Izrada kviza u programskom jeziku C++

Fodor, Filip

Sveučilište Jurja Dobrile u Puli

Fakultet ekonomije i turizma

„Dr. Mijo Mirković“

FILIP FODOR

IZRADA KVIZA U PROGRAMSKOM JEZIKU C++

Završni rad

Pula, rujan 2019.

Sveučilište Jurja Dobrile u Puli

Fakultet ekonomije i turizma

„Dr. Mijo Mirković“

FILIP FODOR

IZRADA KVIZA U PROGRAMSKOM JEZIKU C++

Završni rad

JMBAG: 0303059799 , redovan student

Studijski smjer: Poslovna informatika

Kolegij: Osnove programiranja

Znanstveno područje: Društvene znanosti
Znanstveno polje: Ekonomija
Znanstvena grana: Poslovna informatika

Mentor: doc.dr.sc Tihomir Orehovački

Pula, rujan 2019.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Filip Fodor, kandidat za prvostupnika poslovne ekonomije, smjera poslovna informatika ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, _____, _____ godine

IZJAVA

O KORIŠTENJU AUTORSKOG DJELA

Ja, Filip Fodor dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom „Izrada kviza u programskom jeziku C++ “ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____ (datum)

Potpis

_____

Sadržaj

1. UVOD

Svakako najpopularniji programski jezik današnjice je programski jezik C++. Iako veoma složen, najčešće je namijenjen stručnjacima ali i onima koji samostalno žele naučiti programirati. U ovom radu biti će prikazane osnove stručne terminologije ovoga programskoga jezika uz nekoliko kratkih primjera. Za obavljanje bilo koje vrste poslova neophodni su programi. Program je uputa računalu kako da obavi neki posao, bilo da se radi o programu za obračun plaća, izdavanje računa, uređivanje fotografija i sl. Osnovni element svakog programa je naredba, odnosno instrukcija. Poznato je da računalo razumije samo znamenke binarnog brojevnog sustava (0 i 1). Zbog toga je svaki program koji je pisan govornim jezikom potrebno prevesti u oblik koji je razumljiv računalu. Taj postupak naziva se kompajliranje. Prilikom rješavanja nekog problema ponajprije ga je potrebno analizirati, nakon toga definirati zadatke koji će nam pomoći pri njegovu rješavanju, osmisliti rješenje, napisati program te ga provjeriti radi li on ispravno.

U svakodnevnom životu koristimo korisničke programe te su oni neizostavni u obavljanju bilo koje vrste poslova. Najpoznatiji korisnički programi su oni za pisanje i obradu tekstova, crtanje, programi za rad s proračunskim tablicama, kreiranja baza podataka, slanja elektroničke pošte i sl. Svi ti programi smješteni su u trajnoj memoriji na disku računala. Pokretanje programa obavlja se na način da se s diska u radnu memoriju prebaci dio naredbe nekog programa i nakon toga procesor omogućava njihovo izvršenje.

Rad se sastoji od 6 poglavlja. Prvi dio uvodi u sam rad i strukturu rada. Drugi dio obuhvaća uvod u programiranje, a detaljnije je opisan razvoj programskih jezika kroz povijest te generacije i vrste programskih jezika. Treći dio odnosi se na programiranje u programskom jeziku C++ koji je izabran za praktičan dio ovoga rada. U tom poglavlju opisane su vrste podataka, varijable, osnovni tipovi podataka, logički tip podataka, funkcije i petlje. U četvrtom poglavlju prikazana je usporedba jezika C++ s ostalim programskim jezicima. Peti dio rada prikazuje praktični dio i kreiranje kviza u programskom jeziku C++.

2. PROGRAMSKI JEZICI

Za pisanje računalnih programa koristimo programske jezike koji su najčešće utemeljeni frazama engleskoga govornog područja. Programski jezik najlakše je definirati kao skup riječi ili pravila koje razumije računalo, a koja su neophodna za njegovo korištenje. Danas je gotovo nezamislivo obavljati bilo koju vrstu posla bez računala. No, da bi ono uz niz međusobno povezanih komponenti obavljalo odgovarajuće naredbe ili funkcije mora biti opremljeno različitim programima.

Program je dakle slijed instrukcija koje se odvijaju točno određenim redoslijedom i spremaju u memoriju računala. Procesor ih dohvaća i sprema u registre.[1]

Proces programiranja odnosi se, osim na pisanje kodova, i na izradu algoritma koji je ujedno i uputa za rješavanje nekog zadatka. Algoritme koristimo u svakodnevnom životu a da često toga nismo svjesni. Od nekih jednostavnih životnih navika poput kuhanja kave, do složenijih zadataka kao što je primjerice sastavljanje složenijeg komada namještaja koji dolazi s uputama i koracima sastavljanja. Stoga svaki algoritam mora biti jasno i jednostavno napisan tako da svatko prema uputama može postići određeni cilj.

2.1. Programski jezici kroz povijest

Programiranje se iz svoje kompleksnosti kroz povijest razvijalo i samim time dovelo do toga da su današnji programi jednostavniji i kraći.

Početak programiranja pripisuje se prvoj programerki Adi Byron. Matematičar, von Neumann, zaslužan je za pomicanje granica programiranja kada su se programi pisali tzv. strojnim jezicima. Polovicom 20. st. dolazi do pojave simboličkih jezika u kojima su se naredbe pisale kraticama koje su podsjećale na engleske riječi.[2]

Karakteristika strojnih jezika je da se oni sastoje od nula i jedinica. No da bi čovjek donekle razumio ovakav jezik potrebno je nule i jedinice prikazati simbolima

[1] P. Brođanac, L. Budin, Z. Markučić, S. Perić, *Programski jezik C++*, Zagreb, Školska knjiga, str. 105.
[2] N. Lipljin, *Programiranje 1*, Varaždin, TIVA Tiskara, 2004., str. 16.

koje predstavljaju asemblerski, odnosno simbolički jezik. Kasnije su se razvili viši programski jezici čiji je osnovni element naredba ili instrukcija. Takvi se programski jezici prije izvođenja prevode u strojni oblik koje razumije računalo. Oni su kraći te pregledniji i čovjeku razumljiviji.[3]

Neki od poznatijih programskih jezika koji su se razvili kroz povijest su: FORTRAN (najviše se upotrebljavao za matematičke izračune), jednostavan programski jezik BASIC, COBOL (namijenjen bazama podataka), Pascal, programski jezik C i mnogi drugi. Današnji najpoznatiji programski jezici su C++ kao preteča programskog jezika C, Java, Python i C#. U poglavlju 4. prikazati ću kratku usporedbu programskog jezika C++ s ostalim, poznatijim jezicima današnjice.

I. generacija (rane 50-e)	Za ovu generaciju karakteristični su strojni jezici (0 i 1). Programiranje je otežano te je velika mogućnost da dođe do pogreške rada programa.
II. generacija (sredina 50-tih)	Simbolički jezici (asembler) glavna su karakteristika ove generacije. Svaku je naredbu prije izvođenja potrebno prevesti u strojni oblik.
III. generacija (60-ih godina)	Viši programski jezici i proceduralni jezici. Jezici ove generacije su FORTRAN, BASIC, LOGO, Ada, Pascal i dr.
IV. generacija	Programski jezici prilagođeni krajnjim korisnicima. Pojava objektno orijentiranog programiranja i neproceduralnih jezika.
V. generacija	Temelji se na rješavanju problema korištenjem ograničenja koja se dodjeljuju programu, a ne korištenjem algoritma kojeg je napisao programer.

Tablica 2.1.1. Generacije programskih jezika[4]

[3]P. Brođanac, L. Budin, Z. Markučić, S. Perić, op. cit., str. 107.

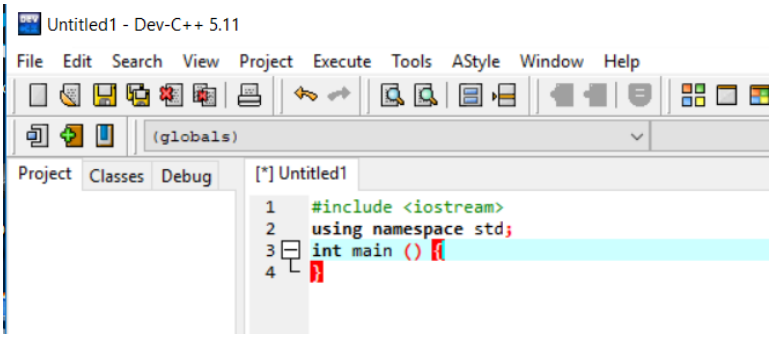
[4] T. Gvozdanović, Z. Ikica, I. Kos, K. Kudumija, M. Kuzminski, LJ. Milijaš, N. Milijaš, G. Sekulić-Stivčević, Lj. Zvonarek, *e-u INFO GIM*, Zagreb, PRO-MIL d.o.o., 2018., Dostupno na: www.e-u.hr (pristupljeno 8. rujna 2019.)

2.3. Razvojna sučelja

U današnje vrijeme postoji mnogo razvojnih sučelja koja se koriste za razvijanje različitih aplikacija (IDE). Razvojna sučelja uključuju i prevoditelje. To su programi koji nam omogućavaju da pisane programe u programskom jeziku C++ prevedemo u izvedbeni kôd. U nastavku će ukratko biti opisani Dev-C++ i Microsoft Visual Studio. Osim njih vrlo popularno je i razvojno okruženje Code::Blockste sučelje za programiranje aplikacija Google Ads (ADI).

2.3.1. Dev-C++

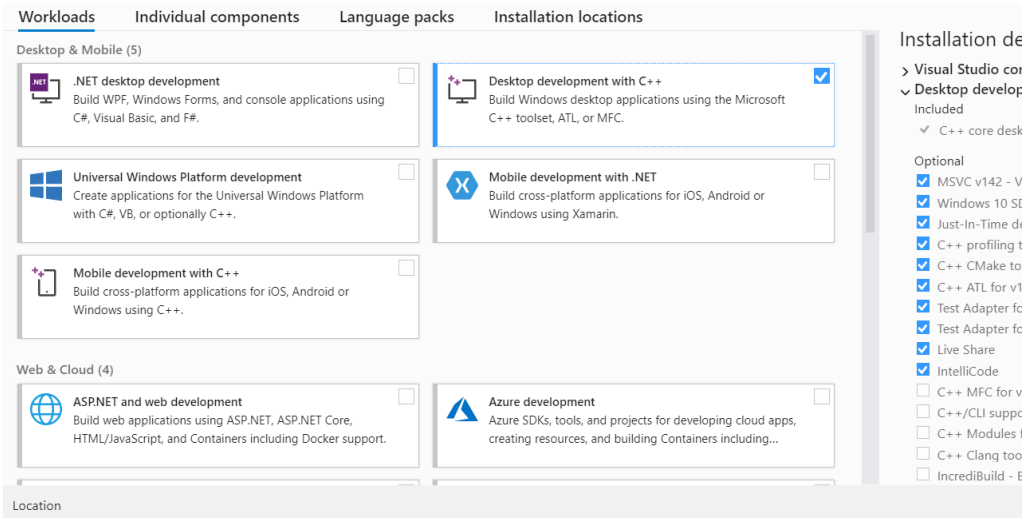
Jedno od najpoznatijih i najčešće korištenih razvojnih sučelja za programske jezike C/C++ je Dev-C++. Ovo razvojno sučelje moguće je besplatno skinuti s različitih web lokacija te je nakon preuzimanja potrebno pokrenuti izvršnu datoteku. Dolazi i na hrvatskom jeziku te je jednostavan za korištenje.



Slika 2.3.1.1. Dio prozora razvojnog sučelja Dev-C++

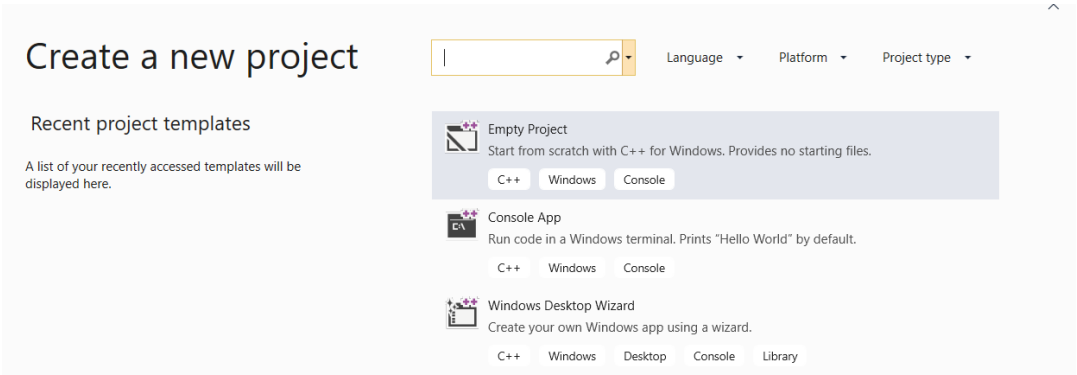
2.3.2. Microsoft Visual Studio i instalacija

Razvojno sučelje opisano u nastavku također je besplatno i jednostavno se može preuzeti sa web stranice tvrtke Microsoft. Trenutno je dostupna najnovija inačica ovog razvojnog okruženja koja omogućuje i izradu aplikacija te pisanje programa u oblaku. Nakon preuzimanja razvojnog sučelja odabiremo jednu od opcija i načina izrade našeg projekta (*workloads*). Svaka skupina dolazi s detaljnim opisom paketa koji se vidi s desne strane na slici 2.3.2.1.



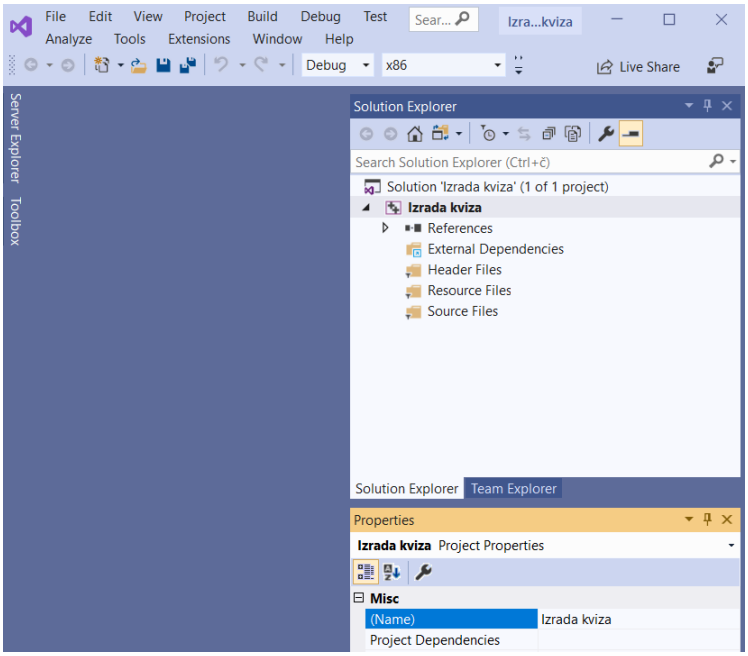
Slika 2.3.2.1. Instalacija razvojnog sučelja Microsoft Visual Studio

Nakon odabrane skupine i dodatnih instalacijskih paketa odaberemo opciju *Create a new project* nakon čega se otvara prozor sličan ranije opisanom razvojnom okruženju.

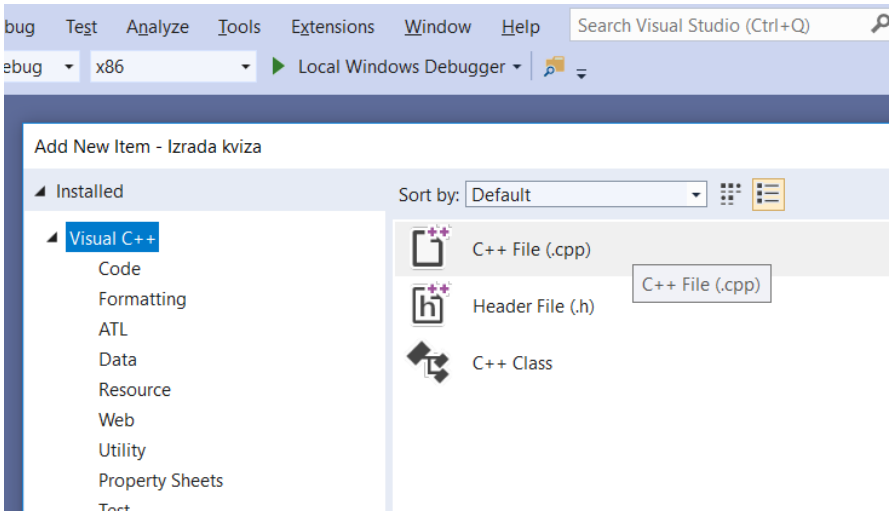


Slika 2.3.2.2. Predlošci ranije kreiranih projekata

Nakon što odaberemo mjesto na koje ćemo spremiti naše projekte otvara se razvojno sučelje. S lijeve strane vidljiv je prozor *Solution Explorer* koji prikazuje naziv našeg programa te njegove mape. Kombinacijom tipaka Ctrl+SHIFT+A dodajemo C++ datoteku u naš program.



Slika 2.3.2.3. Prozor razvojnog sučelja



Slika 2.3.2.4. Dodavanje C++ datoteke

Nakon dodavanja C++ datoteke prikazuje se prozor sličan kao i kod ostalih razvojnih okruženja te je sve spremno za pisanje programa, tj. kôdiranje.

3. PROGRAMSKI JEZIK C++

U ovom poglavlju biti će opisani osnovni pojmovi programskog jezika C++ koji je izuzetno kompleksan i jezgrovit te o kojem su napisane razne knjige, radovi i provedena istraživanja. Kao prikaz definicije samog jezika odlučio sam citirati autore Šribar i Motik.

„Jezik C++ je jezik opće namjene, za poslove od sistemskog programiranja do razvoja korisničkih sučelja te čisto apstraktnih primjena kao što su matematički proračuni i predstavlja najkorišteniji programski jezik za pisanje komercijalnih aplikacija. To je mjesto stekao dobrim dijelom zahvaljujući svom prethodniku – jeziku C. Svi C programi se gotovo u cijelosti mogu prevesti u C++ prevoditeljem potpuno ispravno, te se na već postojeće projekte lako može dograditi objektno orijentirani modul.“[5]

Programski jezik C++ je objektno orijentirani programski jezik te ima četiri važna svojstva:[6]

1. enkapsulacija (*engl.* encapsulation),
2. skrivanje podataka (*engl.* data hiding),
3. nasljeđivanje (*engl.* inheritance) i
4. polimorfizam (*engl.* polymorphism).

Objektno programiranje pridonosi razbijanju velikih i složenijih programa u niz zatvorenih cjelina (objekata) koji sudjeluju u rješavanju problema. Objekti se mogu zamijeniti onima koji će biti brži kod rješavanja nekog problema i raditi bolje te je zbog toga bitno što objekt radi, a ne kako radi.

Enkapsulaciju drugim riječima nazivamo učahurivanje te ono predstavlja zajedništvo podataka i operacija. Odnosno, svaki objekt osim podataka mora sadržavati i metode koje su neophodne za rješavanje problema i izvođenje operacija. Privatnost (skrivanje) podataka je svojstvo koje objektu omogućuje pružanje podataka i operacija koje su neophodne za njegovo korištenje. Takvi javno dostupni podaci čine sučelje programa. Svojstvo nasljeđivanja podrazumijeva ponovno

[5] J. Šribar, B. Motik, *Demistificirani C++*, Zagreb, Element d.o.o., str. 786
[6] Ibidem, str. 3

iskorištavanje podataka koji mogu biti neizmijenjeni ali i dorađeni. Svojstvo višezadaćnosti odnosi se na pojavljivanje različitih oblika i tipova podataka.

Svaki program koji je napisan u programskom jeziku C++ mora sadržavati točno jednu **main ()** funkciju (glavna funkcija svakog programa). Najjednostavniji primjer programa izgleda ovako:

```
int main (){

return 0;

}
```

Riječ **int** predstavlja tip glavne funkcije te će ona nakon izvođenja naredbi vratiti cijeli broj. Unutar otvorene i zatvorene okrugle zagrade dolaze podaci koji se nazivaju argumenti ili parametri funkciji. Otvorena vitičasta zagrada označava početak, a zatvorena kraj bloka naredbi. Naredbom **return 0** završava glavna funkcija i nju pišemo prije zatvorene vitičaste zagrade. Znak **;** označava kraj naredbe ili instrukcije.

Sada ćemo nadograditi gornji jednostavan primjer programa, koji nije vraćao nikakav rezultat, bitnim dijelovima svakog programa pisanog u C++.

```
#include <iostream>
using namespace std;
int main () {
    cout << "U tijeku je pisanje završnoga rada!" << endl;
    return 0;
}
```

Pretprocesorska naredba **#include** u prvom retku u program uključuje biblioteku **iostream** koja omogućava ispis podataka na zaslon računala. Na početku svakog programa navode se pretprocesorske funkcije koje se pišu prije glavne funkcije main. **Using** i **namespace** su ključne riječi programskog jezika C++, a std je naziv imenika koji obuhvaća sve funkcije.

Ukoliko se programski kôd sastoji od mnoštvo redaka nezaobilazni su komentari koji nam omogućavaju lakše snalaženje. Komentari se u C++ pišu dvostrukim kosim crtama (//).

3.1. Vrste podataka

Podatke možemo podijeliti prema vrsti i tipu. Podaci se prema vrsti dijele na sljedeći način:[7]

a) prema **mjestu** na kojem su podaci pohranjeni:
    a. u glavnoj memoriji (interni)
    b. na vanjskim medijima (eksterni)
b) prema **promjenjivosti** mogu biti:
    a. statički – podaci dobivaju potreban memorijski prostor (npr. Cijeli brojevi 16/32 bita, znak jedan bajt)
    b. dinamički–podacima se dodjeljuje minimalna količina memorijskog prostora
c) prema **jednostavnosti**:
    a. jednostavni – ne gube smisao ukoliko se podijele na manje dijelove
    b. strukturirani – sastavljeni od jednostavnih podataka
d) prema **linearnosti**:
    a. precizni podaci kojima je poznati prethodnik i sljedbenik nazivaju se linearno totalno uređeni podaci (brojčani, nebrojčani, logički),
    b. linearno uređene, povezane – njima se ne može odrediti prethodnik ni sljedbenik.

3.2. Varijable

Varijabla predstavlja memorijsku lokaciju i ona tijekom izvršavanja programa može mijenjati svoju vrijednost. Postoji nekoliko pravila za imenovanje varijabli:

1. imena varijabli moraju biti kreirana slovima engleske abecede, brojeva od 0 do 9, te znakom podcrtano _,
2. prvi znak varijable mora biti slovo ili znak _,
3. ime varijabli ne smije biti neka ključna riječ,

[7] N. Lipljin, op. cit., str. 64.

4. razlikujemo velika i mala slova (npr. *ime* i *IME* su dvije različite varijable),
5. broj znakova kod imenovanja varijabli nije ograničen.

Svakoj se varijabli određuje i tip podataka koji će biti opisani u nastavku. Prema tipu podataka računalo prepoznaje koliko mjesta u memorije zauzima pojedina varijabla. Varijable razlikujemo prema tipu podataka i njihovim simboličkim vrijednostima. Na početku svakog programa potrebno je zadati simbolička imena varijablama a taj postupak naziva se deklariranje varijabli. Postupak deklaracije ide po principu da se najprije odredi tip podataka, a nakon toga popis imena koja su odvojena zarezom. Za pridruživanje vrijednosti varijablama koristimo operator pridruživanja koji se označava znakom jednakosti (=).

Deklaracija varijabli	Opis
int a;	varijabla a je cjelobrojna vrijednost – tip *integer*. Slovo a je simboličko ime varijable.
float broj, z;	broj i z su realne varijable odvojene zarezom
int i = 5;	cjelobrojnoj varijabli i pridružena je vrijednost 5
y = y+1;	trenutnoj vrijednosti varijable y dodaj 1 i rezultata spremi u varijablu y

Tablica 3.2.1. Primjeri deklariranja varijabli

U tablici 3.2.1. prikazani su primjeri deklariranja varijabli s njihovim pojašnjenima te primjeri pridruživanja vrijednosti varijablama. Konstante su vrsta varijabli koje su za razliku od njih nepromjenjive. Pravilo za deklariranja konstanti je:

const tip_konstante ime_konstante = pridruzivanje_vrijednosti;

„Koriste se kada želimo zaštiti vrijednost neke varijable u našem programu da ju drugi programeri ne mogu naknadno mijenjati u svom kôdu.“[8]

[8] S. Fajković, op. cit., str. 23.

3.3. Osnovni tipovi podataka

U ovom poglavlju biti će opisani osnovni tipovi podataka koji se koriste u programskom jeziku C++.

3.3.1. Cijeli brojevi (integer)

Ukoliko je podatak cijeli broj tada on poprima oznaku **int**. Varijabla int poprima u memoriji računala 4 bajta (32 bita) za pohranu od kojih je prvi bit rezerviran za predznak broja. Ako se radi o prirodnom broju tada su na raspolaganju 32 bita jer bit za predznak nije potreban. Cjelobrojna varijabla bez predznaka deklarira se riječju **unsigned** ispred oznake varijable int. Cjelobrojnim tipovima podataka mogu se pridružiti i podtipovi kao što su short (manji raspon brojeva), long (veći raspon brojeva) i unsigned (pozitivni brjevi).

3.3.2. Realni brojevi (float)

Float tipu podataka pripadaju decimalni brojevi koji se u programskom jeziku C++ zapisuju simbolom točka. Realni brojevi mogu se prikazati:[9]

a) s nepomičnom decimalnom točkom i
b) s pomičnom decimalnom točkom (eksponencijalni prikaz broja).

*„Eksponencijalni prikaz broja je oblika M*10$^{E.}$ M označava dio broja koji se naziva mantisa, a zapisuje se tako da je prva znamenka različita od nule lijevo od decimalne točke. E je eksponent kojim treba potencirati bazu 10. U programskom jeziku C++ eksponencijalni prikaz broja je u obliku MeE, npr. 9.11e31. Za pohranu realnog broja u memoriji su predviđena 4 bajta (32 bita). Kada se govori o točnosti, u realnu se varijablu sprema sedam znamenki mantise.“*[10]

[9] D. Grundler, S. Šutalo, Računalstvo, Zagreb, Školska knjiga, 2019., str. 354.
[10] Ibidem, str. 354.

Pridruženi realni brojevi	Sprema se	Pravilo pohrane (sprema se samo 7 znamenki mantise)
a=1.23456;	*1.23456*	Uneseno je manje od 7 znamenki pa se spremaju sve znamenke.
x=1.2345671123;	*1.234567*	Uneseno je više od 7 znamenki, zanemarene su znamenke najmanjih težinskih vrijednosti.
y=1.23456739;	*1.234567*	Ista situacija kao u prethodnom primjeru.
z=1.23456789	*1.234568*	Uneseno je više od 7 znamenki, zanemarene su znamenke najmanjih težinskih vrijednosti. Broj je prije zanemarivanja znamenki zaokružen

Tablica 3.3.2.1. Točnost realne varijable[11]

Duplu veću preciznost daje tip podataka **double** koji i zauzima više mjesta u memoriji te je za njega predviđeno 8 bajtova.

*3.3.3. Znakovni tipovi podataka (*eng. *Character)*

Znakovni tipovi podataka nose oznaku **char**. Varijablama se pridružuju znakovi unutar jednostrukih navodnika (' ') ili dekadska vrijednost u ASCII kodu. Npr: *char abeceda = 'A'; char slovo = 22*. Bitno je napomenuti da se pod pojmom znak podrazumijeva samo jedan znak (memorijski prostor od 8 bitova).

Važno je znati da je string tip podatka zapravo skup znakova (simboli, slova i brojke). Nečije ime bi bilo tipa string ali isto tako možemo i brojeve zapisati kao stringove no tada se nad njima neće moći vršiti matematičke operacije. Važno je zapamtiti da se stringovi inicijaliziraju tako da se varijabli pridružuje neka vrijednost koja se nalazi pod dvostrukim navodnicima i da sve što je string tretiramo kao tekst neovisno što se nalazi između dvostrukih navodnika.“[12]

Primjeri **string** tipova podataka: string ime = „program“; string broj = „6“.

[11] Ibidem, str. 355.
[12] S. Fajković, *Uvod u programiranje*, 2015., Dostupno na: http://carpediem.hr/PublikacijeCarpeDiem/Publikacije/C++%20programiranje.pdf (pristupljeno 10. rujna 2019.)

3.4. Bool (logički tip podataka)

Logički tip podataka bool može poprimiti samo dva logička stanja, a to su logička istina (true/1) ili logička laž (false/0).

3.5. Matematička logika i programiranje

Kod logičkih podataka koriste se tri logička operatora:

1. logička negacija koja se (!),
2. logičko I (&&) i
3. logičko ILI (||).

Operator logičke negacije mijenja vrijednost varijable, točnije, istinu pretvara u laž i obrnuto. Logičko I daje istinu ukoliko su svi ulazi istiniti, a logičko ILI vraća neistinu ukoliko su svi ulazi lažni.

3.6. Tipovi podataka sa svojstvima

Na slici 3.6.1. prikazan je naziv tipova podataka s njihovim pripadajućim svojstvima (opis, veličina i raspon).

Name	Description	Size	Range
char	*Char*acter or small *integer*.	1byte	signed: -128 to 127 unsigned: 0 to 255
short *int*(short)	Short *integer*.	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
int	*integer*.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
long *int* (long)	Long *integer*.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
bool	Boolean value. It can take one of two values: true or false.	1byte	true or false
float	*Float*ing po*int* number.	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	Double precision *float*ing po*int* number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	Long double precision *float*ing po*int* number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	Wide *char*acter.	2 *or* 4 bytes	1 wide *char*acter

Slika 3.6.1. Tipovi podataka i njihova svojstva[13]

3.7. Funkcije

Kada, primjerice, želimo da nam program uvijek nakon upisivanja cijene nekog proizvoda/usluge ispisuje cijenu s PDV-om i popustom „pozivamo“ u pomoć funkcije. S jedne strane možemo uvijek ponovo pisati formulu kojom to želimo izračunati ili, jednostavnije, možemo samo jedanput definirati funkciju koja to radi i time si olakšati posao. Funkciju možemo definirati kao jednu vrstu potprograma koji „pozivamo“ dok nam zatreba(najčešće je to funkcija koja ima puno linija koda).

„Funkcija je potprogram , tj. logički samostalan dio programa koji predstavlja djelomično rješenje problema.“[14] Kao skup naredbi funkcije kroz određeni broj naredbi vraćaju neki rezultat. Funkcije se kreiraju izvan, a pozivaju unutar *main* funkcije, a sintaksa funkcija je sljedeća:

[13] S. Fajković, *Uvod u programiranje*, 2015., Dostupno na: http://carpediem.hr/PublikacijeCarpeDiem/Publikacije/C++%20programiranje.pdf (pristupljeno 10. rujna 2019.)
[14] N. Lipljin, op. cit., str 197.

povratni_tip ime_funkcije (parametri)
{
kod koji se izvršava
naredba return
}

Kod pisanja programa najčešće se koriste matematičke funkcije, a neke od njih su prikazane u tablici 3.8.1. Prije njihova korištenja potrebno je uključiti biblioteku u koju su one smještene.

Funkcija	Opis	Biblioteka
sqrt (x)	drugi korijen	math.h
abs (x)	apsolutna vrijednost broja	stdlib.h
round (x)	zaokruživanje	math.h
pow (a, b)	potenciranje	math.h

Tablica 3.7.1. Funkcije i njihove biblioteke

Vratimo se na početni primjer. Funkciju moramo i deklarirati. Deklarirati funkciju znači odrediti tip funkcije (int, float, char, …) i naziv funkcije, ali i tipove podataka svih argumenata koji se daju funkciji kod njezinog „pozivanja“(npr. float *izracun_cijene (float cijena, int kolicina).* Pozivamo funkciju za izračun cijene, a argumenti koje upisujemo biti će cijena (u našem slučaju osnovna neto cijena bez PDV-a, popusta i ostalog), a u funkciji ćemo definirati način izračuna ostalih komponenti koje su nam potrebne. Nakon toga funkciju moramo definirati.

```
float izracun_cijene (float  cijena, int kolicina)
{
float cijena_pdv;
cijena_pdv = cijena * 1,25 * kolicina;
return cijena_pdv;
}
```

Kada želimo da nam program izračuna cijenu s PDV-om i pomnoži s količinom pozvati ćemo funkciju izracun_cijene koja će odraditi dio koda unutar vitičastih zagrada. Osim funkcija koje samostalno definiramo postoje i ome koje su već ugrađene u programskom jeziku C++.

3.8. Petlje

Petlje omogućavaju izvršavanje djela programa sve dok je neki uvjet zadovoljen. U nastavku će biti opisana *for* petlja, while petlja i *do-while* petlja. Petlje drugim imenom nazivamo ponavljanja.

3.8.1. for petlja

Osnovna karakteristika **for** petlje je da se dio programa ponavlja unaprijed poznati broj put. Sintaksa for petlje izgleda ovako:

```
for ( početno stanje; uvjet; povećanje/smanjenje )
{
dio koda koji se izvršava sve dok je uvjet zadovoljen = blok
naredbi
}
naredba iza bloka
```

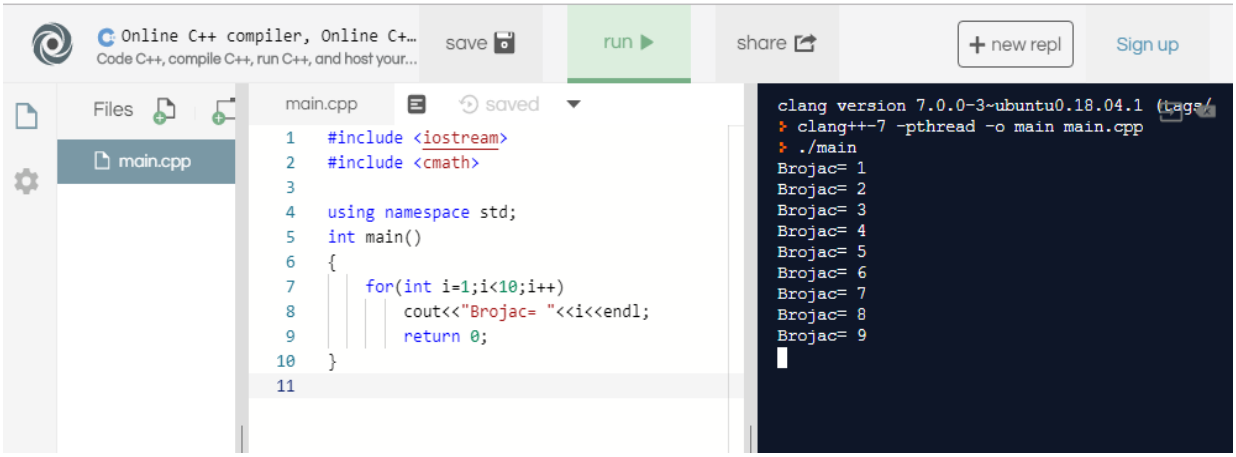
Početno stanje znači da postavljamo vrijednost varijable na određenu vrijednost. Najčešće se koriste slova i/j. Početno stanje varijable nije potrebno definirati prije korištenja for petlje već se vrijednost deklarira prilikom stvaranja te petlje. Uvjetom određujemo do kada će se for petlja izvršavati. Ukoliko se početna vrijednost postavi na 1 (*int i = 1;*), a uvjet je *i < 3*, to znači da će se petlja izvršiti dva puta. U zadnjem parametru određujemo za koliko ćemo uvećati ili smanjiti varijablu.

Primjer *for* petlje:

```
#include<iostream>
#include<cmath>

usingnamespace std;
```

```
int main()
{
for(int i=1;i<10;i++)
        cout<<"Brojac= "<<i<<endl;
return0;
}
```



Slika 3.8.1.1. Provjera for petlje u online C++ compiler-u

3.8.2. while i do-while petlja

Za razliku od *for* petlje kod *while* petlje nije poznat broj ponavljanja. Sintaksa while petlje:

```
while ( uvjet )
{
Kod koji će se izvršavati dok god je uvjet ispunjen
}
```

Primjer *while* petlje:

```
#include<iostream>

usingnamespace std;
int main()
```

```
{
int b = 0;
while (b<10)
{
  cout<<"Broj izvrsavanja while petlje "<< b + 1<< endl;
  b++;
}
}
```

„U „do-while“ petlji prvo navodimo kôd koji želimo da se izvrši a u drugom dijelu navodimo uvjet koji treba biti zadovoljen. Kako znamo da se naš program izvodi od najgornje linije prema dolje možemo uočiti kako u „do-while“ petlji to znači da će prvo doći do dijela u kojem se opisuje koji kôd se treba izvesti tek nakon toga do uvjeta. Upravo to je svojstvo „do-while“ petlje i to je jedina petlja za koju kažemo da će se sigurno izvršiti barem jednom.“[15]

Sintaksa while petlje:

do {
Kod koji će se izvršavati
}
While (uvjet);

[15] S. Fajković, *Uvod u programiranje*, 2015., Dostupno na: http://carpediem.hr/PublikacijeCarpeDiem/Publikacije/C++%20programiranje.pdf (pristupljeno 10. rujna 2019.).

4. USPOREDBA S DRUGIM PROGRAMSKIM JEZICIMA

U ovom ćemo poglavlju usporediti programski jezik C++ sa nekoliko poznatijih jezika današnjice.

4.1. Usporedba C++ sa Java[16]

Osnovna usporedba	C++	Java
Nasljeđivanje	Omogućuje jednostruko i višestruko nasljeđivanje	Ne omogućuje višestruko nasljeđivanje. Koristi koncept sučelja.
Mehanizam za otkrivanje pogrešaka tijekom izvođenja	Odgovornost programera	Odgovornost sustava
Upravljanje programom	Funkcije i podatci se mogu nalaziti izvan razreda. Dostupan je koncept globalnih varijablli I funkcija I dosega imenskog prostora	Sve funkcije varijable nalaze se unutar razreda. Koristi se koncept paketa.
Prenosivost	Platformski ovisno budući da se izvorni kod mora kompajlirati za svaku platformu	Koristi koncept bytecode-a koji je platformski neovisan te se pomoću JVM-a prilagođava za određenu platformu
Polimorfizam	Eksplicitan za metode i podržava mješane hijerarhije	Automatski, koristi statičko i dinamičko povezivanje

Tablica 4.1.1. Usporedba C++ sa Java

16 eduCBA: Best Online Training & Video Courses, < https://www.educba.com/c-plus-plus-vs-java/?fbclid=IwAR1YNf0003P1FUa8lS6w1F9SeGm5EyHDm1VKM164C-hRmLegEAI74E1ABHU> (Pristupljeno 4. rujan 2019.).

4.2. Usporedba C++ sa Python[17]

Osnovna usporedba	Python	C++
Automatsko čišćenje nekorištenih objekata	Podržava automatsko čišćenje nekorištenih objekata	Ne podržava automatsko čišćenje nekorištenih objekata
Korištenje	Jednostavniji za shvaćanje i brzu implementaciju koda	Nije toliko jednostavan za shvaćanje i korištenje, ali je puno moćniji i brži u izvedbi
Brza izrada prototipova	Brza izrada prototipova je moguća zbog jednostavnosti koda	Brza izrada prototipova je teško moguća zbog kompleksnosti koda
Instalacija	Python se lakše koristi na linux platformama nego na windows platformi	Nema problema u instalaciji i korištenju
Doseg varijabli	Varijable su dostupne i izvan bloka naredbi gdje su deklarirane	Varijable su vidljive samo unutar bloka naredbi gdje su deklarirane
Funkcije	Funkcije nemaju nikakvih ograničenja na tip argumenta i na tip povratne vrijednosti	Funkcija može samo prihvatiti i vratiti onakav tip varijable kakav je definiran
Efikasnost kodiranja	Lagan za održavanje, objektno- orjentiran i jednostavan za korištenje	Kompliciraniji za uredno održavanje i kodiranje
Priroda jezika	Dinamički tipiziran jezik	Statički tipiziran jezik

Tablica 4.2.1. Usporedba C++ sa Python

[17] EDUCBA, Differences Between C++ vs Python, https://www.educba.com/python-vs-c-plus-plus/?fbclid=IwAR1m7VBgk_8CJYyiSzT0NxdWn2G73WAtNLZBir37NbX35z-GXyPZwGBjjxs (Pristupljeno 4. rujan 2019.).

4.3. Usporedba C++ sa C#[18]

Osnovne razlike	C++	C#
Veličine binarnih datoteka	C++ će tijekom kompajliranja direktno prevesti izvorni kod u binarne datoteke koje sadrže samo najbitnije. Zato su one manje veličine i kompleksnosti nego u C#.	C# također kompajlira izvorni kod u binarne datoteke, ali će uključiti puno knjižnica i dodatnog koda. Stoga će binarne datoteke generirane od C# biti veće nego one generirane od C++.
Izvedba	C++ je često korišten jezik pogotovo kada jezici visoke razine nisu efikasni budući da se kod napisan u C++ brže izvodi. Na primjer, aplikacije za analizu komunikacijskih mreža moraju biti razvijene u C++ budući da je izvedba ovdje jako bitna.	C# kod je sporiji u izvedbi od C++ koda budući da sadrži puno dodatnog koda i knjižnica. C# možemo koristiti za razvijanje aplikacija u kojima nam izvedba nije toliko bitna.
Automatsko čišćenje nekorištenih objekata	U C++ programiranju programer sam mora brinuti o alokaciji i oslobađanju memorije budući da C++ nema automatsko čišće nekorištenih objekata.	U C# programer se ne mora brinuti oko upravljanja memorijom budući da C# pruža automatsko čišćenje nekorištenih objekata.

[18] EDUCBA, Differences Between C++ vs C#, https://www.educba.com/c-plus-plus-vs-c-sharp/?fbclid=IwAR0iUy7yiEl8frWLMUEZ8CcgmwVNj9xIcEnEb-HIuhHEjJjbisDw-ObZXcY (Pristupljeno 4. rujan 2019.).

Ciljana platforma	C++ programski jezik podržava široku paletu platformi poput windowsa, linuxa, maca itd.	C# programski jezik je microsoftov proizvod te zasad podržava samo windows platformu.
Upozorenja kompajlera	U C++ programskom jeziku, programer može napisati bilokakav kod koji je sintaksno isprava no svejedno taj kod može uzrokovati greške u operacijskom sustavu. Stoga programer mora paziti i na upozorenja kompajlera.	U C# programskom jeziku programer može razvijati kod bez razmišljanja o kompajlerskim upozorenjima budući da se neispravan i opasan kod neće moći kompa

Tablica 4.3.1. Usporedba C++ sa C#

5. IZRADA KVIZA U PROGRAMSKOM JEZIKU C++

Zadatak ovog rada je izraditi programsko rješenje za kviz u c++ programskom jeziku. Sam kviz zasniva se na ispitivanju znanja iz informatičkog područja. Ideja koja stoji iza programa je da natjecatelji odaberu jedan od dva ponuđena načina igre:

1. Svi natjecatelji odgovaraju na ista pitanja
2. Svi natjecatelji odgovaraju na nasumična pitanja

Sva pitanja spremljena su unutar programa i pozivaju se izravno iz programa. Na ponuđena pitanja odgovara se pomoću brojki 1-2-3-4. Nakon unosa odgovora provjerava se točnost odgovora, te slijedi dodjela bodova ako je odgovor točan. Na kraju igre ispisuje se broj bodova te postotak točno odgovorenih pitanja. Pravila kviza su jednostavno i laka za pridržavanje a to su:

1. Svaki igrač odgovara na isti broj pitanja
2. Svaki točan odgovor donosi 1 bod
3. Vremensko ograničenje po pitanju je 15 sekundi
4. Na pitanje se odgovara pomoću brojki 1-2-3-4

Na kraju igre prikazuje se statistika igra, za svakog igrača računa se postotak točno odgovorenih pitanja. Račun se vrši prema navedenoj formuli:

$$p[\%] = \frac{zbroj\ bodova}{broj\ pitanja * \ bod\ po\ pitanju} * 100$$

U navedenoj formuli član „bod po pitanju“ je konstanta te iznosi 1 dok „zbroj bodova“ te „broj pitanja“ su dinamički članovi koji se mijenjaju od igre do igre te od igrača do igrača.

5.1. Knjižne oznake

Kod pisanja programa koristile su se sljedeće knjižne oznake:

5.1.1. #include <iostream>

Standardna knjižna oznaka C++ jezika koja se koristi za unos i ispis podataka. Naredbom **#**include postiže se uključivanje datoteke tj. knjižne oznake u sam program.

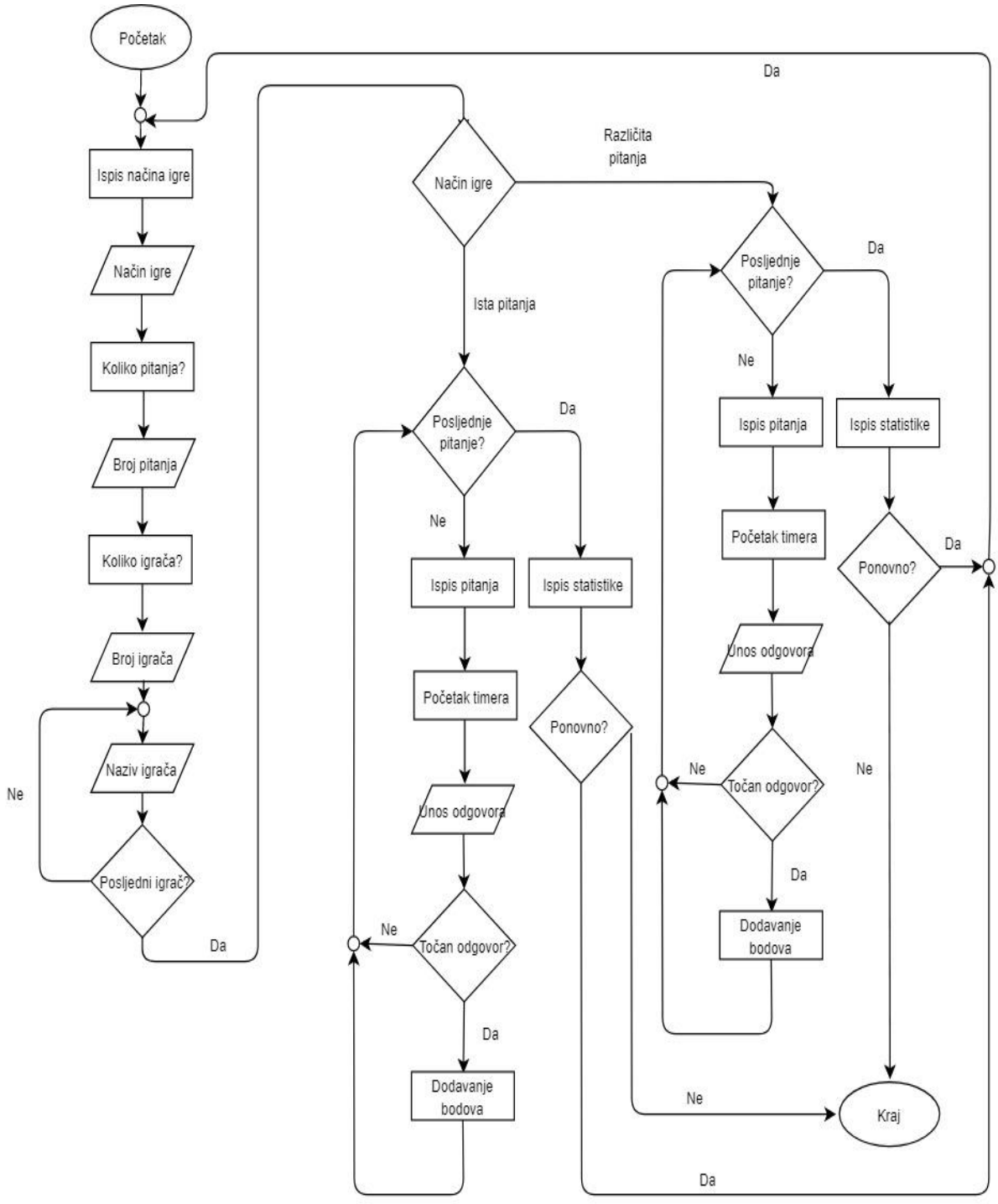
5.1.2. #include <thread>

Korištenjem knjižne oznake thread omogućava se paralelan rad programa na više logičkih jezgri. Konkretno u programu se koristi za rad timera, sam rad timera se odvija na drugoj logički jezgri te kad pošaljemo zahtjev sa prve jezgre on odgovara.

5.1.3. #include <ctime>

Knjižna oznaka koja nam dopušta rad s vremenom. Vrijeme se može mjeriti, zapisivati ili na određeni trenutak pokrenuti određenu funkciju. Sam format zapisa je mm dd yyyy hh:mm:ss

5.2. Dijagram toka programa



Slika 5.2. Dijagram toka programa

5.3. Dijelovi programskog koda

Na samom početku programa potrebno je uključit sve potrebne knjižne oznake također i identifikatore kako bi se spriječila pogreška kod pisanja programa.

Program počinje na slijedeći način:

```
#include<iostream>
#include<ctime>
#include<cstdio>
#include<thread>

Usingnamespace std;
Usingnamespace std::this_thread;
Usingnamespace std::chrono_literals;
Usingnamespace std::chrono:system_clock;
```

Važno je da se navedeni dio koda definira prije funkcije main().
Slijedeći dio programskog koda je definiranje i inicijalizacija potrebnih varijabli. Pozicioniranje koda je unutar funkcije main().

```
int brojIgraca = 0;
int brojPitanja = 0;

string pitanja[30];
string odgovora[30][4];

short int tocOdg[30]
short int nacinIgre=0;

char ponovno;
```

Nakon definiranja svih varijabli ulazi se u logički dio programa. Najprije se ispituje način igre. Ispitivanje se postiglo pomoću IF-ELSE funkcije.

```
  for (int j = 0; j < brojPitanja; j++)
  {
//pokretanje programske rutine kod koje svi igrači dobivaju ista
pitanja
//klica koja nam je potreban za generiranje nasumičnog broja
      srand(time(NULL));
```

```
        nasumicno = rand() % 30 + 1;
//brisanje svega što je trenutno na ekranu ispisano
        system("CLS");
//ispis pitanja koje zavisi do nasumičnog generiranog broja
        cout << pitanje[nasumicno] << endl;
        for (int tr = 0; tr < 4; tr++)
        {
            cout << odgovor[nasumicno][tr] << endl;
        }
        for (int k = 0; k < brojIgraca; k++)
        {
            cout << "Odgovara " << nazivIgraca[k] << " ";
            start = clock();
            cin >> odgovori[k][nasumicno];
            trajanje = (clock() - start) / (double)CLOCKS_PER_SEC;
            if (trajanje > 15) {
                cout << "Ogranicenje po pitanju je 15 sec!" << endl;
                sleep_for(1s);
            }
            else if (odgovori[k][nasumicno] == tocOdg[k]) {
                brojBodova[k] = brojBodova[k] + 1;
            }
            else {
                brojBodova[k] = 0 + brojBodova[k];
            }
            system("CLS");
            cout << pitanje[nasumicno] << endl;
            for (int tr = 0; tr < 4; tr++)
            {
                cout << odgovor[nasumicno][tr] << endl;
            }
        }
```

Nakon same igre potrebno je izračunat statistiku. Vrsta podatka u koju ćemo spremat vrijednost može biti float ili integer. Ukoliko želimo prikazat statistiku sa decimalnom točkom tad ćemo odabrat vrstu podatka float, u suprotnom za cijeli broj odabiremo integer vrstu podatka. Dio koda zadužen za to rješenje je:

```
for (int r = 0;r < brojIgraca; r++) {
//ispis i trenutni izračun bodova prikazan  u postotku
cout << "Broj bodova " << nazivIgraca[r] << " je " << brojBodova[r]
<< " " << float((brojBodova[r] / brojPitanja) * 100) << " % " <<
endl;
  }
```

Poslije prikaza bodova slijedi pitanje od strane programa hoće li igrač ponovno igrat?

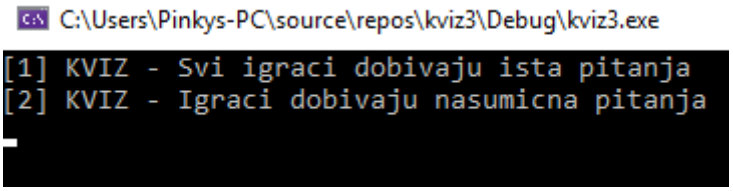
```
cout << "Ponovno igrati DA ili NE (D/N)";
  cin >> ponovno;
  if (ponovno == 'd') {
      system("CLS");
      return main();
  }
```

5.4. Primjena programskog rješenja

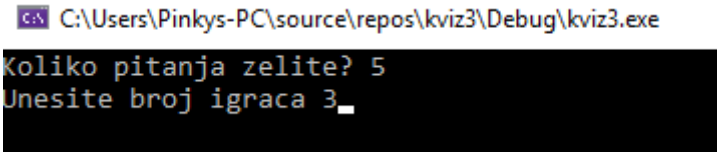
Primjena ovog programa nema ograničenja. Može se koristiti za jednostavne upitnike od strane tvrtke da vidi kako korisnici poznaju brend tvrtke. U školstvu se može upotrijebit kod malih ispita kako bi se provjerilo znanje učenika. Također može se iskoristit kod kviza znanja u svim područjima znanosti.

Ukoliko bi htjeli primijeniti ovaj program u nekim drugim područjima znanosti bilo bi potrebno izmijenit pitanja, tj. potrebno bi bilo pripremit pitanja i odgovore te ih upisat u program kako bi program mogao ispisivat, uspoređivat i ocjenjivat pitanja. Kao što je rečeno sama primjena nema ograničenja i može poslužit u sve svrhe ispitivanja.

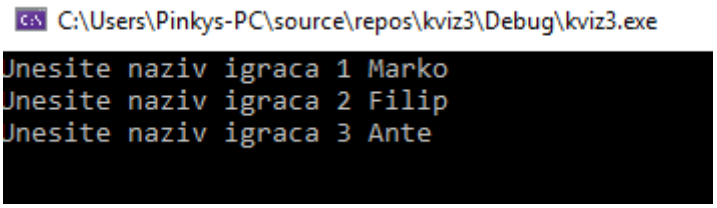
5.5. Izgled pokrenutog programa



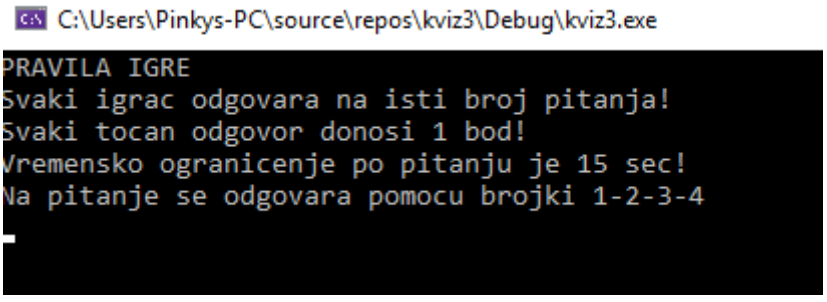
Slika 5.5.1. Odabir vrste kviza



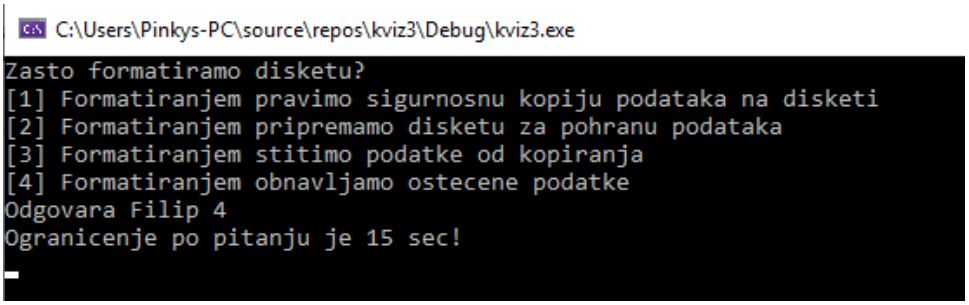
Slika 5.5.2. Odabir broja pitanja i igrača



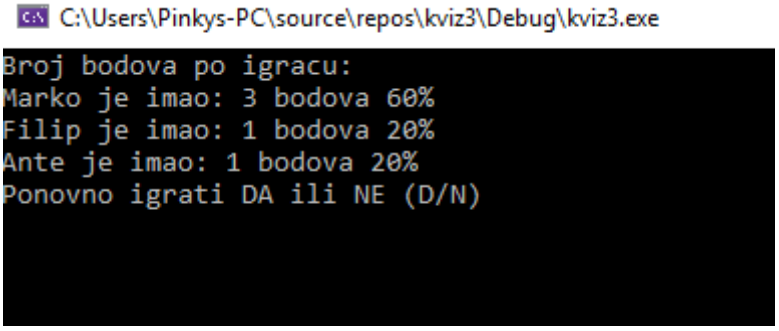
Slika 5.5.3. Unos naziva igrača



Slika 5.5.4. Pravila igre vezana uz kviz



Slika 5.5.5. Prikaz pitanja i odgovora u kvizu



Slika 5.5.6. Broj bodova postignutog u kvizu

6. ZAKLJUČAK

Činjenica je da je računalo danas neophodno u svim područjima života. Sve se više razvija elektroničko poslovanje i upotreba ICT tehnologije. Time se smanjuju mnogi troškovi kojima su izloženi kupci i proizvođači. Većim razvojem ICT-a dolazi i do veće potrebe za programerima te razvojem aplikacija i programa.

Programski jezik C++ vrlo je kompleksan te slovi kao najpopularniji programski jezik današnjice. Mnoge su se aplikacije i programi razvili zahvaljujući njemu. Kod korištenja ovog programskog jezika, kao i kod ostalih, potrebno se pridržavati određenih pravila te logički razmišljati. Prilikom pisanja programa nailazimo na mnogo engleskih termina koje je potrebno prevesti u naredbe razumljive čovjeku. Za to nam služe programi koje nazivamo *compileri*, tj. prevoditelji. Danas postoji veliki broj različitih prevoditelja, iako je, kada je riječ o jeziku C++ najrelevantniji Dev-C++. On je besplatan, lagan za korištenje te omogućuje pokretanje programa.

Kao što je u uvodu rečeno, bez računala ne možemo zamisliti gotovo ni jedan kvalitetno obavljen posao. Svjesni smo činjenice da je računalo stroj koji sve probleme rješava isključivo uz pomoć programa koji u svakom trenutku daju naredbe računalu što treba napraviti. Posao programera je zbog toga ključan kod razvijanja programa i aplikacija. Prilikom pisanja programa koristimo se algoritmima. Algoritam je jednostavan prikaz koji lako razumijemo ukoliko poznajemo taj programski jezik.

Uz programski jezik C++ postoji mnogo kvalitetnih i dobrih programskih jezika, te ukoliko se odlučimo baviti programiranjem nije bitno koji ćemo odabrati. Ukoliko razumijemo jedan programski jezik biti će nam vrlo jednostavno raditi i u drugim programski okruženjima.

POPIS SLIKA

POPIS TABLICA

LITERATURA

Brođanac, P. et al., *Programski jezik C++*, Zagreb, Školska knjiga, 2019.

Lipljin N., *Programiranje 1*, Varaždin, TIVA Tiskara, 2004.

EDUCBA, Differences Between C++ vs Java, https://www.educba.com/c-plus-plus-vs-java/?fbclid=IwAR1YNf0003P1FUa8lS6w1F9SeGm5EyHDm1VKM164C-hRmLegEAI74E1ABHU (pristupljeno 4. rujan 2019.).

EDUCBA, Differences Between C++ vs C#, https://www.educba.com/c-plus-plus-vs-c-sharp/?fbclid=IwAR0iUy7yiEl8frWLMUEZ8CcgmwVNj9xIcEnEb-HIuhHEjJjbisDw-ObZXcY (ristupljeno 4. rujan 2019.).

EDUCBA, Differences Between C++ vs Python, https://www.educba.com/python-vs-c-plus-plus/?fbclid=IwAR1m7VBgk_8CJYyiSzT0NxdWn2G73WAtNLZBir37NbX35z-GXyPZwGBjjxs (pristupljeno 4. rujan 2019.).

Fajković S., *Uvod u programiranje*, 2015., Dostupno na: http://carpediem.hr/PublikacijeCarpeDiem/Publikacije/C++%20programiranje.pdf (pristupljeno 10. rujna 2019.)

Grundler D. i Šutalo S., *Računalstvo*, Zagreb, Školska knjiga, 2019.

Gvozdanović T. et al., *e-u INFO GIM*, Zagreb, PRO-MIL d.o.o., 2018., Dostupno na: www.e-u.hr (pristupljeno 8. rujna 2019.).

Šribar J. i Motik B., *Demistificirani C++*, Zagreb, Element d.o.o. 1997.

SAŽETAK

U današnje doba sve više ljudi je upoznato s programiranjem, odnosno koristi se nekim od programskih jezika (C++,C#, Python, Java, itd..). Programski jezici olakšavaju i pojednostavljuju programerima njihov posao i daju jasne upute računalima. Različiti tipovi programskih jezika imaju i različitu svrhu namjene i postoje na različitim razinama. Za programiranje se može reći da je jedna od najtraženijih vještina danas te svatko tko je malo upućen u programiranje zasigurno je čuo za C++ programski jezik koji daje ogroman utjecaj u računalnom svijetu. Kao najpoznatiji programski jezik koji se u današnje vrijeme najviše i koristi možemo spomenuti programski jezik C++. C++ je programski jezik opće namjene i srednje razine s podrškom za objektno orijentirano programiranje. U svojoj srži C++ programski jezik niz je uputa koje programer upisuje na određeni način, držeći se pritom pravila i poštujući opću logiku. Također C++ programski jezik karakterizira brzina i efikasnost pri zauzimanju resursa. Kao veliku prednost programskog jezika može se spomenuti njegova složenost i kompleksnost, te velike mogućnosti koje pruža u računalom svijetu. U ovom radu prikazana je izrada programskog rješenja za kviz u C++ programskom jeziku.

Ključne riječi: programiranje, računalo, programski jezici, programski jezik C++

ABSTRACT

Today, more and more people are familiar with programming, that is, using some of the programming languages (C ++, C #, Python, Java, etc.). Programming languages make it easy for developers to do their jobs and give clear instructions to computers. Different types of programming languages also have different purpose and exist at different levels. To program it can be said that one of the most sought-after skills today, and anyone who is little versed in programming certainly heard from a C ++ programming language, which gives a huge impact in the count but m world. As the most well-known programming language most used today, we can mention the C ++ programming language. C ++ is a programming language general purpose and intermediate level with support for object oriented programming . At its core C ++ programming language is a set of instructions that a programmer enrolled in a certain way, governed by the rules and post offices stipulating general logic. The C ++ programming language is also characterized by speed and efficiency in taking up resources. One of the great advantages of programming language is its complexity and complexity, as well as its great possibilities in the computer world. This paper describes the development of a software solution for a quiz in C ++ programming language .

Keywords: programming, computer, programming languages, C ++ programming language