Struktura programa, tipovi podataka, operatori

predavač: Nadežda Jakšić

Programiranje programski jezik C++

Razvoj programa

- projektni zadatak
- projektovanje programa (algoritmi)
- pisanje programskog koda, izvorni kod, source code

  (sintaksne i semantičke greške)
- kompajliranje, prevođenje izvornog koda u nule i jedinice; dobija se objektni kod; nešto.OBJ
- linkovanje, povezivanje; na izlazu se dobija izvršna datoteka; nešto.EXE
- izvršavanje programa
- testiranje
- održavanje i modifikacija programa

Code::Blocks

- integrisano razvojno okruženje (IDE) otvorenog koda (free)
- C, C++, Fortran
- instalacija www.codeblocks.org/downloads
  izabrati Download the binary release
  izabrati codeblocks -17.12 mingw-setup.exe

Struktura programa

```
// moj prvi C++ program - komentar u jednom redu
#include <iostream> //pretprocesorske direktive
/*main je glavna funkcija u
programu i može da bude samo jedna - komentar u više redova*/
int main()
{
    std::cout << "Zdravo!"; //svaka naredba se završava ;
return 0;
}

//ili

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

Podaci

- konstanta - podatak koji ne menja vrednost (npr. 5, 10, 3.14159)
- promenljiva - lokacija u memoriji na kojoj mogu da se nalaze različite vrednosti u toku izvršavanja programa - promenljiva se u izvornom kôdu predstavlja svojim imenom (identifikatorom), a ne svojom vrednošću
- identifikatori - imena koja daje programer delovima programa (promenljivima, konstantama, funkcijama…)
- pravila za zadavanje identifikatora:
  - kombinacija slova engleskog alfabeta (A - Z, a - z), cifara (0 - 9) i znaka donja crta _
  - prvi znak mora da bude slovo ili znak donja crta,
  - ne sme da bude rezervisana reč (main, include, asm, auto, bool...)

Podaci

- izbegavati da identifikator sadrži dvostruke znakove donja crta (_ _) ili da počinje znakom donja crta i velikim slovom, jer su takve oznake rezervisane za C++ implementacije i standardne biblioteke
- zbog razumljivosti kôda poželjno je identifikatore odabrati tako da upućuju na stvarno značenje promenljive
- dve notacije za pisanje identifikatora, kada se identifikator sastoji od više reči:

  porezNaPromet
  porez_na_promet

- C++ pravi razliku između malih i velikih slova (identifikator a se razlikuje od identifikatora A)

Tipovi podataka

- char - znak ili mali broj, jedan bajt (’a’, ’z’,’9’,…)
- int - ceo broj, dva ili četiri bajta (1,42, -4578,2…)
- float - realan broj u jednostrukoj tačnosti (3.14 ili 299793.0)
- double - realan broj u dvostrukoj tačnosti
- bool - jedan bajt, može da ima vrednost true ili false

- ako se ne navede tip, podrazumeva se int
- moguće je dodati rezervisane reči short, long, signed i unsigned radi smanjenja i povećanja opsega (unsigned – samo pozitivne vrednosti)
- opseg dozvoljenih vrednosti zavisi od kompajlera
- veličina tipa na određenom računaru se može utvrditi operatorom sizeof (npr. sizeof (int))

Opsezi - primer

- char - znak ili mali broj, 1 bajt, od -128 do 127
- short - mali ceo broj, 2 bajta, od –32 768 do 32 767
- unsigned short int, 2 bajta, od 0 do 65 535
- int - ceo broj, 2 ili 4 bajta, ako je 4 opseg od –2 147 483 648 do 2 147 483 647
- long – dugačak ceo broj, 4 bajta
- unsigned long int – 4 bajta, od 0 do 4 294 967 295
- float - realan broj u jednostrukoj tačnosti, 4 bajta, od $-3{,}4\cdot10^{38}$ do $-3{,}4\cdot10^{-38}$ i $3{,}4\cdot10^{-38}$ do $3{,}4\cdot10^{38}$
- double - realan broj u dvostrukoj tačnosti, 8 bajtova, od $-1{,}7\cdot10^{308}$ do $-1{,}7\cdot10^{-308}$ i $1{,}7\cdot10^{-308}$ do $1{,}7\cdot10^{308}$
- long double - 10 bajtova, od $-1{,}1\cdot10^{4932}$ do $-3{,}4\cdot10^{-4932}$ i $3{,}4\cdot10^{-4932}$ do $1{,}1\cdot10^{4932}$
- bool - logička vrednost, 1 bajt, može da ima vrednosti true ili false
- proveravanje dužine:

```
cout << "Veličina celih brojeva: " << sizeof(int);
cout << "Veličina realnih brojeva: " << sizeof(float);
```

Deklarisanje promenljivih

tip identifikator;

```
int a;
int a,b;
float a;

double porezNaPromet;
char c;
bool t;
```

Konstante

- konstantni tip ima sve osobine osnovnog tipa, samo se objekti konstantnog tipa ne mogu menjati
- pristup konstantama kontroliše se u fazi prevođenja, a ne u fazi izvršavanja – prevodilac često ne odvaja memorijski prostor za konstantu, već njeno korišćenje razrešava u trenutku prevođenja
- deklarisanje i inicijalizacija (konstanti se dodeljuje identifikator i vrednost)

  const double PI=3.14159265359;

  o=2*r*PI; umesto o=2*r*3.14159265359;

- ako se pokuša promena vrednosti konstante, prevodilac će prijaviti grešku

Operator dodele, znak =

```
#include <iostream>
using namespace std;
int main ()
{
    int a, b;   // a:?,  b:?
    a = 10;   // a:10, b:?
    b = 4;    // a:10, b:4
    a = b;    // a:4,  b:4
    b = 7;    // a:4,  b:7
    return 0;
}
```

inicijalizacija promenljivih – dodeljivanje početnih vrednosti pri deklarisanju

```
int a=5;
double b=3.257;
char c= 'M';
```

Izlazni tok Cout

```
int main()
{
cout<<"Mozemo li da ispisujemo i brojeve?";
cout<<154; <<endl; //endl je konstanta koja prebacuje ispis u novi red
cout<<"Kombinacija teksta i brojeva? " <<5<< "  je dobar broj!";
return 0;
}

int main ()
{
    int a, b;
    a = 10;
    b = 4;
    a = b;
    b = 7;
    cout << "vrednost promenljive a je"<<a<<endl;
    cout << " vrednost promenljive b je "<<b;
return 0;}
```

Ulazni tok Cin

```
int a;
cout<<"Unesi vrednost promenljive a"<<endl;
cin>>a;
```

- ulazni tok (input stream) cin je zajedno sa izlaznim tokom definisan u datoteci zaglavlja iostream
- služi za učitavanje podataka sa tastature
- operatorom >> podaci sa tastature se upućuju u memoriju i smeštaju u odgovarajuću promenljivu
- učitavanje počinje tek kada se na tastaturi pritisne taster za novi red tj. taster Enter, to znači da kada pokrenete program, nakon ispisane poruke možete unositi bilo šta i to po potrebi brisati – tek kada pritisnete taster Enter, program će analizirati unos i sačuvati

Formatiranje izlaza

funkcije i manipulatori jezika C++ za formatiranje izlaza su definisani okviru klasa ostream i ios
nekoliko opcija formatiranja koje se mogu koristiti samostalno ili u kombinaciji sa drugim:

- fixed - koristi decimalni zapis za štampanje vrednosti
- scientific - koristi eksponencijalni zapis za štampanje vrednosti
- setprecision (int) – podešava preciznost za ispis realnih brojeva (definisan u iomanip)

- u slučaju upotrebe fiksnog ili eksponencijalnog zapisa, preciznost se odnosi na broj decimalnih mesta koja se koriste za ispis decimalnog dela
- u slučaju da je preciznost manja od broja cifara, izvršiće se zaokruživanje broja

Formatiranje izlaza

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    cout << fixed << endl;
    cout << setprecision (3) << 123.456 << endl;
    cout << setprecision (4) << 123.456 << endl;
    cout << setprecision (5) << 123.456 << endl;
    cout << setprecision (6) << 123.456 << endl;
    cout << setprecision (7) << 123.456 << endl;
    cout << scientific << endl;
    cout << setprecision (3) << 123.456 << endl;
    cout << setprecision (4) << 123.456 << endl;
    cout << setprecision (5) << 123.456 << endl;
    cout << setprecision (6) << 123.456 << endl;
    cout << setprecision (7) << 123.456 << endl;
return 0;
}
```

Formatiranje izlaza

Manipulatori:

- internal - levo poravnanje znaka, i desno poravnanje vrednosti
- left - levo poravnanje znaka (+/-) i vrednosti
- right - desno poravnanje znaka (+/-) i vrednosti
- setfill (char) - postavlja karakter koji ispunjava prazna polja u okviru formata za štampanje (definisan u iomanip)
- setw (int) - postavlja širinu polja za unos i ispis parametara (definisan u iomanip)

```
cout << -12345 << endl;
cout << setw (10) << -12345 << endl
cout << setw (10) << left << -12345 << endl;
cout << setw (10) << right << -12345 << endl
cout << setw (10) << internal << -12345 << endl;
```

Operatori

Aritmetički operator	značenje
+	sabiranje
-	oduzimanje
*	množenje
/	celobrojno deljenje
%	ostatak pri celobrojnom deljenju

Složeni operatori dodele (+=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=)

izraz	jednako sa...
y += x;	y = y + x;
x -= 5;	x = x - 5;
x /= y;	x = x / y;
price *= units + 1;	price = price * (units+1);

Operatori

operatori umanjivanja, odnosno uvećavanja za jedan (++,--) prefiksni i postfiksni

```
x = 3;
y = ++x;  // x je 4, y je 4
x = 3;
y = x++;  // x je 4, y je 3
```

relacioni operatori - ispitivanje uslova, rezultat ispitivanja je uvek true ili false (nula se tumači kao false, a bilo koja nenulta vrednost kao true)

= =	jednako
! =	različito
<	manje od
>	veće od
<=	manje ili jednako
>=	veće ili jednako

Primer: ako je a=2, b=3, c=6

```
(a == 5)     // false
(a*b >= c)   // true
(b+4 > a*c)  // false
((b=2) == a) // true
```

Operatori

Logički operatori

!	logičko NE
&&	logičko I
||	logičko ILI

Primer:

```
!(5 == 5)   // false
!(6 <= 4)   // true
!true       // false
!false      // true
( (5 == 5) && (3 > 6) )  // false ( true && false )
( (5 == 5) || (3 > 6) )  // true ( true || false )
```

Logički operatori

```
#include<iostream>
using namespace std;
int main()
{
bool a,b,c,d,e;
cout<<"Vrednost logickog podatka A="; //uneti nulu ili jedan
cin>>a;
cout<<"Vrednost logickog podatka B="; //uneti nulu ili jedan
cin>>b;
c = !a;
d = a&&b;
e = a||b;
cout<<endl<<"Ako je logicki podatak A="<<a<<" tada je suprotno od
    A="<<c<<endl;
cout<<"Za A = "<<a<<" i B="<<b<<" (A I B)= " <<d<<endl;
cout<<"Za A = "<<a<<" i B="<<b<<" (A ILI B)= " <<e<<endl;
return 0;
}
```

Operatori za rad sa bitovima

Operatori za manipulisanje bitovima

<< logičko šiftovanje ulevo (za n mesta, množenje sa 2^n)
>> logičko šiftovanje udesno (za n mesta, deljenje sa 2^n)
& bitsko I (1 & 0 = 0, 0 & 0 = 0, 1 & 1 = 1, 0 & 1 = 0)
| bitsko ILI (1 | 0 = 1, 0 | 0 = 0, 1 | 1 = 1, 0 | 1 = 1)
~ potpuni komplement (1 postaje 0, 0 postaje 1, plus 1 na mestu najmanje težine)
^ bitsko ekskluzivno ILI (1 ^ 0 = 1, 0 ^ 1 = 1, 1 ^ 1 = 0, 0 ^ 0 = 0)

Operatori za rad sa bitovima

```
main() {
   unsigned int x = 10;    // dekadni broj 10 = 00001010
   unsigned int y = 5;         // dekadni broj 5=00000101
   int z = 0;
   z = x & y;             // 00000000 dekadni broj 0
   cout << "Vrednost promenljive z je : " << z << endl ;
   z = x | y;             // 00001111 dekadni broj 15
   cout << "Vrednost promenljive z je: " << z << endl ;
   z = x ^ y;             // 00001111 dekadni broj 15
   cout << "Vrednost promenljive z je: " << z << endl ;
   z = ~x;                // 11110101 dekadni broj -11
   cout << "Vrednost promenljive z je: " << z << endl ;
   z = x << 2;            // 00101000 dekadni broj 40
   cout << "Vrednost promenljive z je: " << z << endl ;
   z = x >> 2;            // 00000010 dekadni broj 2
   cout << "Vrednost promenljive z je: " << z << endl ;
   return 0;
}
```

sizeof operator

vraća veličinu u bajtovima, x = sizeof (char);

```
cout << "The size of char : " << sizeof (char); //1
cout << "\nThe size of int : " << sizeof (int); //4
cout << "\nThe size of short int : " << sizeof (short int); //2
cout << "\nThe size of long int : " << sizeof (long int); //4
cout << "\nThe size of float : " << sizeof (float); //4
cout << "\nThe size of double : " << sizeof (double); //8
cout << "\nThe size of unsigned long : " << sizeof (unsigned long); //4
cout << "\nThe size of long long : " << sizeof (long long); //8
cout << "\nThe size of long double: " << sizeof (long double); //12
cout << "\nThe size of bool: " << sizeof (bool); //1
```

Eksplicitna konverzija tipova

```
static_cast <oznaka tipa>(izraz) // između numeričkih tipova
const_cast <oznaka tipa>(izraz)  //namenjen uklanjanju ili dodavanju const tipa
dynamic_cast <tip_pokazivaca_ili_ref> (izraz) //za podatke dinamičkog tipa

int main()
{
float a = 5.256;
int b=10, c=11, rez;
int d = static_cast <int> (a);
cout <<d<< endl; //5
float e = static_cast <float> (b);
cout <<e<< endl; //10
float rezf = static_cast <float> (c/b);
cout <<rezf<< endl; //10
return 0;}
```

Implicitna konverzija tipova

- ako su oba operanda istog tipa, takvog je tipa i rezultat
- ako su operandi različitih tipova, svode se na zajednički tip (uobičajeno složeniji tip) pre operacije - npr. ako je jedan operand int, a drugi float, oba se pretvaraju u složeniji tip, a to je float, a takav je i rezultat

```
int a=5,x=2;
float b=3.;
float rez=a*x/b;
cout<<rez; //rezultat ce biti tipa float 3.33333

int a,x;
    float b;
    a=43; b=1.1;
    x=a/b; //promenljiva x će imati vrednost 39

int a,b;
    float x;
    a=43; b=11;
    x=a/b; //promenljiva x će imati vrednost 3
```

Postepeno izvršavanje programa u Code::Blocks-u

- program ne mora da se izvršava u celini, već može da se zaustavi izvršavanje na bilo kojoj naredbi
- takvo mesto se zove tačka prekida (engl. breakpoint) i označeno je crvenim krugom
- pozicioniranje na liniju u programu u kojoj se postavlja tačka prekida:
  - Debug/Toggle Breakpoint ili
  - desni klik unutar editora, a zatim Toggle Breakpoint ili
  - klik u sivi prostor sa leve strane kôda, ali desno od rednog broja linije ili
  - taster F5

- isti redosled akcija deaktivira tačku prekida
- pojavljuje se crveni kružić
- to je lokacija na koju može da se klikne da bismo (de)aktiviriali tačku prekida
- sada treba pokrenuti program izborom Debug/Start ili tasterom F8

Postepeno izvršavanje programa

- ako se izvršava samo jedna naredba, taster F7, ili jedna od ovih strelica u ToolBar-u

- ako se izvršavanje zaustavlja kod kursora, taster F4
- dok se program izvršava deo po deo, žuta strelica na levoj strani pokazuje naredbu koja će sledeća biti izvršena
- ako hoćemo da vidimo vrednosti promenljivih koje sadrže usputne rezultate, opcija Debug / Debugging windows / Watches
- pojaviće se prozor za nadgledanje promenljivih
- desni klik na ovaj prozor / opcija Add watch
- u prozoru koji se pojavi uneti naziv promenljive koja se nadgleda
- u prozoru Watch se mogu uneti nazivi promenljivih jedan ispod drugog kao i jednostavniji izrazi čije vrednosti želimo da vidimo

Postepeno izvršavanje programa

- u prozoru Watch crvenom bojom se prikazuju vrednosti koje su promenjene između prethodnog i tekućeg zaustavljanja programa
- ako u bilo kom trenutku želimo da izvršimo program do kraja (ili do sledeće tačke prekida, ako takva postoji), pritisnuti F8, isto kao da program nije ni počeo sa radom
- ako želimo da potpuno prekinemo izvršavanje - Shift-F8 ili dugme sa crvenim kvadratom ("Stop debugger ") u ToolBar-u

- najkorisnija primena svih ovih mogućnosti je traženje i ispravljanje grešaka u programu
- ako je trenutna konfiguracija "Release ", možda će postojati problem sa gledanjem vrednosti promenljivih - u standardnom ToolBaru konfiguraciju promeniti na Debug, i ponovo pokrenuti program
- Settings/Compiler/kartica Toolchain executables/Resource compiler – izabrati gdb32.exe na putanji CodeBlocks/MinGW/bin
- Settings/Debugger, Default, u polju Executable path postaviti istu putanju za gdb32.exe