Uporaba c/c++ programskih jezika

Cvitković, Vinko

Undergraduate thesis / Završni rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:875165

Rights / Prava: Attribution-ShareAlike 3.0 Unported/Imenovanje-Dijeli pod istim uvjetima 3.0

Download date / Datum preuzimanja: **2024-03-29**

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Vinko Cvitković

UPORABA C/C++ PROGRAMSKIH JEZIKA

ZAVRŠNI RAD

Varaždin, 2019.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Vinko Cvitković

Matični broj: 43974/15–R

Studij: Informacijski sustavi

UPORABA C/C++ PROGRAMSKIH JEZIKA

ZAVRŠNI RAD

Mentor:

Prof. dr. sc. Alen Lovrenčić

Varaždin, rujan 2019.

Vinko Cvitković

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Tema rada je uporaba C/C++ programskih jezika. Glavni fokus biti će na jeziku C++. Počevši sa poviješću jezika, opisano je obrađivanje programskog jezika C/C++ te rad u njima uz primjere programskih kodova. Navedene su i opisane prednosti C++ jezika te su detaljno obrađene osnove programiranja u C++ programskom jeziku. U radu su obrađena 2 zadatka u C++, te također u .NET C#. Uspoređena je učinkovitost algoritama u C++ i .NET C# kroz zauzimanje memorije i količinu napisanog programskog koda potrebnog za izvršenje zadatka.

Ključne riječi: Programski jezici**,** C**,** C++, sortiranje, C#, algoritam

Sadržaj

Sadržaj iii
1. Uvod 1
2. Rješavanje problema 2
2.1. Specificiranje zahtjeva programa 2
2.2. Specificiranje strukture programa 2
2.3. Odabir i pisanje algoritma 2
2.4. Pisanje programskog koda 3
2.5. Otklanjanje pogrešaka 3
2.6. Ispitivanje programskog koda 3
2.7. Održavanje programa 3
3. Programiranje u C i C++ programskim jezicima 4
3.1. Hello World 4
3.2. Tipovi podataka 5
3.2.1. Imenovanje varijabli 6
3.2.2. Deklaracija varijable 6
3.3. Selekcija 7
3.3.1. Nareba if 7
3.3.2. Naredba switch 9
3.4. Iteracija 10
3.4.1. Naredba for 11
3.4.2. Naredba while 12
3.5. Znakovi 14
3.6. Polja 17
3.6.1. Pretraživanje polja 18
3.6.2. Sortiranje polja 19
3.6.2.1. Sortiranje izborom 20
3.6.2.2. Sortiranje zamjenom 21
3.6.2.3. Mjehurićasto sortiranje 21
3.7. Funkcije 23
3.7.1. Poziv funkcije 23
3.7.2. Definicija funkcije 23
3.8. Pokazivači 25
4. Usporedba C++ i C# .NET programskih jezika 26
4.1. Zadatak 1 26
4.2. Zadatak 2 29

5. Zaključak 33
Popis literature 34
Popis slika 35
Popis tablica 36

1. Uvod

U današnjem svijetu uporaba računala i pametnih uređaja smatra se neizostavnom. Postali smo gotovo obavezni posjedovati pametni telefon. Gdje se god okrenemo nalazimo potrebu imati pametni telefon, krenemo li u banku, nude nam sustav mobilnog bankarstva, odemo li u trgovinu, nude nam skupljanje bodova putem aplikacije, postalo je gotovo obavezno posjedovati pametni telefon. U svakom kućanstvu postoji barem jedno računalo, bilo to stolno ili prijenosno, mnoge je fakultete nemoguće pohađati bez posjedovanja računala.

Računala, pametne telefone i sve ostale pametne uređaje napravio je, barem nekim dijelom, programer na svome računalu. Mobilne aplikacije, Internet stranice novina, društvenih mreža, oglasnika, Internet trgovine i ostale usluge na računalima i pametnim telefonima koje koristimo također su zasnovane na programiranju i izrađene u nekom programskom jeziku. Sam Microsoftov Windows napravljen je u programskim jezicima i to baš u onome koji je glavna tema ovog završnog rada, programski jezik C++.

Programski jezici C i C++ su jedni od najpoznatijih programskih jezika. Obrađuju se u mnogim srednjim školama jer su dobar temelj za ulazak u svijet programiranja. Specifični su po brzini izvođenja i minimalnoj količini memorije koju kodovi napisani u tim jezicima zauzimaju.

2. Rješavanje problema

Prije nego li započnemo s pisanjem problema, potrebno ga je dobro razumjeti, kao i osmisliti postupak njegovog rješavanja. Dobrom praksom se pokazalo podijeliti rad oko pisanja programa u nekoliko koraka:

1. Specificiranje zahtjeva programa
2. Specificiranje strukture programa
3. Odabir i pisanje algoritma
4. Pisanje programskog koda
5. Otklanjanje pogrešaka
6. Ispitivanje programa
7. Održavanje programa

2.1. Specificiranje zahtjeva programa

Kako bi se proizveo kvalitetan softverski proizvod važno ga je razumjeti prije no što rad na njemu započne. Kako bi se taj cilj ostvario potrebno je napisati detaljnu specifikaciju koja opisuje kako se program mora ponašati, odnosno, kako će ga vidjeti korisnik.

2.2. Specificiranje strukture programa

Specificiranje strukture programa uključuje raščlanjivanje složenog problema na jednostavnije pod probleme, specifikacija ulaznih/izlaznih podataka te definiranje postupa obrade ulaznih podataka. Pod problemi se u fazi pisanja koda realiziraju kao programske funkcije, o kojima će biti riječ nešto kasnije. Rješavanje problema računalom uključuje obradu podataka, odnosno izvođenje raznih operacija nad podacima, npr. množenje, zbrajanje, oduzimanje brojeva itd. Korisnik postavlja podatke koje računalom obrađujemo te se ti podaci zovu ulazni podaci. Tokom obrade podataka koriste se razne formule te se završetkom obrade podataka dobivaju izlazni podaci, odnosno rezultati.

2.3. Odabir i pisanje algoritma

U ovoj se fazi osmišljavaju mogući načini rješavanja problema. Odabiru se najbolji načini rješavanja te se detaljno opisuju. Detaljan opis načina rješavanja problema naziva se algoritam.

2.4. Pisanje programskog koda

Potrebno je odabrani programski jezik za pisanje programskog koda koji će realizirati algoritam. Moramo dobro poznavati pravila za pisanje naredbi, odnosno sintaksu odabranog programskog jezika kako bi pisanje programskog koda bilo izvedivo.

2.5. Otklanjanje pogrešaka

Kada je program napisan, postoji mogućnost da pokretanje izvršavanja programa bude onemogućeno. Najčešći uzrok tome jest postojanje sintaktički pogrešaka što znači da naredbe programskog koda nisu napisane prema pravilima koja definira određeni programski jezik.

2.6. Ispitivanje programskog koda

Nakon što smo napisali programski kod i uklonili sve sintaktičke pogreške potrebno je ispitati radi li programski kod ispravno. Potrebno je pripremiti dovoljno velik skup ulaznih podataka, za koje je poznat skup izlaznih vrijednosti kako bismo ispitali programski kod. Ustanovljujemo radi li programski kod ispravno.

2.7. Održavanje programa

Kako bi program bio aktualan, konstantno se treba prilagođavati zahtjevima tržišta. Program koji pomaže pri obavljanju važnih poslova zahtjeva veliku brigu. Održavanje programa zahtijeva poznavanje naknadno uočenih pogrešaka te sukladno tome prilagođavanje programa promjenama početnih zahtjeva. Ukoliko se radi o jednostavnijim programima koji se kreiraju u svrhu rješavanja problema, uspješno provedeno ispitivanje ujedno označava završetak upotrebe programa te nema potrebe za naknadnim unaprjeđivanjem programskog koda.

3. Programiranje u C i C++ programskim jezicima

C i C++ programski jezici jedni su od najpopularnijih i najpraktičnijih za početak učenja programiranja. U mnogim srednjim školama i na ranijim godinama studija često se obrađuju baš C i C++. Kao student informatike, na prvoj i drugoj godini studija učio sam baš C++ programski jezik s ciljem shvaćanja objektno orijentiranog programiranja.

3.1. Hello World

Na osnovu određenog algoritma slijedi pisanje programskog koda u željenom programskom jeziku. Ovaj rad odnosi se na programske jezike C i C++ s naglaskom na C++ pa će u nastavku biti obrađivani konkretno jezici C i C++.

Razvoj softverskog proizvoda zahtijeva sljedeće alate: Editor – program za obradu teksta, Prevoditelj (eng. *Compiler*) – tekst programskog koda prevodi u zapis razumljiv računalu te na posljetku program za pronalaženje pogrešaka. Navedeni alati su dostupni u jednoj aplikaciji te čine integrirano razvojno okruženje, a neke od takvih aplikacija su: Microsoft Visual Studio, Microsoft Visual C++ Express, Jet Brains IntelliJ itd.

Primjer „Hello World“ programa u programskom jeziku C:

```
#include<stdio.h>
int main()
{
	printf("Hello World");
	return 0;
}
```

Primjer „Hello World“ programa u programskom jeziku C++:

```
#include<iostream>
using namespace std;
int main()
{
	cout << "Hello World";
	return 0;}
```

C/C++ programski kod čine logički i funkcionalno usklađene naredbe i funkcije. Program mora sadržavati minimalno jednu funkciju, a to je funkcija `main()`. Svaka funkcija je karakterizirana nazivom i okruglim zagradama nakon naziva. Ključna riječ `int` ispred naziva funkcije označava tip rezultata kojeg ta funkcija vraća, odnosno izračunava. Unutar zagrada navode se podaci koje ta funkcija prima u inicijalizaciji same funkcije. Otvorena i zatvorena vitičasta zagrada označavaju gdje počinju i gdje završavaju naredbe funkcije. Programski jezici C i C++ omogućavaju upotrebu već gotovih funkcija iz biblioteke standardnih funkcija, a to su: funkcije za učitavanje vrijednosti podataka, za ispis tekstna na ekranu itd. Kada želimo pozvati funkciju kojom ćemo izvršiti pod zadatak, pozivamo ju po iznad objašnjenoj sintaksi (ime_funkcije(podaci_za_obradu)). U programskom kodu napisanom u jeziku C koristi se funkcija `printf()` za ispis teksta na ekranu, dok se u programskom jeziku C++ uobičajeno koristi naredba `cout` definirana u datoteci `iostream` koju uključujemo sa `#include<iostream>` . Operator << (znak „manje od“ napisan dva puta za redom) upućuje podatak u nastavku na izlazni tok te se on ispisuje na zaslonu računala. Programski jezik C koristi datoteku `stdio.h` koja sadrži funkciju `printf()` te se ona uključuje na isti način kao datoteke u jeziku C++. Prevoditelju su potrebni podaci o funkcijama kako ti tekst programskog koda mogao prevesti u binarni oblik, koji je razumljiv računalu. Podaci o funkcijama iz biblioteke standardnih funkcija nalaze se u raznim datotekama te ih se u kod uključuje ovisno o potrebama zadatka koji ta funkcija izvršava.

`Return 0` je naredba kojom glavni program vraća broj nula te se tom naredbom označava da je program uspješno završio s izvršavanjem.

3.2. Tipovi podataka

Svi podaci koji se obrađuju na računalu i podaci koji su izračunati od strane računala pohranjuju se u radnoj memoriji računala. Elementarne tipove podataka dijelimo na cjelobrojne tipove podataka (char, short int, int, long int, long long int) i decimalne tipove podataka (float, double, long double).

Bitno je spomenuti varijablu koja služi za prikaz logičkih tipova podataka, a to je varijabla bool. Varijabla bool može poprimiti samo dvije vrijednosti, a to su `true` koji označava istinu i `false` koji označava neistinu. Bool je varijabla koja postoji u C++ programskom jeziku, no u programskom jeziku C ne postoji. Bool se često koristi kod rada sa selekcijama koje ćemo obraditi u nastavku rada.

Tablica 1. Elementarni tipovi podataka

tip konstante	bajtova RAM memorije	raspon vrijednosti	točnost
char	1	-128 do 127 Ili 0 do 255 (ovisno o prevoditelju C++ programskog jezika	
short int (short)	2	-32768 do 32767	
int	2 ili 4	-32768 do 32767 ili -2147483648 do 2147483648	
long int (long)	4	-2147483648 do 2147483648	
long long int (long long)	8	-9223372036854775808 do 9223372036854775807	
float	4	$-3{,}40 \cdot 10^{38}$ do $-1{,}17 \cdot 10^{-38}$ i $1{,}17 \cdot 10^{-38}$ do $3{,}40 \cdot 10^{38}$	7 dekadskih. znamenki
double	8	$-1{,}79 \cdot 10^{308}$ do $-2{,}22 \cdot 10^{-308}$ i $2{,}22 \cdot 10^{-308}$ do $1{,}79 \cdot 10^{308}$	15 dekadskih znamenki
Long double	16	$-1{,}18 \cdot 10^{4932}$ do $-3{,}36 \cdot 10^{4932}$ i $3{,}36 \cdot 10^{-4932}$ do $1{,}18 \cdot 10^{4932}$	34 dekadske znamenke

(Prema: Šribar i Motik, 2018)

Željene podatke i dobivene rezultate upisujemo u varijable, varijablama se za vrijeme izvršavanja programa može više puta pristupiti ili mijenjati sadržaj. Kako bismo se bolje organizirali u pisanju programskog koda i kako bi programski kod bio razumljiviji, programski jezici omogućuju da se varijablama daju imena.

3.2.1. Imenovanje varijabli

Kako u svemu, tako i u imenovanju varijabli postoje određena pravila kojih se moramo pridržavati:

1. Koriste se isključivo slova engleske abecede (slova č, ć, š, đ i ž ne mogu se koristiti u imenovanju varijabli)
2. Prvi znak u imenu može biti slovo ili znak _ dok se od drugog znaka nadalje mogu nalaziti slova, brojevi i samo jedan znak: znak _
3. U imenu varijable ne smiju se nalaziti razmaci, a ukoliko želimo imati više riječi u nazivu varijable uobičajeno u programiranju, umjesto razmakom, riječi se odvoje donjom povlakom (eng. *underscore*) (npr. ime_varijable) ili prvo slovo svake riječi pišemo veliko (npr. imeVarijable)
4. Ime varijable mora se razlikovati od imena ključnih riječi u programskom jeziku

3.2.2. Deklaracija varijable

Prije nego upišemo podatak u memoriju, potrebno je rezervirati, odnosno pripremiti dovoljno velik prostor u memoriji gdje ćemo spremiti taj podatak. Taj postupak, kojim rezerviramo memoriju za varijablu i dodjeljujemo toj varijabli ime nazivamo deklaracija

varijable. Postupak deklaracije varijable odrađujemo ključnim riječima, a za ranije navedene osnovne tipove podataka te ključne riječi su: int, float, double, char. Moguće je deklarirati više varijabli istog tipa u jednom redu programskog koda npr. (`double prviBroj, drugiBroj, treciBroj;`)

3.3. Selekcija

U pisanju programskog koda nailazimo na dio gdje moramo, ovisno o nekom uvjetu, odraditi određeni blok naredbi, tada dolazi do potrebe za grananjem. Grananje, odnosno sekvenciju u programskom kodu moguće je ostvariti pomoću jedne od sljedeće dvije naredbe: naredba switch i naredba if. (Šribar i Motik, 2003. Demistificirani C++)

3.3.1. Nareba if

Naredbu za grananje if u programskom kodu ostvarujemo ključnom riječi „`if`“. Ukoliko je vrijednost iza naredbe if istinita blok naredbi koje slijede se izvršava. U slučaju kada vrijednost nije istinita blok naredbe se preskače i izvođenje se nastavlja od prve naredbe iza bloka. (Šribar i Motik, 2003. Demistificirani C++)

Želimo li u programsku kodu dodati blok naredbi koji će se izvršiti kada uvjet iza naredbe if nije ispunjen, koristimo „`else`“. Else je neobavezna klauzula u naredbi if i za razliku od naredbe if nema uvjet u okruglim zagradama nego se izvršava kada vrijednost iza riječi `if` nije istina.

Blokove if naredbi možemo nadovezivati, ugnježđivati jedan unutar drugoga. Takav primjer vidjet ćemo u nastavku ovog poglavlja.

Za zapisivanje uvjeta kod naredbe if, koristimo operatore veće od „>“, manje od „<“, veće ili jednako „>=“, maje ili jednako „<=“, jednako „==“, različito „!=“ te pomoću tih operatora tvorimo jednostavne logičke izraze. Logički izrazi se mogu koristiti i kao izrazi za računanje vrijednosti varijabli, a koristimo ih i u iteracijama for i while koje ćemo obraditi u sljedećim poglavljima.

Ukoliko postoje dva ili više uvjeta ispitujemo ih naredbom if uz korištenje složenijih logičkih izraza koji se tvore od više izraza korištenjem logičke operacije I čiji je simbol „&&“ koja kao rezultat daje istinu samo u slučaju da su sve tvrdnje istinite, te logička operacija ILI čiji je simbol || koja kao rezultat daje istinu kada je minimalno jedan od uvjeta istinit. Uz I i ILI logičke operacije koristi se i logička operacija NE čiji je simbol „!“ te kao rezultat daje suprotan

rezultat. Ukoliko, kada je rezultat `true` ispred izraza stavimo logički operator NE, kao rezultat dobivamo `false` i suprotno.

Primjer zadatka: Treba izračunati površinu pravokutnog trokuta sa katetama duljine a=3, b=4. Ako je površina jednaka 4 neka se ispiše duljina hipotenuze, ako je površina jednaka 6 neka se ispiše opseg trokuta, a ako je rezultat 5 neka se ispiše taj rezultat. Ukoliko niti jedan od navedenih rezultata nije točan, neka se ispiše poruka „Pozdrav!“.

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    double a = 3, b = 4, c, povrsina;
    povrsina = (a*b)/2;
    c = sqrt((a*a)+(b*b));
    if(povrsina == 4)
    {
        cout << c;
    }
    else if (povrsina == 5)
    {
        cout << povrsina;
    }
    else if (povrsina == 6)
    {
        int opseg = a + b + c;
        cout << opseg;
    }
    else
    {
        cout << "Pozdrav";
```

```
        }
    }
```

3.3.2. Naredba switch

Naredbu za grananje switch u programskom kodu ostvarujemo ključnom riječi „`switch`“. Koristimo ju uglavnom kada se korisniku nudi izbor jedne od nekoliko mogućnosti programa. Ovisno o ulaznom rezultatu naredba switch izvršava određeni niz naredbi koji je definiran za taj odabrani slučaj.

Naredbu `break` koristimo na kraju slučaja kako bi program nakon izvršavanja željenog slučaja završio s blokom `switch` te kako bi se izbjeglo ne željeno ulaženje u još neki slučaj u bloku.

Uz sve potrebne slučajeve postoji i slučaj koji označavamo sa `default` te se u njemu navode naredbe koje će se izvršiti ako nije odabran niti jedan od ponuđenih slučajeva.

Primjer zadatka: Potrebno je unijeti 2 broja te odabrati jednu od ponuđenih mogućnosti:

1. Zbroj
2. Razlika
3. Umnožak
4. Kvocijent

Program mora obraditi odabranu aritmetičku operaciju s unesenim brojevima.

```
#include<iostream>
#include<cmath>
using namespace std;


int main()
{
    double a,b, rezultat;
    int izbor;
    cout << "Unesite prvi broj: ";
    cin >> a;
    cout << "Unesite drugi broj: ";
```

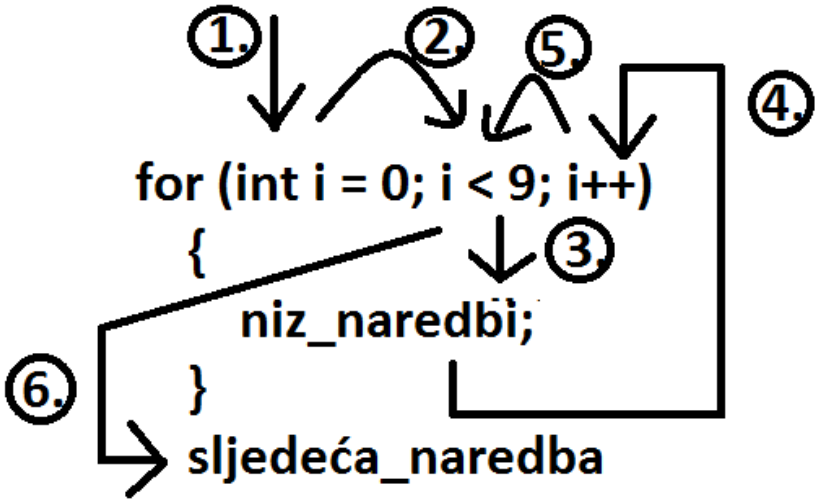
```
    cin >> b;

    cout << "1. Zbroj" << endl << "2. Razlika" << endl << "3. Umnožak" <<
endl << "4. Kvocjent" << endl << "Izbor: ";

    cin >> izbor;

    switch (izbor)

    {

    case 1:

        rezultat = a + b;

        break;

    case 2:

        rezultat = a - b;

        break;

    case 3:

        rezultat = a * b;

        break;

    case 4:

        rezultat = a / b;

        break;

    default:

        break;

    }

    cout << "Rezultat iznosi " << rezultat << endl;

    system("pause");

    return 0;

}
```

3.4. Iteracija

U programiranju često treba uzastopno, nekoliko puta određeni dio koda ponoviti. Na primjer u zadatku moramo ispisati niz od više brojeva, postoji nekoliko načina da se izvede rješenje tog zadatka, a ti načini su korištenje naredbe for, naredbe skoka ili naredbe switch.(Lovrenčić i Konecki, 2017. Programiranje u 14 lekcija)

3.4.1.Naredba for

Kada u pisanju programskog koda želimo određenu naredbu ili niz naredbi ponoviti unaprijed poznati broj puta koristimo naredbu for.



Slika 1Slijed izvršavanja naredbe for

Detaljan opis koraka sa slike:

1. Deklaracija varijable „i" te postavljanje iste kao brojilo izvršavanja u početnu vrijednost.
2. Ispitivanje uvjeta, ukoliko je uvjet zadovoljen odlazi se na korak 3. ukoliko uvjet nije zadovoljen odlazi se na korak 6.
3. Uvjet je zadovoljen. Izvršava se niz naredbi unutar naredbe for.
4. Izvršava se naredba koja se u naredbi for nalazi iza uvjeta
5. Prelazi se ponovno na ispitivanje uvjeta, ukoliko je uvjet zadovoljen odlazi se na korak 3. ukoliko uvjet nije zadovoljen odlazi se na korak 6.
6. Uvjet nije zadovoljen. Izvršavanje naredbe for se prekida te se nastavlja izvršavanje sljedećih naredbi.

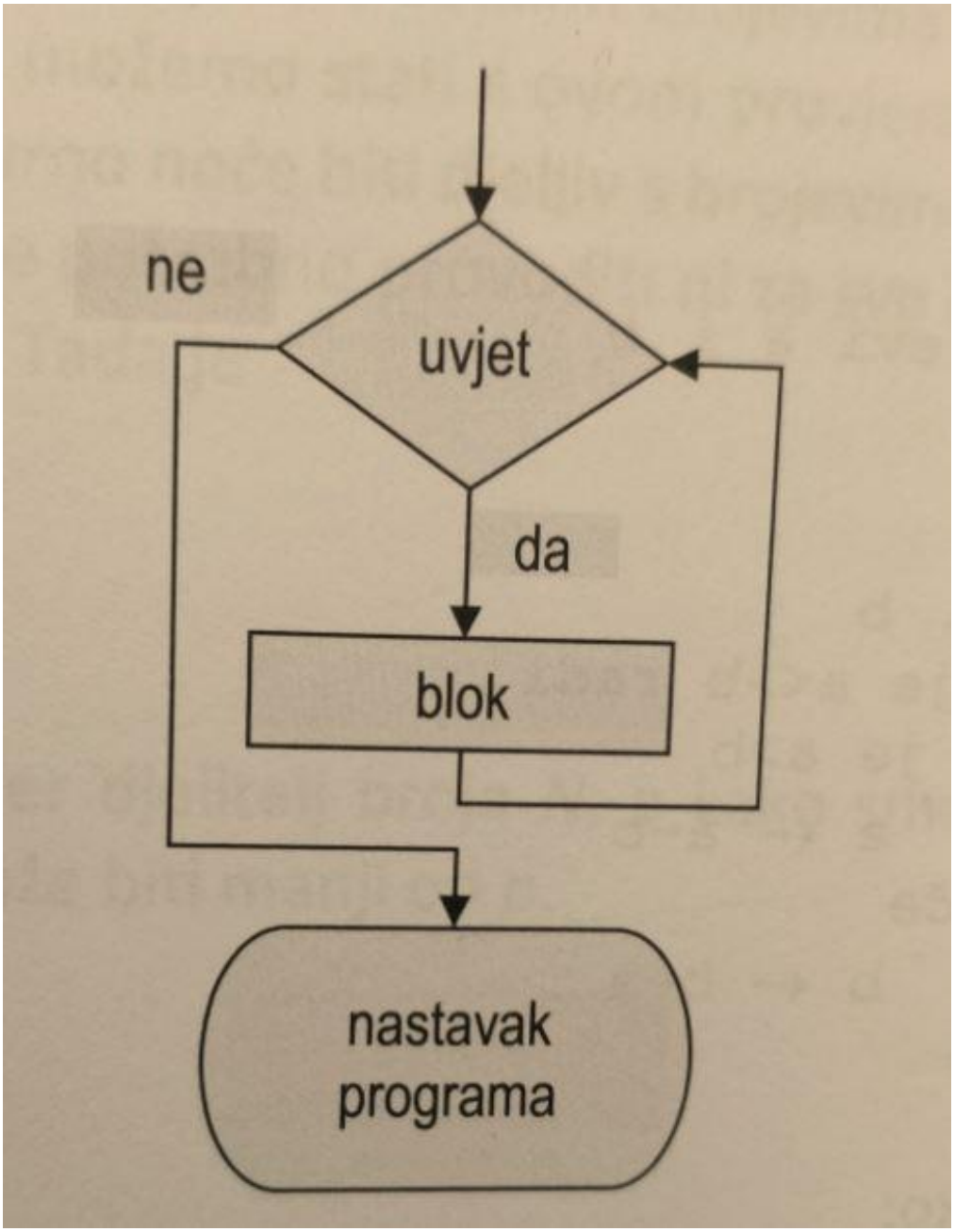
Primjer zadatka: Unesite n brojeva i ispišite najveći broj.

```
#include <iostream>
```

```
using namespace std;

int main()

{

	int n;

	int max;

	int broj;

	cout << "Koliko brojeva zelite unijeti?";

	cin >> n;

	cout << "Unesite 1. broj: ";

	cin >> max;

	for (int i = 1; i < n; i++)

	{

		cout << "Unesite " << (i + 1) << ". broj: ";

		cin >> broj;

		if (broj > max)

		{

			max = broj;

		}

	}

	cout << "Najveci od unesenih brojeva je " << max << endl;

	system("pause");

	return 0;
```

3.4.2. Naredba while

Dok nam naredba for omogućava ponavljanje niza naredbi unaprijed poznati broj puta, naredba while koristi se nešto drugačije. Naredbu while upotrebljavamo kada želimo da se određeni niz naredbi ponavlja dok je ispunjen zadani uvjet, broj ponavljanja nije poznat unaprijed.



Slika 2 Slijed izvršavanja naredbe while (Lovrenčić A., Konecki M.,2017.)

Primjer zadatka: Unositi jednoznamenkaste, dvoznamenkaste i troznamenkaste brojeve dok se upisuju prirodni brojevi. Ispisati koliko je uneseno jednoznamenkastih, koliko dvoznamenkastih i koliko troznamenkastih brojeva.

```
#include<iostream>

using namespace std;

int main()

{
```

```
	int jednoznam=0;
	int dvoznam=0;
	int troznam=0;
	int broj;
	cout << "Unesite prvi broj: ";
	cin >> broj;
	while ((broj > 0) && (broj = (int)broj) && (broj<1000) ){
		if (broj<10)
		{
			jednoznam++;
		}
		else if (broj>=10 && broj <100)
		{
			dvoznam++;
		}
		else
		{
			troznam++;
		}
		cout << "Unesite sljedeći broj: ";
		cin >> broj;
	}
	cout << "Uneseno je " << jednoznam << " jednoznamenkastih, " <<
dvoznam << " dvoznamenkastih i " << troznam << " troznamenkastih brojeva."
<< endl;
    return 0;
}
```

3.5. Znakovi

Znakovi su tipa char te se zapisuju kao jedan znak unutar jednostrukih navodnika ili pomoću određenog brojčanog koda.

„Zanimljivo je da se konstante i varijable tipa char mogu uspoređivati, poput brojeva pri čemu se u biti uspoređuju brojčani kodovi kojima su ti znakovi predstavljeni u aktivnom kodiranju znakova. Za slova engleske abecede, znamenke i standardne simbole interpunkcije su kodovi identični u gotovo svim kodiranjima i odgovaraju ASCII kodovima (kratica od American Standard Code for Information Interchange) – znamenkama, slovima engleske abecede i uobičajenim simbolima pridruženi su brojevi od 32 do 126, dok specijalni znakovi imaju kodove od 0 do 31 uključivo.“
(Šribar i Motik, 2018., str. 83.)

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

Slika 3 Tablica ASCII kodova(https://www.commfront.com/pages/ascii-chart, preuzeto 30.8.2019.)

Primjer zadatka: Unositi znakove dok se ne unese znamenka '0'. Usporediti koliko je uneseno velikih, a koliko malih slova.

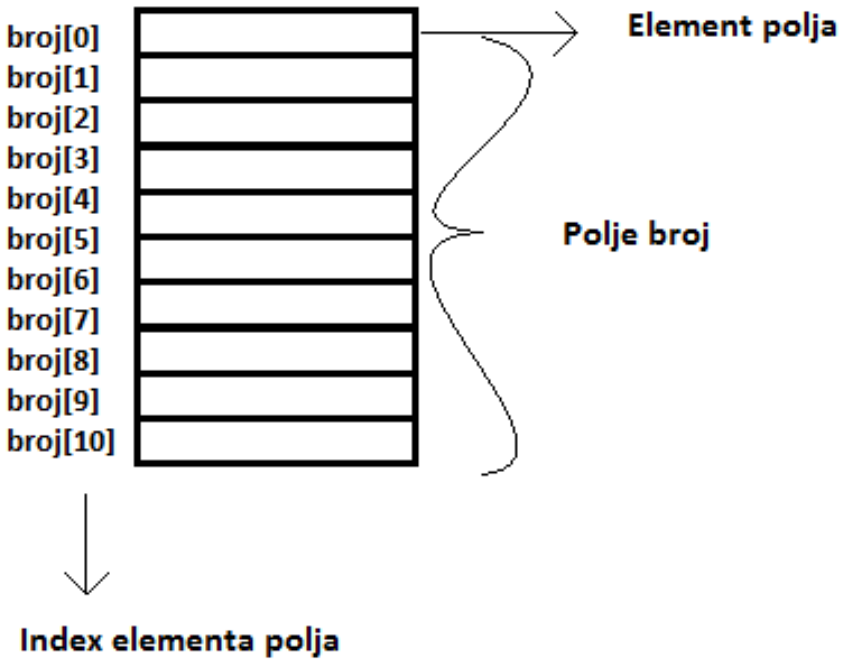
```
#include <iostream>
using namespace std;
int main()
{
	char znak;
	int maloSlovo = 0;
	int velikoSlovo = 0;
	cout << "\n Unesite prvi znak: ";
	cin >> znak;
	while (znak != '0') {
		if (znak>='a' && znak<='z')
		{
			maloSlovo++;
		}
		if (znak>='A' && znak<='Z')
		{
			velikoSlovo++;
		}
		cout << "Unesite sljedeci znak: ";
		cin >> znak;
	}
	if (maloSlovo > velikoSlovo)
	{
		cout << "Upisano je vise malih nego velikih slova!";
	}
	else if (velikoSlovo == maloSlovo)
	{
		cout << "Upisano je jednako velikih i malih slova!";
```

```
        }
        else
        {
              cout << "Upisano je vise velikih nego malih slova!";
        }
        system("pause");
      return 0;
}
```

3.6. Polja

Budući da nam elementarni tipovi podataka ne daju neke veće mogućnosti organizacije podataka, ukoliko želimo unijeti veću količinu brojeva nad kojima ćemo kasnije izvršavati operacije, koristiti ćemo polja. Polja su jedan od mehanizama agregacije te nam omogućuju povezivanje podataka jednostavnijih tipova u veće podatkovne cjeline te tako možemo efikasnije riješiti problem.(Lovrenčić A., Konecki M. 2017., „Programiranje u 14 lekcija“)

Polje ima unaprijed definirani broj varijabli. Svaka vrijednost pojedinačne varijable polja naziva se element polja te ima svoj indeks. Svakom se elementu u polju pristupa putem njegova indeksa. Prvi će element polja uvijek biti pod indeksom 0.(Lovrenčić A., Konecki M. 2017., „Programiranje u 14 lekcija“)



Slika 4 Prikaz polja s 10 elemenata

3.6.1.Pretraživanje polja

Pretraživanje polja jest postupak kojim ispitujemo postoji li zadana vrijednost negdje u zadanom polju odnosno je li ta vrijednost jednaka nekom elementu iz polja.

Ako se ispituje nalazi li se vrijednost u polju, postupak pretraživanja obavlja se dok se u polju ne nađe tražena vrijednost. Kada se ispituje koliko se puta tražena vrijednost pojavljuje u polju onda se ispitivanje vrši dok se ne dođe do kraja polja.

Primjer zadatka: Unijeti 10 brojeva u polje te zatim željeni broj. Provjeriti nalazi li se naknadno uneseni broj u polju.

```
#include<iostream>
using namespace std;
int main(){
    int polje[10];
    int postoji=false;
    int broj;
    for (int i = 0; i < 10; i++)
```

```
    {
        cout << "\n Unesite " << (i+1) << ". broj: ";
        cin >> polje[i];
    }
    cout << "Unesite broj za pretrazivanje: ";
    cin >> broj;
    for (int i = 0; i < 10; i++)
    {
        if (polje[i]==broj)
        {
            postoji = true;
            break;
        }
    }
    if (postoji)
    {
        cout << "\n Traženi broj se nalazi u polju!";
    }
    else
    {
        cout << "\n Traženi se broj ne nalazi u polju!";
    }
}
```

3.6.2.Sortiranje polja

Sortiranjem nazivamo postupak kojim slažemo brojeve po veličini, bilo to od najvećeg do najmanjeg ili suprotno. Sortiranje se također može odnositi i na mnogobrojne druge elemente, odnosno varijable. Na primjer sortirati možemo slova po abecedi, automobile po vrijednosti ili pak studente po prosjeku ocjena i tako dalje.

3.6.2.1. Sortiranje izborom

Moglo bi se reći da je sortiranje izborom (eng. *selection sort*) jedan od najjednostavnijih algoritama za sortiranje. On sortira tako da traži najveću vrijednost te ju mijenja sa vrijednošću iz zadnjeg elementa polja, zatim se postupak ponavlja, ali bez te zadnje, najveće, vrijednosti. Nakon toga na kraju polja imamo sortirane dvije najveće vrijednosti te se postupak opet ponavlja bez dva zadnja elementa i tako do kraja. Primjer sortiranja izborom prikazat ćemo zadatkom u kojem je potrebno unijeti 10 brojeva te izvršiti sortiranje izborom. (Lovrenčić A., Konecki M. 2017., „Programiranje u 14 lekcija“)

```
#include<iostream>

using namespace std;


int main(){
    int brojevi[10];
    int zamjena=0;
    for (int i = 0; i < 10; i++)
    {
        cout << "Unesite " << (i+1) << ". broj: ";
        cin >> brojevi[i];
    }


    for (int i = 9; i > 0; i--)
    {
        int max = 0;
        for (int j = 1; j <= i; j++)
        {
            if (brojevi[j]>brojevi[max])
            {
                max=j;
            }
        }
```

```
            if (i!=max)
            {
                zamjena = brojevi[i];
                brojevi[i] = brojevi[max];
                brojevi[max] = zamjena;
            }
        }
        for (int i = 0; i < 10; i++)
        {
            cout << brojevi[i] << endl;
        }
        system("pause");
        return 0;
    }
```

3.6.2.2. Sortiranje zamjenom

Sortiranje zamjenom (eng. *Exchange sort*) također je vrlo jednostavan algoritam sortiranja. Ovaj se algoritam vrši tako da se nekoliko puta moro proći kroz polje. Svakim prolaskom smješta se jedan element na svoje mjesto. Vrijednost posljednjeg elementa uspoređuje se redom s vrijednostima svih prethodnih elemenata polja. Ako se naiđe na vrijednost veću nego vrijednost posljednjeg elementa, one se zamijene. Na kraju prvog prolaska dobiva se najveća vrijednost u zadnjem elementu polja. Nakon toga koristi se isti postupak za pretposljednji element, te zatim za sljedeći i tako dalje dok ne preostane zadnji element i polje je sortirano. Primjer ovog sortiranja može se vidjeti u poglavlju gdje se programski jezik c++ uspoređuje s .NET C#. (Lovrenčić A., Konecki M. 2017., „Programiranje u 14 lekcija“)

3.6.2.3. Mjehurićasto sortiranje

Mjehurićasto sortiranje (eng. *bubblesort*) smatra se inačicom sortiranja zamjenom. Mjehurićasto sortiranje vrši se tako da se prolazi kroz polje i vrijednost jednog elementa polja se uspoređuje sa vrijednošću svakog ostalog elementa u polju, kada se naiđe na element manje vrijednosti, vrši se zamjena, Na kraju prvog prolaska kroz polje, element najveće vrijednosti naći će se na kraju polja. Zatim se postupak ponavlja za svaki element, potpuno isto kao kod sortiranja zamjenom. No, razlika je u tome što mjehurićasto sortiranje ima još jednu varijablu koja će provjeriti je li se u jednom prolasku kroz polje izvršila ijedna izmjena,

ukoliko nije, znači da je polje sortirano te se proces sortiranja ranije prekida. Primjer mjehurićastog sortiranja također ćemo prikazati zadatkom u kojem je potrebno unijeti 10 brojeva te izvršiti mjehurićasto sortiranje. (Lovrenčić A., Konecki M. 2017., „Programiranje u 14 lekcija“)

```
#include<iostream>
using namespace std;


int main(){
    int brojevi[10];
    int zamjena=0;
    for (int i = 0; i < 10; i++)
    {
        cout << "Unesite " << (i+1) << ". broj: ";
        cin >> brojevi[i];
    }
    bool stanje = true;
    for (int i = 9; stanje && i > 0; i--)
    {
        stanje = false;
        for (int j = 0; j < i; j++)
        {
            int k = j+1;
            if (brojevi[j]>brojevi[k])
            {
                zamjena = brojevi[j];
                brojevi[j] = brojevi[k];
                brojevi[k]= zamjena;
                stanje = true;
            }
        }
```

```
    }
    for (int i = 0; i < 10; i++)
    {
        cout << brojevi[i] << endl;
    }
    system("pause");
    return 0;
}
```

3.7. Funkcije

Funkcijom se smatra dio programskog koda koji obavlja jedan zadatak. Kod korisnički definiranih funkcija u programu moramo imati:

1. Poziv funkcije
2. Definiciju funkcije
3. Deklaraciju funkcije

3.7.1. Poziv funkcije

Funkcija se poziva u onome dijelu programskog koda gdje želimo ju želimo upotrijebiti, odnosno tamo gdje želimo da se izvrši zadatak funkcije. Poziv funkcije se sastoji od njena imena i okruglih zagrada gdje se nalaze varijable, odvojene zarezima, čije vrijednosti šaljemo funkciji na obradu. Poziv funkcije kojoj se šalju dvije varijable izgleda ovako:

```
ime_funkcije(varijabla1,varijabla2);
```

3.7.2. Definicija funkcije

Funkcija se sastoji od zaglavlja funkcije, tijela funkcije i naredbe return. U zaglavlju funkcije vrši se deklaracija funkcije tj. zapisuje se tip varijable koju ta funkcija vraća pozivatelju funkcije, nakon tipa slijedi ime funkcije te zatim u okruglim zagradama deklaracija varijabli koje funkcija prima. Tijelo funkcije čine sve naredbe unutar te funkcije i pozivi ostalih funkcija koje ta funkcija poziva. Zadnja linija u funkciji rezervirana je za naredbu return. Iza naredbe return navodi se varijabla koju ta funkcija vraća pozivatelju.

Također postoje i funkcije koje služe kako bi odradile neki zadatak, ali ne vraćaju vrijednosti pozivatelju funkcije. Te su funkcije tipa void. Deklaracija void funkcije ista je kao i

kod funkcija koje vraćaju neke vrijednosti. Bitna razlika je to da se u void funkciji iza naredbe return ne zapisuju varijable jer se pozivatelju ne šalje rezultat.

Primjer zadatka: Unijeti 10 brojeva i u funkciji ispisati 2 najveća broja.

```
#include<iostream>
using namespace std;
void dvaNajveca(int br1, int br2, int br3){
    if (br1<br2 && br1<br3)
    {
        cout << "Dva najveća broja su: " << br2 << " i " << br3;
    }
    if (br2<br1 && br2<br3)
    {
        cout << "Dva najveća broja su: " << br1 << " i " << br3;
    }
    if (br3<br1 && br3<br2)
    {
        cout << "Dva najveća broja su: " << br1 << " i " << br2;
    }
    return;
}
int main(){
    int broj1, broj2, broj3;
    cout << "Unesite 1. broj: ";
    cin >> broj1;
    cout << "Unesite 2. broj: ";
    cin >> broj2;
    cout << "Unesite 3. broj: ";
    cin >> broj3;
    dvaNajveca(broj1, broj2, broj3);
    return 0;
```

```
}
```

3.8. Pokazivači

U nekim situacijama ne možemo unaprijed odrediti količinu memorijskog prostora potrebnu programu za izvršenje obrade, nego ta količina ovisi o instanci rješavanog problema. Hoće reći. Ta se količina može odrediti tek prilikom izvođenja programa. Tada koristimo pokazivače jer oni omogućuju dinamičku alokaciju memorije. Pokazivač predstavlja varijablu koja sadrži adresu neke druge varijable. Prvenstveno, pokazivači služe kako bi program od sustava imao mogućnost tražiti dodatnu memoriju za rad. (Lovrenčić A., Konecki M. 2017., „Programiranje u 14 lekcija“)

> *„Tip podataka kod pokazivača ne definira kojeg će tipa biti sama pokazivačka varijabla, koja uvijek sadrži memorijsku adresu, već se on odnosi na veličinu memorijskog prostora na koji pokazivač pokazuje kako bi se znalo koliko se memorije treba alocirati (zatražiti od operacijskog sustava) i na koji se način interpretira podatak smješten u toj memoriji.“*
> *(Lovrenčić i Konecki, 2017., str. 218.)*

4. Usporedba C++ i C# .NET programskih jezika

C++ i C#. NET, u daljnjem tekstu samo C#, programske jezike za početak ćemo usporediti s prikazom osnovnih razloga na slici.

C++	C#
Low-level programski jezik	High-level programski jezik
Objektno orijentirano programiranje	Objektno orijentirano programiranje
Potrebno ručno brisanje memorije koja više nije potrebna	Ima garbage collector koji automatski obavlja brisanje nepotrebne memorije
Može se koristiti na bilo kojoj platformi, odnosno operacijskom sustavu	Namijenjen za Windows operacijske sustave
Koristi se za serverske aplikacije, mreže, igre i drivere	Dobar je za desktop, web i mobilne aplikacije
Nema foreach iteracijsku petlju	Sadrži foreach iteracijsku petlju
Dozvoljava pitanje koda na sve načine sve dok nema sintaktičkih grešaka	Visoko zaštićen, javlja greške i upozorenja ukoliko postoji mogućnost da će napisani kod nekako naštetiti programu

Slika 5 Usporedba C++ i C#

Usporediti ćemo jednak zadatak napisan u programskom jeziku C++, a zatim u programskom jeziku C#. Naglasak će biti u količini memorije koju programski kodovi zauzimaju i u količini napisanog programskog koda.

4.1. Zadatak 1

Prvi zadatak glasi ovako: Unesite u polje 10 brojeva te ih ispišite sortirane uzlazno.

Programski kod napisan u C++:

```
#include <iostream>
```

```
using namespace std;
int main() {
	int brojevi[10], t;
	for (int i = 0; i < 10; i++)
	{
		cout << "Unesite" << i + 1 << ". broj: ";
		cin >> brojevi[i];
	}
	for (int i = 0; i < 9; i++)
	{
		for (int j = i + 1; j < 10; j++)
		{
			if (brojevi[i] > brojevi[j])
			{
				t = brojevi[i];
				brojevi[i] = brojevi[j];
				brojevi[j] = t;
			}
		}
	}
	cout << endl << "Sortirano polje je: ";
	for (int i = 0; i < 10; i++)
	{
		cout << brojevi[i] << endl;
	}
	system("pause");
	return 0;
}
```

Programski kod napisan u C#:

```
using System;
```

```
namespace sort
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] brojevi = new int[10];
            for (int i = 0; i < 10; i++)
            {
                Console.WriteLine("Unesite " + (i+1) + ". broj:");
                brojevi[i] = int.Parse(Console.ReadLine());
            }
            for (int i = 0; i < 9; i++)
            {
                  for (int j = i + 1; j < 10; j++)
                  {
                        if (brojevi[i] > brojevi[j])
                        {
                              t = brojevi[i];
                              brojevi[i] = brojevi[j];
                              brojevi[j] = t;
                        }
                  }
            }
            Console.WriteLine("\n Sortirano polje je: ");
            for (int i = 0; i < 10; i++)
            {
                Console.WriteLine(brojevi[i]);
            }
            Console.ReadLine();
```

```
        }
    }
}
```

C# program napisan je u 34 linije koda dok je program s istom funkcijom napisan u C++-u napisan u 29 linija. Nakon provjere zauzete memorije računala programskih kodova rezultati su bili sljedeći: C# napisan u Visual Studiju 2017. zauzima 1.41 MB memorije, dok C++ u Visual Studiju 2017. zauzima 8.76 MB. Rezultatima se na prvi dojam čine nerealni, no kod uporabe C++ programskog jezika, Visual Studio 2017. dodaje razna pomagala koja zauzimaju mnogo memorije. Ukoliko želimo smanjiti utrošenu memoriju, trebali bi koristiti neko drugo okruženje. Kao dobar odabir pokazao se Visual Studio Code, koji je vrlo spretan za pisanje C++ programskog koda, a memorija zauzeta istim kodom u Visual Studio Code zauzima svega 47.1 KB, što je u odnosu na C++ u Visual Studiju 2017. ili u odnosu na C# daleko manje.

4.2. Zadatak 2

Sljedeći zadatak u nizu nešto je kompleksniji od prvog, a glasi ovako: Napravite kamatni račun, unesite iznos glavnice, kamatne stope i trajanja kredita te po pravilima kamatnog računa ispišite redom kroz sve mjesece ostatak posebno glavnice, a posebno kamate kroz otplatu kredita.

Programski kod napisan u C++:

```
#include<iostream>
using namespace std;
int main()
{
   double preostaloGlavnica = 0, preostaloKamata = 0, rata = 0;
   double glavnica, kamata, vrijeme;
   cout << "Unesite iznos glavnice: ";
   cin >> glavnica;
   cout << "Unesite iznos kamate: ";
   cin >> kamata;
   cout << "Unesite trajanje u mjesecima: ";
   cin >> vrijeme;
```

```
      double iznosKamate = glavnica * kamata / 100 * vrijeme / 12;
      rata = (glavnica + iznosKamate) / vrijeme;
      preostaloKamata = iznosKamate;
      preostaloGlavnica = glavnica;
      cout << "---------------------------------" << endl;
      cout << "Iznos rate: " << rata << endl;
      cout << "---------------------------------" << endl;
      for (int i = 1; i <= vrijeme; i++)
      {
            preostaloKamata -= rata;
            if (preostaloKamata < 0)
            {
                  preostaloGlavnica += preostaloKamata;
                  preostaloKamata = 0;
            }
            if (i == vrijeme)
            {
                  preostaloGlavnica = 0;
            }
            cout << i << ". mjesec: Preostali iznos glavnice = " <<
preostaloGlavnica << ", preostali iznos kamate: " << preostaloKamata <<
endl;
      }
      system("pause");
      return 0;
   }
```

Programski kod napisan u C#

```
   using System;
   namespace Kredit
   {
       class Program
```

```
{
    static void Main(string[] args)
    {
        double preostaloGlavnica = 0, preostaloKamata = 0, rata = 0;
        Console.WriteLine("Unesite iznos glavnice: ");
        double glavnica = double.Parse(Console.ReadLine());
        Console.WriteLine("Unesite iznos kamate: ");
        double kamata = double.Parse(Console.ReadLine());
        Console.WriteLine("Unesite trajanje u mjesecima: ");
        double vrijeme = double.Parse(Console.ReadLine());
        Console.SetWindowSize(170, 58);
        double iznosKamate = glavnica * kamata / 100 * vrijeme / 12;
        rata = (glavnica + iznosKamate)/vrijeme;
        preostaloKamata = iznosKamate;
        preostaloGlavnica = glavnica;
        Console.WriteLine("---------------------------------");
        Console.WriteLine("Iznos rate: " + rata);
        Console.WriteLine("---------------------------------");
        for (int i = 1; i <= vrijeme; i++)
        {
            preostaloKamata -= rata;
            if (preostaloKamata < 0)
            {
                preostaloGlavnica += preostaloKamata;
                preostaloKamata = 0;
            }
            if (i==vrijeme)
            {
                preostaloGlavnica = 0;
            }
```

```
                    Console.WriteLine(i + ". mjesec: Preostali iznos
glavnice = " + preostaloGlavnica + ", preostali iznos kamate: " +
preostaloKamata);

                }

                Console.ReadLine();

            }

        }

    }
```

Pogledom na same kodove u oba jezika ovog zadatka nećemo primijetiti velike razlike, čak je i broj redova linija približno jednak. To je zato što u ovom zadatku nemamo nekih posebnih opcija u C#-u, kao što smo imali kod sortiranja, no, bacimo pogled na zauzetu memoriju računala. C# u Visual Studiju 2017. zauzima 0.81 MB, C++ u Visual Studiju 2017 zauzima čak 13 MB (isključivo iz razloga što je uključena kompletna iostream biblioteka i sve što je u njoj), a C++ u Visual Studio Code 47.4 KB, što ponovno potvrđuje neisplativost pisanja C++ koda u Visual Studiju 2017, ukoliko se obaziremo na zauzimanje memorijskog prostora.

5. Zaključak

U radu je obrađeno osnovno programiranje u programskom jeziku C++. Izradom ovog rada iznio sam određeno do sada stečeno znanje većinom o programskom jeziku C++, a manjim dijelom o C i C# jezicima. Po određenim kriterijima usporedio sam C++ i C# te su neki čimbenici išli pod ruku jednom, a neki drugom jeziku. S mojih samo nekoliko godina iskustva učenja programiranje u C++ i C# jezicima, kada bih naišao na pitanje koji je od ta dva programska jezika bolji, ne bih mogao dati odgovor.

C i C++ programske jezike smatram odličnima za rad i za učenje, a pogotovo učenje samih korijena i logike programiranja. Uvijek bih preporučio programski jezik C++ za svaki početak i ulazak u svijet programiranja.

Popis literature

[1] Šribar J, Motik B (2003.) *„Demistificirani C++“*. Zagreb

[2] Lovrenčić A. i Konecki M. (2017.) „*Programiranje u 14 lekcija*“, Varaždin

[3] Lovrenčić A (2018.) „*Apstraktni tipovi podataka i algoritmi*“, Varaždin

[4] Šribar J, Motik B (2018.) *„Demistificirani C++“*. Zagreb

[5] Guru99, „*Koja je razlika između C++ i C#*“ Preuzeto 30.8.2019. s https://www.guru99.com/cpp-vs-c-sharp.html

[6] Jennifer Marsh, „*C# vs. C++ : Koji jezik je dobar za tvoj projekt*“ Preuzeto 30.8.2019. s https://www.upwork.com/hiring/development/c-sharp-vs-c-plus-plus/

Popis slika

Slika 1Slijed izvršavanja naredbe for 11
Slika 2 Slijed izvršavanja naredbe while (Lovrenčić A., Konecki M.,2017.) 13
Slika 3 Tablica ASCII kodova(https://www.commfront.com/pages/ascii-chart, preuzeto 30.8.2019.) 15
Slika 4 Prikaz polja s 10 elemenata 18
Slika 5 Usporedba C++ i C# 26

Popis tablica

Tablica 1. Elementarni tipovi podataka 6