**Министерство науки и высшего образования РФ**

**ФГБОУ ВО**

**Московский авиационный институт**

**(национальный исследовательский университет)**

Институт №3.

Системы управления, информатика и электроэнергетика.

Кафедра 304.

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ**

**По учебной дисциплине “Программирование на языке высокого уровня”**

Группа: М30-209Б-19:

Выполнил: Кузнецов И.И.

Принял:

Доцент к. 304, к.т.н., Новиков П. В.

Москва 2020

**Содержание**

1. **Задание.**
2. **Теоретическая часть**
3. **Реализация**
4. **Примечание**
5. **Структурная схема алгоритма программы и подпрограмм.**
6. **Текст программы.**
7. **Результаты работы программы.**
8. **Вывод.**
9. **Список литературы.**

**Задание**

При создании программы необходимо освоить основные приёмы процедурного (императивного) программирования:

· создание переменных требуемых типов, в том числе структурированных, присвоение переменным значений,

· вычисление алгебраических и логических выражений,

· ветвление вычислительного процесса,

· циклическое выполнение вычислений,

· разработка и использование процедур и функций,

· форматный и бесформатный ввод и вывод, файловый ввод и вывод.

Результаты вычислений вывести в виде числовых таблиц и в виде двумерных графиков, задав удобные масштабы по осям абсцисс и ординат.

Числовые результаты сохранить в текстовых и двоичных файлах.

Сравнить размеры аналогичных текстовых и двоичных файлов.

**Вариант 6**

Разработать рекурсивные и нерекурсивные функции **e(n)**, с точностью до **10-6** вычисляющие число **e** с помощью **2го** замечательного предела. Вывести график **en**, показывающий, как значение функции **en**, приближается к истинному значению **e** с ростом **n**. Результаты вычислений поместить в таблицу и вывести в выходной файл.

**Теоретическая часть**

Число **е** является иррациональным и приблизительно равно **2.718**. Это число принято за основание логарифмов. Данное число можно вычислить с помощью нескольких способов, но мы будем рассматривать вычисление именно через предел, именуемый как *“второй замечательный предел”.*

Предел последовательности

обозначается буквой **е:**

Число **е** мы можем вычислить итерационно, при некоторой заданной точности вычислений, условием достижения точности может быть два выражения:

**или**

Где **e –** число Эйлера, – текущей член последовательности, – предыдущий член последовательности - заданная точность вычислений

**Реализация**

***Не рекурсивный*** метод подсчета реализован следующий образом:

1.Высчитываем значение:

2. Проверяем выполнено ли условие:

3. Если условие выполнено – завершение программы.

3.1 Иначе возвращаемся к **1му** пункту.

***Рекурсивный метод*** подсчета реализован следующий образом:

1.Высчитываем значение:

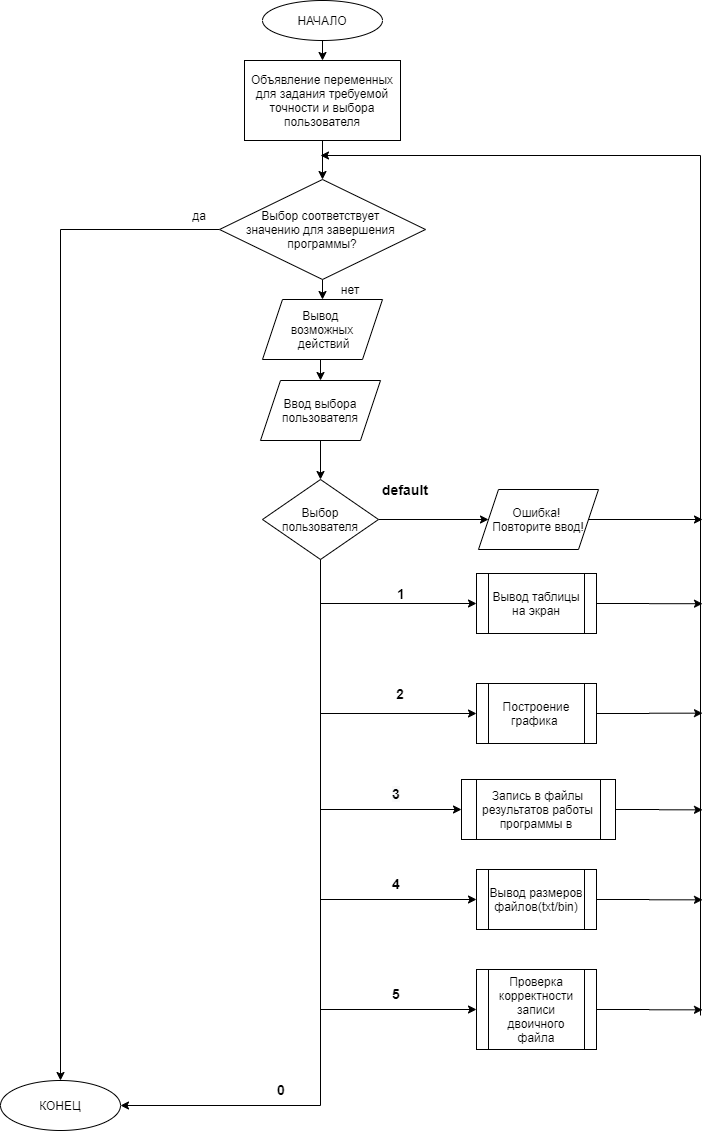
2. Проверяем выполнено ли условие:

3. Если условие выполнено – завершение программы.

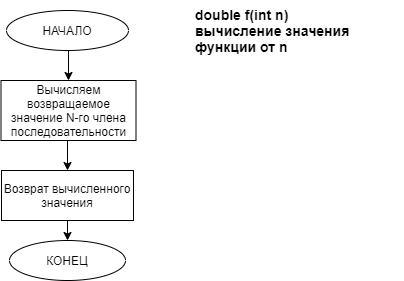
3.1 Иначе высчитываем

и переходим ко **2му** пункту.

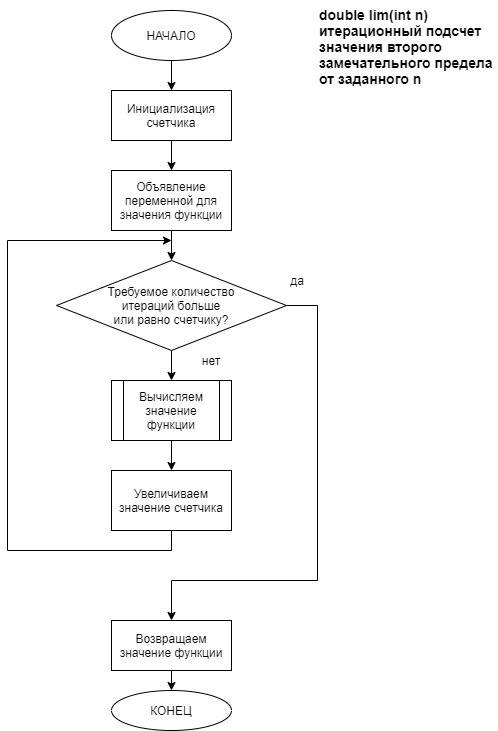
**Структурная схема алгоритма программы.**



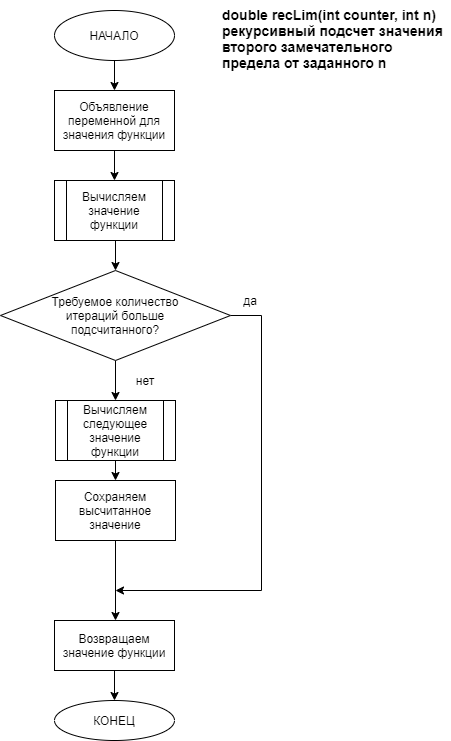
**Структурная схема алгоритма подпрограммы** *f*

****

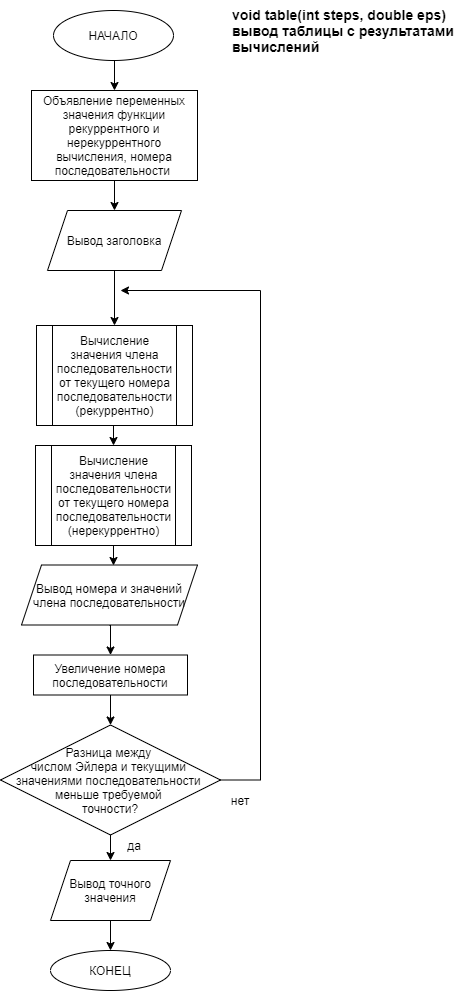
**Структурная схема алгоритма подпрограммы** *lim*

****

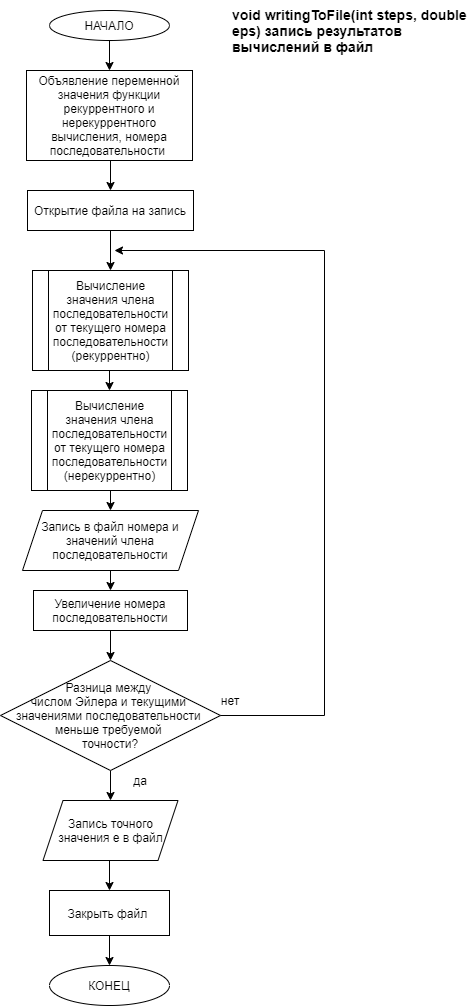
**Структурная схема алгоритма подпрограммы** *recLim*

****

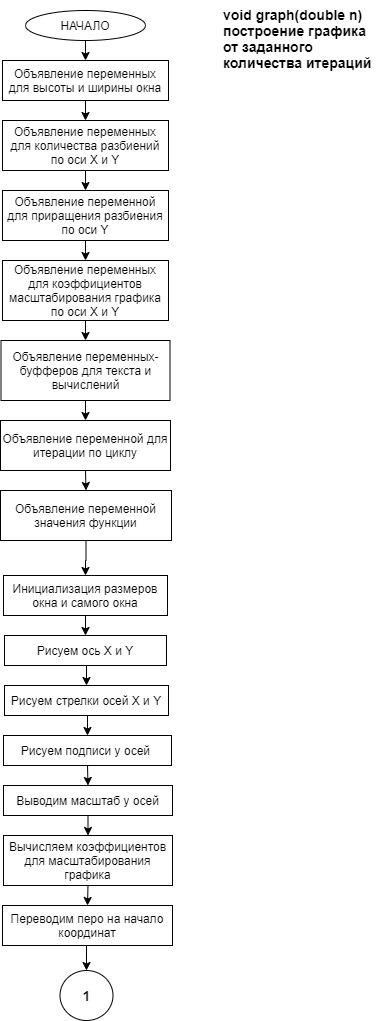
**Структурная схема алгоритма подпрограммы** *table*

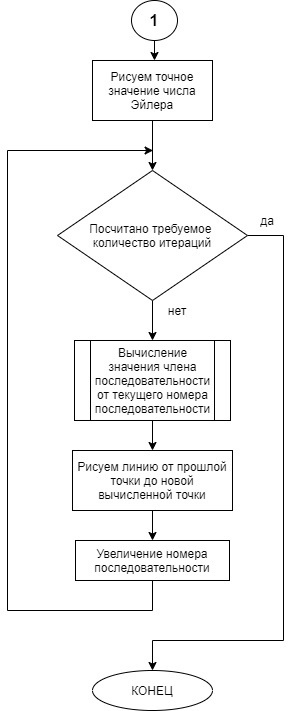


**Структурная схема алгоритма подпрограммы** *writingToFile*

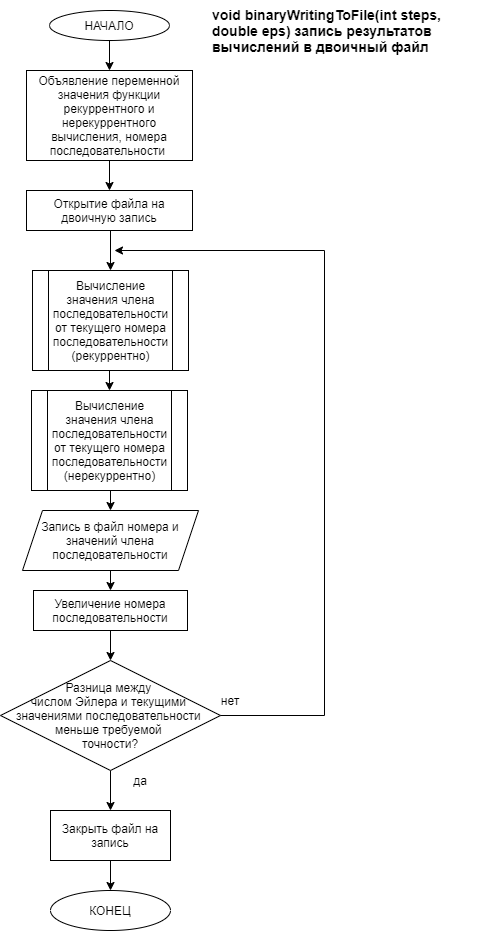


**Структурная схема алгоритма подпрограммы** *graph*

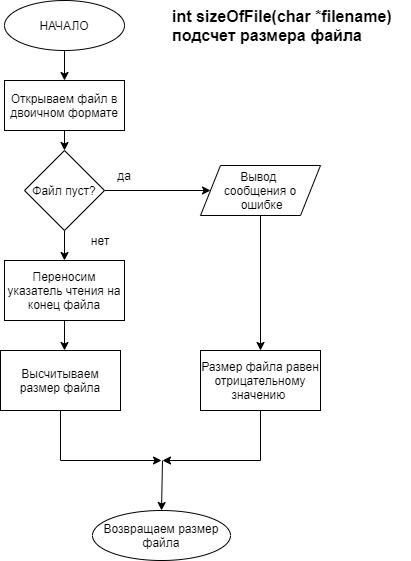


****

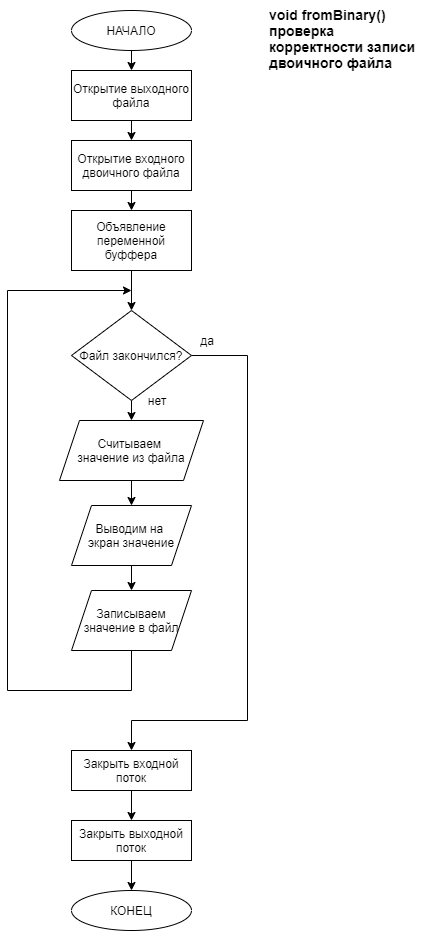
**Структурная схема алгоритма подпрограммы** *binaryWritingToFile*

****

**Структурная схема алгоритма подпрограммы** *sizeOfFile*

****

**Структурная схема алгоитма подпрограммы** *fromBinary*

****

**Текст программы**

#include <math.h>

#include <stdio.h>

#include <graphics.h>

double f(int n) {

return pow(1 + 1.0 / n, n);

}

int countN(double eps) {

int n = 1;

double y;

do {

y = f(n);

n++;

} while (fabs(exp(1) - y) > eps);

return n;

}

double lim(int n) {

int counter = 1;

double y;

while (counter <= n) {

y = f(n);

counter++;

}

return y;

}

double recLim(int counter, int n) {

double y;

y = f(n);

if (counter > n)

y = recLim(counter, n + 1);

return y;

}

void writingToFile(int steps, double eps) {

FILE\* output;

output = fopen("output.txt", "w");

int n = 1; //текущая итерация

double limY; //значение текущего нереккуретного члена последовательности

double recLimY; //значение текущего реккуретного члена последовательности

double maxN;

int coef;

fprintf(output, "%9s %12s %12s\n", "n", "lim", "recLim");

maxN = countN(eps) - 1;

coef = maxN / steps;

do {

if (n == maxN || n % coef == 0 || n == 1) {

limY = lim(n);

recLimY = recLim(n, 1);

fprintf(output, "%9d %12.8f %12.8f\n", n, limY, recLimY);

}

n++;

} while ((fabs(exp(1) - limY) > eps) && (fabs(exp(1) - recLimY) > eps));

fprintf(output, "Exact value e : %12.8f\n", exp(1));

fclose(output);

}

void binaryWritingToFile(int steps, double eps) {

FILE\* output;

output = fopen("bin.bin", "w+b");

double n = 1; //текущая итерация

double limY; //значение текущего нереккуретного члена последовательности

double recLimY; //значение текущего реккуретного члена последовательности

double maxN;

int coef;

maxN = countN(eps) - 1;

coef = maxN / steps;

do {

if (n == maxN || (int)n % coef == 0 || n == 1) {

limY = lim(n);

recLimY = recLim(n, 1);

fwrite(&n, sizeof(double), 1, output);

fwrite(&limY, sizeof(double), 1, output);

fwrite(&recLimY, sizeof(double), 1, output);

}

n++;

} while ((fabs(exp(1) - limY) > eps) && (fabs(exp(1) - recLimY) > eps));

fclose(output);

}

int sizeOfFile(char\* filename) {

FILE\* in;

if ((in = fopen(filename, "rb")) == NULL) {

printf("Error! File is empty!");

return -1;

}

fseek(in, 0, SEEK\_END); //смещение указателя чтения на конец файла

return ftell(in);

}

void table(int steps, double eps) {

int n = 1; //текущая итерация

int maxN;

double limY; //значение текущего нереккуретного члена последовательности

double recLimY; //значение текущего реккуретного члена последовательности

int coef;

maxN = countN(eps) - 1;

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

printf("| Computations require %d steps |\n", maxN);

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

printf("|%6s |%13s |%13s |\n", "n", "lim", "recLim");

// printf("| n | lim | recLim |\n");

coef = maxN / steps;

do {

if (n == maxN || n % coef == 0 || n == 1) {

limY = lim(n);

recLimY = recLim(n, 1);

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

printf("| %5d | %12.8f | %12.8f |\n", n, limY, recLimY);

}

n++;

} while ((fabs(exp(1) - limY) > eps) && (fabs(exp(1) - recLimY) > eps));

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

printf("| eps: %12.8f |\n", eps);

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

printf("| Exact value e: %12.8f |\n", exp(1));

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

}

void graph(double n) {

int height; //высота окна

int width; //ширина окна

int numberSplitsX; //количество разбиений по оси X

int numberSplitsY; //количество разбиений по оси Y

double dy; //приращение для разбиения оси Y

double coefx; //коэффицент масштабирования по оси X

double coefy; //коэффицент масштабирования по оси Y

char temp[4]; //буффер для вывода текста на экран

double buff; //буффер для вычислений

int i; //итератор по циклу

double y; //значение функции

height = 750; //высота окна

width = 750; //ширина окна

initwindow(width, height); //инициализируем размеры окна

line(0, height - 25, width - 10, height - 25); //рисуем ось X

line(25, 10, 25, height); //рисуем ось Y

line(20, 15, 25, 10);

line(30, 15, 25, 10); //стрелка у оси Y

line(width - 15, height - 30, width - 10, height - 25);

line(width - 15, height - 20, width - 10, height - 25); //стрелка у оси X

outtextxy(35, 7, "Y");

outtextxy(width - 17, height - 50, "X"); //подписи у осей X и Y

settextstyle(SMALL\_FONT, HORIZ\_DIR, 0); //стиль текста

//вывод масштаба у оси X

numberSplitsX = 11;

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

buff = n / numberSplitsX \* i;

sprintf(temp, "%1.f", buff);

line((width - 50) / numberSplitsX \* i + 25, height - 20, (width - 50) / numberSplitsX \* i + 25, height - 30);

outtextxy((width - 50) / numberSplitsX \* i + 22, height - 15, temp);

}

//вывод масштаба у оси Y

numberSplitsY = 20;

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

dy = 0.75 / numberSplitsY;

buff = 2.75 - dy \* i;

sprintf(temp, "%1.2f", buff);

outtextxy(3, ((width - 50) / numberSplitsY \* i + 21), temp);

line(20, ((width - 50) / numberSplitsY \* i + 25), 30, ((width - 50) / numberSplitsY \* i + 25));

}

coefx = (width - 25) / n; //вычисляем коэффицент масштабирования по оси X

coefy = (height - 25) / 2.75; //вычисляем коэффицент масштабирования по оси Y

moveto(25, height - 25);//переносим перо в нулевые координаты

setcolor(12);

line(25, (height + 25) - coefy \* exp(1), coefx \* n + 25, (height + 25) - coefy \* exp(1));

settextstyle(0, HORIZ\_DIR, 0); //стиль текста

outtextxy(35, (height + 5) - coefy \* exp(1), "E");

//строим график

setcolor(15);

i = 0;

while (i < n) {

y = f(i);

lineto(coefx \* i + 25, (height + 25) - coefy \* y);

i++;

}

getch();

closegraph();

}

void fromBinary() {

FILE\* output;

FILE\* input;

input = fopen("bin.bin", "rb");

output = fopen("fromBinary.txt", "w");

double check = 0;

int counter = 1;

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

fprintf(output, "%6s %12s %12s\n", "n", "lim", "recLim");

printf("|%6s |%12s |%12s |\n", "n", "lim", "recLim");

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

while (fread(&check, 1, sizeof(double), input) != 0) {

if (counter % 3 == 1) {

printf("|%6.0f |", check);

fprintf(output, "%6.0f ", check);

}

else {

printf("%12.8f |", check);

fprintf(output, "%12.8f ", check);

}

if (counter % 3 == 0) {

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

fprintf(output, "\n");

}

counter++;

}

fclose(output);

fclose(input);

}

int main() {

double eps = 0.0001;

int choose = -1;

int steps = 15;

int buff;

while (choose) {

printf("1.Table\n2.Graph\n3.Write to file\n4.Size of files\n5.Check binary file\n0.Exit\n",

eps);

scanf("%i", &choose);

printf("\n");

switch (choose) {

case 0:

break;

case 1:

table(steps, eps);

break;

case 2:

printf("1.Graph through eps\n2.Graph through steps\n");

scanf("%i", &choose);

switch (choose) {

case 1:

graph(countN(eps));

break;

case 2:

printf("Enter the number of steps: \n");

scanf("%i", &buff);

graph(buff);

break;

}

printf("Graph built! Press any button to close\n\n");

break;

case 3:

writingToFile(steps, eps);

binaryWritingToFile(steps, eps);

printf("The recording is over!\n\n");

break;

case 4:

printf("Size of output file: %d \n", sizeOfFile("output.txt"));

printf("Size of binary file: %d \n\n", sizeOfFile("bin.bin"));

break;

case 5:

fromBinary();

printf("\n");

break;

default:

printf("Error! Repeat please!\n\n");

break;

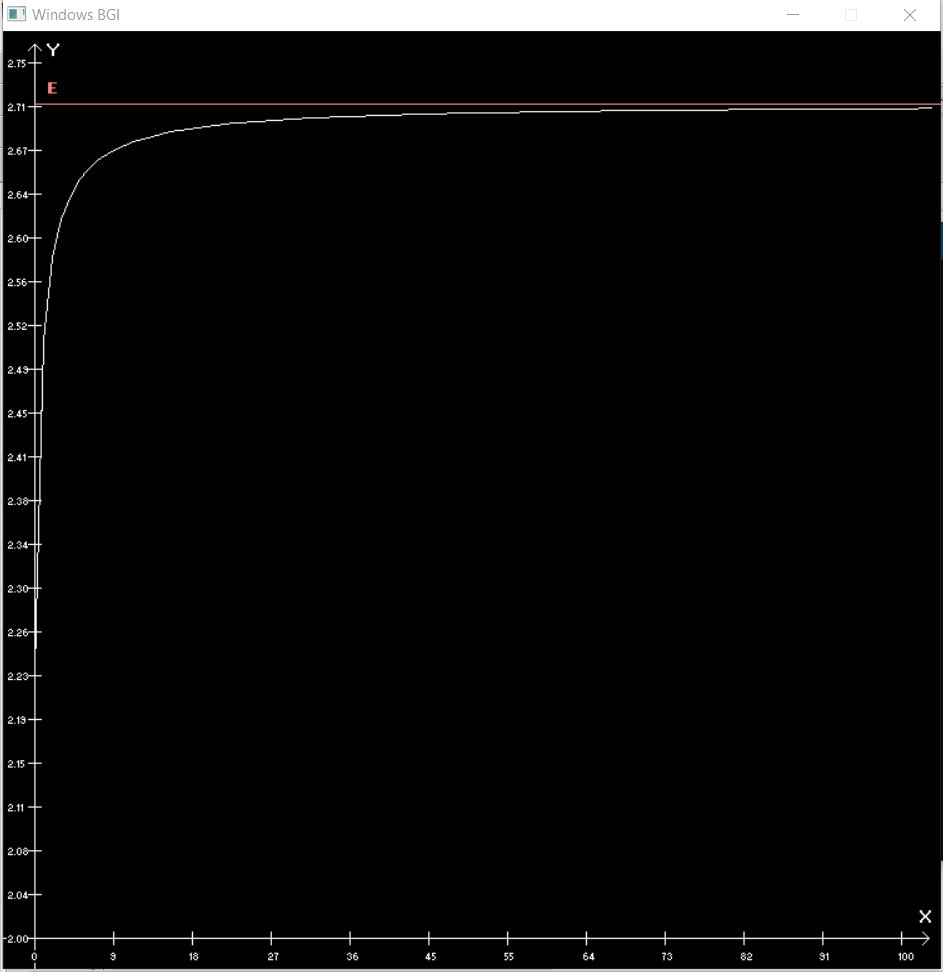
}

}

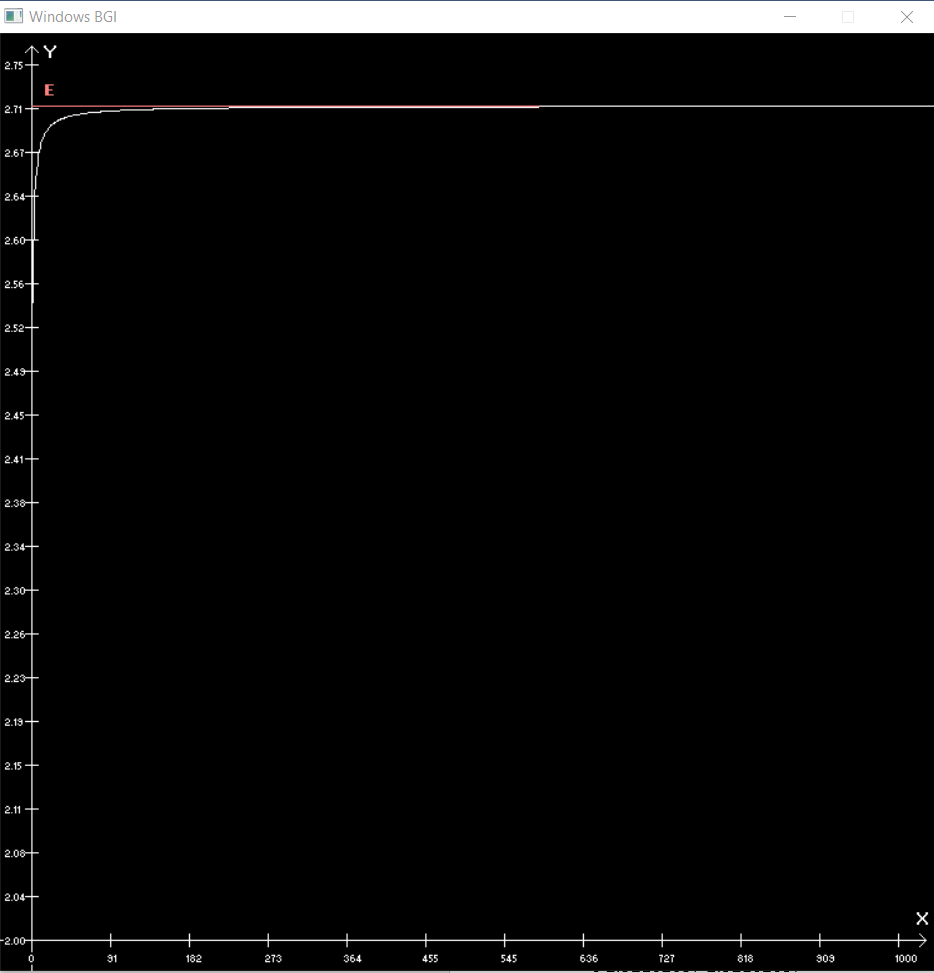
return 0;

}

**Результаты работы программы**

****

*График для первых 100 значений функции* ***en***

**

*График для первых 1000 значений функции* ***en***

*Результаты вычислений:*



n lim recLim

1 2.00000000 2.00000000

339 2.71428337 2.71428337

678 2.71627990 2.71627990

1017 2.71694661 2.71694661

1356 2.71728019 2.71728019

1695 2.71748041 2.71748041

2034 2.71761392 2.71761392

2373 2.71770930 2.71770930

2712 2.71778084 2.71778084

3051 2.71783649 2.71783649

3390 2.71788101 2.71788101

3729 2.71791744 2.71791744

4068 2.71794780 2.71794780

4407 2.71797349 2.71797349

4746 2.71799551 2.71799551

5085 2.71801459 2.71801459

5424 2.71803129 2.71803129

5763 2.71804603 2.71804603

6102 2.71805912 2.71805912

6441 2.71807084 2.71807084

6780 2.71808139 2.71808139

7119 2.71809094 2.71809094

7458 2.71809961 2.71809961

7797 2.71810753 2.71810753

8136 2.71811479 2.71811479

8475 2.71812148 2.71812148

8814 2.71812764 2.71812764

9153 2.71813335 2.71813335

9492 2.71813865 2.71813865

9831 2.71814359 2.71814359

10170 2.71814820 2.71814820

10509 2.71815251 2.71815251

10848 2.71815655 2.71815655

11187 2.71816035 2.71816035

11526 2.71816392 2.71816392

11865 2.71816729 2.71816729

12204 2.71817047 2.71817047

12543 2.71817348 2.71817348

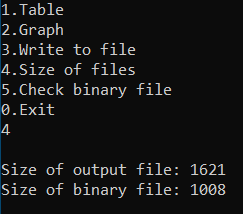
12882 2.71817633 2.71817633

13221 2.71817903 2.71817903

13560 2.71818160 2.71818160

13591 2.71818183 2.71818183

Exact value e : 2.71828183

**

*Сравнение размеров файлов*

**

*Проверка корректности записанных значений в двоичный файл*

**Вывод**

Мы разработали рекурсивную и нерекурсивную функцию вычисляющие e(n) с помощью **2го** замечательного предела с точностью до **10-6,** поработав с циклами и функциями.

Разработали функцию вывода таблицы, в которой предоставлены результаты вычислений, поработав с форматным выводом.

Разработали функцию, которая выводит результаты вычислений в файлы и сравнили размеры этих файлов, также проверив корректность работы записи двоичного файла.

Разработали функцию, которая строит график, показывающий, как значение функции **en**, приближается к истинному значению **e** с ростом **n**, изучив библиотеку **graphics.h.**

**Список литературы**

<http://mycpp.ru/cpp/scpp/cppd_graphics.h.htm> - информация о библиотеки **<graphics.h>**

<https://math.semestr.ru/math/second-limit.php> - информация о втором замечательном пределе