუთი, წამი და მილიწამი

13.1.4. DateTime Tarigi = new DateTime(weli, tve, dge, saati, wuti, wami, miliwami); კოდის შესრულების შედეგად შეიქმნება Tarigi სტრუქტურა, რომელიც:

ა. შეიცავს წელს, თვეს, დღეს, საათს, წუთს, წამს და მილიწამს

ბ. არ შეიცავს წელს, თვეს, დღეს, საათს, წუთს, წამს და მილიწამს

გ. შეიცავს წელს, თვეს, დღეს, წამს და მილიწამს

13.1.5. DateTime სტრუქტურის Now თვისება:

ა. შეიცავს თვის დღის მნიშვნელობას 1÷31 დიაპაზონში

ბ. შეიცავს მიმდინარე თარიღსა და დროს

გ. შეიცავს საათის მნიშვნელობას 0÷23 დიაპაზონში

13.1.6. DateTime სტრუქტურის Today თვისება:

ა. შეიცავს მიმდინარე თარიღსა და დროს

ბ. შეიცავს თვის დღის მნიშვნელობას 1÷31 დიაპაზონში

გ. შეიცავს მიმდინარე თარიღს

13.1.7. DateTime სტრუქტურის Date თვისება:

ა. შეიცავს თარიღს

ბ. შეიცავს თვის დღის მნიშვნელობას 1÷31 დიაპაზონში
გ. შეიცავს მიმდინარე თარიღსა და დროს

13.1.8. DateTime სტრუქტურის Day თვისება:
ა. შეიცავს საათის მნიშვნელობას 0÷23 დიაპაზონში
ბ. შეიცავს თვის დღის მნიშვნელობას 1÷31 დიაპაზონში
გ. შეიცავს მიმდინარე თარიღსა და დროს

13.1.9. DateTime სტრუქტურის DayOfWeek თვისება:
ა. შეიცავს მიმდინარე თარიღსა და დროს
ბ. შეიცავს თვის დღის მნიშვნელობას 1÷31 დიაპაზონში
გ. შეიცავს კვირის დღის მნიშვნელობას 0 (კვირა) ÷ 6 (შაბათი) დიაპაზონში

13.1.10. DateTime სტრუქტურის DayOfWeek თვისება:
ა. შეიცავს წლის დღის ნომერს 1÷366 დიაპაზონში
ბ. შეიცავს მიმდინარე თარიღსა და დროს
გ. შეიცავს თვის დღის მნიშვნელობას 1÷31 დიაპაზონში

13.1.11. DateTime სტრუქტურის Hour თვისება:
ა. შეიცავს მილიწამის მნიშვნელობას 0÷999 დიაპაზონში
ბ. შეიცავს საათის მნიშვნელობას 0÷23 დიაპაზონში
გ. შეიცავს მიმდინარე თარიღსა და დროს

13.1.12. DateTime სტრუქტურის Millisecond თვისება:
ა. შეიცავს მიმდინარე თარიღსა და დროს
ბ. შეიცავს საათის მნიშვნელობას 0÷23 დიაპაზონში
გ. შეიცავს მილიწამის მნიშვნელობას 0÷999 დიაპაზონში

13.1.13. DateTime სტრუქტურის Minute თვისება:
ა. შეცავს წუთის მნიშვნელობას 0÷59 დიაპაზონში
ბ. შეიცავს საათის მნიშვნელობას 0÷23 დიაპაზონში
გ. შეიცავს მილიწამის მნიშვნელობას 0÷999 დიაპაზონში

13.1.14. DateTime სტრუქტურის Month თვისება:
ა. შეიცავს მილიწამის მნიშვნელობას 0÷999 დიაპაზონში
ბ. შეიცავს თვის მნიშვნელობას 1÷12 დიაპაზონში
გ. შეიცავს საათის მნიშვნელობას 0÷23 დიაპაზონში

13.1.15. DateTime სტრუქტურის Second თვისება:
ა. შეიცავს მილიწამის მნიშვნელობას 0÷999 დიაპაზონში
ბ. შეიცავს საათის მნიშვნელობას 0÷23 დიაპაზონში
გ. შეიცავს წამის მნიშვნელობას 0÷59 დიაპაზონში

13.1.16. DateTime სტრუქტურის TimeOfDay თვისება:
ა. შეიცავს შუაღამიდან გასულ დროს
ბ. შეიცავს თვის მნიშვნელობას 1÷12 დიაპაზონში
გ. შეიცავს მილიწამის მნიშვნელობას 0÷999 დიაპაზონში

13.1.17. DateTime სტრუქტურის Year თვისება:

ა. შეიცავს შუაღამიდან გასულ დროს

ბ. შეიცავს წლის მნიშვნელობას 1÷9999 დიაპაზონში

გ. შეიცავს მილიწამის მნიშვნელობას 0÷999 დიაპაზონში

13.1.18. DateTime სტრუქტურის DaysInMonth() მეთოდი:

ა. DateTime სტრუქტურის ეგზემპლარს უმატებს წლების მითითებულ რაოდენობას

ბ. გასცემს true-ს, თუ წელი ნაკიანია, წინააღმდეგ შემთხვევაში false-ს

გ. გასცემს დღეების რაოდენობას მითითებული წლის მითითებული თვისათვის

13.1.19. DateTime სტრუქტურის Equals() მეთოდი:

ა. ტოლობაზე ადარებს ორ DateTime სტრუქტურას

ბ. გასცემს დღეების რაოდენობას მითითებული წლის მითითებული თვისათვის

გ. გასცემს true-ს, თუ წელი ნაკიანია, წინააღმდეგ შემთხვევაში false-ს

13.1.20. DateTime სტრუქტურისIsLeapYear() მეთოდი:

ა. გასცემს დღეების რაოდენობას მითითებული წლის მითითებული თვისათვის

ბ. გასცემს true-ს, თუ წელი ნაკიანია, წინააღმდეგ შემთხვევაში false-ს

გ. DateTime სტრუქტურის ეგზემპლარს უმატებს წლების მითითებულ რაოდენობას

13.1.21. DateTime სტრუქტურის AddDays() მეთოდი:

ა. გასცემს true-ს, თუ წელი ნაკიანია, წინააღმდეგ შემთხვევაში false-ს

ბ. გასცემს დღეების რაოდენობას მითითებული წლის მითითებული თვისათვის

გ. DateTime სტრუქტურის ეგზემპლარს უმატებს დღეების მითითებულ რაოდენობას

13.1.22. DateTime სტრუქტურის AddHours() მეთოდი:

ა. DateTime სტრუქტურის ეგზემპლარს უმატებს საათების მითითებულ რაოდენობას

ბ. გასცემს true-ს, თუ წელი ნაკიანია, წინააღმდეგ შემთხვევაში false-ს

გ. გასცემს დღეების რაოდენობას მითითებული წლის მითითებული თვისათვის

13.1.23. DateTime სტრუქტურის AddMinutes() მეთოდი:

ა. გასცემს დღეების რაოდენობას მითითებული წლის მითითებული თვისათვის

ბ. DateTime სტრუქტურის ეგზემპლარს უმატებს წუთების მითითებულ რაოდენობას

გ. გასცემს true-ს, თუ წელი ნაკიანია, წინააღმდეგ შემთხვევაში false-ს

13.1.24. DateTime სტრუქტურის AddMonths() მეთოდი:

ა. DateTime სტრუქტურის ეგზემპლარს უმატებს წლების მითითებულ რაოდენობას

ბ. გასცემს true-ს, თუ წელი ნაკიანია, წინააღმდეგ შემთხვევაში false-ს

გ. DateTime სტრუქტურის ეგზემპლარს უმატებს თვეების მითითებულ რაოდენობას

13.1.25. DateTime სტრუქტურის AddSeconds() მეთოდი:

ა. DateTime სტრუქტურის ეგზემპლარს უმატებს წლების მითითებულ რაოდენობას

ბ. DateTime სტრუქტურის ეგზემპლარს უმატებს წამების მითითებულ რაოდენობას

გ. გასცემს true-ს, თუ წელი ნაკიანია, წინააღმდეგ შემთხვევაში false-ს

13.1.26. DateTime სტრუქტურის AddYears() მეთოდი:

ა. DateTime სტრუქტურის ეგზემპლარს უმატებს დღეების მითითებულ რაოდენობას

ბ. გასცემს true-ს, თუ წელი ნაკიანია, წინააღმდეგ შემთხვევაში false-ს
გ. DateTime სტრუქტურის ეგზემპლარს უმატებს წლების მითითებულ რაოდენობას

13.1.27. DateTime სტრუქტურის Subtract() მეთოდი:
ა. DateTime სტრუქტურის ეგზემპლარს აკლებს ამავე სტრუქტურის ეგზემპლარს ან TimeSpan კლასის ეგზემპლარს
ბ. გასცემს true-ს, თუ წელი ნაკიანია, წინააღმდეგ შემთხვევაში false-ს
გ. DateTime სტრუქტურის ეგზემპლარს უმატებს წლების მითითებულ რაოდენობას

13.1.28. DateTime სტრუქტურის Add() მეთოდი:
ა. DateTime სტრუქტურის ეგზემპლარს უმატებს დღეების მითითებულ რაოდენობას
ბ. გასცემს true-ს, თუ წელი ნაკიანია, წინააღმდეგ შემთხვევაში false-ს
გ. DateTime სტრუქტურის ეგზემპლარს უმატებს TimeSpan კლასის ეგზემპლარს

დროის ინტერვალები

13.2.1. დროის ინტერვალებთან სამუშაოდ გამოიყენება:
ა. DateTime კლასი
ბ. TimeSpan კლასი
გ. DateTime სტრუქტურა

13.2.2. TimeSpan Drois_Intervali1 = new TimeSpan(5, 15, 45); კოდის შესრულების შედეგად:
ა. შეიქმნება დროის ინტერვალი, რომელიც არ შეიცავს საათს, წუთსა და წამს
ბ. არ შეიქმნება დროის ინტერვალი, რომელიც შეიცავს საათს, წუთსა და წამს
გ. შეიქმნება დროის ინტერვალი, რომელიც შეიცავს საათს, წუთსა და წამს

13.2.3. TimeSpan Drois_Intervali1 = new TimeSpan(5, 15, 45); კოდის შესრულების შედეგად:
ა. შეიქმნება დროის ინტერვალი, რომელიც შეიცავს 5 საათს, 15 წუთსა და 45 წამს
ბ. შეიქმნება დროის ინტერვალი, რომელიც შეიცავს 15 საათს, 45 წუთსა და 5 წამს
გ. შეიქმნება დროის ინტერვალი, რომელიც შეიცავს 45 საათს, 15 წუთსა და 5 წამს

13.2.4. TimeSpan Drois_Intervali2 = new TimeSpan(2, 5, 15, 45); კოდის შესრულების შედეგად:
ა. შეიქმნება დროის ინტერვალი, რომელიც შეიცავს 2 დღეს, 5 საათს, 15 წუთსა და 45 წამს
ბ. შეიქმნება დროის ინტერვალი, რომელიც შეიცავს 2 დღეს, 15 საათს, 5 წუთსა და 45 წამს
გ. შეიქმნება დროის ინტერვალი, რომელიც შეიცავს 45 დღეს, 5 საათს, 15 წუთსა და 2 წამს

13.2.5.
```
DateTime Tarigi_Dro = new DateTime(2007, 04, 12);
TimeSpan Drois_Intervali1 = new TimeSpan(2, 5, 15, 45);
Tarigi_Dro += Drois_Intervali1;
```
კოდის შესრულების შედეგად:
ა. Tarigi_Dro სტრუქტურას გამოაკლდება Drois_Intervali1 ინტერვალი
ბ. Tarigi_Dro სტრუქტურას დაემატება Drois_Intervali1 ინტერვალი
გ. Tarigi_Dro სტრუქტურა გამრავლდება Drois_Intervali1 ინტერვალზე

13.2.6.
DateTime Tarigi_Dro = new DateTime(2010, 04, 12);
TimeSpan Drois_Intervali1 = new TimeSpan(4, 5, 15, 45);
Tarigi_Dro += Drois_Intervali1;
კოდის შესრულების შედეგად Tarigi_Dro სტრუქტურა მიიღებს მნიშვნელობას:
ა. 16.04.2010 45:15:05
ბ. 04.16.2010 05:15:45
გ. 16.04.2010 05:15:45

13.2.7.
TimeSpan Drois_Intervali1 = new TimeSpan(10, 5, 15, 45);
label1.Text = Drois_Intervali1.Days.ToString();
კოდის შესრულების შედეგად გაიცემა:
ა. 10
ბ. 5
გ. 15

13.2.8.
TimeSpan Drois_Intervali1 = new TimeSpan(5, 15, 45);
label1.Text = Drois_Intervali1.Days.ToString();
კოდის შესრულების შედეგად გაიცემა:
ა. 45
ბ. 0
გ. 5

13.2.9. TimeSpan კლასის Days თვისება:
ა. შეიცავს წუთების რაოდენობას TimeSpan კლასის ობიექტში
ბ. შეიცავს წამების რაოდენობას TimeSpan კლასის ობიექტში
გ. შეიცავს დღეების რაოდენობას TimeSpan კლასის ობიექტში

13.2.10. TimeSpan კლასის Hours თვისება:
ა. შეიცავს საათების რაოდენობას TimeSpan კლასის ობიექტში
ბ. შეიცავს წამების რაოდენობას TimeSpan კლასის ობიექტში
გ. შეიცავს დღეების რაოდენობას TimeSpan კლასის ობიექტში

13.2.11. TimeSpan კლასის Minutes თვისება:
ა. შეიცავს წამების რაოდენობას TimeSpan კლასის ობიექტში
ბ. შეიცავს წუთების რაოდენობას TimeSpan კლასის ობიექტში
გ. შეიცავს დღეების რაოდენობას TimeSpan კლასის ობიექტში

13.2.12. TimeSpan კლასის Seconds თვისება:
ა. შეიცავს დღეების რაოდენობას TimeSpan კლასის ობიექტში
ბ. შეიცავს წუთების რაოდენობას TimeSpan კლასის ობიექტში
გ. შეიცავს წამების რაოდენობას TimeSpan კლასის ობიექტში

13.2.13. TimeSpan კლასის Equals() მეთოდი:
ა. ადარებს TimeSpan კლასის ორ ობიექტს. გასცემს შესაბამის ლოგიკურ მნიშვნელობას

ბ. TimeSpan კლასის მოცემულ ობიექტს აკლებს ამავე კლასის ობიექტს
გ. TimeSpan კლასის მოცემულ ობიექტს უმატებს ამავე კლასის ობიექტს

13.2.14. TimeSpan კლასის Add() მეთოდი:
ა. TimeSpan კლასის მოცემულ ობიექტს აკლებს ამავე კლასის ობიექტს
ბ. TimeSpan კლასის მოცემულ ობიექტს უმატებს ამავე კლასის ობიექტს
გ. ადარებს TimeSpan კლასის ორ ობიექტს. გასცემს შესაბამის ლოგიკურ მნიშვნელობას

13.2.15. TimeSpan კლასის Subtract() მეთოდი:
ა. ადარებს TimeSpan კლასის ორ ობიექტს. გასცემს შესაბამის ლოგიკურ მნიშვნელობას
ბ. TimeSpan კლასის მოცემულ ობიექტს აკლებს ამავე კლასის ობიექტს

თავი 14. კოლექციები
დინამიკური მასივები

14.1.1. კოლექციების აღმწერი კლასები გამოცხადებულია:
ა. System.Collections სახელების სივრცეში
ბ. System.IO.Ports სახელების სივრცეში
გ. System.Drawing სახელების სივრცეში

14.1.2. ჩვეულებრივი მასივებისაგან განსხვავებით, დინამიკური მასივი მასში ელემენტების დამატების ან წაშლის შემთხვევაში:
ა. ზომას არ იცვლის
ბ. ზომას იცვლის
გ. ზომას ხანდახან იცვლის

14.1.3. დინამიკურ მასივებში ელემენტების ნუმერაცია:
ა. ერთიდან იწყება
ბ. ნულიდან არ იწყება
გ. ნულიდან იწყება

14.1.4. თვისებები და მეთოდები დინამიკური მასივების ელემენტებთან სამუშაოდ განსაზღვრულია:
ა. ArrayList კლასში
ბ. Hashtable კლასში
გ. SortedList კლასში

14.1.5. ArrayList კლასის Capacity თვისება:
ა. მიუთითებს, არის თუ არა დინამიკური მასივი ფიქსირებული ზომის
ბ. განსაზღვრავს დინამიკური მასივის ტევადობას, ანუ ელემენტების მაქსიმალურ რაოდენობას
გ. დინამიკური მასივიდან შლის ყველა ელემენტს

14.1.6. ArrayList კლასის Count თვისება:
ა. მიუთითებს, არის თუ არა დინამიკური მასივი ფიქსირებული ზომის
ბ. დინამიკური მასივიდან შლის ყველა ელემენტს
გ. შეიცავს დინამიკურ მასივში ელემენტების რაოდენობას

14.1.7. ArrayList კლასის IsFixedSize თვისება:

ა. მიუთითებს, არის თუ არა დინამიკური მასივი ფიქსირებული ზომის

ბ. დინამიკურ მასივს ბოლოში უმატებს ელემენტს

გ. დინამიკური მასივიდან შლის ყველა ელემენტს

14.1.8. ArrayList კლასის IsReadOnly თვისება:

ა. დინამიკური მასივიდან შლის ყველა ელემენტს

ბ. მიუთითებს, არის თუ არა დინამიკური მასივი მხოლოდ წაკითხვადი

გ. შეიცავს დინამიკურ მასივში ელემენტების რაოდენობას

14.1.9. ArrayList კლასის Add() მეთოდი:

ა. შეიცავს დინამიკურ მასივში ელემენტების რაოდენობას

ბ. დინამიკური მასივიდან შლის ყველა ელემენტს

გ. დინამიკურ მასივს ბოლოში უმატებს ელემენტს

14.1.10. ArrayList კლასის AddRange() მეთოდი:

ა. დინამიკურ მასივს ბოლოში უმატებს სხვა კოლექციის ელემენტებს

ბ. შეიცავს დინამიკურ მასივში ელემენტების რაოდენობას

გ. დინამიკური მასივიდან შლის ყველა ელემენტს

14.1.11. ArrayList კლასის Clear() მეთოდი:

ა. დინამიკურ მასივს ბოლოში უმატებს ელემენტს

ბ. დინამიკური მასივიდან შლის ყველა ელემენტს

გ. შეიცავს დინამიკურ მასივში ელემენტების რაოდენობას

14.1.12. დინამიკურ მასივს :

ა. უნდა დავუმატოთ ერთი ტიპის მონაცემები

ბ. არ შეგვიძლია დავუმატოთ სხვადასხვა ტიპის მონაცემები

გ. შეგვიძლია დავუმატოთ სხვადასხვა ტიპის მონაცემები

14.1.13. ArrayList კლასის Contains() მეთოდი:

ა. განსაზღვრავს შეიცავს თუ არა დინამიკური მასივი მითითებულ ელემენტს

ბ. განსაზღვრავს ტოლია თუ არა ორი დინამიკური მასივი

გ. გასცემს ნაპოვნი ელემენტის პოზიციის ნომერს

14.1.14. ArrayList კლასის Equals() მეთოდი:

ა. განსაზღვრავს, შეიცავს თუ არა დინამიკური მასივი მითითებულ ელემენტს

ბ. განსაზღვრავს, ტოლია თუ არა ორი დინამიკური მასივი

გ. ახდენს დინამიკურ მასივში ელემენტის ჩასმას მითითებული პოზიციიდან

14.1.15. ArrayList კლასის GetRange() მეთოდი:

ა. ახდენს დინამიკურ მასივში ელემენტის ჩასმას მითითებული პოზიციიდან

ბ. განსაზღვრავს, ტოლია თუ არა ორი დინამიკური მასივი

გ. გასცემს დინამიკურ მასივს, რომელიც შეიცავს საწყისი მასივის ელემენტების დიაპაზონს

14.1.16. ArrayList კლასის IndexOf() მეთოდი:

ა. გასცემს ნაპოვნი ელემენტის პოზიციის ნომერს

ბ. განსაზღვრავს, ტოლია თუ არა ორი დინამიკური მასივი

გ. ახდენს დინამიკურ მასივში ელემენტის ჩასმას მითითებული პოზიციიდან

14.1.17. ArrayList კლასის Insert() მეთოდი:

ა. განსაზღვრავს, ტოლია თუ არა ორი დინამიკური მასივი

ბ. ახდენს დინამიკურ მასივში ელემენტის ჩასმას მითითებული პოზიციიდან

გ. ალაგებს დინამიკური მასივის ელემენტებს უკუმიმდევრობით

14.1.18. ArrayList კლასის InsertRange() მეთოდი:

ა. ახდენს დინამიკურ მასივში კოლექციის ელემენტების ჩასმას მითითებული პოზიციიდან

ბ. ალაგებს დინამიკური მასივის ელემენტებს უკუმიმდევრობით

გ. დახარისხებულ მასივში ასრულებს მითითებული ელემენტის ორობით ძებნას

14.1.19. ArrayList კლასის BinarySearch() მეთოდი:

ა. დინამიკური მასივის ელემენტებს გადაწერს ჩვეულებრივ მასივში

ბ. დინამიკური მასივის შესაბამის დიაპაზონში ახდენს კოლექციის ელემენტების ჩასმას

გ. დახარისხებულ მასივში ასრულებს მითითებული ელემენტის ორობით ძებნას

14.1.20. ArrayList კლასის LastIndexOf() მეთოდი:

ა. გასცემს ნაპოვნი ელემენტის პირველი პოზიციის ნომერს

ბ. გასცემს ნაპოვნი ელემენტის უკანასკნელი პოზიციის ნომერს

გ. დინამიკური მასივის ელემენტებს უკუმიმდევრობით ალაგებს

14.1.21. ArrayList კლასის Remove() მეთოდი:

ა. დინამიკური მასივის ელემენტებს გადაწერს ჩვეულებრივ მასივში

ბ. გასცემს ნაპოვნი ელემენტის უკანასკნელი პოზიციის ნომერს

გ. დინამიკური მასივიდან შლის პირველ ნაპოვნ ელემენტს

14.1.22. ArrayList კლასის RemoveAt() მეთოდი:

ა. დინამიკური მასივიდან შლის მითითებული პოზიციის მქონე ელემენტს

ბ. ალაგებს დინამიკური მასივის ელემენტებს უკუმიმდევრობით

გ. გასცემს ნაპოვნი ელემენტის უკანასკნელი პოზიციის ნომერს

14.1.23. ArrayList კლასის RemoveRange() მეთოდი:

ა. დინამიკური მასივის შესაბამის დიაპაზონში ახდენს კოლექციის ელემენტების ჩასმას

ბ. დინამიკური მასივიდან შლის ელემენტების დიაპაზონს დაწყებული მითითებული პოზიციიდან

გ. გასცემს ნაპოვნი ელემენტის უკანასკნელი პოზიციის ნომერს

14.1.24. ArrayList კლასის Reverse() მეთოდი:

ა. დინამიკური მასივის ელემენტებს გადაწერს ჩვეულებრივ მასივში

ბ. დინამიკური მასივის შესაბამის დიაპაზონში ახდენს კოლექციის ელემენტების ჩასმას

გ. ალაგებს დინამიკური მასივის ელემენტებს უკუმიმდევრობით

14.1.25. ArrayList კლასის SetRange() მეთოდი:

ა. დინამიკური მასივის შესაბამის დიაპაზონში ახდენს კოლექციის ელემენტების ჩასმას

ბ. დახარისხებულ მასივში ასრულებს მითითებული ელემენტის ორობით ძებნას
გ. გასცემს ნაპოვნი ელემენტის უკანასკნელი პოზიციის ნომერს

14.1.26. ArrayList კლასის Sort() მეთოდი:
ა. დინამიკური მასივის ელემენტებს გადაწერს ჩვეულებრივ მასივში
ბ. ახდენს დინამიკური მასივის ან მისი დიაპაზონის დახარისხებას
გ. დახარისხებულ მასივში ასრულებს მითითებული ელემენტის ორობით ძებნას

14.1.27. ArrayList კლასის ToArray() მეთოდი:
ა. გასცემს ნაპოვნი ელემენტის უკანასკნელი პოზიციის ნომერს
ბ. დახარისხებულ მასივში ასრულებს მითითებული ელემენტის ორობით ძებნას
გ. დინამიკური მასივის ელემენტებს გადაწერს ჩვეულებრივ მასივში

ჰეშ-ცხრილები

14.2.1. ჰეშ-ცხრილები ინახავს:
ა. წყვილებს გასაღები-მნიშვნელობა
ბ. სამეულებს გასაღები-მნიშვნელობა-გასაღები
გ. წყვილებს გასაღები-მნიშვნელობა

14.2.2. ჰეშ-ცხრილის თითოეული ელემენტი შედგება:
ა. მხოლოდ გასაღებისაგან
ბ. გასაღებისა და მნიშვნელობისაგან
გ. მხოლოდ მნიშვნელობისაგან

14.2.3. ჰეშ-ცხრილის გასაღები გამოიყენება:
ა. შეუსაბამო გასაღების მისაღებად
ბ. შესაბამისი გასაღების მისაღებად
გ. შესაბამისი მნიშვნელობის მისაღებად

14.2.4. ჰეშ-ცხრილები განსაზღვრულია:
ა. Hashtable კლასში
ბ. ArrayList კლასში
გ. SortedList კლასში

14.2.5. ჰეშ-ცხრილში გასაღები და მნიშვნელობა შეიძლება იყოს:
ა. მხოლოდ მთელი რიცხვი
ბ. ნებისმიერი ტიპის ობიექტი
გ. მხოლოდ სტრიქონი

14.2.6. Hashtable კლასის Count თვისება:
ა. გასცემს კოლექციას, რომელიც ჰეშ-ცხრილის მნიშვნელობებისაგან შედგება
ბ. გასცემს კოლექციას, რომელიც ჰეშ-ცხრილის გასაღებებისაგან შედგება
გ. შეიცავს ჰეშ-ცხრილში შენახული ელემენტების რაოდენობას

14.2.7. Hashtable კლასის IsFixedSize თვისება:
ა. ამოწმებს, აქვს თუ არა ჰეშ-ცხრილს ფიქსირებული სიგრძე

ბ. გასცემს კოლექციას, რომელიც ჰეშ-ცხრილის მნიშვნელობებისაგან შედგება
გ. გასცემს კოლექციას, რომელიც ჰეშ-ცხრილის გასაღებებისაგან შედგება

14.2.8. Hashtable კლასის IsReadOnly თვისება:
ა. გასცემს კოლექციას, რომელიც ჰეშ-ცხრილის მნიშვნელობებისაგან შედგება
ბ. ამოწმებს, არის თუ არა ჰეშ-ცხრილი მხოლოდ წაკითხვადი
გ. გასცემს კოლექციას, რომელიც ჰეშ-ცხრილის გასაღებებისაგან შედგება

14.2.9. Hashtable კლასის Keys თვისება:
ა. შეიცავს ჰეშ-ცხრილში შენახული ელემენტების რაოდენობას
ბ. გასცემს კოლექციას, რომელიც ჰეშ-ცხრილის მნიშვნელობებისაგან შედგება
გ. გასცემს კოლექციას, რომელიც ჰეშ-ცხრილის გასაღებებისაგან შედგება

14.2.10. Hashtable კლასის Values თვისება:
ა. გასცემს კოლექციას, რომელიც ჰეშ-ცხრილის მნიშვნელობებისაგან შედგება
ბ. გასცემს კოლექციას, რომელიც ჰეშ-ცხრილის გასაღებებისაგან შედგება
გ. შეიცავს ჰეშ-ცხრილში შენახული ელემენტების რაოდენობას

14.2.11. Hashtable კლასის Add() მეთოდი:
ა. ჰეშ-ცხრილიდან შლის ყველა ელემენტს
ბ. ჰეშ-ცხრილს უმატებს მითითებული გასაღებისა და მნიშვნელობის მქონე ელემენტს
გ. ამოწმებს, ტოლია თუ არა ორი ჰეშ-ცხრილი

14.2.12. Hashtable კლასის Clear() მეთოდი:
ა. ჰეშ-ცხრილს უმატებს მითითებული გასაღებისა და მნიშვნელობის მქონე ელემენტს
ბ. ამოწმებს, ტოლია თუ არა ორი ჰეშ-ცხრილი
გ. ჰეშ-ცხრილიდან შლის ყველა ელემენტს

14.2.13. Hashtable კლასის Contains() მეთოდი:
ა. განსაზღვრავს, შეიცავს თუ არა ჰეშ-ცხრილი მითითებულ გასაღებს
ბ. ჰეშ-ცხრილიდან შლის მითითებული გასაღების მქონე ელემენტს
გ. ამოწმებს, ტოლია თუ არა ორი ჰეშ-ცხრილი

14.2.14. Hashtable კლასის ContainsKey() მეთოდი:
ა. ამოწმებს, ტოლია თუ არა ორი ჰეშ-ცხრილი
ბ. განსაზღვრავს, შეიცავს თუ არა ჰეშ-ცხრილი მითითებულ გასაღებს
გ. ჰეშ-ცხრილიდან შლის მითითებული გასაღების მქონე ელემენტს

14.2.15. Hashtable კლასის ContainsValue() მეთოდი:
ა. ამოწმებს, ტოლია თუ არა ორი ჰეშ-ცხრილი
ბ. ჰეშ-ცხრილიდან შლის მითითებული გასაღების მქონე ელემენტს
გ. განსაზღვრავს, შეიცავს თუ არა ჰეშ-ცხრილი მითითებულ მნიშვნელობას

14.2.16. Hashtable კლასის CopyTo() მეთოდი:
ა. ერთგანზომილებიან მასივში ჰეშ-ცხრილიდან გადაწერს გასაღებებს ან მნიშვნელობებს
ბ. ჰეშ-ცხრილიდან შლის მითითებული გასაღების მქონე ელემენტს
გ. ამოწმებს, ტოლია თუ არა ორი ჰეშ-ცხრილი

14.2.17. Hashtable კლასის Equals() მეთოდი:
ა. ჰეშ-ცხრილიდან შლის ყველა ელემენტს
ბ. ამოწმებს, ტოლია თუ არა ორი ჰეშ-ცხრილი
გ. ჰეშ-ცხრილიდან შლის მითითებული გასაღების მქონე ელემენტს

14.2.18. Hashtable კლასის Remove() მეთოდი:
ა. ჰეშ-ცხრილიდან შლის ყველა ელემენტს
ბ. ამოწმებს, ტოლია თუ არა ორი ჰეშ-ცხრილი
გ. ჰეშ-ცხრილიდან შლის მითითებული გასაღების მქონე ელემენტს

დახარისხებული სიები

14.3.1. დახარისხებული სია არის:
ა. დინამიკური მასივისა და ჰეშ-ცხრილის კომბინაცია
ბ. დინამიკური მასივისა და სტეკის კომბინაცია
გ. დინამიკური რიგისა და ჰეშ-ცხრილის კომბინაცია

14.3.2. დახარისხებული სიის ელემენტის მიღება შეიძლება:
ა. როგორც ინდექსის, ისე გასაღების მიხედვით
ბ. მხოლოდ ინდექსის მიხედვით
გ. მხოლოდ გასაღების მიხედვით

14.3.3. გასაღებები ასეთ მასივებში დახარისხებულია:
ა. კლებადობის მიხედვით
ბ. ზრდადობის მიხედვით
გ. როგორც კლებადობის, ისე ზრდადობის მიხედვით

14.3.4. მასივში ახალი ელემენტის ჩამატებისას ის იმ პოზიციაში მოთავსდება, რომ მასივი:
ა. უკუმიმდევრობით დალაგდეს
ბ. დაუხარისხებელი დარჩეს
გ. დახარისხებული დარჩეს

14.3.5. დახარისხებული სია განსაზღვრულია:
ა. SortedList კლასში
ბ. Hashtable კლასში
გ. ArrayList კლასში

14.3.6. SortedList კლასის Capacity თვისება:
ა. გასცემს დახარისხებული სიის მნიშვნელობებისაგან შემდგარ კოლექციას
ბ. განსაზღვრავს დახარისხებული სიის ტევადობას
გ. გასცემს დახარისხებული სიის გასაღებებისაგან შემდგარ კოლექციას

14.3.7. SortedList კლასის Count თვისება:
ა. გასცემს დახარისხებული სიის გასაღებებისაგან შემდგარ კოლექციას
ბ. გასცემს დახარისხებული სიის მნიშვნელობებისაგან შემდგარ კოლექციას
გ. შეიცავს დახარისხებულ სიაში რეალურად არსებული ელემენტების რაოდენობას

14.3.8. SortedList კლასის IsFixedSize თვისება:
ა. ამოწმებს, აქვს თუ არა დახარისხებულ მასივს ფიქსირებული სიგრძე
ბ. გასცემს დახარისხებული სიის გასაღებებისაგან შემდგარ კოლექციას
გ. გასცემს დახარისხებული სიის მნიშვნელობებისაგან შემდგარ კოლექციას

14.3.9. SortedList კლასის IsReadOnly თვისება:
ა. გასცემს დახარისხებული სიის გასაღებებისაგან შემდგარ კოლექციას
ბ. ამოწმებს, არის თუ არა დახარისხებული სია მხოლოდ წაკითხვადი
გ. გასცემს დახარისხებული სიის მნიშვნელობებისაგან შემდგარ კოლექციას

14.3.10. SortedList კლასის Keys თვისება:
ა. შეიცავს დახარისხებულ სიაში რეალურად არსებული ელემენტების რაოდენობას
ბ. გასცემს დახარისხებული სიის მნიშვნელობებისაგან შემდგარ კოლექციას
გ. გასცემს დახარისხებული სიის გასაღებებისაგან შემდგარ კოლექციას

14.3.11. SortedList კლასის Values თვისება:
ა. შეიცავს დახარისხებულ სიაში რეალურად არსებული ელემენტების რაოდენობას
ბ. გასცემს დახარისხებული სიის გასაღებებისაგან შემდგარ კოლექციას
გ. გასცემს დახარისხებული სიის მნიშვნელობებისაგან შემდგარ კოლექციას

14.3.12. SortedList კლასის Add() მეთოდი:
ა. დახარისხებულ სიას ელემენტს უმატებს
ბ. დახარისხებული სიიდან შლის ყველა ელემენტს
გ. მითითებულ გასაღებს ეძებს დახარისხებულ სიაში

14.3.13. SortedList კლასის Clear() მეთოდი:
ა. დახარისხებულ სიას ელემენტს უმატებს
ბ. დახარისხებული სიიდან შლის ყველა ელემენტს
გ.

14.3.14. SortedList კლასის Contains() მეთოდი:
ა. მითითებულ გასაღებს ეძებს დახარისხებულ სიაში
ბ. დახარისხებულ სიას ელემენტს უმატებს
გ. მითითებულ გასაღებს ეძებს დახარისხებულ სიაში

14.3.15. SortedList კლასის ContainsKey() მეთოდი:
ა. მითითებულ გასაღებს ეძებს დახარისხებულ სიაში
ბ. მითითებულ მნიშვნელობას ეძებს დახარისხებულ სიაში
გ. მითითებულ გასაღებს ეძებს დახარისხებულ სიაში

14.3.16. SortedList კლასის ContainsValue() მეთოდი:
ა. მითითებულ გასაღებს ეძებს დახარისხებულ სიაში
ბ. მითითებულ მნიშვნელობას ეძებს დახარისხებულ სიაში
გ. ამოწმებს, ტოლია თუ არა ორი ობიექტი

14.3.17. SortedList კლასის Equals() მეთოდი:
ა. მითითებულ მნიშვნელობას ეძებს დახარისხებულ სიაში

ბ. გასცემს დახარისხებული სიის ყველა გასაღებს
გ. ამოწმებს, ტოლია თუ არა ორი ობიექტი

14.3.18. SortedList კლასის GetByIndex() მეთოდი:
ა. დახარისხებული სიიდან გასცემს მითითებული ინდექსის მქონე მნიშვნელობას
ბ. ამოწმებს, ტოლია თუ არა ორი ობიექტი
გ. გასცემს დახარისხებული სიის ყველა გასაღებს

14.3.19. SortedList კლასის GetKey() მეთოდი:
ა. გასცემს დახარისხებული სიის ყველა გასაღებს
ბ. გასცემს დახარისხებული სიის მითითებული პოზიციის გასაღებს
გ. ამოწმებს, ტოლია თუ არა ორი ობიექტი

14.3.20. SortedList კლასის GetKeyList() მეთოდი:
ა. მითითებულ მნიშვნელობას ეძებს დახარისხებულ სიაში
ბ. ამოწმებს, ტოლია თუ არა ორი ობიექტი
გ. გასცემს დახარისხებული სიის ყველა გასაღებს

14.3.21. SortedList კლასის GetValueList() მეთოდი:
ა. გასცემს დახარისხებული სიის ყველა მნიშვნელობას
ბ. ცვლის მნიშვნელობას ინდექსით მითითებულ პოზიციაში
გ. გასცემს დახარისხებული სიის ყველა გასაღებს

14.3.22. SortedList კლასის IndexOfKey() მეთოდი:
ა. ცვლის მნიშვნელობას ინდექსით მითითებულ პოზიციაში
ბ. გასცემს მითითებული გასაღების ინდექსს
გ. გასცემს დახარისხებული სიის ყველა მნიშვნელობას

14.3.23. SortedList კლასის IndexOfValue() მეთოდი:
ა. გასცემს დახარისხებული სიის ყველა მნიშვნელობას
ბ. ცვლის მნიშვნელობას ინდექსით მითითებულ პოზიციაში
გ. გასცემს მითითებული მნიშვნელობის ინდექსს

14.3.24. SortedList კლასის Remove() მეთოდი:
ა. დახარისხებული სიიდან შლის მითითებული გასაღების შემცველ ელემენტს
ბ. ცვლის მნიშვნელობას ინდექსით მითითებულ პოზიციაში
გ. გასცემს დახარისხებული სიის ყველა მნიშვნელობას

14.3.25. SortedList კლასის RemoveAt() მეთოდი:
ა. ცვლის მნიშვნელობას ინდექსით მითითებულ პოზიციაში
ბ. დახარისხებული სიიდან შლის მითითებული მნიშვნელობის შემცველ ელემენტს
გ. გასცემს დახარისხებული სიის ყველა მნიშვნელობას

14.3.26. SortedList კლასის SetByIndex() მეთოდი:
ა. გასცემს დახარისხებული სიის ყველა გასაღებს
ბ. გასცემს დახარისხებული სიის ყველა მნიშვნელობას
გ. ცვლის მნიშვნელობას ინდექსით მითითებულ პოზიციაში

რიგები

14.4.1. რიგი არის მეხსიერების უბანი, რომელიც მუშაობს პრინციპით:
ა. "პირველი მოვიდა – პირველი წავიდა"
ბ. "პირველი წავიდა – პირველი მოვიდა "
გ. "პირველი მოვიდა – უკანასკნელი წავიდა"

14.4.2. რიგი განსაზღვრულია:
ა. SortedList კლასში
ბ. Queue კლასში
გ. Hashtable კლასში

14.4.3. Queue კლასის Count თვისება:
ა. გასცემს დახარისხებული სიის ყველა მნიშვნელობას
ბ. ამოწმებს, არის თუ არა რიგი მხოლოდ წაკითხვადი
გ. შეიცავს რიგში რეალურად არსებული ელემენტების რაოდენობას

14.4.4. Queue კლასის IsReadOnly თვისება:
ა. ამოწმებს, არის თუ არა რიგი მხოლოდ წაკითხვადი
ბ. შეიცავს რიგში რეალურად არსებული ელემენტების რაოდენობას
გ. გასცემს დახარისხებული სიის ყველა მნიშვნელობას

14.4.5. Queue კლასის Clear() მეთოდი:
ა. რიგის ელემენტებს გადაწერს მასივში
ბ. რიგიდან ყველა ელემენტს შლის
გ. გასცემს რიგის პირველი ელემენტის მნიშვნელობას და მას რიგიდან შლის

14.4.6. Queue კლასის Contains() მეთოდი:
ა. რიგიდან ყველა ელემენტს შლის
ბ. რიგის ელემენტებს გადაწერს მასივში
გ. ამოწმებს, არის თუ არა რიგში მითითებული ელემენტი

14.4.7. Queue კლასის Dequeu() მეთოდი:
ა. გასცემს რიგის პირველი ელემენტის მნიშვნელობას და მას რიგიდან შლის
ბ. რიგიდან ყველა ელემენტს შლის
გ. რიგის ელემენტებს გადაწერს მასივში

14.4.8. Queue კლასის Enqueue() მეთოდი:
ა. რიგიდან ყველა ელემენტს შლის
ბ. რიგს ბოლოში უმატებს ელემენტს
გ. რიგის ელემენტებს გადაწერს მასივში

14.4.9. Queue კლასის Equals() მეთოდი:
ა. რიგიდან ყველა ელემენტს შლის
ბ. რიგის ელემენტებს გადაწერს მასივში
გ. ამოწმებს, ტოლია თუ არა ორი რიგი

14.4.10. Queue კლასის Peek() მეთოდი:
ა. გასცემს რიგის პირველი ელემენტის მნიშვნელობას, მაგრამ არ შლის მას რიგიდან
ბ. რიგიდან ყველა ელემენტს შლის
გ. რიგის ელემენტებს გადაწერს მასივში

14.4.11. Queue კლასის ToArray() მეთოდი:
ა. გასცემს რიგის პირველი ელემენტის მნიშვნელობას და მას რიგიდან შლის
ბ. რიგის ელემენტებს გადაწერს მასივში
გ. რიგიდან ყველა ელემენტს შლის

სტეკები

14.5.1. სტეკი არის მეხსიერების უბანი, რომელიც მუშაობს პრინციპით:
ა. "უკანასკნელი მოვიდა – პირველი გავიდა"
ბ. "უკანასკნელი მოვიდა – უკანასკნელი გავიდა"
გ. "პირველი მოვიდა – პირველი გავიდა"

14.5.2. სტეკი განსაზღვრულია:
ა. Queue კლასით
ბ. Stack კლასით
გ. SortedList კლასით

14.5.3. Stack კლასის Count თვისება:
ა. ამოწმებს, ტოლია თუ არა ორი სტეკი
ბ. სტეკის ელემენტებს გადაწერს მასივში
გ. შეიცავს სტეკში რეალურად არსებული ელემენტების რაოდენობას

14.5.4. Stack კლასის IsReadOnly თვისება:
ა. ამოწმებს, არის თუ არა სტეკი მხოლოდ წაკითხვადი
ბ. შეიცავს სტეკში რეალურად არსებული ელემენტების რაოდენობას
გ. სტეკის ელემენტებს გადაწერს მასივში

14.5.5. Stack კლასის Clear() მეთოდი:
ა. სტეკის ელემენტებს გადაწერს მასივში
ბ. სტეკიდან ყველა ელემენტს შლის
გ. შეიცავს სტეკში რეალურად არსებული ელემენტების რაოდენობას

14.5.6. Stack კლასის Contains() მეთოდი:
ა. შეიცავს სტეკში რეალურად არსებული ელემენტების რაოდენობას
ბ. სტეკის ელემენტებს გადაწერს მასივში
გ. ამოწმებს, არის თუ არა სტეკში მითითებული ელემენტი

14.5.7. Stack კლასის Equals() მეთოდი:
ა. ამოწმებს, ტოლია თუ არა ორი სტეკი
ბ. შეიცავს სტეკში რეალურად არსებული ელემენტების რაოდენობას
გ. სტეკის ელემენტებს გადაწერს მასივში

14.5.8. Stack კლასის Peek() მეთოდი:
ა. სტეკის ელემენტებს გადაწერს მასივში
ბ. გასცემს სტეკის უკანასკნელ (მწვერვალზე მყოფ) ობიექტს, მაგრამ არ შლის მას სტეკიდან
გ. შეიცავს სტეკში რეალურად არსებული ელემენტების რაოდენობას

14.5.9. Stack კლასის Pop() მეთოდი:
ა. შეიცავს სტეკში რეალურად არსებული ელემენტების რაოდენობას
ბ. სტეკის ელემენტებს გადაწერს მასივში
გ. გასცემს სტეკის უკანასკნელ (მწვერვალზე მყოფ) ობიექტს და შლის მას სტეკიდან

14.5.10. Stack კლასის Push() მეთოდი:
ა. სტეკის ელემენტებს გადაწერს მასივში
ბ. შეიცავს სტეკში რეალურად არსებული ელემენტების რაოდენობას
გ. ამატებს ელემენტს სტეკის მწვერვალზე

14.5.11. Stack კლასის ToArray() მეთოდი:
ა. შეიცავს სტეკში რეალურად არსებული ელემენტების რაოდენობას
ბ. სტეკის ელემენტებს გადაწერს მასივში
გ. ამოწმებს, ტოლია თუ არა ორი სტეკი

ბიტების მასივები

14.6.1. ბიტების მასივი არის:
ა. ბულის ცვლადების (true და false) მასივი
ბ. true მნიშვნელობების მასივი
გ. false მნიშვნელობების მასივი

14.6.2. ბიტების მასივში true მნიშვნელობას შეესაბამება:
ა. 0, false მნიშვნელობას კი - 1
ბ. 1, false მნიშვნელობას კი - 0
გ. -1, false მნიშვნელობას კი - 1

14.6.3. ბიტების მასივი განსაზღვრულია:
ა. Queue კლასში
ბ. Stack კლასში
გ. BitArray კლასში

14.6.4. BitArray კლასის Count თვისება:
ა. შეიცავს ბიტების მასივში რეალურად არსებული ელემენტების რაოდენობას
ბ. განსაზღვრავს ელემენტების რაოდენობას, რომლებიც შეიძლება შევინახოთ ბიტების მასივში
გ. ამოწმებს, არის თუ არა ბიტების მასივი მხოლოდ წაკითხვადი

14.6.5. BitArray კლასის IsReadOnly თვისება:
ა. განსაზღვრავს ელემენტების რაოდენობას, რომლებიც შეიძლება შევინახოთ ბიტების მასივში

ბ. ამოწმებს, არის თუ არა ბიტების მასივი მხოლოდ წაკითხვადი
გ. შეიცავს ბიტების მასივში რეალურად არსებული ელემენტების რაოდენობას

14.6.6. BitArray კლასის Length თვისება:
ა. ამოწმებს, არის თუ არა ბიტების მასივი მხოლოდ წაკითხვადი
ბ. შეიცავს ბიტების მასივში რეალურად არსებული ელემენტების რაოდენობას
გ. განსაზღვრავს ელემენტების რაოდენობას, რომლებიც შეიძლება შევინახოთ ბიტების მასივში

14.6.7. BitArray კლასის And() მეთოდი:
ა. ასრულებს „და“ ოპერაციას მიმდინარე ბიტების მასივის ელემენტებსა და პარამეტრად გადაცემული ბიტების მასივის ელემენტებს შორის
ბ. ბიტების მასივის ელემენტებს ერთგანზომილებიან მასივში გადაწერს
გ. მითითებულ მნიშვნელობას ანიჭებს ბიტების მასივის ყველა ელემენტს

14.6.8. BitArray კლასის CopyTo() მეთოდი:
ა. ბიტების მასივის მითითებული პოზიციის ბიტს ანიჭებს მითითებულ მნიშვნელობას
ბ. ბიტების მასივის ელემენტებს ერთგანზომილებიან მასივში გადაწერს
გ. მითითებულ მნიშვნელობას ანიჭებს ბიტების მასივის ყველა ელემენტს

14.6.9. BitArray კლასის Equals() მეთოდი:
ა. ბიტების მასივის ელემენტებს ერთგანზომილებიან მასივში გადაწერს
ბ. მითითებულ მნიშვნელობას ანიჭებს ბიტების მასივის ყველა ელემენტს
გ. ამოწმებს, ტოლია თუ არა ბიტების ორი მასივი

14.6.10. BitArray კლასის Get() მეთოდი:
ა. მითითებულ მნიშვნელობას ანიჭებს ბიტების მასივის ყველა ელემენტს
ბ. გასცემს ბიტების მასივის მითითებული პოზიციის ბიტის მნიშვნელობას
გ. ბიტების მასივის ელემენტებს ერთგანზომილებიან მასივში გადაწერს

14.6.11. BitArray კლასის Not() მეთოდი:
ა. ახდენს ბიტების მასივის ელემენტების ინვერსიას
ბ. მითითებულ მნიშვნელობას ანიჭებს ბიტების მასივის ყველა ელემენტს
გ. ბიტების მასივის ელემენტებს ერთგანზომილებიან მასივში გადაწერს

14.6.12. BitArray კლასის Or() მეთოდი:
ა. ბიტების მასივის ელემენტებს ერთგანზომილებიან მასივში გადაწერს
ბ. მითითებულ მნიშვნელობას ანიჭებს ბიტების მასივის ყველა ელემენტს
გ. ასრულებს „ან“ ოპერაციას მიმდინარე ბიტების მასივის ელემენტებსა და პარამეტრად გადაცემული ბიტების მასივის ელემენტებს შორის

14.6.13. BitArray კლასის Set() მეთოდი:
ა. ბიტების მასივის ელემენტებს ერთგანზომილებიან მასივში გადაწერს
ბ. ბიტების მასივის მითითებული პოზიციის ბიტს ანიჭებს მითითებულ მნიშვნელობას
გ. მითითებულ მნიშვნელობას ანიჭებს ბიტების მასივის ყველა ელემენტს

14.6.14. BitArray კლასის SetAll() მეთოდი:
ა. მითითებულ მნიშვნელობას ანიჭებს ბიტების მასივის ყველა ელემენტს

ბ. ბიტების მასივის ელემენტებს ერთგანზომილებიან მასივში გადაწერს
გ. ბიტების მასივის მითითებული პოზიციის ბიტს ანიჭებს მითითებულ მნიშვნელობას

14.6.15. BitArray კლასის Xor() მეთოდი:
ა. მითითებულ მნიშვნელობას ანიჭებს ბიტების მასივის ყველა ელემენტს
ბ. ასრულებს „გამომრიცხავ ან“ ოპერაციას მიმდინარე ბიტების მასივის ელემენტებსა და პარამეტრად გადაცემული ბიტების მასივის ელემენტებს შორის
გ. ბიტების მასივის მითითებული პოზიციის ბიტს ანიჭებს მითითებულ მნიშვნელობას

თავი 15. შესრულების ნაკადები

შესრულების ნაკადის განსაზღვრა

15.1.1. პროცესი არის შესრულების სტადიაში მყოფი:
ა. პროგრამა
ბ. პროცესორი
გ. კლასი

15.1.2. ოპერაციული სისტემები პროცესებს:
ა. არ იყენებენ სხვადასხვა პროგრამა-დანართების განცალკევებისათვის
ბ. იყენებენ სხვადასხვა პროგრამა-დანართების განცალკევებისათვის
გ. იყენებენ სხვადასხვა პროგრამა-დანართების გაერთიანებისათვის

15.1.3. თითოეული პროცესი პროგრამის კოდის ნაწილების შესრულების მართვის მიზნით:
ა. ქმნის მხოლოდ ერთ ნაკადს
ბ. არ ქმნის ერთ ან მეტ ნაკადს
გ. ქმნის ერთ ან მეტ ნაკადს

15.1.4. ნაკადი არის ძირითადი ერთეული, რომლისთვისაც ოპერაციული სისტემა:
ა. პროცესორის დროს გამოყოფს:
ბ. პროცესორის დროს არ გამოყოფს:
გ. მეხსიერების დროს გამოყოფს:

15.1.5. .NET Framework სისტემა პროცესს ყოფს ადვილად სამართავ:
ა. ზეპროცესებად, რომლებსაც პროგრამა-დანართების დომენები ეწოდება
ბ. ქვეპროცესებად, რომლებსაც პროგრამა-დანართების დომენები ეწოდება
გ. ქვეპროცესებად, რომლებსაც პროგრამა-დანართების დომენები არ ეწოდება

15.1.6. .NET Framework არის იზოლირებული გარემო, რომელშიც პროგრამა-დანართი:
ა. ბლოკირებულია
ბ. არ სრულდება
გ. სრულდება

15.1.7. პროგრამა-დანართების დომენები, რომლებიც წარმოდგენილი არის AppDomain ობიექტებით:
ა. უზრუნველყოფს იზოლირებისა და უსაფრთხოების საზღვრებს შესრულების პროცესში მყოფი კოდისთვის
ბ. არ უზრუნველყოფს იზოლირებისა და უსაფრთხოების საზღვრებს შესრულების პროცესში მყოფი კოდისთვის
გ. უზრუნველყოფს იზოლირებისა და უსაფრთხოების საზღვრებს შესრულებული კოდისთვის

15.1.8. ერთი პროცესის შიგნით ერთი ან მეტი ნაკადი:
ა. არ შეიძლება შესრულდეს ერთ ან მეტ პროგრამა-დანართის დომენში
ბ. შეიძლება შესრულდეს ერთ ან მეტ პროგრამა-დანართის დომენში
გ. შეიძლება შესრულდეს მხოლოდ ერთ პროგრამა-დანართის დომენში

15.1.9. ნაკადებთან სამუშაო კლასები განსაზღვრულია:
ა. System.Data სახელების სივრცეში
ბ. System.Linq სახელების სივრცეში
გ. System.Threading სახელების სივრცეში

15.1.10. ნაკადები:
ა. ორგვარია: შესრულების ნაკადები და მონაცემების ნაკადები
ბ. ერთგვარია: შესრულების ნაკადები ნაკადები
გ. ერთგვარია: მონაცემების ნაკადები ნაკადები

15.1.11. Thread კლასის ApartmentState თვისება:
ა. განსაზღვრავს, ნაკადი ბლოკირებულია თუ არა
ბ. განსაზღვრავს, ნაკადი ერთნაკადურ თუ მრავალნაკადურ გარემოში სრულდება
გ. არ განსაზღვრავს, ნაკადი ერთნაკადურ თუ მრავალნაკადურ გარემოში სრულდება

15.1.12. Thread კლასის CurrentThread თვისება:
ა. გასცემს ნაკადს, რომელიც მოცემულ მომენტში არ სრულდება
ბ. არ გასცემს ნაკადს, რომელიც მოცემულ მომენტში სრულდება
გ. გასცემს ნაკადს, რომელიც მოცემულ მომენტში სრულდება

15.1.13. Thread კლასის IsAlive თვისება:
ა. გასცემს true მნიშვნელობას, თუ ნაკადი იქნა გაშვებული და არ იქნა დამთავრებული
ბ. არ გასცემს true მნიშვნელობას, თუ ნაკადი იქნა გაშვებული და არ იქნა დამთავრებული
გ. გასცემს true მნიშვნელობას, თუ ნაკადი არ იქნა გაშვებული და იქნა დამთავრებული

15.1.14. Thread კლასის IsBackground თვისება:
ა. ნაკადისთვის აყენებს ან არ გასცემს ფონური შესრულების ატრიბუტს
ბ. ნაკადისთვის აყენებს ან გასცემს ფონური შესრულების ატრიბუტს
გ. ნაკადისთვის არ აყენებს ან გასცემს ფონური შესრულების ატრიბუტს

15.1.15. Thread კლასის Name თვისება:
ა. აყენებს ან გასცემს ნაკადის სახელს. თუ სახელი მითითებული არ არის, მაშინ ის არ არის null-ის ტოლი
ბ. არ აყენებს ან გასცემს ნაკადის სახელს. თუ სახელი მითითებული არ არის, მაშინ ის არის

null-ის ტოლი

გ. აყენებს ან გასცემს ნაკადის სახელს. თუ სახელი მითითებული არ არის, მაშინ ის არის null-ის ტოლი

15.1.16. Thread კლასის Priority თვისება:

ა. გასცემს ან აყენებს ნაკადის პრიორიტეტს

ბ. არ გასცემს ან აყენებს ნაკადის პრიორიტეტს

გ. გასცემს ან არ აყენებს ნაკადის მდგომარეობას

15.1.17. Thread კლასის ThreadState თვისება:

ა. გასცემს ნაკადის პრიორიტეტს

ბ. გასცემს ნაკადის მდგომარეობას

გ. არ გასცემს ნაკადის მდგომარეობას

15.1.18. Thread კლასის Abort() მეთოდი:

ა. იწვევს ნაკადის შესრულების გაგრძელებას

ბ. არ იწვევს ნაკადის შესრულების შეწყვეტას

გ. იწვევს ნაკადის შესრულების შეწყვეტას

15.1.19. Thread კლასის AllocateDataSlot() მეთოდი:

ა. ყველა ნაკადისთვის გამოყოფს მონაცემების არასახელდებულ უბანს

ბ. ყველა ნაკადისთვის არ გამოყოფს მონაცემების არასახელდებულ უბანს

გ. ყველა ნაკადისთვის გამოყოფს მონაცემების სახელდებულ უბანს

15.1.20. Thread კლასის AllocateNamedDataSlot() მეთოდი:

ა. ყველა ნაკადისთვის ბლოკავს მონაცემების სახელდებულ უბანს

ბ. ყველა ნაკადისთვის გამოყოფს მონაცემების სახელდებულ უბანს

გ. ყველა ნაკადისთვის არ გამოყოფს მონაცემების სახელდებულ უბანს

15.1.21. Thread კლასის FreeNamedDataSlot() მეთოდი:

ა. ბლოკავს ადრე გამოყოფილ მონაცემების სახელდებულ უბანს

ბ. არ ათავისუფლებს ადრე გამოყოფილ მონაცემების სახელდებულ უბანს

გ. ათავისუფლებს ადრე გამოყოფილ მონაცემების სახელდებულ უბანს

15.1.22. Thread კლასის GetData() მეთოდი:

ა. გასცემს მონაცემების უბნის მნიშვნელობას

ბ. არ გასცემს მონაცემების უბნის მნიშვნელობას

გ. აუქმებს მონაცემების უბნის მნიშვნელობას

15.1.23. Thread კლასის GetDomain() მეთოდი:

ა. გასცემს დომენს, რომელშიც ნაკადი არ სრულდება

ბ. გასცემს დომენს, რომელშიც ნაკადი სრულდება

გ. არ გასცემს დომენს, რომელშიც ნაკადი სრულდება

15.1.24. Thread კლასის GetNamedDataSlot() მეთოდი:

ა. აუქმებს მონაცემების სახელდებული უბნის მნიშვნელობას

ბ. არ გასცემს მონაცემების სახელდებული უბნის მნიშვნელობას

გ. გასცემს მონაცემების სახელდებული უბნის მნიშვნელობას

15.1.25. Thread კლასის Interrupt() მეთოდი:
ა. წყვეტს ნაკადის შესრულებას
ბ. არ წყვეტს ნაკადის შესრულებას
გ. წყვეტს ნაკადის ბლოკირებას

15.1.26. Thread კლასის Join() მეთოდი:
ა. არ ბლოკავს ნაკადის შესრულებას სხვა ნაკადის დამთავრებამდე
ბ. ბლოკავს ნაკადის შესრულებას სხვა ნაკადის დამთავრებამდე
გ. ბლოკავს ნაკადის შესრულებას სხვა ნაკადის დაწყებამდე

15.1.27. Thread კლასის ResetAbort() მეთოდი:
ა. არ აუქმებს მიმდინარე ნაკადის შეწყვეტის მოთხოვნას
ბ. აუქმებს მიმდინარე ნაკადის შესრულების მოთხოვნას
გ. აუქმებს მიმდინარე ნაკადის შეწყვეტის მოთხოვნას

15.1.28. Thread კლასის Resume() მეთოდი:
ა. აგრძელებს შეჩერებული ნაკადის შესრულებას
ბ. არ აგრძელებს შეჩერებული ნაკადის შესრულებას
გ. წყვეტს შეჩერებული ნაკადის შესრულებას

15.1.29. Thread კლასის SetData() მეთოდი:
ა. მონაცემების უბნიდან მნიშვნელობას კითხულობს
ბ. მონაცემების უბანს მნიშვნელობას ანიჭებს
გ. მონაცემების უბანს მნიშვნელობას არ ანიჭებს

15.1.30. Thread კლასის Sleep() მეთოდი:
ა. იწყებს ნაკადის შესრულებას
ბ. არ აჩერებს ნაკადის შესრულებას მითითებული დროის განმავლობაში
გ. აჩერებს ნაკადის შესრულებას მითითებული დროის განმავლობაში

15.1.31. Thread კლასის Start() მეთოდი:
ა. იწყებს ნაკადის შესრულებას
ბ. არ იწყებს ნაკადის შესრულებას
გ. ამთავრებს ნაკადის შესრულებას

15.1.32. Thread კლასის Suspend() მეთოდი:
ა. დროებით არ აჩერებს ნაკადის შესრულებას
ბ. დროებით აჩერებს ნაკადის შესრულებას
გ. დროებით აჩერებს ნაკადის ბლოკირებას

შესრულების ნაკადის შექმნა

15. 2.1. ნაკადის შესაქმნელად:
ა. უნდა შევქმნათ Thread კლასის ობიექტი
ბ. არ უნდა შევქმნათ Thread კლასის ობიექტი
გ. უნდა შევქმნათ Sleep კლასის ობიექტი

15.2.2. Thread კლასის კონსტრუქტორს:

ა. უნდა გადავცეთ თვისება

ბ. არ უნდა გადავცეთ დელეგატი

გ. უნდა გადავცეთ დელეგატი

15.2.3. Thread კლასის კონსტრუქტორისთვის გადაცემული დელეგატი:

ა. მიმართავს იმ მეთოდს, რომელიც უნდა შესრულდეს ნაკადის ქვეშ

ბ. არ მიმართავს იმ მეთოდს, რომელიც უნდა შესრულდეს ნაკადის ქვეშ

გ. მიმართავს იმ მეთოდს, რომელიც არ უნდა შესრულდეს ნაკადის ქვეშ

15.2.4. პროგრამის საწყისი კოდი, რომელშიც სრულდება ნაკადებთან მუშაობა:

ა. არ უნდა იწყებოდეს დირექტივით using System.Threading;

ბ. უნდა იწყებოდეს დირექტივით using System.Threading;

გ. უნდა იწყებოდეს დირექტივით using System.Sleep;

15.2.5. პროგრამის ყოველი გაშვებისას ეკრანზე:

ა. არ შეიძლება სხვადასხვა შედეგები მივიღოთ, რადგან წინასწარ ცნობილია, თუ როდის მოახდენს პროცესორი ერთი ნაკადიდან მეორეზე გადართვას

ბ. არ შეიძლება სხვადასხვა შედეგები მივიღოთ, რადგან წინასწარ უცნობია, თუ როდის მოახდენს პროცესორი ერთი ნაკადიდან მეორეზე გადართვას

გ. შეიძლება სხვადასხვა შედეგები მივიღოთ, რადგან წინასწარ უცნობია, თუ როდის მოახდენს პროცესორი ერთი ნაკადიდან მეორეზე გადართვას

15.2.6. Thread.Sleep(5000) მეთოდი შედეგებს ეკრანზე აყოვნებს:

ა. 5 წამის განმავლობაში

ბ. 50 წამის განმავლობაში

გ. 500 წამის განმავლობაში

15.2.7.

```
class ChemiKlasi {
public static void Datvla_Kenti() {  }
public static void Datvla_Luwi()  {  }
}
{
Thread Nakadi1 = new Thread(new ThreadStart(Datvla_Luwi));
Nakadi1.Start();
Datvla_Kenti();
}
```

მოცემულ კოდში:

ა. Datvla_Luwi() მეთოდი შესრულებას იწყებს ძირითად ნაკადში, Datvla_Kenti() მეთოდი კი – Nakadi1 ნაკადში

ბ. Datvla_Luwi() მეთოდი შესრულებას იწყებს Nakadi1 ნაკადში, Datvla_Kenti() მეთოდი კი – ძირითად ნაკადში

გ. Datvla_Kenti() მეთოდი შესრულებას იწყებს Nakadi1 ნაკადში, Datvla_Luwi() მეთოდი კი – ძირითად ნაკადში

შესრულების ნაკადის პრიორიტეტები

15.3.1. თითოეულ ნაკადს შექმნისას ენიჭება:
ა. Normal პრიორიტეტი
ბ. AboveNormal პრიორიტეტი
გ. BelowNormal პრიორიტეტი

15.3.2. სულ არსებობს პრიორიტეტების:
ა. 3 დონე
ბ. 2 დონე
გ. 5 დონე

15.3.3. არსებობს ნაკადის პრიორიტეტების შემდეგი დონეები:
ა. Lowest, BelowNormal, Normal, AboveNormal, Highest
ბ. Highest, BelowNormal
გ. BelowNormal, AboveNormal

15.3.4. არსებობს ნაკადის პრიორიტეტების შემდეგი დონეები:
ა. უმაღლესი, უდაბლესი
ბ. უდაბლესი, ნორმალურზე დაბალი, ნორმალური, ნორმალურზე მაღალი, უმაღლესი
გ. უდაბლესი, ნორმალური

15.3.5.
{
Thread Nakadi1 = new Thread(new ThreadStart(Datvla_Kenti));
Thread.CurrentThread.Priority = ThreadPriority.Lowest;
} მოცემულ კოდში:
ა. Nakadi1 ნაკადს ენიჭება უდაბლესი, მიმდინარე ნაკადს კი - ნორმალური პრიორიტეტი
ბ. Nakadi1 ნაკადს არ ენიჭება ნორმალური პრიორიტეტი, მიმდინარე ნაკადს კი - უდაბლესი
გ. Nakadi1 ნაკადს ენიჭება ნორმალური პრიორიტეტი, მიმდინარე ნაკადს კი - უდაბლესი

15.3.6.
{
Thread Nakadi1 = new Thread(new ThreadStart(Datvla_Kenti));
Nakadi1.Priority = ThreadPriority.BelowNormal;
Thread.CurrentThread.Priority = ThreadPriority.AboveNormal;
} მოცემულ კოდში:
ა. Nakadi1 ნაკადს ენიჭება ნორმალურზე მაღალი პრიორიტეტი, მიმდინარე ნაკადს კი – ნორმალურზე დაბალი
ბ. Nakadi1 ნაკადს არ ენიჭება ნორმალურზე დაბალი პრიორიტეტი, მიმდინარე ნაკადს კი – ნორმალურზე მაღალი
გ. Nakadi1 ნაკადს ენიჭება ნორმალურზე დაბალი პრიორიტეტი, მიმდინარე ნაკადს კი – ნორმალურზე მაღალი

შესრულების ნაკადის მდგომარეობები

15.4.1. ნაკადის მიმდინარე მდგომარეობის მისაღებად:
ა. გამოიყენება ThreadState თვისება
ბ. არ გამოიყენება ThreadState თვისება
გ. გამოიყენება Timer თვისება

15.4.2. ნაკადი არსებობის მანძილზე:
ა. არ შეიძლება რამდენიმე მდგომარეობაში იმყოფებოდეს
ბ. შეიძლება რამდენიმე მდგომარეობაში იმყოფებოდეს
გ. არ შეიძლება ერთ მდგომარეობაში იმყოფებოდეს

15.4.3. შექმნის შემდეგ ნაკადს:
ა. აქვს Unstarted მდგომარეობა
ბ. არ აქვს Unstarted მდგომარეობა
გ. აქვს Running მდგომარეობა

15.4.4. Start() მეთოდის გამოძახების შემდეგ ნაკადი:
ა. Running მდგომარეობაში არ გადადის
ბ. Running მდგომარეობაში გადადის
გ. Unstarted მდგომარეობაში გადადის

15.4.5. თუ ნაკადის IsBackground თვისებას:
ა. true მნიშვნელობას მივანიჭებთ, მაშინ ის ფონურ რეჟიმში არ იმუშავებს
ბ. false მნიშვნელობას მივანიჭებთ, მაშინ ის ფონურ რეჟიმში იმუშავებს
გ. true მნიშვნელობას მივანიჭებთ, მაშინ ის ფონურ რეჟიმში იმუშავებს

15.4.6. ნაკადი დროის ერთსა და იმავე მომენტში:
ა. შეიძლება რამდენიმე მდგომარეობაში იმყოფებოდეს
ბ. არ შეიძლება რამდენიმე მდგომარეობაში იმყოფებოდეს
გ. არ უნდა იმყოფებოდეს რამდენიმე მდგომარეობაში

15.4.7. ნაკადის Aborted მდგომარეობა ნიშნავს, რომ:
ა. მოთხოვნილია ნაკადის გაუქმება
ბ. ნაკადი სრულდება
გ. ნაკადი გაუქმებულია

15.4.8. ნაკადის Unstarted მდგომარეობა ნიშნავს, რომ:
ა. ნაკადი ფონურ რეჟიმში სრულდება
ბ. ნაკადი სრულდება
გ. ნაკადს შესრულება არ დაუწყია

15.4.9. ნაკადის Running მდგომარეობა ნიშნავს, რომ:
ა. ნაკადი სრულდება
ბ. ნაკადი ფონურ რეჟიმში სრულდება
გ. ნაკადი გაუქმებულია

15.4.10. ნაკადის Background მდგომარეობა ნიშნავს, რომ:
ა. ნაკადი სრულდება
ბ. ნაკადი ფონურ რეჟიმში სრულდება
გ. ნაკადი გაუქმებულია

15.4.11. ნაკადის WaitSleepJoin მდგომარეობა ნიშნავს, რომ:
ა. ნაკადი სრულდება
ბ. ნაკადი გაუქმებულია
გ. ნაკადი ბლოკირებულია Wait, Sleep ან Join მეთოდის მიერ

15.4.12. ნაკადის SuspendRequested მდგომარეობა ნიშნავს, რომ:
ა. მოთხოვნილია ნაკადის დროებით შეჩერება
ბ. ნაკადი სრულდება
გ. ნაკადი გაუქმებულია

15.4.13. ნაკადის Suspended მდგომარეობა ნიშნავს, რომ:
ა. ნაკადი გაუქმებულია
ბ. ნაკადი დროებით შეჩერებულია
გ. მოთხოვნილია ნაკადის გაუქმება

15.4.14. ნაკადის StopRequested მდგომარეობა ნიშნავს, რომ:
ა. მოთხოვნილია ნაკადის გაუქმება
ბ. ნაკადი გაუქმებულია
გ. მოთხოვნილია ნაკადის გაჩერება

15.4.15. ნაკადის Stopped მდგომარეობა ნიშნავს, რომ:
ა. ნაკადი გაჩერებულია
ბ. ნაკადი გაუქმებულია
გ. მოთხოვნილია ნაკადის გაუქმება

15.4.16. ნაკადის AbortRequested მდგომარეობა ნიშნავს, რომ:
ა. ნაკადი გაუქმებულია
ბ. მოთხოვნილია ნაკადის გაუქმება
გ. ნაკადი სრულდება

შესრულების ნაკადის ლოკალური მონაცემები

15.5.1. გადასაწყვეტი ამოცანიდან გამომდინარე ნაკადებმა:
ა. შეიძლება გამოიყენოს ოპერატიული მეხსიერების ერთი და იგივე ან სხვადასხვა უბნები
ბ. არ შეიძლება გამოიყენოს ოპერატიული მეხსიერების ერთი და იგივე ან სხვადასხვა უბნები
გ. აუცილებლად უნდა გამოიყენოს ოპერატიული მეხსიერების ერთი და იგივე უბნები

15.5.2. იმისათვის, რომ რამდენიმე ნაკადმა ერთობლივად გამოიყენოს მონაცემები:
ა. არაა საჭირო ამ მონაცემების გამოცხადება ნაკადებისათვის საერთო ხილვადობის უბანში
ბ. საჭიროა ამ მონაცემების გამოცხადება ნაკადებისათვის საერთო ხილვადობის უბანში
გ. საჭიროა ამ მონაცემების გამოცხადება ნაკადებისათვის სხვადასხვა ხილვადობის უბანში

15.5.3. იმისათვის, რომ პროცესებმა თავიანთ ლოკალურ მონაცემებთან იმუშაოს, ანუ გამოიყენოს ოპერატიული მეხსიერების სხვადასხვა უბნები:
ა. უნდა გამოვიყენოთ Timer ობიექტები
ბ. არ უნდა გამოვიყენოთ LocalDataStoreSlot ობიექტები
გ. უნდა გამოვიყენოთ LocalDataStoreSlot ობიექტები

15.5.4. მიუხედავად იმისა, რომ ორი ნაკადი იყენებს ერთი და იგივე სახელის მქონე მეხსიერების უბნებს:
*ა. მათ მაინც გამოეყოფა მეხსიერების სხვადასხვა ლოკალური უბნები
ბ. მათ არ გამოეყოფა მეხსიერების სხვადასხვა ლოკალური უბნები
გ. მათ მაინც გამოეყოფა მეხსიერების ერთ და იგივე ლოკალური უბნები

შესრულების ნაკადების მართვა

Timer კლასი

15.6.1. Timer კლასი საშუალებას:
ა. გვაძლევს ნაკადი დროის გარკვეული გამეორებადი ინტერვალების შემდეგ, ანუ გარკვეული პერიოდულობით შევასრულოთ
ბ. არ გვაძლევს ნაკადი დროის გარკვეული გამეორებადი ინტერვალების შემდეგ, ანუ გარკვეული პერიოდულობით შევასრულოთ
გ. გვაძლევს ნაკადი დროის გარკვეული არაგამეორებადი ინტერვალების შემდეგ, ანუ გარკვეული პერიოდულობით არ შევასრულოთ

15.6.2. Timer ტიპის ობიექტის შექმნისას კონსტრუქტორს:
ა. ორი პარამეტრი უნდა გადავცეთ
ბ. ოთხი პარამეტრი უნდა გადავცეთ
გ. ერთი პარამეტრი უნდა გადავცეთ

15.6.3. Timer ტიპის ობიექტის შექმნისას კონსტრუქტორს შემდეგი პარამეტრები უნდა გადავცეთ:
ა. callback, state
ბ. dueTime, Period
გ. callback, state, dueTime, Period

15.6.4. callback პარამეტრი არის:
ა. TimeCallback ტიპის დელეგატი, რომელიც მიმართავს იმ მეთოდს, რომელსაც ტაიმერი გამოიძახებს
ბ. TimeCallback ტიპის დელეგატი, რომელიც არ მიმართავს იმ მეთოდს, რომელსაც ტაიმერი გამოიძახებს
გ. TimeCallback ტიპის დელეგატი, რომელიც მიმართავს იმ მეთოდს, რომელსაც ტაიმერი არ გამოიძახებს

15.6.5. state პარამეტრი არის:
ა. ობიექტი, რომელიც TimeCallback მეთოდს არ გადაეცემა. ეს პარამეტრი იმიტომ მიეთითება, რომ ტაიმერისათვის მისაწვდომი იყოს რაიმე მუდმივი მდგომარეობა, წინააღმდეგ შემთხვევაში მიეთითება null
ბ. ობიექტი, რომელიც TimeCallback მეთოდს გადაეცემა. ეს პარამეტრი იმიტომ მიეთითება,

რომ ტაიმერისათვის მისაწვდომი იყოს რაიმე მუდმივი მდგომარეობა, წინააღმდეგ შემთხვევაში მიეთითება null

გ. ობიექტი, რომელიც TimeCallback მეთოდს გადაეცემა. ეს პარამეტრი იმიტომ მიეთითება, რომ ტაიმერისათვის მისაწვდომი არ იყოს რაიმე მუდმივი მდგომარეობა, წინააღმდეგ შემთხვევაში მიეთითება null

15.6.6. dueTime პარამეტრი არის:

ა. წუთების რაოდენობა ტაიმერის პირველ ამუშავებამდე

ბ. წამების რაოდენობა ტაიმერის პირველ ამუშავებამდე

გ. მილიწამების რაოდენობა ტაიმერის პირველ ამუშავებამდე

15.6.7. Period პარამეტრი არის:

ა. მილიწამების რაოდენობა ტაიმერის ამუშავებებს შორის

ბ. წამების რაოდენობა ტაიმერის ამუშავებებს შორის

გ. წუთების რაოდენობა ტაიმერის ამუშავებებს შორის

Join() მეთოდი

15.7.1. Join() მეთოდი:

ა. გამოიყენება ერთი ნაკადის მისაბმელად მეორე ნაკადისათვის

ბ. არ გამოიყენება ერთი ნაკადის მისაბმელად მეორე ნაკადისათვის

გ. გამოიყენება ნაკადის წასაშლელად

15.7.2. მიბმული ნაკადი:

ა. არ დაელოდება პირველი ნაკადის შესრულების დამთავრებას და ამის შემდეგ დაიბლოკება

ბ. დაელოდება პირველი ნაკადის შესრულების დამთავრებას და ამის შემდეგ დაიწყებს მუშაობას

გ. არ დაელოდება პირველი ნაკადის შესრულების დამთავრებას და ამის შემდეგ დაიწყებს მუშაობას

15.7.3. Join() მეთოდი საშუალებას:

ა. გვაძლევს მივუთითოთ ლოდინის მინიმალური დრო

ბ. არ გვაძლევს მივუთითოთ ლოდინის მაქსიმალური დრო

გ. გვაძლევს მივუთითოთ ლოდინის მაქსიმალური დრო

15.7.4. nakadi2.Join(4000) მეთოდის შესრულების შედეგად nakadi2 ნაკადი მეორე ნაკადის შესრულების დამთავრებას დაელოდება:

ა. არა უმეტეს 4 წამისა

ბ. არა უმეტეს 40 წამისა

გ. არა უმეტეს 4000 წამისა

15.7.5.

```
{
Thread nakadi2 = new Thread(new ThreadStart(Datvla_Kenti));
nakadi2.Start();
nakadi2.Join();
Datvla_Luwi();
```

}
მოცემულ კოდში:
ა. ხდება nakadi2 ნაკადის ბლოკირება მანამ, სანამ არ დაიწყება Datvla_Luwi() მეთოდის შესრულება
ბ. ხდება nakadi2 ნაკადის ბლოკირება მანამ, სანამ არ დამთავრდება Datvla_Luwi() მეთოდის შესრულება
გ. ხდება Datvla_Luwi() მეთოდის ბლოკირება მანამ, სანამ არ დამთავრდება nakadi2 ნაკადის შესრულება

lock საკვანძო სიტყვა

15.8.1. lock საკვანძო სიტყვა განსაზღვრავს კოდის იმ ფრაგმენტს, რომლის ბლოკირებაც:
ა. უნდა მოხდეს
ბ. არ უნდა მოხდეს
გ. აკრძალულია

15.8.2. ბლოკირებული ფრაგმენტის შესრულების დროს:
ა. მოხდება სხვა პროცესის ბლოკირება
ბ. მოხდება სხვა პროცესის შესრულებაზე გადართვა
გ. არ მოხდება სხვა პროცესის შესრულებაზე გადართვა

15.8.3. lock სიტყვის გამოყენებით:
ა. აკრძალულია ნებისმიერი ობიექტის ბლოკირება
ბ. არ შეიძლება ნებისმიერი ობიექტის ბლოკირება
გ. შეიძლება ნებისმიერი ობიექტის ბლოკირება

Interlocked კლასი

15.9.1. Interlocked კლასი:
ა. გამოიყენება ცვლადების უსაფრთხო ინკრემენტისა და დეკრემენტისათვის
ბ. არ გამოიყენება ცვლადების უსაფრთხო ინკრემენტისა და დეკრემენტისათვის
გ. გამოიყენება ცვლადების არაუსაფრთხო ინკრემენტისა და დეკრემენტისათვის

15.9.2. Interlocked კლასის Decrement() მეთოდი:
ა. ცვლადის მნიშვნელობას ერთით ზრდის
ბ. ცვლადის მნიშვნელობას ერთით ამცირებს
გ. ცვლადის მნიშვნელობას ერთს უტოლებს

15.9.3. Interlocked კლასის Exchange() მეთოდი:
ა. ცვლადის მნიშვნელობას ერთით ზრდის
ბ. ცვლადის მნიშვნელობას ერთს უტოლებს
გ. ცვლადს მნიშვნელობას ანიჭებს

15.9.4. Interlocked კლასის Increment() მეთოდი:
ა. ცვლადის მნიშვნელობას ერთს უტოლებს
ბ. ცვლადის მნიშვნელობას ერთით ამცირებს
გ. ცვლადის მნიშვნელობას ერთით ზრდის

15.9.5. System.Threading.Interlocked.Decrement(ref Saerto_Mtvleli); კოდის შესრულების შედეგად Saerto_Mtvleli ცვლადის მნიშვნელობა:

ა. ერთით გაიზრდება
ბ. ერთით შემცირდება
გ. განულდება

15.9.6. System.Threading.Interlocked.Increment(ref Saerto_Mtvleli); კოდის შესრულების შედეგად Saerto_Mtvleli ცვლადის მნიშვნელობა:

ა. ერთით გაიზრდება
ბ. ერთით შემცირდება
გ. ნულს გაუტოლდება

Monitor კლასი

15.10.1. Monitor კლასი:

ა. გამოიყენება კოდის ნებისმიერ ობიექტთან სინქრონიზებული მიმართვის განხორციელებისათვის
ბ. არ გამოიყენება კოდის ნებისმიერ ობიექტთან სინქრონიზებული მიმართვის განხორციელებისათვის
გ. გამოიყენება კოდის ნებისმიერ ობიექტთან არასინქრონიზებული მიმართვის განხორციელებისათვის

15.10.2. Monitor კლასის Enter() მეთოდი:

ა. არ ბლოკავს მონიტორისათვის გადაცემულ ობიექტს
ბ. ბლოკავს მონიტორისათვის გადაცემულ ობიექტს
გ. ათავისუფლებს მონიტორისათვის გადაცემულ ობიექტს

15.10.3. Monitor კლასის Exit() მეთოდი:

ა. ახდენს ობიექტის ბლოკირებას
ბ. ახდენს ობიექტის დებლოკირებას
გ. ახდენს ობიექტის წაშლას

15.10.4. Monitor კლასის Pulse() მეთოდი მომდევნო მომლოდინე ნაკადს აუწყებს იმის შესახებ, რომ:

ა. მონიტორმა დროებით არ დაამთავრა მუშაობა ობიექტთან და ნაკადს შეუძლია მუშაობის გაგრძელება
ბ. მონიტორმა დროებით დაამთავრა მუშაობა ობიექტთან და ნაკადს არ შეუძლია მუშაობის გაგრძელება
გ. მონიტორმა დროებით დაამთავრა მუშაობა ობიექტთან და ნაკადს შეუძლია მუშაობის გაგრძელება

15.10.5. Monitor კლასის PulseAll() მეთოდი ყველა ნაკადს აუწყებს იმის შესახებ, რომ ობიექტი მალე:

ა. გათავისუფლდება
ბ. დაიბლოკება
გ. არ გათავისუფლდება

15.10.6. Monitor კლასის TryEnter() მეთოდი ცდილობს:

ა. გაათავისუფლოს გადაცემული ობიექტი

ბ. დაბლოკოს გადაცემული ობიექტი

გ. არ დაბლოკოს გადაცემული ობიექტი

15.10.7. Monitor კლასის Wait() მეთოდი ათავისუფლებს ყველა ბლოკირებას და დროებით აჩერებს მიმდინარე ნაკადის შესრულებას მანამ, სანამ სხვა ნაკადი არ გამოიძახებს:

ა. TryEnter() მეთოდს

ბ. Wait() მეთოდს

გ. Pulse() მეთოდს

Mutex კლასი

15.11.1. ნაკადების სინქრონიზება:

ა. შესაძლებელია Mutex კლასის, ანუ სემაფორების გამოიყენებით

ბ. შეუძლებელია Mutex კლასის, ანუ სემაფორების გამოიყენებით

გ. დაუშვებელია Mutex კლასის, ანუ სემაფორების გამოიყენებით

15.11.2. სემაფორი არის ობიექტი, რომელსაც დროის განსაზღვრულ მომენტში:

ა. შეიძლება იყენებდეს მხოლოდ ორი ნაკადი

ბ. შეიძლება იყენებდეს მხოლოდ ერთი ნაკადი

გ. არ შეიძლება იყენებდეს მხოლოდ ერთი ნაკადი

15.11.3. Mutex კლასის კონსტრუქტორში, რომელსაც ორი პარამეტრი აქვს:

ა. პირველი პარამეტრი არის სემაფორის სახელი, მეორე პარამეტრი ბულის ტიპისაა და განსაზღვრავს ობიექტი-სემაფორი უნდა ეკუთვნოდეს თუ არა მის შემქმნელ ნაკადს

ბ. ორივე პარამეტრი არის სემაფორის სახელი

გ. პირველი პარამეტრი ბულის ტიპისაა და განსაზღვრავს ობიექტი-სემაფორი უნდა ეკუთვნოდეს თუ არა მის შემქმნელ ნაკადს, მეორე პარამეტრი არის სემაფორის სახელი

15.11.4. ნაკადს:

ა. შეუძლია WaitOne() მეთოდის გამოძახება სემაფორის მისაღებად

ბ. არ შეუძლია WaitOne() მეთოდის გამოძახება სემაფორის მისაღებად

გ. ეკრძალება WaitOne() მეთოდის გამოძახება სემაფორის მისაღებად

15.11.5. ნაკადი რჩება ბლოკირებულ მდგომარეობაში მანამ, სანამ სემაფორი სხვა:

ა. ნაკადს არ უკავია

ბ. ნაკადს უკავია

გ. ნაკადს გადაეცემა

15.11.6. როგორც კი ნაკადი სემაფორს გაათავისუფლებს, სემაფორი:

ა. მის მომთხოვნ ნაკადს არასოდეს არ დაეთმობა

ბ. მის მომთხოვნ ნაკადს არ დაეთმობა

გ. მის მომთხოვნ ნაკადს დაეთმობა

15.11.7. როცა ნაკადი სემაფორთან მუშაობას დაამთავრებს, მაშინ მას:

ა. შეუძლია ReleaseMutex() მეთოდის გამოძახება სემაფორის გასათავისუფლებლად

ბ. არ შეუძლია ReleaseMutex() მეთოდის გამოძახება სემაფორის გასათავისუფლებლად
გ. შეუძლია ReleaseMutex() მეთოდის გამოძახება სემაფორის დასაკევებლად

თავი 16. საბაზო ბიბლიოთეკის სხვა კლასები გლობალიზაცია

16.1.1. კლასები, რომლებიც საშუალებას გვაძლევს პროგრამის კოდი სწრაფად გავაწყოთ ჩვენთვის საჭირო ეროვნული კულტურის ელემენტების გამოყენებაზე, განსაზღვრულია:
ა. System.Globalization სახელების სივრცეში
ბ. System.Data სახელების სივრცეში
გ. System.Linq სახელების სივრცეში

16.1.2. ეროვნული კულტურის ელემენტებია:
ა. მხოლოდ ეროვნული ენა
ბ. ეროვნული ენა, ვალუტის სიმბოლო, დროის, თარიღისა და რიცხვის ფორმატი და ა.შ.
გ. მხოლოდ ვალუტის სიმბოლო

16.1.3. .NET გარემოში თითოეული ქვეყნის კულტურას შეესაბამება აღნიშვნა, რომელიც შედგება:
ა. კულტურის კოდისგან, რომელსაც არ მოსდევს ერთი ან მეტი ქვეკულტურის კოდი
ბ. ქვეკულტურის კოდისგან, რომელსაც მოსდევს ერთი ან მეტი კულტურის კოდი
გ. კულტურის კოდისგან, რომელსაც მოსდევს ერთი ან მეტი ქვეკულტურის კოდი

16.1.4. შეთანხმების თანახმად:
ა. კულტურის კოდები პატარა ასოებით აღინიშნება, ხოლო ქვეკულტურის კოდები – დიდი ასოებით
ბ. ქვეკულტურის კოდები პატარა ასოებით აღინიშნება, ხოლო კულტურის კოდები – დიდი ასოებით
გ. კულტურისა და ქვეკულტურის კოდები პატარა ასოებით აღინიშნება

16.1.5. en-CA კოდი განსაზღვრავს:
ა. პორტუგალიურ კულტურას ბრაზილიაში
ბ. ინგლისურ კულტურას კანადაში
გ. ბოსნიის კულტურას იმ მომხმარებლებისათვის, რომლებიც ლათინურ ანბანს იყენებენ

16.1.6. ka-GE კოდი განსაზღვრავს:
ა. ინგლისურ კულტურას
ბ. რუსულ კულტურას
გ. ქართულ კულტურას

16.1.7. System.Globalization სახელების სივრცის Calendar კლასი:
ა. აბსტრაქტული კლასია, რომელიც გვაძლევს საერთო წარმოდგენას კულტურისათვის დამახასიათებელი კალენდრის შესახებ
ბ. შეიცავს ინფორმაციას დროისა და თარიღის ფორმატის შესახებ
გ. შეიცავს ინფორმაციას კულტურის ყველა ასპექტის შესახებ

16.1.8. System.Globalization სახელების სივრცის CultureInfo კლასი შეიცავს ინფორმაციას:
ა. დროისა და თარიღის ფორმატის შესახებ
ბ. კულტურის ყველა ასპექტის შესახებ
გ. კულტურის ყველა ასპექტის შესახებ

16.1.9. System.Globalization სახელების სივრცის RegionInfo კლასი შეიცავს ინფორმაციას:
ა. კულტურის ყველა ასპექტის შესახებ
ბ. დროისა და თარიღის ფორმატის შესახებ
გ. განსაზღვრული რეგიონის ან ქვეყნის შესახებ

16.1.10. System.Globalization სახელების სივრცის NumberFormatInfo კლასი შეიცავს ინფორმაციას:
ა. რიცხვის წარმოდგენის ფორმატის შესახებ
ბ. დროისა და თარიღის ფორმატის შესახებ
გ. კულტურის ყველა ასპექტის შესახებ

16.1.11. System.Globalization სახელების სივრცის DaylightTime კლასი შეიცავს ინფორმაციას:
ა. დროისა და თარიღის ფორმატის შესახებ
ბ. ზაფხულისა და ზამთრის დროზე გადასვლის შესახებ
გ. კულტურის ყველა ასპექტის შესახებ

16.1.12. System.Globalization სახელების სივრცის DateTimeFormatInfo კლასი შეიცავს ინფორმაციას:
ა. რიცხვის წარმოდგენის ფორმატის შესახებ
ბ. ს კულტურის ყველა ასპექტის შესახებ
გ. დროისა და თარიღის ფორმატის შესახებ

16.1.13. CultureInfo კლასის Calendar თვისება:
ა. შეიცავს კალენდარს, რომელიც მოცემულ კულტურაში ავტომატურად გამოიყენება
ბ. შეიცავს ინფორმაციას მოცემული კულტურის თარიღისა და დროის შესახებ
გ. შეიცავს მოცემული ნაკადის მიმდინარე კულტურას

16.1.14. CultureInfo კლასის CurrentCulture თვისება:
ა. შეიცავს კალენდარს, რომელიც მოცემულ კულტურაში ავტომატურად გამოიყენება
ბ. შეიცავს მოცემული ნაკადის მიმდინარე კულტურას
გ. არ შეიცავს მოცემული ნაკადის მიმდინარე კულტურას

16.1.15. CultureInfo კლასის DateTimeFormat თვისება:
ა. შეიცავს მოცემული ნაკადის მიმდინარე კულტურას
ბ. შეიცავს კალენდარს, რომელიც მოცემულ კულტურაში ავტომატურად გამოიყენება
გ. შეიცავს ინფორმაციას მოცემული კულტურის თარიღისა და დროის შესახებ

16.1.16. CultureInfo კლასის EnglishName თვისება:
ა. შეიცავს კულტურის ინგლისურ დასახელებას
ბ. შეიცავს კალენდარს, რომელიც მოცემულ კულტურაში ავტომატურად გამოიყენება
გ. შეიცავს მოცემული ნაკადის მიმდინარე კულტურას

16.1.17. CultureInfo კლასის NativeName თვისება:
ა. შეიცავს მოცემული ნაკადის მიმდინარე კულტურას
ბ. შეიცავს კულტურის ეროვნულ დასახელებას
გ. შეიცავს კალენდარს, რომელიც მოცემულ კულტურაში ავტომატურად გამოიყენება

16.1.18. CultureInfo კლასის GetCultures() მეთოდი:
ა. შლის კულტურების სიას
ბ. არ გასცემს კულტურების სიას
გ. გასცემს კულტურების სიას

მონაცემების დაშიფვრა და გაშიფვრა

16.2.1. .NET გარემო შეიცავს კლასებს, რომლებიც:
ა. უზრუნველყოფს როგორც სიმეტრიულ, ისე ასიმეტრიულ კრიპტოგრაფიას
ბ. არ უზრუნველყოფს სიმეტრიულ კრიპტოგრაფიას
გ. არ უზრუნველყოფს ასიმეტრიულ კრიპტოგრაფიას

16.2.2. სიმეტრიულ კრიპტოგრაფიას ეწოდება კრიპტოგრაფია:
ა. ღია გასაღებით
ბ. დახურული გასაღებით
გ. ჩაკეტილი გასაღებით

16.2.3. ასიმეტრიულ კრიპტოგრაფიას ეწოდება კრიპტოგრაფია:
ა. ჩაკეტილი გასაღებით
ბ. დახურული გასაღებით
გ. ღია გასაღებით

16.2.4. სიმეტრიული კრიპტოგრაფიის შემთხვევაში მონაცემების დაშიფვრისა და გაშიფვრისათვის :
ა. ერთი და იგივე გასაღები გამოიყენება
ბ. სხვადასხვა გასაღები გამოიყენება
გ. გასაღები არ გამოიყენება

16.2.5. .NET გარემოში გამოყენებულია სიმეტრიული კრიპტოგრაფიის ოდნავ გართულებული ფორმა, რომელსაც:
ა. დაუშიფრავი ბლოკების გადაბმა ეწოდება
ბ. დაშიფრული ბლოკების გადაბმა ეწოდება
გ. დაშიფრული ბლოკების განცალკევება ეწოდება

16.2.6. მონაცემების გასაშიფრად სიმეტრიული კრიპტოგრაფიის ოდნავ გართულებული ფორმის გამოყენების შემთხვევაში უნდა ვიცოდეთ:
ა. მხოლოდ მაინიციალებელი ვექტორი
ბ. მხოლოდ გასაღები
გ. არა მხოლოდ გასაღები, არამედ მაინიციალებელი ვექტორიც

16.2.7. .NET გარემოში გამოყენებულია სიმეტრიული კრიპტოგრაფიის:
ა. ოთხი ალგორითმი: DES, RC2, Rijndael და Triple DES
ბ. ხუთი ალგორითმი: RSA, DES, RC2, Rijndael და Triple DES
გ. ექვსი ალგორითმი: DSA, RSA, DES, RC2, Rijndael და Triple DES

16.2.8. ასიმეტრიული კრიპტოგრაფიის შემთხვევაში მონაცემების დაშიფვრისა და გაშიფვრისათვის გამოიყენება ორი გასაღები: ღია და დახურული:
ა. სამი გასაღები: ჩაკეტილი, ღია და დახურული
ბ. ორი გასაღები: ღია და დახურული
გ. ერთი გასაღები: ღია

16.2.9. ღია გასაღების გამოყენებით დაშიფრული მონაცემების გაშიფვრა:
ა. შეუძლებელია
ბ. შეუძლებელია შესაბამისი დახურული გასაღებით
გ. შესაძლებელია მხოლოდ შესაბამისი დახურული გასაღებით

16.2.10. .NET გარემოში გამოყენებულია ასიმეტრიული კრიპტოგრაფიის:
ა. ორი ალგორითმი: RSA და DSA
ბ. სამი ალგორითმი: String, RSA და DSA
გ. ოთხი ალგორითმი: Cryptography, Security, RSA და DSA

16.2.11. მონაცემების ასიმეტრიული დაშიფვრა მოითხოვს:
ა. უფრო ნაკლებ გამოთვლებს, ვიდრე სიმეტრიული
ბ. უფრო მეტ გამოთვლებს, ვიდრე სიმეტრიული
გ. არანაირ გათვლებს

16.2.12. კრიპტოგრაფიის კლასები მოთავსებულია System.Security.Cryptography სახელების სივრცეში:
ა. System.Security.AccessControl სახელების სივრცეში
ბ. System.Security.Authentication სახელების სივრცეში
გ. System.Security.Cryptography სახელების სივრცეში

16.2.13. სიმეტრიული კრიპტოგრაფიისათვის გამოიყენება:
ა. TripleDESCryptoServiceProvider კლასი
ბ. FileStream კლასი
გ. String კლასი

16.2.14. მონაცემების სიმეტრიული დაშიფვრისათვის გამოიყენება:
ა. ფაილები
ბ. ნაკადები
გ. სტრიქონები

16.2.15. დაშიფვრა ხორციელდება:
ა. String ობიექტის საშუალებით
ბ. FileStream ობიექტის საშუალებით
გ. CryptoStream ობიექტის საშუალებით

16.2.16. რადგან ერთი ნაკადის გამოსასვლელი შეიძლება მივაწოდოთ მეორე ნაკადის შესასვლელს, ამიტომ CryptoStream კლასი:

ა. შეგვიძლია გამოვიყენოთ FileStream კლასთან ერთად ფაილში დაშიფრული მონაცემების ჩასაწერად და წასაკითხად

ბ. არ შეგვიძლია გამოვიყენოთ FileStream კლასთან ერთად ფაილში დაშიფრული მონაცემების ჩასაწერად და წასაკითხად

გ. არ უნდა გამოვიყენოთ FileStream კლასთან ერთად ფაილში დაშიფრული მონაცემების ჩასაწერად და წასაკითხად

16.2.17. რადგან ერთი ნაკადის გამოსასვლელი შეიძლება მივაწოდოთ მეორე ნაკადის შესასვლელს, ამიტომ CryptoStream კლასი:

ა. არ შეგვიძლია გამოვიყენოთ NetworkStream კლასთან ერთად ქსელში გადასაცემი მონაცემების დასაშიფრად

ბ. შეგვიძლია გამოვიყენოთ NetworkStream კლასთან ერთად ქსელში გადასაცემი მონაცემების დასაშიფრად

გ. არ უნდა გამოვიყენოთ NetworkStream კლასთან ერთად ქსელში გადასაცემი მონაცემების დასაშიფრად

16.2.18. TripleDESCryptoServiceProvider კლასის BlockSize თვისება:

ა. მაინიციალიზებელი ვექტორია მოცემული ალგორითმისთვის

ბ. განსაზღვრავს სიმეტრიული ალგორითმის ოპერაციის რეჟიმს

გ. დასაშიფრი ან გასაშიფრი ბაიტების რაოდენობაა ერთი ოპერაციის შესრულებისას

16.2.19. TripleDESCryptoServiceProvider კლასის IV თვისება:

ა. მაინიციალიზებელი ვექტორია მოცემული ალგორითმისთვის

ბ. დასაშიფრი ან გასაშიფრი ბაიტების რაოდენობაა ერთი ოპერაციის შესრულებისას

გ. განსაზღვრავს სიმეტრიული ალგორითმის ოპერაციის რეჟიმს

16.2.20. TripleDESCryptoServiceProvider კლასის Key თვისება:

ა. განსაზღვრავს სიმეტრიული ალგორითმის ოპერაციის რეჟიმს

ბ. დაშიფვრის გასაღებია მოცემული ალგორითმისთვის

გ. დასაშიფრი ან გასაშიფრი ბაიტების რაოდენობაა ერთი ოპერაციის შესრულებისას

16.2.21. TripleDESCryptoServiceProvider კლასის KeySize თვისება:

ა. დასაშიფრი ან გასაშიფრი ბაიტების რაოდენობაა ერთი ოპერაციის შესრულებისას

ბ. განსაზღვრავს სიმეტრიული ალგორითმის ოპერაციის რეჟიმს

გ. გასაღებში ბიტების რაოდენობაა

16.2.22. TripleDESCryptoServiceProvider კლასის Mode თვისება:

ა. განსაზღვრავს სიმეტრიული ალგორითმის ოპერაციის რეჟიმს

ბ. დასაშიფრი ან გასაშიფრი ბაიტების რაოდენობაა ერთი ოპერაციის შესრულებისას

გ. გასაღებში ბიტების რაოდენობაა

16.2.23. TripleDESCryptoServiceProvider კლასის Clear() მეთოდი:

ა. ქმნის ობიექტს, რომელიც შეიძლება გამოყენებული იყოს გაშიფვრის ოპერაციის შესასრულებლად

ბ. ათავისუფლებს ალგორითმის მიერ დაკავებულ რესურსებს

გ. განსაზღვრავს, აქვს თუ არა მიმდინარე გასაღებს კორექტული ზომა

16.2.24. TripleDESCryptoServiceProvider კლასის CreateDecryptor() მეთოდი:
ა. ათავისუფლებს ალგორითმის მიერ დაკავებულ რესურსებს
ბ. განსაზღვრავს, აქვს თუ არა მიმდინარე გასაღებს კორექტული ზომა
გ. ქმნის ობიექტს, რომელიც შეიძლება გამოყენებული იყოს გაშიფვრის ოპერაციის შესასრულებლად

16.2.25. TripleDESCryptoServiceProvider კლასის CreateEncryptor() მეთოდი:
ა. ქმნის ობიექტს, რომელიც შეიძლება გამოყენებული იყოს დაშიფვრის ოპერაციის შესასრულებლად
ბ. განსაზღვრავს, აქვს თუ არა მიმდინარე გასაღებს კორექტული ზომა
გ. ათავისუფლებს ალგორითმის მიერ დაკავებულ რესურსებს

16.2.26. TripleDESCryptoServiceProvider კლასის GenerateIV() მეთოდი:
ა. განსაზღვრავს, აქვს თუ არა მიმდინარე გასაღებს კორექტული ზომა
ბ. ახდენს ახალი შემთხვევითი მაინიციალიზებელი ვექტორის გენერირებას
გ. ათავისუფლებს ალგორითმის მიერ დაკავებულ რესურსებს

16.2.27. TripleDESCryptoServiceProvider კლასის GenerateKey() მეთოდი:
ა. ათავისუფლებს ალგორითმის მიერ დაკავებულ რესურსებს
ბ. განსაზღვრავს, აქვს თუ არა მიმდინარე გასაღებს კორექტული ზომა
გ. ახდენს ახალი შემთხვევითი გასაღების გენერირებას

16.2.28. TripleDESCryptoServiceProvider კლასის ValidateKeySize() მეთოდი:
ა. განსაზღვრავს, აქვს თუ არა მიმდინარე გასაღებს კორექტული ზომა
ბ. ათავისუფლებს ალგორითმის მიერ დაკავებულ რესურსებს
გ. ახდენს ახალი შემთხვევითი გასაღების გენერირებას

16.2.29. ასიმეტრიული კრიპტოგრაფიის ალგორითმები განსაზღვრულია:
ა. როგორც ნაკადებთან, ისე მცირე ზომის მონაცემებთან სამუშაოდ
ბ. არა ნაკადებთან, არამედ მცირე ზომის მონაცემებთან სამუშაოდ
გ. არა ნაკადებთან, არამედ დიდი ზომის მონაცემებთან სამუშაოდ

სერიულ პორტებთან მუშაობა

16.3.1. სერიულ პორტებთან სამუშაოდ გამოიყენება:
ა. SerialPort კლასი
ბ. ParalellPort კლასი
გ. Ports კლასი

16.3.2. სერიულ პორტებთან სამუშაო პროგრამის შეტანის წინ using დირექტივების ბლოკს უნდა დავუმატოთ დირექტივა:
ა. using System.IO.Pipes;
ბ. using System.IO.Ports;
გ. using System.IO.IsolatedStorage;

16.3.3. სერიული პორტის მუშაობის სიხშირე უნდა მივუთითოთ:
ა. SerialPort ობიექტის PortName თვისებაში

ბ. SerialPort ობიექტის DataBits თვისებაში
გ. SerialPort ობიექტის BaudRate თვისებაში

16.3.4. მონაცემების ბიტების სტანდარტული რაოდენობა 1 ბაიტში მიეთითება:
ა. SerialPort ობიექტის DataBits თვისებაში
ბ. SerialPort ობიექტის BaudRate თვისებაში
გ. SerialPort ობიექტის PortName თვისებაში

16.3.5. მონაცემების ბიტების სტანდარტული რაოდენობა 1 ბაიტში არის:
ა. 1÷8
ბ. 5÷8
გ. 0÷8

16.3.6. პარიტეტის შემოწმების პროტოკოლს განსაზღვრავს:
ა. SerialPort ობიექტისReadBufferSize თვისება
ბ. SerialPort ობიექტის PortName თვისება
გ. SerialPort ობიექტის Parity თვისება

16.3.7. SerialPort ობიექტის Parity თვისება იღებს მნიშვნელობებს:
ა. 0 და 1
ბ. None, Odd, Even, Mark, Space
გ. true და false

16.3.8. სერიული პორტის სახელს განსაზღვრავს:
ა. SerialPort ობიექტის PortName თვისება
ბ. SerialPort ობიექტის ReadBufferSize თვისება
გ. SerialPort ობიექტის Parity თვისება

16.3.9. IsOpen თვისება:
ა. შეიცავს true-ს, თუ სერიული პორტი ღიაა და false-ს, თუ პორტი დახურულია
ბ. შეიცავს false-ს, თუ სერიული პორტი ღიაა და true-ს, თუ პორტი დახურულია
გ. შეიცავს true-ს, თუ სერიული პორტი ღიაა და true-ს, თუ პორტი დახურულია

16.3.10. სერიული პორტის შესასვლელი (მიმღები) ბუფერის ზომას განსაზღვრავს:
ა. SerialPort ობიექტის PortName თვისება
ბ. SerialPort ობიექტის ReadBufferSize თვისება
გ. SerialPort ობიექტის WriteBufferSize თვისება

16.3.11. სერიული პორტის გამოსასვლელი (გადაცემის) ბუფერის ზომას განსაზღვრავს:
ა. SerialPort ობიექტის WriteBufferSize თვისება
ბ. SerialPort ობიექტის PortName თვისება
გ. SerialPort ობიექტის ReadBufferSize თვისება

16.3.12. Close() მეთოდი:
ა. ხსნის ახალ შეერთებას სერიულ პორტთან
ბ. ხურავს სერიულ პორტთან შეერთებას
გ. ბუფერიდან სერიულ პორტს უგზავნის მითითებული რაოდენობის სიმბოლოს

16.3.13. Open() მეთოდი:

ა. ხსნის ახალ შეერთებას სერიულ პორტთან

ბ. ხურავს სერიულ პორტთან შეერთებას

გ. ბუფერიდან სერიულ პორტს უგზავნის მითითებული რაოდენობის სიმბოლოს

16.3.14. int Read(char[] ბუფერი, int წანაცვლება, int რაოდენობა) მეთოდი:

ა. ბუფერიდან სერიულ პორტს უგზავნის მითითებული რაოდენობის სიმბოლოს

ბ. სერიული პორტიდან კითხულობს მითითებული რაოდენობის სიმბოლოებს და ათავსებს მათ ბუფერში დაწყებული წანაცვლებით განსაზღვრული პოზიციიდან

გ. ხურავს სერიულ პორტთან შეერთებას

16.3.15. void Write(char[] ბუფერი, int წანაცვლება, int რაოდენობა) მეთოდი:

ა. სერიული პორტიდან კითხულობს მითითებული რაოდენობის სიმბოლოებს და ათავსებს მათ ბუფერში დაწყებული წანაცვლებით განსაზღვრული პოზიციიდან

ბ. ბუფერიდან სერიულ პორტს უგზავნის მითითებული რაოდენობის სიმბოლოს

გ. ხსნის ახალ შეერთებას სერიულ პორტთან

თავი 17. ADO.NET ბიბლიოთეკა

მონაცემთა ბაზასთან მიმართვა Visual Studio .NET-ის საშუალებებით

17.1.1. C# პროგრამა უერთდება სერვერზე მოთავსებულ მონაცემთა ბაზას, იღებს საჭირო ობიექტებს, როგორიცაა ცხრილები, წარმოდგენები, ფუნქციები, შენახული პროცედურები და მათ იმ კომპიუტერის (კლიენტის) მეხსიერებაში ათავსებს, რომელზეც:

ა. ეს პროგრამა მუშაობს

ბ. ეს პროგრამა არ მუშაობს

გ. სხვა პროგრამა მუშაობს

17.1.2. C# პროგრამა უერთდება სერვერზე მოთავსებულ მონაცემთა ბაზას, იღებს საჭირო ობიექტებს, როგორიცაა ცხრილები, წარმოდგენები, ფუნქციები და შენახული პროცედურები:

ა. ამ ობიექტების სიმრავლეს მონაცემების გლობალური ნაკრები ეწოდება

ბ. ამ ობიექტების სიმრავლეს მონაცემების ლოკალური ნაკრები ეწოდება

გ. ამ ობიექტების სიმრავლეს მონაცემების ვირტუალური ნაკრები ეწოდება

17.1.3. C# პროგრამა უერთდება სერვერზე მოთავსებულ მონაცემთა ბაზას, იღებს საჭირო ობიექტებს, როგორიცაა ცხრილები, წარმოდგენები, ფუნქციები, შენახული პროცედურები და მათ იმ კომპიუტერის (კლიენტის) მეხსიერებაში ათავსებს:

ა. ამის შემდეგ, ჩვენ შეგვეძლება ამ ობიექტებთან მუშაობა. კერძოდ, შეუძლებელი იქნება ცხრილების სტრიქონების ცვლილება, ფუნქციებისა და შენახული პროცედურების გამოძახება და ა.შ.

ბ. ამის შემდეგ, ჩვენ აღარ შეგვეძლება ამ ობიექტებთან მუშაობა. კერძოდ, ცხრილების სტრიქონების ცვლილება, ფუნქციებისა და შენახული პროცედურების გამოძახება და ა.შ.

გ. ამის შემდეგ, ჩვენ შეგვეძლება ამ ობიექტებთან მუშაობა. კერძოდ, ცხრილების სტრიქონების ცვლილება, ფუნქციებისა და შენახული პროცედურების გამოძახება და ა.შ.

17.1.4. რომელი მსჯელობაა სწორი:

ა. C# პროგრამა პერიოდულად უნდა დავუკავშიროთ მონაცემთა ბაზას იმ ცვლილებების შესანახად, რომლებიც ჩვენ შევიტანეთ მონაცემების ლოკალურ ნაკრებში (ასლში)

ბ. C# პროგრამა პერიოდულად არ უნდა დავუკავშიროთ მონაცემთა ბაზას იმ ცვლილებების შესანახად, რომლებიც ჩვენ შევიტანეთ მონაცემების ლოკალურ ნაკრებში (ასლში)

გ. C# პროგრამა პერიოდულად უნდა დავუკავშიროთ მონაცემთა ბაზას იმ ცვლილებების შესანახად, რომლებიც ჩვენ შევიტანეთ მონაცემების გლობალურ ნაკრებში

17.1.5. რომელი მსჯელობაა სწორი:

ა. მონაცემთა ბაზა აუცილებლად უნდა იმყოფებოდეს იმ კომპიუტერზე, რომელზეც ჩვენი პროგრამა მუშაობს

ბ. მონაცემთა ბაზა არ შეიძლება იმყოფებოდეს იმ კომპიუტერზე, რომელზეც ჩვენი პროგრამა მუშაობს

გ. მონაცემთა ბაზა შეიძლება იმყოფებოდეს იმ კომპიუტერზე, რომელზეც ჩვენი პროგრამა მუშაობს

17.1.6. რომელი მსჯელობაა სწორი:

ა. უმჯობესია მონაცემთა ბაზის მოთავსება უფრო სუსტ კომპიუტერზე (სერვერზე), რომელსაც შეეძლება ერთდროულად მოემსახუროს იმ პროგრამებსა და მომხმარებლებს, რომლებიც SQL მოთხოვნებს აგზავნიან

ბ. უმჯობესია მონაცემთა ბაზის მოთავსება უფრო მძლავრ კომპიუტერზე (სერვერზე), რომელსაც შეეძლება ერთდროულად მოემსახუროს იმ პროგრამებსა და მომხმარებლებს, რომლებიც SQL მოთხოვნებს აგზავნიან

გ. უმჯობესია მონაცემთა ბაზის მოთავსება უფრო მძლავრ კომპიუტერზე (სერვერზე), რომელსაც არ შეეძლება ერთდროულად მოემსახუროს იმ პროგრამებსა და მომხმარებლებს, რომლებიც SQL მოთხოვნებს აგზავნიან

17.1.7. Visual Studio .NET გარემოს:

ა. აქვს ფუნქციები, რომლებიც საშუალებას იძლევა დავუკავშირდეთ მონაცემთა ბაზას და ვიმუშაოთ მის ცხრილებთან

ბ. არ აქვს ფუნქციები, რომლებიც საშუალებას იძლევა დავუკავშირდეთ მონაცემთა ბაზას და ვიმუშაოთ მის ცხრილებთან

გ. აქვს ფუნქციები, რომლებიც საშუალებას არ იძლევა დავუკავშირდეთ მონაცემთა ბაზას და ვიმუშაოთ მის ცხრილებთან

17.1.8. Visual Studio .NET გარემოს აქვს ფუნქციები, რომლებიც საშუალებას იძლევა დავუკავშირდეთ მონაცემთა ბაზას და ვიმუშაოთ მის ცხრილებთან:

ა. ეს ფუნქციები თავმოყრილია Server Explorer ფანჯარაში

ბ. ეს ფუნქციები თავმოყრილი არაა Server Explorer ფანჯარაში

გ. ეს ფუნქციები თავმოყრილია Solution Explorer ფანჯარაში

17.1.9. მონაცემთა ბაზასთან შეერთების დასამყარებლად, ჯერ უნდა გავუშვათ Visual Studio .NET სისტემა, შემდეგ კი შევასრულოთ:

ა. Tools მენიუს Connect to Server ბრძანება

ბ. Tools მენიუს Connect to Database ბრძანება

გ. Tools მენიუს Attach to Process ბრძანება

17.1.10. Add Connection ფანჯარის Server name ველში შეგვაქვს:

ა. პაროლი

ბ. სერვერის (კომპიუტერის) სახელი

გ. მომხმარებლის სახელი

17.1.11. Add Connection ფანჯარის User name ველში შეგვაქვს:

ა. სერვერის (კომპიუტერის) სახელი

ბ. პაროლი

გ. მომხმარებლის სახელი

17.1.12. Add Connection ფანჯარის Password ველში შეგვაქვს:

ა. მომხმარებლის სახელი

ბ. სერვერის (კომპიუტერის) სახელი

გ. პაროლი

17.1.13. Add Connection ფანჯარის Select or enter a database name გაშლადი სიიდან ვირჩევთ:

ა. მონაცემთა ბაზას

ბ. პაროლს

გ. მომხმარებლის სახელს

17.1.14. მონაცემთა ბაზასთან შეერთების დამყარების შემდეგ:

ა. არ შეგვეძლება ცხრილების ნახვა და ცვლილება

ბ. შეგვეძლება ცხრილების ნახვა და ცვლილება

გ. შეგვეძლება ცხრილების მხოლოდ ნახვა

17.1.15. ცხრილებთან მიმართვისათვის ვხსნით:

ა. Solution Explorer ფანჯარას

ბ. Properties ფანჯარას

გ. Server Explorer ფანჯარას

17.1.16. ცხრილთან მიმართვისათვის Server Explorer ფანჯარის:

ა. კატალოგების ხის უბანში ჯერ ვხსნით Tables განშტოებას, შემდეგ კი Serveri.Baza.dbo განშტოებას

ბ. კატალოგების ხის უბანში ჯერ ვხსნით Serveri.Baza.dbo განშტოებას, შემდეგ კი Tables განშტოებას

გ. ფაილების ხის უბანში ჯერ ვხსნით Serveri.Baza.dbo განშტოებას, შემდეგ კი Files განშტოებას

17.1.17. Personali ცხრილიდან სტრიქონების მისაღებად მიმთითებელს ვათავსებთ ამ ცხრილის პიქტოგრამაზე,:

ა. ვხსნით კონტექსტურ მენიუს და ვასრულებთ Show Table Data ბრძანებას

ბ. არ ვხსნით კონტექსტურ მენიუს და ვასრულებთ Show Table Data ბრძანებას

გ. ვხსნით კონტექსტურ მენიუს და არ ვასრულებთ Show Table Data ბრძანებას

17.1.18. SQL მოთხოვნის შესატანად ვაჭერთ ინსტრუმენტების პანელის:

ა. Show Criteria Pane კლავიშს

ბ. Show Diagram Pane კლავიშს

გ. Show SQL Pane კლავიშს

17.1.19. მოთხოვნის შესასრულებლად ვაჭერთ:

ა. მხოლოდ Ctrl+R კლავიშებს

ბ. Ctrl+R კლავიშებს ან ინსტრუმენტების პანელის  კლავიშს

გ. მხოლოდ ინსტრუმენტების პანელის  კლავიშს

17.1.20. სვეტების თვისებების სანახავად უნდა გავხსნით Personal განშტოება, მოვნიშნოთ საჭირო სვეტი, გავხსნათ კონტექსტური მენიუ და შევასრულოთ Properties ბრძანება:

ა. უნდა გავხსნით Personal განშტოება, მოვნიშნოთ საჭირო სვეტი, გავხსნათ კონტექსტური მენიუ და შევასრულოთ Properties ბრძანება

ბ. არ უნდა გავხსნით Personal განშტოება, მოვნიშნოთ საჭირო სვეტი, გავხსნათ კონტექსტური მენიუ და შევასრულოთ Properties ბრძანება

გ. უნდა გავხსნით Personal განშტოება, მოვნიშნოთ საჭირო სვეტი, გავხსნათ კონტექსტური მენიუ და არ უნდა შევასრულოთ Properties ბრძანება

ვიზუალური და არავიზუალური კომპონენტებით მონაცემთა ბაზასთან მუშაობა

17.2.1. მონაცემთა ბაზასთან სამუშაოდ using დირექტივების ბლოკში ჩავუმატოთ:

ა. using System.Data.SqlClient; დირექტივა

ბ. using System.Data.Common; დირექტივა

გ. using System.Data.SqlTypes; დირექტივა

17.2.2. dataGridView კომპონენტის Properties ფანჯრის DataSource სიიდან ვირჩევთ:

ა. label კომპონენტს

ბ. bindingSource კომპონენტს

გ. textBox კომპონენტს

17.2.3. bindingNavigator კომპონენტის დასაკავშირებლად Personali ცხრილთან, Properties ფანჯრის BindingSource სიიდან ვირჩევთ:

ა. textBox კომპონენტს

ბ. label კომპონენტს

გ. bindingSource კომპონენტს

17.2.4. bindingNavigator1 კომპონენტის  კლავიში გამოიყენება:

ა. ცხრილში მონაცემების ჩასამატებლად

ბ. მონაცემებით ცხრილის შესავსებად

გ. ცხრილიდან მონაცემების წასაშლელად

17.2.5. bindingNavigator1 კომპონენტის  კლავიში გამოიყენება:

ა. ცხრილში მონაცემების გასაფილტრად

ბ. ცხრილში მონაცემების შესაცვლელად

გ. ცხრილში მონაცემების დასახარისხებლად

17.2.6. bindingNavigator1 კომპონენტის  კლავიში გამოიყენება:

ა. მონაცემებით ცხრილის შესავსებად

ბ. ცხრილში მონაცემების გასაფილტრად

გ. ცხრილში მონაცემების შესაცვლელად

17.2.7. bindingNavigator1 კომპონენტის  კლავიში გამოიყენება:

ა. ცრილში მონაცემების შესაცვლელად

ბ. ცხრილიდან მონაცემების წასაშლელად

გ. ცხრილში მონაცემების ჩასამატებლად

17.2.8. bindingNavigator კომპონენტისთვის კლავიშების დასამატებლად ან წასაშლელად:

ა. გამოიყენება მისი Items თვისება

ბ. არ გამოიყენება მისი Items თვისება

გ. გამოიყენება მისი this თვისება

17.2.9. ჩვენ მიერ ლოკალური ნაკრების Personali ცხრილში შესრულებულ ცვლილებებს სერვერზე შეინახავს კოდი:

ა.
```
{
this.Validate();
this.bindingSource1.EndEdit();
this.personaliTableAdapter.Fill(this.Shekveta1DataSet.Personali);
}
```

ბ.
```
{
this.Validate();
this.bindingSource1.StartEdit();
this.personaliTableAdapter.Update(this.Shekveta1DataSet.Personali);
}
```

გ.
```
{
this.Validate();
this.bindingSource1.EndEdit();
this.personaliTableAdapter.Update(this.Shekveta1DataSet.Personali);
}
```

17.2.10. Update() მეთოდი ითხოვს, რომ ცხრილს, რომელშიც ხდება სტრიქონების შენახვა:

ა. ჰქონდეს პირველადი გასაღები

ბ. არ ჰქონდეს პირველადი გასაღები

გ. ხანდახან ჰქონდეს პირველადი გასაღები

17.2.11. სერვერიდან ჩვენს ლოკალურ მონაცემთა ნაკრებში Personali ცხრილის გადმოსაწერად უნდა შევასრულოთ კოდი:

ა. this.personaliTableAdapter.Update(this.shekvetaDataSet.Personali);

ბ. this.personaliTableAdapter.Fill(this.shekvetaDataSet.Personali);

გ. this.personaliTableAdapter.Fill(this.shekvetaDataSet.Shemkveti);

17.2.12. textBox1 კომპონენტი და მივაბათ ის Personali ცხრილის gvari სვეტს ვიყენებთ:

ა. Text თვისებას

ბ. Dock თვისებას

გ. DataBindings თვისებას

ცხრილებს შორის ბმების შექმნა

17.3.1. ორ ცხრილს შორის კავშირის შესაქმნელად ვიყენებთ:

ა. DataSet კომპონენტს

ბ. bindingSource კომპონენტს

გ. dataView კომპონენტს

17.3.2. bindingSource1 კომპონენტი დაკავშირებულია Personali ცხრილთან, რომელიც არის მთავარი. bindingSource2 კომპონენტი დაკავშირებულია Shemkveti ცხრილთან, რომელიც არის დამოკიდებული. ამ ცხრილებს შორის ბმის შესაქმნელად:

ა. bindingSource2 კომპონენტის DataSource სიიდან ავირჩიოთ კავშირის სახელი, ხოლო DataMember სიიდან კი - bindingSource1
ბ. bindingSource2 კომპონენტის DataSource სიიდან ავირჩიოთ bindingSource1, ხოლო DataMember სიიდან კი - კავშირის სახელი
გ. bindingSource1 კომპონენტის DataSource სიიდან ავირჩიოთ bindingSource1, ხოლო DataMember სიიდან კი - კავშირის სახელი

17.3.3. Personali (მთავარი) და Shemkveti (დამოკიდებული) ცხრილებს შორის კავშირის შექმნისას Relation ფანჯარაში:
ა. Childe Table სიიდან ვირჩევთ მთავარი ცხრილის სახელს - Personali, ხოლო Parent Table სიიდან კი - დამოკიდებული ცხრილის სახელს - Shemkveti
ბ. Parent Table სიიდან ვირჩევთ მთავარი ცხრილის სახელს - Shemkveti, ხოლო Childe Table სიიდან კი - დამოკიდებული ცხრილის სახელს - Personali
გ. Parent Table სიიდან ვირჩევთ მთავარი ცხრილის სახელს - Personali, ხოლო Childe Table სიიდან კი - დამოკიდებული ცხრილის სახელს - Shemkveti

17.3.4. Personali (მთავარი) და Shemkveti (დამოკიდებული) ცხრილებს შორის კავშირის შექმნისას Relation ფანჯარაში:
ა. Foreign Key Columns სიიდან ვირჩევთ მთავარი ცხრილის პირველად გასაღებს. Key Columns სიიდან ვირჩევთ დამოკიდებული ცხრილის გარე გასაღებს
ბ. Key Columns სიიდან ვირჩევთ მთავარი ცხრილის გარე გასაღებს. Foreign Key Columns სიიდან ვირჩევთ დამოკიდებული ცხრილის პირველად გასაღებს
გ. Key Columns სიიდან ვირჩევთ მთავარი ცხრილის პირველად გასაღებს. Foreign Key Columns სიიდან ვირჩევთ დამოკიდებული ცხრილის გარე გასაღებს

მონაცემების გაფილტვრა

17.4.1. მონაცემების გაფილტვრისათვის გამოიყენება:
ა. label კომპონენტი
ბ. dataView კომპონენტი
გ. textBox data კომპონენტი

17.4.2. dataView კომპონენტის დასაკავშირებლად Personali ცხრილთან, მის:
ა. Table თვისებაში უნდა ავირჩიოთ shekvetaDataSet განშტოების Personali ცხრილი
ბ. Table თვისებაში არ უნდა ავირჩიოთ shekvetaDataSet განშტოების Personali ცხრილი
გ. RowFilter თვისებაში უნდა ავირჩიოთ shekvetaDataSet განშტოების Personali ცხრილი

17.4.3. dataGridView1 კომპონენტში გაფილტვრისა და დახარისხების შედეგების გამოსატანად:
ა. მის Cursor თვისებაში უნდა ავირჩიოთ dataView1 კომპონენტი
ბ. მის DataMember თვისებაში უნდა ავირჩიოთ dataView1 კომპონენტი
გ. მის DataSource თვისებაში უნდა ავირჩიოთ dataView1 კომპონენტი

17.4.4. მოცემულ კოდში

```
{
dataView1.RowFilter = "asaki > 35";
dataGridView1.DataSource = dataView1;
}
```

ცხრილის სტრიქონები იფილტრება:
ა. asaki სვეტის მნიშვნელობის მიხედვით

ბ. raodenoba სვეტის მნიშვნელობის მიხედვით
გ. gvari სვეტის მნიშვნელობის მიხედვით

17.4.5.
{
dataView1.RowFilter = "asaki > 35";
dataGridView1.DataSource = dataView1;
}
მოცემული კოდის შესრულების შედეგად dataGridView1 კომპონენტში გამოჩნდება ცხრილის ის სტრიქონები, სადაც:
ა. asaki სვეტის მნიშვნელობა ტოლია 35-ის
ბ. asaki სვეტის მნიშვნელობა არ აღემატება 35-ს
გ. asaki სვეტის მნიშვნელობა აღემატება 35-ს

17.4.6.
{
dataView1.RowFilter = "";
dataGridView1.DataSource = dataView1;
}
მოცემული კოდის შესრულების შედეგად:
ა. ფილტრი არ გაუქმდება
ბ. ფილტრი გაუქმდება
გ. სტრიქონები გაიფილტრება

17.4.7. დავუშვათ, გვაინტერესებს ინფორმაცია სავაჭრო განყოფილებაში მომუშავე იმ თანამშრომლების შესახებ, რომელთა სტაჟი ნაკლებია 10 წელზე. ფილტრს ექნება სახე:
ა. dataView1.RowFilter ="ganyofileba = 'სავაჭრო' AND staji < 10";
ბ. dataView1.RowFilter ="ganyofileba = 'სავაჭრო' AND staji > 10";
გ. dataView1.RowFilter ="ganyofileba = 'სავაჭრო' AND staji = 10";

17.4.8. dataView კომპონენტში ფილტრის შესატანად:
ა. არ შეგვიძლია textBox კომპონენტის გამოყენება
ბ. შეგვიძლია textBox კომპონენტის გამოყენება
გ. შეგვიძლია label კომპონენტის გამოყენება

17.4.9. რომელი მინიჭებაა სწორი:
ა. dataView1.RowFilter = textBox1.Text;
ბ. dataView1.RowFilter = 5;
გ. dataView1.RowFilter = 5.5;

17.4.10. გვაინტერესებს ის თანამშრომლები, რომლებიც დაიბადნენ 2005 წლის 19 მარტამდე. ფილტრს ექნება სახე:
ა. dataView1.RowFilter = " tarigi_dabadebis > '19.03.2005' "
ბ. dataView1.RowFilter = " tarigi_dabadebis < 19.03.2005 "
გ. dataView1.RowFilter = " tarigi_dabadebis < '19.03.2005' "

17.4.11. dataView კომპონენტის ფილტრში თარიღის შესატანად:
ა. შეიძლება dateTimePicker კომპონენტის გამოყენება

ბ. არ შეიძლება dateTimePicker კომპონენტის გამოყენება
გ. შეიძლება label კომპონენტის გამოყენება

მონაცემების დახარისხება

17.5.1. მონაცემების დახარისხებისათვის გამოიყენება:
ა. label კომპონენტი
ბ. dataView კომპონენტი
გ. textBox data კომპონენტი

17.5.2.
{
dataView1.Sort = "asaki DESC";
dataGridView1.DataSource = dataView1;
}
მოცემული კოდის შესრულების შედეგად შესრულდება სტრიქონების დახარისხება:
ა. raodenoba სვეტის მნიშვნელობების კლების მიხედვით
ბ. gvari სვეტის მნიშვნელობების კლების მიხედვით
გ. asaki სვეტის მნიშვნელობების კლების მიხედვით

17.5.3.
{
dataView1.Sort = "asaki DESC";
dataGridView1.DataSource = dataView1;
}
მოცემული კოდის შესრულების შედეგად შესრულდება სტრიქონების დახარისხება:
ა. asaki სვეტის მნიშვნელობების როგორც ზრდის, ისე კლების მიხედვით
ბ. asaki სვეტის მნიშვნელობების ზრდის მიხედვით
გ. asaki სვეტის მნიშვნელობების კლების მიხედვით

17.5.4.
{
dataView1.Sort = "asaki ASC";
dataGridView1.DataSource = dataView1;
}
მოცემული კოდის შესრულების შედეგად შესრულდება სტრიქონების დახარისხება:
ა. asaki სვეტის მნიშვნელობების როგორც ზრდის, ისე კლების მიხედვით
ბ. asaki სვეტის მნიშვნელობების ზრდის მიხედვით
გ. asaki სვეტის მნიშვნელობების კლების მიხედვით

17.5.5.
{
dataView1.Sort = "asaki";
dataGridView1.DataSource = dataView1;
}
მოცემული კოდის შესრულების შედეგად შესრულდება სტრიქონების დახარისხება:
ა. asaki სვეტის მნიშვნელობების როგორც ზრდის, ისე კლების მიხედვით

ბ. asaki სვეტის მნიშვნელობების ზრდის მიხედვით
გ. asaki სვეტის მნიშვნელობების კლების მიხედვით

17.5.6. დახარისხების გასაუქმებლად უნდა შევასრულოთ კოდი:
ა.
```
{
dataView1.Sort = "";
dataGridView1.DataSource = dataView1;
}
```
ბ.
```
{
dataView1.Sort = "asaki";
dataGridView1.DataSource = dataView1;
}
```
გ.
```
{
dataView1.Sort = "asaki DESC";
dataGridView1.DataSource = dataView1;
}
```

17.5.7.
```
{
dataView1.Sort = "";
dataGridView1.DataSource = dataView1;
}
```
მოცემული კოდის შესრულების შედეგად:
ა. გაუქმდება ფილტრაცია
ბ. არ გაუქმდება დახარისხება
გ. გაუქმდება დახარისხება

17.5.8. ცხრილის დახარისხება რამდენიმე სვეტის მნიშვნელობების მიხედვით:
ა. შეუძლებელია
ბ. შესაძლებელია
გ. ხანდახანაა შესაძლებელი

17.5.9.
```
{
dataView1.Sort = "asaki DESC, tarigi ASC";
dataGridView1.DataSource = dataView1;
}
```
მოცემული კოდის შესრულების შედეგად სტრიქონების დახარისხება შესრულდება:
ა. ერთი სვეტის მნიშვნელობების მიხედვით
ბ. ორი სვეტის მნიშვნელობების მიხედვით
გ. სამი სვეტის მნიშვნელობების მიხედვით

17.5.10.
```
{
dataView1.Sort = "asaki DESC, tarigi ASC";
```

```
dataGridView1.DataSource = dataView1;
}
```
მოცემული კოდის შესრულების შედეგად სტრიქონების დახარისხება შესრულდება:
ა. asaki და gvari სვეტების მიხედვით
ბ. asaki და tarigi სვეტების მიხედვით
გ. raodenoba და tarigi სვეტების მიხედვით

17.5.11. მონაცემების დახარისხებისათვის dataGridView1 კომპონენტის გამოყენება:
ა. შეგვიძლია
ბ. არ შეგვიძლია
გ. ხანდახან შეგვიძლია

17.5.12.
```
{
dataGridView1.Sort(this.dataGridView1.Columns[1],
                System.ComponentModel.ListSortDirection.Descending);
}
```
მოცემულ კოდში სტრიქონების დახარისხებისათვის გამოიყენება:
ა. dataView1 კომპონენტი
ბ. dataGridView1 კომპონენტი
გ. textBox1 კომპონენტი

17.5.13. როცა ცხრილის დახარისხება ხდება ორი სვეტის მიხედვით, მაშინ:
ა. მონაცემები ორივე სვეტის მიხედვით უნდა დახარისხდეს კლებადობით
ბ. მონაცემები ორივე სვეტის მიხედვით უნდა დახარისხდეს ზრდადობით
გ. მონაცემები თითოეული სვეტის მიხედვით შეიძლება დახარისხდეს როგორც ზრდადობით, ისე კლებადობით

17.5.14.
```
{
dataGridView1.Sort(this.dataGridView1.Columns[1],
                System.ComponentModel.ListSortDirection.Ascending);
}
```
მოცემულ კოდში სრულდება dataGridView1 კომპონენტის სტრიქონების დახარისხება:
ა. ზრდადობით იმ სვეტის მნიშვნელობების მიხედვით, რომლის ინდექსია 1
ბ. ზრდადობით იმ სვეტის მნიშვნელობების მიხედვით, რომლის ინდექსია 2
გ. ზრდადობით იმ სვეტის მნიშვნელობების მიხედვით, რომლის ინდექსია 0

მონაცემების ძებნა

17.6.1. ცხრილში მონაცემების მოსაძებნად შეგვიძლია გამოვიყენოთ bindingSource1 კომპონენტის:
ა. Add მეთოდი
ბ. Find მეთოდი
გ. CopyTo მეთოდი

17.6.2. Find მეთოდის პირველი პარამეტრი მიუთითებს:
ა. იმ სტრიქონის სახელს, რომელშიც უნდა შესრულდეს ძებნა

ბ. იმ სვეტის სახელს, რომელშიც უნდა შესრულდეს ძებნა
გ. საძებნ სიდიდეს

17.6.3. Find მეთოდის მეორე პარამეტრი მიუთითებს:
ა. იმ სვეტის სახელს, რომელშიც უნდა შესრულდეს ძებნა
ბ. იმ სტრიქონის სახელს, რომელშიც უნდა შესრულდეს ძებნა
გ. საძებნ სიდიდეს

17.6.4. int index = bindingSource1.Find("gvari", "სამხარაძე რომანი");
მოყვანილი კოდი:
ა. gvari სვეტში ასრულებს "სამხარაძე რომანი" მნიშვნელობის ძებნას
ბ. gvari სვეტში ასრულებს მნიშვნელობის დახარისხებას
გ. ასრულებს gvari სვეტის გაფილტვრას "სამხარაძე რომანი" მნიშვნელობის მიხედვით

17.6.5. bindingSource კომპონენტის Find მეთოდის შესრულების შედეგად გაიცემა:
ა. ცხრილის პირველი სტრიქონის ინდექსი
ბ. მოძებნილი მნიშვნელობის შემცველი სტრიქონის ინდექსი
გ. მოძებნილი მნიშვნელობის შემცველი სვეტის ინდექსი

17.6.6. თუ bindingSource კომპონენტის Find მეთოდის შესრულების შედეგად ვერ მოიძებნა მნიშვნელობა, მაშინ გაიცემა:
ა. 0
ბ. 1
გ. -1

17.6.7. string gvari = this.shekvetaDataSet.Personali.Rows[5][1].ToString();
კოდის შესრულების შედეგად gvari ცვლადს ენიჭება:
ა. shekvetaDataSet ცხრილის მეხუთე სტრიქონის პირველი სვეტის მნიშვნელობა
ბ. Personali ცხრილის მეხუთე სვეტის პირველი სტრიქონის მნიშვნელობა
გ. Personali ცხრილის მეხუთე სტრიქონის პირველი სვეტის მნიშვნელობა

17.6.8. იმისათვის, რომ dataGridView კომპონენტში მოინიშნოს მოძებნილი სტრიქონი, მისი ინდექსის მნიშვნელობა bindingSource კომპონენტის:
ა. Position თვისებას უნდა მივანიჭოთ
ბ. Current თვისებას უნდა მივანიჭოთ
გ. Site თვისებას უნდა მივანიჭოთ

ნავიგაცია

17.7.1. ცხრილის მომდევნო სტრიქონზე გადასასვლელად გამოიყენება:
ა. bindingSource1.MoveNext() მეთოდი
ბ. bindingSource1.MovePrevious() მეთოდი
გ. bindingSource1.MoveFirst() მეთოდი

17.7.2. ცხრილის წინა სტრიქონზე გადასასვლელად გამოიყენება:
ა. bindingSource1.MoveNext() მეთოდი
ბ. bindingSource1.MovePrevious() მეთოდი
გ. bindingSource1.MoveFirst() მეთოდი

17.7.3. ცხრილის პირველ სტრიქონზე გადასასვლელად გამოიყენება:

ა. bindingSource1.MovePrevious() მეთოდი

ბ. bindingSource1.MoveLast() მეთოდი

გ. bindingSource1.MoveFirst() მეთოდი

17.7.4. ცხრილის უკანასკნელ სტრიქონზე გადასასვლელად გამოიყენება:

ა. bindingSource1.MoveFirst() მეთოდი

ბ. bindingSource1.MoveLast() მეთოდი

გ. bindingSource1.MovePrevious() მეთოდი

17.7.5. თუ მოვნიშნავთ dataGridView1 კომპონენტის რომელიმე უჯრას, მაშინ მისი შემცველი სტრიქონის ინდექსი:

ა. ავტომატურად მიენიჭება RowIndex თვისებას

ბ. ავტომატურად არ მიენიჭება RowIndex თვისებას

გ. ავტომატურად მიენიჭება RowIndex მოვლენას

17.7.6. ინდექსის საშუალებით შეგვიძლია მივმართოთ არჩეული სტრიქონის:

ა. ნებისმიერ სვეტს

ბ. ნებისმიერ ინდექსს

გ. ნებისმიერ ცხრილს

17.7.7. string qalaqi = this.dataGridView1.CurrentRow.Cells[3].Value .ToString(); მინიჭების ოპერატორში მიმართვა ხდება:

ა. Value კომპონენტის მესამე სვეტთან

ბ. dataGridView1 კომპონენტის მესამე სვეტთან

გ. CurrentRow კომპონენტის მესამე სვეტთან

17.7.8. string qalaqi = this.dataGridView1.CurrentRow.Cells[0].Value .ToString(); მინიჭების ოპერატორში მიმართვა ხდება:

ა. dataGridView1 კომპონენტის მეშვიდე სვეტთან

ბ. dataGridView1 კომპონენტის მესამე სვეტთან

გ. dataGridView1 კომპონენტის ნულოვან სვეტთან

17.7.9.

int staji = Convert.ToInt32(this.shekveta_1DataSet.Personali.Rows[CurrentRowIndex]["staji"]); მინიჭების ოპერატორში მიმართვა ხდება:

ა. staji სვეტთან

ბ. Personali სვეტთან

გ. CurrentRowIndex სვეტთან

17.7.10. int CurrentRowIndex1 = this.dataGridView6.CurrentCell.RowIndex; მინიჭების შედეგად:

ა. Personali ცხრილის მიმდინარე სტრიქონის მეოთხე სვეტის მნიშვნელობას მიენიჭება tanxa ცვლადის მნიშვნელობა

ბ. CurrentRowIndex1 ცვლადს მიენიჭება dataGridView6 კომპონენტში მონიშნული სტრიქონის ინდექსი

გ. xelfasi ცვლადს მიენიჭება Personali ცხრილის მიმდინარე სტრიქონის მეოთხე სვეტის მნიშვნელობა

17.7.11. double xelfasi = Convert.ToDouble(this.shekveta_1DataSet.Personali.Rows[CurrentRowIndex1][4].ToString()); მინიჭების შედეგად:

ა. xelfasi ცვლადს მიენიჭება Personali ცხრილის მიმდინარე სტრიქონის მეოთხე სვეტის მნიშვნელობა

ბ. CurrentRowIndex1 ცვლადს მიენიჭება dataGridView6 კომპონენტში მონიშნული სტრიქონის ინდექსი

გ. Personali ცხრილის მიმდინარე სტრიქონის მეოთხე სვეტის მნიშვნელობას მიენიჭება tanxa ცვლადის მნიშვნელობა

17.7.12. this.shekveta_1DataSet.Personali.Rows[CurrentRowIndex1][4] = tanxa; მინიჭების შედეგად:

ა. CurrentRowIndex1 ცვლადს მიენიჭება dataGridView6 კომპონენტში მონიშნული სტრიქონის ინდექსი

ბ. xelfasi ცვლადს მიენიჭება Personali ცხრილის მიმდინარე სტრიქონის მეოთხე სვეტის მნიშვნელობა

გ. Personali ცხრილის მიმდინარე სტრიქონის მეოთხე სვეტის მნიშვნელობას მიენიჭება tanxa ცვლადის მნიშვნელობა

17.7.13. ცხრილის მოცემული სტრიქონის რომელიმე სვეტს:

ა. არ შეგვიძლია მივმართოთ როგორც სახელით, ისე ინდექსით

ბ. შეგვიძლია მივმართოთ როგორც სახელით, ისე ინდექსით

გ. შეგვიძლია მივმართოთ მხოლოდ სახელით

გამოთვლები

17.8.1. გამოთვლების შესასრულებლად გამოიყენება DataSet კომპონენტის:

ა. Compute მეთოდი

ბ. Equals მეთოდი

გ. EndInit მეთოდი

17.8.2. DataSet კომპონენტის Compute მეთოდი საშუალებას გვაძლევს მითითებული სვეტის მნიშვნელობებზე შემდეგი ფუნქციები შევასრულოთ:

ა. SUM, AVG, COUNT, MAX, POW

ბ. SUM, AVG, COUNT, MAX, MIN

გ. SUM, AVG, STR, MAX, MIN

17.8.3. Compute მეთოდის:

ა. პირველი პარამეტრია შესასრულებელი ფუნქცია, მეორე პარამეტრი არ აქვს

ბ. პირველი პარამეტრია ფილტრი, მეორე კი – შესასრულებელი ფუნქცია

გ. პირველი პარამეტრია შესასრულებელი ფუნქცია, მეორე კი – ფილტრი

17.8.4. object sumObject = shekveta_1DataSet.Personali.Compute("SUM(xelfasi)", ""); მინიჭების ოპერატორი ასრულებს:

ა. xelfasi სვეტის მნიშვნელობების დათვლას

ბ. xelfasi სვეტის მნიშვნელობების საშუალო არითმეტიკულის გამოთვლას

გ. xelfasi სვეტის მნიშვნელობების შეკრებას

17.8.5. DataSet კომპონენტის Compute მეთოდის:

ა. ორივე პარამეტრი შეიცავს შესასრულებელ ფუნქციას

ბ. პირველი პარამეტრი შეიცავს ფილტრს, მეორე კი – შესასრულებელ ფუნქციას

გ. პირველი პარამეტრი შეიცავს შესასრულებელ ფუნქციას, მეორე კი – ფილტრს

17.8.6. object sumObject = shekveta_1DataSet.Personali.Compute("SUM(xelfasi)", ""); მინიჭების ოპერატორი:

ა. არ ასრულებს ცხრილის გაფილტვრას

ბ. ასრულებს ცხრილის გაფილტვრას

გ. ხანდახან ასრულებს ცხრილის გაფილტვრას

17.8.7. object avgObject = (this.shekveta_1DataSet.Personali.Compute("AVG(asaki)", "xelfasi > 825"); მოცემულ კოდში გამოითვლება:

ა. asaki სვეტის საშუალო არითმეტიკული იმ სტრიქონებისთვის, რომლებშიც xelfasi სვეტის მნიშვნელობა 825-ს აღემატება

ბ. asaki სვეტის საშუალო არითმეტიკული იმ სტრიქონებისთვის, რომლებშიც xelfasi სვეტის მნიშვნელობა 825-ს აღემატება

გ. asaki სვეტის საშუალო არითმეტიკული იმ სტრიქონებისთვის, რომლებშიც xelfasi სვეტის მნიშვნელობა 825-ის ტოლია

17.8.8. object countObject = shekveta_1DataSet.Personali.Compute("COUNT(asaki)", ""); მოცემულ კოდში გამოითვლება:

ა. asaki სვეტის მინიმალური მნიშვნელობა

ბ. asaki სვეტში მნიშვნელობების რაოდენობა

გ. asaki სვეტის მნიშვნელობების ჯამი

17.8.9. object maxObject = shekveta_1DataSet.Personali Compute("MAX(asaki)", ""); მოცემულ კოდში გამოითვლება:

ა. asaki სვეტის მნიშვნელობების ჯამი

ბ. asaki სვეტის მაქსიმალური მნიშვნელობა

გ. asaki სვეტის მინიმალური მნიშვნელობა

17.8.10. object minObject = shekveta_1DataSet.Personali.Compute("MIN(asaki)", ""); მოცემულ კოდში გამოითვლება:

ა. asaki სვეტის მნიშვნელობების საშუალო არითმეტიკული

ბ. asaki სვეტის მნიშვნელობების ჯამი

გ. asaki სვეტის მინიმალური მნიშვნელობა

17.8.11. DataSet კომპონენტის Compute მეთოდის გარდა გამოთვლების შესრულება შეიძლება, აგრეთვე:

ა. DataRow ობიექტის გამოყენები

ბ. DataColumn ობიექტის გამოყენები

გ. DataTable ობიექტის გამოყენებით

17.8.12.

```
{
DataTable table = magalitebiDataSet1.Tables["Comput"];
object sumObject = table.Compute("Sum(Asaki)", "");
```

}
მოცემულ კოდში გამოთვლების შესასრულებლად გამოიყენება:
ა. DataTable ობიექტი
ბ. DataSet კომპონენტის Compute მეთოდი
გ. objec მეთოდი

17.8.13.
{
DataTable table = magalitebiDataSet1.Tables["Comput"];
object sumObject = table.Compute("Sum(Asaki)", "");
}
მოცემულ კოდში გამოითვლება:
ა. asaki სვეტის მნიშვნელობების საშუალო არითმეტიკული
ბ. asaki სვეტის მნიშვნელობების ჯამი
გ. asaki სვეტის მინიმალური მნიშვნელობა

17.8.14.
{
DataTable table = magalitebiDataSet1.Tables["Comput"];
object sumObject = table.Compute("Avg(Asaki)", "");
}
მოცემულ კოდში გამოითვლება:
ა. asaki სვეტის მნიშვნელობების საშუალო არითმეტიკული
ბ. asaki სვეტის მნიშვნელობების ჯამი
გ. asaki სვეტის მინიმალური მნიშვნელობა

17.8.15.
{
DataTable table = magalitebiDataSet1.Tables["Comput"];
object sumObject = table.Compute("Count(Asaki)", "");
}
მოცემულ კოდში გამოითვლება:
ა. asaki სვეტში მნიშვნელობების რაოდენობა
ბ. asaki სვეტის მნიშვნელობების ჯამი
გ. asaki სვეტის მინიმალური მნიშვნელობა

17.8.16.
{
DataTable table = magalitebiDataSet1.Tables["Comput"];
object sumObject = table.Compute("Min(Asaki)", "");
}
მოცემულ კოდში გამოითვლება:
ა. asaki სვეტის მნიშვნელობების საშუალო არითმეტიკული
ბ. asaki სვეტის მნიშვნელობების ჯამი
გ. asaki სვეტის მინიმალური მნიშვნელობა

17.8.17.

```
{
DataTable table = magalitebiDataSet1.Tables["Comput"];
object sumObject = table.Compute("Max(Asaki)", "");
}
```

მოცემულ კოდში გამოითვლება:

ა. asaki სვეტში მაქსიმალური მნიშვნელობა

ბ. asaki სვეტის მნიშვნელობების ჯამი

გ. asaki სვეტის მინიმალური მნიშვნელობა

ცხრილებს შორის არსებული ბმებით მანიპულირება

17.9.1. რომელი კოდი ამყარებს ბმას Personali (მთავარი) და Shemkveti (დამოკიდებული) ცხრილებს შორის:

ა.

```
{ bindingSource2.DataSource = bindingSource1;
bindingSource2.DataMember = "Personali_Shemkveti"; }
```

ბ.

```
{ bindingSource2.DataSource = "Personali_Shemkveti";
bindingSource2.DataMember = bindingSource1; }
```

გ.

```
{ bindingSource2.DataSource = bindingSource2;
bindingSource2.DataMember = "Personali_Shemkveti"; }
```

17.9.2. რომელი კოდი აუქმებს ბმას Personali (მთავარი) და Shemkveti (დამოკიდებული) ცხრილებს შორის:

ა.

```
{ bindingSource1.DataSource = null;
bindingSource1.DataSource = shekvetaDataSet.Personali.ToString();
bindingSource1.DataMember = shekvetaDataSet;
bindingSource2.DataSource = shekvetaDataSet.Shemkveti.ToString();;
bindingSource2.DataMember = shekvetaDataSet }
```

ბ.

```
{ bindingSource1.DataSource = null;
bindingSource1.DataSource = shekvetaDataSet;
bindingSource1.DataMember = shekvetaDataSet.Personali.ToString();
bindingSource2.DataSource = shekvetaDataSet;
bindingSource2.DataMember = shekvetaDataSet.Shemkveti.ToString(); }
```

გ.

```
{ bindingSource1.DataMember = shekvetaDataSet.Personali.ToString();
bindingSource2.DataSource = shekvetaDataSet;
bindingSource2.DataMember = shekvetaDataSet.Shemkveti.ToString(); }
```

ADO.NET კლასების მომიხილვა

ADO.NET კლასები და ობიექტები

17.10.1. ADO.NET არის კლასების სიმრავლე, რომელიც:

ა. გამოიყენება C# და .NET Framework გარემოსთან ერთად სხვადასხვა ფორმატის მონაცემთა ბაზასთან სამუშაოდ

ბ. არ გამოიყენება C# და .NET Framework გარემოსთან ერთად სხვადასხვა ფორმატის მონაცემთა ბაზასთან სამუშაოდ

გ. გამოიყენება C# და .NET Framework გარემოსთან ერთად მხოლოდ ერთი ფორმატის მონაცემთა ბაზასთან სამუშაოდ

17.10.2. ADO.NET ტექნოლოგია:

ა. არაა ინტეგრირებული .NET Framework გარემოსთან და ორიენტირებულია .NET-ის ნებისმიერ ენასთან, განსაკუთრებით კი - C#-თან სამუშაოდ

ბ. ინტეგრირებულია .NET Framework გარემოსთან და ორიენტირებულია .NET-ის ნებისმიერ ენასთან, განსაკუთრებით კი - C#-თან სამუშაოდ

გ. ინტეგრირებულია .NET Framework გარემოსთან და არაა ორიენტირებული .NET-ის ნებისმიერ ენასთან, განსაკუთრებით კი - C#-თან სამუშაოდ

17.10.3. .NET Framework გარემო:

ა. არ მოიცავს System.Data სახელების სივრცეს და მასში შემავალ ჩადგმულ სახელების სივრცეებს: System.Data.SqlClient, System.Data.Linq და ა.შ.

ბ. მოიცავს System.Data სახელების სივრცეს და მასში შემავალ ჩადგმულ სახელების სივრცეებს: System.Data.SqlClient, System.Data.Linq და ა.შ.

გ. მოიცავს System.Data სახელების სივრცეს და არ მოიცავს მასში შემავალ ჩადგმულ სახელების სივრცეებს: System.Data.SqlClient, System.Data.Linq და ა.შ.

17.10.4. ADO.NET უზრუნველყოფს მონაცემთა რელაციურ ბაზებთან, როგორიცა Microsoft SQL Server და Microsoft Access:

ა. რთულ მიმართვას

ბ. მარტივ მიმართვას

გ. არანაირ მიმართვას

17.10.5. ADO.NET კლასების საშუალებით:

ა. არ შეგვიძლია განვსაზღვროთ ცხრილები, სვეტები და სტრიქონები

ბ. შეგვიძლია განვსაზღვროთ ცხრილები, სვეტები და სტრიქონები

გ. შეგვიძლია განვსაზღვროთ მხოლოდ ცხრილები

17.10.6. ADO.NET-ში განსაზღვრულია DataSet კლასი, რომელიც:

ა. არ წარმოადგენს რელაციური ცხრილების ერთობლიობას, მათ შორის არსებულ კავშირებთან ერთად

ბ. წარმოადგენს რელაციური ცხრილების ერთობლიობას, მათ შორის არსებულ კავშირებთან ერთად

გ. წარმოადგენს რელაციური ცხრილების ერთობლიობას, მათ შორის არსებულ კავშირების გარეშე

17.10.7. სხვადასხვა ფორმატის მონაცემთა ბაზებთან მუშაობის გასამარტივებლად Microsoft-მა:

ა. არ შეიმუშავა ODBC ინტერფეისი

ბ. შეიმუშავა ODBC ინტერფეისი
გ. შეიმუშავა Data ინტერფეისი

17.10.8. ODBC ინტერფეისს აქვს ფუნქციების სრული ნაკრები, რომლების გამოყენებაც:
ა. არ შეიძლება სხვადასხვა ფორმატის მონაცემთა ბაზის მიმართ
ბ. შეიძლება სხვადასხვა ფორმატის მონაცემთა ბაზის მიმართ
გ. შეიძლება მხოლოდ ერთი ფორმატის მონაცემთა ბაზის მიმართ

17.10.9. ODBC ტექნოლგია კარგად მუშაობს ტრადიციულ მონაცემთა ბაზებთან, მაგრამ ის:
ა. იძლევა ისეთ მონაცემებთან მიმართვის შესაძლებლობას, რომლებიც არ არის წარმოდგენილი სტრიქონებისა და სვეტების სახით ან არ აქვს რეგულარული სტრუქტურა
ბ. არ იძლევა ისეთ მონაცემებთან მიმართვის შესაძლებლობას, რომლებიც არ არის წარმოდგენილი სტრიქონებისა და სვეტების სახით ან არ აქვს რეგულარული სტრუქტურა
გ. არ იძლევა ისეთ მონაცემებთან მიმართვის შესაძლებლობას, რომლებიც არის წარმოდგენილი სტრიქონებისა და სვეტების სახით ან არ აქვს რეგულარული სტრუქტურა

17.10.10. OLE DB ტექნოლოგია:
ა. არ წარმოადგენს შუალედურ რგოლს კლიენტის პროგრამასა და მონაცემთა ბაზას შორის და კლიენტის პროგრამას მონაცემებს წარუდგენს ცხრილური სახით
ბ. წარმოადგენს შუალედურ რგოლს კლიენტის პროგრამასა და მონაცემთა ბაზას შორის და კლიენტის პროგრამას მონაცემებს წარუდგენს ცხრილური სახით
გ. წარმოადგენს შუალედურ რგოლს კლიენტის პროგრამასა და მონაცემთა ბაზას შორის და კლიენტის პროგრამას მონაცემებს არ წარუდგენს ცხრილური სახით

17.10.11. ADO.NET უზრუნველყოფს:
ა. მხოლოდ OLE DB ტექნოლოგიას
ბ. როგორც OLE DB, ისე ODBC ტექნოლოგიას
გ. მხოლოდ ODBC ტექნოლოგიას

17.10.12. ADO პროგრამული სისტემაა, რომელიც მაღალი დონის ენებზე დაწერილ პროგრამებს საშუალებას:
ა. არ აძლევს მიმართონ OLE DB მონაცემებს
ბ. აძლევს მიმართონ OLE DB მონაცემებს
გ. აძლევს არ მიმართონ OLE DB მონაცემებს

17.10.13. ADO.NET საბაზო კლასები შეგვიძლია წარმოვადგინოთ:
ა. მხოლოდ როგორც კლასები, რომლებიც აღწერს ობიექტ-მომხმარებლებს
ბ. როგორც კლასები, რომლებიც აღწერს ობიექტ-მომხმარებლებს (ესაა ობიექტები, რომლებიც მონაცემებს იღებს) და კლასები, რომლებიც აღწერს ობიექტ-მიმწოდებლებს (ესაა ობიექტები, რომლებიც გასცემს მონაცემებს)
გ. მხოლოდ როგორც კლასები, რომლებიც აღწერს ობიექტ-მიმწოდებლებს

17.10.14. ობიექტი-მიმწოდებლები ახდენს მონაცემების კითხვას მონაცემთა ბაზებიდან და მათ მიწოდებას:

ა. ობიექტი-მიმწოდებლებისათვის

ბ. ობიექტი-მომხმარებლებისათვის

გ. არავისთვის

17.10.15. ობიექტი-მიმწოდებლები ახდენს მონაცემების კითხვას მონაცემთა ბაზებიდან და მათ მიწოდებას ობიექტი-მომხმარებლებისათვის. ამიტომ, ობიექტი-მიმწოდებლები:

ა. ითხოვს კლიენტთან აქტიური შეერთების არსებობას

ბ. ითხოვს სერვერთან აქტიური შეერთების არსებობას

გ. არ ითხოვს სერვერთან აქტიური შეერთების არსებობას

17.10.16. ობიექტ-მომხმარებლებს მონაცემების მიღების შემდეგ ამ მონაცემებთან მუშაობა:

ა. არ შეუძლია ავტონომურ (ოფლაინ) რეჟიმში, ანუ რეჟიმში, როცა მონაცემთა ბაზასთან შეერთება დახურულია

ბ. შეუძლია ავტონომურ (ოფლაინ) რეჟიმში, ანუ რეჟიმში, როცა მონაცემთა ბაზასთან შეერთება დახურულია

გ. შეუძლია მხოლოდ ონლაინ რეჟიმში

17.10.17. DataSet კლასის ობიექტი გამოიყენება მონაცემთა ბაზაში მოთავსებული მონაცემების:

ა. ლოკალური ასლების შესანახად სერვერზე

ბ. ლოკალური ასლების შესანახად კლიენტის კომპიუტერზე

გ. გლობალური ასლების შესანახად კლიენტის კომპიუტერზე

17.10.18. DataSet კლასის ობიექტებია:

ა. ცხრილები, წარმოდგენები, შენახული პროცედურები და ინდექსები

ბ. ცხრილები, ცხრილებს შორის კავშირები (ბმები), წარმოდგენები, შენახული პროცედურები და ფუნქციები

გ. წარმოდგენები, შენახული პროცედურები და ტრიგერები

17.10.19. DataSet კლასის ობიექტებთან მათთან მუშაობა:

ა. არ სრულდება ავტონომურ რეჟიმში

ბ. სრულდება ავტონომურ რეჟიმში

გ. სრულდება ონლაინ რეჟიმში

17.10.20. DataSet ობიექტები:

ა. არ შეიძლება გამოვიყენოთ მონაცემების წარმოსადგენად XML ფაილებიდან

ბ. შეიძლება გამოვიყენოთ მონაცემების წარმოსადგენად XML ფაილებიდან

გ. ხან შეიძლება და ხან არ შეიძლება გამოვიყენოთ მონაცემების წარმოსადგენად XML ფაილებიდან

17.10.21. DataTable კლასის ობიექტი:

ა. არ გამოიყენება ერთი ცხრილის შესანახად

ბ. გამოიყენება ერთი ცხრილის შესანახად

გ. გამოიყენება ორი ცხრილის შესანახად

17.10.22. ერთ DataSet ობიექტში:

ა. არ შეიძლება რამდენიმე DataTable ობიექტის ანუ რამდენიმე ცხრილის მოთავსება

ბ. შეიძლება რამდენიმე DataTable ობიექტის ანუ რამდენიმე ცხრილის მოთავსება

გ. შეიძლება მხოლოდ ერთი DataTable ობიექტის

17.10.23. DataRow კლასის ობიექტი:

ა. არ გამოიყენება ცხრილის ერთი სტრიქონის შესანახად

ბ. გამოიყენება ცხრილის ერთი სტრიქონის შესანახად

გ. გამოიყენება ცხრილის ორი სტრიქონის შესანახად

17.10.24. ერთ DataTable ობიექტში:

ა. არ შეიძლება რამდენიმე DataRow ობიექტის ანუ რამდენიმე სტრიქონის მოთავსება

ბ. შეიძლება რამდენიმე DataRow ობიექტის ანუ რამდენიმე სტრიქონის მოთავსება

გ. შეიძლება მხოლოდ ერთი DataRow ობიექტის

17.10.25. DataColumn კლასის ობიექტი:

ა. არ გამოიყენება ერთი სვეტის შესანახად

ბ. გამოიყენება ერთი სვეტის შესანახად

გ. გამოიყენება ორი სვეტის შესანახად

17.10.26. ერთ DataRow ობიექტში:

ა. არ შეიძლება რამდენიმე DataColumn ობიექტის ანუ რამდენიმე სვეტის მოთავსება

ბ. შეიძლება რამდენიმე DataColumn ობიექტის ანუ რამდენიმე სვეტის მოთავსება

გ. შეიძლება მხოლოდ ერთი DataColumn ობიექტის მოთავსება

17.10.27. DataRelation კლასის ობიექტი გამოიყენება:

ა. სამ DataTable ობიექტს ანუ სამ ცხრილს შორის კავშირის დასამყარებლად

ბ. ორ DataTable ობიექტს ანუ ორ ცხრილს შორის კავშირის დასამყარებლად

გ. ერთ DataTable ობიექტს ანუ ერთ ცხრილს შორის კავშირის დასამყარებლად

17.10.28. DataRelation ობიექტები შეგვიძლია გამოვიყენოთ მონაცემთა ბაზაში:

ა. სამ ცხრილს შორის "მთავარი-დამოკიდებული" კავშირის შესაქმნელად

ბ. ორ ცხრილს შორის "მთავარი-დამოკიდებული" კავშირის შესაქმნელად

გ. ერთ ცხრილს შორის "მთავარი-დამოკიდებული" კავშირის შესაქმნელად

17.10.29. ერთ DataSet ობიექტში:

ა. არ შეიძლება რამდენიმე DataRelation ობიექტის მოთავსება

ბ. შეიძლება რამდენიმე DataRelation ობიექტის მოთავსება

გ. შეიძლება მხოლოდ ერთი DataRelation ობიექტის მოთავსება

17.10.30. Constraint კლასის ობიექტი გამოიყენება ცხრილის შეზღუდვების შესანახად, როგორიცაა:

ა. სვეტის მნიშვნელობის შეზღუდვა მხოლოდ უნიკალურობაზე

ბ. სვეტის მნიშვნელობის შეზღუდვა უნიკალურობაზე ან გარე გასაღების შეზღუდვა, რომელიც სხვა ცხრილს მიმართავს

გ. მხოლოდ გარე გასაღების შეზღუდვა, რომელიც სხვა ცხრილს მიმართავს

17.10.31. Constraint კლასის ობიექტი:

ა. არ გამოიყენება ცხრილის შეზღუდვების შესანახად

ბ. გამოიყენება ცხრილის შეზღუდვების შესანახად

გ. გამოიყენება ცხრილებს შორის კავშირების შესანახად

17.10.32. ერთ DataTable ობიექტში:

ა. არ შეიძლება რამდენიმე Constraint ობიექტი მოვათავსოთ

ბ. შეიძლება რამდენიმე Constraint ობიექტი მოვათავსოთ

გ. შეიძლება მხოლოდ ერთი Constraint ობიექტი მოვათავსოთ

17.10.33. DataView კლასის ობიექტი:

ა. არ გამოიყენება DataTable ობიექტში მოთავსებული ცხრილის გაფილტვრისა და დახარისხებისათვის

ბ. გამოიყენება DataTable ობიექტში მოთავსებული ცხრილის გაფილტვრისა და დახარისხებისათვის

გ. გამოიყენება DataTable ობიექტში მოთავსებული ცხრილის მხოლოდ გაფილტვრისათვის

17.10.34. ერთი DataSet ობიექტი:

ა. არ შეიძლება რამდენიმე DataView ობიექტს შეიცავდეს

ბ. შეიძლება რამდენიმე DataView ობიექტს შეიცავდეს

გ. მხოლოდ ერთ DataView ობიექტს უნდა შეიცავდეს

17.10.35. DataSet, DataTable, DataRow, DataColumn, DataRelation, Constraint და DataView კლასები განსაზღვრულია:

ა. System.Linq სახელების სივრცეში

ბ. System.Data სახელების სივრცეში

გ. System.Collections.Generic სახელების სივრცეში

17.10.36. SqlConnection კლასის ობიექტი:

ა. არ გამოიყენება SQL სერვერზე მონაცემთა ბაზასთან შეერთების დასამყარებლად

ბ. გამოიყენება SQL სერვერზე მონაცემთა ბაზასთან შეერთების დასამყარებლად

გ. გამოიყენება SQL სერვერზე მონაცემთა ბაზასთან შეერთების გასაუქმებლად

17.10.37. OleDbConnection კლასის ობიექტი გამოიყენება იმ მონაცემთა ბაზის მართვის სისტემასთან მისაერთებლად, რომელიც:

ა. ODBC ტექნოლოგიას უზრუნველყოფს

ბ. OLEDB ტექნოლოგიას უზრუნველყოფს

გ. OLEDB ტექნოლოგიას არ უზრუნველყოფს

17.10.38. OLEDB ტექნოლოგიას:

ა. უზრუნველყოფს მხოლოდ Access

ბ. უზრუნველყოფს Oracle და Access

გ. არ უზრუნველყოფს Oracle და Access

17.10.39. OdbcConnection კლასის ობიექტი გამოიყენება იმ მონაცემთა ბაზის მართვის სისტემასთან მისაერთებლად, რომელიც:

ა. ODBC ტექნოლოგიას არ უზრუნველყოფს

ბ. ODBC ტექნოლოგიას უზრუნველყოფს
გ. OLEDB ტექნოლოგიას უზრუნველყოფს

17.10.40. ყველა ძირითადი მონაცემთა ბაზა უზრუნველყოფს ODBC ტექნოლოგიას, მაგრამ მისი საშუალებით მონაცემთა ბაზასთან მიმართვა .NET სისტემაში:
ა. სწრაფად სრულდება
ბ. ნელა სრულდება
გ. არ სრულდება

17.10.41. SqlCommand, OleDbCommand, OdbcCommand კლასების ობიექტები:
ა. არ გამოიყენება SQL მოთხოვნის ფორმირებისათვის ან შენახული პროცედურის გამოძახებისათვის
ბ. გამოიყენება SQL მოთხოვნის ფორმირებისათვის ან შენახული პროცედურის გამოძახებისათვის
გ. შენახული პროცედურის წასაშლელად

17.10.42. SqlDataReader, OleDbDataReader, OdbcDataReader კლასების ობიექტები გამოიყენება ინფორმაციის წასაკითხად მონაცემთა ბაზიდან, რომელიც:
ა. არაა მოთავსებული SQL სერვერზე
ბ. მოთავსებულია SQL სერვერზე, ან მუშაობს იმ მონაცემთა ბაზის მართვის სისტემის ქვეშ, რომელიც უზრუნველყოფს OLEDB ან ODBC ტექნოლოგიას
გ. არ მუშაობს იმ მონაცემთა ბაზის მართვის სისტემის ქვეშ, რომელიც უზრუნველყოფს OLEDB ან ODBC ტექნოლოგიას

17.10.43. SqlDataReader, OleDbDataReader, OdbcDataReader კლასების ობიექტები გამოიყენება მონაცემების:
ა. მხოლოდ ჩასაწერად მონაცემების წყაროდან და წარმოადგენს DataSet ობიექტის ალტერნატივას
ბ. მხოლოდ წასაკითხად მონაცემების წყაროდან და წარმოადგენს DataSet ობიექტის ალტერნატივას
გ. მხოლოდ წასაკითხად მონაცემების წყაროდან და არ წარმოადგენს DataSet ობიექტის ალტერნატივას

17.10.44. SqlDataReader, OleDbDataReader, OdbcDataReader კლასების ობიექტებით მონაცემების წაკითხვა უფრო:
ა. ნელა სრულდება, ვიდრე DataSet ობიექტის გამოყენებით
ბ. სწრაფად სრულდება, ვიდრე DataSet ობიექტის გამოყენებით
გ. სწრაფად არ სრულდება, ვიდრე DataSet ობიექტის გამოყენებით

17.10.45. SqlDataAdapter, OleDbDataAdapter, OdbcDataAdapter კლასების ობიექტები გამოიყენება სტრიქონების გადასაადგილებლად DataSet ობიექტსა და მონაცემთა ბაზას შორის, რომელიც:
ა. მოთავსებულია SQL სერვერზე, ან მუშაობს იმ მონაცემთა ბაზის მართვის სისტემის ქვეშ, რომელიც უზრუნველყოფს OLEDB ან ODBC ტექნოლოგიას
ბ. არაა მოთავსებული SQL სერვერზე
გ. არ მუშაობს იმ მონაცემთა ბაზის მართვის სისტემის ქვეშ, რომელიც უზრუნველყოფს OLEDB ან ODBC ტექნოლოგიას

17.10.46. მართვადი კლასები SQL სერვერისათვის გამოცხადებულია:
ა. System.Data.SqlClient სახელების სივრცეში
ბ. System.Data.Linq სახელების სივრცეში
გ. System.Data სახელების სივრცეში

17.10.47. კლასები მონაცემთა ბაზებისათვის, რომლებიც უზრუნველყოფს ODBC ტექნოლოგიას, გამოცხადებულია:
ა. System.Data.Odbc სახელების სივრცეში
ბ. System.Data.Linq სახელების სივრცეში
გ. System.Data.Data სახელების სივრცეში

17.10.48. C# პროგრამაში ADO.NET კლასების გამოყენებამდე using დირექტივების ბლოკში უნდა მოვათავსოთ დირექტივა:
ა. using System.Data
ბ. using System.ComponentModel
გ. using System.Linq

17.10.49. ADO.NET კლასები მოთავსებულია:
ა. using System.Data სახელების სივრცეში
ბ. using System.Linq სახელების სივრცეში
გ. using System.ComponentModel სახელების სივრცეში

17.10.50. თუ ვმუშაობთ SQL სერვერზე მოთავსებულ მონაცემთა ბაზასთან, მაშინ using დირექტივების ბლოკში უნდა მოვათავსოთ დირექტივა:
ა. using System.Data.SqlClient
ბ. using System.Data.Odbc
გ. using System.Data.OleDB

17.10.51. თუ ვიყენებთ SQL სერვერისა და Oracle-საგან განსხვავებული მონაცემთა ბაზას, მაგალითად Access, მაშინ using დირექტივების ბლოკში უნდა მოვათავსოთ დირექტივა:
ა. using System.Data.OleDB
ბ. using System.Data.SqlClient
გ. using System.Data.Odbc

17.10.52. თუ ვიყენებთ მონაცემთა წყაროებს, რომლებისთვისაც არ არსებობს „მშობლიური“ OLE DB მომწოდებელი, მაშინ using დირექტივების ბლოკში უნდა მოვათავსოთ დირექტივა:
ა. using System.Data.Odbc
ბ. using System.Data.OleDB
გ. using System.Data.SqlClient

მოთხოვნების შესრულება ADO.NET კლასების გამოყენებით

მონაცემების წაკითხვა DataReader ობიექტის საშუალებით

17.11.1. DataReader ობიექტი SELECT ბრძანებაში (მოთხოვნაში) მითითებული ცხრილიდან:
ა. თითო სტრიქონს კითხულობს

ბ. თითო სტრიქონს არ კითხულობს
გ. ორ-ორ სტრიქონს კითხულობს

17.11.2. DataReader ობიექტი:
ა. UPDATE ბრძანებაში (მოთხოვნაში) მითითებული ცხრილიდან თითო სტრიქონს კითხულობს
ბ. SELECT ბრძანებაში (მოთხოვნაში) მითითებული ცხრილიდან თითო სტრიქონს კითხულობს
გ. INSERT ბრძანებაში (მოთხოვნაში) მითითებული ცხრილიდან თითო სტრიქონს კითხულობს

17.11.3. DataReader ობიექტი SELECT ბრძანებაში (მოთხოვნაში) მითითებული ცხრილიდან თითო სტრიქონს კითხულობს. წაკითხვის პროცესი შემდეგი მოქმედებებისაგან შედგება:
ა. 1. შეერთების ობიექტის შექმნა; 2. SELECT ბრძანების ფორმირება; 3. SqlDataReader ტიპის ობიექტის საშუალებით მონაცემების წაკითხვა და ეკრანზე მათი გამოტანა; 4. DataReader-ის დახურვა
ბ. 1. შეერთების ობიექტის შექმნა; 2. შეერთების გახსნა; 3. SELECT ბრძანების ფორმირება; 4. SqlDataReader ტიპის ობიექტის საშუალებით მონაცემების წაკითხვა და ეკრანზე მათი გამოტანა; 5. DataReader-სა და შეერთების დახურვა
გ. 1. შეერთების ობიექტის შექმნა; 2. SqlDataReader ტიპის ობიექტის საშუალებით მონაცემების წაკითხვა და ეკრანზე მათი გამოტანა; 3. შეერთების დახურვა

17.11.4. მონაცემთა წყაროსთან შეერთების ობიექტი შემდეგნაირად იქმნება:
ა. SqlConnection myConnection =
new SqlConnection("database=Shekveta; uid=sa; pwd=paroli");
ბ. SqlConnection myConnection =
new SqlConnection("server=ROMANI-PC; database=Shekveta; uid=sa; pwd=paroli");
გ. SqlConnection myConnection =
new SqlConnection("server=ROMANI-PC; uid=sa; pwd=paroli");

17.11.5. შეერთების სტრიქონი შემდეგ პარამეტრებს შეიცავს:
ა. მონაცემთა ბაზის სახელი, მომხმარებლის სახელი, პაროლი
ბ. სერვერის სახელი, მონაცემთა ბაზის სახელი, მომხმარებლის სახელი, პაროლი
გ. მონაცემთა ბაზის სახელი, მომხმარებლის სახელი, პაროლი

17.11.6. სერვერის სახელი მიეთითება შეერთების სტრიქონის:
ა. uid პარამეტრში
ბ. database პარამეტრში
გ. server პარამეტრში

17.11.7. თუ SQL სერვერი ქსელურ კომპიუტერზეა მოთავსებული, მაშინ:
ა. უნდა მივუთითოთ ფაილის სახელი
ბ. არ უნდა მივუთითოთ კომპიუტერის სახელი
გ. უნდა მივუთითოთ კომპიუტერის სახელი

17.11.8. თუ სერვერი ჩვენს (ლოკალურ) კომპიუტერზე მუშაობს, მაშინ:
ა. აუცილებლად უნდა მივუთითოთ localhost სიტყვა

ბ. არ შეგვიძლია მივუთითოთ როგორც კომპიუტერის სახელი, ისე localhost სიტყვა
გ. შეგვიძლია მივუთითოთ როგორც კომპიუტერის სახელი, ისე localhost სიტყვა

17.11.9. მონაცემთა ბაზის სახელი მიეთითება შეერთების სტრიქონის:
ა. uid პარამეტრში
ბ. server პარამეტრში
გ. database პარამეტრში

17.11.10. მომხმარებლის სახელი მიეთითება შეერთების სტრიქონის:
ა. server პარამეტრში
ბ. database პარამეტრში
გ. uid პარამეტრში

17.11.11. პაროლი მიეთითება შეერთების სტრიქონის:
ა. database პარამეტრში
ბ. uid პარამეტრში
გ. pwd პარამეტრში

17.11.12. შეერთების გასახსნელად უნდა შევასრულოთ:
ა. myReader.Close(); მეთოდი
ბ. myConnection.Close(); მეთოდი
გ. myConnection.Open(); მეთოდი

17.11.13. myCommand.CommandText = "SELECT gvari, ganyofileba, asaki, tarigi_dabadebis FROM Personali"; ამ SELECT ბრძანების შესრულების შედეგად მიღებულ სტრიქონში მოთავსებული იქნება მხოლოდ:
ა. asaki და tarigi_dabadebis სვეტები
ბ. gvari და ganyofileba სვეტები
გ. მითითებული სვეტები

17.11.14. myCommand.CommandText = "SELECT gvari, ganyofileba, asaki, tarigi_dabadebis FROM Personali"; ამ SELECT ბრძანების შესრულების შედეგად მიღებულ სტრიქონში მოთავსებული რომ იყოს ყველა სვეტი, SELECT სიტყვის შემდეგ სვეტების სახელების ნაცვლად უნდა მივუთითოთ:
ა. +
ბ. !
გ. *

17.11.15. ExecuteReader() მეთოდი ეკუთვნის:
ა. SqlDataReader ობიექტს
ბ. SqlConnection ობიექტს
გ. SqlCommand ობიექტს

17.11.16. ExecuteReader() მეთოდი ასრულებს:
ა. UPDATE ბრძანებას და ქმნის ობიექტ-წამკითხველს გენერირებული შედეგების წაკითხვისათვის
ბ. SELECT ბრძანებას და არ ქმნის ობიექტ-წამკითხველს გენერირებული შედეგების

წაკითხვისათვის

გ. SELECT ბრძანებას და ქმნის ობიექტ-წამკითხველს გენერირებული შედეგების წაკითხვისათვის

17.11.17. ობიექტი-წამკითხველი:

ა. უნდა მივანიჭოთ SELECT ობიექტს

ბ. არ უნდა მივანიჭოთ SqlDataReader ობიექტს

გ. უნდა მივანიჭოთ SqlDataReader ობიექტს

17.11.18. ობიექტი-წამკითხველიდან შედეგების მიღების ერთ-ერთი გზაა:

ა. SqlDataWriter ობიექტის Read() მეთოდის გამოყენება

ბ. SqlDataReader ობიექტის Write() მეთოდის გამოყენება

გ. SqlDataReader ობიექტის Read() მეთოდის გამოყენება

17.11.19. SqlDataReader ობიექტის Read() მეთოდი კითხულობს ერთ სტრიქონს, რომელიც მიღებული იყო SELECT ბრძანების შესრულების შედეგად და:

ა. გასცემს true-ს, თუ არ არის წასაკითხი სტრიქონები, წინააღმდეგ შემთხვევაში – false-ს

ბ. გასცემს false-ს, თუ კიდევ არის წასაკითხი სტრიქონები, წინააღმდეგ შემთხვევაში –true-ს

გ. გასცემს true-ს, თუ კიდევ არის წასაკითხი სტრიქონები, წინააღმდეგ შემთხვევაში –false-ს

17.11.20. DataReader-ის დასახურად უნდა შევასრულოთ:

ა. myReader.Open(); მეთოდი

ბ. Fill() მეთოდი

გ. myReader.Close(); მეთოდი

17.11.21. შეერთების დასახურად უნდა შევასრულოთ:

ა. Fill() მეთოდი

ბ. myReader.Close(); მეთოდი

გ. myConnection.Close(); მეთოდი

17.11.22. string connectionstring = "server=Computer; database=Shekveta_1; uid=sa; pwd=paroli"; მონაცემთა ბაზასთან შეერთების სტრიქონში სერვერის სახელია:

ა. Shekveta_1

ბ. sa

გ. Computer

17.11.23. string connectionstring = "server=Computer; database=Shekveta_1; uid=sa; pwd=paroli"; მონაცემთა ბაზასთან შეერთების სტრიქონში მონაცემთა ბაზის სახელია:

ა. Shekveta_1

ბ. sa

გ. Computer

17.11.24. string connectionstring = "server=Computer; database=Shekveta_1; uid=sa; pwd=paroli"; მონაცემთა ბაზასთან შეერთების სტრიქონში მომხმარებლის სახელია:

ა. Shekveta_1

ბ. sa

გ. Computer

17.11.25. string connectionstring = "server=Computer; database=Shekveta_1; uid=sa; pwd=paroli"; მონაცემთა ბაზასთან შეერთების სტრიქონში პაროლია:

ა. Shekveta_1

ბ. paroli

გ. Computer

17.11.26. თუ სერვერი ჩვენს (ლოკალურ) კომპიუტერზე მუშაობს, მაშინ:

ა. არ შეგვიძლია მივუთითოთ როგორც კომპიუტერის სახელი, ისე localhost სიტყვა, server = localhost

ბ. შეგვიძლია მივუთითოთ როგორც კომპიუტერის სახელი, ისე localhost სიტყვა, server = localhost

გ. უნდა მივუთითოთ მხოლოდ კომპიუტერის სახელი

მონაცემთა ბაზასთან მუშაობა DataSet ობიექტის გამოყენებით

მონაცემების კითხვა

17.12.1. DataSet ობიექტი მოიცავს:

ა. DataRelation და DataTable ობიექტებს

ბ. მხოლოდ DataRelation ობიექტს

გ. მხოლოდ DataTable ობიექტს

17.12.2. DataTable ობიექტი მოიცავს:

ა. DataRow ობიექტებს

ბ. DataRelation ობიექტებს

გ. DataColumn ობიექტებს

17.12.3. DataRow ობიექტი მოიცავს:

ა. DataColumn ობიექტებს

ბ. DataTable ობიექტებს

გ. DataRelation ობიექტებს

17.12.4. DataSet ობიექტში შეგვიძლია მოვათავსოთ:

ა. მხოლოდ ერთი ცხრილი

ბ. ერთი ან მეტი ცხრილი

გ. მხოლოდ ორი ცხრილი

17.12.5. ცხრილებით DataSet ობიექტის შესავსებად გამოიყენება:

ა. ადაპტერის Update() მეთოდი

ბ. ადაპტერის Fill() მეთოდი

გ. ადაპტერის Insert() მეთოდი

17.12.6. ადაპტერი არის ობიექტი, რომელიც გამოიყენება მონაცემების გადასაგზავნად:

ა. მხოლოდ კლიენტიდან სერვერზე

ბ. კლიენტიდან (ჩვენი კომპიუტერიდან) სერვერზე და პირიქით

გ. მხოლოდ სერვერიდან კლიენტზე

17.12.7. DataSet ობიექტს აქვს Tables თვისება, რომელიც:

ა. არ წარმოადგენს DataTable ტიპის ობიექტების კოლექციას

ბ. წარმოადგენს DataTable ტიპის ობიექტების კოლექციას

გ. წარმოადგენს DataSet ტიპის ობიექტების კოლექციას

17.12.8. DataSet ობიექტის Tables თვისება საშუალებას გვაძლევს ცხრილებს მივმართოთ:

ა. მხოლოდ სახელის მიხედვით

ბ. როგორც სახელის, ისე ინდექსის მიხედვით

გ. მხოლოდ ინდექსის მიხედვით

17.12.9. არასწორია ჩანაწერი:

ა. myDataSet.Tables["Personali"];

ბ. myDataSet.Tables["Personali", 0];

გ. myDataSet.Tables[0];

17.12.10. DataSet ობიექტში ცხრილების ნუმერაცია (ინდექსაცია):

ა. ერთიდან იწყება

ბ. ნულიდან იწყება

გ. ორიდან იწყება

17.12.11. სწორია კოდი:

ა.

```
DataRow personaliTable = new DataTable();
personaliTable = myDataSet.Tables["Personali"];
```

ბ.

```
DataTable personaliTable = new DataTable();
personaliTable = myDataSet.Tables["Personali"];
```

გ.

```
DataColumn personaliTable = new DataTable();
personaliTable = myDataSet.Tables["Personali"];
```

17.12.12. Tables თვისებას აქვს Rows თვისება, რომელიც არის:

ა. DataColumn ობიექტების კოლექცია

ბ. DataRow ობიექტების კოლექცია

გ. DataTable ობიექტების კოლექცია

17.12.13. Tables თვისების Rows თვისება საშუალებას:

ა. არ გვაძლევს ცხრილის სტრიქონს მივმართოთ ინდექსის მიხედვით

ბ. გვაძლევს ცხრილის სტრიქონს მივმართოთ ინდექსის მიხედვით

გ. გვაძლევს ცხრილის სვეტს მივმართოთ ინდექსის მიხედვით

17.12.14. DataRow myRow = myDataSet.Tables["Personali"].Rows[3]; კოდის შესრულების შედეგად myRow ობიექტში ჩაიწერება Personali ცხრილის რიგით:

ა. მეორე სტრიქონი

ბ. მეოთხე სტრიქონი

გ. მესამე სტრიქონი

17.12.15. Tables თვისების Rows თვისებაში სტრიქონების ინდექსაცია იწყება :

ა. 2-დან

ბ. 0-დან

გ. 1-დან

17.12.16. Tables თვისების Rows თვისება საშუალებას გვაძლევს მივმართოთ სტრიქონის სვეტებს:

ა. მხოლოდ სახელის მიხედვით

ბ. სახელის ან ინდექსის მიხედვით

გ. მხოლოდ ინდექსის მიხედვით

17.12.17. არასწორია კოდი:

ა. object gvari_1 = myDataSet.Tables["Personali"].Rows[2]["gvari"];

ბ. object gvari_1 = myDataSet.Tables["Personali"].Rows[2]["gvari", 0];

გ. object gvari_2 = myDataSet.Tables["Personali"].Rows[2][1];

17.12.18. მოცემული კოდის:

string gvari = myDataSet.Tables["Personali"].Rows[2]["gvari"].ToString();

შესრულების შედეგად:

ა. gvari ცვლადში ჩაიწერება "Personali" ცხრილის მნიშვნელობა

ბ. gvari ცვლადში ჩაიწერება "gvari" სვეტის მნიშვნელობა

გ. "gvari" სვეტში ჩაიწერება gvari ცვლადის

17.12.19. Tables თვისების Rows თვისებაში სვეტების ნუმერაცია იწყება:

ა. 2-დან

ბ. 0-დან

გ. 1-დან

17.12.20. myAdapter.Fill(myDataset,"Personali"); კოდის შესრულების შედეგად:

ა. myDataset ობიექტში არ იქმნება DataTable ტიპის Personali ობიექტი, რომელიც ივსება სტრიქონებით Shekveta მონაცემთა ბაზის Personali ცხრილიდან

ბ. myDataset ობიექტში იქმნება DataTable ტიპის Personali ობიექტი, რომელიც ივსება სტრიქონებით Shekveta მონაცემთა ბაზის Personali ცხრილიდან

გ. myDataset ობიექტში იქმნება DataTable ტიპის Personali ობიექტი, რომელიც არ ივსება სტრიქონებით Shekveta მონაცემთა ბაზის Personali ცხრილიდან

17.12.21. Fill() მეთოდი ასრულებს myAdapter ობიექტთან დაკავშირებულ:

ა. INSERT მოთხოვნას

ბ. SELECT მოთხოვნას

გ. UPDAT მოთხოვნას

17.12.22. myAdapter.Fill(myDataset,"Personali"); კოდის შესრულების შედეგად:

ა. Personali ობიექტი მოთავსებული იქნება არა კლიენტის კომპიუტერზე, არამედ სერვერზე

ბ. Personali ობიექტი მოთავსებული იქნება არა სერვერზე, არამედ კლიენტის კომპიუტერზე

გ. Personali ობიექტი მოთავსებული იქნება მხოლოდ სერვერზე

17.12.23. ცხრილებით DataSet ობიექტის შევსების შემდეგ, ამ ცხრილებთან მუშაობა:

ა. არ შეგვეძლება ავტონომიურ (ოფლაინ) რეჟიმში ანუ სერვერთან კავშირის გარეშე

ბ. შეგვეძლება ავტონომიურ (ოფლაინ) რეჟიმში ანუ სერვერთან კავშირის გარეშე
გ. არ შეგვეძლება ავტონომიურ (ოფლაინ) რეჟიმში ანუ სერვერთან კავშირის გარეშე

17.12.24. მონაცემთა ბაზასთან შეერთების აშკარად გახსნა ან დახურვა:
ა. აუცილებელია, რადგან ამას ადაპტერი ავტომატურად არ აკეთებს
ბ. აუცილებელი არ არის, რადგან ამას ადაპტერი ავტომატურად აკეთებს
გ. აუცილებელი არ არის, რადგან ამას ადაპტერი ავტომატურად არ აკეთებს

17.12.25. ობიექტი-ადაპტერი, საჭიროების მიხედვით:
ა. არ ხსნის, მაგრამ ხურავს შეეთებას მონაცემთა ბაზასთან
ბ. ხსნის და ხურავს შეეთებას მონაცემთა ბაზასთან
გ. ხსნის, მაგრამ არ ხურავს შეეთებას მონაცემთა ბაზასთან

17.12.26. ადაპტერი შეერთების მდგომარეობას უცვლელად ინარჩუნებს. ამიტომ, თუ შეერთება გახსნილი იყო ადაპტერის მუშაობის:
ა. დაწყებამდე, მაშინ ადაპტერის მუშაობის დამთავრების შემდეგ შეერთება აღარ დარჩება გახსნილი
ბ. დაწყებამდე, მაშინ ადაპტერის მუშაობის დამთავრების შემდეგ შეერთება ისევ დარჩება გახსნილი
გ. დაწყების შემდეგ, მაშინ ადაპტერის მუშაობის დამთავრების შემდეგ შეერთება ისევ დარჩება გახსნილი

17.12.27. DataReader ობიექტი მისი მუშაობის მთელი პერიოდის განმავლობაში:
ა. არ ითხოვს მონაცემთა ბაზასთან შეერთების არსებობას
ბ. ითხოვს მონაცემთა ბაზასთან შეერთების არსებობას
გ. ხანდახან ითხოვს მონაცემთა ბაზასთან შეერთების არსებობას

17.12.28. dataGridView კომპონენტში ცხრილის სტრიქონების გამოტანა გაცილებით:
ა. ნელა სრულდება, ვიდრე label კომპონენტში
ბ. სწრაფად სრულდება, ვიდრე label კომპონენტში
გ. ნელა სრულდება, ვიდრე textBox კომპონენტში

17.12.29.
SqlDataAdapter personaliAdapter = new SqlDataAdapter("SELECT * FROM Personali", myConnection);
კოდის შესრულებისას გაიცემა Personali ცხრილის:
ა. პირველი სვეტი და ყველა სტრიქონი
ბ. ყველა სვეტი და ყველა სტრიქონი
გ. ყველა სვეტი და პირველი სტრიქონი

17.12.30. თუ ადაპტერის შექმნისას SELECT ბრძანება გასცემს ყველა სტრიქონსა და ყველა სვეტს, მაშინ:
ა. ასეთი მიდგომა უმჯობესია გამოვიყენოთ დიდი ზომის ცხრილებისათვის, რომლებშიც სვეტებისა და სტრიქონების რაოდენობა დიდია
ბ. ასეთი მიდგომა უმჯობესია გამოვიყენოთ მცირე ზომის ცხრილებისათვის, რომლებშიც სვეტებისა და სტრიქონების რაოდენობა მცირეა
გ. ასეთი მიდგომა უმჯობესია არ გამოვიყენოთ მცირე ზომის ცხრილებისათვის, რომლებშიც სვეტებისა და სტრიქონების რაოდენობა მცირეა

17.12.31. როცა ცხრილში სტრიქონების რაოდენობა აღემატება 100000, მაშინ:

ა. უმჯობესია გამოვიყენოთ მხოლოდ ვერტიკალური ფილტრი

ბ. უმჯობესია გამოვიყენოთ როგორც ვერტიკალური, ისე ჰორიზონტალური ფილტრი

გ. უმჯობესია გამოვიყენოთ მხოლოდ ჰორიზონტალური ფილტრი

17.12.32. ვერტიკალური ფილტრი გვექნება მაშინ, როცა SELECT ბრძანებაში:

ა. არ მივუთითებთ ცხრილის მხოლოდ იმ სვეტებს, რომლებთანაც რეალურად ვაპირებთ მუშაობას

ბ. მივუთითებთ ცხრილის მხოლოდ იმ სვეტებს, რომლებთანაც რეალურად ვაპირებთ მუშაობას

გ. მივუთითებთ ცხრილის მხოლოდ იმ სვეტებს, რომლებთანაც რეალურად არ ვაპირებთ მუშაობას

17.12.33. SqlDataAdapter personaliAdapter = new SqlDataAdapter(
"SELECT gvari, staji, xelfasi FROM Personali", myConnection); კოდში:

ა. გამოყენებულია ჰორიზონტალური ფილტრი

ბ. გამოყენებულია ვერტიკალური ფილტრი

გ. არაა გამოყენებული ვერტიკალური ფილტრი

17.12.34. ვერტიკალური ფილტრის გამოყენება არ გამოდგება იმ შემთხვევაში, როცა:

ა. ცხრილიდან სტრიქონებს ვშლით

ბ. ცხრილს სტრიქონებს ვუმატებთ, რადგან ამ დროს სტრიქონის თითოეულ სვეტს უნდა მივანიჭოთ შესაბამისი მნიშვნელობა

გ. ცხრილში სტრიქონებს ვცვლით

17.12.35. ჰორიზონტალური ფილტრის შემთხვევაში SELECT ბრძანებაში:

ა. არ უნდა მივუთითოთ WHERE განყოფილება

ბ. უნდა მივუთითოთ WHERE განყოფილება

გ. უნდა მივუთითოთ GROUP BY განყოფილება

17.12.36. ჰორიზონტალური ფილტრის შემთხვევაში ამოირჩევა ცხრილის:

ა. ერთი ან მეტი სვეტი, რომლებთანაც რეალურად ვაპირებთ მუშაობას

ბ. ერთი ან მეტი სტრიქონი, რომლებთანაც რეალურად ვაპირებთ მუშაობას

გ. მხოლოდ ერთი სტრიქონი, რომელთანაც რეალურად ვაპირებთ მუშაობას

17.12.37. SqlDataAdapter personaliAdapter = new SqlDataAdapter(
"SELECT * FROM Personali WHERE staji >= 20", myConnection); კოდში:

ა. გამოყენებულია ვერტიკალური ფილტრი

ბ. გამოყენებულია ჰორიზონტალური ფილტრი

გ. არაა გამოყენებული ჰორიზონტალური ფილტრი

17.12.38. SqlDataAdapter personaliAdapter = new SqlDataAdapter(
"SELECT gvari, asaki, staji, xelfasi FROM Personali WHERE staji >= 20", myConnection); კოდში:

ა. გამოყენებულია მხოლოდ ვერტიკალური ფილტრი

ბ. გამოყენებულია როგორც ვერტიკალური, ისე ჰორიზონტალური ფილტრი

გ. გამოყენებულია მხოლოდ ჰორიზონტალური ფილტრი

მონაცემების გაფილტვრა

17.13.1. მონაცემების გასაფილტრად:

ა. არ შეგვიძლია DataView ობიექტის გამოყენება

ბ. შეგვიძლია DataView ობიექტის გამოყენება

გ. შეგვიძლია SqlDataAdapter ობიექტის გამოყენება

17.13.2. ფილტრი უნდა მივანიჭოთ dataView ობიექტის:

ა. Sort თვისებას

ბ. RowFilter თვისებას

გ. Count თვისებას

17.13.3. dataView ობიექტის RowFilter თვისებას უნდა მივანიჭოთ ფილტრი, რომელიც:

ა. მთელ რიცხვს წარმოადგენს

ბ. სტრიქონს წარმოადგენს

გ. წილადს წარმოადგენს

17.13.4. dataView1.RowFilter = "ganyofileba = 'სავაჭრო' AND asaki > 30"; კოდის შესრულების შედეგად ეკრანზე გამოჩნდება სტრიქონები, რომლებიც შეიცავს ინფორმაციას იმ თანამშრომლების შესახებ, რომლებიც:

ა. არ მუშაობენ სავაჭრო განყოფილებაში და მათი ასაკი აღემატება 30 წელს

ბ. მუშაობენ სავაჭრო განყოფილებაში და მათი ასაკი აღემატება 30 წელს

გ. მუშაობენ სავაჭრო განყოფილებაში და მათი ასაკი არ აღემატება 30 წელს

17.13.5. ფილტრის გასაუქმებლად უნდა შევასრულოთ კოდი:

ა. dataView1.RowFilter = "ganyofileba = 'სავაჭრო' AND asaki > 30";

ბ. dataView1.RowFilter = "";

გ. dataView1.RowFilter = 0;

17.13.6. dataView კომპონენტის RowFilter თვისებაში ფილტრის შესატანად:

ა. არ შეგვიძლია textBox კომპონენტის გამოყენებაც

ბ. შეგვიძლია textBox კომპონენტის გამოყენებაც

გ. შეგვიძლია dataGridView კომპონენტის გამოყენებაც

17.13.7. რომელი კოდია სწორი:

ა. dataView1.RowFilter = dataGridView1.Text;

ბ. dataView1.RowFilter = textBox1.Text;

გ. dataView1.RowFilter = label1.Text;

17.13.8. textBox კომპონენტში ფილტრი შეგვაქვს:

ა. ბრჭყალების მითითებით

ბ. ბრჭყალების გარეშე

გ. აპოსტროფების მითითებით

17.13.9. dataView1.RowFilter = "tarigi_dabadebis > '01.01.1990'" კოდის შესრულების შედეგად გაიცემა ინფორმაცია იმ თანამშრომლების შესახებ, რომლებიც:

ა. არ დაიბადნენ 1990 წლის 1-ლი იანვრის შემდეგ

ბ. დაიბადნენ 1990 წლის 1-ლი იანვრის შემდეგ

გ. დაიბადნენ 1990 წლის 1-ლი იანვრამდე

17.13.10. dataView კომპონენტის RowFilter თვისებაში თარიღის შეტანა:
ა. შეუძლებელია dateTimePicker კომპონენტიდან
ბ. შეგვიძლია dateTimePicker კომპონენტიდან
გ. შეგვიძლია label კომპონენტიდან

მონაცემების დახარისხება

17.14.1. მონაცემების დასახარისხებლად:
ა. არ შეგვიძლია DataView ობიექტის გამოყენება
ბ. შეგვიძლია DataView ობიექტის გამოყენება
გ. შეგვიძლია label ობიექტის გამოყენება

17.14.2. მონაცემების დასახარისხებლად უნდა გამოვიყენოთ DataView ობიექტის:
ა. RowFilter თვისება
ბ. Sort თვისება
გ. Count თვისება

17.14.3. dataView1.Sort = "ganyofileba, asaki DESC"; კოდის შესრულების შედეგად ცხრილი დახარისხდება:
ა. კლებადობით ganyofileba სვეტის მიხედვით და ზრდადობით asaki სვეტის მიხედვით
ბ. ზრდადობით ganyofileba სვეტის მიხედვით და კლებადობით asaki სვეტის მიხედვით
გ. ზრდადობით ganyofileba სვეტის მიხედვით და ზრდადობით asaki სვეტის მიხედვით

17.14.4. დახარისხების გაუქმება შემდეგნაირად შეგვიძლია:
ა. dataView1.Sort = 0;
ბ. dataView1.Sort = "";
გ. dataView1.Sort = "ganyofileba, asaki DESC";

სტრიქონების ძებნა

17.15.1. ცხრილში სტრიქონის მოსაძებნად გამოიყენება:
ა. Sort თვისება
ბ. Find() მეთოდი
გ. RowFilter თვისება

17.15.2. Find() მეთოდი:
ა. არ ითხოვს ცხრილში პირველადი გასაღების დაყენებას
ბ. ითხოვს ცხრილში პირველადი გასაღების დაყენებას
გ. ითხოვს ცხრილში გარე გასაღების დაყენებას

17.15.3. Find() მეთოდი საჭირო მნიშვნელობას:
ა. ეძებს გარე გასაღებში
ბ. ეძებს პირველად გასაღებში
გ. არ ეძებს პირველად გასაღებში

17.15.4. პირველადი გასაღები:
ა. არ შეიძლება შედგებოდეს ერთი ან მეტი სვეტისგან, რომელთა მნიშვნელობები ცალსახად განსაზღვრავს ერთ სტრიქონს

ბ. შეიძლება შედგებოდეს ერთი ან მეტი სვეტისგან, რომელთა მნიშვნელობები ცალსახად განსაზღვრავს ერთ სტრიქონს
გ. შეიძლება შედგებოდეს ერთი ან მეტი სვეტისგან, რომელთა მნიშვნელობები ცალსახად არ განსაზღვრავს ერთ სტრიქონს

17.15.5. Find() მეთოდის მიერ შესრულებული ძებნის შედეგი ყოველთვის:
ა. არ იქნება ერთი სტრიქონი
ბ. იქნება ერთი სტრიქონი
გ. იქნება რამდენიმე სტრიქონი

17.15.6. DataRow findRow = myDataset.Tables["Personali"].Rows.Find(find_key); კოდის შესრულების შედეგად findRow ობიექტს:
ა. მიენიჭება null მნიშვნელობა
ბ. მიენიჭება ნაპოვნი სტრიქონი
გ. არ მიენიჭება ნაპოვნი სტრიქონი

17.15.7. თუ Find() მეთოდმა სტრიქონი ვერ იპოვა, მაშინ გაიცემა:
ა. 0
ბ. null
გ. -1

17.15.8. { DataColumn[] keys = new DataColumn[1];
keys[0] = myDataset.Tables["Personali"].Columns["PersonaliID"];
myDataset.Tables["Personali"].PrimaryKey = keys; }
კოდის შესრულების შედეგად:
ა. განისაზღვრება გარე გასაღები
ბ. განისაზღვრება პირველადი გასაღები
გ. არ განისაზღვრება პირველადი გასაღები

17.15.9. { DataColumn[] keys = new DataColumn[1];
keys[0] = myDataset.Tables["Personali"].Columns["PersonaliID"];
myDataset.Tables["Personali"].PrimaryKey = keys; }
კოდის შესრულების შედეგად:
ა. პირველადი გასაღები იქნება PersonaliID ცხრილის Personali სვეტი
ბ. პირველადი გასაღები იქნება Personali ცხრილის PersonaliID სვეტი
გ. გარე გასაღები იქნება Personali ცხრილის PersonaliID სვეტი

17.15.10. Find() მეთოდი:
ა. არ გასცემს DataRow ობიექტს
ბ. გასცემს DataRow ობიექტს
გ. გასცემს DataColumn ობიექტს

17.15.11. Find() მეთოდს არგუმენტად:
ა. არ გადაეცემა საძებნი სიდიდე
ბ. გადაეცემა საძებნი სიდიდე
გ. არაფერი არ გადაეცემა

17.15.12. თუ პირველადი გასაღები ორი და მეტი სვეტისგან შედგება, მაშინ Find() მეთოდს არგუმენტად:

ა. არ უნდა გადავცეთ ობიექტების მასივი

ბ. უნდა გადავცეთ ობიექტების მასივი

გ. უნდა გადავცეთ ერთი მნიშვნელობა

მონაცემთა ბაზაში მონაცემების შეცვლა

17.16.1. ცხრილებში მოთავსებული მონაცემების შესაცვლელად უნდა შევასრულოთ შემდეგი მოქმედებები:

ა. 1. DataSet ობიექტის შევსება იმ ცხრილებით, რომლებშიც გვინდა მონაცემების შეცვლა; 2. DataSet ობიექტში მოთავსებული ცხრილების გადაგზავნა სერვერზე მონაცემთა ბაზაში; 3. DataSet ობიექტში ცხრილების შეცვლა

ბ. 1. DataSet ობიექტის შევსება იმ ცხრილებით, რომლებშიც გვინდა მონაცემების შეცვლა; 2. DataSet ობიექტში ცხრილების შეცვლა; 3. DataSet ობიექტში მოთავსებული ცხრილების გადაგზავნა სერვერზე მონაცემთა ბაზაში

გ. 1. DataSet ობიექტში ცხრილების შეცვლა; 2. DataSet ობიექტში მოთავსებული ცხრილების გადაგზავნა სერვერზე მონაცემთა ბაზაში; 3. DataSet ობიექტის შევსება იმ ცხრილებით, რომლებშიც გვინდა მონაცემების შეცვლა

17.16.2. ადაპტერის შექმნის შემდეგ უნდა შევქმნათ:

ა. არაკორექტული SQL ბრძანებები: INSERT, UPDATE და DELETE, ცხრილებში მონაცემების შეცვლის მიზნით

ბ. კორექტული SQL ბრძანებები: INSERT, UPDATE და DELETE, ცხრილებში მონაცემების შეცვლის მიზნით

გ. კორექტული SQL ბრძანებები: UPDATE და DELETE, ცხრილებში მონაცემების შეცვლის მიზნით

17.16.3. ადაპტერის შექმნის შემდეგ უნდა შევქმნათ კორექტული SQL ბრძანებები: INSERT, UPDATE და DELETE, ცხრილებში მონაცემების შეცვლის მიზნით. ამას ავტომატურად აკეთებს:

ა. SqlDataAdapter კლასი

ბ. SqlCommandBuilder კლასი

გ. SqlConnection კლასი

17.16.4. SqlCommandBuilder კლასის კონსტრუქტორს არგუმენტად გადაეცემა:

ა. SqlConnection კლასის ობიექტი

ბ. SqlDataAdapter კლასის ობიექტი

გ. Dataset კლასის ობიექტი

17.16.5. SqlCommandBuilder კლასის ობიექტის შექმნისას:

ა. არ ხდება SQL ბრძანებების გენერირება და ასოცირება SqlDataAdapter ობიექტთან კონსტრუქტორის მიერ

ბ. ხდება SQL ბრძანებების გენერირება და ასოცირება SqlDataAdapter ობიექტთან კონსტრუქტორის მიერ

გ. ხდება SQL ბრძანებების მხოლოდ გენერირება

17.16.6. myAdapter.Update(myDataset, "Personali"); კოდში:

ა. myAdapter ობიექტის Update() მეთოდი არ ასრულებს myDataset ობიექტიდან Personali ცხრილის გადაგზავნას სერვერზე Shekveta მონაცემთა ბაზაში

ბ. myAdapter ობიექტის Update() მეთოდი ასრულებს myDataset ობიექტიდან Personali ცხრილის გადაგზავნას სერვერზე Shekveta მონაცემთა ბაზაში

გ. myAdapter ობიექტის Update() მეთოდი ასრულებს myDataset ობიექტის Personali ცხრილში სერვერზე მოთავსებული Shekveta მონაცემთა ბაზიდან Personali ცხრილის გადმოწერას

17.16.7. Update() მეთოდი:

ა. არ ამოწმებს Personali ცხრილის თითოეულ სტრიქონს, შეიცვალა თუ არა ის

ბ. ამოწმებს Personali ცხრილის თითოეულ სტრიქონს, შეიცვალა თუ არა ის

გ. ამოწმებს Personali ცხრილის მხოლოდ პირველ სტრიქონს, შეიცვალა თუ არა ის

17.16.8. Rows კოლექციის თითოეულ სტრიქონს აქვს RowState თვისება, რომელიც განსაზღვრავს:

ა. სტრიქონი იყო თუ არა წაშლილი ან დამატებული

ბ. სტრიქონი იყო თუ არა წაშლილი, დამატებული, შეცვლილი თუ დარჩა უცვლელი

გ. სტრიქონი იყო თუ არა წაშლილი ან შეცვლილი

17.16.9. იმისათვის, რომ ცხრილის რომელიმე სტრიქონის ველს მივანიჭოთ საჭირო მნიშვნელობა:

ა. არ უნდა მივუთითოთ სტრიქონის ინდექსი (ნომერი) და სვეტის სახელი ან ინდექსი

ბ. უნდა მივუთითოთ სტრიქონის ინდექსი (ნომერი) და სვეტის სახელი ან ინდექსი

გ. უნდა მივუთითოთ სტრიქონის მხოლოდ ინდექსი

17.16.10. რომელი კოდია სწორი:

ა. myDataset.Tables["Personali"].Rows[3]["asaki"] = Convert.ToInt32;

ბ. myDataset.Tables["Personali"].Rows[3]["asaki"] = Convert.ToInt32(textBox1.Text);

გ. myDataset.Tables["Personali"].Rows[3]["asaki"] = Update();

17.16.11. myDataset.Tables["Personali"].Rows[3]["asaki"] = Convert.ToInt32(textBox1.Text); კოდის შესრულების შედეგად:

ა. არ შეიცვლება ჩვენს ლოკალურ კომპიუტერში მოთავსებული myDataset ობიექტის Personali ცხრილი და არა სერვერზე მოთავსებული Shekveta მონაცემთა ბაზის Personali ცხრილი

ბ. შეიცვლება ჩვენს ლოკალურ კომპიუტერში მოთავსებული myDataset ობიექტის Personali ცხრილი და არა სერვერზე მოთავსებული Shekveta მონაცემთა ბაზის Personali ცხრილი

გ. შეიცვლება ჩვენს ლოკალურ კომპიუტერში მოთავსებული myDataset ობიექტის Personali ცხრილი და სერვერზე მოთავსებული Shekveta მონაცემთა ბაზის Personali ცხრილი

SELECT, UPDATE, INSERT და DELETE ბრძანებების ნახვა

17.17.1. CommandBuilder ობიექტი, ახდენს SQL ბრძანებების გენერირებას, რომლებიც:

ა. არ გამოიყენება მონაცემების მოდიფიცირებისთვის

ბ. გამოიყენება მონაცემების მოდიფიცირებისთვის

გ. გამოიყენება მონაცემების წაკითხვისთვის

17.17.2. CommandBuilder ობიექტი, ახდენს SQL ბრძანებების გენერირებას, რომლებიც გამოიყენება მონაცემების მოდიფიცირებისთვის. ეს ბრძანებებია:

ა. UPDATE და DELETE
ბ. INSERT, UPDATE და DELETE
გ. INSERT და DELETE

17.17.3. CommandBuilder ობიექტის მიერ INSERT, UPDATE და DELETE ბრძანებების გენერირება:

ა. არ ხდება ადაპტერში მითითებული SELECT ბრძანების საფუძველზე
ბ. ხდება ადაპტერში მითითებული SELECT ბრძანების საფუძველზე
გ. ხდება ადაპტერში მითითებული UPDATE ბრძანების საფუძველზე

17.17.4. გენერირებული ბრძანებების მისაღებად და სანახავად შეგვიძლია შესაბამისად გამოვიყენოთ CommandBuilder ობიექტის:

ა. GetUpdateCommand() და GetDeleteCommand() მეთოდები
ბ. GetInsertCommand(), GetUpdateCommand() და GetDeleteCommand() მეთოდები
გ. GetInsertCommand() და GetDeleteCommand() მეთოდები

17.17.5. GetInsertCommand() მეთოდი გასცემს გენერირებულ:

ა. INSERT ბრძანებას
ბ. UPDATE ბრძანებას
გ. DELETE ბრძანებას

17.17.6. GetUpdateCommand()მეთოდი გასცემს გენერირებულ:

ა. DELETE ბრძანებას
ბ. UPDATE ბრძანებას
გ. INSERT ბრძანებას

17.17.7. GetDeleteCommand() მეთოდი გასცემს გენერირებულ:

ა. INSERT ბრძანებას
ბ. DELETE ბრძანებას
გ. UPDATE ბრძანებას

17.17.8. { SqlCommand updateCommand = myBuilder.GetUpdateCommand();
label1.Text += updateCommand.CommandText + "\n"; } კოდის შესრულების შედეგად გაიცემა:

ა. გენერირებული UPDATE ბრძანება
ბ. გენერირებული DELETE ბრძანება
გ. გენერირებული INSERT ბრძანება

17.17.9. { SqlCommand insertCommand = myBuilder.GetInsertCommand();
label1.Text += insertCommand.CommandText + "\n"; } კოდის შესრულების შედეგად გაიცემა:

ა. გენერირებული UPDATE ბრძანება
ბ. გენერირებული DELETE ბრძანება
გ. გენერირებული INSERT ბრძანება

17.17.10. { SqlCommand deleteCommand = myBuilder.GetDeleteCommand();
label1.Text += deleteCommand.CommandText + "\n"; } კოდის შესრულების შედეგად გაიცემა:

ა. გენერირებული UPDATE ბრძანება
ბ. გენერირებული DELETE ბრძანება
გ. გენერირებული INSERT ბრძანება

მონაცემთა ბაზაში სტრიქონების დამატება

17.18.1. ცხრილში ახალი სტრიქონების დასამატებლად შემდეგი მოქმედებები უნდა შევასრულოთ:

ა. 1. ახალი DataRow ობიექტის შექმნა; 2. მონაცემებით მისი შევსება; 3. შევსებული სტრიქონის დამატება DataSet ობიექტის Rows კოლექციისთვის; 4. ადაპტერის Update() მეთოდის შესრულება
ბ. 3. ადაპტერის Update() მეთოდის შესრულება; 2. მონაცემებით მისი შევსება; 3. შევსებული სტრიქონის დამატება DataSet ობიექტის Rows კოლექციისთვის; 4. ახალი DataRow ობიექტის შექმნა;
გ. 1. შევსებული სტრიქონის დამატება DataSet ობიექტის Rows კოლექციისთვის; 2. ახალი DataRow ობიექტის შექმნა; 3. მონაცემებით მისი შევსება; 4. ადაპტერის Update() მეთოდის შესრულება

17.18.2. Rows კოლექციას აქვს Count თვისება, რომელშიც:

ა. ავტომატურად იწერება შესაბამის ცხრილში არსებული სტრიქონების რაოდენობა
ბ. ავტომატურად არ იწერება შესაბამის ცხრილში არსებული სტრიქონების რაოდენობა
გ. ავტომატურად იწერება შესაბამის ცხრილში არსებული სვეტების რაოდენობა

17.18.3. label1.Text = myDataset.Tables["Personali"].Rows.Count.ToString() + "\n"; კოდის შესრულების შედეგად გაიცემა:

ა. Personali ცხრილის სტრიქონების რაოდენობა
ბ. Personali ცხრილის სვეტების რაოდენობა
გ. myDataset ცხრილის სტრიქონების რაოდენობა

17.18.4. ცხრილში ახალი სტრიქონის დამატების შემდეგ, შესაბამისი ცხრილის Rows კოლექციის Count თვისების მნიშვნელობა:

ა. ერთით გაიზრდება
ბ. ერთით შემცირდება
გ. განულდება

17.18.5. DataRow myRow = myDataset.Tables["Personali"].NewRow(); კოდში :

ა. იქმნება DataRow ტიპის myRow ობიექტი myDataset ობიექტის NewRow() მეთოდის გამოყენებით
ბ. იქმნება myDataset ტიპის myRow ობიექტი myDataset ობიექტის NewRow() მეთოდის გამოყენებით
გ. იქმნება DataRow ტიპის myDataset ობიექტი myDataset ობიექტის NewRow() მეთოდის გამოყენებით

17.18.6. DataRow myRow = myDataset.Tables["Personali"].NewRow(); კოდის შესრულების შედეგად მიღებული myRow ობიექტის სტრუქტურა:

ა. ემთხვევა Personali ცხრილის სტრუქტურას

ბ. არ ემთხვევა Personali ცხრილის სტრუქტურას
გ. ემთხვევა myDataset ცხრილის სტრუქტურას

17.18.7. DataRow myRow = myDataset.Tables["Personali"].NewRow(); კოდის შესრულების შედეგად მიღებული myRow ობიექტი:
ა. შეიცავს Personali ცხრილის მხოლოდ იმ სვეტებს, რომლებიც ადაპტერში განსაზღვრულ SELECT ბრძანებაშია მოთავსებული
ბ. არ შეიცავს Personali ცხრილის მხოლოდ იმ სვეტებს, რომლებიც ადაპტერში განსაზღვრულ SELECT ბრძანებაშია მოთავსებული
გ. შეიცავს Personali ცხრილის მხოლოდ იმ სტრიქონებს, რომლებიც ადაპტერში განსაზღვრულ SELECT ბრძანებაშია მოთავსებული

17.18.8. DataRow myRow = myDataset.Tables["Personali"].NewRow(); კოდის შესრულების შედეგად მიღებული myRow ობიექტში:
ა. სვეტები ცარიელია და მათ უნდა მივანიჭოთ კონკრეტული მნიშვნელობები
ბ. სვეტები არაა ცარიელი და მათ არ უნდა მივანიჭოთ კონკრეტული მნიშვნელობები
გ. სვეტები ცარიელია და მათ არ უნდა მივანიჭოთ კონკრეტული მნიშვნელობები

17.18.9. როცა ყველა სვეტს მივანიჭებთ საჭირო მნიშვნელობებს, მხოლოდ ამის შემდეგ უნდა დავუმატოთ myRow ობიექტი Personali ცხრილს Rows კოლექციის:
ა. Add() მეთოდის გამოყენებით
ბ. Close() მეთოდის გამოყენებით
გ. Remove() მეთოდის გამოყენებით

17.18.10. myDataset.Tables["Personali"].Rows.Add(myRow); კოდის შესრულების შედეგად:
ა. myRow ობიექტი დაემატება Personali ცხრილის Rows კოლექციას
ბ. myRow ობიექტი წაიშლება Personali ცხრილის Rows კოლექციიდან
გ. myRow ობიექტი დაემატება Personali ცხრილის myDataset კოლექციას

17.18.11. როცა ყველა სვეტს მივანიჭებთ საჭირო მნიშვნელობებს და შევასრულებთ Rows კოლექციის Add() მეთოდს, მხოლოდ:
ა. ამის შემდეგ უნდა შევასრულოთ ადაპტერის Update() მეთოდი ცვლილებების შესანახად სერვერზე მოთავსებულ მონაცემთა ბაზაში
ბ. ამის შემდეგ უნდა შევასრულოთ Dataset ობიექტის Update() მეთოდი ცვლილებების შესანახად სერვერზე მოთავსებულ მონაცემთა ბაზაში
გ. ამის შემდეგ უნდა შევასრულოთ ადაპტერის Fill() მეთოდი ცვლილებების შესანახად სერვერზე მოთავსებულ მონაცემთა ბაზაში

17.18.12. იმისათვის, რომ რომელიმე სვეტს მივანიჭოთ მნიშვნელობები, საჭიროა ეს სვეტი მითითებული:
ა. იყოს SELECT მოთხოვნაში, რომელიც მიეთითება ადაპტერის შექმნისას
ბ. არ იყოს SELECT მოთხოვნაში, რომელიც მიეთითება ადაპტერის შექმნისას
გ. იყოს INSERT მოთხოვნაში, რომელიც მიეთითება ადაპტერის შექმნისას

17.18.13. იმისათვის, რომ რომელიმე სვეტს მივანიჭოთ მნიშვნელობები, საჭიროა ეს სვეტი მითითებული იყოს SELECT მოთხოვნაში, რომელიც მიეთითება ადაპტერის შექმნისას. თუ სვეტი მითითებული არ არის, მაშინ მისი მნიშვნელობა იქნება:
ა. null

ბ. 0
გ. 1

სტრიქონების წაშლა

17.19.1. ცხრილიდან სტრიქონის წასაშლელად გამოიყენება DataRow ობიექტის:
ა. Delete() მეთოდი
ბ. Add() მეთოდი
გ. Insert() მეთოდი

17.19.2. ცხრილიდან სტრიქონის წასაშლელად Delete() მეთოდის შემდეგ:
ა. აუცილებელია Update() მეთოდის შესრულება
ბ. არაა აუცილებელი Update() მეთოდის შესრულება
გ. აუცილებელია Insert() მეთოდის შესრულება

17.19.3. საქმე ის არის, რომ Delete() მეთოდი:
ა. არ შლის სტრიქონს, არამედ მონიშნავს მას შემდგომი წაშლისთვის
ბ. შლის სტრიქონს, არამედ მონიშნავს მას შემდგომი წაშლისთვის
გ. არ შლის სტრიქონს, არამედ არ მონიშნავს მას შემდგომი წაშლისთვის

17.19.4. Update() მეთოდი ახდენს ცვლილებების რეალურად ფიქსირებას, ე.ი. :
ა. წასაშლელად მონიშნული სტრიქონის წაშლას
ბ. წასაშლელად მოუნიშნავი სტრიქონის წაშლას
გ. წასაშლელად მონიშნული სტრიქონის ჩამატებას

17.19.5. DataRow findRow = myDataset.Tables["Personali"].Rows.Find(find_key);

```
if ( findRow != null )
{
        findRow.Delete();
        myAdapter.Update(myDataset, "Personali");
}
```

კოდის შესრულების შედეგად:
ა. წაიშლება ნაპოვნი სტრიქონი
ბ. არ წაიშლება ნაპოვნი სტრიქონი
გ. წაიშლება ყველა სტრიქონი

ორ ცხრილს შორის ბმის შექმნა

17.20.1. ორ ცხრილს შორის ბმის შესაქმნელად გამოიყენება:
ა. DataRelation ობიექტი
ბ. DataSet ობიექტი
გ. DataTable ობიექტი

17.20.2. DataRelation ობიექტი კავშირს:
ა. ამყარებს DataTable ობიექტებს შორის
ბ. არ ამყარებს DataTable ობიექტებს შორის
გ. ამყარებს DataSet ობიექტებს შორის

17.20.3. მთავარია Personali ცხრილი. მისი პირველადი გასაღებია personaliID სვეტი. Shemkveti ცხრილი დამოკიდებულია. მისი გარე გასაღებია personaliID სვეტი. ამ ცხრილებს შორის კავშირის დასამყარებლად:

ა. Personali ცხრილის პირველადი გასაღები უნდა დავუკავშიროთ Shemkveti ცხრილის გარე გასაღებს

ბ. Shemkveti ცხრილის პირველადი გასაღები უნდა დავუკავშიროთ Personali ცხრილის გარე გასაღებს

გ. Personali ცხრილის გარე გასაღები უნდა დავუკავშიროთ Shemkveti ცხრილის პირველადი გასაღებს

17.20.4. DataRelation ობიექტის შესაქმნელად გამოიყენება Relations ობიექტის:

ა. Add() მეთოდი

ბ. Remove() მეთოდი

გ. Insert() მეთოდი

17.20.5. Add() მეთოდის:

ა. პირველი პარამეტრი ბმის სახელია, მეორე - პირველადი გასაღები, მესამე კი - გარე გასაღები

ბ. პირველი პარამეტრია გარე გასაღები, მეორე - პირველადი გასაღები, მესამე კი - ბმის სახელი

გ. პირველი პარამეტრია პირველადი გასაღები, მეორე - გარე გასაღები, მესამე კი - გარე გასაღები

17.20.6. DataRelation personaliShemkvetiRelation = myDataset.Relations.Add("PersonaliShemkveti", myDataset.Tables["Personali"].Columns["PersonaliID"], myDataset.Tables["Shemkveti"].Columns["PersonaliID"]); კოდის შესრულების შედეგად:

ა. Personali ცხრილის PersonaliID სვეტი დაუკავშირდება Shemkveti ცხრილის PersonaliID სვეტს. ბმის სახელია PersonaliShemkveti

ბ. Shemkveti ცხრილის Personali სვეტი დაუკავშირდება Personali ცხრილის Shemkveti სვეტს. ბმის სახელია PersonaliShemkveti

გ. Personali ცხრილის PersonaliID სვეტი დაუკავშირდება Shemkveti ცხრილის PersonaliID სვეტს. ბმის სახელია myDataset.Tables["Personali"]

17.20.7. DataRelation personaliShemkvetiRelation = myDataset.Relations.Add("PersonaliShemkveti", myDataset.Tables["Personali"].Columns["PersonaliID"], myDataset.Tables["Shemkveti"].Columns["PersonaliID"]); კოდის შესრულების შედეგად ბმის სახელი იქნება:

ა. PersonaliShemkveti

ბ. DataRelation

გ. myDataset

17.20.8. DataRelation personaliShemkvetiRelation = myDataset.Relations.Add("PersonaliShemkveti", myDataset.Tables["Personali"].Columns["PersonaliID"], myDataset.Tables["Shemkveti"].Columns["PersonaliID"]); კოდის შესრულების შედეგად:

ა. Personali ცხრილის PersonaliID პირველადი გასაღები დაუკავშირდება Shemkveti ცხრილის PersonaliID გარე გასაღებს

ბ. Shemkveti ცხრილის PersonaliID პირველადი გასაღები დაუკავშირდება Personali ცხრილის PersonaliID გარე გასაღებს
გ. Personali ცხრილის PersonaliID პირველადი გასაღები დაუკავშირდება Shemkveti ცხრილის PersonaliID პირველად გასაღებს

17.20.9. DataRelation personaliShemkvetiRelation = myDataset.Relations.Add("PersonaliShemkveti", myDataset.Tables["Personali"].Columns["PersonaliID"], myDataset.Tables["Shemkveti"].Columns["PersonaliID"]); კოდში:
ა. პირველადი გასაღებია Personali ცხრილის PersonaliID სვეტი, გარე გასაღებია Shemkveti ცხრილის PersonaliID სვეტი
ბ. გარე გასაღებია Personali ცხრილის PersonaliID სვეტი, პირველადი გასაღებია Shemkveti ცხრილის PersonaliID სვეტი
გ. პირველადი გასაღებია Personali ცხრილის PersonaliID სვეტი, პირველად გასაღებია Shemkveti ცხრილის PersonaliID სვეტი

17.20.10. ორ ცხრილს შორის კავშირის დამყარების წინ:
ა. საჭიროა myDataset ობიექტის შევსება ორივე ცხრილით
ბ. არაა საჭირო myDataset ობიექტის შევსება ორივე ცხრილით
გ. საჭიროა myDataset ობიექტის შევსება ერთ-ერთი ცხრილით

17.20.11. DataRelation personaliShemkvetiRelation = myDataset.Relations.Add("PersonaliShemkveti", myDataset.Tables["Personali"].Columns["PersonaliID"], myDataset.Tables["Shemkveti"].Columns["PersonaliID"]); კოდში:
ა. Personali და Shemkveti ცხრილებს შორის კავშირის დასამყარებლად იქმნება personaliShemkvetiRelation ობიექტი
ბ. Personali და Shemkveti ცხრილებს შორის კავშირის გასაწყვეტად იქმნება personaliShemkvetiRelation ობიექტი
გ. Personali და Shemkveti ცხრილებს შორის კავშირის დასამყარებლად იქმნება PersonaliShemkveti ობიექტი

17.20.12. იმისათვის, რომ Personali ცხრილის თითოეული სტრიქონისთვის გამოვიტანოთ შესაბამისი ბმული სტრიქონები Shemkveti ცხრილიდან, უნდა გამოვიყენოთ DataRow ობიექტის:
ა. GetChildRows() მეთოდი, რომელსაც პარამეტრად უნდა გადავცეთ DataRelation ობიექტი
ბ. SetChildRows() მეთოდი, რომელსაც პარამეტრად უნდა გადავცეთ DataRelation ობიექტი
გ. GetChildRows() მეთოდი, რომელსაც პარამეტრად უნდა გადავცეთ Dataset ობიექტი

SQL ბრძანებების უშუალო შესრულება

ერთი მნიშვნელობის მიღება

17.21.1. თუ საჭიროა სტრიქონების წაშლა ან სტრიქონებში სვეტების მნიშვნელობების შეცვლა, მაშინ დიდი ცხრილებისთვის:
ა. გაცილებით ეფექტური იქნება ამის გაკეთება ერთი SQL-ბრძანებით
ბ. ნაკლებად ეფექტური იქნება ამის გაკეთება ერთი SQL-ბრძანებით
გ. გაცილებით ეფექტური იქნება ამის გაკეთება ორი SQL-ბრძანებით

17.21.2. როცა საჭიროა SQL მოთხოვნისაგან ერთი შედეგის მიღება, მაშინ უმჯობესია SlqCommand ობიექტის:

ა. ExecuteScalar() მეთოდის გამოყენება
ბ. EndExecuteReader()მეთოდის გამოყენება
გ. BeginExecuteReader()მეთოდის გამოყენება

17.21.3. ExecuteScalar() გასცემს:

*ა. სკალარულ მნიშვნელობას
ბ. ვექტორულ მნიშვნელობას
გ. არასკალარულ მნიშვნელობას

17.21.4. { ... myCommand.CommandText = "SELECT COUNT(*) FROM Personali"; Object countResult = myCommand.ExecuteScalar(); ... } კოდის შესრულების შედეგად:

ა. countResult ცვლადს მიენიჭება Personali ცხრილის სტრიქონების რაოდენობა
ბ. countResult ცვლადს მიენიჭება Personali ცხრილის სტრიქონების ჯამი
გ. countResult ცვლადს მიენიჭება Personali ცხრილის სტრიქონების სხვობა

17.21.5. { ... myCommand.CommandText = "SELECT AVG(asaki) FROM Personali WHERE ganyofileba = N'სამედიცინო'"; Object countResult = myCommand.ExecuteScalar(); ... } კოდის შესრულების შედეგად:

ა. countResult ცვლადს მიენიჭება იმ თანამშრომლების საშუალო ასაკი, რომლებიც სამედიცინო განყოფილებაში მუშაობენ
ბ. countResult ცვლადს მიენიჭება იმ თანამშრომლების მაქსიმალური ასაკი, რომლებიც სამედიცინო განყოფილებაში მუშაობენ
გ. countResult ცვლადს მიენიჭება იმ თანამშრომლების საშუალო ასაკი, რომლებიც სამედიცინო განყოფილებაში არ მუშაობენ

UPDATE მოთხოვნის უშუალოდ შესრულება

17.22.1. INSERT, UPDATE და DELETE ბრძანებების შესასრულებლად გამოიყენება SqlCommand ობიექტის:

ა. ExecuteNonQuery() მეთოდი
ბ. EndExecuteReader მეთოდი
გ. ExecuteScalar მეთოდი

17.22.2. ExecuteNonQuery() მეთოდი გასცემს:

ა. შეცვლილი სტრიქონების რაოდენობას
ბ. შეუცვლელი სტრიქონების რაოდენობას
გ. შეცვლილი სვეტების რაოდენობას

17.22.3. SqlCommand ობიექტის CommandText თვისებას ენიჭება შესასრულებელი:

ა. UPDATE ბრძანება
ბ. ExecuteNonQuery() მეთოდი
გ. SqlConnection ობიექტი

17.22.4. შესასრულებელი UPDATE ბრძანება ენიჭება SqlCommand ობიექტის:

ა. CommandText თვისებას

ბ. CommandType თვისებას
გ. Connection თვისებას

17.22.5. თუ UPDATE ბრძანებას დავუმატებთ WHERE განყოფილებას, მაშინ:
ა. შეგვეძლება რამდენიმე სტრიქონში მონაცემების შეცვლა
ბ. არ შეგვეძლება რამდენიმე სტრიქონში მონაცემების შეცვლა
გ. შეგვეძლება რამდენიმე სტრიქონში მონაცემების წაშლა

17.22.6. { ... myCommand.CommandText = "UPDATE Personali SET xelfasi = xelfasi * 2"; int rowsAffected = myCommand.ExecuteNonQuery(); ... } კოდის შესრულების შედეგად:
ა. გაორმაგდება Personali ცხრილის xelfasi სვეტის მნიშვნელობა
ბ. შემცირდება Personali ცხრილის xelfasi სვეტის მნიშვნელობა
გ. გაორმაგდება xelfasi ცხრილის Personali სვეტის მნიშვნელობა

17.22.7. სამედიცინო განყოფილების პერსონალისთვის ხელფასის გასაზრდელად 10%-ით UPDATE ბრძანება ასე უნდა შევიტანოთ:
ა. myCommand.CommandText =
"UPDATE Personali SET xelfasi = xelfasi * 1.1 WHERE ganyofileba = N'სამედიცინო'";
ბ. myCommand.CommandText =
"UPDATE Personali SET xelfasi = xelfasi * 2 WHERE ganyofileba = N'სამედიცინო'";
გ. myCommand.CommandText =
"UPDATE Personali SET xelfasi = xelfasi * 30 WHERE ganyofileba = N'სავაჭრო'";

17.22.8. UPDATE ბრძანებას პარამეტრები:
ა. შეიძლება გადავცეთ
ბ. არ შეიძლება გადავცეთ
გ. აუცილებლად უნდა გადავცეთ

17.22.9. UPDATE ბრძანებაში პარამეტრის წინ უნდა მოვათავსოთ სიმბოლო:
ა. @
ბ. #
გ. *

17.22.10. UPDATE ბრძანებაში პარამეტრის დასამატებლად გამოიყენება:
ა. SqlCommand ობიექტის Parameters თვისების Add() მეთოდი
ბ. Parameters ობიექტის SqlCommand თვისების Add() მეთოდი
გ. SqlCommand ობიექტის Parameters თვისების Insert() მეთოდი

17.22.11. Add() მეთოდის პირველი პარამეტრია:
ა. დასამატებელი პარამეტრის სახელი
ბ. დასამატებელი პარამეტრის ტიპი
გ. დასამატებელი პარამეტრის სიგრძე

17.22.12 Add() მეთოდის მეორე პარამეტრია:
ა. დასამატებელი პარამეტრის ტიპი
ბ. დასამატებელი პარამეტრის სახელი
გ. დასამატებელი პარამეტრის სიგრძე

17.22.13. Add() მეთოდის მესამე პარამეტრია:
ა. დასამატებელი პარამეტრის სიგრძე
ბ. დასამატებელი პარამეტრის ტიპი
გ. დასამატებელი პარამეტრის სახელი

17.22.14. mySqlCommand.Parameters.Add("@personaliID", SqlDbType.Int, 4); კოდის შესრულების შედეგად mySqlCommand:
ა. ობიექტს დაემატება @personaliID პარამეტრი
ბ. ობიექტიდან წაიშლება @personaliID პარამეტრი
გ. ობიექტს დაემატება SqlDbType.Int პარამეტრი

17.22.15. mySqlCommand.Parameters.Add("@personaliID", SqlDbType.Int, 4); კოდის შესრულების შედეგად mySqlCommand ობიექტს დაემატება @personaliID პარამეტრი, რომელიც არის:
ა. მთელი რიცხვი
ბ. სტრიქონი
გ. წილადი

17.22.16. mySqlCommand.Parameters.Add("@gvari", SqlDbType.NVarChar, 20); კოდის შესრულების შედეგად mySqlCommand ობიექტს დაემატება @gvari პარამეტრი, რომელიც არის:
ა. უნიკოდების 20 ბაიტიანი სტრიქონი
ბ. უნიკოდების 20 სიმბოლოიანი სტრიქონი
გ. წილადი

17.22.17. mySqlCommand.Parameters["@asaki"].Value = Convert.ToInt32(textBox5.Text); კოდის შესრულების შედეგად @asaki პარამეტრს მიენიჭება:
ა. მთელი რიცხვი
ბ. წილადი
გ. სტრიქონი

17.22.18. mySqlCommand.Parameters["@gvari"].Value = textBox1.Text; კოდის შესრულების შედეგად @gvari პარამეტრს მიენიჭება:
ა. სტრიქონი
ბ. წილადი
გ. თარიღი

17.22.19. UPDATE ბრძანებით:
ა. შეიძლება რამდენიმე სტრიქონის შეცვლა
ბ. არ შეიძლება რამდენიმე სტრიქონის შეცვლა
გ. შეიძლება მხოლოდ ერთი სტრიქონის შეცვლა

17.22.20. mySqlCommand.CommandText =
"UPDATE Personali SET xelfasi = xelfasi + @xelfasi WHERE (ganyofileba = @ganyofileba)"; კოდის შესრულების შედეგად ხელფასი გაეზრდება:
ა. მითითებული განყოფილების თანამშრომლებს

ბ. მითითებული ასაკის თანამშრომლებს

გ. მითითებული სტაჟის მქონე თანამშრომლებს

INSERT მოთხოვნის უშუალოდ შესრულება

17.23.1. mySqlCommand.CommandText =
"INSERT INTO Personali (gvari, ganyofileba, qalaqi, xelfasi, asaki, staji, tarigi_dabadebis) " +
"VALUES (@gvari, @ganyofileba, @qalaqi, @xelfasi, @asaki, @staji, @tarigi_dabadebis)"; კოდის შესრულების შედეგად:

ა. Personali ცხრილს ჩაემატება ახალი სტრიქონი, რომლის gvari სვეტს მიენიჭება @gvari პარამეტრის მნიშვნელობა

ბ. Personali ცხრილს ჩაემატება ახალი სტრიქონი, რომლის gvari სვეტს მიენიჭება @ganyofileba პარამეტრის მნიშვნელობა

გ. Personali ცხრილს ჩაემატება ახალი სტრიქონი, რომლის @gvari სვეტს მიენიჭება gvari პარამეტრის მნიშვნელობა

17.23.2. mySqlCommand.CommandText =
"INSERT INTO Personali (gvari, ganyofileba, qalaqi, xelfasi, asaki, staji, tarigi_dabadebis) " +
"VALUES (@gvari, @ganyofileba, @qalaqi, @xelfasi, @asaki, @staji, @tarigi_dabadebis)"; კოდში არ არის მითითებული პირველადი გასაღები, რადგან მისთვის:

ა. Identity თვისება დაყენებულია Yes მდგომარეობაში

ბ. Identity თვისება დაყენებულია No მდგომარეობაში

გ. Identity თვისება დაყენებულია Null მდგომარეობაში

17.23.3. mySqlCommand.CommandText =
"INSERT INTO Personali (gvari, ganyofileba, qalaqi, xelfasi, asaki, staji, tarigi_dabadebis) " +
"VALUES (@gvari, @ganyofileba, @qalaqi, @xelfasi, @asaki, @staji, @tarigi_dabadebis)"; კოდში არ არის მითითებული პირველადი გასაღები, რადგან მისთვის Identity თვისება დაყენებულია Yes მდგომარეობაში, ამიტომ:

ა. სტრიქონის დამატების დროს ავტომატურად ხდება პირველადი გასაღებისთვის სწორი მნიშვნელობის გენერირება

ბ. სტრიქონის დამატების დროს ავტომატურად არ ხდება პირველადი გასაღებისთვის სწორი მნიშვნელობის გენერირება

გ. სტრიქონის დამატების დროს ავტომატურად ხდება გარე გასაღებისთვის სწორი მნიშვნელობის გენერირება

17.23.4. mySqlCommand.CommandText =
"INSERT INTO Personali (gvari, ganyofileba, qalaqi, xelfasi, asaki, staji, tarigi_dabadebis) " +
"VALUES (@gvari, @ganyofileba, @qalaqi, @xelfasi, @asaki, @staji, @tarigi_dabadebis)"; კოდში არ არის მითითებული პირველადი გასაღები. მისთვის Identity თვისება დაყენებულია No მდგომარეობაში, ამიტომ:

ა. სტრიქონის დამატების დროს პირველადი გასაღებისთვის სწორი მნიშვნელობის გენერირება ჩვენ უნდა უზრუნველვყოთ

ბ. სტრიქონის დამატების დროს ავტომატურად ხდება პირველადი გასაღებისთვის სწორი მნიშვნელობის გენერირება

გ. სტრიქონის დამატების დროს გარე გასაღებისთვის სწორი მნიშვნელობის გენერირება ჩვენ უნდა უზრუნველვყოთ

17.23.5. SqlCommand ობიექტის CommandText თვისებას ენიჭება შესასრულებელი:
ა. INSERT ბრძანება
ბ. ExecuteNonQuery() მეთოდი
გ. SqlConnection ობიექტი

17.23.6. შესასრულებელი INSERT ბრძანება ენიჭება SqlCommand ობიექტის:
ა. CommandText თვისებას
ბ. CommandType თვისებას
გ. Connection თვისებას

17.23.7. INSERT ბრძანებას პარამეტრები:
ა. შეიძლება გადავცეთ
ბ. არ შეიძლება გადავცეთ
გ. აუცილებლად უნდა გადავცეთ

17.23.8. INSERT ბრძანებაში პარამეტრის წინ უნდა მოვათავსოთ სიმბოლო:
ა. @
ბ. #
გ. *

17.23.9. INSERT ბრძანებაში პარამეტრის დასამატებლად გამოიყენება:
ა. SqlCommand ობიექტის Parameters თვისების Add() მეთოდი
ბ. Parameters ობიექტის SqlCommand თვისების Add() მეთოდი
გ. SqlCommand ობიექტის Parameters თვისების Insert() მეთოდი

17.23.10. Add() მეთოდის პირველი პარამეტრია:
ა. დასამატებელი პარამეტრის სახელი
ბ. დასამატებელი პარამეტრის ტიპი
გ. დასამატებელი პარამეტრის სიგრძე

17.23.11. Add() მეთოდის მეორე პარამეტრია:
ა. დასამატებელი პარამეტრის ტიპი
ბ. დასამატებელი პარამეტრის სახელი
გ. დასამატებელი პარამეტრის სიგრძე

17.23.12. Add() მეთოდის მესამე პარამეტრია:
ა. დასამატებელი პარამეტრის სიგრძე
ბ. დასამატებელი პარამეტრის ტიპი
გ. დასამატებელი პარამეტრის სახელი

17.23.13. mySqlCommand.Parameters.Add("@staji", SqlDbType.Int, 4); კოდის შესრულების შედეგად mySqlCommand:
ა. ობიექტს დაემატება @staji პარამეტრი
ბ. ობიექტიდან წაიშლება @staji პარამეტრი
გ. ობიექტს დაემატება SqlDbType.Int პარამეტრი

17.23.14. mySqlCommand.Parameters.Add("@staji", SqlDbType.Int, 4); კოდის შესრულების შედეგად mySqlCommand ობიექტს დაემატება @staji პარამეტრი, რომელიც არის:

ა. მთელი რიცხვი
ბ. სტრიქონი
გ. წილადი

17.23.15. mySqlCommand.Parameters.Add("@gvari", SqlDbType.NVarChar, 25); კოდის შესრულების შედეგად mySqlCommand ობიექტს დაემატება @gvari პარამეტრი, რომელიც არის:

ა. უნიკოდების 25 ბაიტიანი სტრიქონი
ბ. უნიკოდების 25 სიმბოლოიანი სტრიქონი
გ. წილადი

17.23.16. mySqlCommand.Parameters["@staji"].Value = Convert.ToInt32(textBox5.Text); კოდის შესრულების შედეგად @staji პარამეტრს მიენიჭება:

ა. მთელი რიცხვი
ბ. წილადი
გ. სტრიქონი

17.23.17. mySqlCommand.Parameters["@gvari"].Value = textBox1.Text; კოდის შესრულების შედეგად @gvari პარამეტრს მიენიჭება:

ა. სტრიქონი
ბ. წილადი
გ. თარიღი

17.23.18. INSERT ბრძანებით:

ა. შეიძლება რამდენიმე სტრიქონის ჩამატება
ბ. არ შეიძლება მხოლოდ ერთი სტრიქონის ჩამატება
გ. შეიძლება მხოლოდ ერთი სტრიქონის ჩამატება

DELETE მოთხოვნის უშუალოდ შესრულება

17.24.1. SqlCommand ობიექტის CommandText თვისებას ენიჭება შესასრულებელი:

ა. DELETE ბრძანება
ბ. ExecuteNonQuery() მეთოდი
გ. SqlConnection ობიექტი

17.24.2. შესასრულებელი DELETE ბრძანება ენიჭება SqlCommand ობიექტის:

ა. CommandText თვისებას
ბ. CommandType თვისებას
გ. Connection თვისებას

17.24.3. DELETE ბრძანებას პარამეტრები:

ა. შეიძლება გადავცეთ
ბ. არ შეიძლება გადავცეთ
გ. აუცილებლად უნდა გადავცეთ

17.24.4. DELETE ბრძანებაში პარამეტრის წინ უნდა მოვათავსოთ სიმბოლო:

ა. @

ბ. #
გ. *

17.24.5. DELETE ბრძანებაში პარამეტრის დასამატებლად გამოიყენება:
ა. SqlCommand ობიექტის Parameters თვისების Add() მეთოდი
ბ. Parameters ობიექტის SqlCommand თვისების Add() მეთოდი
გ. SqlCommand ობიექტის Parameters თვისების Insert() მეთოდი

17.24.6. Add() მეთოდის პირველი პარამეტრია:
ა. დასამატებელი პარამეტრის სახელი
ბ. დასამატებელი პარამეტრის ტიპი
გ. დასამატებელი პარამეტრის სიგრძე

17.24.7. Add() მეთოდის მეორე პარამეტრია:
ა. დასამატებელი პარამეტრის ტიპი
ბ. დასამატებელი პარამეტრის სახელი
გ. დასამატებელი პარამეტრის სიგრძე

17.24.8. Add() მეთოდის მესამე პარამეტრია:
ა. დასამატებელი პარამეტრის სიგრძე
ბ. დასამატებელი პარამეტრის ტიპი
გ. დასამატებელი პარამეტრის სახელი

17.24.9. mySqlCommand.Parameters.Add("@asaki", SqlDbType.Int, 4); კოდის შესრულების შედეგად mySqlCommand:
ა. ობიექტს დაემატება @asaki პარამეტრი
ბ. ობიექტიდან წაიშლება @asaki პარამეტრი
გ. ობიექტს დაემატება SqlDbType.Int პარამეტრი

17.24.10. mySqlCommand.Parameters.Add("@asaki", SqlDbType.Int, 4); კოდის შესრულების შედეგად mySqlCommand ობიექტს დაემატება @asaki პარამეტრი, რომელიც არის:
ა. მთელი რიცხვი
ბ. სტრიქონი
გ. წილადი

17.24.11. mySqlCommand.Parameters.Add("@gvari", SqlDbType.NVarChar, 30); კოდის შესრულების შედეგად mySqlCommand ობიექტს დაემატება @gvari პარამეტრი, რომელიც არის:
ა. უნიკოდების 30 ბაიტიანი სტრიქონი
ბ. უნიკოდების 30 სიმბოლოიანი სტრიქონი
გ. წილადი

17.24.12. mySqlCommand.Parameters["@staji"].Value = Convert.ToInt32(textBox5.Text); კოდის შესრულების შედეგად @staji პარამეტრს მიენიჭება:
ა. მთელი რიცხვი
ბ. წილადი
გ. სტრიქონი

17.24.13. mySqlCommand.Parameters["@ganyofileba"].Value = textBox1.Text; კოდის შესრულების შედეგად @ganyofileba პარამეტრს მიენიჭება:

ა. სტრიქონი

ბ. მთელი რიცხვი

გ. თარიღი

17.24.14. DELETE ბრძანებით:

ა. შეიძლება რამდენიმე სტრიქონის წაშლა

ბ. არ შეიძლება რამდენიმე სტრიქონის წაშლა

გ. შეიძლება მხოლოდ ერთი სტრიქონის წაშლა

17.24.15. თუ DELETE ბრძანებას დავუმატებთ WHERE განყოფილებას, მაშინ:

ა. შეგვეძლება რამდენიმე სტრიქონის წაშლა

ბ. არ შეგვეძლება რამდენიმე სტრიქონის წაშლა

გ. შეგვეძლება რამდენიმე სტრიქონის ჩამატება

17.24.16. Personali ცხრილიდან სამედიცინო განყოფილების პერსონალის წასაშლელად DELETE ბრძანება ასე უნდა შევიტანოთ:

ა. myCommand.CommandText =
"DELETE FROM Personali WHERE ganyofileba = N'სამედიცინო'";

ბ. myCommand.CommandText =
"DELETE FROM Personali WHERE ganyofileba = N'სავაჭრო'";

გ. myCommand.CommandText =
"DELETE FROM Personali WHERE Personali = N'სამედიცინო'";

17.24.17. mySqlCommand.CommandText =
"DELETE FROM Personali WHERE (ganyofileba = @ganyofileba)"; კოდის შესრულების შედეგად Personali:

ა. ცხრილიდან წაიშლება მითითებული განყოფილების თანამშრომლების შესაბამისი სტრიქონები

ბ. ცხრილიდან წაიშლება მითითებული ასაკის თანამშრომლების შესაბამისი სტრიქონები

გ. ცხრილს დაემატება მითითებული განყოფილების თანამშრომლების შესაბამისი სტრიქონები

ტრანზაქციებთან მუშაობა ADO.NET საშუალებებით

17.25.1. ADO.NET გარემოში ტრანზაქციებთან სამუშაოდ გამოიყენება:

ა. SqlTransaction ობიექტი

ბ. SqlCommand ობიექტი

გ. SqlConnection ობიექტი

17.25.2. ტრანზაქციის დასაწყებად უნდა შევასრულოთ:

ა. SqlConnection ობიექტის BeginTransaction() მეთოდი

ბ. ChangeDatabaze()მეთოდი

გ. Close() მეთოდი

17.25.3. ტრანზაქციის გამოსაყენებლად SqlCommand ობიექტის შექმნის შემდეგ მის:

ა. Transaction თვისებას უნდა მივანიჭოთ SqlTransaction ობიექტი

ბ. CommandText თვისებას უნდა მივანიჭოთ SqlTransaction ობიექტი
გ. Parameters თვისებას უნდა მივანიჭოთ SqlTransaction ობიექტი

17.25.4. ტრანზაქციის დასაფიქსირებლად უნდა გამოვიყენოთ SqlTransaction ობიექტის:
ა. Commit() მეთოდი
ბ. Rollback() მეთოდი
გ. Equals() მეთოდი

17.25.5. ტრანზაქციის უკუსაქცევად უნდა გამოვიყენოთ SqlTransaction ობიექტის:
ა. Rollback()მეთოდი
ბ. Commit() მეთოდი
გ. Equals() მეთოდი

შენახული პროცედურის გამოძახება

17.26.1. შესასრულებლად გამოძახებულ შენახულ პროცედურას:
ა. შეიძლება გადავცეთ პარამეტრები
ბ. აუცილებლად უნდა გადავცეთ პარამეტრები
გ. არ უნდა გადავცეთ პარამეტრები

17.26.2. თუ შენახული პროცედურა გასცემს სკალარულ შედეგს, მაშინ ამ პროცედურის შესასრულებლად:
ა. უნდა გამოვიყენოთ ExecuteScalar() და ExecuteNonQuery() მეთოდები
ბ. არ უნდა გამოვიყენოთ ExecuteScalar() და ExecuteNonQuery() მეთოდები
გ. უნდა გამოვიყენოთ Commit() მეთოდი

17.26.3. თუ პროცედურა გასცემს სტრიქონებს, მაშინ მის შესასრულებლად:
ა. უნდა გამოვიყენოთ ExecuteReader() მეთოდი
ბ. არ უნდა გამოვიყენოთ ExecuteReader() მეთოდი
გ. უნდა გამოვიყენოთ Commit() მეთოდი

17.26.4. SqlCommand ობიექტის CommandType თვისებას უნდა მივანიჭოთ CommandType კოლექციის:
ა. StoredProcedure თვისება
ბ. TableDirect თვისება
გ. Text თვისება

17.26.5. SqlCommand ობიექტის CommandText თვისებას:
ა. უნდა მივანიჭოთ გამოსაძახებელი შენახული პროცედურის სახელი
ბ. უნდა მივანიჭოთ გამოსაძახებელი შენახული პროცედურის ტიპი
გ. უნდა მივანიჭოთ გამოსაძახებელი ფუნქციის სახელი

17.26.6. mySqlCommand.Parameters.Add("@xelfasi", SqlDbType.Float, 4); კოდის შესრულების შედეგად mySqlCommand:
ა. ობიექტს დაემატება @asaki პარამეტრი
ბ. ობიექტიდან წაიშლება @asaki პარამეტრი
გ. ობიექტს დაემატება SqlDbType.Int პარამეტრი

17.26.7. mySqlCommand.Parameters.Add("@xelfasi", SqlDbType.Float, 4); კოდის შესრულების შედეგად mySqlCommand ობიექტს დაემატება @xelfasi პარამეტრი, რომელიც არის:

ა. მთელი რიცხვი
ბ. სტრიქონი
გ. წილადი

17.26.8. mySqlCommand.Parameters.Add("@gvari", SqlDbType.NVarChar, 50); კოდის შესრულების შედეგად mySqlCommand ობიექტს დაემატება @gvari პარამეტრი, რომელიც არის:

ა. უნიკოდების 50 ბაიტიანი სტრიქონი
ბ. უნიკოდების 50 სიმბოლოიანი სტრიქონი
გ. წილადი

17.26.9. mySqlCommand.Parameters["@xelfasi"].Value = Convert.ToDouble(textBox5.Text); კოდის შესრულების შედეგად @xelfasi პარამეტრს მიენიჭება:

ა. მთელი რიცხვი
ბ. წილადი
გ. სტრიქონი

17.26.10. mySqlCommand.Parameters["@raioni"].Value = textBox1.Text; კოდის შესრულების შედეგად @raioni პარამეტრს მიენიჭება:

ა. სტრიქონი
ბ. მთელი რიცხვი
გ. თარიღი

17.26.11. შენახული პროცედურის შესრულება:

ა. შეიძლება ადაპტერის Fill() მეთოდის გამოყენებით
ბ. არ შეიძლება ადაპტერის Fill() მეთოდის გამოყენებით
გ. შეიძლება ადაპტერის Update() მეთოდის გამოყენებით

ფუნქციის გამოძახება

17.27.1. შესასრულებლად გამოძახებულ ფუნქციას:

ა. შეიძლება გადავცეთ პარამეტრები
ბ. აუცილებლად უნდა გადავცეთ პარამეტრები
გ. არ უნდა გადავცეთ პარამეტრები

17.27.2. mySqlCommand.CommandText = "SELECT * FROM Maqsimaluri_Xelfasi(@ganyofileba)"; კოდის შესრულების შედეგად შესასრულებლად გამოძახებული იქნება:

ა. Maqsimaluri_Xelfasi(@ganyofileba) ფუნქცია
ბ. mySqlCommand(@ganyofileba) ფუნქცია
გ. CommandText(@ganyofileba) ფუნქცია

17.27.3. mySqlCommand.CommandText = "SELECT * FROM Maqsimaluri_Xelfasi(@ganyofileba)"; კოდის შესრულების შედეგად შესასრულებლად გამოძახებულ ფუნქციას:

ა. გადაეცემა @ganyofileba პარამეტრი
ბ. არ გადაეცემა @ganyofileba პარამეტრი
გ. გადაეცემა @Maqsimaluri_Xelfasi პარამეტრი

17.27.4. mySqlCommand.CommandText =
"SELECT * FROM Maqsimaluri_Xelfasi(@ganyofileba)";
კოდის შესასრულებლად უნდა შევასრულოთ ადაპტერის:
ა. Fill(myDataSet, "Maqsimaluri_Xelfasi"); მეთოდი
ბ. Update(myDataSet, "Maqsimaluri_Xelfasi"); მეთოდი
გ. Insert(myDataSet, "Maqsimaluri_Xelfasi"); მეთოდი

დანართი. ტესტების პასუხები

თავი 2. C# ენის საფუძვლები

საწყისი ცნებები

2.1.1. ბ
2.1.2. გ
2.1.3. ა
2.1.4. ა
2.1.5. ა
2.1.6. ბ
2.1.7. გ
2.1.8. ბ
2.1.9. ა
2.1.10. ბ
2.1.11. ა
2.1.12. გ
2.1.13. ბ
2.1.14. ა
2.1.15. ა
2.1.16. გ
2.1.17. ა
2.1.18. გ
2.1.19. ა
2.1.20. ბ
2.1.21. გ
2.1.22. ა
2.1.23. ბ

C# ენის საბაზო ტიპები

2.2.1. ა
2.2.2. ბ
2.2.3. ა
2.2.4. გ
2.2.5. ბ
2.2.6. ა
2.2.7. გ
2.2.8. ა
2.2.9. ა
2.2.10 გ
2.2.11. ბ
2.2.12. ა
2.2.13. გ
2.2.14. გ
2.2.15. გ
2.2.16. ბ
2.2.17. ბ
2.2.18. ა
2.2.19. ა
2.2.20. ბ
2.2.21. გ
2.2.22. ა
2.2.23. ა
2.2.24. გ
2.2.25. ა
2.2.26. გ
2.2.27. ბ
2.2.28. ა
2.2.29. ბ
2.2.30. ბ
2.2.31. გ
2.2.32. ა
2.2.33. გ
2.2.34. ა
2.2.35. ბ
2.2.36. გ
2.2.37. ა
2.2.38. ბ
2.2.39. ბ
2.2.40. გ
2.2.41. გ
2.2.42. ა
2.2.43. ა
2.2.44. ბ
2.2.45. გ
2.2.46. ა
2.2.47. ბ
2.2.48. ა
2.2.49. ა

ცვლადები და მათი ინიციალიზება

2.3.1. ა
2.3.2. ბ
2.3.3. ა
2.3.4. ბ
2.3.5. გ
2.3.6 ა

ოპერატორები

არითმეტიკის ოპერატორები

2.4.1. ა
2.4.2. ბ
2.4.3. გ

2.4.4. გ
2.4.5. გ
2.4.6. გ
2.4.7. ა
2.4.8. გ
2.4.9. გ
2.4.10. გ
2.4.11. ა
2.4.12. გ

ინკრემენტისა და დეკრემენტის ოპერატორები

2.4.13. გ
2.4.14. გ
2.4.15. ა
2.4.16. გ
2.4.17. გ
2.4.18. ა
2.4.19. გ
2.4.20. გ
2.4.21. ა
2.4.22. გ
2.4.23. გ
2.4.24. გ
2.4.25. გ
2.4.26. გ
2.4.27. გ
2.4.28. გ
2.4.29. გ
2.4.30. გ
2.4.31. გ
2.4.32. ა

შედარებისა და ლოგიკის ოპერატორები

2.5.1. გ
2.5.2. გ
2.5.3. ა
2.5.4. ა
2.5.5. გ
2.5.6. გ
2.5.7. გ
2.5.8. გ
2.5.9. გ
2.5.10. ა
2.5.11. გ
2.5.12. გ
2.5.13. გ
2.5.14. გ
2.5.15. ა
2.5.16. ა
2.5.17. გ
2.5.18. ა
2.5.19. გ
2.5.20. გ
2.5.21. ა
2.5.22. გ
2.5.23. ა
2.5.24. ა
2.5.25. გ
2.5.26. გ
2.5.27. ა
2.5.28. ა
2.5.29. გ
2.5.30. გ
2.5.31. ა

მინიჭების ოპერატორი. შედგენილი ოპერატორები

2.6.1. ა
2.6.2. გ
2.6.3. ა
2.6.4. გ
2.6.5. ა
2.6.6. გ
2.6.7. ა
2.6.8. გ
2.6.9. გ
2.6.10. გ
2.6.11. გ
2.6.12. ა

ოპერატორების პრიორიტეტები

2.7.1. ა
2.7.2. გ
2.7.3. ა
2.7.4. გ
2.7.5. ა

გამოსახულებები. მათემატიკის მეთოდები

2.8.1. გ
2.8.2. ა
2.8.3. გ
2.8.4. გ
2.8.5. გ

2.8.6. ა
2.8.7. ა
2.8.8. გ
2.8.9. ბ
2.8.10. ბ
2.8.11. ა
2.8.12. გ
2.8.13. გ
2.8.14. ბ
2.8.15. ა
2.8.16. ბ
2.8.17. ა

ტიპების გარდაქმნა
2.9.1. გ
2.9.2. ბ
2.9.3. ა
2.9.4. ა
2.9.5. გ
2.9.6. ა
2.9.7. ა
2.9.8. გ
2.9.9. გ
2.9.10. ა
2.9.11. ბ
2.9.12. ა
2.9.13. ა
2.9.14. გ
2.9.15. ბ
2.9.16. ა
2.9.17. გ
2.9.18. ბ
2.9.19. ბ
2.9.20. ბ
2.9.21. გ
2.9.22. ბ
2.9.23. ა
2.9.24. ა
2.9.25. გ
2.9.26. გ
2.9.27. გ
2.9.28. ა
2.9.29. ა
2.9.30. ბ
2.9.31. ბ
2.9.32. ა
2.9.33. ბ
2.9.34. გ
2.9.35. გ
2.9.36. ა
2.9.37. ბ
2.9.38. ა
2.9.39. ა
2.9.40. ბ
2.9.41. გ
2.9.42. გ
2.9.43. ა
2.9.44. ბ
2.9.45. ა
2.9.46. გ
2.9.47. ბ
2.9.48. გ
2.9.49. ბ
2.9.50. ა

თავი 3. მმართველი ოპერატორები
ამორჩევის ოპერატორები. if ოპერატორი
3.1.1. გ
3.1.2. ა
3.1.3. ა
3.1.4. გ
3.1.5. ბ
3.1.6. გ
3.1.7. ა
3.1.8. გ
3.1.9. ბ

ხილვადობის უბანი
3.2.1. ა
3.2.2. ბ
3.2.3. ა
3.2.4. ა
3.2.5. გ
3.2.6. გ
3.2.7. ა
3.2.8. ბ
3.2.9. გ
3.2.10. ბ
3.2.11. ა
3.2.12. ა
3.2.13. ბ
3.2.14. გ
3.2.15. ა
3.2.16. ბ

3.2.17. ა
3.2.18. ა
3.2.19. ა
3.2.20. ბ
3.2.21. ა
3.2.22. გ
3.2.23. გ
3.2.24. ა

ამორჩევის ოპერატორები.
switch ოპერატორი
3.3.1. ა
3.3.2. ბ
3.3.3. ა
3.3.4. გ
3.3.5. ა
3.3.6. ბ
3.3.7. ა
3.3.8. გ
3.3.9. ა
3.3.10. გ
3.3.11. ბ
3.3.12. გ
3.3.13. ბ

იტერაციის ოპერატორები
for ოპერატორი
3.4.1. გ
3.4.2. ბ
3.4.3. ა
3.4.4. ბ
3.4.5. გ
3.4.6. ა
3.4.7. ბ
3.4.8. ბ
3.4.9. გ
3.4.10. გ
3.4.11. ა
3.4.12. ბ
3.4.13. ა
3.4.14. ბ
3.4.15. გ
3.4.16. გ
3.4.17. გ
3.4.18. გ
3.4.19. ა
3.4.20. ბ
3.4.21. გ
3.4.22. ა
3.4.23. გ

while ოპერატორი
3.5.1. ა
3.5.2. ბ
3.5.3. გ
3.5.4. ა
3.5.5. ბ

do-while ოპერატორი
3.6.1. ა
3.6.2. ბ
3.6.3. გ
3.6.4. ა
3.6.5. ბ

გადასვლის ოპერატორები.
break ოპერატორი
3.7.1. ა
3.7.2. ბ
3.7.3. გ
3.7.4. ბ
3.7.5. გ
3.7.6. ა
3.7.7. გ

continue ოპერატორი
3.8.1. ა
3.8.2. ბ
3.8.3. გ
3.8.4. ბ
3.8.5. ბ
3.8.6. გ

goto ოპერატორი
3.9.1. ა
3.9.2. ბ
3.9.3. გ

ტერნარული ოპერატორი
3.10.1. ბ
3.10.2. ა
3.10.3. ბ
3.10.4. გ
3.10.5. ა

3.10.6. გ

თავი 4. მასივები, ბიტობრივი ოპერაციები და ჩამოთვლები

მასივები

ერთგანზომილებიანი მასივები

4.1.1. ა
4.1.2. გ
4.1.3. გ
4.1.4. გ
4.1.5. ა
4.1.6. გ
4.1.7. გ
4.1.8. ა
4.1.9. ა
4.1.10. გ
4.1.11. გ
4.1.12. გ
4.1.13. ა
4.1.14. გ
4.1.15. გ
4.1.16. გ
4.1.17. ა

ორგანზომილებიანი მასივები

4.2.1. გ
4.2.2. გ
4.2.3. ა
4.2.4. გ
4.2.5. გ
4.2.6. გ
4.2.7. გ
4.2.8. ა
4.2.9. გ

გაუსწორებელი მასივები

4.3.1. ა
4.3.2. გ
4.3.3. გ

Length თვისება

4.4.1. გ
4.4.2. ა
4.4.3. გ
4.4.4. გ
4.4.5. ა
4.4.6. გ
4.4.7. ა

foreach ოპერატორი

4.5.1. ა
4.5.2. გ
4.5.3. გ
4.5.4. ა

ბიტობრივი ოპერატორები

4.6.1. გ
4.6.2. ა
4.6.3. გ
4.6.4. გ
4.6.5. გ
4.6.6. გ
4.6.7. გ
4.6.8. გ
4.6.9. გ
4.6.10. გ
4.6.11. ა
4.6.12. გ
4.6.13. ა
4.6.14. გ
4.6.15. გ
4.6.16. ა
4.6.17. ა

ჩამოთვლები

4.7.1. ა
4.7.2. გ
4.7.3. გ
4.7.4. ა
4.7.5. გ
4.7.6. გ
4.7.7. ა
4.7.8. გ
4.7.9. გ
4.7.10. ა

თავი 5. კლასები, ინკაფსულაცია, მეთოდები

კლასების გამოცხადება

5.1.1. გ
5.1.2. ა
5.1.3. გ
5.1.4. გ
5.1.5. გ
5.1.6. გ

5.1.7. ა
5.1.8. ა
5.1.9. გ
5.1.10. ა
5.1.11. ბ
5.1.12. ბ
5.1.13. ა
5.1.14. გ
5.1.15. გ
5.1.16. გ
5.1.17. ბ

ინკაფსულაცია
5.2.1. ბ
5.2.2. ბ
5.2.3. ა
5.2.4. გ
5.2.5. ბ
5.2.6. გ
5.2.7. ა
5.2.8. ა
5.2.9. ბ
5.2.10. გ
5.2.11. ბ
5.2.12. ა
5.2.13. ბ
5.2.14. გ
5.2.15. ბ
5.2.16. ა
5.2.17. ბ
5.2.18. ა
5.2.19. გ

მეთოდები
5.3.1. ა
5.3.2. ბ
5.3.3. გ
5.3.4. ბ
5.3.5. ბ
5.3.6. ა
5.3.7. გ
5.3.8. გ
5.3.9. ა
5.3.10. ბ
5.3.11. ბ
5.3.12. ა
5.3.13. გ
5.3.14. გ
5.3.15. ა
5.3.16. ბ
5.3.17. გ
5.3.18. ბ
5.3.19. ა
5.3.20. ბ
5.3.21. ა

კონსტრუქტორი
5.4.1. ა
5.4.2. ა
5.4.3. გ
5.4.4. ა
5.4.5. ბ
5.4.6. გ
5.4.7. ა
5.4.8. ბ
5.4.9. გ
5.4.10. ა
5.4.11. გ
5.4.12. ბ
5.4.13. ბ
5.4.14. გ

დესტრუქტორი
5.5.1. ა
5.5.2. ბ
5.5.3. გ
5.5.4. ა
5.5.5. ბ
5.5.6. ბ
5.5.7. გ

this სიტყვა
5.6.1. ბ
5.6.2. ბ
5.6.3. გ
5.6.4. გ
5.6.5. ა
5.6.6. ა
5.6.7. ბ
5.6.8. გ
5.6.9. ა

რთული კლასები
5.7.1. ა

5.7.2. ბ
5.7.3. ბ
5.7.4. ა

ჩადგმული კლასები
5.8.1. ბ
5.8.2. ა
5.8.3. გ
5.8.4. გ
5.8.5. ა

სტრუქტურები
5.9.1. გ
5.9.2. ა
5.9.3. გ
5.9.4. ბ
5.9.5. ბ
5.9.6. ა
5.9.7. გ
5.9.8. ბ
5.9.9. ა
5.9.10. ა
5.9.11. ბ
5.9.12. ა

შემთხვევითი რიცხვების გენერატორი
5.10.1. ა
5.10.2. ბ
5.10.3. გ
5.10.4. ბ
5.10.5. ბ

თავი 6. მეთოდები უფრო დაწვრილებით. პოლიმორფიზმი
მეთოდისთვის ობიექტის გადაცემა
6.1.1. ბ
6.1.2. ა
6.1.3. გ
6.1.4. ა

მეთოდის მიერ ობიექტის დაბრუნება
6.2.1. ბ
6.2.2. ა
6.2.3. გ

მეთოდისთვის არგუმენტების გადაცემის ხერხები
6.3.1. ა
6.3.2. ბ
6.3.3. ბ
6.3.4. გ
6.3.5. ა
6.3.6. ა
6.3.7. ბ

ref და out მოდიფიკატორები
ref მოდიფიკატორი
6.4.1. გ
6.4.2. ა
out მოდიფიკატორი
6.5.3. ბ
6.5.4. ა
6.5.5. ბ
6.5.6. გ

params მოდიფიკატორი
6.6.1. გ
6.6.2. გ
6.6.3. ა

მეთოდის გადატვირთვა. პოლიმორფიზმი
6.7.1. ა
6.7.2. ბ
6.7.3. გ
6.7.4. ა
6.7.5. ბ
6.7.6. გ
6.7.7. გ
6.7.8. ა
6.7.9. ბ
6.7.10. გ

კონსტრუქტორების გადატვირთვა
6.8.1. ა
6.8.2. გ
6.8.3. ბ
6.8.4. ა

გადატვირთვადი კონსტრუქტორის გამოძახება this სიტყვის გამოყენებით
6.9.1. ბ
6.9.2. ა

6.9.3. გ
6.9.4. ა
6.9.5. გ

რეკურსია
6.10.1. ა
6.10.2. გ
6.10.3. გ

სტატიკური კომპონენტები
static მოდიფიკატორი
6.11.1. ა
6.11.2. გ
6.11.3. გ
6.11.4. გ
6.11.5. ა
6.11.6. გ
6.11.7. გ
6.11.8. ა
6.11.9. ა
6.11.10. გ

მუდმივები
6.12.1. გ
6.12.2. ა
6.12.3. გ

მხოლოდ წაკითხვადი ველები
6.13.1. ა
6.13.2. გ
6.13.3. გ

თავი 7. მემკვიდრეობითობა. ვირტუალური მეთოდები
მემკვიდრეობითობის საფუძვლები
7.1.1. გ
7.1.2. გ
7.1.3. ა
7.1.4. გ
7.1.5. ა
7.1.6. გ
7.1.7. გ
7.1.8. ა
7.1.9. გ

protected მოდიფიკატორი
7.2.1. გ
7.2.2. გ
7.2.3. ა
7.2.4. ა
7.2.5. გ
7.2.6. გ
7.2.7. ა
7.2.8. გ

კონსტრუქტორები და მემკვიდრეობითობა. base საკვანძო სიტყვა
7.3.1. გ
7.3.2. გ
7.3.3. გ
7.3.4. გ
7.3.5. ა
7.3.6. გ
7.3.7. გ
7.3.8. ა
7.3.9. გ
7.3.10. გ

ცვლადების დამალვა მემკვიდრეობითობის დროს
7.4.1. გ
7.4.2. გ
7.4.3. ა
7.4.4. გ
7.4.5. გ
7.4.6. გ

კლასების მრავალდონიანი იერარქიის შექმნა
7.5.1. ა
7.5.2. ა
7.5.3. გ

მიმართვები წინაპარი და მემკვიდრე კლასის ობიექტებზე
7.6.1. გ
7.6.2. ა
7.6.3. ა
7.6.4. გ

ვირტუალური მეთოდები
7.7.1. გ
7.7.2. ა
7.7.3. გ

7.7.4. გ
7.7.5. ა
7.7.6. გ
7.7.7. ა
7.7.8. გ
7.7.9. გ
7.7.10. ა
7.7.11. გ
7.7.12. გ
7.7.13. გ

აბსტრაქტული კლასები
7.8.1. გ
7.8.2. გ
7.8.3. ა
7.8.4. გ
7.8.5. გ
7.8.6. ა
7.8.7. გ
7.8.8. ა
7.8.9. გ
7.8.10. გ

მემკვიდრეობითობის აკრძალვა
7.9.1. ა
7.9.2. გ

პოლიმორფიზმის აკრძალვა
7.10.1. ა
7.10.2. გ
7.10.3. გ
7.10.4. გ

ობიექტების ტიპების დაყვანა
7.10.1. ა
7.10.2. გ
7.10.3. გ
7.10.4. გ
7.10.5. გ
7.10.6. ა

თავი 8. თვისებები, ინდექსატორები და ოპერატორების გადატვირთვა
თვისებები
8.1.1. გ
8.1.2. გ
8.1.3. ა
8.1.4. გ
8.1.5. ა
8.1.6. გ
8.1.7. გ
8.1.8. ა
8.1.9. გ
8.1.10. ა
8.1.11. გ
8.1.12. გ
8.1.13. ა

ინდექსატორი
ერთგანზომილებიანი ინდექსატორები
8.2.1. ა
8.2.2. ა
8.2.3. გ
8.2.4. გ
8.2.5. გ
8.2.6. ა

ოპერატორების გადატვირთვა
8.3.1. ა
8.3.2. გ
8.3.3. გ
8.3.4. გ
8.3.5. ა
8.3.6. გ
8.3.7. გ
8.3.8. გ
8.3.9. ა
8.3.10. გ
8.3.11. გ
8.3.12. გ
8.3.13. ა
8.3.14. ა
8.3.15. ა
8.3.16. გ
8.3.17. გ
8.3.18. გ
8.3.19. გ
8.3.20. გ
8.3.21. ა

თავი 9. დელეგატები, მოვლენები, ინტერფეისები და სახელების სივრცე
დელეგატები
9.1.1. ა

9.1.2. ა
9.1.3. ბ
9.1.4. გ
9.1.5. ა
9.1.6. გ
9.1.7. ბ
9.1.8. ა
9.1.9. ა
9.1.10. ბ
9.1.11. გ
9.1.12. ბ

დელეგატების მრავალმისამართიანობა
9.1.13. ა
9.1.14. ბ
9.1.15. ა
9.1.16. ბ
9.1.17. ბ
9.1.18. გ

მოვლენები
9.2.1. გ
9.2.2. ა
9.2.3. ბ
9.2.4. ბ
9.2.5. ა
9.2.6. ბ
9.2.7. გ

ფართოსამაუწყებლო მოვლენა
9.2.8. ბ
9.2.9. ა
9.2.10. გ
9.2.11. გ
9.2.12. ბ

ინტერფეისები
ინტერფეისების რეალიზება
9.3.1. გ
9.3.2. ა
9.3.3. ბ
9.3.4. ბ
9.3.5. ა
9.3.6. გ
9.3.7. გ
9.3.8. ბ
9.3.9. გ
9.3.10. ბ
9.3.11. ა
9.3.12. ბ

კლასების მემკვიდრეობითობა და ინტერფეისის რეალიზება
9.4.1. ა
9.4.2. ბ
9.4.3. გ
9.4.4. ბ
9.4.5. ა

წარმოებული ინტერფეისები
9.5.1. ა
9.5.2. ბ
9.5.3. ბ
9.5.4. გ

ინტერფეისული მიმართვები
9.6.1. ა
9.6.2. ბ
9.6.3. გ
9.6.4. ა
9.6.5. ბ
9.6.6. ბ
9.6.7. გ

ინტერფეისის თვისებები
9.7.1. ა

ცხადი რეალიზება
9.8.1. ა
9.8.2. ბ

სახელების სივრცე
9.9.1. ა
9.9.2. ბ
9.9.3. ა
9.9.4. გ
9.9.5. ბ
9.9.6. ბ

using დირექტივა
9.10.7. ა
9.10.8. გ
9.10.9. ა
9.10.10. ბ

ჩადგმული სახელების სივრცე
9.11.11. გ
9.11.12. ა

ავტომატურად განსაზღვრული სახელების სივრცე
9.12.13. გ
9.12.14. გ

თავი 10. ინფორმაციის შეტანა-გამოტანა
შეტანა-გამოტანის ნაკადების კლასები
10.1.1. ა
10.1.2. გ
10.1.3. გ
10.1.4. გ
10.1.5. ა
10.1.6. გ
10.1.7. ა
10.1.8. ა
10.1.9. გ

ბაიტების ნაკადები
10.1.10. გ
10.1.11. გ
10.1.12. ა
10.1.13. ა
10.1.14. გ
10.1.15. გ
10.1.16. გ
10.1.17. ა
10.1.18. გ
10.1.19. გ
10.1.20. გ
10.1.21. ა
10.1.22. ა
10.1.23. გ
10.1.24. გ

სიმბოლოების ნაკადები
10.1.25. გ
10.1.26. გ
10.1.27. გ
10.1.28. ა
10.1.29. გ
10.1.30. გ
10.1.31. გ
10.1.32. გ
10.1.33. ა
10.1.34. ა
10.1.35. გ
10.1.36. გ
10.1.37. გ
10.1.38. გ
10.1.39. ა
10.1.40. გ
10.1.41. გ
10.1.42. ა

ორობითი ნაკადები
10.1.43. გ

FileStream კლასი
ფაილის გახსნა და დახურვა
10.2.1. ა
10.2.2. გ
10.2.3. გ
10.2.4. გ
10.2.5. ა
10.2.6. ა
10.2.7. გ
10.2.8. ა
10.2.9. გ
10.2.10. გ
10.2.11. ა
10.2.12. გ

ფაილიდან ბაიტების წაკითხვა
10.2.13. გ
10.2.14. ა
10.2.15. ა
10.2.16. გ

ფაილში ბაიტების ჩაწერა
10.2.17. გ
10.2.18. გ
10.2.19. ა
10.2.20. ა
10.2.21. გ
10.2.22. გ

ფაილში სიმბოლოების შეტანა-გამოტანა
10.3.1. გ
10.3.2. ა

10.3.3. გ

ორობითი მონაცემების შეტანა-გამოტანა

BinaryWriter კლასი

10.4.1. გ
10.4.2. ა
10.4.3. ბ
10.4.4. ბ
10.4.5. გ
10.4.6. ა
10.4.7. ბ
10.4.8. გ
10.4.9. ბ
10.4.10. ა
10.4.11. ა
10.4.12. გ
10.4.13. ბ
10.4.14. ა
10.4.15. გ
10.4.16. ბ

BinaryReader ნაკადი

10.4.17. ა
10.4.18. გ
10.4.19. ბ
10.4.20. ა
10.4.21. ბ
10.4.22. ა
10.4.23. გ
10.4.24. ა
10.4.25. ბ
10.4.26. გ
10.4.27. ა
10.4.28. ა
10.4.29. ა
10.4.30. ა
10.4.31. ა
10.4.32. ბ
10.4.33. გ
10.4.34. ბ
10.4.35. გ
10.4.36. გ
10.4.37. ა
10.4.38. ა
10.4.39. ბ
10.4.40. გ

ფაილებთან პირდაპირი მიმართვა

10.5.1. ა
10.5.2. ბ
10.5.3. გ
10.5.4. ბ
10.5.5. ა
10.5.6. გ
10.5.7. გ
10.5.8. ა
10.5.9. გ
10.5.10. ბ
10.5.11. ბ
10.5.12. გ
10.5.13. ა

ფაილის შესახებ ინფორმაციის მიღება

10.6.1. ა
10.6.2. ბ
10.6.3. გ
10.6.4. ა
10.6.5. ა
10.6.6. ბ
10.6.7. გ
10.6.8. ა
10.6.9. ბ
10.6.10. ა
10.6.11. გ
10.6.12. ა
10.6.13. ბ
10.6.14. ა
10.6.15. ბ
10.6.16. გ
10.6.17. ა
10.6.18. ა
10.6.19. ბ
10.6.20. ბ
10.6.21. გ
10.6.22. გ
10.6.23. ა
10.6.24. ა
10.6.25. ბ

კატალოგებთან მუშაობა

10.7.1. ა
10.7.2. ბ
10.7.3. გ
10.7.4. ბ

10.7.5. ა
10.7.6. ა
10.7.7. გ
10.7.8. ბ
10.7.9. ა
10.7.10. გ
10.7.11. ბ
10.7.12. ა
10.7.13. ბ
10.7.14. გ
10.7.15. ა
10.7.16. ბ
10.7.17. გ
10.7.18. ა
10.7.19. ბ

NetworkStream კლასი

10.8.1. ბ
10.8.2. გ
10.8.3. ა
10.8.4. გ
10.8.5. ბ
10.8.6. ა
10.8.7. გ
10.8.8. გ
10.8.9. ა
10.8.10. ბ
10.8.11. ბ
10.8.12. ა
10.8.13. ბ
10.8.14. გ
10.8.15. ბ
10.8.16. ა
10.8.17. ა

MemoryStream და BufferedStream კლასები

10.9.1. ა
10.9.2. გ
10.9.3. გ
10.9.4. ბ
10.9.5. ბ
10.9.6. ა
10.9.7. ა
10.9.8. ბ

მონაცემების ასინქრონული შეტანა-გამოტანა

10.10.1. გ
10.10.2. ა
10.10.3. ბ
10.10.4. გ
10.10.5. ა

თავი 11. განსაკუთრებული სიტუაციები

განსაკუთრებული სიტუაციების დამუშავების საფუძვლები

11.1.1. გ
11.1.2. ბ
11.1.3. ა
11.1.4. ა
11.1.5. გ
11.1.6. ბ
11.1.7. ბ
11.1.8. ა
11.1.9. გ
11.1.10. ბ
11.1.11. გ
11.1.12. ბ
11.1.13. გ
11.1.14. ა
11.1.15. ბ
11.1.16. ბ
11.1.17. ა
11.1.18. გ
11.1.19. ა
11.1.20. ბ
11.1.21. ბ
11.1.22. გ
11.1.23. გ
11.1.24. ა

try და catch ბლოკები

11.2.1. ა
11.2.2. გ
11.2.3. ბ
11.2.4. ა
11.2.5. ბ
11.2.6. გ
11.2.7. გ
11.2.8. ბ
11.2.9. ა
11.2.10. ა

11.2.11. ბ

ჩადგმული try ბლოკები

11.3.1. გ
11.3.2. ა

throw ოპერატორი

11.4.1. ა

finally ბლოკი

11.5.1. ბ
11.5.2. გ

checked და unchecked ოპერატორები

11.6.1. ა
11.6.2. ბ
11.6.3. გ
11.6.4. ბ
11.6.5. ა
11.6.6. გ
11.6.7. ბ
11.6.8. ა
11.6.9. ბ
11.6.10. გ
11.6.11. ბ

თავი 12. სტრიქონები

სტრიქონები

12.1.1. ა
12.1.2. ა
12.1.3. ბ
12.1.4. ბ
12.1.5. ა
12.1.6. გ
12.1.7. გ
12.1.8. გ
12.1.9. ბ
12.1.10. ა
12.1.11. ბ
12.1.12. გ
12.1.13. გ
12.1.14. ა

სტრიქონების მასივები

12.1.15. ბ

სტრიქონების უცვლელობა

12.1.16. ბ

დინამიკური სტრიქონები

12.2.1. ა
12.2.2. ბ
12.2.3. გ
12.2.4. ა
12.2.5. ბ
12.2.6. გ
12.2.7. ა
12.2.8. ბ
12.2.9. გ
12.2.10. ა
12.2.11. ბ
12.2.12. გ
12.2.13. ა
12.2.14. ბ
12.2.15. გ
12.2.16. ა
12.2.17. ბ
12.2.18. გ
12.2.19. ა
12.2.20. ბ
12.2.21. გ
12.2.22. ა
12.2.23. ბ
12.2.24. გ

თავი 13. თარიღი, დრო და დროის ინტერვალები

თარიღი და დრო

13.1.1. ა
13.1.2. ბ
13.1.3. გ
13.1.4. ა
13.1.5. ბ
13.1.6. გ
13.1.7. ა
13.1.8. ბ
13.1.9. გ
13.1.10. ა
13.1.11. ბ
13.1.12. გ
13.1.13. ა
13.1.14. ბ
13.1.15. გ

13.1.16. ა
13.1.17. ბ
13.1.18. გ
13.1.19. ა
13.1.20. ბ
13.1.21. გ
13.1.22. ა
13.1.23. ბ
13.1.24. გ
13.1.25. ბ
13.1.26. გ
13.1.27. ა
13.1.28. გ

დროის ინტერვალები

13.2.1. ბ
13.2.2. გ
13.2.3. ა
13.2.4. ა
13.2.5. ბ
13.2.6. გ
13.2.7. ა
13.2.8. ბ
13.2.9. გ
13.2.10. ა
13.2.11. ბ
13.2.12. გ
13.2.13. ა
13.2.14. ბ
13.2.15. გ

თავი 14. კოლექციები

დინამიკური მასივები

14.1.1. ა
14.1.2. ბ
14.1.3. გ
14.1.4. ა
14.1.5. ბ
14.1.6. გ
14.1.7. ა
14.1.8. ბ
14.1.9. გ
14.1.10. ა
14.1.11. ბ
14.1.12. გ
14.1.13. ა
14.1.14. ბ
14.1.15. გ
14.1.16. ა
14.1.17. ბ
14.1.18. ა
14.1.19. გ
14.1.20. ბ
14.1.21. გ
14.1.22. ა
14.1.23. ბ
14.1.24. გ
14.1.25. ა
14.1.26. ბ
14.1.27. გ

ჰეშ-ცხრილები

14.2.1. ა
14.2.2. ბ
14.2.3. გ
14.2.4. ა
14.2.5. ბ
14.2.6. გ
14.2.7. ა
14.2.8. ბ
14.2.9. გ
14.2.10. ა
14.2.11. ბ
14.2.12. გ
14.2.13. ა
14.2.14. ბ
14.2.15. გ
14.2.16. ა
14.2.17. ბ
14.2.18. გ

დახარისხებული სიები

14.3.1. ა
14.3.2. ა
14.3.3. ბ
14.3.4. გ
14.3.5. ა
14.3.6. ბ
14.3.7. გ
14.3.8. ა
14.3.9. ბ
14.3.10. გ
14.3.11. გ
14.3.12. ა

14.3.13. ბ
14.3.14. გ
14.3.15. ა
14.3.16. ბ
14.3.17. გ
14.3.18. ა
14.3.19. ბ
14.3.20. გ
14.3.21. ა
14.3.22. ბ
14.3.23. გ
14.3.24. ა
14.3.25. ბ
14.3.26. გ

რიგები
14.4.1. ა
14.4.2. ბ
14.4.3. გ
14.4.4. ა
14.4.5. ბ
14.4.6. გ
14.4.7. ა
14.4.8. ბ
14.4.9. გ
14.4.10. ა
14.4.11. ბ

სტეკები
14.5.1. ა
14.5.2. ბ
14.5.3. გ
14.5.4. ა
14.5.5. ბ
14.5.6. გ
14.5.7. ა
14.5.8. ბ
14.5.9. გ
14.5.10. გ
14.5.11. ბ

ბიტების მასივები
14.6.1. ა
14.6.2. ბ
14.6.3. გ
14.6.4. ა
14.6.5. ბ
14.6.6. გ
14.6.7. ა
14.6.8. ბ
14.6.9. გ
14.6.10. ბ
14.6.11. ა
14.6.12. გ
14.6.13. ბ
14.6.14. ა
14.6.15. ბ

თავი 15. შესრულების ნაკადები

შესრულების ნაკადის განსაზღვრა
15.1.1. ა
15.1.2. ბ
15.1.3. გ
15.1.4. ა
15.1.5. ბ
15.1.6. გ
15.1.7. ა
15.1.8. ბ
15.1.9. გ
15.1.10. ა
15.1.11. ბ
15.1.12. გ
15.1.13. ა
15.1.14. ბ
15.1.15. გ
15.1.16. ა
15.1.17. ბ
15.1.18. გ
15.1.19. ა
15.1.20. ბ
15.1.21. გ
15.1.22. ა
15.1.23. ბ
15.1.24. გ
15.1.25. ა
15.1.26. ბ
15.1.27. გ
15.1.28. ა
15.1.29. ბ
15.1.30. გ
15.1.31. ა
15.1.32. ბ

შესრულების ნაკადის შექმნა

15.2.1. ა
15.2.2. გ
15.2.3. ა
15.2.4. ბ
15.2.5. გ
15.2.6. ა
15.2.7. ბ

შესრულების ნაკადის პრიორიტეტები

15.3.1. ა
15.3.2. გ
15.3.3. ა
15.3.4. ბ
15.3.5. გ
15.3.6. გ

შესრულების ნაკადის მდგომარეობები

15.4.1. ა
15.4.2. ბ
15.4.3. ა
15.4.4. ბ
15.4.5. გ
15.4.6. ა
15.4.7. გ
15.4.8. გ
15.4.9. ა
15.4.10. ბ
15.4.11. გ
15.4.12. ა
15.4.13. ბ
15.4.14. გ
15.4.15. ა
15.4.16. ბ

შესრულების ნაკადის ლოკალური მონაცემები

15.5.1. ა
15.5.2. ბ
15.5.3. გ
15.5.4. ა

შესრულების ნაკადების მართვა

Timer კლასი

15.6.1. ა
15.6.2. ბ
15.6.3. გ
15.6.4. ა
15.6.5. ბ
15.6.6. გ
15.6.7. ა

Join() მეთოდი

15.7.1. ა
15.7.2. ბ
15.7.3. გ
15.7.4. ა
15.7.5. ბ

lock საკვანძო სიტყვა

15.8.1. ა
15.8.2. გ
15.8.3. გ

Interlocked კლასი

15.9.1. ა
15.9.2. ბ
15.9.3. გ
15.9.4. გ
15.9.5. ა
15.9.6. ბ

Monitor კლასი

15.10.1. ა
15.10.2. ბ
15.10.3. ბ
15.10.4. გ
15.10.5. ა
15.10.6. ბ
15.10.7. გ

Mutex კლასი

15.11.1. ა
15.11.2. ბ
15.11.3. გ
15.11.4. ა
15.11.5. ბ
15.11.6. გ
15.11.7. ა

თავი 16. საბაზო ბიბლიოთეკის სხვა კლასები

გლობალიზაცია

16.1.1. ა
16.1.2. ბ
16.1.3. გ
16.1.4. ა
16.1.5. ბ
16.1.6. გ
16.1.7. ა
16.1.8. ბ
16.1.9. გ
16.1.10. ა
16.1.11. ბ
16.1.12. გ
16.1.13. ა
16.1.14. ბ
16.1.15. გ
16.1.16. ა
16.1.17. ბ
16.1.18. გ

მონაცემების დაშიფვრა და გაშიფვრა

16.2.1. ა
16.2.2. ბ
16.2.3. გ
16.2.4. ა
16.2.5. ბ
16.2.6. გ
16.2.7. ა
16.2.8. ბ
16.2.9. გ
16.2.10. ა
16.2.11. ბ
16.2.12. გ
16.2.13. ა
16.2.14. ბ
16.2.15. გ
16.2.16. ა
16.2.17. ბ
16.2.18. გ
16.2.19. ა
16.2.20. ბ
16.2.21. გ
16.2.22. ა
16.2.23. ბ
16.2.24. გ
16.2.25. ა
16.2.26. ბ
16.2.27. გ
16.2.28. ა
16.2.29. ბ

სერიულ პორტებთან მუშაობა

16.3.1. ა
16.3.2. ბ
16.3.3. გ
16.3.4. ა
16.3.5. ბ
16.3.6. გ
16.3.7. ბ
16.3.8. ა
16.3.9. ა
16.3.10. ბ
16.3.11. ა
16.3.12. ბ
16.3.13. ა
16.3.14. ბ
16.3.15. ბ

თავი 17. ADO.NET ბიბლიოთეკა

მონაცემთა ბაზასთან მიმართვა Viaual Studio .NET-ის საშუალებებით

17.1.1. ა
17.1.2. ბ
17.1.3. გ
17.1.4. ა
17.1.5. გ
17.1.6. ბ
17.1.7. ა
17.1.8. ა
17.1.9. ბ
17.1.10. ბ
17.1.11. გ
17.1.12. გ
17.1.13. ა
17.1.14. ბ
17.1.15. გ
17.1.16. ბ
17.1.17. ა
17.1.18. გ
17.1.19. ბ
17.1.20. ა

მონაცემთა ბაზასთან მუშაობა ვიზუალური და არავიზუალური კომპონენტებით

17.2.1. ა
17.2.2. ბ
17.2.3. გ
17.2.4. გ
17.2.5. ბ
17.2.6. ა

ცხრილებს შორის ბმების შექმნა

17.3.1. ა
17.3.2. ბ
17.3.3. გ
17.3.4. გ

მონაცემების გაფილტვრა

17.4.1. ბ
17.4.2. ა
17.4.3. გ
17.4.4. ა
17.4.5. გ
17.4.6. ბ
17.4.7. ა
17.4.8. ბ
17.4.9. ა
17.4.10. გ
17.4.11. ა

მონაცემების დახარისხება

17.5.1. ბ
17.5.2. გ
17.5.3. გ
17.5.4. ბ
17.5.5. ბ
17.5.6. ა
17.5.7. გ
17.5.8. ბ
17.5.9. გ
17.5.10. ა
17.5.11. ა
17.5.12. ბ
17.5.13. გ
17.5.14. ა

მონაცემების ძებნა

17.6.1. ბ
17.6.2. ბ
17.6.3. გ
17.6.4. ა
17.6.5. ბ
17.6.6. გ
17.6.7. გ
17.6.8. ა

ნავიგაცია

17.7.1. ა
17.7.2. ბ
17.7.3. გ
17.7.4. ბ
17.7.5. ა
17.7.6. ა
17.7.7. ბ
17.7.8. გ
17.7.9. ა
17.7.10. ბ
17.7.11. ა
17.7.12. გ
17.7.13. ბ

გამოთვლები

17.8.1. ა
17.8.2. ბ
17.8.3. ბ
17.8.4. გ
17.8.5. გ
17.8.6. ა
17.8.7. ა
17.8.8. ბ
17.8.9. ბ
17.8.10. გ
17.8.11. გ
17.8.12. ა
17.8.13. ბ
17.8.14. ა
17.8.15. ა
17.8.16. გ
17.8.17. ა

ცხრილებს შორის არსებული ბმებით მანიპულირება

17.9.1. ა
17.9.2. ბ

ADO.NET კლასების მიმოხილვა

ADO.NET კლასები და ობიექტები

17.10.1. ა
17.10.2. ბ
17.10.3. ბ
17.10.4. ბ
17.10.5. ბ
17.10.6. ბ
17.10.7. ბ
17.10.8. ბ
17.10.9. ბ
17.10.10. ბ
17.10.11. ბ
17.10.12. ბ
17.10.13. ბ
17.10.14. ბ
17.10.15. ბ
17.10.16. ბ
17.10.17. ბ
17.10.18. ბ
17.10.19. ბ
17.10.20. ბ
17.10.21. ბ
17.10.22. ბ
17.10.23. ბ
17.10.24. ბ
17.10.25. ბ
17.10.26. ბ
17.10.27. ბ
17.10.28. ბ
17.10.29. ბ
17.10.30. ბ
17.10.31. ბ
17.10.32. ბ
17.10.33. ბ
17.10.34. ბ
17.10.35. ბ
17.10.36. ბ
17.10.37. ბ
17.10.38. ბ
17.10.39. ბ
17.10.40. ბ
17.10.41. ბ
17.10.42. ბ
17.10.43. ბ
17.10.44. ბ
17.10.45. ა
17.10.46. ა
17.10.47. ა
17.10.48. ა
17.10.49. ა
17.10.50. ა
17.10.51. ა
17.10.52. ა

მოთხოვნების შესრულება ADO.NET კლასების გამოყენებით

მონაცემების წაკითხვა DataReader ობიექტის საშუალებით

17.11.1. ა
17.11.2. ბ
17.11.3. ბ
17.11.4. ბ
17.11.5. ბ
17.11.6. გ
17.11.7. გ
17.11.8. გ
17.11.9. გ
17.11.10. გ
17.11.11. გ
17.11.12. გ
17.11.13. გ
17.11.14. გ
17.11.15. გ
17.11.16. გ
17.11.17. გ
17.11.18. გ
17.11.19. გ
17.11.20. გ
17.11.21. გ
17.11.22. გ
17.11.23. ა
17.11.24. ბ
17.11.25. ბ
17.11.26. ბ

მონაცემთა ბაზასთან მუშაობა DataSet ობიექტის გამოყენებით

მონაცემების კითხვა

17.12.1. ა
17.12.2. ა
17.12.3. ა
17.12.4. ბ
17.12.5. ბ

17.12.6. ბ
17.12.7. ბ
17.12.8. ბ
17.12.9. ბ
17.12.10. ბ
17.12.11. ბ
17.12.12. ბ
17.12.13. ბ
17.12.14. ბ
17.12.15. ბ
17.12.16. ბ
17.12.17. ბ
17.12.18. ბ
17.12.19. ბ
17.12.20. ბ
17.12.21. ბ
17.12.22. ბ
17.12.23. ბ
17.12.24. ბ
17.12.25. ბ
17.12.26. ბ
17.12.27. ბ
17.12.28. ბ
17.12.29. ბ
17.12.30. ბ
17.12.31. ბ
17.12.32. ბ
17.12.33. ბ
17.12.34. ბ
17.12.35. ბ
17.12.36. ბ
17.12.37. ბ
17.12.38. ბ

მონაცემების გაფილტვრა

17.13.1. ბ
17.13.2. ბ
17.13.3. ბ
17.13.4. ბ
17.13.5. ბ
17.13.6. ბ
17.13.7. ბ
17.13.8. ბ
17.13.9. ბ
17.13.10. ბ

მონაცემების დახარისხება

17.14.1. ბ
17.14.2. ბ
17.14.3. ბ
17.14.4. ბ

სტრიქონების ძებნა

17.15.1. ბ
17.15.2. ბ
17.15.3. ბ
17.15.4. ბ
17.15.5. ბ
17.15.6. ბ
17.15.7. ბ
17.15.8. ბ
17.15.9. ბ
17.15.10. ბ
17.15.11. ბ
17.15.12. ბ

მონაცემთა ბაზაში მონაცემების შეცვლა

17.16.1. ბ
17.16.2. ბ
17.16.3. ბ
17.16.4. ბ
17.16.5. ბ
17.16.6. ბ
17.16.7. ბ
17.16.8. ბ
17.16.9. ბ
17.16.10. ბ
17.16.11. ბ

SELECT, UPDATE, INSERT და DELETE ბრძანებების ნახვა

17.17.1. ბ
17.17.2. ბ
17.17.3. ბ
17.17.4. ბ
17.17.5. ა
17.17.6. ბ
17.17.7. ბ
17.17.8. ა
17.17.9. გ
17.17.10. ბ

მონაცემთა ბაზაში სტრიქონების დამატება

17.18.1. ა
17.18.2. ა
17.18.3. ა
17.18.4. ა
17.18.5. ა
17.18.6. ა
17.18.7. ა
17.18.8. ა
17.18.9. ა
17.18.10. ა
17.18.11. ა
17.18.12. ა
17.18.13. ა

სტრიქონების წაშლა

17.19.1. ა
17.19.2. ა
17.19.3. ა
17.19.4. ა
17.19.5. ა

ორ ცხრილს შორის ბმის შექმნა

17.20.1. ა
17.20.2. ა
17.20.3. ა
17.20.4. ა
17.20.5. ა
17.20.6. ა
17.20.7. ა
17.20.8. ა
17.20.9. ა
17.20.10. ა
17.20.11. ა
17.20.12. ა

SQL ბრძანებების უშუალო შესრულება

ერთი მნიშვნელობის მიღება

17.21.1. ა
17.21.2. ა
17.21.3. ა
17.21.4. ა
17.21.5. ა

UPDATE მოთხოვნის უშუალოდ შესრულება

17.22.1. ა
17.22.2. ა
17.22.3. ა
17.22.4. ა
17.22.5. ა
17.22.6. ა
17.22.7. ა
17.22.8. ა
17.22.9. ა
17.22.10. ა
17.22.11. ა
17.22.12. ა
17.22.13. ა
17.22.14. ა
17.22.15. ა
17.22.16. ა
17.22.17. ა
17.22.18. ა
17.22.19. ა
17.22.20. ა

INSERT მოთხოვნის უშუალოდ შესრულება

17.23.1. ა
17.23.2. ა
17.23.3. ა
17.23.4. ა
17.23.5. ა
17.23.6. ა
17.23.7. ა
17.23.8. ა
17.23.9. ა
17.23.10. ა
17.23.11. ა
17.23.12. ა
17.23.13. ა
17.23.14. ა
17.23.15. ა
17.23.16. ა
17.23.17. ა
17.23.18. ა

DELETE მოთხოვნის უშუალოდ შესრულება

17.24.1. ა
17.24.2. ა
17.24.3. ა
17.24.4. ა
17.24.5. ა
17.24.6. ა
17.24.7. ა

17.24.8. ა
17.24.9. ა
17.24.10. ა
17.24.11. ა
17.24.12. ა
17.24.13. ა
17.24.14. ა
17.24.15. ა
17.24.16. ა
17.24.17. ა

ტრანზაქციებთან მუშაობა ADO.NET საშუალებებით

17.25.1. ა
17.25.2. ა
17.25.3. ა
17.25.4. ა
17.25.5. ა

შენახული პროცედურის გამოძახება

17.26.1. ა
17.25.2. ა
17.25.3. ა
17.25.4. ა
17.25.5. ა
17.25.6. ა
17.25.7. გ
17.25.8. ა
17.25.9. ა
17.25.10. ა
17.25.11. ა

ფუნქციის გამოძახება

17.27.1. ა
17.27.2. ა
17.27.3. ა
17.27.4. ა

ლიტერატურა

1. რ. სამხარაძე. Visual C#.NET. საქართველოს ტექნიკური უნივერსიტეტი. 2009. - 458 გვ.
2. Шилдт. Г. C# .
3. Прайс дж., Гандерлой М. Visual C#.NET .
4. Смайли Дж. Учимся программировать на C# .
5. Davis. H. Visual C#.NET Programming.
6. Butow E., Ryan T. C# .
7. Hejlsberg A., Wiltamuth S. C# Language Reference.
8. O'Reilly. Programming C# .
9. Г. Дейтел. Введение в операционные системы. В 2-х томах. Пер. с англ. - М.: Мир, 1987.
10. Э. Таненбаум. Современные операционные системы. - СПб.: Питер, 2002. - 1040 с.: ил.
11. Saurabh Nandu. C#.NET. Web Developer's Guide.
12. O'Reilly. Programming C#.
13. Разработка Windows-приложений на Microsoft Visual Basic .NET и Microsoft Visual C# -NET. Учебный курс MCAD/MCSD/Пер. с англ. - М.: Издательско-торговый дом "Русская Редакция", 2003. - 512 стр.: ил.
14. E.Butow, T. Ryan. Your visual blueprint for building .NET applications.
15. Harold Davis. Visual C# .NET Programming.
16. A. Hejlsberg, S.Wiltamuth. C# Language Reference.
17. Г. Шилдт. Полный справочник по C#.
18. C# Frequently Asked Questions <http://blogs.msdn.com/csharpfaq/default.aspx>.
19. O'REILLY. C# 3.0 Cookbook. 2007.

რედაქტორი ა. ჩიქოვანი

გადაეცა წარმოებას 21.12.2011. ხელმოწერილია დასაბეჭდად 20.01.2012. ქაღალდის ზომა 60X84 1/8. პირობითი ნაბეჭდი თაბახი 16. ტირაჟი 50 ეგზ.

საგამომცემლო სახლი “ტექნიკური უნივერსიტეტი”, თბილისი, კოსტავას 77