# Simplest "Hello world" program

A **C** program consists of a main function and several program functions. The program can also access many external functions that are contained in the header file and **C** library.

The roles of the *main function* include declaring global variables, defining program functions and specifying the sources of external functions.

The *header file* normally contains frequently used utility functions such as IO library, etc.

An array is indexed by a pointer. The pointer starts at 0, rather than 1. To under each elements of this short program and try to add additional features to the program.

#include <stdio.h> // 1

int main(void){ // 2

 printf("Hello world!\n"); // 3

 return 0; // 4

}

# Condition statements

**if** (<condition>){
 <operators if condition is **True**>;
} **else** {
 <operators if condition is **False**>;
}

# Operators

|  |  |  |
| --- | --- | --- |
| Operator | Meaning of operator | Example |
| ***Simple assignment*** |
| **=** | Assign some variablea value | x = 10; |
| ***Arithmetic*** |
| **/** | division | c = a / b; |
| **+** | addition or unary plus | c = a + b; b = +10; |
| **-** | subtraction or unary minus | c = a – b; a = -11; |
| **\*** | multiplication | c = a \* b; |
| **%** | remainder after division (modulo division) | c = a % b; |
| ***Increment and decrement*** |
| **++** | Increment increases the value by 1 | a++; ++b; |
| **--** | Decrement decreases the value by 1 | b--; --a; |
| ***Extended assignment*** |
| **/=** | Same as a = a / a; | a /= 10; a = a / 10; |
| **+=** | Same as a = a + a; | a += 10; a = a + 10; |
| **-=** | Same as a = a - a; | a -= 10; a = a - 10; |
| **\*=** | Same as a = a \* a; | a \*= 10; a = a \* 10; |
| **%=** | Same as a = a % a; | a %= 10; a = a % 10; |
| ***Relational*** |
| **==** | Equal to | 7 == -1; return 0 |
| **!=** | Not equal to | 7 != -1; return 1 |
| **>** | Greater than | 10 > 2; return 1 |
| **<** | Less than | 10 < 2; return 0 |
| **>=** | Greater than or equal to | 10 >= 10; return 1 |
| **<=** | Less than or equal to | 10 <= 11; return 1 |
| ***Logical*** |
| **!** | Logical NOT | !(7 == -1); return 1 |
| **&&** | Logical AND | 7 != 1 && 3 > 1; return 1 |
| **||** | Logical OR | 7 == 5 || 4 < 1; return 0 |
| ***Bitwise*** |
| **&** | Bitwise AND | 4 & 3 as 0b00000100 & 0b00000011return 0b00000000 as 0 |
| **|** | Bitwise OR | 4 & 3 as 0b00000100 & 0b00000011return 0b00000111 as 7 |
| **^** | Bitwise exclusive OR | 5 & 3 as 0b00000101 & 0b00000011return 0b00000110 as 6 |
| **~** | Bitwise complement | ~ 69 as 0b10000101return 0b00111010 as 58 |
| **<<** | Shift left | 1 << 2 as 0b00000001return 0b00000100 as 4 |
| **>>** | Shift right | 8 >> 2 as 0b00001000return 0b00000010 as 2 |
| ***Other*** |
| **sizeof** | Unary operator which returns the size of data | sizeof(a); |
| ***Ternary*** |
| <CondExp>**?**<expTrue>**:**<expFalse>;The CondExp is evaluated first. If it is true then expTrue is returned else returned expFlase | a=10;b=8; a>b?a:b; return 10a=2;b=11; a<b?a:b; return 2 |

# Command line arguments

Command line argument is a parameter supplied to the program when it is invoked. They are mostly used when you need to control your program by outside.

Next example output all passed command line arguments. Note first argument always is full path to the invoked program.

#include <stdio.h>

int main(int argc, char \* argv[]){

    int i;

    printf("Passed arguments count is %u\n", argc);

    for( i=0; i<argc; ++i ){

     printf("Argument # %u is %s\n", i, argv[i]);

    }

    return 0;

}

Invoke the program "cla.exe" with argument firstArg 10 -4 67.4 "one and two" output is:

Total passed arguments count is 6

Argument # 0 is C:\cla.exe

Argument # 1 is firstArg

Argument # 2 is 10

Argument # 3 is -4

Argument # 4 is 67.4

Argument # 5 is one and two

Next example demonstrate control program flow by pass some command line arguments

#include <stdio.h>

int main(int argc, char \* argv[]) {

    if (argc > 1) {

        printf("Hello, %s!\n", argv[1]);

    }

    else {

        printf("Use prog.exe <name>\n");

    }

    return 0;

}

The output without arguments is "Use prog.exe <name>".

The output with argument "YourName" is "Hello, YourName!".

## Comments

Where are two types of comments: one line comment and multiline comment. No words, only example:

#include <stdio.h>

int main(void){

    int x = 10; // x is some var

    int y = 11; // y is some var two

    int res;

    /\* Next code very important

     because contains main algorithm of program

    \*/

        res = y \* x;

    /\* dead code

        res = y \* x + 3;

    \*/

    printf("Result is %u\n", res);

    return 0;

}

Note comment no effected on control or size of program. They are removed before complied.