# Разработка на програми с използване на функции. Предаване на параметри по стойност.

# Функции в езика С

Функциите по своят смисъл представляват изнесени части от програмен код, наименуван с име и изпълняващ конкретна задача, алгоритъм или част от алгоритъм. За цел имат да намалят повторяемостта на код, който би затруднил четенето и би удължил ненужно програмата. Всяка програма в езика С има поне една функция – main() функцията, с която започва изпълнението на всяка програма или проект.

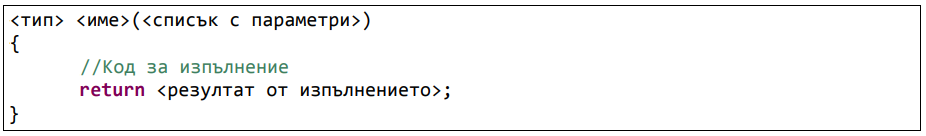
Декларацията на функция казва на компилатора името на функцията, типа данни, който връща и параметрите, които използва. Дефиницията на функцията представлява самото тяло с кода за изпълнение.

Функциите в програмирането са заимствани от математиката, където функциите имат резултат от изпълнението си и определен брой променливи, за които се извършва изчислението.

Стандартните библиотеки в езика предоставят достъп до множество функции, които могат да бъдат използвани на готово. Към функциите може да се отнесем както към методи или под – разделения на процедури.

# Структура на функция в програмния език С

Всяка функция, която се реализира в С има структура като тази:



Където:

**<тип>** - това е типът на резултата, който функцията ще върне. Типът може да е най-разнообразен – могат да се използват елементарните типове от данни в С, разгледани в предходните упражнения, както и потребителски типове, които ще разгледаме в следващи такива. Възможно е някои функции да не се налага да връщат резултат, тогава в полето се записва “void”.

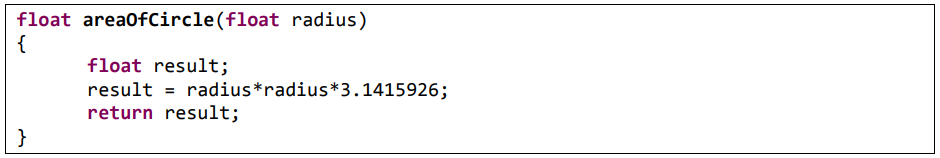
**<име>** - това е името на функцията, чрез която ще бъде използвана в програмата. Името на функцията е като име на променлива, с тази разлика, че след него се поставят скоби и следват евентуалните параметри

**<списък с параметри>** - параметри, които ще се използват във функцията, които са записани един след друг, като всеки параметър си има тип и име. Параметрите записани във вида си трябва да бъдат подадени по абсолютно същият начин и при извикването на функцията.

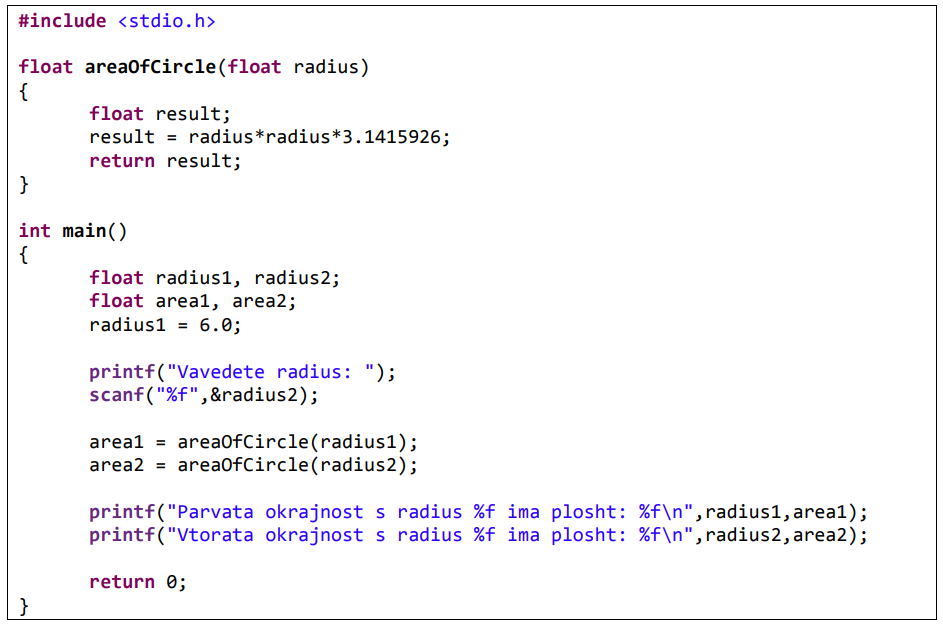
**върнат резултат** – всяка функция, която има тип, различен от ***void*** трябва задължително да върне резултат от изпълнението си. Този резултат се използва от програмата, където е извикана тази функция. Резултата може да се присвоява на променлива, подава като параметър на друга функция или използва в някой от наличните оператори.

# Пример за функция

Да се напише функция, която връща лицето на окръжност по зададен радиус.



Код на програмата:



Функция може да бъде деклариранa, след което описана след main() метода.

**<тип> <име> (<списък с параметри>);**

Функция, която връща по – голямото от две числа

int max(int, int);

void main()

{

int x=4,y=5;

printf(“%d”,max(x,y));

}

int max(int num1, int num2)

{

int result;

if (num1 > num2)

result = num1;

else

result = num2;

return result;

}

В случаят описваме прототипа на функцията преди основния метод, а самата декларация може да позиционираме след него. Имената на променливите не са задължителни при дефиницията, само типовете им и тяхната последователност.

При извикване на функция контрола над програмата се предава на нея, като всяка функция реализира заделяне на памет от общата за програмата и работи самостоятелно. След приключване на работата и връщане на конкретна стойност (или не връщане ) тази памет се освобождава. Това е и причината създадените променливи по време на функцията и обработките им да останат като живот само по време на самата функция и да не променят променливи от викащата или други функции.

***Една функция може да върне само една променлива.***

# Край на функция

Една функция се прекратява по два начина – чрез изпълнение на целия код в нея и достигане на скобата за завършване на тялото й или чрез запазената дума ***return.*** Return е дума оповестяваща, че функцията приключва своето действие и се връща при извикващия я. В този случай обикновено се добавя стойност, която да се върне на викащия, като така показваме каква е извършената работа от текущата функция. НЕ е нужно return да има стойност. При функция без тип – void – също може да имаме return, но без стойност, което просто ще прекрати функцията до момента на изпълнение. Една функция може да има повече от един return, което дава възможност за разклонение и генериране на сложни алгоритми. Командата в който и да е случай се изпълнява само веднъж и терминира работата на функцията, независимо къде се намираме в нейното тяло.

# Параметри на функция

При създаване на функция параметрите, които подаваме се създават като променливи в паметта на конкретната функция. Те имат живот с нея и при приключването й също биват изтрити. При викане на функция имаме възможност за подаване и работа с два вида параметри – по стойност и по референция.

* Викане по стойност – копират се стойностите от подаваните променливи към функцията в локални променливи за самата функция. По този начин работим с копията и оригиналната информация се запазва и не се променя.
* Викане по референция – копира се адреса на подаваната към функцията променлива, като по този начин работим с паметта от мястото, от което викаме функцията. При промяна на променливата се променя стойността не в паметта на функцията, а в реферираното пространство.

По подразбиране се използва викане по стойност.

Чрез функции може да се изгражда по – сложен код, който позволява генерирането на алгоритми, недостъпни по друг начин. Такъв пример е рекурсията, която ще се разгледа по – нататък в обучението по синтез и анализ на алгоритми.

# Задачи:

Да се напише програма, в която се въвежда цяло число, което представлява код на геометрична фигура – 1 – квадрат, 2 – правоъгълник, 3 – правоъгълен триъгълник, 4 – окръжност. След това се въвеждат размерите на избраната геометрична фигура. Програмата трябва да изведе типа на геометричната фигура, както и нейното лице. За пресмятането на лицето да се напишат подходящи функции

Напишете функция за размяна на две числа. Демонстрирайте работата й.

Напишете функция, която проверява дали конкретно число е четно или нечетно.

Напишете функция, която превръща от десетично в двоично едно число.