OPERATORI

Dragan dr Dinu

KOMENTARI

- Služe da se umetne tekst namenjen čoveku a ne kompajleru
- Eliminiše ih pretprocesor
- Dobro je komentarisati kod kako bi se u njemu lakše snalazilo
- Dva osnovna tipa:

  /* */ – umeće se bilo gde u kodu, može da ide ispred programskog koda i da se prenese u više redova

  // – važi od mesta definisanja do kraja programske linije

```
/* Program koji
   radi…          */
int main ()
{
```

```
int main()
{
   int i = 1;
   getch(); // prihvati karakter
```

C OPERATORI

- Određuju aktivnost nad operandom
- Izvršavanje rezultuje nekom vrednošću
- Vrednost izvršavanja operatora može poslužiti kao operand za drugi operand
- U C programskom jeziku **sve ima svoju vrednost** s kojom se može manipulisati
- Operatori C programskog jezika:
  - aritmetički
  - relaciono-logički
  - bit-orijentisani
  - dodeljivanje vrednosti
  - tenarni

ARITMETIČKI OPERATORI ...

Unarni operatori		
+	operator zadržavanja predznaka operanda	+ operand
-	operator promene predznaka operanda	- operand
++	prefiksni operator inkrementiranja	++ operand
++	sufiksni operator inkrementiranja	operand ++
--	prefiksni operator dekrementiranja	-- operand
--	sufiksni operator dekrementiranja	operand --

... ARITMETIČKI OPERATORI ...

- Primeri unarnih aritmetičkih operatora

x++;	x= x + 1
++x;	x = x + 1
y= x--;	y = x, x = x – 1
y = --x;	x = x – 1, y = x
(y + x)--;	nije dozvoljeno
++(y + x);	nije dozvoljeno
+ -3.14	-3.14

... ARITMETIČKI OPERATORI ...

Binarni operatori		
+	operator sabiranja	**operand + operand**
–	operator oduzimanja	**operand - operand**
*	operator množenja	**operand * operand**
/	operator deljenja, tip operanada određuje vrstu deljenja, ako su oba celobrajna, rezultat je celobrajan bez ostatka ako je barem jedan razlomljen, rezultat je razlomljen broj	**operand / operand**
%	operator ostatka celobrojnog deljenja, oba operanda moraju biti celobrojni, predznak ostatka je isti kao predznak prvog operanda	**operand % operand**

- Prioritet je isti kao u matematici

... ARITMETIČKI OPERATORI

- Primeri binarnih aritmetičkih operatora

x + y – z	(x + y) – z
x - y * z	x – (y * z)
9 / 2 * 2	8
9. / 2 * 2	9.0
-8 % -3	-2

- Vežba:
  Implementirati program koji prihvata od korisnika dve celobrojne vrednosti i realizuje aritmetičke binarne operacije (+, -, *, /, %) nad njima. Rezultat prikazuje korisniku.

RELACIONI OPERATORI ...

- Ispituje relacije između dva operanda
- Rezultat je:
  - **1 ako relacija važi**
  - **0 ako relacija ne stoji**

Binarni operatori		
<	operator manje	operand < operand
<=	operator manje ili jednako	operand <= operand
>	operator veće	operand > operand
=>	operator veće ili jednako	operand => operand
==	operator jednako	operand == operand
!=	operator nije jednako	operand != operand

... RELACIONI OPERATORI

- Grade relacione izraze

  x < y < z, isto što i **(x < y) < z**

- Ako se ne želi predefinisani redosled operacija, koji zavisi od prioriteta operacija, koristi se zagrada **( )**

  x < (y < z)

  Tako će:

  x == y => z, zbog prioriteta operacija će biti **x == (y => z)**

- Operacije **<**, **<=**, **>**, **=>** su većeg prioriteta od **==**, **!=**

LOGIČKI OPERATORI ...

- Formiraju logičke izraze
- Rezultat je:
  - **1 za istinu**
  - **0 za neistinu**
- Rade na nivou čitavog operanda

Logički operatori		
!	prefiksni unarni operator logičke negacije	**! operand**
&&	binarni operator logičke I operacije	**operand && operand**
||	binarni operator logičke ILI operacije	**operand || operand**

... LOGIČKI OPERATORI ...

- Primeri logičkih operatora:

!0 == 1	!1 == 0	!99 == 0	
0 && 0 == 0	0 && 1 == 0	17 && 0 == 0	(-14) && 5 == 1
0 || 0 == 0	0 || 1 == 1	-12 || 0 == 1	33 || 39 == 1

- Koriste se najčešće kao veznici u relacionim izrazima
- C kompajler skraćuje izvršavanje ako je rezultat jasan već u nekom koraku

... LOGIČKI OPERATORI

- Primeri upotrebe logičkih operatora:

x && y || z && k;	(x && y) || (z && k)
a == !b && c < d ;	(a == (!b)) && (c < d)
7 + 2 || ++x;	skraceno do istine
10 % 5 && x != y;	skraceno do neistine

BIT-ORIJENTISANI OPERATORI ...

- Ne mešati sa logičkim operatorima
- Rade na nivou pojedinačnih bita

Bit-orijentisani operatori		
~	prefiksni unarni operator logičke negacije	**! operand**
&	bit-orijentisani operator logičke I operacije	**operand & operand**
|	bit-orijentisani operator logičke ILI operacije	**operand | operand**
^	bit-orijentisani operator logičke ekskluzivno ili operacije	**operand ^ operand**
<<	bit-orijentisani operator pomeranja ulevo	**operand << br_pomeranja**
>>	bit-orijentisani operator pomeranja udesno	**operand >> br_pomeranja**

... BIT-ORIJENTISANI OPERATORI ...

- Primeri upotrebe bit-orijentisanih operatora...

unsigned char uc1 = 110, uc2 = 3; signed char sc1 = 16, sc2 = -32;

~uc1 = ~(109) = ~01101110 = 10010001 = 145

---

uc1 & uc2 = 110 & 3 = 2

```
  01101110
& 00000011
----------
  00000010
```

---

uc1 | uc2 = 110 | 3 = 111

```
  01101110
| 00000011
----------
  01101111
```

---

uc1 ^ uc2 = 110 ^ 3 = 109

```
  01101110
^ 00000011
----------
  01101101
```

... BIT-ORIJENTISANI OPERATORI

unsigned char uc1 = 110, uc2 = 3; signed char sc1 = 16, sc2 = -32;

---

uc1 << uc2 = 110 << 3 = 112

01101110
11011100
10111000
01110000

---

uc1 >> uc2 = 110 >> 3 = 13

01101110
00110111
00011011
00001101

---

sc1 >> uc2 = 16 >> 3 = 2

00010000
00001000
00000100
00000010

---

sc2 >> uc2 = -32 >> 3 = -4

11100000
11110000
11111000
11111100

OPERATORI DODELE VREDNOSTI ...

- Jedna od osnovnih aktivnosti računara
- Sama po sebi vraća vrednost

x = 12	promenljiva **x** dobija vrednost **12**
y = x = 3	i promenljiva **x** i promenljiva **y** dobijaju vrednost **3**
k = (z = x * 10) * y	promenljiva **z** dobija vrednost **30** promenljiva **k** dobija vrednost **90**

- Obratiti pažnju na razliku između **==** i **=**, iako jedno proverava jednakost a drugo dodeljuje vrednost, obe vraćaju vrednosti pa se mogu koristiti i u logičkim izrazima

... OPERATORI DODELE VREDNOSTI

Binarni operatori		
=	satandardno dodeljivanje vrednosti	x = y
+=	sabiranje i dodeljivanje vrednosti	x += y , x = x + y
-=	oduzimanje i dodeljivanje vrednosti	x -= y , x = x - y
*=	množenje i dodeljivanje vrednosti	x *= y , x = x * y
/=	deljenje i dodeljivanje vrednosti	x /= y , x = x / y
%=	ostatak celobrojnog deljenja i dodeljivanje vrednosti	x %= y , x = x % y
...	...	...

TENARNI (USLOVNI) OPERATOR/IZRAZ

- Tenarni operator:

  ? :

- Deo je uslovnog izraza:

```
izraz1 ? izraz2 : izraz3;
```

- Skoro pa ekvivalentno:

```
if (izraz1)
    izraz2;
else
    izraz3;
```

TENARNI IZRAZ PRIMERI


```
#define MIN(a, b) ((a < b) ? a : b)
#define MAX(a, b) ((a > b) ? a : b)

int a = 13, b = 5;
int veci  = MAX(a, b);
int manji = MIN(a, b);

...

veci = ((a > b) ? a : b);
// veci = ((13 > 5) ? 13 : 5) = ((1) ? 13 : 5) = 13;

...

manji = ((a < b) ? a : b);
// manji = ((13 < 5) ? 13 : 5) = ((0) ? 13 : 5) = 5;

...

printf( "Manja od dve vrednosti (%d, %d) je %d\n", a, b, MIN(a, b));
```

PRIORITETI IZVRŠAVANJA OPERATOR

- Čest izvor grešaka, razmišljati o prioritetu operatora
- Može se regulisati pomoću zagrad, ali se unose dodatne operacije
- C ima čak **15 nivoa operatora**, od najvišeg ka najnižem:

1. **() [ ] -> .**
2. **- + -- ++ ! ~ * & sizeof ()**
3. *** / % + -**
4. **<< >>**
5. **< <= > >= == !=**
6. **& ^ | && ||**
7. **? :**
8. **= += -= *= /= %= &= |= ^= >>= <<=**
9. **,**

- Neke ćemo tek raditi ...

OSNOVNO O DEBUGGING-U

DEBUGGING

- Proces pronalaženja i ispravljanja grešaka u kodu
- Fundamentalna veština u programiranju
- **Najbolje je ne uvoditi greške u kod uopšte!**
  - Pronaći testirano parče koda koji radi ono što želite od pre i uključiti ga u ono što trenutno radite (**reuse**)
  - Razmisliti o problemu, osmisliti rešenje, pa tek onda kodirati (**design**)
  - Koristiti preporučene metode/tehnike kako bi se izbegle uobičajene greške (**best practices**)
- **Pronaći i ispraviti greške u ranim fazama razvoja**
  - Testirati dizajn
  - Testirati svako novo parče koda
  - Testirati kod kao celinu

KONVERZIJE TIPOVA

KONVERZIJE TIPOVA

- Konverzija tipova predstavlja pretvaranje vrednosti jednog tipa u vrednost drugog tipa
- Programski jezik C je veoma fleksibilan po pitanju konverzije tipova
- U mnogim situacijama dopušta korišćenje vrednosti jednog tipa na mestu na kome se očekuje vrednost drugog tipa
- **Programski jezik C podržava dve vrste konverzije tipova:**
  - Implicitnu – kada kompajler prepozna da postoji potreba za konverzijom, pa manji tip konvertuje u veći
  - Eksplicitnu
- Neke konverzije je mogu će izvesti bez gubitka informacija, dok se u nekim slučajevima prilikom konverzije vrši izmena same vrednosti podatka

VRSTE KONVERZIJE TIPOVA

- **Implicitna konverzija tipova** – kada kompajler prepozna da postoji potreba za konverzijom, pa manji tip konvertuje u veći

```
int a = 2;   // a = 2
float a = 2; // a = 2.0 (implicitna konverzija)
```

- **Eksplicitna konverzije tipova** – kada programer eksplicitno navede u koji tip želi da se napravi konverzija (voditi računa, ne moraju biti podržane konverzije između svih tipova)

```
int a = (int) 51.4;   // a = 51 (eksplicitna)
float b = (float)2/3; // b = 0.6666 (eksplicitna)
```

NAPREDOVNJE TIPA

- Neke konverzije mogu se izvesti bez gubitka informacija
- U nekim slučajevima prilikom konverzije vrši izmena same vrednosti podatka
- Jedan oblik konverzije predstavlja **konverzija vrednosti „nižeg tipa“ u vrednost „višeg tipa“** (ovo se naziva promocija ili napredovanje) i ne dolazi do gubitka informacija
  - na primer, short u long, int u float ili float u double

```
float a = 2;
```

(2 je int i implicitno se konvertuje u float 2.0f bez gubitka informacija)

NAZADOVANJE TIPOVA …

- Drugi oblik konverzije predstavlja konverzija vrednosti „višeg tipa“ u vrednost „nižeg tipa“
  - na primer, long u short, double u int
- Ovaj oblik konverzije ponekad se naziva democija ili nazadovanje
- Prilikom ovog oblika konverzije može doći do gubitka informacija (u slučaju da se polazna vrednost ne može predstaviti u okviru novog tipa)

```
int a = 51.0f;
```

(51.0f je float i implicitno se konvertuje u int 51)

```
int a = 51.4f;
```

(51.4f je float i implicitno se konvertuje u int 51 i dolazi do gubitka informacija)

… NAZADOVANJE TIPOVA

- Prilikom konverzije iz brojeva u pokretnom zarezu u celobrojne tipove podataka i obratno potrebno je potpuno izmeniti interni zapis podataka
- Dolazi do odbacivanja (odsecanja decimala), a ne do zaokruživanja
- Kada se konvertuju celobrojni tipovi koji se isto interno zapisuju ali su različite širine (ne zauzimaju isti broj bajtova), dolazi do odsecanja vodećih bitova u zapisu (kod nazadovanja)

```
unsigned char a = 300;
```

(a dobija vrednost 0, jer je 300 veći od najveće vrednosti koje unsigned char može predstaviti)

- Kod napredovanja, kada se konvertuju celobrojni tipovi koji se isto interno zapisuju, ali su različite širine, dolazi do proširivanja zapisa vodećim bitovima