CODE ACADEMY

ОБУЧЕНИЕ ПО ПРОГРАМИРАНЕ

Лекция 2 Структура на една С програма.

CODE ACADEMY

Да си припомним

Какво научихме предния път?

CODE ACADEMY

Първата С програма

hello.c

```c
#include<stdio.h>

int main()
{
    printf("Hello! \n");

    return 0;
}
```

Кодът на програмата се записва като текст в текстов файл и се запазва с име с разширение **.c** .

CODE ACADEMY

Компилиране на С програма

> gcc -o hello hello.c

gcc *използва* ***-o опцията*** *за да промените името на обектния файл*

В противен случай името на компилираната програма е ***a.out*** *или* ***a.exe***

gcc -o program hello.c

./program

gcc -o program.exe hello.c

program

Структура на С програмата

header

Блок -тяло на функцията

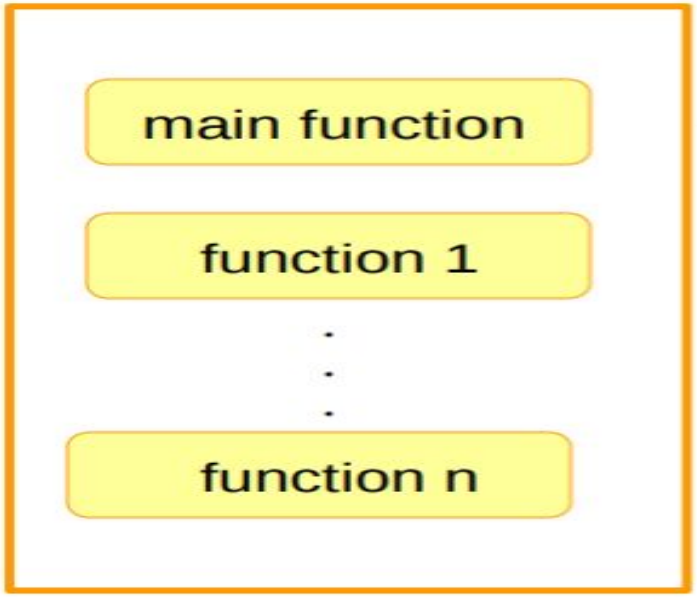
```
void print(int n)
{
    int tens = n /10;
    int units = n % 10;
    printf("number=%d \t  tens=%d \t units=%d",n);
}
```

header

Блок -тяло на функцията

```
int main()
{
    int number=56;
    print(number);
   return 0;
}
```

Всяка С програма трябва да има main() функция

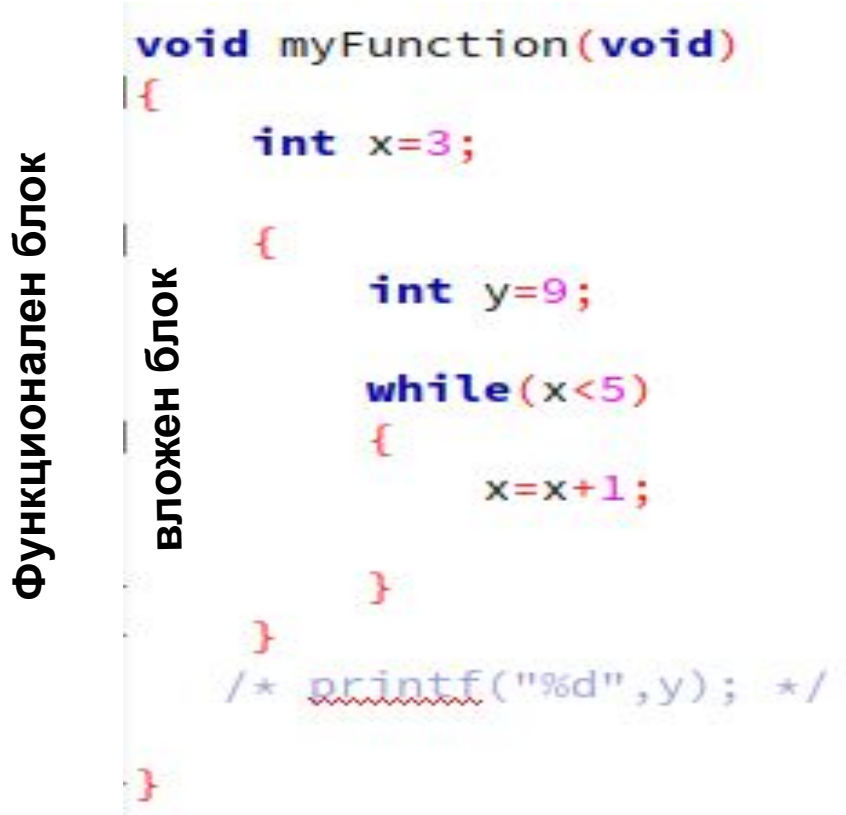


Всяка функция се състои от заглавие – header и основен блок, наречен още тяло на функцията

CODE ACADEMY

Блок в С програмата

- *Блок - това е последователност от изпълними оператори, заградена в { }.*
- *Блокът се състои от един или повече оператори, завършващи с ;*
- *Позволено е влагане на блокове.*



CODE ACADEMY

SCOPE в С програмата

- ***Обхват, видимост*** *на променливите и функциите, т.е.*
  scope – *се използва, за да се ограничи възможността на достъп до данни, които в момента манипулираме.*
- ***Обхватът*** *описва нивото, на което определени данни или функция са видими.*

В езика С има ***два вида scope***

- ***глобално*** *– от всяка точка на програмата*
- ***локално*** *– само в рамките на блока { }, в който е декларирано*

```c
int main()
{
    int i = 6;
    {
        printf("%d\n", i);
        int i = 5;
        printf("%d\n", i);
    }
    printf("%d\n", i);

    return 0;
}
```

CODE ACADEMY

Функцията main

- *Това е **входната точка** на програмата.*
- *Всяка програма трябва да има **точно 1 функция main***
- *Всяка функция main трябва **да връща число int***
  - ***число 0** -> означава програмата се е изпълнила **без грешка***
  - *число **различно от 0** -> код за **грешка***
- *Има **резервирани кодове за грешка** от операционната система.*
- ***Тялото на main** функцията се огражда с **{ }** както всички останали функции.*

```
int main()              // header
{                       // beginning of basic block
                        // ...
    return 0;           // program ending successfully
}                       // end of basic block
```

CODE ACADEMY

Параметри на команден ред

При изпълнение на програмата от команден ред можем да предадем.

- За целта дефинираме main с два аргумента

int argc →броят command-line параметри

char **argv или **char *argv[ ]** → масив от указатели към стрингове, **argv[0]** е винаги името на праграмата.

```c
#include<stdio.h>

int main(int argc,char* argv[])
{
    int counter;
    printf("Program Name Is: %s",argv[0]);
    if(argc==1)
        printf("\nNo Extra Command Line Argument Passed Other Than Program Name");
    if(argc>=2)
    {
        printf("\nNumber Of Arguments Passed: %d",argc);
        printf("\n----Following Are The Command Line Arguments Passed----");
        for(counter=0;counter<argc;counter++)
            printf("\nargv[%d]: %s",counter,argv[counter]);
    }
    return 0;
}
```

```
CodeACADEMY> gcc mainArgv.c
CodeACADEMY> ./a.out Hello
```

```
Program Name Is: ./a.out
Number Of Arguments Passed: 2
----Following Are The Command Line Arguments Passed----
argv[0]: ./a.out
argv[1]: Hello(base)
```

CODE ACADEMY

Коментари в С програмата

- *Правят програмата лесна за четене и модифициране*
- *Игнорират се от компилатора на С*
- ***Коментиране на ред*** *→ всичко след // се игнорира*

  пример: **double sum;** *// променлива за сума*

- *Коментиране на блок →* **/*** ***.....*** ***/**

```
/*
Program:    ch02First
Purpose:    Display Go Tigers!
Author:     Ima Programmer
Date:       mm/dd/yy
*/
```

Директиви на ПРЕПРОЦЕСОРА

Изпълняват се преди компилатора.
В някои дистрибуции препроцесорът е отделна програма.
Препроцесорът има директиви, които имат префикс #

`#include <stdio.h>` – съдържанието на посочения файл **stdio.h** ще бъде вмъкнато там, където се срещне **#.** Такъв файл е известен като **header файл**. **< >**-ъгловите скоби означават, че този header файл е стандартен за компилатора.

`#define MAX_LENGTH 1024` – навсякъде, където срещне макроса MAX_LENGTH ще го замести с 1024

CODE ACADEMY

Идентификатори

- ***Идентификаторът*** *е дума, използвана за име или за референция към данни, които се манипулират от програмата.*

- ***Резервирана*** *е* ***дума*** *с определено значение за компилатора на C, като* ***int, return, string, include, ...***

- *Конвенция за имената*
  - *променливите с малки букви - lower case* ***number, name_2***
  - *константите с главни букви - upper case* ***TOTAL, MINIMUM***
  - *променливите с повече от една дума* ***bookPrice***

CODE ACADEMY

Валидни имена за идентификатори

- започват с буква или “ _” (underscore)

- съдържат латински букви (A-Za-z), цифри или “ _” (underscore)

- case sensitive → total различно от Total

- не са резервирана дума

CODE ACADEMY

Променливи

- *Декларират се* ***преди да се използват***

  type name;

  int x; char c;

- ***Името е:***
  - ***валиден идентификатор***
  - ***описателно и съответства на употребата***
  - ***абревиатури, общоприети за разбиране*** → **amt=amount**

При даване на името на променливата трябва да се съобразяваме с правилата, които са приети в компанията

- ***Стойността се съхранява в паметта***

CODE ACADEMY

Деклариране на променливи

optional_modifier data_type name_1, name_2, …, name_n;

- *Използва се, за да прави разлика между **signed** и **unsigned** цели числа. По премълчаване е signed.*
- *Използва се, за да определи размера – **short, long***
- *Определя константите с ключавата дума **const***
- ***Определя типа на стойността***
- ***Позволява на компилатора да знае кои операции са валидни***
- ***Компилаторът знае как да представи стойността в паметта***
- ***Имена на идентификаторите в програмата***

CODE ACADEMY

Инициализиране на променлива

Инициализиране на променливите при деклариране

```
int age=22;

double rate=0.75;

char vowel='a';

int count=0, total=0;
```

CODE ACADEMY

Символи

*Латински букви – **A-Z a-z** и цифри – **0-9***

*Разделители – **интервал, табулация, нов ред***

Специални символи - **; , . : ( ) [ ] { } + - * / = | & ! < > # $ ^ % ~ _ ‘ "**

Type Name	Memory Used	Sample Size Range
char	1 byte	All ASCII characters

ASCII = American Standard Code for Information Interchange

CODE ACADEMY

Въпроси?

CODE ACADEMY

Типове данни

- За да се обработят данните, трябва да бъдат класифицирани в определени типове.

- *Операциите са приложими към определен тип данни.*

- ***Тип данни - набор от стойности и набор от операции, които могат да бъдат приложени към тези стойности.***

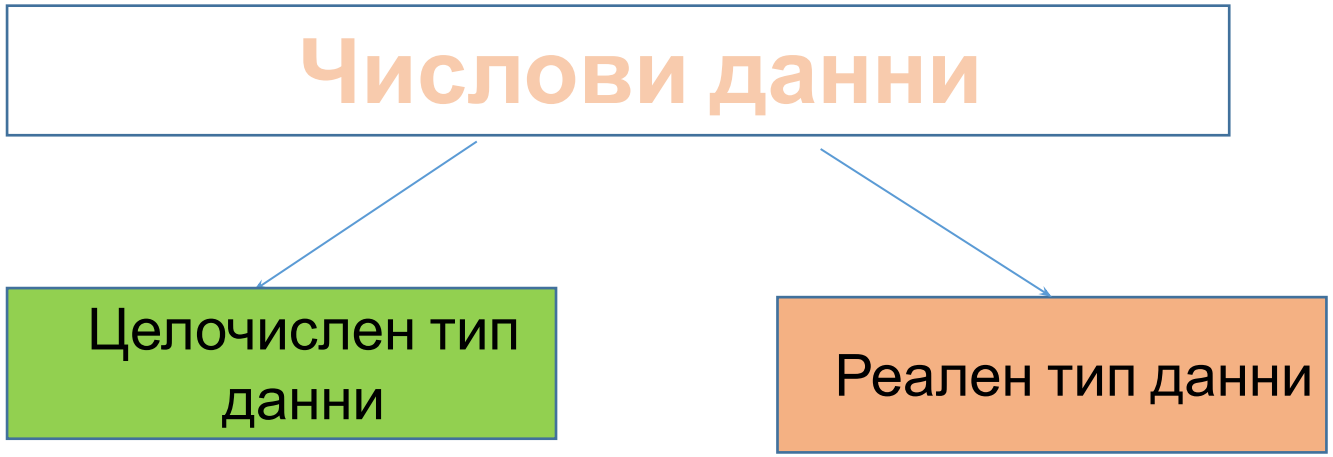
пример:

***М**ножеството от всички цели (**int**) числа съставлява множество на стойности и множеството от всички реални числа (числа, съдържащи десетична запетая).*

Тези два набора от числа обаче не представляват тип данни, докато не се включи набор от операции - математически операции и операции за сравнение.

CODE ACADEMY

Числов тип данни



+ - * \ %
= == != <= >=
sizeof( )
битови операции

**+ - * **
= == != <= >=
sizeof( )

Целочислен тип данни

Целочисле
н тип данни

- bool
- char
- short int
- int
- long int
- unsigned char
- unsigned short int
- unsigned int
- unsigned long int

Целочислена стойност
се състои само от цифри
и може да бъде предшествана
от знак плюс (+) или минус (-).

цяло число - число без десетична запетая

- числото 0
- всяко положително число
- всяко отрицателно число

0, -10, 1000, -26351, 5, +25, 253, +36

CODE ACADEMY

SIZEOF

- *Компилаторът различава в своите вътрешни лимити най-голямата и най-малката отрицателна целочислена стойност, която може да се съхрани за всеки тип данни.*
- *Операторът* ***sizeof*** *връща* ***размера на даден тип данни в байтове***

```c
#include<stdio.h>
#include<stdbool.h>
int main()
{
    printf("bool =%d bytes\n", sizeof(bool));
    printf("char =%d bytes\n", sizeof(char));
    printf("int =%d bytes\n", sizeof(int));
    printf("short int =%d bytes\n", sizeof(short));
    printf("long int =%d bytes\n", sizeof(long));
    printf("long long int =%d bytes\n", sizeof(long long));

    printf("unsigned char =%d bytes\n", sizeof(unsigned char));
    printf("unsigned int =%d bytes\n", sizeof(unsigned int));
    return 0;
}
```

```
bool =1 bytes
char =1 bytes
int =4 bytes
short int =2 bytes
long int =8 bytes
long long int =8 bytes
unsigned char =1 bytes
unsigned int =4 bytes
```

CODE ACADEMY

Символи

- ***Символен тип данни*** *–* ***char*** *се използва за съхранение на единични символи, в това число латински букви (uppercase и lowercase), цифри от 0 до 9 и специални символи.*
  Стойността на коя да е единична буква, цифра или специален символ се загражда с единични кавички, като напр. 'A', '$, 'b', '7', '!'.

- *Символните стойности обикновено се съхраняват в ASCII код.*

Letter	ASCII Code	Letter	ASCII Code
A	01000001	N	01001111
B	01000010	O	01001110
C	01000011	P	01010000
D	01000100	Q	01010001
E	01000101	R	01010010
F	01000110	S	01010011
G	01000111	T	01010100
H	01001000	U	01010101
I	01001001	V	01010110
J	01001010	W	01010111
K	01001011	X	01011000
L	01001100	Y	01011001
M	01001101	Z	01011010

CODE ACADEMY

Escape последователност

- ***Escape-символ*** *е обратната наклонена черта (\) и има специално значение в С.*

Escape Sequence	Character Represented	Meaning	ASCII Code
\n	Newline	Move to a new line	00001010
\t	Horizontal tab	Move to the next horizontal tab setting	00001001
\v	Vertical tab	Move to the next vertical tab setting	00001011
\b	Backspace	Move back one space	00001000
\r	Carriage return	Move the cursor to the start of the current line; used for overprinting	00001101
\f	Form feed	Issue a form feed	00001100
\a	Alert	Issue an alert (usually a bell sound)	00000111
\\	Backslash	Insert a backslash character (used to place an actual backslash character in a string)	01011100
\?	Question mark	Insert a question mark character	00111111
\'	Single quotation	Insert a single-quote character (used to place an inner single quote within a set of outer single quotes)	00100111
\"	Double quotation	Insert a double-quote character (used to place an inner double quote within a set of outer double quotes)	00100010
nnn	Octal number	Consider the number *nnn* (*n* is a digit) an octal number	Dependent on *nnn*
\x*hhhh*	Hexadecimal number	Consider the number *hhhh* (*h* is a digit) a hexadecimal number	Dependent on *hhh*
\0	Null character	Insert the null character, which is defined as having the value 0	00000000

CODE ACADEMY

Логически тип данни bool

Приемат стойности ***true и false***

За да се идползва ***bool,*** *трябва да се включи библиотеката* ***stdbool.h***

#include<stdbool.h>

иначе за логически променливи могат да се използват с ***тип int със стойности 0 и 1***

Логическите действия са

- отрицание !

- логическо умножение И &&

-логическо събиране ИЛИ ||

Реални типове данни - Floating-Point

- Числото 0 и всяко отрицателно и положително **число с десетична точка**.
  +10.625 5. -6.2 3251.92 0.0 0.33 -6.67 +2.

- *Има три типа поддържащи floating-point променливи **float** , **double** и **long double.***

- ***double -** double-precision number , **float** - single-precision number (2 пъти по точно)*

- ***“precision” –** количеството значещи цифри в числото.*

```c
int main()
{
    printf("%d\n", sizeof(long double));
    printf("%d\n", sizeof(double));
    printf("%d\n", sizeof(float));

    return 0;
}
```

```
16
8
4

Process returned 0 (0x0)
Press ENTER to continue.
```

CODE ACADEMY

Оператор за присвояване

Присвоява стойност на променлива

Общ вид: **identifier = expression;**

‘=‘ присвояваме expression на identifier

Примери:

- **1) double sum = 0;**

2) int x;
x=5;
3) char ch;
ch = ‘a’;

- **a = (b = 15);**
- **a = b = 15;**

CODE ACADEMY

Преобразуване на типовете

- **Неявно преобразуване**

  bool -> char -> short int -> int -> unsigned int ->

  long -> unsigned -> long long -> float -> double -> long double

  Пример

```
int x = 5;
float y = 12.5;
y = x ;
```

- **Явно преобразуване**

```
int x = 5;
float y = 12.5;
x = (int) y ;
```

CODE ACADEMY

Упражнение **swap**

Напишете програма, която разменя стойностите на две целочислени променливи

Първи начин – с формула

```c
#include<stdio.h>

int main()
{
    int a=4;
    int b=23;
    printf("a=%d b=%d\n",a,b);
    a=a+b;
    b=a-b;
    a=a-b;
    printf("a=%d b=%d\n",a,b);
return 0;
}
```

```
a=4 b=23
a=23 b=4
```

Втори начин – с помощна променлива

```c
#include<stdio.h>

int main()
{
    int a=4;
    int b=23;
    printf("a= %d b=%d\n",a,b);
    int c=a;
    a=b;
    b=c;
    printf("a= %d b=%d\n",a,b);
return 0;
}
```

```
a=4 b=23
a=23 b=4
```

CODE ACADEMY

Указатели

Pointer (указател) - променлива, която съдържа адрес, който сочи към адрес реално се намира друга променлива, в която са записани данните.

```
#include <stdio.h>
int main(){
    int a = 42;
    int *p_a = NULL;
    p_a = &a;
    printf("p_a = %d\n", *p_a);
    return 0;
}
```

Входни и изходни операции в С

Вход - Input *означава, че някакви данни влизат в програмата.*

Изход – Output *означава да се изведат някакви данни на екран, файл, принтер..*

#include <stdio.h>

scanf( ) →библиотечна *функция за вземане на входни данни от user*

printf( ) →библиотечна *функция за извеждане на дисплея*

```c
#include <stdio.h>
int main()
{
float a;
printf("Enter value: ");
scanf("%f",&a);
printf("Value=%f",a);return 0;
}
```

```
Enter value: 3
Value=3.000000
Process returned 0 (0x0)
Press ENTER to continue.
```

CODE ACADEMY

Изход в езика С

- ***Изход – Output*** *означава да се изведат някакви данни на екран, файл, принтер.*

```
#include <stdio.h>

int printf(const char *format-string, argument-list);
```

списък форматиращи параметри

списък с променливи или изрази, разделени със запетая

Определя как ще бъде отпечатани изразите, като на мястото в стринга се поставят заместители за всеки израз.
Заместителите се разпознават по това, че пред тях стои символът %

printf връща цяло число = броя на отпечатаните символи

CODE ACADEMY

printf – заместители за цели числа

%d или **%i** за цели числа

%l за long

```
printf("%d", age);
printf("%l", big_num);
```

%o за цели числа в осмична бройна система

```
printf("%o", a);
```

%o за цели числа в 16-тична бройна система

```
printf("%x", b);
```

CODE ACADEMY

printf – заместители за реални числа

“%f, %e, %g за **float** реални числа

%f отпечатва стойността по стандартен начин

%e отпечатва стойността scientific notation

%g printf избира между **“%f** и **%e** и автоматично премахва водещите нули

%lf за double

CODE ACADEMY

Извеждане стойност на променлива

В изхода могат да се включат и литерали

Използваме %% за да отпечатаме символа %

Използваме %n за да преминем на нов ред

```
printf("x=%d\n",   x );

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

printf("Rate is %d%%\n",  rate*100 );
```

CODE ACADEMY

Форматиращ стринг на printf

https://www.cprogramming.com/tutorial/printf-format-strings.html

https://www.codingunit.com/printf-format-specifiers-format-conversions-and-formatted-output

https://www.l3harrisgeospatial.com/docs/format_codes_cprintf.html

Вход в езика С

- ***Вход - Input*** *означава, че някакви данни влизат в програмата.*

```
#include <stdio.h>

int scanf(const char *format-string, argument-list);
```

списък форматиращи параметри

списък с аргументи напр. & x, & y

Вход от клавиатурата - scanf

- *Общ вид* **scanf(format-string, address-list);**
- *Пример:* **scanf(“%d”, &age);**
- Форматиращият стринг съдържа placeholders -заместители, по един за адрес, използвани за конвертиране на входа.
- ***%d*** *казва, че програмата очаква да се въведе ASCII кодирано цяло число и че scanf трябва да конвертира стринга от ASCII символи в двоично цяло число вътре в паметта.*
- *Списъкът от адреси – списък от адреси в паметта, където да се съхранят входните стойности*

CODE ACADEMY

Адресите в scanf

- *Списъкът от адреси трябва да съдържа само адреси*
- *scanf поставя въведената стойност в паметта на посочения адрес*
- *Променливата (age) не е адрес, тя е съдържание от паметта, което има адрес*
- *Адресът на променливата се взема с помощта на адресен оператор* ***& (адрес на)*** *и този адресен оператор се поставя вътре в scanf*
- *За всеки елемент на адресния списък, във форматиращия стринг има заместител - placeholder*

https://en.wikipedia.org/wiki/Scanf_format_string

Върната стойност от scanf

- *Успешното завършване на scanf връща броя на прочетените входни стойности или EOF ако стигне край на входен файл, който чете*

Пример

```
int dataCount;
dataCount = scanf(“%d %d”, &height, & weight );
```

Ако scanf е успешно, стойността на dataCount ще е 2.

ПРОВЕРЕТЕ!

CODE ACADEMY

Форматиращи заместители в scanf

Когато четем данни използваме следните placeholders

%d	за цели числа
%i	за цели числа
%f	за float
%lf	за double

Тук не се определя ширина и други специални опции както в printf

CODE ACADEMY

Пример със scanf

```c
int main() {
    // declare variables
    int x;
    int y;
    int sum;

    // read values for x and y from standard input
    printf("Enter value for x: ");
    scanf("%d", &x);

    printf("Enter value for y: ");
    scanf("%d", &y);

    sum = x + y;

    // print
    printf("x = %d\n", x);
    printf("y = %d\n", y);
    printf("x + y = %d\n", sum);
    printf("%d + %d = %d\n", x, y, sum);
    printf("%d - %d = %d\n", x, y, (x - y));
    printf("%d * %d = %d\n", x, y, (x * y));
    return 0;

}
```

CODE ACADEMY

Оператори в езика С

- **Операторът е** символ, който казва на компилатора да изпълни конкретни математически или логически действия или манипулации.

- В езика С има *следните* ***типове вградени*** **оператори**:

  - *Аритметични оператори*
  - Релационни оператори
  - Логически оператори
  - Побитови оператори
  - *Оператори на присвояване*
  - Оператори на увеличение и намаляване
  - Условни оператори
  - Разнообразни други оператори

CODE ACADEMY

Аритметични оператори

- **Събиране** **+** sum = num1 + num2;
- **Изваждане** **-** age = 2007 – my_birth_year;
- **Умножение** ***** area = side1 * side2;
- **Деление** **/** avg = total / number;
- **Модули** **%** lastdigit = num % 10;
  - Модулите връщат остатъка от делението на две цели числа
  - Например, 5%2 връща стойност 1
- Бинарни и Унарни оператори
  - Всички от горните оператори са бинарни (защо?)
  - **-** (минус) е унарен оператор, a = -3 * -4

CODE ACADEMY

Целочислено и реално деление

●Когато **делим две цели числа, получаваме цяло число**.

● Резултатът от целочислено деление се реже – truncate, не се закръгля – round

●Example:

- **int A=5/3** → A ще има стойност 1
- **int B=3/6;** → B ще има стойност 0

●За да имаме реални стойности:

- **double A=5.0/3;** → A ще има стойност 1.666
- **double B=3.0/6.0;** → B ще има стойност 0.5

CODE ACADEMY

Приоритет на аритметичните операции

- В аритметичен израз действията започват да се изпълняват отляво надясно, като се спазват следните приоритети:

1) **Скоби ( )** → **първо най-вътрешните**

2) **Унарни оператори + - (type)** → **първо най-отдясно**

3) **Бинарни оператори * / %** →**отляво надясно**

4) **Бинарни оператори + -** →**отляво надясно**

5) **Присвояване =** →**отдясно наляво**

CODE ACADEMY

Релационни оператори - за сравнение

<	<=
>	>=
==	!=

CODE ACADEMY

Условни оператори

- Условният оператор се използва, когато има разклоняване на алгоритъма.

- **If (условие)**
  {
  действия, които да се изпълнят, ако условието е **вярно**
  }
  else
  {
  действия, които да се изпълнят, ако условието **не е вярно**
  }

- **result = (условие)? a : b ;**
  резултатът ще приеме стойност **a**, ако условието е изпълнено
  резултатът ще приеме стойност **b**, ако условието не е изпълнено

CODE ACADEMY

Оператори за цикъл

Цикъл е част от кода, която се повтаря при определени условия

- Цикъл с определен брой стъпки

```
for ( int i=0; i < 100; i++) {    тяло на цикъла   }
```

- Цикъл, който се повтаря, ако е изпълнено определено условие

```
while ( условие ) {     тяло на цикъла   }
```

- Цикъл, който се повтаря, докато се изпълни определено условие

```
do {     тяло на цикъла   } while ( условие );
```

CODE ACADEMY

Масиви и структури

```
#include <stdio.h>

int main(){
    int arr[3] = {1,2,3};
    return 0;
}
```

Масиви – съвкупност от няколко **еднотипни променливи** с общо име – име на масива - записани на едно и също място в паметта.

int arr [56]; // посочва се броят елементи
дефинира масив от 56 цели числа, като първото от тях е **arr[0]**

Достъпът до елементите е с **индекс в диапазона 0 – (n-1)**, където n е броят елементи.

Структура – съвкупност от **разнотипни променливи** с общо име.

Структури могат да се разглеждат като нов тип данни.

```
struct car
{
    int wheels;
    int speed;
    char* model;
};
```

CODE ACADEMY

Въпроси?

CODE ACADEMY

32 Ключови думи

```
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
```

CODE ACADEMY

32 Ключови думи

auto	използва се за указване на автоматичен тип
break	използва се за терминиране на цикъл или опция
case	представя опция в изброима условна конструкция
char	използва се за дефиниране на символен тип
const	използва се за дефиниране на константа
continue	прекъсва текущата итерация на цикъл
default	използва се за подразбираща се обработка
do	дефинира блок в do-while цикъл

CODE ACADEMY

32 Ключови думи

double	дефинира разширен тип с плаваща запетая
else	използва се за алтернативния блок на if
enum	използва се за дефиниране на изброим тип
extern	използва се за декларации на външни типове
float	дефинира тип с плаваща запетая
for	дефинира цикъл със стъпка и условие за изход
goto	използва се за безусловен преход
if	използва се в условен блок

CODE ACADEMY

32 Ключови думи

int	дефинира целочислен тип
long	модификатор, който променя базовия тип
register	използва се регистров модификатор
return	терминира функция и евентуално връща резултат
short	модификатор, който променя базовия тип
signed	модификатор, който променя базовия тип
sizeof	връща паметта в байтове на дадена променлива
static	дефинира статични променливи и функции

CODE ACADEMY

32 Ключови думи

struct	дефинира структури
switch	дефинира изброим условен блок
typedef	дефинира име за даден тип
union	дефинира обединение
unsigned	модификатор, който променя базовия тип
void	използва се за липса на тип
volatile	използва се за непостоянни променливи (прекъсвания от устройства)
while	дефинира цикъл с условие за изход

CODE ACADEMY

Задачи:

Задача 1. Гравитацията на Луната е само 17 % от тази на земята. Напишете програма, в която потребителя да въвежда своето тегло на Земята и след изчисляване да показва теглото му на Луната.

Задача 2. След излизането от Еврозоната Великобритания въвежда отново унцията като мерна единица в заведенията за обществено хранене. Чашата има обем 8 унции. Напишете програма, която да пита потребителя колко унции желае и тя да връща броя на чашите, които е поръчал.

Задача 3.Една година на Юпитер(времето необходимо на Юпитер за да направи една пълна обиколка на Слънцето) има продължителност 12 земни години. Напишете програма, която конвертира въведените от потребителя земни дни в Юпитерски години.

CODE ACADEMY

Задачи:

Задача 4. Да се напише програма, която намира и отпечатва сумата от цифрите на едно четирицифрено число, което се въвежда на вход от клавиатурата.
Задача 5.Програма, която проверява дали едно число се дели на 7.
Задача 6. Програма, която въвежда 3 цели числа и ги извежда в нарастващ ред.
Задача 7. Програма, която чете число, докато бъде въведено число в интервала [2, 12)

Задача 8. Напишете програма, която отпечатва фигурата стълбичка

```
******
      ******
            ******
                  ******
```

CODE ACADEMY

Задачи

9. *Кой от посочените оператори printf() трябва да се използва, за да се отпечатат a и b?*

```
#include<stdio.h>
float a=3.14;
double b=3.14;
```

A. printf("%f %lf", a, b); B. printf("%Lf %f", a, b); C. printf("%Lf %Lf", a, b); D. printf("%f %Lf", a, b);

10. *За да се прочетат a и b, както и дадено по-долу, кой от следните scanf() оператори трябва да се използва?*

```
#include<stdio.h>
float a;
double b;
```

A. scanf("%f %f", &a, &b);

B. scanf("%Lf %Lf", &a, &b);

C. scanf("%f %Lf", &a, &b);

D. scanf("%f %lf", &a, &b);

CODE ACADEMY

Задачи

11. *Кой от посочените оператори printf() трябва да се използва, за да се отпечатат a и b?*

```
#include<stdio.h>
float a=3.14;
double b=3.14;
```

A. printf("%f %lf", a, b); B. printf("%Lf %f", a, b); C. printf("%Lf %Lf", a, b); D. printf("%f %Lf", a, b);

12. *За да се прочетат a и b, както и дадено по-долу, кои от следните scanf() оператори трябва да се използва?*

```
#include<stdio.h>
float a;
double b;
```

A. scanf("%f %f", &a, &b);

B. scanf("%Lf %Lf", &a, &b);

C. scanf("%f %Lf", &a, &b);

D. scanf("%f %lf", &a, &b);

CODE ACADEMY

Задачи:

13. Какъв ще бъде изходът от следния код?

```
#include <stdio.h>
int main()
{
   int i = 10, j = 2;
   printf("%d\n", printf("%d %d ", i, j));
  return 0;
}
```

A. Compile time error B. 10 2 4 C. 10 2 2 D. 10 2 5

14. Какъв ще бъде изходът от следния код?

```
#include <stdio.h>
int main()
{
  int i = 10, j = 3;
  printf("%d %d %d", i, j);
  return 0;
}
```

A. Compile time error B. 10 3 C. 10 3 some garbage value D. Undefined behavior

CODE ACADEMY

Задачи:

15. *Какъв ще бъде изходът от следния код?*

```
#include <stdio.h>
int main()
{
   int i = 10, j = 3, k = 3;
   printf("%d %d ", i, j, k);
}
```

A. Compile time error B. 10 3 3 C. 10 3 D. 10 3 somegarbage value

16. *Какъв ще бъде изходът от следния код?*

```
#include <stdio.h>
int main() {
  int n;
  scanf("%d", n);
  printf("%d\n", n);
}
```

A. Compilation error B. Undefined behavior C. Whatever user types D. Depends on the standard

CODE ACADEMY

Задачи:

17. *Какъв ще бъде изходът от следния код?*

```
#include <stdio.h>
int main() {
   short int i;
   scanf("%hd", &i);
   printf("%hd", i);
 }
```

A. Compilation error B. Undefined behavior C. Whatever user types D. None of the mentioned

18. *Посочете грешката в програмата*

```
#include<stdio.h>
int main() {
   char ch;
   int i;
   scanf("%c", &i);
   scanf("%d", &ch);
   printf("%c %d", ch, i);
 }
```

A. Error: suspicious char to in conversion in scanf() B. Error: we may not get input for second scanf() statement

C. No error D. None of above

CODE ACADEMY

КАКВО НАУЧИХМЕ ДНЕС!

Домашно - задачи 1, 2, 3 и задачи от 8 до 18.

Тестовите задачи 9 -18 не се предават!

CODE ACADEMY

Въпроси?