Tema 02

Tipovi i strukture podataka, izrazi i upravljačke naredbe u jeziku C++

Prof. dr Miodrag Živković

Tehnički fakultet

OBJEKTNO ORIJENTISANO PROGRAMIRANJE 2

Sadržaj

1. Uvod
2. Ugrađeni tipovi podataka u jeziku C++
3. Operatori i izrazi u jeziku C++
4. Osnovne upravljačke naredbe u jeziku C++
5. Strukture podataka u jeziku C++
6. Primeri programa

1. Uvod

1. Elementi proceduralnih programskih jezika
2. Identifikatori i specijalni znakovi u jeziku C++
3. Promenljive u jeziku C++

1.1 Elementi proceduralnih programskih jezika

- Osnovni elementi proceduralnih programskih jezika su programske naredbe, koje načelno odgovaraju rečenicama prirodnog jezika
- Naredbe u jeziku C++ završavaju znakom ";", npr.
  - deklarativne naredbe
    ```
    int x;
    ```
  - naredba dodele vrednosti
    ```
    x = 10;
    ```
  - ulazno-izlazne naredbe
    ```
    cout << x;
    ```
  - ostale naredbe: upravljačke, definicije struktura, ...

Elementi proceduralnih programskih jezika

- Elementi naredbe za dodelu vrednosti su promenljive i izrazi
- Izrazi se sastoje od *promenljivih*, *konstanti*, *operatora* i poziva *funkcija*
  - funkcije su imenovane grupe naredbi, koje se mogu višestruko upotrebiti i program čine jasnijim i lakšim za održavanje
  - glavni program u jeziku C++ se definiše kao funkcija main()
- Za predstavljanje podataka koriste se
  - osnovni tipovi podataka (int, float, double, bool , char) koji odgovaraju pojedinačnim vrednostima
  - strukture podataka, gde je više vrednosti zapamćeno pod jednim nazivom. Zapamćene vrednosti mogu biti istog tipa (npr. polja) ili različitih tipova (korisnički definisani tipovi)

Elementi proceduralnih programskih jezika

- Za realizaciju bilo kog algoritma u proceduralnom programskom jeziku dovoljna su tri načina upravljanja izvršavanjem pojedinih koraka algoritma
  - sekvencijalno izvršavanje, jedan korak za drugim (podrazumeva se)
  - uslovno izvršavanje, gde naredni korak zavisi od određenih uslova
  - ponavljanje, gde se određeni niz koraka izvršava više puta

1.2 Identifikatori i specijalni znaci u jeziku C++

- Elementi jezika imaju nazive (identifikatore), koji se formiraju od slova engleske abecede (A..Z i a..z), cifara 0..9 i znaka "_", s tim da prvi znak mora biti slovo ili donja crta
  - najveća dužina identifikatora je ≥ 1024 karaktera (Microsoft C++ 2048)
- Specijalni znaci (*escape sequences*) navode se posebnim oznakama, koje počinju obrnutom kosom crtom, npr.

Specijalni znak	Naziv	Opis
\n	newline	Pomera kursor u sledeći red
\t	tab	Pomera kursor na sledeći tabulator
\a	alarm	Računar se oglašava
\b	backspace	Vraća kursor za jednu poziciju unazad
\r	carriage return	Pomera kursor na početak tekućeg reda
\\	backslash	Prikazuje obrnutu kosu crtu
\'	single quote	Prikazuje apostrof
\''	double quote	Prikazuje navodnik
\?	question mark	Prikazuje znak pitanja

1.3 Promenljive u jeziku C++

- Naredbe u jeziku C++ kreiraju, upotrebljavaju i brišu različite programske objekte. Svaki objekt zauzima odgovarajući prostor u radnoj memoriji. Većina objekata se koristi u obliku imenovanih objekata - promenljivih
- Promenljive se kreiraju deklarativnim naredbama za definisanje promenljivih, npr.

```
int x
float y
char z
```

kojima se samo rezerviše određeni deo memorije za promenljivu zadanog naziva i tipa vrednosti

Promenljive u jeziku C++

- U jeziku C++ ne vrši se automatska *inicijalizacija* deklarisane promenljive, već je to odgovornost programera
- Razlog su performanse, mada rad s neinicijalizovanim promenljivima daje nepredvidljive rezultate, npr.

```
int z;
cout << z << endl;
```

- Prema konvenciji, naziv promenljive počinje malim slovom, a ako se sastoji od kombinacije više reči, one se odvajaju donjom crtom ili velikim početnim slovima (*CamelCase*)
- Imena treba da budu *opisna*, tako da treba izbegavati trivijalna (*i*, *x*) i suviše opšta imena (*brojac*, *podatak*)

2. Ugrađeni tipovi podataka u jeziku C++

1. Promenljive i konstante
2. Celobrojni podaci (int)
3. Znakovni podaci (char)
4. Brojevi u pokretnom zarezu (float, double)
5. Logički tip (bool)

Osnovni tipovi podataka i proširenja

- Jezik C++ podržava sedam osnovnih tipova podataka:
  - char - niz znakova kodirnih pomoću 8 bita
  - wchar_t - niz znakova kodiranih s više od 8 bita (*wide character*)
  - int - celi broj
  - float - decimalni broj
  - double - decimalni broj dvostruke preciznosti
  - bool - logička vrednost
  - void - bez vrednosti
- Predviđena je mogućnost proširenja osnovnih tipova pomoću *modifikatora* (signed, unsigned, long i short), koji se navode ispred naziva tipa

Dozvoljene kombinacije osnovnih tipova podataka i modifikatora

Tip	Veličina u bajtovima	Opseg definisan ANSI/ISO standardom
`bool`	1	`true` ili `false`
`char`	1	`-128` do `127`
`unsigned char`	1	`0` do `255`
`signed char`	1	`-128` do `127`
`int`	4	`-2 147 483 648` do `2 147 483 647`
`unsigned int`	4	`0` do `4 294 967 295`
`signed int`	4	isto kao `int`
`short int`	2	`-32 768` do `32 767`
`unsigned short int`	2	`0` do `65 535`
`signed short int`	2	isto kao `short int`
`long int`	4	`-2 147 483 648` do `2 147 483 647`
`signed long int`	4	isto kao `long int`
`unsigned long int`	4	`0` do `4 294 967 295`
`float`	4	$\pm 3{,}4 \times 10^{\pm 38}$ sa tačnošću od približno 7 cifara
`double`	8	$\pm 1{,}7 \times 10^{\pm 308}$ sa tačnošću od približno 15 cifara
`long double`	8	isto kao `double`

Veličina u bajtovima zavisi od arhitekture računara i prevodioca.
Standard definiše *minimalnu* veličinu

Deklaracija i definisanje promenljive

- Deklaracija svake promenljive u programu je obavezna, pa je jezik C++ *strogo tipiziran* jezik
- Deklaracija promenljive u jeziku C++ može se navesti *bilo gde* u programu pre prve upotrebe promenljive
- *Deklarisanje* promenljive vrši se navođenjem tipa i naziva promenljive, npr.

```
int tezina; // deklaracija celobrojne promenljive
```

(deklarisanje ne podrazumeva inicijalizaciju)

- *Definisanje* promenljive podrazumeva i *inicijalizaciju*, npr.

```
int tezina = 70;   // definisanje promenljive
int tezina = {70}; // definisanje promenljive u C++11
```

2.1 Promenljive i konstante

- Promenljive su podaci određenog tipa čija se vrednost može menjati u toku izvršavanja programa
- Konstante (literali) su vrednosti određenog tipa koje se ne menjaju u toku izvršavanja programa, npr.
  - vrednosti tipa char: 'A', 'z', '8', '*'
  - vrednosti tipa int: -77, 65, 0x9FE
  - vrednosti tipa unsigned int: 10U, 64000U
  - vrednosti tipa long: -77L, 65L, 12345L
  - vrednosti tipa unsigned long: 12345UL
  - vrednosti tipa float: 3.14f
  - vrednosti tipa double: 3.14
  - vrednosti tipa bool: true, false

2.2 Celobrojni podaci (int)

- Celobrojni tip podataka za predstavljanje pozitivnih i negativnih celih brojeva koristi najmanje 2, a tipično 4 bajta
  - short int 2 bajta
- Većina procesora podržava celobrojnu aritmetiku u punom komplementu, gde je 16-bitni broj u rasponu -32768..32767
- Standardni celobrojni tip je označen (*signed*), a modifikator *unsigned* treba koristiti samo za predstavljanje pozitivnih vrednosti (prevodilac to ne kontroliše)
- Celobrojne konstante su standardno tipa int, pa se tip može izostaviti, npr.

```
brojSoba = 300;  // promenljiva brojSoba je tipa int
```

Tip celobrojnog literala

- Tip celobrojnog literala može se *definisati* i pomoću tipa dodeljene vrednosti, npr.

```
brojSpratova = 32L;
brojSoba = 300L;
brojApartmana = 10u;
```

- Kada tip literala nije zadat, prevodilac jezika C++ koristi najmanji celobrojni tip koji može da predstavi vrednost, najčešće je to tip int
- Konstante se mogu zadati i *heksadecimalno* i *oktalno*, npr.

```
 int cetrnaestPatuljaka = 0x0E; // broj 14 (decimalno)
int SnezanaIPatuljci = 010;    // broj 8 (decimalno)
```

(heksadecimalne vrednosti imaju prefiks 0x, a oktalne 0)

2.3 Znakovni podaci (char)

- Znakovni tip podataka smešta se u bajtove, koji se mogu interpretirati i kao celi brojevi

```
char jednoSlovo = 'A'
jednoSlovo = 65 // ASCII kod slova A
```

- Program:

```
int main() {
    char jednoSlovo;
    jednoSlovo = 65;
    cout << jednoSlovo << endl;
    jednoSlovo = 'B';
    cout << jednoSlovo << endl;
    return 0
}
```

daje rezultat:

```
A
B
```

Tip string

- U jeziku C++ postoji tip *literala* string za niz znakova zadanih između dvostrukih navodnika, npr.

```
"dobar dan"
"ovo je string"
```

- Jezik C++ samo koristi *konstante* ovog tipa, ali nema ugrađeni tip podatka string, već su nizovi znakova strukture podržane preko biblioteke klasa, u kojoj postoji klasa string

- Napomena:
  - znakovni tip wchar_t koristi se za zapis znakova jezika kao što je kineski, gde 8 bita nije dovoljno za predstavljanje jednog znaka, već se koristi veći broj bita, fiksni ili promenljiv

2.4 Brojevi u pokretnom zarezu (float, double)

- Decimalne vrednosti se mogu zadavati u decimalnoj ili eksponencijalnoj notaciji, npr. isti broj u dva zapisa

```
123.123
1.23123e2
```

  - decimalni zapis mora da sadrži decimalnu tačku ili eksponent
- Decimalni broj se može deklarisati kao *float* ili *double*, pri čemu je opseg brojeva tipa *double* za red veličine veći
- Tip *double* se zbog većeg raspona vrednosti koristi češće, a koristi ga i većina matematičkih funkcija u C++ biblioteci

```
float pi = 3.1415;
 float naelektrisanjeElektrona = 1.6e-19;
 double masaSunca = 2e30
```

2.5 Logički tip (bool)

- Koristi se za smeštanje dveju mogućih logičkih vrednosti, istina (*true*) i laž (*false*). Vrednost istina je bilo koja vrednost različita od nule, dok je vrednost laž jednaka nuli. Program...

```
#include <iostream>
using namespace std;
int main() {
  bool b = false; // laž
  cout << "b je" << b << endl;
  b = true;        // istina
  cout << "b je" << b << endl;
  return 0;
}
```

... daje rezultat:

```
b je 0
b je 1
```

3. Operatori i izrazi u jeziku C++

1. Pregled operatora u jeziku C++
2. Evaluacija izraza i prioritet operatora
3. Konverzija tipova podataka (cast)
4. Imenovane konstante

3.1 Pregled operatora u jeziku C++

- Operator dodele vrednosti
- Aritmetički operatori
- Relacioni i logički operatori
- Definisanje sinonima za tipove podataka (typedef)
- Operator sizeof
- Operator nabrajanja (,)
- Osnovne ulazno-izlazne operacije

Operator dodele vrednosti

- Osnovna forma naredbe za dodelu vrednosti je

```
promenljiva = izraz;
```

- Dozvoljena je višestruka dodela ili nizanje operatora, npr.

```
int x, y, z;
x = y = z = 100;
```

  - operator menja *vrednost* objekta, ali *tip* objekta ostaje isti

- S leve strane operatora mogu biti samo promenljivi objekti (*lvalues*) , a s desne strane i konstante (*rvalues*) , npr.

```
3.1415 = pi;  // greška s leve strane
int i;
i = i+5;
int j = 5;
```

Aritmetički operatori

- U jeziku C++ definisani su aritmetički operatori "+", "-", "*" i "/", koji se mogu primeniti na sve numeričke tipove podataka, kao i na podatke tipa char
- Deljenje je celobrojno kad se primeni na celobrojni tip podataka. Operator % (modul) daje ostatak celobrojnog deljenja i ne primenjuje se na ostale tipove vrednosti
- Operatori inkrementiranja i dekrementiranja su "++" i "--".

```
int i = 0;
++i;
cout << i;  // ispisuje 1
--i;
cout  <  i; // ispisuje 0
      <
```

Operatori inkrementa i dekrementa

- Operator inkrementa x++ vrši operaciju x = x+1, a operator dekrementa x-- operaciju x = x-1
- Operatori se mogu koristiti *prefiksno* i *postfiksno*, odnosno ispred i iza operanda
- Kada je operand *ispred* promenljive, prvo se vrednost promeni, a zatim se dalje u izrazima koristi nova vrednost
- Kada je operand *iza* promenljive, prvo se postojeća vrednost upotrebi u izrazu, a nakon toga se promeni, npr.

```
int x = 1, y;
y = ++x; // vrednost y je 2, a x je 2
y = x--; // vrednost y je 2, a x je 1
```

Provera (mali test): Evaluacija operatora inkrementa i dekrementa

(a) Konačno x kad je promenljiva s obe strane operatora dodele?

```
int x = 1;
x = x++;
```

(b) Koja vrednost će se prikazati na ekranu?

```
#include <iostream>
use namespace std;
int main() {
  int x, a=2, b=4, c=5;
  x = a-- + b++ - ++c;
  cout << "x = " << x << endl;
  return 0
}
```

Skraćena dodela vrednosti

- Operatori inkrementa/dekrementa mogu se koristiti za skraćeni zapis naredbe dodele vrednosti u obliku naredbe:

  leva_strana operator = *desna_strana*

  što je ekvivalentno naredbi:

  leva_strana = *leva_strana operator desna_strana*

- Mogu se koristiti sledeći aritmetički i logički operatori:

  +, -, *, /, %, <<, >>, &, ^, >

- Ovo svojstvo je preuzeto iz jezika C (efikasniji izvršni kod)

  Sledeće tri naredbe su ekvivalentne:

```
x = x + 1; // standardna dodela vrednosti
x++;       // standardni inkrement
x+=1;      // skraćena dodela vrednosti
```

Relacioni i logički operatori

- Relacioni operatori omogućavaju međusobno poređenje različitih vrednosti:

x > y	*da li je x veće od y?*
x < y	*da li je x manje od y?*
x >= y	*da li je x veće ili jednako y?*
x <= y	*da li je x manje ili jednako y?*
x == y	*da li je x jednako y?*
x != y	*da li je x različito od y?*

- U jeziku C++ jednostruki znak jednakosti (=) predstavlja operator dodele vrednosti, a kao relacioni operator koristi se dvostruka jednakost (==)
- Rezultat poređenja može biti 0 (netačno, *false*) ili 1 (*true*)

Primer: Prikaz vrednosti istinitosti

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

int main() {
  int istinaVrednost, lazVrednost, x = 5, y = 10;
  istinaVrednost = x < y;      // 5 < 10
  lazVrednost = x == y; // 5 = 10
  cout << "Istina je:" << istinaVrednost << endl;
  cout << "Laz je:" << lazVrednost << endl;
  return 0;
}
```

Program daje rezultat:

```
Istina je 1
Laz je 0
```

Logički operatori

- Logički operatori omogućavaju povezivanje više relacionih izraza u složeniji logički izraz
- Operatori mogu biti (po prioritetu izvršavanja):

!	NOT	*logička negacija (unarni)*
&&	AND	*logičko i (binarni)*
||	OR	*logičko ili (binarni)*

  - logička negacija menja vrednost operanda *true* u *false* i obrnuto
  - logičko i daje rezultat *true* samo ako su oba operanda *true*, inače je rezultat *false*
  - logičko ili daje rezultat *true* ako je bar jedan od operanada *true*, a rezultat je *false* je samo ako su oba oparanda *false*

Definisanje sinonima za tipove podataka (typedef)

- Jezik C++ omogućava definisanje sopstvenih tipova podataka pomoću deklaracije *typedef*, npr.

```
typedef long int VelikiCeoBroj
```

- Na taj način se tip long int može definisati u naredbama kao

```
VelikiCeoBroj brojacSlogova = 0L;
```

što je ekvivalentno i može se koristiti istovremeno sa:

```
long int brojacSlogova = 0L;
```

- Upotreba sinonima
  1. olakšava korišćenje složenih deklaracija i
  2. omogućava *lakše prenošenje* programa na druge platforme, jer se sve deklaracija koje zavise od platforme mogu grupisati i promeniti na jednom mestu

Operator sizeof

- Zapis različitih tipova podataka zavisi od arhitekture računara i samog prevodioca
- Podaci o rasponu vrednosti mogu se naći u zaglavljima, za celobrojne vrednosti u `climits` (npr. `INT_MIN` i `INT_MAX`), a za decimalne vrednosti u zaglavlju `cfloat`
- Veličina memorije (u bajtovima) pojedinačnih tipova podataka može se dobiti pomoću funkcije `sizeof(tip)`

Primer: Upotreba funkcije sizeof

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

int main() {
  cout << "Najmanji int:" << INT_MIN << endl;
  cout << "Najveći int:" << INT_MAX << endl;
  cout << "Najmanji double:" << DBL_MIN << endl;
  cout << "Najveći float:" << FLT_MAX << endl;
  cout << "Tip double zauzima:" << sizeof(double) << " bajtova"<< endl;
  cout << "Tip long zauzima:" << sizeof(long) << " bajta" << endl;
  return 0;
}
```

Program daje rezultat:

```
Najmanji int: -2147483648
Najveći int: 2147483648
Najmanji double: 2.22507e-308
Tip double zauzima 8 bajtova
Tip long zauzima 4 bajta
```

Operator nabrajanja (,)

- Operator nabrajanja (razdvajanja) omogućava istovremeno računanje niza izraza u jednoj naredbi
- Rezultat izračunavanje je vrednost *poslednjeg* izraza u nizu, računajući s leva u desno
  - Napomena: prioritet operatora nabrajanja je nizak, tako da treba pažljivo razmotriti redosled izračunavanja u složenim izrazima

Primer: Izračunavanje niza izraza

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

int main() {
  long num1=0, num2=0, num3=0, num4=0;
  num4 = (num1=10, num2=20, num3=30); // niz izraza
  cout << "Vrednost niza izraza je vrednost:" <<
  " poslednjeg u nizu: " << num4 << << endl;
  return 0;
}
```

Izvršavanje programa daje rezultat:

```
Vrednost niza izraza je vrednost poslednjeg u nizu: 30
```

Osnovne ulazno-izlazne operacije

- Ulaz i izlaz podataka sa standardnih ulaznih i izlaznih uređaja (tastatura i ekran) vrši se operatorima umetanja (*insertion*) i izdvajanja (*extraction*)
  - Operator umetanja (<<) umeće znakove u izlazni tok (*output stream*) i vrši automatsku konverziju osnovnih tipova podataka u niz znakova
  - Operator izdvajanja (>>) izdvaja podatke iz ulaznog toka (od početka linije do Enter) i vrši neophodne konverzije, npr.

```
int i;
cout << "Unesite ceobrojnu vrednost: ";
cin >> i;
cout << "Uneli ste vrednost " << i;
```

- Preciznije formatiranje podataka vrši se posebnim manipulatorskim funkcijama (*Tema 11*)

3.2 Evaluacija izraza i prioritet operatora

- Evaluacija izraza
- Prioritet oparatora

Evaluacija izraza

- Redosled evaluacije/računanja vrednosti izraza određen je vrstom i prioritetom operatora
- Prefiksni, unarni i operatori dodele vrednosti izvršavaju se s desna u levo, a svi ostali operatori s leva u desno, npr.
  - izraz `x-y-z` računa se s leva kao `(x-y)-z`
  - u izrazu `x=y=z` dodela vrednosti `=` se računa s desna u levo
- Zagrade menjaju osnovni redosled izračunavanja vrednosti izraza

Prioritet operatora (1)

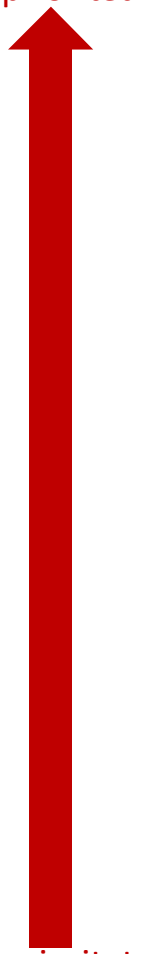
viši prioritet

niži prioritet

Operator	Opis
::	Razrešenje dosega (scope) najviši prioritet
.	Pristup podatku članu
->	Dereferenciranje i pristup podatku članu
[]	Indeksiranje niza
()	Poziv funkcije
++	Inkrementiranje
--	Dekrementiranje
typeid	Identifikacija tipa tokom izvršavanja
const_cast	Ukidanje nepromenljivosti tipa
dynamic_cast	Konverzija tipa tokom izvršavanja
static_cast	Konverzija tipa tokom prevođenja
reinterpret_cast	Neproverena konverzija tipa
!	Logička negacija
~	Bitska negacija
+ (unarni) - (unarni)	Promena znaka
* (unarni)	Dereferenciranje pokazivača
& (unarni)	Adresa promenljive
new	Dinamička alokacija memorije
delete	Oslobađanje dinamičke memorije
sizeof	Veličina promenljive ili tipa

Prioritet operatora (2)

viši prioritet



niži prioritet

Operator	Opis
.* (unarni) ->*	Pristup podatku članu Dereferenciranje i pristup podatku članu
* / %	Množenje Deljenje Modulo (ostatak deljenja)
+ -	Sabiranje i oduzimanje
<< >>	Pomeranje za jedan bit ulevo ili udesno
< <= > >=	Relacioni operatori
== !=	Operatori jednakosti
&	Bitsko i (AND)
^	Bitsko isključivo ili (XOR)
|	Bitsko ili (OR)
&&	Logičko i
||	Logičko ili
:?	Uslovni izraz
= *= /= %= += -= &= ^= >= <<= >>=	Operatori dodele
,	Operator nabrajanja

najniži prioritet

3.3 Konverzija tipova (cast)

- Usaglašavanje tipova objekata s leve i desne strane vrši se prema pravilima konverzije tipova
- Vrednost izraza s desne strane konvertovaće se u vrednost promenljive s leve strane
- Prilikom evaluacije, vrši se konverzija tipova (*type casting*) samih vrednosti u mešovitom izrazu u zajednički kompatibilni tip, npr. tip int u double, a tip char u int
- Konverzija u ciljni tip podatka može se izvršiti i eksplicitno, pomoću naredbe: static_cast <*tip*> (*izraz*), npr.

```
float pi = 3.1415926;
cout << static_cast <int> (pi) << endl;
```

3.4 Imenovane konstante

- U programu se određene nepromenjive vrednosti mogu zadati kao simboličke konstante na dva načina:
  - pomoću naredbe const
  - pomoću direktive #define
- Primer:

```
const double pi = 3.141592653; // u tabeli simbola
#define PI 3.141592653
pi = 2*pi // greška (konstanta s leve strane)!
PI = 2*PI // greška (konstanta s leve strane)!
```

4. Osnovne upravljačke naredbe u jeziku C++

1. Selekcija
2. Iteracija
3. Ostale upravljačke naredbe

4.1 Selekcija

- Osnovne naredbe za selekciju koda koji se izvršava pod određenim uslovima u jeziku C++ su naredbe if i switch.
- Izabrani kod može biti jedna naredba ili blok naredbi u velikim zagradama, npr.

```
int main() {
  double povrsina;
  cin >> povrsina;
  if (povrsina >= 0) {
     double stranica = sqrt(povrsina);
    ...
  }
  ...
  return 0;
}
```

Promenljive definisane u okviru nekog bloka naredbi vidljive su samo u okviru tog bloka

Naredba *if*

- Osnovni oblik naredbe *if* za selekciju koda koji se izvršava pod određenim uslovima je:

```
if (izraz)
   naredbe;
else
   naredbe;
```

- Izraz vraća vrednost na osnovu koje se bira naredba koja će se izvršiti. Rezultat ne mora biti logički, jer se 0 automatski konvertuje u *false*, a bilo koji drugi rezultat u *true*
- Deo else nije obavezan, a izabrane naredbe mogu biti pojedinačna naredba ili blok naredbi

Primer 1: Grananje prema logičkoj ili celobrojnoj vrednosti izraza

```
// Primer grananja
#include <iostream>
using namespace std;
int main() {
  int a;
  cout << "Unesite celi broj: ";
  cin >> a;
  if (a < 0)
     cout "Uneti broj je negativan" << endl;
  if (a % 2)
     cout << "Uneti broj je neparan";
    else
     cout << "Uneti broj je paran";
  cout << endl;
  return 0
}
```

true/false

celi broj

Izvršavanje programa:

```
Unesite celi broj: -5
Uneti broj je negativan
Uneti broj je neparan
```

Primer 2: Grananje prema celobrojnoj vrednosti izraza

```
// Primer grananja
#include <iostream>
using namespace std;
int main() {
  int a, b;
  cout << "Unesite deljenik: ";
  cin >> a;
  cout << "Unesite delilac: ";
  cin >> b;
  if (b)
     cout "Rezultat je " << a/b;
    else
     cout "Deljenje nulom nije dozvoljeno.";
  cout << endl;
  return 0;
}
```

celi broj

Izvršavanje programa:

```
Unesite deljenik: 50
Unesite delilac: 10
Rezultat je: 10
```

```
Unesite deljenik: 50
Unesite delilac: 0
Deljenje nulom nije dozvoljeno.
```

Primer 3: Višestruka selekcija (*if-else-if*)

```
// Primer računanje diskriminante kvadratne jednačine
# include <iostream>
using namespace std;
int main() {
  float a, b, c;
  cout << "Unesite koeficijente kvadratne jednacine: " << endl;
  cin >> a >> b >> c;
  float diskriminanta = b*b - 4*a*c;
  cout << "Kvadratna jednacina ima ";
  if (diskriminanta > 0)
     cout << "dva realna korena.";
   else if (diskriminanta < 0)
     cout << "dva kompleksna korena.";
   else
     cout << "dvostruki realni koren.";
  cout << endl;
  return 0;
}
```

Izvršavanje programa:

```
Unesite koeficijente kvadratne jednačine:
-1 2 -1
Kvadratna jednačina ima dvostruki realni koren.
```

Naredba switch

- Osnovni oblik narebe switch za višestruko grananje prema vrednosti nekog izraza je:

```
switch (izraz) {
  case (konstanta1):
     niz naredbi;
     break;
  case (konstanta2):
     niz naredbi;
     break;
  ...
  default:
     niz naredbi;
}
```

nije obavezna, ali je efikasnije prekinuti dalje ispitivanje

Operator selekcije

- Posebna vrsta selekcije je upotreba *operatora* selekcije

```
uslov ? vrednost1 : vrednost2
```

- Korišćenjem ovog operatora naredba

```
if (x >= 0) y = x;
else y = -x;
```

može se kraće napisati kao:

```
y = x >= 0 ? x : -x;
```

4.2 Iteracija

- U proceduralnim programskim jezicima ponavljanje se može realizovati u odnosu na uslov okončanja ponavljanja (petlje)
  - unapred određen broj puta
  - prema logičkom uslovu, koji se proverava pre početka ponavljanja ili nakon svakog ponavljanja
- Ponavljenje delova koda u jeziku C++ unapred poznati broj puta vrši se pomoću naredbe for, a ponavljanje prema uslovu pomoću naredbi while i do-while

Naredba *for*

- Sintaksa naredbe *for* je

```
for (incijalizacija; izraz; inkrement) blok naredbi;
```

- Inicijalizacija je naredba kojom se postavlja početna vrednost promenljive koja određuje broj izvršavanja petlje
  - ova promenljiva je često potrebna samo kao brojač, pa se deklariše *unutar* definicije for petlje, tako da izvan petlje nije definisana
- Izraz je uslov koji na svakom koraku određuje da li će ponavljanje biti izvršeno
  - mora biti *true*, inače se izvršavanje programa nastavlja od prve sledeće naredbe *iza* petlje *for*
- Inkrement je definicija operacije promene promenljive koja upravlja ponavljanjem

Naredba *while*

- Naredba *while* ima sintaksu

```
while (izraz) blok naredbi;
```

- Npr. za Euklidov algoritam se ne zna unapred broj ponavljanja

```
#include <iostream>
using namespace std;
int main() {
  int x, y;
  cout << "Unesite brojeve ciji NZD trazite: " << endl;
  cin >> x >> y;
  while (x != y)
    if (x > y) x = x-y;
    else y = y-x;
  cout << "NZD=" << x << endl;
  return 0;
}
```

Naredba *do-while*

- Naredba *do-while* ima sintaksu

```
do
   blok naredbi
while (izraz);
```

- Primer: naknadno odlučivanje o nastavku programa:

```
#include <iostream>
using namespace std;
int main() {
  const int TAJNI_BROJ = 15;
  int broj = 0;
  do {
    cout << "Unesite tajni broj: " << endl;
    cin >> broj;
  } while (broj != TAJNI_BROJ);
  cout << "Pogodili ste tajni broj!"<< endl;
  return 0;
}
```

4.3 Ostale upravljačke naredbe

- Naredba bezuslovnog skoka (*goto*)
- Naredbe *break* i *continue*

Naredba bezuslovnog skoka (*goto*)

- Naredba bezuslovnog skoka na lokaciju u programskom kodu koja je navedena iza reči *goto*, npr.

```
if (imenilac == 0) goto deljenjeNulom;
// naredbe koje se preskaču ...
deljenjeNulom:
  cout << "Deljenje nulom nije dozvoljeno"
```

- Naredba odgovara mašinskoj instrukciji bezuslovnog skoka
- U savremenom programiranju praktično se *ne koristi*, jer čini programski kod manje čitljivim i teškim za održavanje
- U savremenim programskim jezicima uvek postoji drugi, elegantniji način realizacije istog algoritma *bez* naredbe *goto*

Naredba *break*

- Naredba break se koristi za prekid izvršavanja petlji, nakon čega se program nastavlja na prvoj sledećoj naredbi *iza petlje*
- Npr. sledeća petlja se prekida kada kontrolna promenljiva x dostigne vrednost 3:

```
for (int x=10; x > 0; x--)
    if (x == 3) {
        cout << "Odbrojavanje je završeno.";
        break;
    }
```

- Kod ugnježdenih petlji, *break* prekida samo unutrašnju petlju
- Naredba se koristi za prekid petlje u posebnim situacijama, kao što je npr. prekid beskonačne petlje

Naredba *continue*

- Naredba continue se koristi za prekid izvršavanja samo *jedne iteracije* petlje
- Program se nastavlja od provere uslova petlje za izvršenje sledeće iteracije i eventualno prelazi na *sledeću iteraciju*

```
for (int x=0; x <= 100; x++) {
    if (x % 2) // preskače neparne brojeve
       continue;
    else
       cout << x << endl;
}
```

5. Strukture podataka u jeziku C++

1. Polja (nizovi)
2. Stringovi
3. Polja stringova
4. Pokazivači
5. Korisnički definisane strukture
6. Povezane liste

Strukture podataka u jeziku C++

- Polja i stringovi predstavljaju skup promenljivih u memoriji kojima se pristupa pomoću zajedničkog imena
  - mogu imati jednu ili više dimenzija
- Tip string ne predstavlja osnovni tip podataka u jeziku C++, već jednodimenzionalno polje (niz) znakova
- Pokazivači su promenljive čiji sadržaj predstavlja neku memorijsku adresu, koja može da služi za pristup drugim objektima u memoriji

5.1 Polja (nizovi)

- Skup povezanih promenljivih istog tipa deklariše se naredbom

  tip ime_polja [$dimenzija_1$] [$dimenzija_2$] .. [$dimenzija_N$];

- Jednodimenzionalno polje ili vektor je npr.

```
int primer [10];
```

- Višedimenzionalno polje je npr.

```
int primer2D[10][20];
```

- Pristup elementima polja vrši se pomoću *indeksa*, npr.

```
primer[10]
primer2D[3][5]
```

- U jeziku C++ indeksi počinju od 0, tako da polje od 10 elemenata ima indekse od 0 do 9, npr. `primer[0]`.. `primer[9]`

Primer: Deklarisanje i inicijalizacija polja

```
const int BROJ_REDOVA = 3;
const int BROJ_KOLONA = 4;
// Deklaracija polja celih brojeva od 3 reda i 4 kolone
int brojevi[BROJ_REDOVA][BROJ_KOLONA];
// Prvi indeks predstavlja red, a drugi kolonu
for (int red=0; red < BROJ_REDOVA; red++)
    for (int kolona=0; kolona < BROJ_KOLONA ; kolona++){
        brojevi[red][kolona]= 4*red + kolona + 1
    }
}
```

Šta predstavlja ovaj izraz?

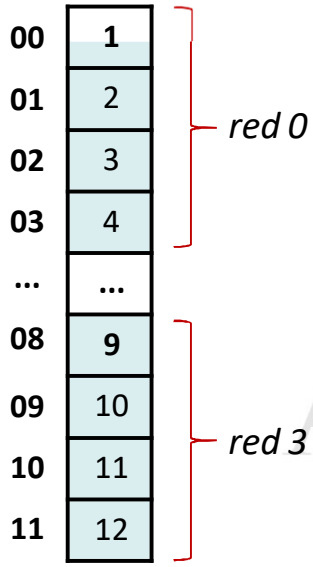
- Vrednost promenljive `brojevi[0][0]` je 1, `brojevi[0][1]` je 2 , `brojevi[0][2]` je 3, a npr. `brojevi[2][3]` je 12

Predstavljanje polja u memoriji

- Polja se u memoriji predstavljaju obrnuto od redosleda indeksa, tako da se poslednji indeks menja najbrže
- Npr. dvodimenzionalno polje *(a)* u memoriji se predstavlja tako da su *redovi* u sukcesivnim memorijskim lokacijama *(b)*

		kolona 0	1	2	3
	0	1	2	3	4
red	1	5	6	7	8
	2	9	10	11	12

(a) dvodimenzionalno polje



(b) raspored elemenata 2D polja u memoriji

Važno za upotrebu programskih *biblioteka*:

A. predstavljanje polja prvo po *redovima*: *C*, *C++*, *Objective-C (samo za C-polja)*, *Pascal*, *C#*

B. predstavljanje polja prvo po *kolonama* : *Fortran*, *OpenGL* , *MATLAB/GNU Octave*, *Objective-C*, *R* (jezici *Java*, *Javascript*, *Python* i *PHP* koriste pokazivače)

Predstavljanje polja u memoriji

- Memorijski prostor za predstavljanje polja određuje se u toku prevođenja kao

  $dimenzija_1$ x $dimenzija_2$ x .. $dimenzija_N$ x sizeof(*tip elementa*)

- Zauzeće memorije raste progresivno s brojem i veličinom dimenzija, npr.
  - četvorodimezionalno polje znakova 100 x 60 x 90 x 40 zauzima 21.600.000 bajtova (oko 20 MB)
  - rasterska slika u RGB koloru rezolucije 800x600 piksela zauzima 480.000 x 24 bita ili oko 1,4 MB, a u rezoluciji 1920x1080 oko 6 MB

5.2 Stringovi

- Stringovi su jednodimenzionalna polja (nizovi) znakova
- U jeziku C++ koriste se *dva načina* predstavljanja nizova znakova:
  1. C string, niz znakova koji se terminira nul-znakom \0 (*null terminated string*), nasleđen iz jezika C, koji omogućava efikasno predstavljanje i detaljnu kontrolu programera nad operacijama sa stringovima
  2. Klasa string iz biblioteke klasa jezika C++
- String ili niz znakova koji završava nulom deklariše se tako da dimenzija niza bude *za jedan veća* od najveće dužine stringa koji će se koristiti
  - npr. za string str dužine 10 znakova ispravna deklaracija je

```
char str[11]
```

Učitavanje stringa s tastaure

- Niz znakova se može direktno učitati naredbom cin, samo se mora voditi računa da naredba string čita do prve "beline" (*whitespace*: prazno mesto, tabulator ili znak za novi red)
- Bibliotečna funkcija gets() omogućava čitanje svih znakova stringa sve do znaka za novi red (*carriage return*, \r), npr.

```
#include <iostream>
using namespace std;
int main() {
  char str[81];
  cout << "Unesite string :";
  gets_s(str, 80); // nova verzija ograničava broj znakova
  cout << "Uneli ste: " << str << endl;
  return 0;
}
```

Funkcije za rad sa stringovima

- Najčešće korišćene funkcije za rad sa stringovima u biblioteci funkcija jezika C++ su:

Funkcija	Opis

`strcpy(s1, s2)`
`strcpy_s()`
`wcscpy()`
	Kopira string s2 u string s1. Niz s1 mora da bude dovoljno dugačak, jer će se u suprotnom biti prekoračena granica niza.

`strcat(s1, s2)`
`strcat_s()`
`wcscat()`
	Dodaje s2 na kraj s1; s2 ostaje neizmenjen. s1 mora da bude dovoljno dugačak da u njega stane sadrži prvobitni sadržaj i ceo niz s2.

`strcmp(s1, s2)` `wcscmp()`	Poredi stringove s1 i s2; ako su jednaki, vraća 0, ako je s1 > s2 (po leksikografskom redu) vraća 1, u suprotnom vraća negativan broj.
`strlen(s)`	Vraća dužinu stringa s.

Vraća 0 == *false* kad su stringovi jednaki

Funkcije za rad sa pojedinačnim znakovima

- U zaglavlju <cctype> standardne biblioteke jezika C++ definisane su funkcije za rad sa znakovima, npr.

Funkcija	Opis
`toupper(c)`	prevodi znak u veliko slovo (uppercase)
`tolower(c)`	prevodi znak u malo slovo (lowercase)
`isupper(c)`	true ako je znak veliko slovo
`islower(c)`	false ako je znak veliko slovo
`isalnum(c)`	true ako je znak alfanumerički
`isdigit(c)`	true ako je znak cifra
`isspace(c)`	true ako je znak razmak

Primer: Konverzija stringa u velika slova

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

int main() {
    char str[80];
    strcpy(str, "Ovo je test!");
    // Konverzija znakova stringa u velika slova
    for (int i=0; str[i], i++)
        str[i] = toupper(str[i]);
    cout << str << endl;
    return 0;
}
```

Izvršavanje programa:

```
OVO JE TEST!
```

Ekvivalentno str[i] == 0 (kraj stringa) odnosno *false*

Inicijalizacija stringova

- Inicijalizacija polja vrši se naredbom oblika

```
tip ime_polja [dimenzija] = {skup_vrednosti}
```

- Skup vrednosti je niz vrednosti odgovarajućeg tipa odvojenih zarezima, npr. {10, 20, 30, 40}
- Kada se string istovremeno deklariše i inicijalizuje, dimenzije se mogu izostaviti, jer ih prevodilac može odrediti automatski
- Dimenzije stringova se mogu izostaviti, npr. umesto

```
char str[4] = "C++";
```

može se napisati (sistem dodaje oznaku \0 na kraju stringa)

```
char str[] = "C++";
```

5.3 Polja stringova

- Polje stringova je poseban oblik dvodimezionalnog niza, koji se često koristi za različite operacije pretraživanja
- U deklaraciji polja, prvi indeks je broj nizova, a drugi maksimalna dužina niza (uključujući oznaku kraja), npr.

```
char daniUnedelji[7][11] = {"Ponedeljak", "Utorak",
  "Sreda", "Cetvrtak", "Petak", "Subota", "Nedelja"};
```

- Stringovima se pristupa pomoću jednog, a elementima stringa pomoću dva indeksa, npr.

```
daniUnedelji[3]      // Cetvrtak
daniUnedelji[3][2]   // Prvi znak "t" u stringu Cetvrtak
```

Primer: Telefonski imenik

```
#include  <iostream>
using namespace std;
int main() {
    int i;   // brojač, dostupan izvan petlje
    char str[80];
    // Definisanje niza od 10 stringova dužine do 79 znakova
    char brojevi[10][80] = {
        "Pera", "065-234-1123",
        "Deki", "063-1123400",
        "Sandra", "062-3256343",
        "Laza", "061-3453453"
    }
    cout << "Unesite ime: ";
    cin >> str
    for (i=0; i < 10; i+=2) {
        if (!strcmp(brojevi[i], str) {
           cout << "Broj je: " << brojevi[i+1] << endl;
           break;
        }
    }
    if (i = 10)
       cout << "Broj nije pronadjen." << endl;
    return 0;
}
```

Izvršavanje programa:

```
Unesite ime: Deki
Broj je: 063-1123400
```

```
Unesite ime: Filip
Broj nije pronadjen.
```

Funkcija *strcmp* poredi stringove i vraća 0 odnosno *false* kad su stringovi isti

5.4 Pokazivači

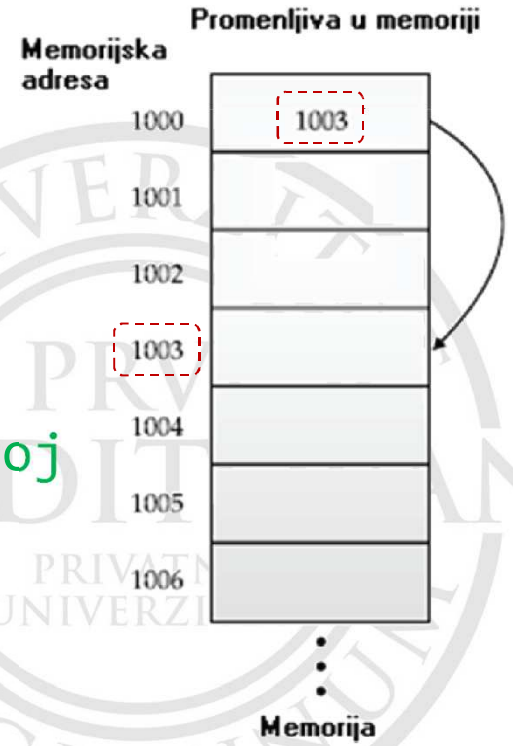
- Pokazivač (*pointer*) je promenljiva koja sadrži memorijsku adresu nekog objekta ("pokazuje" na objekt)
- Adresa nije običan celi broj, pa se deklariše na poseban način:

```
tip *promenljiva
```

- Tip predstavlja *tip podatka* na koje pokazivač pokazuje, npr. pokazivač `intp` na promenljivu tipa *int* deklariše se kao

```
int *intp // intp pokazivač na celi broj
```

(zvezdica ispred imena promenljive označava pokazivač)

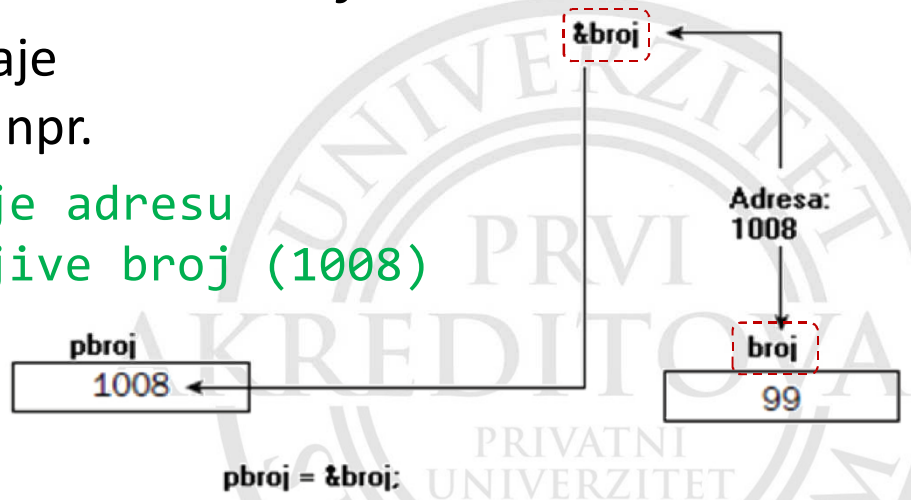


Operacije nad pokazivačima

- Da se izbegnu opasne reference usled neinicijalizovanih pokazivača, usvojeno je da se inicijalizuju na vrednost NULL (nula), sa značenjem da pokazivač ne pokazuje ni u šta
- Operatori koji omogućavaju operacije nad pokazivačima su unarni operatori *adresiranja* & i *indirekcije* *
  - operator adresiranja & (slika) daje *vrednost adrese* nekog objekta, npr.

```
pbroj = &broj // dodeljuje adresu
              // promenljive broj (1008)
```

  - operator indirekcije * daje *vrednost promenljive* koja se adresira




```
vredn = *pbroj // dodeljuje vrednost (99)
```

Osnovni tip pokazivača i dodela vrednosti promenljivoj

- Osnovni tip pokazivača (tip promenljive na koju pokazuje) služi za određivanje broja bajtova koji će se upotrebiti u dodeli vrednosti promenljivoj pomoću pokazivača, npr.

```
int *p;    // pokazivač na tip int
double f;
p = &f;    // greška, ne moze da pokazuje na tip double
```

  - *Napomena*: dodela i provera NULL vrednosti

```
p = NULL; // pokazivač koji ne pokazije ni u šta
p = 0;    // ista vrednost kao NULL
if (!p)
   cout << "Pokazivac ne pokazuje ni u sta" << endl;
```

Dodela vrednosti promenljivoj i pokazivačka aritmetika

- Pokazivač se može koristiti za dodelu vrednosti promenljivoj čiju adresu poznajemo, npr. inkrementiranje oblika (*p++)

```
int *pbroj = NULL;
int broj = 10;
pbroj = & broj; // pokazivač na promenljivu broj  (==10)
(*pbroj)++;     // inkrementiranje preko pokazivača (==11)
```

- Pokazivači se mogu koristiti u većini izraza u jeziku C++, ali se koriste samo operatori +, -, ++ i --
- *Rezultat ovih operacija zavisi od osnovnog tipa pokazivača, zbog različitog zauzeća memorije*

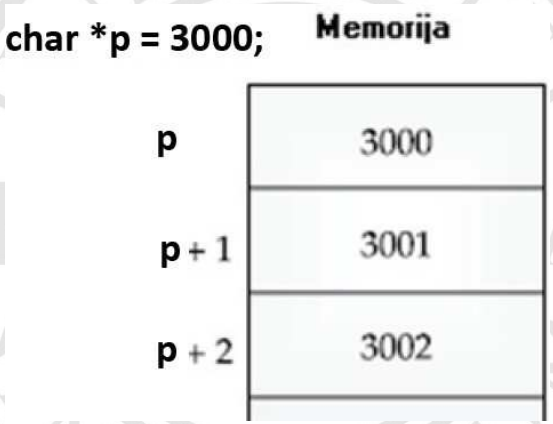
Pokazivači i polja

- Elementima polja se može pristupiti na dva načina: pomoću indeksa i (ponekad efikasnije) pomoću pokazivača, npr.

```
char str[80];
char *p;
p = str;  // dodela adrese prvog znaka str ili str[0]
```

Promenljiva p je pokazivač na znakove, a prva dodeljena vrednost pokazuje na `str[0]`

- *Rad s pokazivačima može biti efikasniji zbog manjeg broja mašinskih instrukcija kad se elementima pristupa po redosledu u memoriji*: *p, p+1, ...*



Primer: Isti program na dva načina

```
// Pristup pomoću indeksa
#include  <iostream>
using namespace std;

int main() {
  char str[] = "Ovo Je Test";
  for (int i=0; str[i]; i++) {
       if (isupper(str[i])
          str[i] = tolower(str[i]);
      else if (islower(str[i])
          str[i] = toupper(str[i]);
  }
  cout << str << endl;
  return 0;
}
```

```
// Pristup pomoću pokazivača
#include  <iostream>
using namespace std;

int main() {
  char str[] = "Ovo Je Test";
  char *p = str; //*p za str[i]
  while (*p) {
    if (isupper(*p))
         *p = tolower(*p);
      else if (islower(*p)
         *p  =  toupper(*p);
    p++; // sledeći element str
  }
  cout << str << endl;
  return 0;
}
```

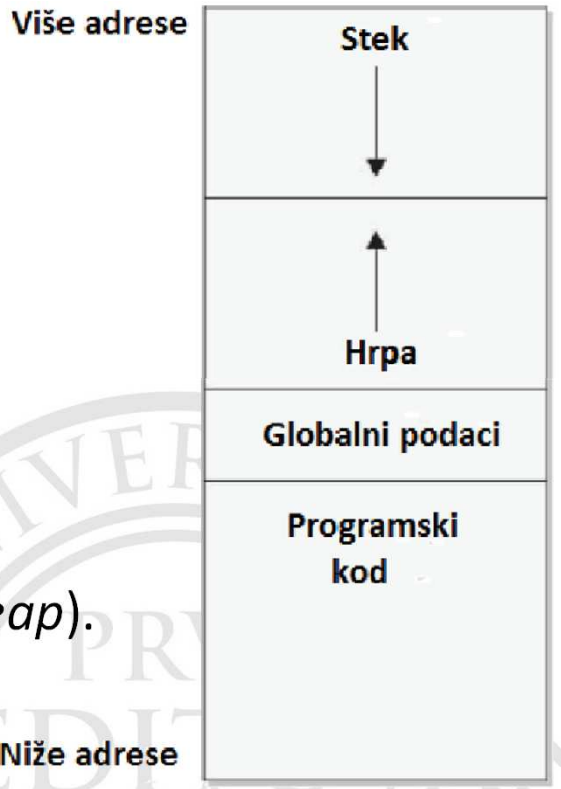
Izvršavanje programa:

```
oVO jE tEST
```

Dinamička alokacija memorije

- Struktura memorije programa
  - Statičke lokalne promenljive i promenljive kreirane izvan svih funkcija su u zoni *globalnih* podataka
  - Lokalne promenljive, koje se kreiraju u bloku naredbi su u zoni koja se koristi kao stek (*stack*) i uništavaju se nakon izlaska iz bloka
  - Dinamički kreirane promenljive (po potrebi), alociraju se u posebnom delu memorije (hrpa, *heap*). Koriste se operatori *new* i *delete*, npr.

```
int *p = new int;
int *q = new int[100];
delete p;
delete [] q;
```



5.5 Korisnički definisane strukture

- Strukture podataka koje sadrže podatke različitih tipova mogu se definisati naredbom struct

```
struct naziv { naziv_polja tip; ... };
```

- Stvarni objekti najčešće imaju svojstva različitih tipova, npr.

```
struct Knjiga {
  char naslov[80];
  char autor[80];
  char izdavac;
  int godina;
}
```

- Svojstva strukture su polja (*fields*, *members*) i mogu biti bilo kog tipa osim istog onog koji se definiše naredbom *struct*

Upotreba korisničkih struktura

- Promenljiva ovakvog tipa deklariše se na uobičajen način

```
Knjiga roman;
```

- Inicijalizacija se vrši dodelom skupa vrednosti odgovarajućih tipova:

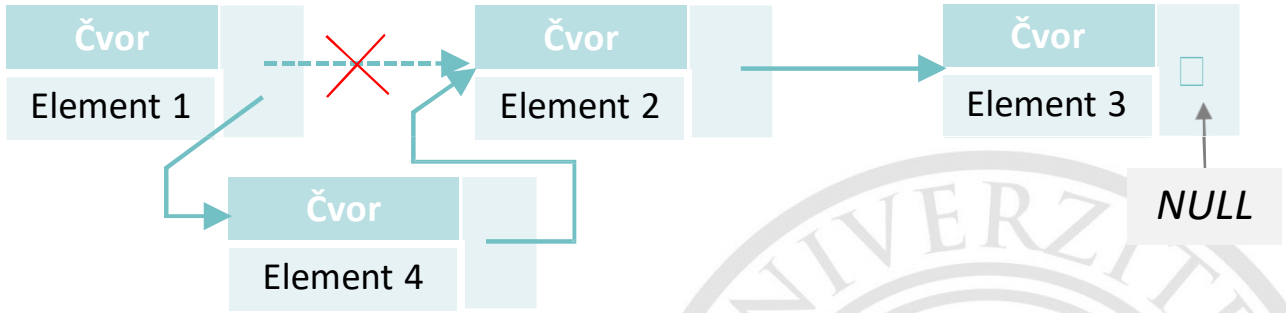
```
Knjiga roman {
  "Na Drini cuprija",
  "Ivo Andric",
  "Zavod za udzbenike",
  2009
}
```

- Upotreba promenljive novog tipa vrši se pomoću *operatora selekcije*, npr.

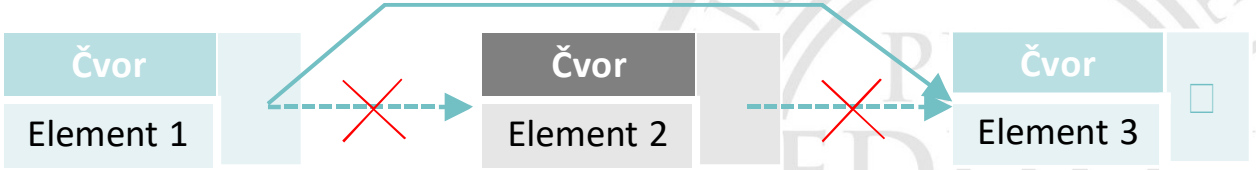
```
roman.godina += 2; // novije izdanje
```

5.6 Povezane liste

- Povezana lista (*linked list*) je struktura koja se sastoji od niza međusobno povezanih elemenata (*nodes*), tako da je dodavanje i uklanjanje elemenata iz liste efikasno
  - *dodavanje*



  - *uklanjanje*



- Lista može biti jednostruko ili dvostruko povezana
  - može da sadrži jedan pokazivač na sledeći element ili dva pokazivača, na sledeći i prethodni element

Definisanje povezane liste

- Lista se može definisati kao korisnička struktura podataka, koja osim podataka sadrži i pokazivače na sledeći i/ili prethodni element liste, jer pokazivač *može* da pokazuje na strukturu *istog* tipa:

```
struct ElementListe {
  // definicija elementa strukture
  ...
  ElementListe *sledeci;   // pokazivač na element
  ElementListe *prethodni; // pokazivač na element
};
```

- Povezana lista se može upotrebiti za implementaciju drugih struktura podataka, npr. stek (*stack*) i red (*queue*)

6. Primeri programa

1. Kreditni kalkulator u jeziku C++
   (bez upotrebe korisničkih funkcija)
2. Sortiranje polja: Selection sort

6.1 Kreditni kalkulator u jeziku C++

- Mesečna rata otplate kredita zavisi od iznosa zajma, mesečne kamate i broja otplatnih rata, koji se uprošćeno može izraziti brojem godina otplate
- Izraz za računanje mesečne rate je [8]

$$rata = \frac{iznos * kamata}{1-(1+kamata)^{-n}} = \frac{iznos * kamata}{1-\frac{1}{(1+kamata)^{n}}}$$

gde su:

rata - iznos mesečne rate

kamata - mesečna kamata (godišnja/12)

iznos - iznos kredita

n - broj otplatnih rata kredita (broj godina*12)

Kreditni kalkulator u jeziku C++ (1/2)

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

int main() {

  // Godisnja kamata
  cout << "Unesi godisnju kamatu (decimalni broj): ";
  double godisnjaKamata;
  cin >> godisnjaKamata;

  // Mesecna kamata
  double mesecnaKamata = godisnjaKamata / 1200;

  // Otplatni period u godinama
  cout << "Unesi broj godina otplate zajma (celi broj): ";
  int brojGodina;
  cin >> brojGodina;
```

Kreditni kalkulator u jeziku C++ (2/2)

```
    // Iznos zajma
    cout << "Unesi iznos zajma (decimalni broj): ";
    double iznosZajma;
    cin >> iznosZajma;

    // Racunanje mesecne rate
    double mesecnaRata = iznosZajma * mesecnaKamata /
                         (1 - 1/pow(1 + mesecnaKamata, brojGodina * 12));
    double ukupnoVraceno = mesecnaRata * brojGodina * 12;

    // Formatiranje izlaza na dve decimale
    mesecnaRata = static_cast<int>(mesecnaRata * 100) / 100.0;
    ukupnoVraceno = static_cast<int>(ukupnoVraceno * 100) / 100.0;

    // Prikaz rezultata
    cout << "Mesecna rata " << mesecnaRata <<
      "\nUkupno se otplati " << ukupnoVraceno << endl;
      return 0;
}
```

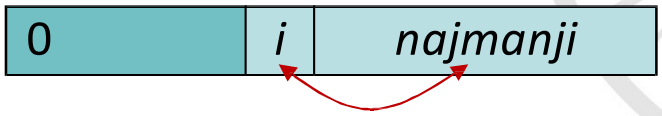
```
Unesi godisnju kamatu (decimalni broj): 4.5
Unesi broj godina otplate zajma (celi broj): 20
Unesi iznos zajma (decimalni broj): 50000
Mesecna rata 316.32
Ukupno se otplati 75917.9
```

6.2 Sortiranje polja: Selection sort

- Ako se sortiranje vrši u rastućem redosledu, metod pronalazi najmanji element u polju i menja ga s prvim elementom
  - sortiranje u opadajućem redosledu traži najveći element
- Postupak pronalaženja najmanjeg elementa i zamene s prvim ponavlja se za ostatak polja, sve dok ne preostane samo jedan element. Opšti oblik metoda je

```
for (i=0, i<broj_elemenata, i++)
    // Izabere se najmanji element polja, od "i+1" do kraja
    // Ako je potrebno, zamene se najmanji i element "i"
```

- Na kraju svake iteracije element i je na konačnoj poziciji. Sledeća iteracija se primenjuje na ostatak polja od i+1 do kraja



Program

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

int main () {

  // Sortiranje polja brojeva metodom zamene

  int polje [10] = {33,100,4,55,66,88,7,800,11,0};
  int brojElemenata = 10;

  for (int i=0; i<brojElemenata-1; i++) {
    int currentMin    = polje[i];
    int currentMinInd = i;
    // Pronadje se najmanji element od polje[i] do kraja
    for (int j=i+1; j<brojElemenata; j++) {    // Selekcija
        if (currentMin > polje[j]) {
           currentMin      = polje[j];
           currentMinInd = j;
        }
    }
    // Zamene se polje[i] i polje[currentMinIndex], po potrebi
    if (currentMinInd != i) {
        polje[currentMinInd] = polje[i];
        polje[i] = currentMin;
    }
  }
  for (int i=0; i<brojElemenata; i++)
      cout << polje [i] << " ";
  cout << endl;

  return 0;
}
```

```
0 4 7 11 33 55 66 88 100 800
```

Literatura

1. Branović I., *Osnove objektno orijentisanog programiranja: C++*, Univerzitet Singidunum, 2013
2. Stroustrup B., *The C++ Programming Language*, 4th Ed, Addison Wesley, 2013
3. Horton I., Van Weert P., *Beginning C++ 20*, 6th Edition, Apress, 2020
4. Horton I., *Beginning C++*, Apress, 2014
5. Horton I., *Beginning Visual C++ 2013*, Wox/John Wiley&Sons, 2014
6. Horton I., *Using the C++ Standard Template Libraries*, Apress, 2015
7. Galowitz J., *C++ 17 STL Cookbook*, Packt, 2017
8. Liang D., *Introduction to Programming with C++*, Pearson Education, 2007
9. Veb izvori
   - http://www.cplusplus.com/doc/tutorial/
   - http://www.learncpp.com/
   - http://www.stroustrup.com/
10. Knjige i priručnici za *Visual Studio* 2010/2012/2013/2015/2017/2019