Prvi međuispit iz Programskih paradigmi i jezika
31. ožujka 2010.

1. **(1,5 bodova)** Navedite koja se vremena povezivanja varijable s njenom vrijednošću koriste u navedena tri programska odsječka. Vremena povezivanja je potrebno samo navesti, a ne detaljno opisivati.

1.	`private const int TITLE_BAR_COLOR = 0xFF;`
	`...`
	`titleBar.color = TITLE_BAR_COLOR;`

2.	`titleBar.color = 0xFF;`
3.	`titleBar.color = ReadTitleBarColor();`

2. **(1,5 bodova)** Navedena je funkcija koja pretvara milje u kilometre i obrnuto. Parametri postupka su udaljenost i vrsta pretvorbe koju funkcija treba obaviti. Programski odsječak napišite ponovno uvažavajući sljedeće smjernice:

a) Predložite adekvatno imenovanje funkcije, argumenata i varijabli
b) Uklonite magične brojeve
c) Uvedite blokove s eksplicitnim granicama

(1 bod) Za a) je potrebno navesti koje su se smjernice za imenovanje koristile pri restrukturiranju odsječka. Smjernice nije potrebno detaljno objašnjavati.

```
float x(float y,bool c){
  if(y<0.0f){return 0.0f;}
  if(c==true) {
        return y*0.621371f;
  }else
        return y*1.609344f;
}
```

3. **(2 boda)** U najviše tri rečenice komentirajte koheziju funkcije iz prethodnog zadatka. Ukoliko kohezija nije adekvatna predložite izmjene u cilju postizanja funkcionalne kohezije te napišite restrukturirani programski odsječak s funkcionalnom kohezijom.

4. **(2 boda)** Navedite čemu služi `foreach` petlja i navedite barem 3 programska jezika koji je podržavaju.

5. **(2 boda)** Ukratko objasnite princip objektno orijentiranog programiranja koji nazivamo polimorfizmom. Riječima opišite jedan primjer iz života (ili programiranja) u kojem se koristi polimorfizam. Primjer s vratima iz predavanja nije dozvoljen i neće se prihvaćati kao točno rješenje.

6. **(5 bodova)** U programskom jeziku C# ostvariti razred `Trokut` koji predstavlja koncept trokuta kao geometrijskog lika koji ima definirane točke vrhova i ugrađenu funkcionalnost izračunavanja duljina stranica, opsega i površine. Napisati i glavni program kojim se stvara 5 objekata razreda trokut te ih se pohranjuje u generičku listu (tip `List<>`) (glavni program ne treba sadržavati ništa osim navedenog).

Rješenja:

1)

1. Povezivanje u trenutku prevođenja programa
2. Povezivanje u trenutku pisanja programa
3. Povezivanje u trenutku izvođenja programa

2)

```
const float COEF_MILE_TO_KM = 1.609344f;
const float COEF_KM_TO_MILE = 0.621371f;

float PretvoriUMiljeIliKilometre(float duljina, bool miljeUKm)
{

    if (duljina < 0.0f)

     {
           return 0.0f;
     }

     if (miljeUKm)
     {
           return duljina * COEF_MILE_TO_KM;
     }
     else
     {
           return duljina * COEF_KM_TO_MILE;
     }

}
```

Korištene smjernice:

a) Funkcije:
- ime postupka mora opisivati ono što postupak radi
- Ime se obično sastoji od:
  # glagola – signalizira radnju
  # imena objekta – nad kojim se radnja obavlja

b) Varijable i argumenti:
- ime varijable mora u potpunosti i točno opisivati entitet kojeg varijabla predstavlja
- dobro ime varijable opisuje problem (što), a ne njegovo rješenje (kako)

c) Konstante
- pišu se velikim slovima
- riječi su odvojene donjom crticom ('_')

3)

Kohezija zadanog odsječka je logička. Postoji kada se nekoliko zadataka ugura u jedan te isti postupak, te se parametrom postupka odabire koji se od njih izvršava. U konkretnom slučaju pretvorba milja u kilometre i obrnute se računa unutar istog postupka, a odabir zadatka izvršavanja odvija se mijenjanjem parametra „c".

Rješenje postizanja funkcionalne kohezije

```
const float COEF_MILE_TO_KM = 1.609344f;
const float COEF_KM_TO_MILE = 0.621371f;

float ConvertMileToKM ( float mile )
{

     if ( mile < 0.0f )
     {
          return 0.0f;
     }

     return mile * COEF_MILE_TO_KM;
}

float ConvertKmToMile ( float km )
{

     if ( km < 0.0f )
     {
          return 0.0f;
     }

     return km * COEF_KM_TO_MILE;
}
```

4)

„foreach" petlja služi za obavljanje operacija nad svakim elementom polja ili klase (eng. container class)
Korisnika lišava brige o inicijalizaciji i napredovanju petlje.

Koristi se u: C#, VB, Phyton, Java, ...

5)

Polimorfizam se odnosi slična ponašanja različitih klasa, pri tome svaka klasa za sebe precizno definira zajedničko ponašanje.

Primjer: klasa televizor, DVD dijele zajedničko ponašanje upali (osnovna klasa elektronički uređaj), no svaka klasa posebno za sebe definira proces koji se odvija prilikom paljenja uređaja.

Primjer #2: ptica i avion dijele ponašanje Leti() no ptica leti pomoću mahanja krila, a avion pomoću motora.

6)

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Text;
```

```
namespace TrokutConsole
{
    /// <summary>
    /// Razred koji prikazuje trokut. Sadrzi koordinate svojih
triju
    /// tocaka i ima javna svojstva koja izracunavaju duljine
stranica,
    /// opseg i povrsinu.
    /// </summary>
         public class Trokut
         {
             private Point tockaA;
             private Point tockaB;
             private Point tockaC;

             public Point TockaA {
                 get { return tockaA; }
                 set { tockaA = value; }
             }

             public Point TockaB {
                 get { return tockaB; }
                 set { tockaB = value; }
             }

             public Point TockaC {
                 get { return tockaC; }
                 set { tockaC = value; }
             }

             public Trokut(Point tockaA, Point tockaB, Point tockaC)
     {
                 this.tockaA = tockaA;
                 this.tockaB = tockaB;
                 this.tockaC = tockaC;
             }
             public Trokut() {
                 this.tockaA = new Point();
                 this.tockaB = new Point();
                 this.tockaC = new Point();
             }

             private double izracunajDuzinu(Point prvaTocka, Point
     drugaTocka) {
                 double razlikaX = drugaTocka.X - prvaTocka.X;
                 double razlikaY = drugaTocka.Y - prvaTocka.Y;
                 return Math.Sqrt(Math.Pow(razlikaX, 2) +
     Math.Pow(razlikaY, 2));
             }

             public double StranicaA{
                 get {
```

```
                    return izracunajDuzinu(this.tockaB,
     this.tockaC);
                }
            }

            public double StranicaB {
                get {
                    return izracunajDuzinu(this.tockaA,
     this.tockaC);
                }
            }

            public double StranicaC {
                get {
                    return izracunajDuzinu(this.tockaA,
     this.tockaB);
                }
            }

            public double Opseg {
                get {
                    return StranicaA + StranicaB + StranicaC;
                }
            }

            /// <summary>
            /// Povrsina koja se racuna Heronovom formulom.
            /// </summary>
            public double Povrsina {
                get {
                    double poluOpseg = 0.5 * (StranicaA + StranicaB
     + StranicaC);
                    return Math.Sqrt(poluOpseg * (poluOpseg -
     StranicaA) * (poluOpseg - StranicaB)
                                                * (poluOpseg -
     StranicaC));
                }
            }
        }

    /// <summary>
    /// U ovom prikazu funkcionalnosti stvaraju se 5 objekata
razreda
    /// trokut i pohranjuje se u generičku listu (tip List<>)
    /// </summary>
    class Program
    {
        private const int BR_TROKUTA = 5;

        static void Main(string[] args) {

          List<Trokut> listaTrokuta = new List<Trokut>();
          Trokut trokutListe = null;
```

```
            for(int i = 0; i < BR_TROKUTA; i++)
            {
                  trokutListe = new Trokut();
                  listaTrokuta.Add(trokutListe);
            }
          }
      }

}
```