Kolegij Računalstvo | Datum:
Vježba # 13 | Tema: Petlje i brojači
Vježbu pripremio: doc. dr. sc. Edouard Ivanjko

Upute za izradu vježbi/zadataka

Vježbe je potrebno izraditi pomoću razvojne okoline MS Visual Studio u programskom jeziku C#. Prije dolaska na vježbu potrebno je izraditi pripremu. Bez napravljene pripreme nije moguće pristupiti izradi vježbi.

Prilikom završetka izrade vježbe, nakon što je dežurni asistent sve pregledao:

1. Potrebno pokrenuti i osigurati da se izvodi bez pogrešaka, izrađene vježbe koje u sebi sadrže programske pogreške NEĆE biti priznate kao odrađene!
2. Sve napravljene projekte u razvojnoj okolini MS Visual Studio potrebno je arhivirati s nazivom u slijedećem obliku – *JMBAG_GRUPA_GGGG-MM-DD_VJEŽBE#_ZADATAK#.zip* te unijeti u sustav Merlin.

Primjer ispravnog naziva ZIP datoteke: 0135123456_A_2011-10-25_VJEZBA01.zip

Vrijeme za izradu navedenih zadataka iznosi 90 minuta.

Cilj vježbe: Upoznavanje s naredbama petlji. Rješavanje istog problema korištenjem različitih petlji. Korištenje brojača unutar petlje.

Opis vježbe

Prilikom obrade podataka u računalu je često potrebno obraditi veću skupinu podataka na isti način. Pri tome se za obradu pojedinog podatka koristi isti programski kôd, samo se mijenjaju ulazni podaci te naravno pripadni rezultat. U tu svrhu se kod izrade programa koriste tzv. petlje koje imaju mogućnost prolaska podacima i na svaki podatak iz skupa primijeniti isti programski kôd. Pri tome se kreira logički uvjet koji uvjetuje izvođenje kôda unutar petlje. Jedno izvođenje programskog kôda petlje se naziva iteracija. Pripadni programski kôd koji se izvrši pri svakoj iteraciji petlje se naziva i tijelo petlje. Implementacija petlje može se napraviti na tri različita načina: (i) petlja s ispitivanjem uvjeta na početku, (ii) petlja s ispitivanjem uvjeta na kraju, te (iii) petlja s poznatim brojem ponavljanja odnosno iteracija.

Prva vrsta petlje je petlja s ispitivanjem uvjeta na početku. Pri tome je bitno imati na umu da se prvo ispita logički uvjet, a tek se nakon toga kreće u izvođenje tijela petlje. U slučaju da logički uvjet nije ispunjen, tijelo petlje se neće nikada izvesti. Pseudokôd izgleda ovako (u zagradama je dan pripadni kôd u programskom jeziku C#):

dok je (logički_izraz) („**while** (logički_izraz)“)
niz_naredbi

Pridruženi niz naredbi izvršava se tako dugo dok je logički izraz ispunjen, odnosno dok je rezultat provjere logička jedinica. Pri tome je potrebno u tijelu petlje, odnosno u pridruženom nizu naredbi

osigurati približavanje logičkog izraza situaciji da logički uvjet više nije ispunjen kako bi petlja mogla završiti, odnosno program nastavio sa svojim izvršavanjem. U protivnom dolazi se do beskonačne petlje i program nikada ne bi mogao nastaviti s izvršavanje i eventualno završiti.

Druga vrsta petlje je petlja s ispitivanjem uvjeta na kraju. Pri tome je bitno imati na umu da će se u ovom slučaju tijelo petlje izvesti najmanje jednom. Pseudokôd sada izgleda ovako (u zagradama je dan pripadni kôd u programskom jeziku C#):

ponavljaj	(„**do**“)
niz_naredbi	
dok je (logički_izraz)	(„**while** (logički_izraz)“)

Pridruženi niz naredbi se ponovo izvršava tako dugo dok je logički izraz ispunjen uz razliku prema prethodnoj petlji da se tijelo petlje izvrši najmanje jednom. Pri tome je također potrebno u tijelu petlje odnosno u pridruženom nizu naredbi osigurati približavanje logičkog izraza situaciji da logički uvjet više nije ispunjen kako bi petlja mogla završiti, odnosno program nastavio sa svojim izvršavanjem.

Treća vrsta petlje jest petlja s poznatim brojem ponavljanja. U programskom kôdu za definiciju ove petlje je automatski sadržan dio kôda koji se odnosi na približavanje završetku petlje te provjeru završetka petlje. Pri tome se provjera uvjeta izvršavanja petlje odvija na početku tako da postoji opcija da se tijelo petlje neće izvesti ni jedanput. Pseudokôd u slučaju ove petlje izgleda ovako (u zagradama je dan pripadni kôd u programskom jeziku C#):

za i = početak **do** kraj (**korak** k)	(„**for** (i=početak; i <= kraj; i+=k)“)
niz_naredbi	

Kako je dio kôda za približavanje završetku petlje sadržan u definiciji petlje u tijelu petlje nije potrebno dodavati nikakav poseban kôd za taj dio. Bitno je primijetiti da je samo kod ove vrste petlje dio za približavanje uvjetu završetka riješen u samoj definiciji petlje čime je smanjena opasnost od kreiranja beskonačne petlje nepažnjom.

Kod obrađivanja većeg skupa podataka se često pojavljuje potreba za određivanje broja podataka određenog tipa ili broja specifičnih slučajeva koji su se pojavili tijekom obrade. U tu svrhu služe brojači odnosno varijable koje se inkrementiraju ili dekrementiraju u trenutku detekcije specifičnog slučaja. U slučaju inkrementa se brojač, odnosno pripadna kontrolna varijabla u početku inicijalizira na vrijednost 0. Konačna vrijednost brojača tada označava broj prepoznatih specifičnih slučajeva. Za slučaj primjene dekrementa se vrijednost brojača inicijalizira na neku početnu vrijednost veću od 0, a zatim se detektira trenutak kada brojač dostigne vrijednost 0.

Kao naredbe inkrementa se u programskom jeziku C# koriste sljedeće naredbe, odnosno linije kôda:

i = i + 1;	-> uvećava vrijednost varijable i za 1;
i++;	-> uvećava vrijednost varijable i za 1 (kraći zapis);
i+=k;	-> uvećava vrijednost varijable i za k (varijabla ili konstanta).

Kao naredbe dekrementa se u programskom jeziku C# koriste sljedeće naredbe, odnosno linije kôda:

i = i - 1;	-> umanjuje vrijednost varijable i za 1;
i--;	-> umanjuje vrijednost varijable i za 1 (kraći zapis);
i-=k;	-> umanjuje vrijednost varijable i za k (varijabla ili konstanta).

Priprema za vježbu

- Proučiti predavanja broj 13 i 14 vezana za korištenje te implementaciju petlji i brojača.
- Popuniti sljedeću tablicu i pokazati dežurnom asistentu na početku vježbe.
- Riješiti zadatak u nastavku (napisati programski kôd) i pokazati ga dežurnom asistentu na početku vježbe.

Napomena:

Pripremu za laboratorijsku vježbu je potrebno napisati vlastoručno na ovom papiru. Student koji prilikom ulaska u dvoranu nema napisanu pripremu nema pravo pristupa laboratoriju i smatra se da student nije odradio laboratorijsku vježbu.

Ime naredbe	Namjena naredbe	Primjer programskog kôda
while		
for		
do while		
++		
+=		
--		
-=		
%		

Program za izračun zbroja reda

Potrebno je napisati program u programskom jeziku C# koji će izračunati vrijednosti zbroja S dan funkcijom dolje. Radi testiranja programskog kôda u dolje danu tablicu predložiti tri vrijednosti *n* za testiranje te ručno izračunati vrijednost zbroja *S*. Prilikom zadavanja vrijednosti za testiranje potrebno je obratiti pažnju da je vrijednost zbroja *S* moguće lako izračunati.

$$S = \sum_{i=1}^{n} \frac{i^2 + 3}{i + 1}$$

Programski kôd

Iz kojeg Ste razloga za rješavanje pripreme upotrijebili petlju? Objasnite odabir vrste petlje!

Vrijednost za testiranje	Iznos zbroja *S*

Ime i prezime studenta:

Potpis studenta

Potpis dežurnog asistenta

Rad na vježbi

Rad na vježbi sastoji se od unosa danog programskog kôda koji rješava isti problem na tri različita načina korištenjem navedene tri različite petlje.

Program dan na slici 13.1 rješava problem računanja broja i sume svih brojeva djeljivih s brojem 3 u zadanom intervalu [m, n]. Pri tome se vrijednosti granice intervala, varijable m i n, zadaju preko tipkovnice. Isti problem se rješava korištenjem petlje while, petlje do-while te petljom for. Pri tome je potrebno u program ugraditi zaštitu od krivog unosa granica, odnosno ispravak slučaja da je korisnik unio vrijednost varijable n koja je manja od vrijednosti varijable m.

Korištenje petlje while

Za testiranje rješenja pomoću „**while**“ petlje potrebno je u razvojnoj okolini MS Visual Studio definirati novi projekt za izradu aplikacije sa tekstualnim sučeljem imena „**DijeljenjeWhile**“. U glavnu metodu potrebno je zatim unijeti programski kôd prikazan na slici 13.1. Nakon provjere ispravnosti programskog kôda i kreiranja izvršne datoteke potrebno je pokrenuti program s proizvoljno odabranim testnim podacima te ispuniti tablicu 13.1. Pri tome je kod zadavanja testnih ulaznih podataka potrebno ispitati oba slučaja: ispravan unos (**m < n**) i neispravan unos (**m > n**) granica intervala. Rezultat izvršavanja je potrebno prvo izračunati ručno te usporediti s rezultatom programa.

Slučaj	Granice intervala		Varijabla „suma“		Varijabla „brojač“	
	m	n	Ručni izračun	Program	Ručni izračun	Program
m < n						
n < m						

Tablica 13.1 Testni podaci i dobiveni rezultati za slučaj petlje „**while**“

```
// deklaracija i inicijalizacija varijabli
int suma = 0, brojac = 0;
Console.Write("Unesi m: ");
int m = Convert.ToInt32(Console.ReadLine());
Console.Write("Unesi n: ");
int n = Convert.ToInt32(Console.ReadLine());

// provjera ispravnosti unosa granica
if (m > n)
{
    int x = m;
    m = n;
    n = x;
}

// obrada u while petlji
int i = m;
while (i <= n)
{
    if (i % 3 == 0)
    {
        brojac++;
        suma += i;
    }
    i++;
}

// ispis rezultata
Console.WriteLine("Suma iznosi: " + suma);
Console.WriteLine("U intervalu ima " + brojac + " brojeva djeljivih s  3.");

// da se vidi ispis konzole
Console.ReadLine();
```

Slika 13.1 Programski kôd za rješenje pomoću „**while**“ petlje

Korištenje petlje do-while

Za testiranje rješenja pomoću petlje „**do-while**“ potrebno je prvo zatvoriti projekt iz prethodnog dijela vježbe te definirati novi projekt za izradu aplikacije s tekstualnim sučeljem imena

„**DijeljenjeDoWhile**“. U glavnu metodu potrebno je zatim unijeti programski kôd prikazan na slici 13.2. Nakon provjere ispravnosti programskog kôda i kreiranja izvršne datoteke potrebno je pokrenuti program s odabranim testnim podacima te ispuniti tablicu 13.2. Za testiranje je potrebno koristiti iste testne podatke kao i prethodnom slučaju te provjeriti jeli se rezultati slažu s vrijednostima dobivenim ručnim izračunom, odnosno rezultatima programa iz prethodnog dijela vježbe.

Slučaj	Varijabla „suma“	Varijabla „brojač“
	Program	Program
m < n		
n < m		

Tablica 13.2 Testni podaci i dobiveni rezultati za slučaj petlje „**do-while**“

```
// deklaracija i inicijalizacija varijabli
int suma = 0, brojac = 0;
Console.Write("Unesi m: ");
int m = Convert.ToInt32(Console.ReadLine());
Console.Write("Unesi n: ");
int n = Convert.ToInt32(Console.ReadLine());

// provjera ispravnosti unosa granica
if (m > n)
{
    int x = m;
    m = n;
    n = x;
}

// obrada u while petlji
int i = m;
do
{
    if (i % 3 == 0)
    {
        brojac++;
        suma += i;
    }
    i++;
} while (i <= n) ;

// ispis rezultata
Console.WriteLine("Suma iznosi: " + suma);
Console.WriteLine("U intervalu ima " + brojac + " brojeva dijeljivih s  3.");

// da se vidi ispis konzole
Console.ReadLine();
```

Slika 13.2 Programski kôd za rješenje pomoću petlje „**do-while**“

Korištenje petlje for

Za testiranje rješenja pomoću petlje „**for**“ potrebno je prvo zatvoriti projekt iz prethodnog dijela vježbe te definirati novi projekt za izradu aplikacije s tekstualnim sučeljem imena „**DijeljenjeFor**“. U glavnu metodu potrebno je zatim unijeti programski kôd prikazan na slici 13.3. Nakon provjere ispravnosti programskog kôda i kreiranja izvršne datoteke potrebno je pokrenuti program s odabranim testnim podacima te ispuniti tablicu 13.3. Za testiranje je potrebno koristiti iste testne podatke kao i u prethodnom dijelu vježbe te provjeriti jeli se rezultati slažu s vrijednostima dobivenim ručnim izračunom, odnosno rezultatima programa iz prethodna dva dijela vježbe.

Slučaj	Varijabla „suma“	Varijabla „brojač“
	Program	Program
m < n		
n < m		

Tablica 13.3 Testni podaci i dobiveni rezultati za slučaj petlje „**for**“

```
// deklaracija i inicijalizacija varijabli
int suma = 0, brojac = 0;
Console.Write("Unesi m: ");
int m = Convert.ToInt32(Console.ReadLine());
Console.Write("Unesi n: ");
int n = Convert.ToInt32(Console.ReadLine());

// provjera ispravnosti unosa granica
if (m > n)
{
    int x = m;
    m = n;
    n = x;
}

// obrada u while petlji
for(int i = m;i<=n;i++)
{
    if (i % 3 == 0)
    {
        brojac++;
        suma += i;
    }
 }

// ispis rezultata
Console.WriteLine("Suma iznosi: " + suma);
Console.WriteLine("U intervalu ima " + brojac + " brojeva dijeljivih s  3.");

// da se vidi ispis konzole
Console.ReadLine();
```

Slika 13.3 Programski kôd za rješenje pomoću „**for**“ petlje

Koje su razlike u implementaciji programskog kôda rješenja istog problema korištenjem različitih petlji?

Odgovor:

Testiranje vlastitog programskog kôda

Program napisan u pripremi potrebno je iskoristiti za kreiranje novog projekta u programskom alatu MS Visual Studio pod imenom „IzracunZbroja“. Kao ulazne vrijednosti za testiranje programa iskoristiti vrijednosti predložene u pripremi. Rezultat izvođenja programa unijeti u tablicu dolje, usporediti s ručno izračunatim vrijednostima te pokazati dežurnom asistentu.

Vrijednost za testiranje	Iznos zbroja *S*	Vrijednost iz programa

Tablica 13.4 Rezultat izvršavanja programa iz pripreme

Napomena

Sve rezultate laboratorijske vježbe potrebno je na kraju laboratorijske vježbe pokazati dežurnom asistentu. U slučaju da student nije pokazao rezultate laboratorijske vježbe smatra se da istu nije odradio te ju je dužan nadoknaditi. Napravljene projekte u razvojnoj okolini MS Visual Studio te slike ekrana s rezultatima izvršavanja programa potrebno je spremiti u obliku arhive (ZIP datoteke) te spremiti u sustav Merlin.

Potpis studenta Potpis dežurnog asistenta

Korisne napomene za rad na vježbi

- Vitičaste zagrade se dobivaju na HR tipkovnici kombinacijama tipki „Alt Gr“ + „b“ (lijeva vitičasta zagrada -> „**{**“) te „Alt Gr“ + „n“ (desna vitičasta zagrada -> „**}**“).
- Uglate zagrade se dobivaju na HR tipkovnici kombinacijama tipki „Alt Gr“ + „f“ (lijeva uglata zagrada -> „**[**“) te „Alt Gr“ + „g“ (desna uglata zagrada -> „**]**“).
- Tipkom „Delete“ ili „Del“ se briše znak ispred kursora.
- Tipkom „Backspace“ se briše znak iza kursora.
- Tipkom „Ins“ ili „Insert“ se aktivira opcija pisanja preko postojećeg teksta tako da novi znak prebriše postojeći znak. Aktivira se pritiskom na tipku „Ins“ ili „Insert“, a deaktivira ponovnim pritiskom.
- Potvrda unosa podatka u tekstualnom sučelje je pritisak na tipku „Enter“.
- Znak „**|**“ se dobiva na HR tipkovnici kombinacijom tipki „Alt Gr“ + „w“.
- Znakovi manje i veće se dobivaju na hrvatskoj tipkovnici kombinacijama tipki „Alt Gr“ + „**,**“ za znak „**<**“ te „Alt Gr“ + „**.**“ za znak „**>**“.