ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»  
(РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

ОТЧЁТ  
О ЛАБОРАТОРНОЙ РАБОТЕ №3

По дисциплине «Языки программирования»

Вариант 9

Выполнил: ст. гр. ТКИ - 141

Сивунов Пётр Сергеевич

Проверил: к.т.н., доц. Васильева М. А.

(Проверил: к.т.н, доц. Балакина Е. П.)

Москва 2024

3.1.1 Формулировка задания

Протабулировать заданную в таблице функцию. Использовать данные в таблице значения шага и интервала в качестве ввода пользователя для решения тестового примера. При невозможности расчёта функции в конкретной точке выводить её значение и надпись, означающую отсутствие решения. При решении данного задания в MatLab необходимо построить график!

Таблица  – Исходные данные

|  |  |  |
| --- | --- | --- |
| Вариант | Функция | Константы |
| 9 |  |  |

3.1.2 Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы функций представлены ниже (Рисунок 2).

Изображение выглядит как текст, снимок экрана, диаграмма, чек

Автоматически созданное описание

Рисунок  ­ Блок-схема основного алгоритма

Изображение выглядит как текст, диаграмма, План, Технический чертеж

Автоматически созданное описание

Рисунок  – Блок-схема print\_table

Изображение выглядит как диаграмма, зарисовка, рисунок, шаблон

Автоматически созданное описание

Рисунок 3 – Блок-схема get\_y и функций ввода

3.1.3 Текст программы на языке C

#include <corecrt\_wctype.h>

#include <float.h>

#include <locale.h>

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

/\*\*

\* @brief Выводит все значения функции

\* @param bottom\_limit Нижняя граница

\* @param top\_limit Верхняя граница

\* @param delta шаг

\*/

void print\_table(const double bottom\_limit, const double top\_limit, const double delta);

/\*\*

\* @brief Расчитывает значение функции в точке x

\* @param x

\* @return значение функции в точке x

\*/

double get\_y(const double x);

/\*\*

\* @brief Функция ввода вещественного числа

\* @return вещественное число

\*/

double double\_input(void);

/\*\*

\* @brief Ввод и проверка вещественного числа на положительность

\* @return Вещественное число

\*/

double get\_positive\_double(void);

/\*\*

\* @brief Точка входа в программу

\* @returns 0 в случае успеха

\*/

int main(int argc, char\* argv[])

{

setlocale(LC\_ALL, "Rus");

puts("Нижняя граница");

double bottom\_limit = double\_input();

puts("Верхняя граница");

double top\_limit = double\_input();

puts("Шаг");

const double delta = get\_positive\_double();

print\_table(bottom\_limit, top\_limit, delta);

return 0;

}

void print\_table(const double bottom\_limit, const double top\_limit, const double delta)

{

for (double i = bottom\_limit; i <= top\_limit + DBL\_EPSILON; i += delta)

{

if (i <= DBL\_EPSILON)

{

printf("Нет значения в точке %.2f\n", i);

}

else

{

const double y = get\_y(i);

printf("%.2f: %.2f\n", i, y);

}

}

}

double get\_y(const double x)

{

return 0.1 \* pow(x, 2) - x \* log10(x);

}

double double\_input(void)

{

double value = 0;

if (scanf\_s("%lf", &value) != 1)

{

abort();

}

return value;

}

double get\_positive\_double(void)

{

const double value = double\_input();

if (value < DBL\_EPSILON)

{

abort();

}

return value;

}

3.1.4 Результаты выполнения программы

Результаты выполнения программы представлены ниже.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 4 – Результаты выполнения программы

3.1.5 Выполнение тестовых примеров

Изображение выглядит как текст, снимок экрана, число

Автоматически созданное описание

Рисунок 5 – Тест в Excel

3.2.1 Формулировка задания

Составьте две программы:

a) вычислить сумму первых n членов последовательности (k = 1, 2, 3 ..., n).

b) вычислить сумму всех членов последовательности, не меньших заданного числа e.

Помните о проверке пользовательского ввода. Все результаты вывести на экран. Отчёт дополнить блок-схемой. При вычислении факториалов рекомендуется отказаться от использования рекурсивных методов

Таблица 2 – Исходные данные

|  |  |
| --- | --- |
| Вариант | Функции |
| 9 |  |

3.2.2 БЛОК-СХЕМА АЛГОРИТМА

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 7 – Блок схема функции main

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 8 - Блок схема функций для задания

Изображение выглядит как текст, снимок экрана, Шрифт, Графика

Автоматически созданное описание

Рисунок 9 - Блок схема функций ввода

3.2.3 ТЕКСТ ПРОГРАММЫ НА ЯЗЫКЕ C

#include <float.h>

#include <locale.h>

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

/\*\*

\* @brief Элемент последовательности по счету k

\* @param previous\_element - предыдущий элемент

\* @param k - порядковый номер

\* @return Элемент последовательности по счету k

\*/

double get\_element(const double previous\_element, const int k);

/\*\*

\* @brief Сумма первых n элементов

\* @param n Количество первых элементов

\* @return Сумма первых n элементов

\*/

double get\_elements\_sum(const int n);

/\*\*

\* @brief Функция ввода целого числа

\* @return Целое число

\*/

int int\_input(void);

/\*\*

\* @brief Функция ввода вещественного числа

\* @return Вещественное число

\*/

double double\_input(void);

/\*\*

\* @brief Сумма элементов которые больше e

\* @param e ограничение

\* @return Сумма всех элементов больше e

\*/

double get\_sum\_greater\_equal\_e(const double e);

/\*\*

\* @brief Ввод и проверка вещественного числа на положительность

\* @return Вещественное число

\*/

double get\_positive\_double(void);

/\*\*

\* @brief Ввод натурального числа

\* @return Натуральное число

\*/

int get\_natural\_int(void);

/\*\*

\* @brief Выбор пользователя

\*/

typedef enum

{

TASK\_A = 1, /\*\*< Задание 1 \*/

TASK\_B = 2 /\*\*< Задание 2 \*/

} UserChoice;

/\*\*

\*@brief Точка входа

\* @return 0 в случае успеха

\*/

int main(int argc, char\* argv[])

{

setlocale(LC\_ALL, "Rus");

printf("Введите %d или %d\n", TASK\_A, TASK\_B);

UserChoice choice = int\_input();

switch (choice)

{

case TASK\_A:

{

puts("Введите n");

int n = get\_natural\_int();

double sum = get\_elements\_sum(n);

printf("Сумма первых %d элементов последовательности = %lf", n, sum);

break;

}

case TASK\_B:

{

puts("Введите e");

double e = get\_positive\_double();

printf("Cуммa всех членов последовательности, не меньших заданного числа %lf = %lf", e,

get\_sum\_greater\_equal\_e(e));

break;

}

case default:

{

puts("Неверный ввод");

return 1;

}

}

return 0;

}

double get\_element(const double previous\_element, const int k)

{

return previous\_element \* -1 / (4 \* pow(k, 2) + 2 \* k);

}

double get\_elements\_sum(const int n)

{

double current = -1;

double sum = current;

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

{

sum += get\_element(current, i);

}

return sum;

}

double get\_sum\_greater\_equal\_e(const double e)

{

double current = -1;

double sum = 0;

int k = 1;

while (fabs(current) >= e - DBL\_EPSILON)

{

sum += current;

current = get\_element(current, k);

k++;

}

return sum;

}

int get\_natural\_int(void)

{

const int value = int\_input();

if (value < 1)

{

abort();

}

return value;

}

int int\_input(void)

{

int value = 0;

if (scanf\_s("%d", &value) != 1)

{

abort();

}

return value;

}

double double\_input(void)

{

double value = 0;

if (scanf\_s("%lf", &value) != 1)

{

abort();

}

return value;

}

double get\_positive\_double(void)

{

double value = double\_input();

if (value < 0)

{

abort();

}

return value;

}

3.2.4 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Результаты выполнения программы представлены ниже.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 9 – Результаты выполнения программы

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

Рисунок 10 – Результаты выполнения программы

3.2.5 ВЫПОЛНЕНИЕ ТЕСТОВЫХ ПРИМЕРОВ

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 11 – Тест в Excel

3.3.1 ФОРМУЛИРОВКА ЗАДАНИЯ

Протабулировать заданную функцию и сумму функционального ряда разложения этой функции на интервале [a,b] и с шагом h (шаг и интервал задается в константах). Функциональнй ряд вычисляется по соответствующей рекуррентной формуле с заданной точностью ɛ. В результате показать три столбца: значение аргумента, значение функции в данной точке и значение суммы ряда, вычисленное с заданной точностью в данной точке. Два последних столбца должны иметь близкие результаты.

Таблица 3 – Исходные данные

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вар | Функция y | Сумма | Интервал | ɛ |
| 9 |  |  |  | 20-4 |

3.3.2 БЛОК-СХЕМА АЛГОРИТМА

Изображение выглядит как текст, снимок экрана, черно-белый, дизайн

Автоматически созданное описание

Рисунок 13 – Блок схема main

Изображение выглядит как текст, снимок экрана, Шрифт, черно-белый

Автоматически созданное описание

Рисунок 14 – Блок схема sum\_series

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 15 – Блок схема функций ввода

3.3.3 ТЕКСТ ПРОГРАММЫ НА ЯЗЫКЕ C

#include <float.h>

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

/\*\*

\* @brief Рассчитывает сумму на области определения

\* @param x Значение x

\* @param eps погрешность

\* @return Сумма на области определения x

\*/

long double sum\_series(const double x, const long double eps);

/\*\*

\* @brief Функция ввода вещественного числа

\* @return Вещественное число

\*/

double double\_input(void);

/\*\*

\* @brief Ввод и проверка вещественного числа на положительность

\* @return Вещественное число

\*/

double get\_positive\_double(void);

/\*\*

\* @brief Точка входа в программу

\* @return 0 в случае успеха

\*/

int main(void)

{

puts("Введите a, b, h, eps\n");

const double a = get\_positive\_double();

const double b = get\_positive\_double();

const double h = get\_positive\_double();

const double eps = get\_positive\_double();

printf(" x\t e^(2x)\t\t S(x)\n");

printf("-------------------------------\n");

for (double x = a; x < b + h; x += h)

{

const long double func\_val = expl(2 \* x);

const long double series\_sum = sum\_series(x, eps);

printf("%0.1f\t%0.10Lf\t%0.10Lf\n", x, func\_val, series\_sum);

}

return 0;

}

long double sum\_series(const double x, const long double eps)

{

long double curr = 1.0;

long double sum = 0;

int n = 1;

while (fabsl(curr) >= eps - DBL\_EPSILON)

{

sum += curr;

curr \*= 2 \* x / n;

n++;

}

return sum;

}

double double\_input(void)

{

double value = 0;

if (scanf\_s("%lf", &value) != 1)

{

abort();

}

return value;

}

double get\_positive\_double(void)

{

double value = double\_input();

if (value < 0)

{

abort();

}

return value;

}

3.3.4 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Результаты выполнения программы представлены ниже.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 16 – Результаты выполнения программы

3.3.5 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Тест в Excel

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 17 – Excel

**Approve скриншоты**

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, снимок экрана

Автоматически созданное описание