Как функциите правят живота ни по-лесен

Програмиране най-лесно и приятно се учи чрез примери и аналогии на неща които вече знаем и разбираме, затова нека вместо да се убеждаваме кое е полезно и кое не, просто да го покажем. За тази цел ще погледнем какво са функциите и как се използват с един прост пример, който използва функция.

Що е то функция?

Нека разгледаме долния ред код.

```
int number = 136;
```

Тук се случват няколко неща:

- Декларира се променлива с **име** `number` от **тип** `int`
- Инициализира се със **стойност** `136`.

Ако двете операции се правят на отделни редове биха изглеждали така:
Декларация: `int number;`

Инициализация: `number = 136;`

Както забелязваме за инициализация на променлива е нужен **оператор =**, когато работим с функции, нямаме инициализация, но имаме по-сложна декларация, тази декларация не включва само име и тип, а и още няколко елемента. Нека разгледаме как се декларира функция. Тъй като е по-лесно да имаме някакъв контекст нека разгледаме тази програма, която показва дали резултатът от операцията **13 % 4** е четно или нечетно число.

```
#include <iostream>

bool remainderIsEven(int number, int divisor)
{
  int remainder = number % divisor;

  return remainder % 2 == 0;
}

int main()
{
  bool remainderOfThirteenAndFourIsEven = remainderIsEven(13, 4);

  if (remainderOfThirteenAndFourIsEven)
  {
    std::cout << "The result of 13 % 4 is even." << std::endl;
  }
  else
  {
    std::cout << "The result of 13 % 4 is odd." << std::endl;
```

```
  }

  return 0;
}
```

Тук се случват много неща, нека ги разгледаме едно по едно. Ще започнем с различните аспекти на функцията **remainderIsEven** като даваме допълнителна, полезна информация в контекста на обясненията.
Декларация на функция:

```
bool remainderIsEven(int number, int divisor)
{
  int remainder = number % divisor;

  return remainder % 2 == 0;
}
```

Тази функция има следните елементи:

- Име – `remainderIsEven` – както променливите, функциите също имат имена, нужни са ни, за да можем да ги достъпваме по някакъв начин
- Тип – `bool` – функциите могат да имат или да нямат резултат, например **int main()** има тип **int** – както се сещаме, в края на програмата връщаме 0 с цел да съобщим за успешно изпълнение на програмата. **main** е функция, точно както е и **remainderIsEven**, разликата е в това, че е прието **main** да е функцията, от която започва изпълнението на дадена програма. **Типът на функцията трябва да отговаря на типа на стойността, която връщаме от функцията – връщаме стойност чрез ключовата дума `return`. В случая** `remainder % 2 == 0;` **е булев израз, който приема стойности true/false.**

- Параметри – `(int number, int divisor)` – функцията може да приема произволен брой параметри, това са просто променливи, които се изброяват със запетая и са в скоби, непосредствено след името на функцията. Тези променливи се създават извън тази функция и стойностите на тези променливи се подават на функцията, например `remainderIsEven(13, 4);` - това е извикване на функцията с **number = 13** и **divisor = 4.** Важно е правим разлика между декларация на функция и извикването и. Когато я декларираме ние и „казваме“ какви действия да извърши със стойностите в променливите(които ще наричаме параметри). Тези стойности се появяват едва след като тази функция бъде извикана. Компилаторът намира всички срещания на променливата **number** и слага числото 13 на това място, също се прави и за **divisor.** След това се правят сметките и се връща резултат. (Последното изречение не е напълно вярно, но е добър начален начин на мислене, който да ни помогне да разберем същността на функциите)

- Тяло -

```
int remainder = number % divisor;

return remainder % 2 == 0;
```

Тялото на функцията е мястото, където става самото изпълнение на функцията – след като тя се извика от друго място, в нашия случай от **main -** `remainderIsEven(13, 4);` изпълнението на програмата отива на първия ред от функцията, точно на реда на първата кудрава скоба, в този момент всички параметри имат стойности и започва изпълнението на функцията. В момента, в който се стигне до ключовата дума **return,** програмата продължава изпълнение от мястото, където е извиката функцията – в нашия случай следващата стъпка е да се вземе резултатът от извикването на тази функция и да се запази в променлива, която се използва по-надолу, за да се изпишат неща на конзолата -

```
bool remainderOfThirteenAndFourIsEven = remainderIsEven(13, 4);
```

Тялото на функцията винаги трябва да бъде обградено от кудрави скоби. Това е начинът компилаторът да знае къде започва и къде завърша функцията. Както казахме има функции, които могат да не връщат никаква стойност – тип **void**, тогава те не са задължени да имат ключовата дума **return** в тялото си. Функцията просто ще приключи в момента, в които се стигне затварящата кудрава скоба.

Нека обобщим. Функциите имат 4 „елемента“ – **име**, **тип**, **параметри**(възможно е да са 0) и **тяло.**

- Извикването на функцията става като добавим **()** след името, ако функцията няма никакви параметри – **myFunction()**, или като добавим параметрите в тези скоби, ако има такива - **isEven(2), multiplyThreeNumbers(3,5,7).** Тези параметри може и да са променливи - **multiplyThreeNumbers(a, b, c) –** където **a, b, c** са декларирани преди извикването на функцията.
- Ако искаме да запазим стойността от извикването на функцията, трябва да я запазим в променлива, която да използваме надолу в програмата.

Въпроси?