«Колледж информатики и программирования при Финансовом Университете при правительстве Российской Федерации»

**Программы на С++ по предмету**

**«Численные методы в программировании»**

Выполнил студент

Группы 3ПКС-116

Зайцев Н.В.

Москва 2019

**Метод Эйлера и его Модфикационые**

#include "stdafx.h"

#include <iostream>

using namespace std;

double func(double x, double y)

{

return x+2.5\*(y\*y)+2;

}

void stdand(double h, double \*\*mas, double n)

{

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

mas[0][i] = mas[0][0] + i \* h;

mas[1][i] = mas[1][i - 1] + h \* func(mas[0][i - 1], mas[1][i - 1]);

}

};

void Modif1(double h, double \*\*mas, double n)

{

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

mas[0][i] = mas[0][0] + i \* (h / 2);

mas[1][i] = mas[1][i - 1] + (h / 2) \* func(mas[0][i - 1], mas[1][i - 1]);

}

};

void Modif2(double h, double \*\*mas, double n){

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

mas[0][i] = mas[0][0] + i \* (h / 2);

mas[1][i] = mas[1][i - 1] + (h / 2) \* (func(mas[0][i - 1], mas[1][i - 1]) + func(mas[0][i], mas[1][i - 1] + h \* func(mas[0][i - 1], mas[1][i - 1])));

}

}

;

void RungKut(double h, double \*\*mas, double n)

{

double k1, k2, k3, k4;

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

k1 = func(mas[0][i - 1], mas[1][i - 1]);

k2 = func(mas[0][i - 1] + h / 2, mas[1][i - 1] + (h / 2) \* k1);

k3 = func(mas[0][i - 1] + h / 2, mas[1][i - 1] + (h / 2) \* k2);

k4 = func(mas[0][i - 1] + h, mas[1][i - 1] + h \* k3);

mas[0][i] = mas[0][i - 1] + h;

mas[1][i] = mas[1][i - 1] + (h / 6) \* (k1 + 2 \* k2 + 2 \* k3 + k4);

}

};

int main()

{

setlocale(LC\_ALL, "rus");

int s, n = 100;

double\*\* mas = new double \*[2];

for (int i = 0; i < n; i++) mas[i] = new double[i + 1];

double b, h;

bool t = true;

while (t)

{

cout << "0 - Выход\n1 - Ввод значений\n2 - Метод Эйлера\n3 - первый модификационный\n4 - второй модификационный\n5 - метод Рунге Кутта\n";

cin >> s;

switch (s)

{

case 0:

t = false;

break;

case 1:

cout << "a = ";

cin >> mas[0][0];

cout << "b = ";

cin >> b;

cout << "h = ";

cin >> h;

cout << "y0 = ";

cin >> mas[1][0];

cout << endl << endl;

cout << "a = " << mas[0][0] << ", b = " << b << ", h = " << h << ", y0 = " << mas[1][0] << endl << endl;

break;

case 2:

n = (b - mas[0][0]) / h;

stdand(h, mas, n);

cout << "xi" << " yi" << endl;

for (int i = 0; i < n; i++)

{

cout << mas[0][i] << " " << mas[1][i] << endl;

}

break;

case 3:

n = (b - mas[0][0]) / h;

Modif1(h, mas, n);

cout << "xi" << " yi" << endl;

for (int i = 0; i < n; i++)

{

cout << mas[0][i] << " " << mas[1][i] << endl;

}

break;

case 4:

n = (b - mas[0][0]) / h;

Modif2(h, mas, n);

cout << "xi" << " yi" << endl;

for (int i = 0; i < n; i++)

{

cout << mas[0][i] << " " << mas[1][i] << endl;

}

break;

case 5:

n = (b - mas[0][0]) / h;

RungKut(h, mas, n);

cout << "xi" << " yi" << endl;

for (int i = 0; i < n; i++)

{

cout << mas[0][i] << " " << mas[1][i] << endl;

}

break;

default:

break;

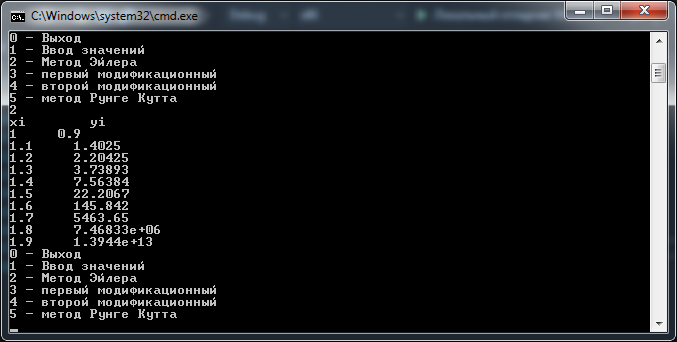
}

}

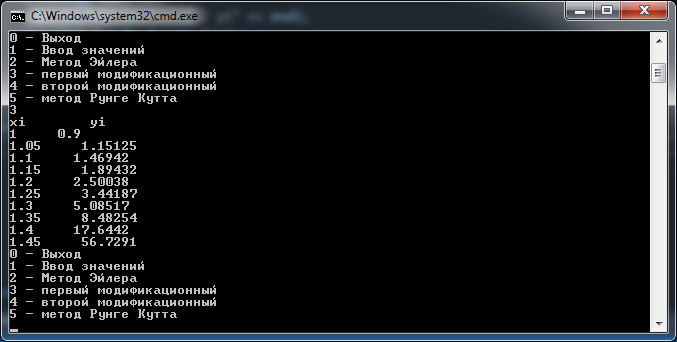
}

Результаты Работ Программы:

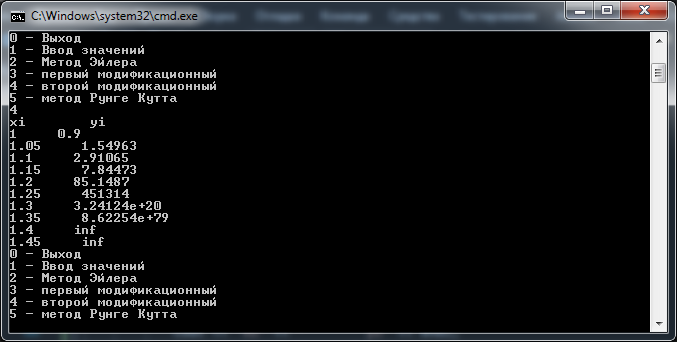
Метод Эйлера:



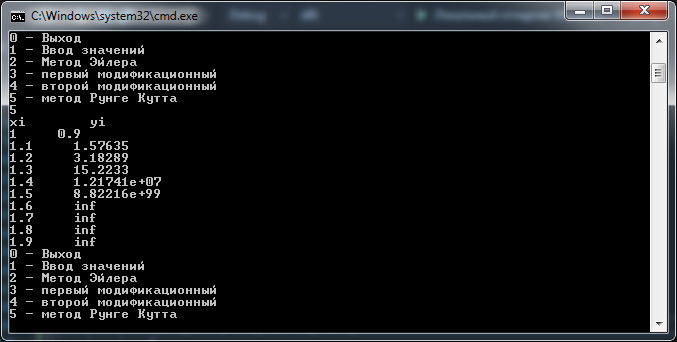
Первый Модифицированный:



Второй Модифицированный:



Метод Рунге-Кутта:



**Метод Гаусса-Жегалкина**

#include <iostream>

using namespace std;

void show(double \*\*a, double \*y, int n)

{

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

cout << a[i][j];

if (j < n - 1)

cout << " ";

}

cout << " " << y[i] << endl;

}

return;

}

double \* method(double \*\*a, double \*y, int n)

{

double \*x, max;

int k, in;

const double аps = 0.00001;

x = new double[n];

k = 0;

while (k < n)

{

max = abs(a[k][k]);

in = k;

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

{

if (abs(a[i][k]) > max)

{

max = abs(a[i][k]);

in = i;

}

}

if (max < аps)

{

cout << "Присутствует нулевой столбец, решение найти не возможно ";

cout << in << "матрицы A" << endl;

return 0;

}

for (int j = 0; j < n; j++)

{

double t = a[k][j];

a[k][j] = a[in][j];

a[in][j] = t;

}

double t = y[k];

y[k] = y[in];

y[in] = t;

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

{

double t = a[i][k];

if (abs(t) < аps) continue;

for (int j = 0; j < n; j++)

a[i][j] = a[i][j] / t;

y[i] = y[i] / t;

if (i == k) continue;

for (int j = 0; j < n; j++)

a[i][j] = a[i][j] - a[k][j];

y[i] = y[i] - y[k];

}

k++;

}

for (k = n - 1; k >= 0; k--)

{

x[k] = y[k];

for (int i = 0; i < k; i++)

y[i] = y[i] - a[i][k] \* x[k];

}

return x;

}

int main()

{

setlocale(0, "");

double \*\*a, \*y, \*x;

int n;

system("cls");

cout << "Введите количество уравнений: ";

cin >> n;

a = new double\*[n];

y = new double[n];

for (int i = 0; i < n; i++)

{

a[i] = new double[n];

for (int j = 0; j < n; j++)

{

cout << "a[" << i << "][" << j << "]= ";

cin >> a[i][j];

}

}

cout << endl;

for (int i = 0; i < n; i++)

{

cout << "b[" << i << "]= ";

cin >> y[i];

}

cout << endl;

show(a, y, n);

x = method(a, y, n);

cout << endl;

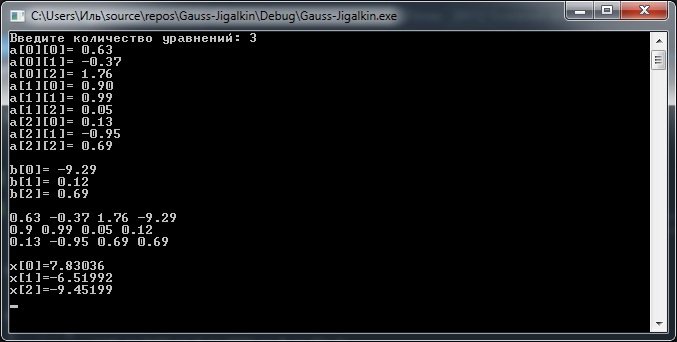
for (int i = 0; i < n; i++)

cout << "x[" << i << "]=" << x[i] << endl;

cin.get(); cin.get();

return 0;

}



**Метод Хорд**

Функция:

Погрешность: E=0,001

Вершины:-10;-11

#include "stdafx.h"

#include <iostream>

#include <math.h>

using namespace std;

double func(double x)

{

return (x\*x\*x\*x - 3 \* x\*x + 75 \* x - 10000);

}

bool MTZ(double a, double b);

void rez(double a, double b, double e);

void main()

{

setlocale(0, "");

double a, b, e;

cout << " Метод хорд" << endl;

cout << " Введите погрешность\n E = ";

cin >> e;

cout << " Введите интервал:\n";

do

{

cout << " a = "; cin >> a;

cout << " b = "; cin >> b;

if (MTZ(a, b) == 0) cout << " Интервал задан неправильно, введите еще раз : \n";

} while (MTZ(a, b) != 1);

rez(a, b, e);

system("pause");

}

bool MTZ(double a, double b)

{

bool k = 0;

if (func(a)\*func(b) > 0) k = 0;

else k = 1;

return k;

}

void rez(double a, double b, double e)

{

double m[1000];

int i = 0;

if (func(a) > 0)

{

m[0] = b;

cout << "\n x[0] = " << m[0] << endl;

do

{

i++;

m[i] = m[i - 1] - (func(m[i - 1]) / (func(m[i - 1]) - func(a)))\*(m[i - 1] - a);

cout << " x[" << i << "] = " << m[i] << endl;

} while (abs(m[i] - m[i - 1]) > e);

}

else

{

m[0] = a;

cout << "\n x[0] = " << m[0] << endl;

do

{

i++;

m[i] = m[i - 1] - (func(m[i - 1]) / (func(b) - func(m[i - 1])))\*(b - m[i - 1]);

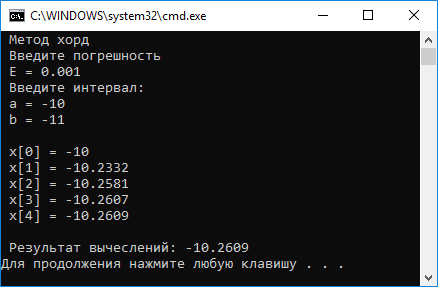
cout << " x[" << i << "] = " << m[i] << endl;

} while (abs(m[i] - m[i - 1]) > e);

}

cout << "\n Результат вычеслений: " << m[i] << endl;

}



**Метод Комбинированный**

Функция:

Погрешность: E=0,001

Вершины:-10;-11

#include "stdafx.h"

#include <iostream>

#include <math.h>

using namespace std;

double func(double x)

{

return (x\*x\*x\*x - 3 \* x\*x + 75 \* x - 10000);

}

double shtrih1(double x)

{

double f;

f = 4 \* pow(x, 3) - 6 \* x + 75;

return f;

}

bool MTZ(double a, double b);

void rez(double a, double b, double e);

void main()

{

setlocale(0, "");

double a, b, e;

cout << " Метод хорд" << endl;

cout << " Введите погрешность\n E = ";

cin >> e;

cout << " Введите интервал:\n";

//do

//{

cout << " a = "; cin >> a;

cout << " b = "; cin >> b;

//if (MTZ(a, b) == 0) cout << " Интервал задан неправильно, введите еще раз : \n";

//} while (MTZ(a, b) != 1);

rez(a, b, e);

system("pause");

}

bool MTZ(double a, double b)

{

bool k = 0;

if (func(a)\*func(b) > 0) k = 0;

else k = 1;

return k;

}

void rez(double a, double b, double e)

{

double m[1000];

double d[1000];

int i = 0;

m[0] = a;

d[0] = b;

if (func(a) < 0 && func(b) > 0) {

cout << "\n x[0] = " << m[0] << endl;

do

{

i++;

m[i] = m[i-1] - func(m[i - 1]) / (func(d[i - 1]) - func(m[i - 1])) \* (d[i - 1] - m[i - 1]);

d[i] = d[i - 1] - func(d[i - 1]) / shtrih1(d[i - 1]);

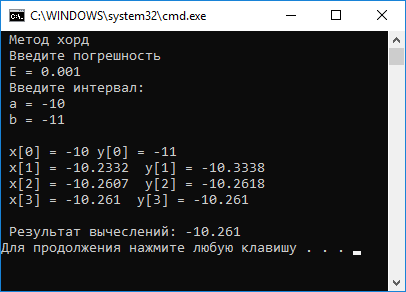
cout << " x[" << i << "] = " << m[i] << " y["<<i<<"] = " <<d[i]<< endl;

} while (abs(d[i] - m[i]) > e);

cout << "\n Результат вычеслений: " << (m[i]+d[i])/2 << endl;

}

}



**Метод Левых Прямоугольников**

#include "stdafx.h"

#include <stdlib.h>

#include <stdio.h>

#include <math.h>

#include <locale.h>

#include <iostream>

using namespace std;

double F(double x) { return x\*x\*sin(x)/10; }

double RightRectangleMethod(double a, int n, double h)

{

double s = 0;

for (int i = 0; i <= n-1; i++)

s += F(a + i \* h) \* h;

cout << "Погрешность метода левых прямоугольников = " << h / 2 \* (F(a + h \* n) - F(a));

return s - h / 2 \* (F(a + h \* n) - F(a));

}

int main()

{

setlocale(0, "");

int n;

double a, b, h;

cout << "Численное интегрирование\n\nИнтервал от ";

cin >> a;

system("cls");

cout << "Численное интегрирование\n\nИнтервал от " << a << " до ";

cin >> b;

cout << "Шагов ";

cin >> n;

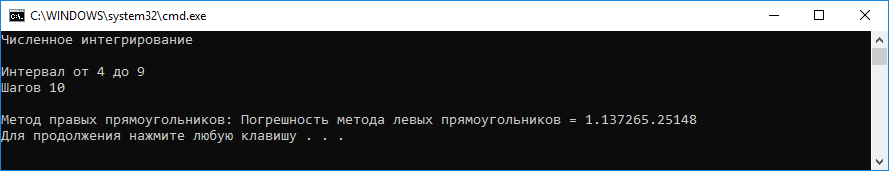
h = (b - a) / n;

cout << "\nМетод правых прямоугольников: " << RightRectangleMethod(a, n, h) << endl;

system("pause");

return 0;

}



**Метод Трапеций**

#include "stdafx.h"

#include <iostream>

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

using namespace std;

void derivative1(double \*&y, double h) {

double f[4];

f[0] = (1 / (6 \* h)) \* (-11 \* y[0] + 18 \* y[1] - 9 \* y[2] + 2 \* y[3]) - (h \* h \* h / 4);

f[1] = (1 / (6 \* h)) \* (-2 \* y[0] - 3 \* y[1] + 6 \* y[2] - y[3]) + (h \* h \* h / 12);

f[2] = (1 / (6 \* h)) \* (y[0] - 6 \* y[1] + 3 \* y[2] + 2 \* y[3]) - (h \* h \* h / 12);

f[3] = (1 / (6 \* h)) \* (-2 \* y[0] + 9 \* y[1] - 18 \* y[2] + 11 \* y[3]) + (h \* h \* h / 4);

y = f;

}

double F(double x) { return x \* x\*sin(x) / 10; }

double MethodOfTrapezium(double a, int n, double h)//методТрапеций

{

double s = 0;

double \*deva, \*devb;

deva = devb = new double[4];

s += F(a) / 2 \* h;

for (int i = 1; i < n; i++) s += F(a + i \* h) \* h;

s += F(a + n \* h) / 2 \* h;

for (int i = 0; i < 4; i++) // заполняем точки для производной

deva[i] = F(a + i \* h / 2);

for (int i = 3; i >= 0; i--)

devb[i] = F(a + n \* h - i \* h / 2);

derivative1(deva, h / 2);

derivative1(devb, h / 2);

cout << "Погрешность метода трапеций = " << -h \* h \* h / 12 << " \* " << (devb[3] - deva[0]) << endl;

return s;

}

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

{

char c;

double a, b;

double step;

setlocale(LC\_ALL, "rus");

cout << "Нижний предел интегрирования ";

cin >> a;

cout << "Высший предел интегрирования ";

cin >> b;

cout << "Шаг ";

cin >> step;

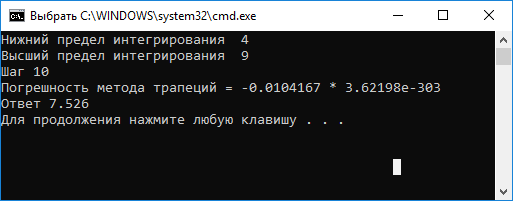
double h = (b - a) / step;

double trap = MethodOfTrapezium(a, step, h);

cout << "Ответ " << trap << endl;

system("pause");

}



**Метод Ньютона-Котеса**

#include "stdafx.h"

#include <iostream>

#include <cmath>

using namespace std;

double F(double x);

double Nuton\_Kotens(double ba, int n, double a, double h);

int main()

{

setlocale(LC\_ALL, "RUS");

int n;

double a, b, h;

cout << "Интервал от -> ";

cin >> a;

cout << "\nИнтервал от " << a << " до -> ";

cin >> b;

cout << "Шагов ";

cin >> n;

h = (b - a) / n;

if (n <= 7)

cout << "\nОтвет: " << Nuton\_Kotens(b - a, n, a, h);

else

cout << "число шагов больше 6!";

cout << "\n";

system("pause");

return 0;

}

double F(double x) {

return x \* x\*sin(x) / 10;;

}

double Nuton\_Kotens(double ba, int n, double a, double h)

{

double C[7], s = 0;

switch (n) {

case 1: {

C[0] = (ba / 2);

C[1] = C[0];

break;

}

case 2: {

C[0] = (ba / 6);

C[1] = (4 \* ba / 6);

C[2] = C[0];

break;

}

case 3: {

C[0] = (ba / 8);

C[1] = (3 \* ba / 8);

C[2] = C[1];

C[3] = C[0];

break;

}

case 4: {

C[0] = (7 \* ba / 90);

C[1] = (16 \* ba / 45);

C[2] = (2 \* ba / 15);

C[3] = C[1];

C[4] = C[0];

break;

}

case 5: {

C[0] = (19 \* ba / 288);

C[1] = (25 \* ba / 96);

C[2] = (25 \* ba / 144);

C[3] = C[2];

C[4] = C[1];

C[5] = C[0];

break;

}

case 6: {

C[0] = (41 \* ba / 840);

C[1] = (9 \* ba / 35);

C[2] = (9 \* ba / 280);

C[3] = (34 \* ba / 105);

C[4] = C[2];

C[5] = C[1];

C[6] = C[0];

break;

}

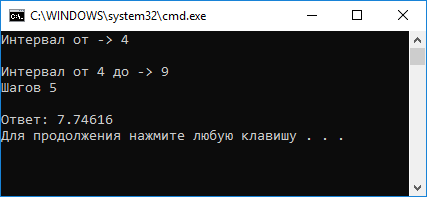
};

for (int i = 0; i <= n; i++)

s += F(a + i \* h) \* C[i];

return s;

}



**Метод Ньютона (метод касательных)**

Функция:

Погрешность: E=0,001

Вершины:-10;-11

#include "stdafx.h"

#include <iostream>

#include<math.h>

using namespace std;

double func(double x)

{

return (x\*x\*x\*x - 3\*x\*x + 75\*x - 10000);

}

double shtrih1(double x)

{

double f;

f = 4 \* pow(x, 3) - 6\*x + 75;

return f;

}

double shtrih2(double x)

{

double f;

f = 12 \* x\*x - 6;

return f;

}

bool MTZ(double a, double b);

void rez(double a, double b, double e);

void main()

{

setlocale(0, "");

double a, b, e;

cout << " Метод Ньютона/Касательной" << endl;

cout << " Введите погрешность\n E = ";

cin >> e;

cout << " Введите интервал:\n";

do

{

cout << "a = "; cin >> a;

cout << "b = "; cin >> b;

if (MTZ(a, b) == 0) cout << " Интервал задан неправильно,введите еще раз:\n";

} while (MTZ(a, b) != 1);

rez(a, b, e);

system("pause");

}

bool MTZ(double a, double b)//MoreThanZero

{

bool k = 0;

if (func(a)\*func(b) > 0) k = 0;

else k = 1;

return k;

}

void rez(double a, double b, double e)

{

double m[1000];

int i = 0;

if (func(a)\*shtrih2(a) < 0) m[0] = b;

else m[0] = a;

cout << "\nx[0] = " << m[0] << endl;

do

{

i++;

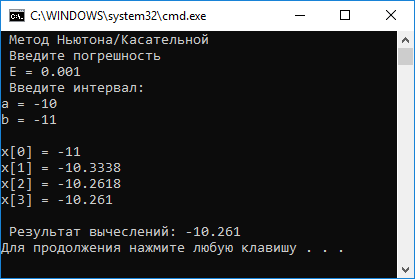
m[i] = m[i - 1] - func(m[i - 1]) / shtrih1(m[i - 1]);

cout << "x[" << i << "] = " << m[i] << endl;

} while (abs(m[i] - m[i - 1]) > e);

cout << "\n Результат вычеслений: " << m[i] << endl;

}



**Метод Правых Прямоугольников**

#include "stdafx.h"

#include <stdlib.h>

#include <stdio.h>

#include <math.h>

#include <locale.h>

#include <iostream>

using namespace std;

double F(double x) { return x\*x\*sin(x)/10; }

double RightRectangleMethod(double a, int n, double h)

{

double s = 0;

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

s += F(a + i \* h) \* h;

cout << "Погрешность метода правых прямоугольников = " << h / 2 \* (F(a + h \* n) - F(a));

return s - h / 2 \* (F(a + h \* n) - F(a));

}

int main()

{

setlocale(0, "");

int n;

double a, b, h;

cout << "Численное интегрирование\n\nИнтервал от ";

cin >> a;

system("cls");

cout << "Численное интегрирование\n\nИнтервал от " << a << " до ";

cin >> b;

cout << "Шагов ";

cin >> n;

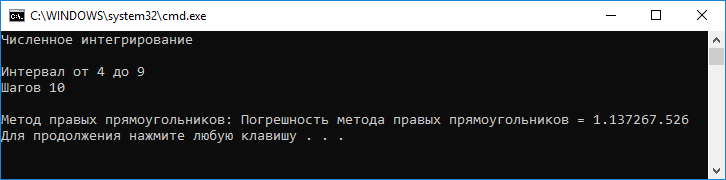
h = (b - a) / n;

cout << "\nМетод правых прямоугольников: " << RightRectangleMethod(a, n, h) << endl;

system("pause");

return 0;

}



**Метод Симпсона**

#include "stdafx.h"

#include <iostream>

#include <cmath>

using namespace std;

double SimpsonFunction(double x);

double MainSimpson(double a, double b, int n);

int main()

{

setlocale(0, "");

double y;

double raz1, y1;

cout << "Метод Симсона:\n";

double a, b;

double step;

cout << "Нижний предел интегрирования ";

cin >> a;

cout << "Высший предел интегрирования ";

cin >> b;

cout << "Шаг ";

cin >> step;

y = MainSimpson(a, b, step);

y1 = MainSimpson(a, b, step - 1);

raz1 = abs(y - y1);

cout << "Ответ " << y << endl;

cout << "Ошибка " << raz1 << endl;

system("pause");

return 0;

}

double SimpsonFunction(double x)

{

return x\*x\*sin(x)/10;

}

double MainSimpson(double a, double b, int n)

{

double h, s, x, f;

int i = 1;

h = (b - a) / n;

x = a;

f = SimpsonFunction(x);

s = f;

while (i <= n) {

x = x + h;

f = SimpsonFunction(x);

s = s + 4 \* f;

i = i + 2;

x = x + h;

f = SimpsonFunction(x);

s = s + 2 \* f;

}

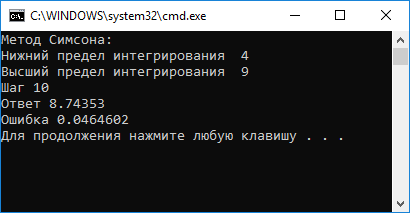
x = b;

f = SimpsonFunction(x);

s = (s + f) \* (h / 3);

return s;

}



**Метод Средних прямоугольников**

#include "stdafx.h"

#include <iostream>

#include <iomanip>

#include <math.h>

#include <cmath>

using namespace std;

class MethodOfMiddleRectangles

{

double F(double x) { return x \* x\*sin(x) / 10; }

double methodOfMiddleRectangles(double a, int n, double h)

{

double s = 0;

for (int i = 0; i < n; i++) s += F(a + i \* h + h / 2) \* h;

return s;

}

public:

void calculate() {

int n;

double a, b, h;

cout << "Численное интегрирование\n\nИнтервал от "; cin >> a;

system("cls");

cout << "Численное интегрирование\n\nИнтервал от " << a << " до "; cin >> b;

cout << "Шагов "; cin >> n;

cout << endl;

h = (b - a) / n;

cout << "Метод средних прямоугольников: " << methodOfMiddleRectangles(a, n, h) << endl;

cout << endl;

}

};

int main()

{

setlocale(0, "");

MethodOfMiddleRectangles Middle\_Rec;

system("cls");

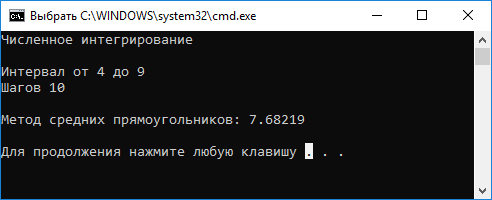
Middle\_Rec.calculate();

system("pause");

system("cls");

return 0;

}



**Метод Зейделя**

#include "stdafx.h"

#include <stdlib.h>

#include <stdio.h>

#include <math.h>

#include <locale.h>

#include <iostream>

using namespace std;

void change(double\*\* nc, int i1, int i2, int n);

int main()

{

setlocale(0, "");

double \*\*data, k, help, \*helper, \*\*miss, \*mishelp, copy, E, \*h2;

int i, j, l, \*hlp, n, counter;

printf("Введите порядок системы: ");

cin >> n;

data = (double\*\*)malloc(n \* sizeof(double\*));

miss = (double\*\*)malloc(n \* sizeof(double\*));

hlp = (int\*)malloc(n \* sizeof(int));

helper = (double\*)malloc(n \* sizeof(int));

h2 = (double\*)malloc(n \* sizeof(int));

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

data[i] = (double\*)malloc((n + 1) \* sizeof(double));

miss[i] = (double\*)malloc((n + 1) \* sizeof(double));

};

printf("Вводите элементы расширенной матрицы: \n");

for (i = 0; i < n; i++)

for (j = 0; j < n + 1; j++) {

printf("Введите элемент %d-%d: ", i + 1, j + 1);

cin >> data[i][j];

miss[i][j] = 1;

copy = data[i][j];

while (floor(copy) != ceil(copy)) {

copy \*= 10;

miss[i][j] /= 10;

//printf("+");

}

if ((data[i][j] - floor(data[i][j]) >= 0.001) && (data[i][j] - floor(data[i][j]) <= 0.009))

miss[i][j] = 0.001;

if (miss[i][j] == 1)

miss[i][j] = 0;

}

printf("\nВаша расширенная матрица: ");

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

printf("\n");

for (j = 0; j < n + 1; j++)

printf("%5.5lf ", data[i][j]);

}

printf("\n\nПогрешности элементов: ");

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

printf("\n");

for (j = 0; j < n + 1; j++)

printf("%lf ", miss[i][j]);

}

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

hlp[i] = 0;

help = fabs(data[i][0]);

copy = help;

for (j = 1; j < n; j++) {

copy = copy + fabs(data[i][j]);

if (fabs(data[i][j]) > help) {

help = fabs(data[i][j]);

hlp[i] = j;

}

}

if (fabs(copy - fabs(help)) > fabs(help)) {

cout << "\n\nНельзя решить методом Зейделя!";

system("pause");

exit(0);

}

}

for (i = 0; i < n; i++)

printf("%d", hlp[i]);

for (i = 0; i < n; i++)

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

if (hlp[j] == hlp[i]) {

printf("\n\nНельзя решить методом Зейделя\n ");

system("pause");

exit(0);

}

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

for (j = i; j < n; j++)

if (hlp[j] == i) {

change(data, i, j, n);

change(miss, i, j, n);

help = hlp[i];

hlp[i] = hlp[j];

hlp[j] = help;

break;

}

}

printf("\nУравнения в правильном порядке: ");

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

printf("\n");

for (j = 0; j < n + 1; j++)

printf("%5.5lf ", data[i][j]);

}

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

data[i][n] = data[i][n] / data[i][i];

miss[i][n] = (fabs(data[i][n]) \* miss[i][i] + fabs(data[i][i]) \* miss[i][n]) / (data[i][i] \* data[i][i]);

for (j = n - 1; j >= 0; j--)

{

if (j != i) {

miss[i][j] = (fabs(data[i][j]) \* miss[i][i] + fabs(data[i][i]) \* miss[i][j]) / (data[i][i] \* data[i][i]);

data[i][j] = -data[i][j] / data[i][i];

}

}

data[i][i] = data[i][n];

miss[i][i] = miss[i][n];

}

printf("\n\nВведите E: ");

cin >> E;

do {

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

helper[i] = data[i][i];

h2[i] = helper[i];

data[i][i] = data[i][n]; //текущий х1, х2 и т.д хранятся на главной диагонали

miss[i][i] = miss[i][n];

}

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

for (j = 0; j < n; j++) {

if (j != i) {

data[i][i] += data[i][j] \* helper[j];

miss[i][i] = miss[i][i] + fabs(data[i][j]) \* miss[j][j] + fabs(helper[j]) \* miss[i][j];

}

}

helper[i] = data[i][i];

}

help = 0;

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

if (fabs(fabs(h2[i]) - fabs(data[i][i])) > E) {

help = 1;

break;

}

}

printf("\nВаша расширенная матрица: ");

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

printf("\n");

for (j = 0; j < n + 1; j++)

printf("%5.5lf ", data[i][j]);

}

} while (help);

printf("\n\nКорни уравнения:");

for (i = 0; i < n; i++)

// printf("\nx%d = %lf (погрешность %lf)", i, data[i][i], miss[i][i]);

printf("\nx%d = %lf", i, data[i][i]);

return 0;

}

void change(double\*\* nc, int i1, int i2, int n)

{

int j;

double buf;

for (j = 0; j < n + 1; j++) {

buf = nc[i1][j];

nc[i1][j] = nc[i2][j];

nc[i2][j] = buf;

}

}

