

Programski jezici i strukture podataka

3

IZRAZI I NAREDBE

leksičke komponente

promenljive
konstante

operandi

+

operatori



izrazi

Svaki izraz ima **tip** i **vrednost**.

Operandi se implicitno konvertuju u odgovarajući tip.

Tip izraza se može eksplicitno promeniti tzv. **cast** operatorom (ispred izraza se u zagradi navede ime tipa u koji želimo da konvertujemo izraz).

(tip)<izraz>

```
int i=3;  
float f=2.4;  
f=i; /* f ce imati vrednost 3.0*/
```

```
int i=3;  
float f=2.4;  
i=f; /* i ce imati vrednost 2*/
```

- Izraz
 - Obavlja računanje i daje rezultat.
 - Uzrokuje neku aktivnost (**bočni efekat**).
- Izuzetak (exception)(**z10.c**)

je stanje programa pri kome određivanje vrednosti nekog izraza daje matematički neodređen rezultat ili rezultat izvan opsega važenja.

- Rezultat aritmetičkih operacija je izraz istog tipa kao i operandi.
- Ako operandi nisu istog tipa tada se vrši implicitna konverzija užeg u širi tip.

Naredba

- Osnovna naredba je izraz iza koga sledi znak ";"
- Imamo više vrsta naredbi:
 - Označena naredba
 - Složena naredba (blok)
 - Naredba izbora
 - Izraz
 - Naredba ponavljanja
 - Naredba skoka

Operatori

- Dodele
- Aritmetički
- Operator sizeof()
- Operatori poređenja
- Logički
- Operatori nad bitima
- Inkrementacija i dekrementacija
- Pomeranje
- Adresa -od i posredni pristup

Operatori

- **Unarni operatori** - prefiksni i sufiksni.
- **Binarni operatori** - infiksni.

Prioritet i asocijativnost operatora

- Unarni operatori imaju viši prioritet od binarnih.
- Aritmetički operatori su višeg prioriteta od relacionih, a ovi višeg od logičkih.
- Operatori dodele su najnižeg prioriteta.
- Ako dva operatora imaju isti prioritet, onda se u obzir uzima asocijativnost, koja može biti s leva na desno, ili s desna na levo.
- Prioritet operatora se može promeniti korišćenjem zagrada ().

Deljenje - konverzije

Kako izbeći celobrojno deljenje

```
int a,b;
```

```
float c;
```

```
a = 5;
```

```
b = 2;
```

```
c = a/b; /* Celobrojno deljenje, c=2*/
```

```
c = (1.0*a)/b; /*Implicitna konverzija: 1.0*a je realan
```

```
broj pa prilikom deljenja sa b dobija se realan rezultat c=2.5*/
```

```
c = (0.0+a)/b; /* Implicitna konverzija: (0.0+a) je realan broj pa prilikom  
deljenja sa b dobija se
```

```
realan rezultat c=2.5*/
```

```
c = (float)a/(float)b; /* Eksplicitna konverzija*/
```

Operatori sa bočnim efektima

- Pojava da se prilikom izračunavanja nekog izraza menja vrednost neke promenljive naziva se **bočni efekat** (eng. side effect).
- Operatori koji imaju bočni efekat su operatori dodele i operatori uvećanja i umanjenja.
- Naredba dodele se u C-u predstavlja izrazom dodele, a njeno izvršavanje se zasniva na bočnim efektima operatora dodele.

Operatori dodele =

- Levi operand ovog operatora je leva vrednost (ime promenljive). Desni operand je proizvoljni izraz.
- Najpre se izračunava izraz na desnoj strani. Njegova vrednost se zatim po potrebi konvertuje u tip promenljive na levoj strani. Nakon toga se dobijena vrednost upisuje u memorijsku lokaciju koja je rezervisana za čuvanje te promenljive.

Operatori umanjenja i uvećanja

- Unarni operatori **++** i **--** nazivaju se operatori **uvećanja** (inkrementacije) i **umanjenja** (dekrementacije) respektivno.
- Mogu biti prefiksni i sufiksni.
- U oba slučaja vrši se uvećanje promenljive za 1 ali izraz **++n** uvećava promenljivu *n* *pre* nego što se njena vrednost koristi, dok **n++** uvećava *n* *nakon* što se njena vrednost koristi. Tako se `x=++n;` razlikuje od `x=n++;`.
- Slično za operator **--**, s tim što je u pitanju umanjenje za jedan.

(z11.c)

Prosta naredba

- **Prosta naredba** se sastoji iz izraza za kojim sledi karakter ';'.
- Naredba se izvršava tako što se izračunava izraz iz koga se sastoji uz sve bočne efekte koji se pri tom javljaju.
- Jasno je da prosta naredba ima smisla samo ako sadrži bar jedan bočni efekat.

Kontrola toka - if

```
if (izraz)  
    naredba1  
else  
    naredba2
```

- Naredba može biti prosta naredba a može biti i složena naredba (blok) koja se dobije kada se više prostih naredbi grupišu navođenjem vitičastih zagrada.

Else-if

- `if (izraz1)`
 `iskaz1`
`else if (izraz2)`
 `iskaz2`
`else if (izraz3)`
 `iskaz3`
`else if (izraz4)`
 `iskaz4`
`else iskaz`

Ciklusi

- `while`
- `do while`
- `for`

for Ciklus

- `for (izraz1; izraz2; izraz3)`
`naredba`

- Ovo je ekvivalentno kodu:

```
izraz1;  
while (izraz2)  
{  
    naredba  
    izraz3;  
}
```

Primer – for ciklus

- `#include <stdio.h>`

```
int main()
```

```
{
```

```
    int x;
```

```
    for (x = 1; x < 10; x++)
```

```
        printf("x = %d\n", x);
```

```
}
```

- Napomena: izraz1, izraz2, izraz3 i naredba mogu biti izostavljeni. Ako je izraz2 izostavljen podrazumeva se da je stalno tačan.
- `for (; ;) ;`
- predstavlja "beskonačnu" for petlju.

Naredbe skoka `break` i `continue`

- Naredba **`break`** (z12.c)
omogućava prevremeni izlazak iz ciklusa
- Naredba **`continue`** (z13.c)
omogućava izlazak iz tekuće iteracije u ciklusu i nastavak izvršenja ciklusa počev od sledeće iteracije.

Višesmerno grananje switch (z14.c)

```
switch (izraz)
{
    case oznaka1:
        naredba1
    case oznaka2:
        naredba2
    ...
    default
        naredbaN
}
```

Ternarni operator

- condition ? expression1 : expression2
- Funkcioniše kao if-else

```
min = ( x < y ) ? x : y;
```

- Programski jezik C omogućava uslovni(ternarni) operator ?: koji odgovara IF-ELSE kontrolnoj strukturi.
- Uslovni (ternarni) operator je operator koji ima tri operanda koji zajedno sa operatorom čine uslovni izraz.
- Prvi operand je **uslovni izraz**, drugi operand je ona vrednost koju će poprimiti ceo izraz ukoliko je vrednost prvog operanda tačna, a treći operand je ona vrednost koju će poprimiti ceo izraz ukoliko je vrednost prvog operanda netačna. **(z15.c)**

NIZOVI

Niz je objektni tip

- Serija objekata istog tipa, koji se nalaze na susednim memorijskim lokacijama.
- Tip elementa niza može biti bilo koji objektni tip (tip je objektni ako njegov opis sadrži skladištnu veličinu).
- Ceo niz takođe predstavlja objekat.
- Tip niza je određen tipom elementa.
- Ako elementi niza imaju tip T, onda je imamo niz elemenata tipa T.

Definicija niza

- Definicija niza određuje njegov identifikator, tip i broj elemenata.

```
tip ime [broj_elementa];
```

- Broj elemenata mora biti celobrojni izraz pozitivne vrednosti.

```
char buffer [4*512];
```

- Niz buffer koji sadrži 2048 elemenata tipa char.

Nizovi konstantne dužine

- Niz konstantne dužine može biti definisan u bloku ili van svake funkcije, bilo kog veka trajanja.

```
int a[10]; // Niz a ima spoljnje povezivanje.
static int b[10]; // Niz b je lokalni objekat
                  //oblast važenja mu je datoteka.

void func( )
{
static int c[10]; // Niz c je lokalni objekat
                  //oblast važenja mu je blok.
int d[10]; // Niz d je automatski objekat -
           //extern.

/* ... */
}
```

Nizovi promenljive dužine

- Niz se može definisati pomoću promenljivog izraza za broj elemenata, ako je po životnom veku automatski objekat (bez specifikatora `static`)

```
void func( int n )
{
    int vla[2*n]; // OK: automatski objekat.
    static int e[n]; // Pogrešno: niz promenljive
                    //dužine ne može biti static objekat.
    struct S { int f[n]; }; // Pogrešno : f nije
//običan identifikator, članovi unije ili
strukture ne mogu biti nizovi prom. duž.
/* ... */
}
```

Pristupanje elementima niza

- Operator indeksiranja [], omogućava pristup elementima niza pomoću indeksa:

```
long myArray[4];
```

```
myArray[3] = 8;
```

- Elementima niza se može pristupati aritmetikom pokazivača:

```
for ( long *p = myArray; p < myArray + A_SIZE; ++p )  
    *p *= 2;
```

Inicijalizacija nizova

- Ako niz ima automatsku određen životni vek, vrednosti elemenata su nedefinisane.
- U suprotnom, elementima niza se dodeljuje vrednost 0, kod nizova pokazivača NULL.
- **(z100.c)**

Inicijalizacija nizova

```
int a[4] = { 1, 2, 4, 8 }; //inicijalizaciona lista  
a[0] = 1, a[1] = 2, a[2] = 4, a[3] = 8  
int a[] = { 1, 2, 4, 8 }; // niz od 4 elementa
```

- Sledeća 4 izraza su ekvivalentna:

```
int a[4] = { 1, 2 };  
int a[] = { 1, 2, 0, 0 };  
int a[] = { 1, 2, 0, 0, };  
int a[4] = { 1, 2, 0, 0, 5 };
```

Inicijalizacija nizova

- Nizove promenljive dužine ne možemo inicijalizovati u definiciji.
- Ako je niz statičkog životnog veka, inicijalizatori moraju biti konstantni izrazi.
- Ako se napiše inicijalizaciona lista, nemora se navesti dužina niza u definiciji.
- Ukoliko u definiciji imamo i dužinu niza i inicijalizacionu listu, važeća je dimenzija zadata u uglastim zagradama. (**z101.c**)

Pristup elementima niza

- Elementi niza se identifikuju pomoću rednog broja unutar niza koji se naziva indeks.
- Indeksi u jeziku C su celi brojevi od 0 do $n-1$, gde je n broj elemenata (tj. dužina) niza.
- Pristup elementima niza naziva se indeksiranjem.
- Indeksiranje u jeziku C smatra se binarnim operatorom i obeležava se sa `[]`.

Pristup elementima niza

- Niz kao operand može da se koristi samo kao prvi operand operatora [] ili operatora sizeof
- Posebno, ne može se operatorom = preneti sadržaj jednog niza u drugi niz.
- To može da se uradi isključivo unutar ciklusa, element po element. (**z17.c**)

Pridruživanje identifikatora

- Nizovnim tipovima mogu da se pridružuju identifikatori tipa pomoću naredbe **typedef**:

```
typedef tip Identifikator_tipa[dužina]...[dužina];
```

- Ako opis tipa predstavlja identifikator tipa koji je jednodimenzionalni niz i ako se za naziv podatka navodi oblik za jednodimenzionalni niz, dobijeni podatak biće dvodimenzionalni.

```
typedef double Niz[10];  
Niz Q[3];
```

- **Niz** je identifikator tipa koji predstavlja niz od 10 realnih brojeva u dvostrukoj tačnosti.
- **q** je matrica od 3 x 10 elemenata (3 vrste i 10 kolona).

Veličina niza

- Ukupno zauzeće memorije nekog niza može da se dobija unarnim operatorom sizeof().
- Operand može da bude identifikator niza ili nizovnog tipa.
- Rezultat je veličina celog niza izražen u bajtovima.
- Veličina niza je uvek jednaka proizvodu veličine jednog elementa i broja elemenata niza.

```
typedef double Niz[10];  
Niz Q[3];
```

- Pretpostavljajući da podaci tipa double zauzimaju po 8 bajtova

```
sizeof(Niz)           // 10x8=80 bajtova  
sizeof Q              // 3x10x8=240 bajtova  
sizeof Q[1]           // iznosi 10x8=80  
sizeof Q[2][7]        // 8 bajtova
```

(z18.c)

Matrice

- Dvodimenzionalni nizovi se zovu i *matrice*.
- Pošto se često koriste, zaslužuju posebnu pažnju.
- Biće vam lakše ako zamislite da su elementi matrice raspoređeni u redove i kolone.
- Na primer, matrica `mat` u narednoj definiciji ima tri reda i pet kolona:

```
float mat[3][5];
```

- Tri elementa `mat[0]`, `mat[1]` i `mat[2]` su redovi matrice `mat`.
- Svaki red predstavlja niz od pet elemenata tipa `float`.
- To znači da matrica sadrži ukupno $3 \times 5 = 15$ elemenata tipa `float`.

```
float mat[3][5];  
for ( int row = 0; row < 3; ++row )  
    for ( int col = 0; col < 5; ++col )  
        mat[row][col] = row + (float)col/10;
```

- Prvi indeks u petlji označava red, a drugi kolonu u tom redu.
- Tri reda se u memoriji smeštaju na susednim lokacijama jer predstavljaju elemente niza mat.
- Sve vrednosti tipa **float** iz ove matrice čuvaju na susednim memorijskim lokacijama po rastućem redosledu adresa.

	0	1	2	3	4
mat[0]	0.0	0.1	0.2	0.3	0.4
mat[1]	1.0	1.1	1.2	1.3	1.4
mat[2]	2.0	2.1	2.2	2.3	2.4