Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра систем управления

Отчет по лабораторной работе № 1

Тема: «Программирование линейных алгоритмов»

Вариант 7

Выполнил:

студент группы 322441 Калинин В.А.

Проверил:

зам.каф ИТАС Навроцкий А.А.

Минск

2023

Цель работы

Цель работы: изучить правила составления текстов программ, научиться реализовывать линейные алгоритмы; написать и отладить программу, реализующую линейный алгоритм.

**Краткие теоретические сведения**

Арифметические операции

Самыми простыми арифметическими операциями являются +, –, \*, /. Данные операции применимы как к целым, так и к вещественным типам данных и правила их использования аналогичны их использованию в математических вычислениях. Порядок следования операций можно изменять с помощью скобок.

Для работы только с целыми числами существует операция получения остатка от деления – %.

Например: 10 % 6 = 4, 7 % 10 =7, 10 % 5 =0.

Формат операции:

оператор\_1 = оператор\_2;

В переменную оператор\_1 заносится значение оператора\_2. В качестве оператора\_1 можно использовать только переменную. В качестве оператора\_2 можно использовать константу, переменную, выражение или функцию.

Допустимо использовать следующее написание:

a = b = c = d; что равнозначно a = d; b = d; c =d ;

Часто в программировании используются операции такого типа:

оператор\_1 = оператор\_1 знак\_операции оператор\_2;

Для сокращения записей таких операторов можно использовать сокращенную форму записи:

оператор\_1 знак\_операции = оператор\_2;

Например, оператор

s = s + 2;

можно заменить оператором

s += 2;

Если оператор\_2 для операций суммирования и вычитания равен единице, то лучше использовать операции инкремента:

оператор\_1++;

или декремента

оператор\_1--;

Например, вместо написания i = i + 1 можно использовать i++, а вместо написания i = i – 1 можно использовать i --.

Знак инкремента или декремента может быть записан в двух формах: в префиксной (например ++i) или в постфиксной (например i++). Способ написания влияет на порядок выполнения операций в выражении. При префиксной форме сначала выполняется инкремент или декремент, а затем арифметические операции. Если используется постфиксная форма, то сначала выполняются арифметические операции, а затем инкремент или декремент.

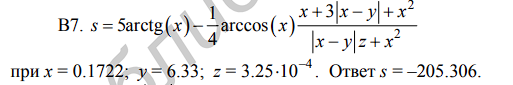
Приоритет операций в С++

Приоритет операций в языке С++ представлен в табл. 3.2. Приоритет уменьшается сверху вниз.



Задание

Задание: Вычислить значение выражения при заданных исходных данных. Сравнить полученное значение с указанным правильным результатом.



Листинг программы

#include <iostream>

#include <math.h>

using namespace std;

double resolver(double x, double y, double z) {

double first = 5 \* atan(x);

double second = 0.25 \* acos(x);

double third = x + 3 \* abs(x-y) + x \* x ;

double forth = abs(x-y) \* z + x \* x;

return first - second \* third / forth;

}

int main()

{

double x,y,z;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

x = 0.1722;

y = 6.33;

z = 3.25e-4;

double s = -205.306;

cout << "S =" << resolver(x,y,z) << endl << "expected S =" << s << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

system("pause");

return 0;

}

Результат работы программы

