```
#include <stdio.h>
int main ( )
{
        int x;
        scanf ("%d", &x);
        printf ("Number is %d\n", x);
        return 0;
}
```

```
if  (условие)
        оператор1;
else
        оператор2;
```

```
x = (условие) ? expr_true : expr_false;
```

```
switch (выражение) {
        case константа1:
                оператор;
                break;
        case константа2:
                оператор;
                break;
        default: оператор;
}
```

```
 for (x = 0; x < 10; x++)
        printf ("%d\n", x);
```

```
while (условие_продолжение_цикла)
        оператор;
```

```
do
        оператор;
while (условие_продолжение_цикла) ;
```

```
int a [3] = {-2, 1, 17};          int m[3][4];
a [0] = a[1] +7;                  m[0][2] = 7;
```

```
int x,                    a[3];
int * px,                 *pa = NULL;
px = &x;                  pa = a;
*px = 7;                  pa[0] = -7;
```

```
char s [ ] = "Hello";  s[0] = 'h'; // ok
char * p = "Hello";    p[0]='h';   // wrong
```

```
struct A {
        int n;
        char c;
} ;
struct A x = { 10, 'z' };          struct A * px = & x;
x.n = 11;                          px -> n = 12;
```

```
10 10L  3.14 3.14F  'z' "Hello"          константы
```

```
char, short int, int, long int, long long int
        signed и unsigned
float, double, double double
```

Оператор	Описание
()	
[]	массивы
.	поле структуры
->	поле структуры
++ --	постфиксный
++ --	префиксный
+ -	унарный
! ~	НЕ, дополнение
(*type*)	преобразование типа
*	разыменование
&	адрес
sizeof	количество байт
* / %	
+ -	
<< >>	побитовый сдвиг
< <= > >=	сравнения
== !=	равно, НЕ равно
&	побитовое AND
^	побитовое XOR (исключ)
|	побитовое OR
&&	логическое AND
||	логическое OR
?:	условный оператор
= += -= *= /= %= &= ^= |= <<= >>=	присвоение
,	разделение выражений

```
int global_var = 2;                     // 0 by default
int myfunct (int x, int y)
{
        static int n;                   // 0 by default
        int z = x + y + global_var + n++;
        return z;
}
int main ( ) {
        int k = myfunct (1, 3);         // k = 6
        k = myfunct (1, 3);             // k = 7
        return 0;
}
```

```
int main (int argc, char * argv [ ])
{
        int I;
        for ( i=0; i < agrc; i++)
                printf("argv[%d]=%s\n", i, argv[i]);
        return 0;
}
```

```
#define      N 10
#define square(x)   ((x)*(x))
```

Character Class Tests **<ctype.h>** (macro)

isalnum(c)	alphanumeric?
isalpha(c)	alphabetic?
isdigit(c)	decimal digit?
isxdigit(c)	hexadecimal digit?
ispunct(c)	printing char except space, letter, dig?
isspace(c)	space, \f, \n, \r, \t, \v?
islower(c)	lower case letter?
isupper(c)	upper case letter?
tolower(c)	convert to lower case
toupper(c)	convert to upper case

String Operations **<string.h>**

s,t are strings, cs,ct are constant strings

strlen(s)	length of s
strcpy(s,ct)	copy ct to s
strncpy(s,ct,n)	up to n chars
strcat(s,ct)	concatenate ct after s
strncat(s,ct,n)	up to n chars
strcmp(cs,ct)	compare cs to ct
strncmp(cs,ct,n)	only first n chars
strchr(cs,c)	pointer to first c in cs
strrchr(cs,c)	pointer to last c in cs
memcpy(s,ct,n)	copy n chars from ct to s
memmove(s,ct,n)	copy n chars from ct to s (may overlap)
memcmp(cs,ct,n)	compare n chars of cs with ct
memchr(cs,c,n)	pointer to first c in first n chars of cs
memset(s,c,n)	put c into first n chars of cs

Standard Utility Functions **<stdlib.h>**

rand()	pseudo-random integer [0,RAND_MAX]
srand(n)	set random seed to n
exit(status)	terminate program execution
strtod(s,endp)	convert pre_x of s to double
strtol(s,endp,b)	convert pre_x of s (base b) to long
strtoul(s,endp,b)	same, but unsigned long
malloc(size), calloc(nobj,size)	allocate storage
realloc(pts,size)	change size of object
free(ptr)	deallocate space
bsearch(key,array,n,size,cmp())	search array for key
qsort(array,n,size,cmp())	sort array ascending order

getchar()	fgetc(fp)	feof(fp)
putchar(c)	fputc(c, fp)	fgets(s,max,fp)
FILE * fp = fopen(filename, mode);		fclose(fp);

d - decimal int	c - single char	l - long
o - octal	s - string	h - half (short)
x - hex	u - unsigned	L - long double