Лабораторно упражнение N 10
Програмни модули в C/C++. Функции

I. Теоретична обосновка

1. Видове програмни модули

В програмните езици, за видовете програмни модули се използва следната класификация:

- главна програма;
- подпрограми (процедури, функции).

Главна програма е програмен модул, на който операционната система предава управлението при стартиране на изпълнимия код. При изпълнението си, главната програма може да предаде управлението на други програмни модули, наречени ***подпрограми***. След завършване, главната програма връща управлението на операционната система.

Подпрограма е програмен модул, който получава управлението от друг програмен модул т.е. подпрограмата се извиква от друг модул. След приключване, подпрограмата връща управлението в програмния модул, от където е извикана.

За разлика от други програмни езици, където подпрограмите се разделят на процедури и функции, в C и C++ съществува само един тип програмни модули - ***функции***.

Функция е самостоятелен фрагмент от програма, съдържащ описания на променливи и набор оператори на езика. Фрагментът е затворен във скоби {…} и в крайна сметка се изпълнява като един обобщен оператор.

Всяка програма на C или C++ се състои от една или повече функции. Задължително трябва да съществува една главна функция с име ***main().*** Функция ***main()*** изпълнява роля на главна програма. Това е функцията, към която първоначално се предава управлението от операционната система. След приключване на изпълнението, функция ***main()*** връща управлението на операционната система.

Като самостоятелен програмен модул, функцията може да бъде извиквана многократното, като се стартира с различни входни данни (променливи). Входните променливи се наричат още ***параметри*** на функцията.

След завършването си функцията връща ***резултат***. Като програмен модул функцията връща чрез името си, един резултат.

Параметрите и ***резултатът*** са връзката на функцията с останалите програмни модули. Идея за използването на функции е те да могат да се извикват с различни входни данни.

2. Дефиниране на функции. Оператор return

Дефиниция на функция представлява цялостното описание на функцията. Описанието на функцията се състои две части:

- **заглавна част (*прототип*);**
- **тяло,** което съдържа декларации на локални променливи и оператори.

Общият вид на дефиницията на функция е следният:

```
тип  име (списък формални параметри)        // прототип на функция
{                                            // тяло на функция
   // описание на локални променливи
```

// операции;

}

Тип на функцията е типът на стойността, която функцията ще върне като резултат. По подразбиране функциите са от тип ***int*** т.е. ако не бъде записан тип, счита се, че функцията е от тип ***int***. Чрез името си, функцията връща един резултат.

Име на функцията е идентификатор, чрез който еднозначно се определя функцията.

Формални параметри са списък входни променливи, чрез който се осъществява връзката между функцията и останалите функции. Те имат описателно значение.

Списък формални параметри се представя в следния вид:
(тип1 променлива1 , тип2 променлива2, ...)

Тялото на функцията включва множество декларации на локални променливи и оператори, реализиращи задачата на функцията.

3. Извикване на функция. Предаване на параметри

Извикването на функция става чрез името и. При извикване на функцията, в скобите след името се задават ***фактическите параметри*.** Извикването на функция се задава като:

име (списък фактически параметри);

Подобно на формалните, фактическите параметри са разделени със запетаи.

Фактическите или **действителни** параметри са константи, променливи или изрази, които при извикване на функцията предават своите стойности на формалните параметри, за да се изпълни функцията. Този процес се нарича ***свързване на формалните с фактическите параметри***.

Фактическите и формалните параметри си съответстват по брой и по тип.

4. Декларации на функции

В случай, че се наложи функцията да бъде извикана преди нейната дефиниция, необходимо е, функцията да бъде декларирана.

Декларацията на функцията е нейният прототип (заглавната част на функцията) последван от символ ***;***

5. Примери за реализация на функции

5.1. Намиране на N факториел

// намиране на N факториел чрез използване на функция

```
#include <stdio.h>
int Factoriel(int n);

main()
{
      int n,fact;
      printf("n=");
      scanf("%d",&n);
      fact = Factoriel(n);
```

```
	printf("N Factoriel = %d",fact);
	return 0;
}

int Factoriel(int n)
{
	int nf=1,i;
	for(i=1;i<=n;i++)
		nf*=i;
	return nf;
}
```

II. Задачи за изпълнение

1. Да се създаде конзолно приложение, което пресмята лицата на различни фигури: кръг, правоъгълник, триъгълник. Пресмятането на лицата на фигурите да се извършва с отделни функции. В проекта да се включи и функция, която проверява дали стойностите за страните на триъгълника са валидни.

2. Да се дефинира функция, която намира разстояние между две точки, координатите на който се предават като параметри. Да се създаде конзолно приложение, което намира разстояние между обща дължина на начупена линия.

Упътване: Координатите на точките да се дефинират като два отделни масива – първият да съдържа x- координатите, а втория – y-координатите.

3. Да се състави конзолно приложение за намиране на e^x в ред на Фурие. Натрупването на междинната сума да се прекрати, когато поредния член стане по-малък от 10^{-8}. Формулата за пресмятане е: $e^x=1+x/1+x^2/2!+x^3/3!+...+x^n/n!$

Да се сравни получената стойност със стойността, получена чрез използване на функция exp().

Упътване: Да се използва функция за намиране стойността на n!.