Hello World!

Типове данни, променливи & операции

Hello World

```
#include <stdio.h>

int main()
{
	printf("Hello World!");
	printf("Hello, \nFA");

	return 0;
}
```

Директиви на препроцесора

Препроцесор

- Програма, която се извиква преди компилатора и прави предварителна обработка на написания код
- http://www.cprogramming.com/reference/preprocessor/

#include

- Заглавен (header) файл – съдържа декларации на готови функции, които могат да бъдат използвани
  - Част от библиотека или подготвени от нас
  - Реализацията на самите функции обикновено е в cорс файл
  - Могат да съдържат и много други неща – декларации на променливи, дефиниции, имплементации на функции (in-line функции)
- #include <HEADER_FN> - файлът се търси в директориите описани в
  - след VS 2013 - Project Properties→Configuration→VC++ Directories, по-ранни версии Tools→Options→Projects and Solutions→VC++ Directories
  - Project Properties→Configuration →C/C++→General→Additional Include Directories
- #include “HEADER_FN”
  - “ ” първо се търси в същата папка, в която се намира cорс файла, в който е сложен include-а, след това в папките, в които се намират другите включени заглавни файлове и накрая в зададените в настройките папки
- #include <stdio.h> - ще се използват функции (и др.) описани в заглавния файл stdio.h
  - stdio.h – standard input output
  - math.h – математически функции
  - http://en.cppreference.com/w/c/header

#define

- Дефиниране на макрос за константа
  - #define *token* [*value*]
    - #define FA
    - #define PI 3.14
- Дефиниране на параметричен макрос
  - #define *token*(<arg> [, <arg>s ...]) *statement*
    - #define MAX(a, b) ((a) > (b) ? (a) : (b))

#if, #ifdef, #ifndef, #endif

```
#include <stdio.h>

#define FA

int main()
{
	printf("Hello World!");
#ifdef FA
	printf("Hello, \nFA");
#endif
	return 0;
}
```

Функции

Funktion

- Група от операции, които заедно решават определена задача
- Базова модулна единица в езика за програмиране C
- Декларация на функция – прототип на функция
  - return_type function_name(списък с параметри разделени със запетая);
- Дефиниция на функция
  - return_type function_name(списък с параметри разделени със запетая) { тяло на функция }
  - Тялото на функцията са операциите, които се изпълняват при извикването на функцията

Главна функция **main**

- Главната функция на всяка (изпълнима) програма
  - Изпълнението на една програма винаги започва с **main** функцията
  - Всяка програма може да има само една **main**
- int main (void) / int main()
- int main (int argc, char *argv[])
- Връщан резултат от функция main
  - 0 – програмата е изпълнена успешно
  - ≠ 0 – при изпълнението на програмата за възникнали грешки
    - Върнатата стойност е код на грешката

Функция за извеждане – printf

- int printf(const char *format, ...);
  - Служи за форматиран изход
- Най-проста употреба – извеждане на константен текст
  - printf(„Text to print“);
  - „“ – всичко между кавичките е константен текст и се извежда точно така, както е написано
  - Изключения
    - Escape sequences (управляващи последователности)
    - Форматиращи символи / спецификатори

Escape sequences

Escape sequence	Description	Representation
\'	single quote	byte 0x27 in ASCII encoding
\"	double quote	byte 0x22 in ASCII encoding
\?	question mark	byte 0x3f in ASCII encoding
\\	backslash	byte 0x5c in ASCII encoding
\a	audible bell	byte 0x07 in ASCII encoding
\b	backspace	byte 0x08 in ASCII encoding
\f	form feed - new page	byte 0x0c in ASCII encoding
\n	line feed - new line	byte 0x0a in ASCII encoding
\r	carriage return	byte 0x0d in ASCII encoding
\t	horizontal tab	byte 0x09 in ASCII encoding
\v	vertical tab	byte 0x0b in ASCII encoding
\nnn	arbitrary octal value	byte nnn
\xnn	arbitrary hexadecimal value	byte nn
\unnnn	universal character name (arbitrary Unicode value); may result in several characters	code point U+nnnn
\Unnnnnnnn	universal character name (arbitrary Unicode value); may result in several characters	code point U+nnnnnnnn

Типове данни в C

Типове данни в C

Целочислени			С плаваща запетая			
Тип	Памет	Форматиращ символ	Тип	Памет	Форматиращ символ	Точност
char	1B	%c, %d	float	4B	%f	6 позиции
short	2B	%d	double	8B	%lf	15 позиции
int	2B/4B	%d				

- char – понякога се описва като символен тип (за съхранение на символи), но всъщност съдържа цяло число, което може да се интерпретира като код на символ от ASCII таблица
- Оператор **sizeof** (тип) – връща броя байтове, използвани от посочения тип – например sizeof(int) ще върне 4 (за 32 битова архитектура)
- Модификатори **signed** (подразбиращ се)/ **unsigned** – позволява съхранението само на положителни числа
  - char – -128 до 127, unsigned char – 0 до 255
- Модификатор **long** - +2B памет
  - long int (често само long) – 4B, long double – 10B

Забележки към числа с плаваща запетая

- Представят се с помощта на три компонента
  - Знак
  - Мантиса
  - Експонента
- $m*b^e$
- Мантиса – съдържа стойността
- B – база – двоична система – 2
- e – Степен
- $1 = 1.000*2^0$, $2 = 1.000*2^1$, $0.375 = 1.100*2^{-2}$
  - $0.1_2 = 1 * 2^{-1} = 0.5_{10}$

```
#define EPSILON 1.0e-7

 #define flt_equals(a, b) (fabs((a)-(b)) < EPSILON)
```

http://www.cprogramming.com/tutorial/floating_point/understanding_floating_point_representation.html

Грешки от закръгление с реални числа

```
#include <stdio.h>
int main(void)
{
   float a, b;
   a = 2.0e6f + 1.0;
   b = a - 2.0e6f;
   printf ("%e\n", b);

   a = 2.0e7f + 1.0;
   b = a - 2.0e7f;
   printf ("%e\n", b);
   return 0;
}
```

Резултати:

1.000000e+000
0.000000e+000

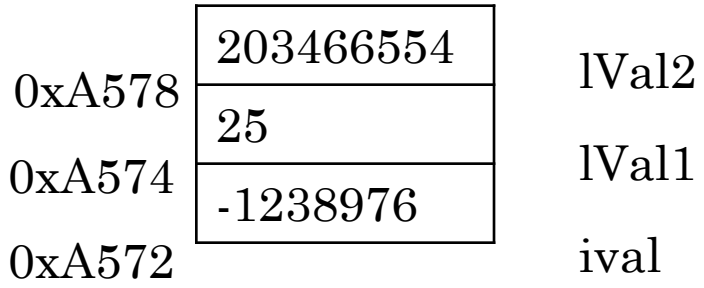
Променливи

Декларация & инициализация

- Декларация
  - тип_данни име_променлива;
  - **тип_данни** е комбинация от тип и модификатор
  - **име_променлива** – може да съдържа букви, цифри и _ (долна черта)
    - Не може да съдържа други символи освен _
    - Не може да започва с цифра
    - Не може да съвпада с ключова дума от езика
    - C е case-sensitive (прави разлика между малки и големи букви
      - name ≠ Name ≠ NAME
- Инициализация
  - Задаване на първоначална стойност на променлива
  - Може да стане и при декларирането на променливата

Примери

```
short ival;
long lVal1 = 25, lVal2 = 203466554;
long double _mDeg;
const float PI = 3.14f;
```



- Употреба
  - Където трябва да се използва стойността на променливата се записва името ѝ
    - _mDeg = lVal1 * PI;
  - С помощта на операция & може да се получи адреса на променливата
    - scanf(“%d”, &ival);
- Необходими променливи при решаване на задача
  - входни данни
  - изходни данни
  - съхраняване на междинни стойности

C Keywords

auto break case char const continue default do double else enum extern	float for goto if inline (since C99) int long register restrict (since C99) return short	signed sizeof static struct switch typedef union unsigned void volatile while	_Alignas (since C11) _Alignof (since C11) _Atomic (since C11) _Bool (since C99) _Complex (since C99) _Generic (since C11) _Imaginary (since C99) _Noreturn (since C11) _Static_assert (since C11) _Thread_local (since C11)

http://en.cppreference.com/w/c/keyword

Форматиране на вход/изход

- Форматиране на стойност
- % [[флаг]] [[ширина]][[.точност]] [[интерпретация]] тип

флаг		Подравняване отдясно
	-	Подравняване отляво
	+	Отпечатване на стойност със знак + или –
ширина	n	n позиции
.точност	.n	n позиции
интерпретация на аргумента	h	short int
	l	long int или double
	L	long double

тип	%d	Цяло число в десетична бройна система
	%i	
	%o	Цяло число в осмична бройна система
	%u	Цяло число без знак
	%x	Шестнадесетично число без знак
	%X	
	%f	Число с плаваща запетая в десетична бройна система
	%e	Число с плаваща запетая в експоненциален запис (scientific format)

Вход/Изход

int printf(const char *format, ...);

- Връща броя на отпечатаните символи
- Извеждане на константен текст
  - printf(“Enter speed in MpH:”);
- Извеждане на стойност
  - printf(“The result is %f”, kmh);
- Извеждане на няколко стойности
  - printf(“The sum of %.5d entered values is %10.2lf.\n”);
  - The sum of 00015 entered values is ____215.20.

int scanf (const char * format, ...);

- Връща броя на успешно прочетените стойности (контрол за вярно въвеждане)
- Прочитане на една стойност
  - scanf(„%d“, &mph);
- Прочитане на няколко стойности
  - scanf(“%f%f%f”, &a, &b, &c)
- Прочитане на форматиран вход
  - scanf(“%d-%d-%d”, &day, &month, &year);

Преобразуване на мили/час до километри/час

```
/* Converts miles/hour
to kilometers/hour*/
#include <stdio.h>
// Makro definition for the conversion coefficient
#define MILES_INTO_KILOMETERS 1.60934

int main ()
{
   float velocity_mph, velocity_kmph;
   printf („Enter speed in Mph: ");
   scanf ("%f", &velocity_mph);
   velocity_kmph=MILES_INTO_KILOMETERS*velocity_mph;
   printf („The speed in Kmh = %.3f [Km/S]\n",
              velocity_kmph);
   return 0;
}
```

Операции и приоритет

Precedence	Operator	Description	Associativity
1	++ -- () [] . -> (*type*){*list*}	Suffix/postfix increment and decrement Function call Array subscripting Structure and union member access Structure and union member access through pointer Compound literal(C99)	Left-to-right
2	++ -- + - ! ~ (*type*) * & sizeof _Alignof	Prefix increment and decrement Unary plus and minus Logical NOT and bitwise NOT Type cast Indirection (dereference) Address-of Size-of Alignment requirement(C11)	Right-to-left
3	* / %	Multiplication, division, and remainder	Left-to-right
4	+ -	Addition and subtraction	
5	<< >>	Bitwise left shift and right shift	
6	< <= > >=	For relational operators < and ≤ respectively For relational operators > and ≥ respectively	
7	== !=	For relational = and ≠ respectively	
8	&	Bitwise AND	
9	^	Bitwise XOR (exclusive or)	
10	|	Bitwise OR (inclusive or)	
11	&&	Logical AND	
12	||	Logical OR	
13[note 1]	?:	Ternary conditional[note 2]	Right-to-Left
14	= += -= *= /= %= <<= >>= &= ^= |=	Simple assignment Assignment by sum and difference Assignment by product, quotient, and remainder Assignment by bitwise left shift and right shift Assignment by bitwise AND, XOR, and OR	
15	,	Comma	Left-to-right

Често използвани операции

Common operators

assignment	increment decrement	arithmetic	logical	comparison	member access	other

`a = b`
`a += b`
`a -= b`
`a *= b`
`a /= b`
`a %= b`
`a &= b`
`a |= b`
`a ^= b`
`a <<= b`
`a >>= b`
	`++a`
	`--a`
	`a++`
	`a--`
		`+a`
		`-a`
		`a + b`
		`a - b`
		`a * b`
		`a / b`
		`a % b`
		`~a`
		`a & b`
		`a | b`
		`a ^ b`
		`a << b`
		`a >> b`
			`!a`
			`a && b`
			`a || b`
				`a == b`
				`a != b`
				`a < b`
				`a > b`
				`a <= b`
				`a >= b`
					`a[b]`
					`*a`
					`&a`
					`a->b`
					`a.b`
						`a(...)`
						`a, b`
						`(type) a`
						`? :`
						`sizeof`
						`_Alignof` (since C11)

Бележки относно операциите

- С помощта на операциите се образуват изрази (например a+b)
- За да се изчисли израза, трябва a и b да имат един и същи тип на данните, примерно и двете да са от тип int – int a, b;
- Ако не са → преобразуване на типа
  - Неявно преобразуване (implicit cast)
    - Извършва се автоматично от компилатора – от по-малкия към по-големия тип данни
    - int a; long b;
      - a+b → **стойността** на a неявно се преобразува към long
    - float a; int b;
      - a+b → **стойността** на b неявно се преобразува към float
  - Явно преобразуване (explicit cast)
    - (тип_данни)променлива
    - (float)a / b

Присвояване (Assignment)

- a = b, на a се присвоява стойността на b
- D = b*b – 4*a*c
- Първо се изчислява израза отдясно на =, след което резултата се присвоява на променливата отляво
  - Отляво може да стои само lvalue (https://msdn.microsoft.com/en-us/library/f90831hc.aspx) – т.е. отляво не може да стои друг израз (rvalue)
- Съкратени начини на запис
  - a += b означава a = a + b, a *= b → a = a * b, и т.н.

```
a = b
a += b
a -= b
a *= b
a /= b
a %= b
a &= b
a |= b
a ^= b
a <<= b
a >>= b
```

операции за инкрементиране / декрементиране

- Increment – увеличаване на стойността с 1
- Decrement – намаляване на стойността с 1
- Префиксен запис – ++var или --var – първо се променя стойността на var и след това стойността се използва в израза
- Постфиксен запис – var++ или var-- - първо се използва **оригиналната** стойност на var в израза и след това стойността се променя
- Променлива, чиято стойност се инкрементира / декрементира не бива да участва отново в същия израз → получава се неизвестен резултат
  - x = a++ + a
  - Сравнение Visual Studio vs https://www.tutorialspoint.com/compile_c_online.php

```
int x, y;
x = 5;
y = 5;
printf ("++x = %d\n", ++x);
printf ("y++ = %d\n", y++);
printf ("x = %d\n", x);
printf ("y = %d\n", y);
```

Резултати:

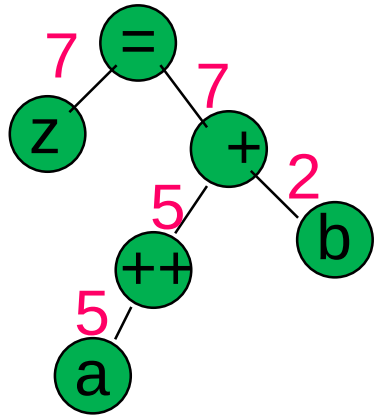
```
++x = 6
y++ = 5
x = 6
y = 6
```

a = 5;
b = 2;
z = a++ + b;

Результат:

z = 5 + 2 = 7

a = 6

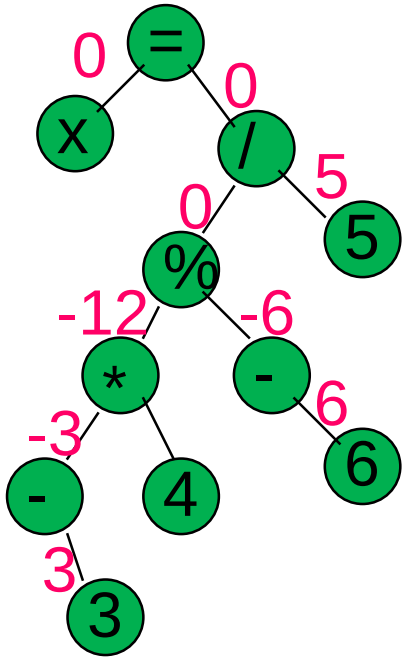


x=-3 * 4 % -6 / 5;

Результат:

x=(-3) * 4 % (-6) / 5 = (((-3) * 4) % (-6)) / 5 =

((-12) % (-6)) / 5 = 0 / 5 = 0



Логика в С

- В С не съществуват понятия вярно или грешно
- Ако една стойност е != 0 се приема за вярно
- Само ако стойността е == 0 се приема за грешно
- Примери
  - 7 – вярно, -20 – вярно, 0 - ?
  - a = 5; a – вярно, a + 2 – вярно, a – 5 - ?
  - a = 5, b = -3; a + b – вярно

- Резултатът от логическите операции и операциите за сравнение е винаги и само 0 или 1

Логически операции и операции за сравнение

Логически

```
  !a
a && b
a || b
```

- Таблици на истинност

!	a	Резултат
	≠0	0
	0	1

||	a	b	Резултат
	0	0	0
	≠0	0	1
	0	≠0	1
	≠0	≠0	1

&&	a	b	Резултат
	0	0	0
	≠0	0	0
	0	≠0	0
	≠0	≠0	1

Сравнени

```
a == b
a != b
 a < b
 a > b
a <= b
a >= b
```

- a == b – проверява дали стойността на a е равна на стойността на b
- a != b – проверява дали стойността на a е различна от тази на b

Операция ? :

- Операция с три операнда
- (условие) ? оп_ако_усл_е_вярно : оп_ако_усл_е_грешно
- Определяне на по-голяма стойност
  - a > b ? a : b