МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра № 304 «Вычислительные машины, системы и сети»

Информатика

Отчет по лабораторной работе № 2

Вариант № 9

Выполнила бригада группы М3О-111Бк-21

Багиров Э. Р.

Нуриев Н. Н.

Проверил Секретарев Виталий Евгеньевич

Москва 2021 г.

Оглавление

[**Задание** 3](#_Toc86736331)

[**Диаграмма алгоритма** 4](#_Toc86736332)

[**Алгоритмический вид программы** 5](#_Toc86736333)

[**Исходный код программы** 6](#_Toc86736334)

[**Некорректные тесты** 7](#_Toc86736335)

[**Корректные тесты** 8](#_Toc86736336)

[**Протокол расчета ожидаемого результата** 11](#_Toc86736337)

[**Вывод** 12](#_Toc86736338)

# **Задание**

Кафедра: 304 Курс: ИНФОРМАТИКА

Задание 2: Табулирование функций

ВАРИАНТ № 9

Разработать, отладить и протестировать программу, которая разбивает отрезок [A;B] на N интервалов; в зависимости от значений A, B, N вычисляет величину шага h=(B-A)/N, с которым происходит приращение аргумента xi=A+ i \* h, и в каждой точке xi печатает значение функций:

Результаты вычислений представить в виде таблицы с элементами псевдографики. В качестве одного из вариантов исходных данных взять: A=0, B=pi, N=10.

# **Диаграмма алгоритма**

# **Алгоритмический вид программы**

Лабораторная работа 2 по информатике

9 вариант

АЛГ Задание 2: Табулирование функции

ПЕР A, B, N, x, h, i

НАЧАЛО

//ИНИЦИАЛИЗАЦИЯ ПЕРЕМЕННЫХ

А //Координата начала отрезка [A;B]

В //Координата конца отрезка [A;B]

N //Количество интервалов, на которые поделили отрезок [A;B]

h //Величина шага

x //Приращение аргумента

i //Индекс приращения

//ВВОД И ПРОВЕРКА ДАННЫХ

ВВОД A, B

ЕСЛИ А > B

ТО ПЕЧАТЬ (сообщение об ошибке)

КОНЕЦ

КОНЕСЛИ

ВВОД N

ЕСЛИ N < 0 //Предотвращение некорректного ввода количества интервалов

ТО ПЕЧАТЬ (сообщение об ошибке)

КОНЕЦ

КОНЕСЛИ

h = (B - A) / N //Расчет величины шага

// Если входной отрезок есть точка, изменить количество разделений

ЕСЛИ A = B

ТО N = 0

КОНЕСЛИ

НЦ ДЛЯ i ОТ 0 ДО N

x = A + i \* h //Приращение аргумента

F = (x - 1) \*\* 3 //Вычисление F(x)

G = ((х + 5) \*\* 3) / (1 + sin(x) \*\* 2) //Вычисление G(x)

//ВЫВОД

ВЫВОД В ТАБЛИЦЕ i, x, F, G

КЦ

КОНЕЦ

# **Исходный код программы**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\

\* C++ наш, сущий в памяти! \*

\* да компилируется код Твой; \*

\* да приидет царствие Софта Твоего; \*

\* да будут действительны указатели Твои \*

\* и в ОЗУ, как на жестком диске; \*

\* массив наш насущный подавай нам на каждый день; \*

\* и прости нам варнинги наши, \*

\* как и мы избавляемся от ошибок наших; \*

\* и не введи нас в бесконечный цикл, \*

\* но избавь нас от винды. \*

\* Ибо Твое есть Царство и сила и слава во веки. \*

\* Энтер. \*

\\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <iomanip>

using namespace std;

// псевдографика - заголовок таблицы

void HeadTable() {

cout << "\t"

<< char(218) << setfill(char(196)) << setw(20)

<< char(194) << setfill(char(196)) << setw(20)

<< char(194) << setfill(char(196)) << setw(20)

<< char(194) << setfill(char(196)) << setw(20)

<< char(191) << endl;

cout << "\t"

<< char(179) << setfill(' ') << setw(19)<< "#"

<< char(179) << setfill(' ') << setw(19)<< "x"

<< char(179) << setfill(' ') << setw(19)<< "F(x)"

<< char(179) << setfill(' ') << setw(19)<< "G(x)"

<< char(179) << endl;

cout << "\t"

<< char(195) << setfill(char(196)) << setw(20)

<< char(197) << setfill(char(196)) << setw(20)

<< char(197) << setw(20)

<< char(197) << setw(20) << char(180) << endl;

cout.setf(ios::showpoint | ios::fixed);

}

// псевдографика - завершение таблицы

void BottomTable() {

cout << "\t"

<< char(192) << setfill(char(196)) << setw(20)

<< char(193) << setfill(char(196)) << setw(20)

<< char(193) << setfill(char(196)) << setw(20)

<< char(193) << setfill(char(196)) << setw(20)

<< char(217) << endl;

}

// псевдографика - тело таблицы

void BetweenTheRaws() {

cout << "\t"

<< char(195) << setfill(char(196)) << setw(20)

<< char(197) << setfill(char(196)) << setw(20)

<< char(197) << setfill(char(196)) << setw(20)

<< char(197) << setfill(char(196)) << setw(20)

<< char(180) << endl;

}

double power(double a, int n) {

double b = a;

for (int i = 1; i < n; i++) {

b \*= a;

}

return b;

}

// расчет значения функции F(x) в точке x

double F(double x) {

return power(x - 1, 3);

}

// расчет значения функции G(x) в точке x

double G(double x) {

return power(x + 5, 3) / (1 + power(sin(x), 2));

}

// вывод строк таблицы с автоматическим закрытием таблицы

void PrintTable(double x, int i, int N, bool debug = false) {

if (debug) {

cout << x << "\t" << F(x) << "\t" << G(x) << endl;

}

else {

cout << "\t" << char(179) << setw(19) << setfill(char(255)) << i

<< char(179) << setw(19) << setprecision(4) << x

<< setfill(char(255))

<< char(179) << setw(19) << setprecision(4) << F(x)

<< setfill(char(255))

<< char(179) << setw(19) << setprecision(4) << G(x)

<< setfill(char(255)) << char(179) << endl;

if (i != N)

BetweenTheRaws();

else

BottomTable();

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

system("color F0");

//Инициализация переменных

double A;

double B;

int N;

double h;

double x;

int i;

bool debug = false;

// Ввод и проверка данных

cout << "Введите А - координату начала отрезка [A;B]: \t";

cin >> A;

cout << "Введите B - координату конца отрезка [A;B]: \t";

cin >> B;

if (A > B) {

cout << "\nОШИБКА! А должен быть меньше или равен В!";

return 1;

}

cout << "Введите N - количество интервалов, на которые разделен отрезок [A;B]: \t";

cin >> N;

if (N <= 0) {

cout << "\nОШИБКА! N должно быть больше 0! ";

return 1;

}

// Подготовка таблицы

setlocale(LC\_ALL, "C");

if (!debug) HeadTable();

h = (B - A) / N; // Расчет величины шага

// Если входной отрезок есть точка, изменить количество разделений

if (A == B) N = 0;

for (i = 0; i <= N; i++) {

x = A + i \* h;

PrintTable(x, i, N, debug);

}

return 0;

}

# **Некорректные тесты**

**Тест №1**

Цель теста: проверить работу программы на границе некорректной области исходных данных

Исходные данные: Eps = 0

Ожидаемый результат: вывод сообщения об ошибке: “ ОШИБКА - Eps должен быть больше 0.”

Полученный результат:

Изображение выглядит как текст

Автоматически созданное описание

Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

**Тест №2**

Цель теста: проверить работу программы в некорректной области исходных данных

Исходные данные: Eps = -0.1

Ожидаемый результат: вывод сообщения об ошибке: “ ОШИБКА - Eps должен быть больше 0.”

Полученный результат:

Изображение выглядит как текст

Автоматически созданное описание

Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

# **Корректные тесты**

**Тест №1**

Цель теста: проверить работу программы в корректной области

Исходные данные: Eps = 0.5

Ожидаемый результат:

Расчет ожидаемой суммы:

|  |  |  |  |
| --- | --- | --- | --- |
| N | Формула | Sum | Delta (Eps) |
| 1 |  | .5 | .5 |

Вывод сообщения:

Сумма получена

Количество просуммированных элементов N = 1

Сумма Sum = 0.5

Достигнутая точность Delta = 0.5

Полученный результат:

Изображение выглядит как текст

Автоматически созданное описание

Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

**Тест №2**

Цель теста: проверить работу программы в корректной области

Исходные данные: Eps = 0.3

Ожидаемый результат:

Расчет ожидаемой суммы:

|  |  |  |  |
| --- | --- | --- | --- |
| N | Формула | Sum | Delta (Eps) |
| 1 |  | .5 | .5 |
| 2 |  | .(6) | .(3) |
| 3 |  | .75 | .25 |

Вывод сообщения:

Сумма получена

Количество просуммированных элементов N = 3

Сумма Sum = 0.75

Достигнутая точность Delta = 0.25

Полученный результат:

Изображение выглядит как текст

Автоматически созданное описание

Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

**Тест №3**

Цель теста: проверить работу программы в корректной области

Исходные данные: Eps = 0.2

Ожидаемый результат:

Расчет ожидаемой суммы:

|  |  |  |  |
| --- | --- | --- | --- |
| N | Формула | Sum | Delta (Eps) |
| 1 |  | .5 | .5 |
| 2 |  | .(6) | .(3) |
| 3 |  | .75 | .25 |
| 4 |  | .8 | .2 |

Вывод сообщения:

Сумма получена

Количество просуммированных элементов N = 4

Сумма Sum = 0.8

Достигнутая точность Delta = 0.2

Полученный результат:

Изображение выглядит как текст

Автоматически созданное описание

Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

**Тест №4**

Цель теста: проверить работу программы в корректной области

Исходные данные: Eps = 0.15

Ожидаемый результат:

Расчет ожидаемой суммы:

|  |  |  |  |
| --- | --- | --- | --- |
| N | Формула | Sum | Delta (Eps) |
| 1 |  | .5 | .5 |
| 2 |  | .(6) | .(3) |
| 3 |  | .75 | .25 |
| 4 |  | .8 | .2 |
| 5 |  | .8(3) | .1(6) |
| 6 |  | .(857142) | .(142857) |

Вывод сообщения:

Сумма получена

Количество просуммированных элементов N = 6

Сумма Sum = 0.857143

Достигнутая точность Delta = 0. 142857

Полученный результат:

Изображение выглядит как текст

Автоматически созданное описание

Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

**Тест №5**

Цель теста: проверить работу программы в корректной области

Исходные данные: Eps = 0.1

Ожидаемый результат:

Расчет ожидаемой суммы:

|  |  |  |  |
| --- | --- | --- | --- |
| N | Формула | Sum | Delta (Eps) |
| 1 |  | .5 | .5 |
| 2 |  | .(6) | .(3) |
| 3 |  | .75 | .25 |
| 4 |  | .8 | .2 |
| 5 |  | .8(3) | .1(6) |
| 6 |  | .(857142) | .(142857) |
| 7 |  | .875 | .125 |
| 8 |  | .(8) | .(1) |
| 9 |  | .9 | .1 |

Вывод сообщения:

Сумма получена

Количество просуммированных элементов N = 9

Сумма Sum = 0.9

Достигнутая точность Delta = 0.1

Полученный результат:

Изображение выглядит как текст

Автоматически созданное описание

Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

**Тест №6**

Цель теста: проверить работу программы в корректной области

Исходные данные: Eps = 0.00000001

Ожидаемый результат: вывод сообщения об ошибке:

ОШИБКА - Достигнуто ограничение по точности подсчета.

Полученный результат:

Изображение выглядит как текст

Автоматически созданное описание

Вывод: полученный результат совпал с ожидаемым. Тест ошибки не обнаружил.

# **Вывод**

Проанализирован алгоритм по разбитию отрезка на интервалы, вычислению значения функций в найденных концах интервалов, форматированию выходных данных в виде таблицы. В ходе выполнения лабораторной работы были составлены диаграмма алгоритма и алгоритмический текст программы для дальнейшего написания программы. Проведены тесты для проверки соответствия исходных данных и корректные тесты. В результате проведения тестов ошибок в программе не обнаружено. Разработка программы завершена на основании того, что полученные результаты совпали с ожидаемыми, набор тестов считаем полным.