Tipovi, operatori i izrazi

Vladimir Filipović

vladaf@matf.bg.ac.rs

Imena promenljivih

-promenljive

-simboličke konstante

-**ključne reči**

auto	break	case	char
const	continue	default	do
double	else	enum	extern
float	for	goto	if
int	long	register	return
short	signed	sizeof	static
struct	switch	typedef	union
unsigned	void	volatile	while

Tipovi podataka i njihova veličina

char, int, float, double
short int sh;
long int counter;
signed, unsigned
long double

Konstante

l, L, u, U, ul, UL
f, F, l, L
0, 0x, 0X (dekadno 31, oktalno 037, heksadekadno 0x1f ili 0X1F)
0XFUL konstanta tipa *unsigned long* čija je dekadna vrednost 15
Znakovna konstanta: 'x' - 120, '0' - 48
'\0' - nulti znak
Specijalne sekvence: \n, \a, \b, \f, \r, \t, \v, \\, \?, \', \", \ooo, \xhh

```
#define VTAB  '\013'/* ASCII vertikalni tabulator */
#define BELL  '\007' /*            ASCII zvučni znak */

#define VTAB  '\xb'              /* ASCII vertikalni tabulator */
#define BELL  '\x7'              /* ASCII zvučni znak */

#define LEAP 1    /* u prestupnoj godini */
int days[31+28+LEAP+31+30+31+30+31+31+30+31+30+31];
```

Konstantan izraz:

```
#define MAXLINE 1000
char line[MAXLINE+1];
```

"hello," " world" → u vreme kompajliranja: "hello, world"

String konstanta ili doslovni string:

"Ja sam string"
"" /* prazan string */

```
/* strlen:  return length of s */
int strlen(char s[])
{
    int i;

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

'\0'

strlen(s)

<string.h>

'x' "x"

Konstanta nabrajanja:

```
enum boolean { NO, YES };

enum escapes { BELL = '\a', BACKSPACE = '\b', TAB = '\t',
               NEWLINE = '\n', VTAB = '\v', RETURN = '\r' };

enum months { JAN = 1, FEB, MAR, APR, MAY, JUN,
              JUL, AUG, SEP, OCT, NOV, DEC };      /* FEB je 2, MAR je 3 i tako dalje */
```

Deklaracije

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

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


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


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

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

Aritmetički operatori

+, -, *, /, %

x % y
ne može se primeniti na float ili double

```
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);
```

Relacioni i logički operatori

> >= < <=

== !=

&&, ||

```
for (i=0; i<lim–1 && (c=getchar()) != '\n' && c != EOF; ++i)
        s[i] = c;
```

```
i<lim–1 && (c = getchar()) != '\n' && c != EOF
```

```
(c = getchar()) != '\n'
```

```
if (!ispravno)        if (ispravno == 0)
```

Konverzije tipova

```
/* atoi:  convert s to integer */
int atoi(char s[])
{
    int i, n;

    n = 0;
    for (i = 0; s[i] >= '0' && s[i] <= '9'; ++i)
        n = 10 * n + (s[i] - '0');
    return n;
}
```

s[i] – '0'

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

tolower(c)
<ctype.h>

c >= '0' && c <= '9'

↓

isdigit()
<ctype.h>

```
int  i;
char c;

i = c;
c = i;
```

d = c >= '0' && c <='9'

Unarni operator: promena tipa (cast), (tip - ime) izraz

sqrt((double) n)

double sqrt(double) root2 = sqrt(2);

```
unsigned long int next = 1;

/* rand:  return pseudo-random integer on 0..32767 */
int rand(void)
{
    next = next * 1103515245 + 12345;
    return (unsigned int)(next/65536) % 32768;
}
/* srand:  set seed for rand() */
void srand(unsigned int seed)
{
    next = seed;
}
```

Operatori uvećanja i umanjenja

```
if (c == '\n')
    ++nl;

x = n++;

x = ++n;
```

(i+j)++

```
if (c == '\n')
    nl++;               /* ++nl  */
```

```
/* 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';
}
```

```
if (s[i] != c) {
    s[j] = s[i];
    j++;
}
```

```
/*primer kod funkcije getline*/
if (c == '\n') {
    s[i] = c;
    ++i;
}
```

```
if (c == '\n')
    s[i++] = c;
```

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

Operatori nad bitovima

&, I, ^, <<, >>, ~

n = n & 0177;
(dodeljuje nulu svim bitovima, osim 7 bitova nižeg reda promenljive n)

x = x | SET_ON
(dodeljuje jedinicu onim bitovima u x-u koji su jedinica u SET_ON)

x = x & ~077
(pretvara poslednjih 6 bitova promenljive x u nule)

x=1, y=2
x & y daje 0
x && y daje 1

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

getbits(x, 4, 3)
(vraća tri bita koja su u pozicijama 4, 3 i 2, desno poravnata)

Operatori i izrazi dodeljivanja

i = i + 2 i += 2

+ – * / % << >> & ^ |

izraz1 op= izraz2, ekvivalentno je: izraz1 = (izraz1) op (izraz2)

x *= y + 1 x = x * (y + 1) ~~x = x * y + 1~~

```
/* 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;
}
```

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

while ((c = getchar()) != EOF)
…

Uslovni izrazi

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

izr_1 ? izr_2 : izr_3

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

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

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

```
printf("You have %d item%s.\n", n, n==1 ? "" : "s");
```

Prioritet i redosled izračunavanja

```
if ((x & MASK) == 0) …
```

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

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

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

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

Precedence and Associativity of Operators in C

Precedence	Associativity	Operators			
1	L → R	`()`	`[]`	`->`	`.`
2	R → L	`!`	`~`	`++`	`--`
		(*type*)	`-` (unary)	`+` (unary)	`sizeof`
		`*` (deference)	`&` (address of)		
3	L → R	`*` (multiply)	`/`	`%`	
4	L → R	`+`	`-`		
5	L → R	`<<`	`>>`		
6	L → R	`<`	`<=`	`>`	`>=`
7	L → R	`==`	`!=`		
8	L → R	`&` (bitwise and)			
9	L → R	`^`			
10	L → R	`|`			
11	L → R	`&&`			
12	L → R	`||`			
13	R → L	`? :`			
14	R → L	`=`	`*=`	`/=`	`%=`
		`+=`	`-=`	`<<=`	`>>=`
		`&=`	`|=`	`^=`	
15	L → R	`,`			

ASCII Table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	NUL	44	2C	,	88	58	X
1	1	SOH	45	2D	-	89	59	Y
2	2	STX	46	2E	.	90	5A	Z
3	3	ETX	47	2F	/	91	5B	[
4	4	EOT	48	30	0	92	5C	\
5	5	ENQ	49	31	1	93	5D	]
6	6	ACK	50	32	2	94	5E	^
7	7	BEL	51	33	3	95	5F	
8	8	BS	52	34	4	96	60	‘
9	9	HT	53	35	5	97	61	a
10	A	LF	54	36	6	98	62	b
11	B	VT	55	37	7	99	63	c
12	C	FF	56	38	8	100	64	d
13	D	CR	57	39	9	101	65	e
14	E	SO	58	3A	:	102	66	f
15	F	SI	59	3B	;	103	67	g
16	10	DLE	60	3C	<	104	68	h
17	11	DC1	61	3D	=	105	69	i
18	12	DC2	62	3E	>	106	6A	j
19	13	DC3	63	3F	?	107	6B	k
20	14	DC4	64	40	@	108	6C	l
21	15	NAK	65	41	A	109	6D	m
22	16	SYN	66	42	B	110	6E	n
23	17	ETB	67	43	C	111	6F	o
24	18	CAN	68	44	D	112	70	p
25	19	EM	69	45	E	113	71	q
26	1A	SUB	70	46	F	114	72	r
27	1B	ESC	71	47	G	115	73	s
28	1C	FS	72	48	H	116	74	t
29	1D	GS	73	49	I	117	75	u
30	1E	RS	74	4A	J	118	76	v
31	1F	US	75	4B	K	119	77	w
32	20	SP	76	4C	L	120	78	x
33	21	!	77	4D	M	121	79	y
34	22	“	78	4E	N	122	7A	z
35	23	#	79	4F	O	123	7B	{
36	24	$	80	50	P	124	7C	|
37	25	%	81	51	Q	125	7D	}
38	26	&	82	52	R	126	7E	~
39	27	‘	83	53	S	127	7F	
40	28	(	84	54	T			
41	29	)	85	55	U			
42	2A	*	86	56	V			
43	2B	+	87	57	W			