МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Инженерно-экономический факультет

Кафедра экономики

Отчет

по лабораторной работе №8

на тему «**ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ФУНКЦИЙ**»

по курсу «Основы алгоритмизации и программирования»

Вариант №7

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Студент гр. 274003  Жерко В.A. |
| Проверила: |  | Петрович Ю.Ю. |

Минск 2022

**Цель: Изучить принцип работы функций и прототипов функций**

**Краткие теоретические сведения:**

**Функции** — это блок кода, который вы можете использовать в любом участке вашей программы неограниченное количество раз.

<тип данных, который будет возвращаться функцией> <имя> (<аргументы функции>) {

  < блок кода > }

* **Тип данных функции**. В самом начале нам нужно указать тип данных ,который в конечном итоге будет передавать функция.
* **Имя функции.**Нам нужно задать функции имя (исключениями являются зарезервированные слова в C++, имена начинающиеся с цифр, а также имена разделенные пробелом). Лучше всего задавать такое имя, которое будет говорить нам в будущем, за что отвечает эта функция. И тогда в будущем нам не придется  вспоминать, за что она отвечает.
* **Аргументы функции.**В скобках (после имени функции) могут находиться аргументы функции. **Аргумент функции** — это значение, которое можно передать функции при ее вызове. Если аргумент функции не один, а их несколько, то их нужно разделять запятой.
* Если аргументов в функции нет, то в скобках можно указать тип void. Но писать его необязательно, он стоит по умолчанию.
* **Блок кода**. После открывающей скобки идет блок кода, который будет начинать работать при вызове функции.

**Прототипы функций**

**Прототип функции** — это функция, в которой отсутствует блок кода (тело функции). В прототипе функции находятся:

* Полное имя функции.
* Тип возвращаемого значения функции.

int Sum\_numbers(int a, int b);   // прототип функции

int main() {

int a,b;

cin >> a,b;

Sum\_numbers(a, b);

}

int Sum\_numbers(int a, int b) {  // сама функция

  cout << a + b;

}

Можем заметить, что реализация функции происходит после блока main, однако функция объявлена до main и компилятор знает о ее существовании и готов к ее исполнению.

**Индивидуальное задание (Вариант 7):**

Вывести на экран таблицу значений функции Y(x) и ее разложения в ряд

S(x) с точностью ε . Вывести число итераций, необходимое для достижения заданной точности. Вычисление S(x) и Y(x) оформить в виде функций.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | *a* | b | *S*(x) | ε | *Y*(x) |



**Код программы:**

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

typedef double (\*datavar)(double, double, int&);

void chart(double startpoint, double endpoint, double step, double eps, datavar nameofchart) {

int k = 0;

double sum;

for (double x = startpoint \* 10; x <= endpoint \* 10; x += 1) {

sum = nameofchart(double(x / 10), eps, k);

cout << setw(10) << double(x / 10) << setw(13) << sum << setw(10) << k << endl;

}

}

double SecondFunction(double x, double eps, int& j) {

double iteration = 1;

j = 0;

iteration = (double)(1 / 4) \* log((1 + x) / (1 - x)) - pow((double)1 / (2 \* tan(x)),-1);

return iteration;

}

double FirstFunction(double x, double eps, int& k) {

double sum = 0, iteration = x;

k = 1;

while (fabs(iteration) > eps) {

iteration = pow(x, 4 \* k - 1) / (4 \* k - 1);

sum += iteration; k++;

}

return sum;

}

int main() {

cout << setw(10) << "x " << setw(19) << "FirstFunction(x)" << setw(3) << " Number of Iterations : " << endl;

chart(-0.5, 0.5, 0.1, 0.00001, FirstFunction);

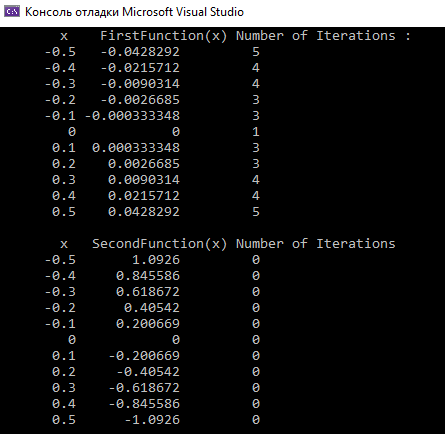
cout << endl;

cout << setw(10) << "x " << setw(19) << "SecondFunction(x)" << setw(3) << " Number of Iterations" << endl;

chart(-0.5, 0.5, 0.1, 0.00001, SecondFunction);

return 0;

}

Вывод результата в консоль:

**Вывод:**

В восьмой лабораторной работе изучил метод решений задач при помощи функций. Данный метод позволяет очень сильно экономить время, занимаемое пространство в коде, когда необходимо использовать один и тот же алгоритм несколько раз. Также применил typedef для определения псевдонима функции / переменной и т.д, что опять-таки экономит время написания и занимаемые строки кода.