III - Konstante

- Svaki izraz koji se pojavljuje u programu je promenljiva, konstanta, poziv funkcije ili složen izraz.
- Konstanta je objekat u programu koji ima ime i koji dobija konkretnu vrednost jednom pre početka izvršavanja programa i ne menja ga više.
- Konstante su fiksne vrednosti kao: **0**, **2**, **2017**, **3.5**, **1.4e2** ili 'a'.
- Ista vrednost se ponekad može predstaviti različitim konstantama.
- Za sve konstante i za sve izraze, pravilima jezika jednoznačno su određeni njihovi tipovi.
- Poznavanje tipova konstanti i izraza je važno jer od tih tipova zavisi vrednost složenog izraza u kojem figuriše konstanta ili neki pod-izraz.
- Od tipova konstanti i izraza zavisi i koje operacije je moguće primeniti
- Tipovi konstanti i izraza su neophodni i da bise znalo koliko memorije treba rezervisati za neke među rezultate u fazi izvršavanja.
- Konstanta se definiše rezervisanom reči **define** pre početka glavne funkcije programa (pre void main()). **Primer**: **#define PI 3.14**;
- Razlikujemo: celobrojne, realne i znakovne (karakter) konstante

III – Celobrojne konstante

- Za celobrojne konstante u programu najčešće se koristi tip **int**: 100,123
- Ako konstante ne mogu biti predstavljene tipom **int**, korisiti se tip **long**
- Ako ne mogu da budu reprezentovane ni tipom **long**, onda se koristi tip **unsigned long**.
- Tačan tip dekadne celobrojne konstante ne može da se odredi ako se ne znaju detalji sistema.
- Na primer, na većini sistema poziv printf ("%d %d", 2147483647, 2147483648) ispisuje 2147483647 -2147483648 jer se 2147483647 tumači kao konstanta tipa **int**, dok se 2147483648 tumači kao konstanta tipa **unsigned long**, što ne odgovara formatu %d.
- Ukoliko se želi da se neka celobrojna konstanta tretira kao da ima tip **unsigned**, onda se na njenom kraju zapisuje slovo **u** ili **U**.
- Tip takve konstante biće **unsigned long**, ako ona može da bude reprezentovana tim tipom, a **unsigned int** inače.
- Ukoliko se želi eksplicitno naglasiti da se neka celobrojna konstanta tretira kao da je tipa long,onda se na njenom kraju zapisuje slovo **l** ili **L**
- Primer: 12345l je tip long, 12345 je tip int, a 12345ul je unsigned long.

III – Celobrojne konstante

- Osim u **dekadnom**, celobrojne konstante mogu biti zapisane i u **oktalnom** i u **heksadekadnom** sistemu.
- Zapis konstante u oktalnom sistemu počinje cifrom **0**, a zapis konstante u heksadekadnom sistemu počinje simbolima **0x** ili **0X**.
- Na primer, broj 31 se može u programu zapisati na sledeće načine: **31** (dekadni zapis), **037** (oktalni zapis), **0x1f** (heksadekadni zapis).
- I oktalne i heksadekadne konstante mogu da budu označene slovima **U** ili **u** tj. **l** i **L** na kraju svog zapisa.
- Negativne konstante ne postoje, ali se efekat može postići izrazima gde se ispred konstante navodi **unarni operator -**.
- Slično, može se navesti i operator plus, ali to nema efekta
- Primeri: **+123** je isto kao i **123**.

```
123, 987, -765      //decimalne konstante tipa int
0456, -0547         //oktalne konstante tipa int
0x1a2, 0XA0F        //hexadekadne konstante tipa int
0x1a2u              //hexa konstanta tipa unsigned int
8987655ul           //dekadna konstanta tipa unsigned long
```

III – Realne konstante

- Realne konstante se pišu u dekadnom brojnom sistemu i mogu biti predstavljene u fiksnom zarezu i eksponencijalnom obliku.
- Konstante u fiksnom zarezu obavezno sadrže decimalnu tačku i to tačno tamo gde je njeno stvarno mesto u broju (npr. **34.67, 56., .98**).
- Vrednosti ispred i iza decimalne tačke mogu biti izostavljene (ali ne istovremeno).
- Na primer, ispravne konstante su i **.98** ili **56.** (ali ne i **.**).
- Brojevi su označeni i konstante mogu počinjati znakom **-** ili znakom **+** (koji ne proizvodi nikakav efekat).
- Za veoma velike ili veoma male brojeve koristi se eksponencijalni zapis oblika: ***m*E*e*** gde ***m*** predstavlja mantisu broja, a ***e*** eksponent (npr. 2.23e2, -0.5e6, .2E-2, 7e-23)
- Po *default*-u realne konstante su tipa **double**.
- Dodavanjem sufiksa: **F** ili **f** dobijaju se konstante tipa **float** (npr. 1.23f)
- Slova **L** i **l** na kraju zapisa označavaju da je tip vrednosti **long double**.

III – Znakovne konstante

- **Znakovne konstante** - to su konstante tipa **char**.
- Navode se unutar jednostrukih navodnika.
- Znakovna konstanta je ceo broj čija je vrednost jednaka kodu znaka navedenog između jednostrukih navodnika (npr. **'a', '>'**,...).
- Na primer, u ASCII skupu znakova, znakovna konstanta ‘5’ ima vrednost 53, ‘A’ ima vrednost 65.
- ASCII tabela znakova sadrži i izvestan broj upravljačkih simbola (simboli koji ne ostavljaju trag na ekranu ili štampaču).
- Za takve znakove u C-u su uvedene određene simboličke oznake:

\n	prelaz u sledeći red
\t	horizontalna tabulacija
\v	vertikalna tabulacija
\b	znak za vraćanje (backspace)
\f	nova strana (form feed)
****	obrnuta kosa crta (backslash)
\?	znak pitanja
\'	apostrof
\”	znaci navoda(dvostruki)

III - Literali - Stringovi

- Literal je sažeti način zapisivanja niza znakovnih konstanti.
- Na primer: **"Ovo je literal"**
- C prevodilac svaki znak prevodi u jedan ceo broj tipa char (ASCII ekvivalent) i na kraju dodaje još jedan znak sa vrednošću 0 (‘\0’), tzv. *null terminated symbol*.
- **Primer**:**"Zdravo“** se u memoriji zapisuje kao **‘Z’ ‘d’ ‘r’ ‘a’ ‘v’ ‘o’ ‘\0’**

korektno

```
" a string of text"
""
"   "
"a = b+c;"
"/* this is not a comment */"
"a string with double quotes \" within"
"a single backslash \\ is in this string"
"abd" "efg"
```

nekorektno

```
/* "this is not a string" */
"and
Neither is this "
'dgashgahg'
```

III – Operatori

- U C jeziku postoje **tri** osnovna tipa podataka: **numerički**, **logički** i **znakovni** koji određuju operatore koji se mogu nad njima primenjivati
- Operatori su simboli koji označavaju određenu operaciju i koji povezuju jedan ili više operanada (promenljivh) u izraz.
- Operatori se klasifikuju po broju operanada i po funkcionalnosti
- U zavisnosti od **broja operanada** operatori u C-u se dele na:
  - **I. Unarne** – operatori nad jednim operandom
  - **II. Binarne** – operatori nad dva operanda
  - **III. Ternarne** – operatori nad tri operanda
- Grupe operatora u odnosu na **funkcionalnost**:

1. operatori za pristup članovima polja i struktura
2. operator za poziv funkcije
3. aritmetički operatori
4. relacioni operatori
5. logički operatori
6. operatori za rad sa bitovima
7. operatori dodeljivanja vrednosti
8. operator grananja
9. sizeof- operator
10. comma operator
11. cast operator
12. operatori referenciranja i dereferenciranja

III – Numerički operatori

- Imaju numeričke operande i rezultati su, takođe, numeričkog tipa.
- **Po prioritetu:**

1. Unarni operatori + i -
2. Operatori inkrementiranja (++) i dekrementiranja (--)
3. Operatori *, /, %
4. Binarni operatori + i -

1. **Unarni operatori + i -**

- Unarni operator - menja znak operanda koji sledi.
- Ako je operand unsigned, rezultat se računa oduzimanjem operanda od 2*n* , gde je *n* broj bitova u binarnoj predstavi tipa rezultata.
- Unarni operator + zadržava znak operanda koji sledi (to znači da je on praktično operator bez dejstva).

III – Numerički operatori

2. **Operatori inkrementiranja (++) i dekrementiranja (--)**

- Ovi operatori su unarni i mogu biti prefiksni i postfiksni.
- Oni vrše povećanje (smanjenje) vrednosti svog operanda za 1.
- Rezultat izraza sa prefiksnim operatorima ++ (ili --) je nova vrednost, a rezultat izraza sa postfiksnim operatorom je stara vrednost operanda
- To znači da ukoliko se vrednost izraza ++a/--a koristi u nekom širem izrazu, prvo će se izvršiti promena vrednosti promenljive a i ta nova vrednost će se iskoristiti za izračunavanje vrednosti šireg izraza.
- Ukoliko se vrednost izraza a++ (ili a--) koristi u nekom širem izrazu, izračunaće se vrednosti šireg izraza sa starom vrednošću promenljive a i nakon toga će se izvršiti promena vrednosti promenljive a.

Primer: a=1; b=++a +5; (a=2, b=7)
a=1; b=a-- +5; (a=0, b=6)

III – Numerički operatori

3. **Operatori *, /, %**

***** - označava množenje;

/ - označava deljenje (u slučaju primene nad celobrojnim podacima predstavlja celobrojno deljenje, tj. u rezultatu se vrši jednostavno odbacivanje cifara iza decimalne tačke): 5 / 2 = 2

% - (po modulu) označava ostatak deljenja i primenjuje se isključivo nad celobrojnim podacima: 5 % 2 = 1

4. **Binarni operatori + i -**

+ - označava sabiranje,

– - označava oduzimanje

Primeri:

```
int a,b,c,d;
a=3; b=6;
c=a*--b;  //ekvivalentne naredbe: b=b-1; c=a*b;
d=a*b--;  //ekvivalentne naredbe: d=a*b; b=b-1;
```

III – Relacioni operatori

- Ovo su operatori za poređenje vrednosti operanada.
- Operandi u ovom slučaju mogu biti bilo kog numeričkog tipa ili pokazivači, a rezultat je logičkog tipa.
- Kako logički tip u C-u nije definisan, ukoliko je poređenje tačno (tj. ukoliko je relacija zadovoljena), rezultat je 1 u suprotnom rezultat je 0
- Ovoj grupi operatora pripadaju:

1. `<` (manje),
2. `<=` (manje ili jednako),
3. `>` (veće),
4. `>=` (veće ili jednako),
5. `==` (jednako),
6. `!=` (različito).

- Prioritet prva 4 operatora je viši od prioriteta poslednja dva.

III – Logički operatori

- U programskom jeziku C, operandi u logičkim operacijama mogu biti:
  - numeričkog tipa
  - tipa pokazivača.
- S obzirom da su ovi operatori prirodno (u matematici) primenjuju nad logičkim podacima, pri izvršenju ovih operacija svaka vrednost različita od nule se tretira kao logička vrednost tačno (*true*), jedino 0 označava netačno (*false*).
- **U ovu grupu operatora spadaju:**
  - **!** (negacija),
  - **&&** (konjukcija, tj. logičko I),
  - **||** (disjunkcija, tj. logičko ILI).

prioritet ↓

- Rezultat operacije je 1 ako je rezultat logičke operacije tačno, u suprotnom, rezultat je 0.

III – Operatori za rad sa bitovima

- Izvođenje operacija nad bitovima unutar celobrojnih podataka.
- Ovakvi operatori su definisanu u asemblerskim jezicima, a od viših programskih jezika, jedino ih C podržava.

- **Ovoj grupi operatora pripadaju**:

1. ~ nepotpuni (jedinični) komplement – komplementira se svaki bit ponaosob,
2. & pokomponento logičko I,
3. ^ pokomponentno isključivo ILI,
4. | pokomponentno logičko ILI,
5. << pomeranje ulevo - vrši pomeranje bitova levog operanda za onoliko mesta ulevo kolika je vrednost desnog operanda,
6. >> pomeranje udesno - vrši pomeranje bitova levog operanda za onoliko mesta udesno kolika je vrednost desnog operanda.

III – Operatori za rad sa bitovima

Primeri:

```
int a,b;
a = 0723;          // mem. lok a: 0000000111010011
b = ~a;            // b: 1111111000101100
a = a << 4;        // a: 0001110100110000
a = a >> 3;        // a: 0000001110100110
b = a | 071;       // b: 0000001110111111
```

III – Operatori dodele

- Ovo je binarni operator koji smešta vrednost desnog operanda na lokaciju čije se ime nalazi sa leve strane operatora.
- Levi operand mora biti ime memorijske lokacije.

- Postoje dve grupe ovakvog operatora:

1. elementarni operator dodeljivanja (=),
2. složeni operatori dodeljivanja (opšti oblik ovih operatora je **<op>=**)

- Operator <op> može biti jedan od sledećih:

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

- Opšti izraz a <op> = b se može napisati i u obliku a = a <op> b
- **Primeri:**

```
a += 3;                 //ekvivalntno:  a = a+3;
b -= a;                 //ekvivalentno:b = b-a;
proizvod *= a-5;        // proizvod = proizvod*(a-5);
```

III – Operatori grananja

- Ovo je jedini ternarni operator u C-u.
- Opšti oblik izraza kreiranog pomoću ovog operatora je:

<izraz1> ?< izraz2> : <izraz3>

Dejstvo operatora:

- Izračunava se najpre vrednost izraza <izraz1>.
- Ako je vrednost ovog izraza **različita od nule** (*true*), izračunava se vrednost izraza <izraz2> i njegova vrednost je rezultat operacije.
- Ako je vrednost izraza <izraz1> **jednaka nuli** (*false*), izračunava se vrednost izraza <izraz3> i njegova vrednost je rezultat operacije.

Primeri:

```
maximum = ( a >= b)? a : b;

abs_x = ( x >= 0 ) ? x : -x;
```

III - Sizeof operator

- Svaka promenljiva ili konstanta zauzima određeni deo memorijskog prostora.
- Veličina tog memorijskog prostora zavisi od tipa promenljive (konstante), tj. od internog načina predstavljanja podataka pojedinih tipova u memoriji.
- **sizeof** - izračunava broj bajtova koji zauzima neki podatak ili tip u memoriji.
- Ovaj operator je unaran i njegov opšti oblik je:

sizeof (<objekat>) ili sizeof (tip).

Primeri:

```
int broja, brojb;

broja = sizeof (short int);

brojb = sizeof (broja);
```

III – Comma operator

- Ovo je binarni operator.
- Služi za formalno spajanje više izraza u jedan izraz.
- Levi i desni operand operatora "," su izrazi, pri čemu je redosled njihovog izvršavanja s leva na desno. (Uglavnom se koristi kod poziva funkcija)

Primer:

```
a++,  b *= 2,  c = a+b;
```

III – Cast operator

- Koristi se za eksplicitno pretvaranje promenljive (ili izraza) jednog tipa u neki drugi tip.
- Operator je unaran i opšti oblik izraza za konverziju tipa je:

  (<tip>) <izraz>

  gde je: <tip> - neki od tipova,
  <izraz> - je promenljiva ili neki složeniji izraz čiji tip treba promeniti.

Primeri:

```
int   a;
float x;
double y;
... (char) a ...
y=sin((double) x);      //parametar f-je sin mora da bude tipa double
```

III – Operatori u C jeziku

Aritmetički operatori u C jeziku	**Komentar**
Dodela vrednosti (=)	
Aritmetički operatori (+, -, *, /, %)	
Inkrement (++)	Efekti primene izraza su različiti ukoliko stoje pre i posle neke promenljive
Dekrement (--)	Efekti primene izraza su različiti ukoliko stoje pre I posle neke promenljive
+=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=	
Relacioni operatori (==, !=, >, <, >=, <=)	Koristimo ih kod logičkih izraza (npr. IF naredbe)
Logički operatori (!, &&, ||)	Za povezivanje više logičkih izraza
Uslovni operator (**uslov ? iz1; iz2**)	a>b ? a : b
Operatori nad bitovima (&, |, ^, ~, <<, >>)	
sizeof(**data_type**)	x = sizeof (char);
cast operator	int i; float f = 3.14; i = (int) f;

III – Prioritet operatora

- Prioritet operatora određuje redosled izvođenja operacija kada u izrazu figuriše veći broj operatora.
- U izrazima se uvek izvršavaju operatori od operatora najvišeg prioriteta prema operatorima nižih prioriteta.
- Ovaj redosled se može promeniti korišćenjem zagrada.
- U tabeli su poređani operatori programskog jezika C po njihovom prioritetu (počev od operatora najvišeg prioriteta prema operatorima nižih prioriteta).

1. **( ) [ ] . ->**
2. **unarni - ~ & | ++ --**
   sizeof
3. *** / %**
4. **+ -**
5. **<< >>**
6. **< > <= >=**
7. **== !=**
8. **&**
9. **^**
10. **|**
11. **&&**
12. **||**
13. **? :**
14. **= *op*=**
15. **,**

III – Tipovi podataka u C jeziku

- Promenljive u prog. jeziku su objekti koji imaju neku vrednost, pri čemu se vrednost može menjati u toku izvršavanja programa.
- Pri tome ta vrednost mora biti jednog istog tipa
- Svaka promenljiva koja se koristi u C-u mora biti deklarisana i to:
  - ✓ **ime promenljive** (to je ime prema sintaksi jezika C),
  - ✓ **tip promenljive** (tip kome pripada vrednost date promenljive) i
  - ✓ **početna vrednost** (vrednost u početnom trenutku) – neobavezno.
- Time se za promenljivu u memoriji rezerviše potreban prostor, a ostatak programa postaje svestan postojanja promenljive i njenog tipa.
- Deklaracija promenljive ima sledeći zapis:

ime_tipa ime_promenjive [= vrednost];

Primer:

int i, j, n=10; - deklarisane su promenljive celobrojnog tipa *i*, *j* i *n*

float a, e=2.71, pi=3.14; -deklarisane su promenljive realnog tipa *a*, *e* i *pi*

char c, d='?'; - deklarisane su promenljive znakovnog tipa *c* i *d*.

Da bi se promenljiva mogla upotrebljavati u programu ona se mora na početku programa deklarisati!

III – Tipovi podataka u C jeziku

- Sve promenljive u C-u se moraju deklarisati.
- Deklaracija se sastoji iz imena tipa za kojim slede imena promenljivih (identifikatori) koje se deklarišu, i koja su razdvojena zapetama.
- Deklaracija se završava simbolom '**;**'.
- Svako ime promenljive u deklaraciji može biti praćeno inicijalizatorom koji se sastoji iz karaktera '**=**' za kojim sledi inicijalna vrednost**.**
- Promenljiva može sadržati proizvoljno ime ali ono treba da opisuje namenu promenljive, kako bi ona bila lakše prepoznatljiva
- Imena promenljivih mogu da sadrže slova,brojeve i znak donje crte(_).
- Na početku imena se mora naći ili slovo ili donja crta, **broj ne sme**!
- Prilikom deklaracije može se izvršiti i početna inicijalizacija.

Primeri:

```
int broj;                /* Deklaracija celog broja */
int vrednost=5;          /* Deklaracija i inicijalizacija celog broja */
```

- Postoji i kvalifikator ***const*** koji može biti dodeljen deklaraciji bilo koje promenljive da bi označio da se ona neće menjati

Primer: const double e=2.71828182845905;

III – Tipovi podataka u C jeziku

- U program. jeziku C definisani su samo numerički tipovi podataka
- Tipovi podataka se dele na osnovne tipove podataka (*built-in type*) i kreirane (izvedene) tipove podataka (*custom type*).

Osnovni tipovi za predstavljanje celobrojnih podataka

- **int** - celobrojni podatak za čije se predstavlajnje koriste 16 ili 32 bita
- **char** - mali celobrojni podatak (dužine 1 bajt) koji može da primi i kod jednog znaka pa se koristi i za predstavljanje znakovnih podataka.

Tipovi za predstavljanje realnih vrednosti

- **float** - realni podatak u jednostrukoj tačnosti (32 bita)
- **double** - realni podatak u dvostrukoj tačnosti (64 bita)
- Mogu se dobiti različite varijante navedenih tipova podataka ako se ispred oznake osnovnog tipa podataka dodaju sledeće ključne reči:

1. **short** - memor. prostor za predstavljanje podatka se smanjuje na pola,
2. **long** - memorijski prostor za predstavljanje podatka se udvostručava,
3. **unsigned** - kaže da se radi o neoznačenom podatku,
4. **signed** - označeni podatak - po *default*-u svi podaci su označeni

III – Tipovi podataka u C jeziku

- Kvalifikatori **signed** i **unsigned** mogu se pridružiti celobrojnim tipovima **(int, char, short i long).**
- Tipu **int** mogu se pridružiti oba tipa (**short** i **long**) pri čemu navođenje ključne reči **int** nije obavezno jer se podrazumeva
- Tipu **double** može se pridružiti **long**, dok se tipu **float** ne može pridružiti ni jedan kvalifikator.
- Neozančeni brojevi (**unsigned**) su uvek pozitivni ili nula.
- U praksi veličine svih tipova zavise od platforme na kojoj se one realizuju ali na svakom računaru važi sledeće pravilo:

$$broj\ bitova(short) \leq broj\ bitova(int) \leq broj\ bitova(long)$$

Primer:

1. Promenljiva tipa ***signed char*** uzima vrednosti od −128 do 127.
2. Promenljiva tipa ***unsigned char*** uzima vrednosti od 0 do 255.

- Postoji još jedan tip podataka **void** koji označava prazan tip podataka

II – Tipovi podataka u C jeziku

- Osnovni tipovi podataka služe za građenje složenih tipova podataka:
  nizovi
  strukture
  unije
- Promenjive bilo kog tipa podataka se deklarišu, odnosno najavljuju, tako što napišemo tip promenjive pa zatim ime promenjive, ili celu listu promenjivih tog tipa koje se koriste u programu.
- U listi promenjive se razdvajaju zarezima, a po potrebi im se odmah mogu dodeljivati vrednosti. **Primeri:** **char ch;** **int a,b=1,c;**
- Svaki tip podataka definiše:
  1. Količinu memorije koju će promenljiva zauzeti u radnoj memoriji
  2. Maksimalnu i minimalnu vrednost koju promenljiva može da ima
  3. Tip operacija koje su dozvoljene sa ovim tipom promenljive
  4. Mesto u memoriji gde će promenljiva biti zapamćena u trenutku izvršavanja (*run time)*
- Kod složenih tipova podataka definiše se i članovi i metode klase koju taj tip promenljive sadrži, kao i klase koje on nasleđuje

II – Tipovi podataka u C jeziku

- Informacije o tipovima podataka koristi kompajler kako bi proverio da li su sve operacije koje se obavljaju u progr.kodu sigurne (*type safe*).
- **Primer:** ako je deklarisana promenljiva tipa **int**, kompajler dozvoljava da se ona koristi u operacijama sabiranja i oduzimanja.
- Ukoliko iste ove operacija probamo da izvršimo sa tipom promenljive **bool**, kompajler će generisati grešku kao u sledećem primeru.

```
int a = 5;
int b = a + 2; //OK
bool test = true;
// Error. Operator '+' cannot be applied to operands of type 'int'
and 'bool'.
int c = a + test;
```

II – Tipovi podataka u C jeziku

Tip promenljive	*Ključna reč*	*Broj bajtova*	*Opseg*
Character	char	1	-128 do 127
Unsigned character	unsigned char	1	0 do 255
Integer	int	2	-32768 do 32767
Short Integer	short int	2	-32768 do 32767
Long integer	long int	4	-2147483648 do 21474368647
Unsigned integer	unsigned int	2	0 do 65535
Float	float	4	3.4E-38 do 3.4E38
Double	double	8	-1.7E308 do 1.7E308
Long double	long double	10	3.4E-4932 do 1.1E4932

II – Konverzija tipova podataka u C

- Dozvoljene implicitne konverzije podataka su prikazane u tabeli
- U ovom slučaju ne dolazi do gubitka podatala i kompajler neće prjaviti nikakvu grešku niti upozorenje.
- **Primer** implicitne konverzije bez gubitka podataka:

```
int a;    double b = (a = 3.5);   // a će biti 3.0
```

- **Primer** eksplicitne, koristeći cast operator

```
int a = 13, b = 4;
printf("%d\t", a/b);
printf("%f\n", (double)a/(double)b);   // 3.250000
```

Tip podataka	Može se pretvoriti u:
Byte	Char, short, Int, long, float double
Short	Int, long, float double
Int	Long, float double
Long	Float, double
float	Double
Double	l

Level	Precedence group	Operator	Description	Grouping
1	Scope	::	scope qualifier	Left-to-right
2	Postfix (unary)	++ --	postfix increment / decrement	Left-to-right
		()	functional forms	
		[]	subscript	
		. ->	member access	
3	Prefix (unary)	++ --	prefix increment / decrement	Right-to-left
		~ !	bitwise NOT / logical NOT	
		+ -	unary prefix	
		& *	reference / dereference	
		new delete	allocation / deallocation	
		sizeof	parameter pack	
		(*type*)	C-style type-casting	
4	Pointer-to-member	.* ->*	access pointer	Left-to-right
5	Arithmetic: scaling	* / %	multiply, divide, modulo	Left-to-right
6	Arithmetic: addition	+ -	addition, subtraction	Left-to-right
7	Bitwise shift	<< >>	shift left, shift right	Left-to-right
8	Relational	< > <= >=	comparison operators	Left-to-right
9	Equality	== !=	equality / inequality	Left-to-right
10	And	&	bitwise AND	Left-to-right
11	Exclusive or	^	bitwise XOR	Left-to-right
12	Inclusive or	|	bitwise OR	Left-to-right
13	Conjunction	&&	logical AND	Left-to-right
14	Disjunction	||	logical OR	Left-to-right
15	Assignment-level expressions	= *= /= %= += -= >>= <<= &= ^= |=	assignment / compound assignment	Right-to-left
		?:	conditional operator	
16	Sequencing	,	comma separator	Left-to-right

Hvala na pažnji !!!

Pitanja

? ? ?