String konstante

- `"Ja sam string"`
- `"" /* prazan string */`
- String konstante se mogu spajati u vreme kompajliranja
  - `"hello, " "world"` se spaja u `"hello, world"`
- Stringovi su nizovi karaktera koji se završavaju znakom '\0'
- ```
  /* strlen: return length of s */
  int strlen(char s[])
  {
    int i=0;

    while (s[i] != '\0')
      ++i;

    return i;
  }
  ```
- Funkcije za rad sa stringovima: `<string.h>`
  ```

String konstante

- **'a'** nije isto što i **"a"**

- **'a' –** ceo broj koji predstavlja numeričku vrednost slova **a** u skupu znakova koje računar koristi
- **"a"** – niz znakova koji sadrži znak `'a'` i `'\0'`

Nabrojive konstante

```
enum <naziv_tipa> {vrednost1, vrednost2,...};
```

Primer

```
enum logicki {NETACNO, TACNO};

enum months { JAN = 1, FEB, MAR, APR, MAY, JUN, JUL, AUG,
  SEP, OCT, NOV, DEC };
enum escapes { BELL = '\a', TAB = '\t', NEWLINE = '\n',
  VTAB = '\v',  RETURN = '\r' };
```

Imena u različitim nabrojivim tipovima moraju biti različita.

Vrednosti u istom nabrojivom tipu ne moraju biti različite.

Deklaracije

```
int lower, upper, step;
char c, line[1000];
```

ili

```
int lower;
int upper;
int step;
char c;
char line[1000];
```

- Inicijalizacija prilikom deklarisanja

```
int i = 0;
int limit = MAXLINE+1;
float eps = 1.0e-5;
char esc = '\\';
```

- Promenljive čija se vrednost **ne** može menjati

```
const double e = 2.71828182845905;
const char msg[] = "warning: ";

int strlen(const char[]);
```

Podrazumevana incijalizacija

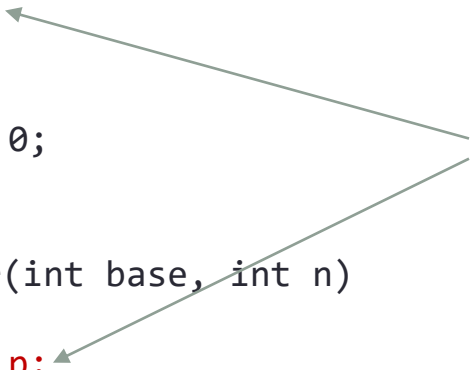
```
#include <stdio.h>

int power(int m, int n);

main()
{
   int i;

...
   return 0;
}

int power(int base, int n)
{
   int i, p;
...
   return p;
}
```



Automatske promenljive nisu podrazumevano incijalizovane!

Spoljašnje i statičke promenljive se inicijalzuju na nulu po podrazumevanom načinu rada.

O njima kasnije!

Aritmetički operatori

- **+, -, *, /** - sabiranje, oduzimanje, množenje i deljenje
- **%** - ostatak pri deljenju

```
if ((godina % 4 == 0 && godina % 100 != 0) || godina % 400 == 0)
  printf("%d je prestupna godina\n", godina);
else
  printf("%d nije prestupna godina\n", godina);
```

`int / int` – celobrojno deljenje!

Relacioni i logički operatori

- Relacioni operatori
  - `>` `>=` `<` `<=`
  - `==` `!=`
- Logički operatori
  - `&&` `||`
  - `if(i<MAX && name[i]=='r')`
  - `if(!ispravno)` umesto `if(ispravno==0)`
  - `if (a+b)` isto što i `if (a+b<>0)`

    0 – netačno

    Sve ostalo – tačno
- Šta je vrednost x, ako je `x = 12>3;` ?

Konverzija tipova

- **int** u **float**
  - f+i
- Automatski – ‘uži’ u ‘širi’ tip
- Dodela vrednosti ‘šireg’ tipa ‘užem’ proizvodi upozorenje.

- Koju vrednost daje izraz 14/4*2.5?

Konverzija tipova

- **char** u **int**

1 ☺	33 !	65 A	97 a	129 ü	161 ¡	193 Á	225 á
2 ☻	34 "	66 B	98 b	130 ,	162 ¢	194 Â	226 â
3 ♥	35 #	67 C	99 c	131 ƒ	163 £	195 Ã	227 ã
4 ♦	36 $	68 D	100 d	132 „	164 ¤	196 Ä	228 ä
5 ♣	37 %	69 E	101 e	133 …	165 ¥	197 Å	229 å
6 ♠	38 &	70 F	102 f	134 †	166 ¦	198 Æ	230 æ
7 •	39 '	71 G	103 g	135 ‡	167 §	199 Ç	231 ç
8 ◘	40 (	72 H	104 h	136 ˆ	168 ¨	200 È	232 è
9	41)	73 I	105 i	137 ‰	169 ©	201 É	233 é
10	42 *	74 J	106 j	138 Š	170 ª	202 Ê	234 ê
11 ♂	43 +	75 K	107 k	139 ‹	171 «	203 Ë	235 ë
12 ♀	44 ,	76 L	108 l	140 Œ	172 ¬	204 Ì	236 ì
13	45 -	77 M	109 m	141 ï	173 -	205 Í	237 í
14 ♫	46 .	78 N	110 n	142 Ž	174 ®	206 Î	238 î
15 ☼	47 /	79 O	111 o	143 ï	175 ¯	207 Ï	239 ï
16 ►	48 0	80 P	112 p	144 ï	176 °	208 Ð	240 ð
17 ◄	49 1	81 Q	113 q	145 '	177 ±	209 Ñ	241 ñ
18 ↕	50 2	82 R	114 r	146 '	178 ²	210 Ò	242 ò
19 ‼	51 3	83 S	115 s	147 "	179 ³	211 Ó	243 ó
20 ¶	52 4	84 T	116 t	148 "	180 ´	212 Ô	244 ô
21 §	53 5	85 U	117 u	149 •	181 µ	213 Õ	245 õ
22 ▬	54 6	86 V	118 v	150 –	182 ¶	214 Ö	246 ö
23 ↨	55 7	87 W	119 w	151 —	183 ·	215 ×	247 ÷
24 ↑	56 8	88 X	120 x	152 ˜	184 ¸	216 Ø	248 ø
25 ↓	57 9	89 Y	121 y	153 ™	185 ¹	217 Ù	249 ù
26 →	58 :	90 Z	122 z	154 š	186 º	218 Ú	250 ú
27 ←	59 ;	91 [	123 {	155 ›	187 »	219 Û	251 û
28	60 <	92 \	124 |	156 œ	188 ¼	220 Ü	252 ü
29	61 =	93]	125 }	157 ï	189 ½	221 Ý	253 ý
30	62 >	94 ^	126 ~	158 ž	190 ¾	222 Þ	254 þ
31	63 ?	95 _	127 ⌂	159 Ÿ	191 ¿	223 ß	255 ÿ
32	64 @	96 `	128 €	160	192 À	224 à	

Konverzija char u int

```
/* lower: convert c to lower case; ASCII only */
int lower(char c)
{
  if (c >= 'A' && c <= 'Z')
       return c + 'a' - 'A';
  else
       return c;
}
```

- **Funkcije za rad sa karakterima**: `<ctype.h>`

- Da li pretvaranja `char` u `int` može proizvesti negativan broj?
- Svi vidljivi znaci nikad neće biti negativni
- Radi portabilnosti navodi se `signed` ili `unsigned` ako se podaci različiti od znakova čuvaju u promenljivim tipa `char`.

Eksplicitna konverzija

- **cast** operator

  (novi tip) izraz

```
double sqrt(double x)
{
  ...
}

...

int n;
sqrt((double) n )

...

root = sqrt(2)
```

Inkrement i dekrement

- **++** - uvećava vrednost za jedan
- **--** - umanjuje vrednost za jedan

```
n=5;

x=++n;      /* PREFIKSNA upotreba
               prvo se n uvecava za 1, pa x dobija vrednost 6 */
x=n++;      /* POSTFIKSNA upotreba
               prvo x dobija vrednost 5, pa se n uvecava za 1 */
```

```
s[i++] = c;          s[i] = c;          s[i] = c;
                     i++;               ++i;
```

```
x=(a+b)++; nije dozvoljeno      x=a+b++;  dozvoljeno
```

```
/* squeeze: delete all c from s */
void squeeze(char s[], int c)
{
   int i, j;

   for (i = j = 0; s[i] != '\0'; i++)
      if (s[i] != c)
         s[j++] = s[i];

   s[j] = '\0';
}
```

s = “bacaab” c = ‘a’

i,j = 0 s[0] != ‘a’ ? s[0] = s[0] = ‘b’, j = 1

i = 1 s[1] != ‘a’ ?

i = 2 s[2] != ‘a’ ? s[1] = s[2] = ‘c’, j = 2

i = 3 s[3] != ‘a’ ?

i = 4 s[4] != ‘a’ ?

i = 5 s[5] != ‘a’ ? s[2] = s[5] = ‘b’, j = 3

s[3] = ‘\0’ → s = “bcb”

```
/* strcat: concatenate t to end of s; s must be big enough */
void strcat(char s[], char t[])
{
   int i, j;

   i = j = 0;
   while (s[i] != '\0')   i++;            /* find end of s */

   while ((s[i++] = t[j++]) != '\0');     /* copy t */
}
```

Binarni operatori

- Operatori za manipulisanje bitovima, koji se mogu primeniti samo na celobrojne operande
  - `&` - logičko I nad bitovima
  - `|` - logičko ILI nad bitovima
  - `^` - logičko isključujuće ILI nad bitovima
  - `<<` - pomeranje u levo
  - `>>` - pomeranje u desno
  - `~` - komplement jedinice (unarni operator)

```
n = 389  --   1 1000 0101
   0177 --      111 1111
&              0 0000 0101 --  5
```

```
n = 389  --   01 1000 0101
  ~077 --    11 1100 0000
&              01 1000 0000 --  384
```

- Šta je rezultat sledećih komandi?

```
n = n & 0177;
n = n & ~077;
n = n << 2;
```

```
n = 389   --   1 1000 0101
389 << 2 --   1 1000 010100 -- 1556
```

- Kako postaviti 1 na bitu najmanje težine broja koji je zabeležen u promenljivoj n, bez promene ostalih bitova?

Binarni operatori

- Ako je x=1 i y=2, šta se dobija kao rezultat operacija
  `x & y` i `x && y`?

- Odrediti n-bitno polje u promenljivoj x, počev od pozicije p

```
/* getbits: get n bits from position p */
unsigned getbits (unsigned x, int p, int n)
{
        return (x >> (p+1-n) & ~(~0 << n));
}
```

- Šta je dobija kao rezultat poziva `getbits(315,4,3)`?

Binarni operatori

- Ako je x=1 i y=2, šta se dobija kao rezultat operacija
  `x & y` i `x && y`?

- Odrediti n-bitno polje u promenljivoj x, počev od pozicije p

```
/* getbits: get n bits from position p */
unsigned getbits (unsigned x, int p, int n)
{
        return (x >> (p+1-n) & ~(~0 << n));
}
```

- Šta je dobija kao rezultat poziva `getbits(315,4,3)`?

315	0000000100111011
315 >> (4+1-3)	0000000001001110
~0 << 3	1111111111111000
~(~0 << 3)	0000000000000111
315 >> 2 & ~(~0 << 3)	0000000000000110

Operatori i izrazi dodeljivanja

- `i = i + 2` `i += 2`
- `i = i * 3` `i *= 3`
- Za bilo koji operator *op* iz skupa

  `+` `-` `*` `/` `%` `<<` `>>` `&` `^` `|`

Operator dodeljivanja

izr1 op= izraz2 *je ekvivalentno sa* ***izr1 = (izr1) op (izr2)***

- Bitno:

  `x *= y + 1` je ekvivalentno sa `x = x * (y + 1)`

  a ne `x = x * y + 1`

Operatori i izrazi dodeljivanja

- Naredba dodeljivanja ima vrednost pa se može koristiti u izrazima, kao na primer

```
while ((c = getchar()) != EOF) {…}
```

- Tip izraza dodeljivanja je tip njegovog levog operanda

- Čitljiviji kod

```
yyval[yypv[p3+p4]+yypv[p1+p2]] += 2
```

Operatori i izrazi dodeljivanja

- Broj bitova jedinica u broju

```
/* bitcount: count 1 bits in x */
int bitcount(unsigned x)
{
    int b;
    for (b=0; x!=0; x>>=1)
        if (x & 01) b++;
    return b;
}
```

- Zašto **unsigned** kao tip argumenta?

Uslovni izrazi

- Određivanje maksimuma od *a* i *b*

```
if (a > b)
  z = a;
else
  z = b;
```

- Ternarni operator

```
izr1 ? izr2 : izr3
```

```
z = (a > b) ? a : b; /* z = max(a, b) */
```

Uslovni izrazi

- Pravila konverzije tipova

```
(n > 0) ? f : n
```

je izraz tipa float

- Šta je rezultat sledeće naredbe?

```
for(i = 0; i < n; i++)
    printf("%6d%c", a[i], (i%10==9 || i==n-1) ? '\n' : ' ');
```

Prioritet operatora

Operatori	Asocijativnost
`() [] -> .`	sa leva na desno
`! ~ ++ -- + - * (type)sizeof`	sa desna na levo
`* / %`	sa leva na desno
`+ -`	sa leva na desno
`<< >>`	sa leva na desno
`< <= > >=`	sa leva na desno
`== !=`	sa leva na desno
`&`	sa leva na desno
`^`	sa leva na desno
`|`	sa leva na desno
`&&`	sa leva na desno
`||`	sa leva na desno
`?:`	sa desna na levo
`= += -= *= /= %= &= ^= |= <<= >>=`	sa desna na levo
`,`	sa leva na desno

Prioritet operatora

- Redosled po kome se izračunavaju operandi nije uvek definisan – izuzeci su opreatori &&, ||, ?: i ,

```
x = f() + g();
```

- Primeri loših poziva:

```
printf("%d %d\n", ++n, power(2,n));

a[i] = i++;
```

Kontrola toka

- Naredbe
- Blokovi
- If-Else
- Else-If
- Switch
- While
- For
  - While ili For?
- Do-While
- Break i Continue
- ~~Goto~~

FUNKCIJE I STRUKTURA PROGRAMA

Funkcije

- Razbijaju velike računarske zadatke u manje delove
- Skrivaju detalje postupka od delova programa koji ne moraju da znaju o njima
- Čine program jasnijim i jednostavnijim za menjanje

Struktura programa

- Programi se najčešće sastoje od više manjih funkcija
- Program se može nalaziti u jednoj ili više izvornih datoteka
- Izvorne datoteke se mogu kompajlirati odvojeno i od njih se može formirati izvršni program

Definicija i deklaracija funkcije

- Primer **definicije** funkcije:

```
tip_rez ime_funk (tip1 param1, tip2 param2, ..., tipN paramN)
{
        /*telo funkcije*/
}
```

- Primer **deklaracije** funkcije:

```
tip_rez ime_funk (tip1 [param1], tip2 [param2], ..., tipN [paramN]);
```

- Prazna funkcija

```
prazna()  {}
```

Definicija i deklaracija funkcije

- Poziv funkcije u programu se ostvaruje navođenjem imena funkcije i liste stvarnih argumenata funkcije

```
faktorijel(5);
```

- Mehanizam za vraćanje vrednosti iz funkcije predstavlja naredba **return**

```
return izraz;
```

- Vrednost može biti bilo koji izraz.
- Tip izraza se pretvara u tip rezultata ako je neophodno.
- Ako tip rezultata nije naveden podrazumeva se int.
- Pozivalac može ignorisati vraćenu vrednost.
- Iza return može i da se ne stavi ništa, ali u tom slučaju se ni jedna vrednost ne vraća pozivaocu.
- Funkcija koja nema povratnu vrednost deklariše se da ima povratni tip void.

Definicija i deklaracija funkcije

```
#include <stdio.h>
/* Deklaracija funkcije zbir() */
int zbir(int, int); /* int zbir(int a, int b); */

int main() {
/* Poziv funkcije */
printf("%d\n", zbir(3,5));
return 0;
}

/* Definicija funkcije */
int zbir(int a, int b) {
return a+b;
}
```