ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ им. Р.Е.Алексеева

ПРАКТИКУМ ПО ЧИСЛЕННЫМ МЕТОДАМ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА ПРОГРАММИРОВАНИЯ С++ К ЛАБОРАТОРНЫМ РАБОТАМ ПО КУРСУ « ИНФОРМАТИКА»

Методическая разработка
для студентов дневной, вечерней и заочной формы обучения
для всех специальностей

Нижний Новгород 2009

Составители: Т.В. Моругина, С.П. Никитенкова, О.И. Чайкина

УДК 651.3.06

Практикум по численным методам с использованием языка программирования С++ к лабораторным работам по курсу «Информатика»: Методическая разработка для студентов дневной, вечерней и заочной формы обучения для всех специальностей/ НГТУ; сост.: Т.В. Моругина, С.П. Никитенкова, О.И. Чайкина. Н.Новгород, 2009, 23 с.

Приведены примеры программной реализации численных методов решения на ЭВМ различных классов математических задач. Методическая разработка может быть использована при подготовке к лабораторным работам по курсу «Информатика»(язык программирования С/С++).

Научный редактор С.Н. Митяков
Редактор Э.Г

681
Ч 69 Бр

Практикум по
м методам с
м языка
С++ к

Нижегородск
Типо

еева.

ический

Лабораторная работа №1
Решение нелинейного уравнения с одной неизвестной. Методы отделения и уточнения корней

Постановка задачи. Для данного нелинейного уравнения **y(x)=0** отделить корни с шагом **h** на промежутке **[a,b]** и уточнить корень с точностью **ε** :

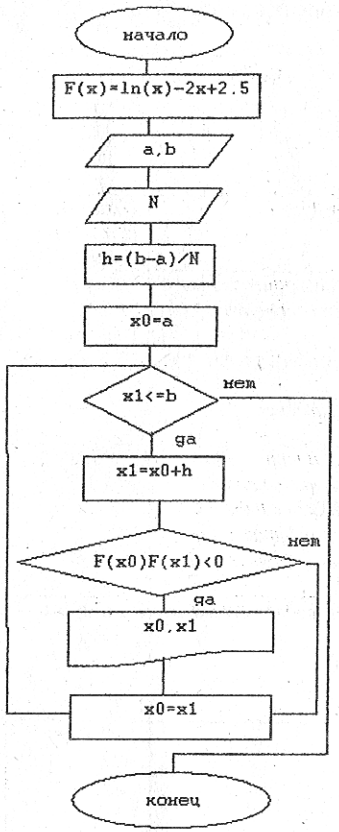
- методом половинного деления,
- методом Ньютона,
- методом простой итерации.

Идея метода:

Название метода	Выбор начального значения	Итерационная формула	Окончание процесса вычисления
Шаговый метод	x=a	y=f(x) – *значение функции в точке x* x1=x+h – *следующее значение переменной x* y1=f(x1) - *значение функции в точке x1* y·y1<0 - *признак интервала изоляции*	x1<=b
Метод половинного деления (бисекции)	[a,b] – *интервал изоляции*	x=(a+b)/2 – *середина интервала* f(a) –*значение функции в точке a* f(x) –*значение функции в точке x* *если* f(a)·f(x)<0, *то выбираем* [a,x] *если* f(a)·f(x)>0, *то выбираем* [x,b]	$\lvert f(x)\rvert<\varepsilon$
Метод Ньютона	x_0 = a *или* x_0 = b f2(x)-*вторая производная функции* f(x) $f(x_0)*f2(x_0)>0$	f1(x) – *первая производная функции* f(x) $x_{i+1} = x_i - f(x_i)/f1(x_i)$	$\lvert f(x_i)\rvert<\varepsilon$
Метод простой итерации (1-й способ)	*привести уравнение к виду* x= φ(x) x_0 = a *или* x_0 = b |φ'(a)|<1 |φ'(b)|<1 *если* |φ(a)| > |φ(b)|,*то* x_0=a *если* |φ(a)| < |φ(b)|,*то* x_0=b	x_{i+1}= φ(x_i)	$\lvert f(x_i)\rvert<\varepsilon$
Метод простой итерации (2-й способ)	f1(x) – *первая производная функции* f(x) *если* |f1(a)| > |f1(b)| ,*то* x_0=a *если* |f1(a)| < |f1(b)| ,*то* x_0=b	c=1/max(|f1(a)|;|f1(b)|) $x_{i+1} = x_i - c \cdot f(x_i)$	$\lvert f(x_i)\rvert<\varepsilon$

Шаговый метод

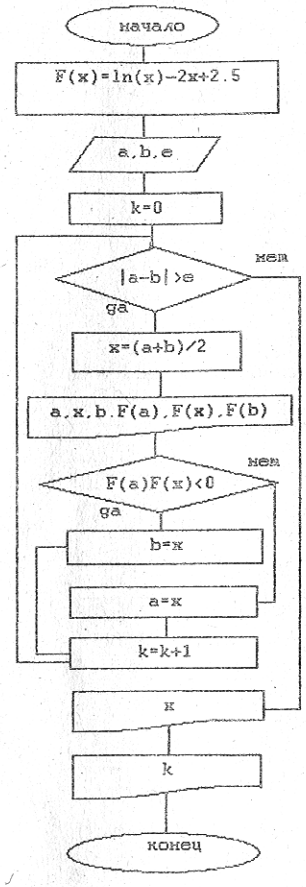
Постановка задачи: шаговым методом найти интервал изоляции корня нелинейного уравнения $ln(x) - 2x + 2{,}5 = 0$ на интервале [1; 2].




```
#include <iostream.h>
#include <iomanip.h>
#include <math.h>

double F (double x)
{ return (log(x)-2*x+2.5);}

 int main()
{ double a,b,x0,x1,h;
  int N;

  cout<<"Введите a и b"<<endl;
  cin>>a>>b;
  cout<<"Введите число разбиений"<<endl;
  cin>>N;

  h=(b-a)/N;//вычисляем шаг

  cout.precision(5);
  cout.setf(ios::left);

cout<<"_____"<<endl;
cout<<setw(12)<<"x"<<setw(12)<<"F(x)"<<endl;
cout<<"_____"<<endl;

  x0=a;
  cout<<setw(12)<<x0<<setw(12)<<F(x0)<<endl;

  x1=x0+h;
  while(x1<=b)
{
   x1=x0+h;
   cout<<setw(12)<<x1<<setw(12)<<F(x1);

   if(F(x0)*F(x1)<0)
   cout<<"на интервале  ["<<
            x0<<","<<x1<<"] корень";
   x0=x1;
  cout<<endl;
 }
 return 0;}
```


Метод половинного деления

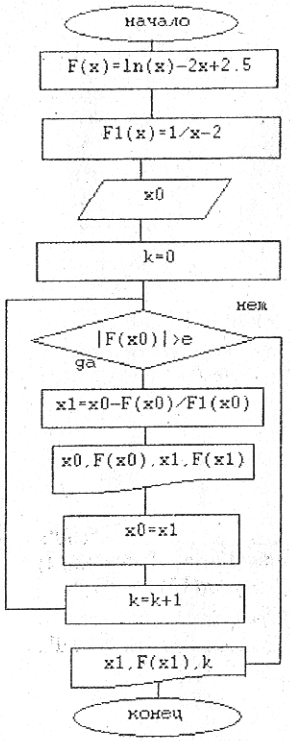
Постановка задачи: найти корень нелинейного уравнения $ln(x) - 2x + 2{,}5 = 0$ на отрезке [1; 2] методом половинного деления с точностью e = 0,001.




```
#include <iostream.h>
#include <iomanip.h>
#include <math.h>

double F (double x)
{ return (log(x)-2*x+2.5);}

int main()
{ double a,b,x,eps;
  int k=0;

  cout<<"Введите a и b"<<endl;
  cin>>a>>b;
  cout<<"Введите точность"<<endl;
  cin>>eps;

  cout.precision(5);
  cout.setf(ios::left);

cout<<"_____"<<endl;
cout<<setw(12)<<"a"<<setw(12)<<"x"<<setw(12)
<<"b"<<setw(12)<<"F(a)"<<setw(12)<<"F(x)"
<<setw(12)<<"F(b)"<<endl;
cout<<"_____"<<endl;

while(fabs(a-b)>eps)
{x=(a+b)/2;

cout<<setw(12)<<a<<setw(12)<<x<<setw(12)<<b
 <<setw(12)<<F(a)<<setw(12)<<F(x)<<setw(12)<<
F(b)<<endl;

if(F(a)*F(x)<0)   b=x; else  a=x;

k++;
}

cout<<endl<<"корень="<<x<<"  F("<<x<<")="<<F(x);
cout<<endl<<"max число итераций"<<k;

return 0;}
```

Метод Ньютона

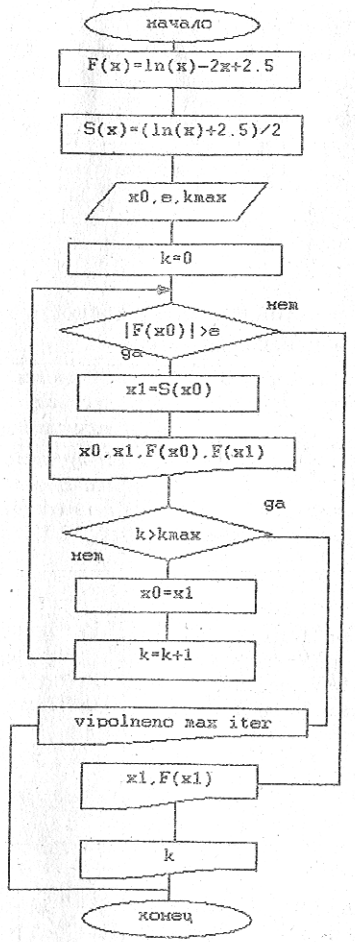
Постановка задачи: найти корень нелинейного уравнения $\ln(x) - 2x + 2{,}5 = 0$ на отрезке [1; 2] методом Ньютона с точностью e = 0,001.




```
#include <iostream.h>
#include <iomanip.h>
#include <math.h>

double F (double x)
{ return (log(x)-2*x+2.5);}

double F1 (double x)
{ return ( 1/x-2 );}

int main()
{ double x0,x1,eps;
  int k=0;

  cout<<"Введите начальное значение x"<<endl;
  cin>>x0;
  cout<<"Введите точность"<<endl;
  cin>>eps;

  cout.precision(5);
  cout.setf(ios::left);

  cout<<"_____"<<endl;
  cout<<setw(12)<<"x0"<<setw(12)<<"F(x0)"<<
        setw(12)<<"x1"<<setw(12)<<"F(x1)"<<endl;
  cout<<"_____"<<endl;

  while(fabs(F(x0))>eps)
  {
    x1=x0-F(x0)/F1(x0);

    cout<<setw(12)<<x0<<setw(12)<<F(x0)<<
          setw(12) <<x1<<setw(12)<<F(x1)<<endl;

    x0=x1;
    k++;
  }
cout<<endl<<"корень="<<x1<<
                "F("<<x1<<")="<<F(x1);
cout<<endl<<"выполнено итераций"<<k;
return 0;
}
```


Метод простой итерации

Постановка задачи: найти корень нелинейного уравнения $ln(x) - 2x + 2{,}5 = 0$ на отрезке [1; 2] методом Ньютона с точностью e = 0,001.




```
#include <iostream.h>
#include <iomanip.h>
#include <math.h>

double F (double x)
{ return (log(x)-2*x+2.5);}
double S (double x)
{ return ( (log(x)+2.5/2 );}

int main()
{ double x0,x1,eps;    int k,kmax;
  cout<<"Введите начальное значение"<<endl;
  cin>>x0;
  cout<<"Введите точность"<<endl;
  cin>>eps;
  cout<<"Введите max число итераций"<<endl;
  cin>>kmax;

  cout.precision(5);
  cout.setf(ios::left);
  cout<<"_____"<<endl;
  cout<<setw(12)<<"x0"<<setw(12)<<"F(x0)"
  <<setw(12)<<"x1"<<setw(12)<<"F(x1)"<<endl;
  cout<<"_____"<<endl;

  k=0;
  while(fabs(F(x0))>eps)
  {
  x1=S(x0);
cout<<setw(12)<<x0<<setw(12)<<F(x0)<<setw(12)
<<x1<<setw(12)<<F(x1)<<endl;
if(k>kmax) {cout<<"выполнено max  итераций";
            break;}
  x0=x1;
  k++;
  }
cout<<"корень="<<x1<<"F("<<x1<<")="<<F(x1);
cout<<endl<<"число итераций"<<k;
return 0;}
```

Лабораторная работа №2
Решение систем линейных уравнений
Прямые и итерационные методы

Постановка задачи: Дана система линейных уравнений

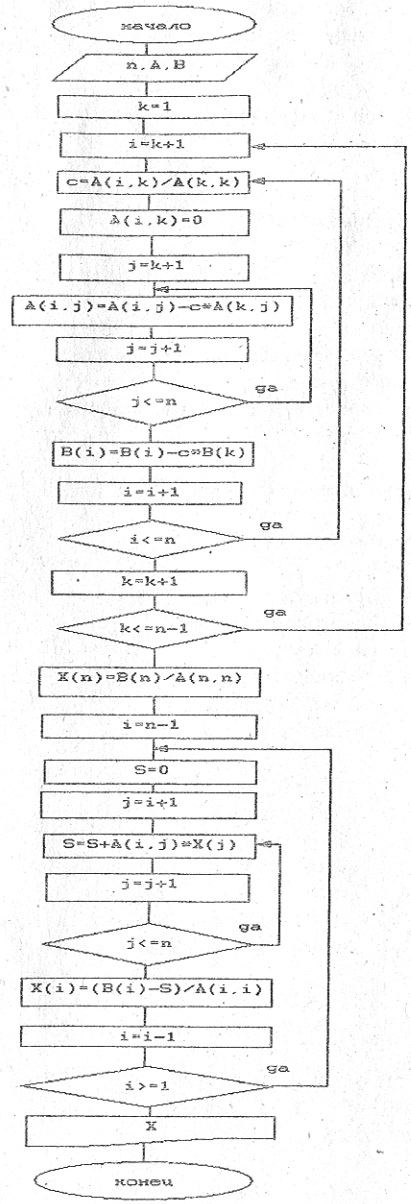
$$\begin{cases} A_{11}x1 + A_{12}x2 + A_{13}x3 + A_{14}x4 = B_1, \\ A_{21}x1 + A_{22}x2 + A_{23}x3 + A_{24}x4 = B_2, \\ A_{31}x1 + A_{32}x2 + A_{33}x3 + A_{34}x4 = B_3, \\ A_{41}x1 + A_{42}x2 + A_{43}x3 + A_{44}x4 = B_4. \end{cases}$$

Запись системы линейных уравнений в матричном виде

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix} \begin{bmatrix} X1 \\ X2 \\ X3 \\ X4 \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \end{bmatrix}$$

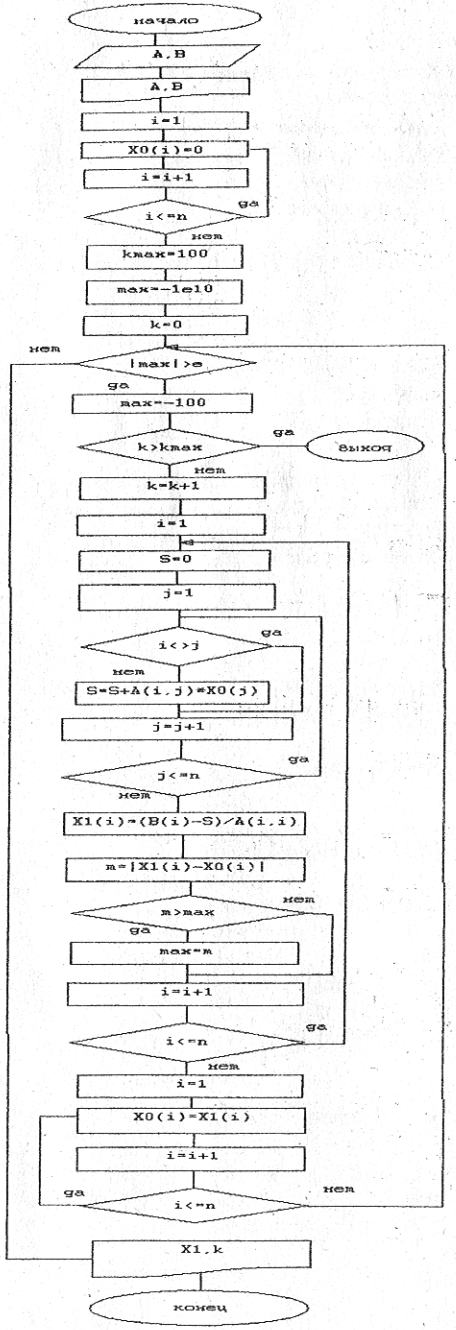
- найти точное решение методом Гаусса,
- найти приближённое решение методом простой итерации с точностью ε,
- найти приближённое решение методом Зейделя с точностью ε.

Метод	Начальное приближение	Итерационная формула	Остановка процесса вычисления
Метод Гаусса	*Определитель матрицы не равен нулю*	*Прямой ход – приведение матрицы к треугольному виду.* *Обратный ход – вычисление неизвестных, начиная с последнего уравнения*	*Получение значений неизвестных* $X1, X2, X3, X4$
Метод простой итерации	*Проверка условия сходимости* $\|A_{11}\|>\|A_{12}\|+\|A_{13}\|+\|A_{14}\|$ $\|A_{22}\|>\|A_{21}\|+\|A_{23}\|+\|A_{24}\|$ $\|A_{33}\|>\|A_{31}\|+\|A_{32}\|+\|A_{34}\|$ $\|A_{44}\|>\|A_{41}\|+\|A_{42}\|+\|A_{43}\|$ *Выбор начального приближения* $X1^0=0, X2^0=0,$ $X3^0=0, X4^0=0$	$X1^{i+1}=(B_1-(A_{12}X2^i+A_{13}X3^i+A_{14}X4^i))/A_{11}$ $X2^{i+1}=(B_2-(A_{21}X1^i+A_{23}X3^i+A_{24}X4^i))/A_{22}$ $X3^{i+1}=(B_3-(A_{31}X1^i+A_{32}X2^i+A_{34}X4^i))/A_{33}$ $X4^{i+1}=(B_4-(A_{41}X1^i+A_{42}X2^i+A_{43}X3^i))/A_{44}$	$\|X1^{i+1}-X1^i\|<\varepsilon$ $\|X2^{i+1}-X2^i\|<\varepsilon$ $\|X3^{i+1}-X3^i\|<\varepsilon$ $\|X4^{i+1}-X4^i\|<\varepsilon$
Метод Зейделя	*Проверка условия сходимости* $\|A_{11}\|>\|A_{12}\|+\|A_{13}\|+\|A_{14}\|$ $\|A_{22}\|>\|A_{21}\|+\|A_{23}\|+\|A_{24}\|$ $\|A_{33}\|>\|A_{31}\|+\|A_{32}\|+\|A_{34}\|$ $\|A_{44}\|>\|A_{41}\|+\|A_{42}\|+\|A_{43}\|$ *Выбор начального приближения* $X1^0=0, X2^0=0,$ $X3^0=0, X4^0=0$	$X1^{i+1}=(B_1-(A_{12}X2^i+A_{13}X3^i+A_{14}X4^i))/A_{11}$ $X2^{i+1}=(B_2-(A_{21}X1^{i+1}+A_{23}X3^i+A_{24}X4^i))/A_{22}$ $X3^{i+1}=(B_3-(A_{31}X1^{i+1}+A_{32}X2^{i+1}+A_{34}X4^i))/A_{33}$ $X4^{i+1}=(B_4-(A_{41}X1^{i+1}+A_{42}X2^{i+1}+A_{43}X3^{i+1}))/A_{44}$	$\|X1^{i+1}-X1^i\|<\varepsilon$ $\|X2^{i+1}-X2^i\|<\varepsilon$ $\|X3^{i+1}-X3^i\|<\varepsilon$ $\|X4^{i+1}-X4^i\|<\varepsilon$



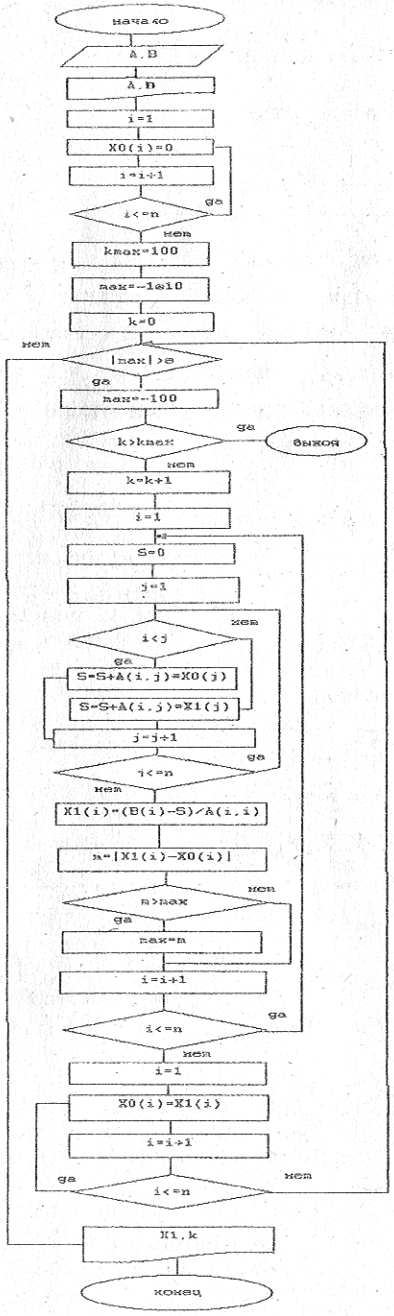
Метод Гаусса

```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
int main()
{ double a[4][4]={{5,1,1,1},
                  {1,7,1,1},
                  {1,1,6,1},
                  {1,1,1,4}};
double b[4]={8,10,9,7};
double x[4];
int n,k,i,j;

n=4;
double c,s;
for(k=0;k<n-1;k++)
for(i=k+1;i<n;i++)
{
c=a[i][k]/a[k][k];
a[i][k]=0;

for(j=k+1;j<n;j++)
a[i][j]=a[i][j]-c*a[k][j];

b[i]=b[i]-c*b[k];
}

x[n-1]=b[n-1]/a[n-1][n-1];

for (i=n-1;i>=0;i--)
{
s=0;
for(j=i+1;j<n;j++)
{s=s+a[i][j]*x[j];
x[i]=(b[i]-s)/a[i][i];
}
}
cout<<"Решение"<<endl;
for (i=0;i<n;i++)
cout<<x[i];
return 0;}
```



Метод простой итерации

```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
int main()
{ double A[4][4]={{5,1,1,1},
                  {1,7,1,1},
                  {1,1,6,1},
                  {1,1,1,4}};
double B[4]={8,10,9,7};
double X0[4],X1[4];
double eps,tochnost,max_tochnost,s;
int k,kmax,i,j;

cout<<"Введите точность"<<endl;
cin>>eps;
cout<<"Введите max итераций"<<endl;
cin>>kmax;

cout.precision(5); cout.setf(ios::left);

for(i=0;i<4;i++) X0[i]=0;
k=0;
while(fabs(max_tochnost)>eps)
{       if(k>=kmax) break;
        k=k+1;
        cout<<setw(5)<<k;
         max_tochnost=-100;
        for(i=0;i<4;i++)
        {s=0;
        for(j=0;j<4;j++)
        if(i!=j) s=s+A[i][j]*X0[j];
        X1[i]=(B[i]-s)/A[i][i];
        cout<<setw(10)<<X1[i];
        tochnost=fabs(X1[i]-X0[i]);
        if(tochnost>max_tochnost)
         max_tochnost=tochnost;
        }
        cout<<endl;
        for(i=0;i<4;i++) X0[i]=X1[i];
}
cout<<endl<<"Решение"<<endl;
for(i=0;i<4;i++)
cout<<setw(10)<<X1[i];

cout<<endl<<"Число итераций="<<k;
return 0;}
```

Метод Зейделя

```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
int main()
{ double A[4][4]={{5,1,1,1},
                  {1,7,1,1},
                  {1,1,6,1},
                  {1,1,1,4}};
double B[4]={8,10,9,7};
double X0[4],X1[4];
double eps,tochnost,max_tochnost,s;
int k,kmax,i,j;

cout<<"Введите точность"<<endl;
cin>>eps;
cout<<"Введите max число итераций"<<endl;
cin>>kmax;

cout.precision(5); cout.setf(ios::left);

for(i=0;i<4;i++) X0[i]=0;

k=0;
while(fabs(max_tochnost)>eps)
{       if(k>=kmax) break;
        k=k+1;
        cout<<setw(5)<<k;
         max_tochnost=-100;
        for(i=0;i<4;i++)
        {s=0;
        for(j=0;j<4;j++)
        if(i!=j) s=s+A[i][j]*X0[j];
        X1[i]=(B[i]-s)/A[i][i];
        cout<<setw(10)<<X1[i];
        tochnost=fabs(X1[i]-X0[i]);
        if(tochnost>max_tochnost)
         max_tochnost=tochnost;
        X0[i]=X1[i];
}
cout<<endl;
}

cout<<endl<<"Решение"<<endl;
for(i=0;i<4;i++)
cout<<setw(10)<<X1[i];

cout<<endl<<"Число итераций="<<k;

return 0;}
```

Лабораторная работа №3

Аппроксимация и Интерполяция

Постановка задачи: Дана таблица координат точек $\{x_i,y_i\}$

x	x_0	x_1	x_2	x_3	x_4
y	y_0	y_1	y_2	y_3	y_4

Заданную таблицей функцию $y(x)$ заменяем функцией $f(x, a_1,...,a_n)$ таким образом, чтобы отклонение функции от приближающей функции было наименьшим. Функция $f(x,a_1,...,a_n)$ в общем случае называется аппроксимирующей, а если параметры $a_1,...,a_n$ определяются из условия равенства $f(x, a_1,...,a_n)= y(x)$, то такую функцию называют интерполирующей.

Название метода	Система для нахождения коэффициентов полинома	Ответ
Метод наименьших квадратов (аппроксимация)	полином 1-й степени $\begin{cases} na_0 + a_1\sum_i x_i = \sum_i y_i \\ a_0\sum_i x_i + a_1\sum_i x_i^2 = \sum_i x_i y_i \end{cases}$ полином 2-й степени $\begin{cases} na_0 + a_1\sum_i x_i + a_2\sum_i x_i^2 = \sum_i y_i \\ a_0\sum_i x_i + a_1\sum_i x_i^2 + a_2\sum_i x_i^3 = \sum_i x_i y_i \\ a_0\sum_i x_i^2 + a_1\sum_i x_i^3 + a_2\sum_i x_i^4 = \sum_i x_i^2 y_i \end{cases}$	$P1(x)=a_0+a_1x$ $P2(x)=a_0+a_1x+a_2x^2$
Метод неопределенных коэффициентов (интерполяция)	полином n-й степени $\begin{cases} a_0 + a_1x_0 + a_2x_0^2 + ... + a_nx_0^n = y_0 \\ a_0 + a_1x_1 + a_2x_1^2 + ... + a_nx_1^n = y_1 \\ ..... \\ a_0 + a_1x_n + a_2x_n^2 + ... + a_nx_n^n = y_n \end{cases}$	$P(x)=a_0+a_1x+..+a_nx^n$

Метод наименьших квадратов

```
#include <iostream.h>
#include <iomanip.h>
#include <math.h>
void gauss(double **c, int n)
{int i,j;
 int k;
 for (k=0; k<n; k++){
 for (j=n; j>=k; j--)
 c[k][j]/=c[k][k];
 for (i=k+1; i<n; i++)
 for (j=n; j>=k; j--)
 c[i][j]-=c[k][j]*c[i][k];}

 double *a;
 a=new double[n*sizeof(double)];
//Обратный ход
 for (i=0; i<n; i++) a[i]=c[i][n];

 for (i=n-2; i>=0; i--)
 for (j=i+1; j<n; j++)
 a[i] -= a[j]*(c[i][j]);
//Печать результата
 cout.precision(3);
 cout<<"Коэффициенты полинома \n";
 for (j=0; j<n; j++)
 cout<<"a"<<j<<"="<<a[j]<<endl;
 delete[] a;
}//gauss

void vvod(double *a, int n)
{
 for(int i=0;i<n;i++)  cin>>a[i];
}//vvod

int main()
{
 int n,i,j;
 cout<<"введите количество
точек"<<endl;
 cin>>n;
 double **c;
 double *x, *y;
  x= new double[n];
  cout<<"введите координаты x"<<endl;
  vvod(x,n);
  y= new double[n];
  cout<<"введите координаты  y"<<endl;
  vvod(y,n);
cout<<endl<<"проверьте
 координаты"<<endl;
for(i=0;i<n;i++)
cout<<"("<<x[i]<<","<<y[i]<<")"<<endl;

int m;
cout<<"степень полинома m"<<endl;
cin>>m;
 m=m+1; //если m=1, то уравнений 2

c= new double *[m];
for(i=0;i<m;i++)
c[i]=new double[m];

int k;
double s;

for(i=0;i<m;i++)
for(j=0;j<m;j++){ s=0;

for(k=0;k<n;k++){
s=s+pow(x[k],(i+j));}

c[i][j]=s;}

for(i=0;i<m;i++){s=0;
for(k=0;k<n;k++)
{ s=s+y[k]*pow(x[k],i);}
 c[i][m]=s;}

 cout<<"Матрица C+Y"<<endl;

 for(i=0;i<m;i++)
{for(j=0;j<m+1;j++)
cout<<setw(5)<<c[i][j]<<"   ";
cout<<endl;  }

cout<<endl;
gauss(c,m);

 delete[] x;
 delete[] y;
 delete[] c;

  cin.get();

   return 0;}
```

Метод неопределенных коэффициентов

```
#include <iostream.h>
#include <iomanip.h>
#include <math.h>

void gauss(double **c, int n)
{
int i,j;
int k;

for (k=0; k<n; k++){
for (j=n; j>=k; j--)
c[k][j]/=c[k][k];
for (i=k+1; i<n; i++)
for (j=n; j>=k; j--)
c[i][j]-=c[k][j]*c[i][k];
}

double *a;
a=new double[n*sizeof(double)];

//Обратный ход
for (i=0; i<n; i++)
a[i]=c[i][n];

for (i=n-2; i>=0; i--)
for (j=i+1; j<n; j++)
a[i]-=a[j]*(c[i][j]);

//Печать результата
cout.precision(3);
cout<<"Коэффициенты полинома\n";
for (j=0; j<n; j++)
cout<<"a"<<j<<"="<<a[j]<<endl;

delete[] a;
}//gauss

void vvod(double *a, int n)
{
 for(int i=0;i<n;i++)
 cin>>a[i];
}//vvod

int main()
{
 int n,i,j;
 cout<<"Введите кол-во
точек"<<endl;
 cin>>n;
 double **c;
 double *x, *y;
  x= new double[n*sizeof(double)];
  cout<<"Введите x"<<endl;
 vvod(x,n);

  y= new double[n*sizeof(double)];

 cout<<"Введите y"<<endl;
 vvod(y,n);

cout<<endl<<"проверьте координаты"<<endl;

for(i=0;i<n;i++)
cout<<"("<<x[i]<<","<<y[i]<<")"<<endl;

c= new double *[n];
for(i=0;i<n;i++)
c[i]=new double[n];


 for(i=0;i<n;i++)
 {
 for(j=0;j<n;j++) c[i][j]=pow(x[i],j);
 c[i][n]=y[i];
 }

 cout<<"Matritsa C+Y"<<endl;
 for(i=0;i<n;i++)
 {for(j=0;j<n+1;j++)
  cout<<setw(7)<<c[i][j];
  cout<<endl;}
  cout<<endl;

 gauss(c,n);

 delete[] x;
 delete[] y;

//  for(i=0;i<n;i++)
//  delete c[i];
   delete[] c;

 cin.get();

 return 0;}
```


Лабораторная работа №4
Вычисление определённого интеграла

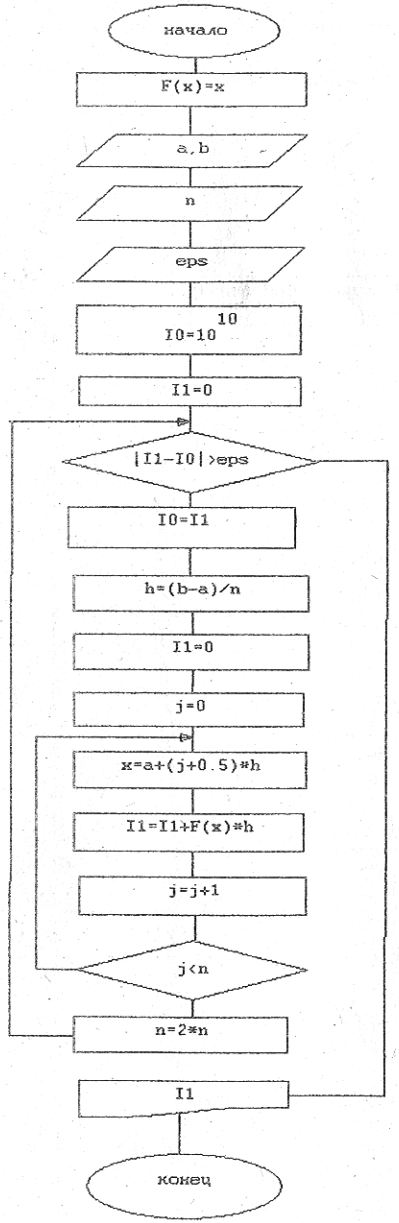
Постановка задачи: Вычислить определённый интеграл $I = \int_a^b f(x)$

($hx=(b-a)/n$- шаг разбиения, n - количество разбиений) методами:

- метод левых прямоугольников,
- метод правых прямоугольников,
- метод центральных прямоугольников,
- метод трапеций,
- метод Симпсона.

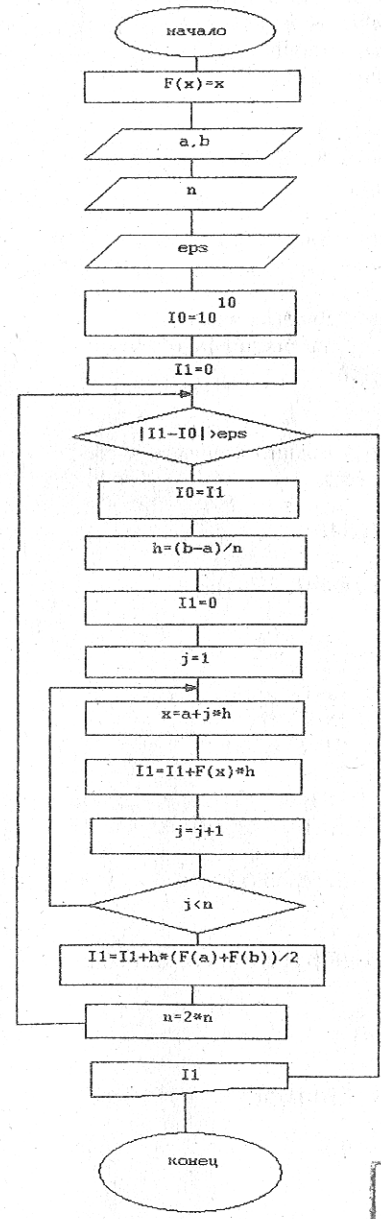
Название метода	Итерационная формула
Метод левых прямоугольников	$x_i=a+i\,hx$ $i=0..n-1$ $I = hx\sum_{i=0}^{n-1} f(x_i)$
Метод правых прямоугольников	$x_i=a+i\,hx$ $i=1..n$ $I = hx\sum_{i=1}^{n} f(x_i)$
Метод центральных прямоугольников	$x_i=a+(i-0{,}5)\,hx$ $i=1..n$ $I = hx\sum_{i=1}^{n} f(x_i)$
Метод трапеций	$x_i=a+i\,hx$ $i=1..n-1$ $I = hx\left[\frac{f(a)+f(b)}{2} + \sum_{i=1}^{n-1} f(x_i)\right]$
Метод Симпсона	$x_i=a+i\,hx$, $i=1,3..n-1$ $S1 = \sum_i f(x_i)$ $x_i=a+i\,hx$, $i=2,4..n-2$ $S2 = \sum_i f(x_i)$ $I = \frac{hx}{3}(f(a)+2\cdot S1+4\cdot S2+f(b))$

Метод центральных прямоугольников



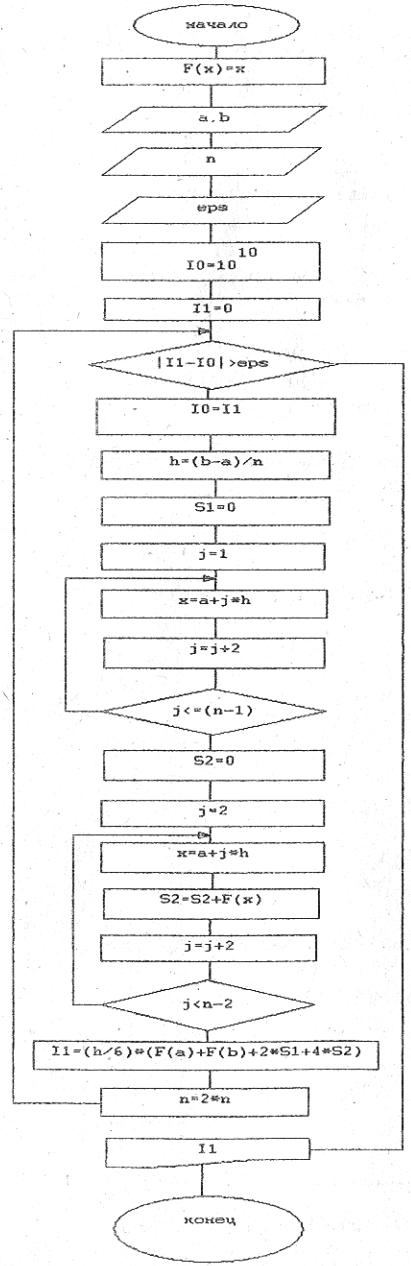

```
#include <iostream.h>
#include <math.h>
double F(double x)
{return(x);}

int main()
{double I0,I1,h,eps,a,b,x;
int n,j;

cout<<"Введите a и b"<<endl;
cin>>a>>b;

cout<<"Введите число
        разбиений [a,b]"<<endl;
cin>>n;

cout<<"Введите точность"<<endl;
cin>>eps;

I0=1E+10;
I1=0;
while(fabs(I1-I0)>eps)
{
        h=(b-a)/n;
        I1=0;
        for(j=0;j<n;j++)
        {x=a+(j+0.5)*h;
        I1=I1+F(x)*h;
        }

n=2*n;
I0=I1;}

cout<<"Интеграл I="<<I1<<endl;

return 0;}
```


Метод трапеций




```
#include <iostream.h>
#include <math.h>
double F(double x)
{return(x);}

int main()
{double I0,I1,h,eps,a,b,x;
int n,j;

cout<<"Введите a и b"<<endl;
cin>>a>>b;

cout<<"Введите число
        разбиений [a,b]"<<endl;
cin>>n;

cout<<"Введите точность"<<endl;
cin>>eps;

I0=1E+10;
I1=0;
while(fabs(I1-I0)>eps)
{
        h=(b-a)/n;
        I1=0;
        for(j=1;j<n;j++)
        {x=a+j*h;
        I1=I1+F(x)*h;
        }
 I1=I1+h*(F(a)+F(b))/2;

n=2*n;
I0=I1;}

cout<<"Интеграл I="<<I1<<endl;

return 0;}
```

Метод Симпсона




```
#include <iostream.h>
#include <math.h>
double F(double x)
{return(x);}

int main()
{double I0,I1,S1,S2,h,eps,a,b,x;
int n,j;

cout<<"Введите a и b"<<endl;
cin>>a>>b;

cout<<"Введите число
        разбиений [a,b]"<<endl;
cin>>n;


cout<<"Введите точность"<<endl;
cin>>eps;

I0=1E+10;
I1=0;
while(fabs(I1-I0)>eps)
{
        h=(b-a)/n;
        S1=0;
        for(j=1;j<=n-1;j+2)
        {x=a+j*h;
        S1=S1+F(x);
        }
         S2=0;
        for(j=2;j<n;j+2)
        {x=a+j*h;
        S2=S2+F(x);
        }

 I1=(h/6)*(F(a)+F(b)+2*S1+4*S2);

 n=2*n;
 I0=I1;}

cout<<"Интеграл I="<<I1<<endl;

return 0;}
```

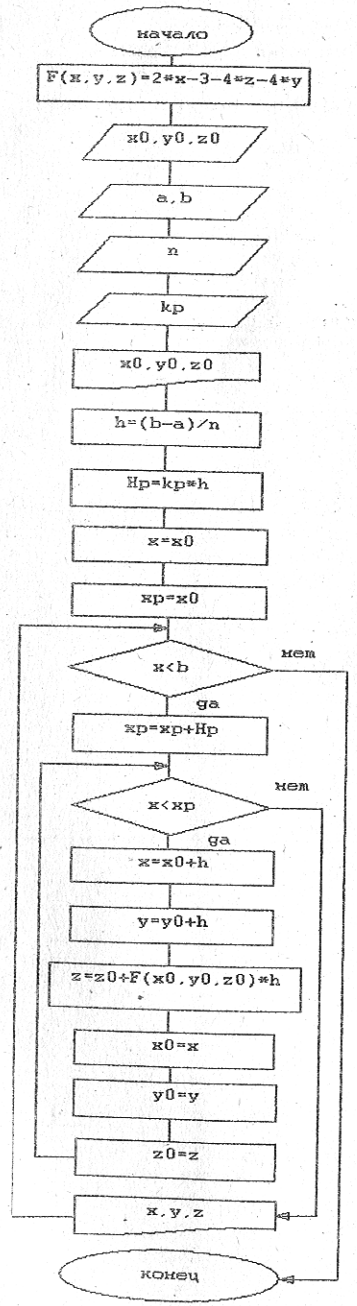

Лабораторная работа №5
Обыкновенные дифференциальные уравнения.
Численное решение задач с начальными условиями Коши

Постановка задачи: Дано дифференциальное уравнение второго порядка с начальными условиями

$$\begin{cases} y'' = f(x, y, y'), \\ y(x_0) = y_0, \\ y'(x_0) = y_0. \end{cases}$$

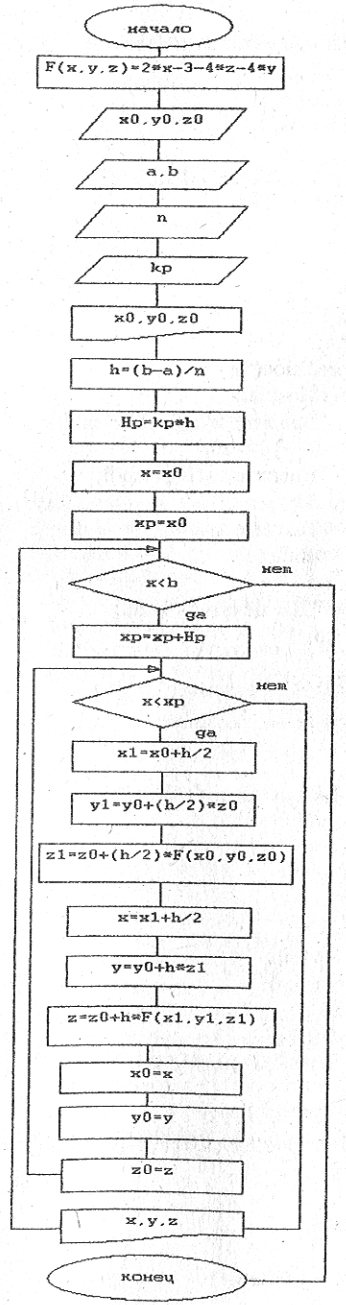
Выполнить интегрирование на отрезке $[a,b]$.

Название метода	Итерационные формулы
Метод Эйлера Обозначения: $y'=z$ $z'=f(x,y,z)$	$\begin{cases} x_{i+1} = x_i + hx, \\ y_{i+1} = y_i + hxz_i, \\ z_{i+1} = z_i + hxf(x_i, y_i, z_i) \end{cases}$
Модифицированный метод Эйлера-1-я модификация Обозначения: $y'=z$ $z'=f(x,y,z)$	Первый этап: $\begin{cases} x_{i+1/2} = x_i + hx/2, \\ y_{i+1/2} = y_i + (hx/2)z_i, \\ z_{i+1/2} = z_i + (hx/2)f(x_i, y_i, z_i) \end{cases}$ Второй этап: $\begin{cases} x_{i+1} = x_i + hx, \\ y_{i+1} = y_i + hxz_{i+1/2}, \\ z_{i+1} = z_i + hxf(x_{i+1/2}, y_{i+1/2}, z_{i+1/2}) \end{cases}$
Модифицированный метод Эйлера-2-я модификация Обозначения: $y'=z$ $z'=f(x,y,z)$	Первый этап: $\begin{cases} x_{i+1} = x_i + hx, \\ y1_{i+1} = y_i + hxz_i \\ z1_{i+1} = z_i + hxf(x_i, y_i, z_i) \end{cases}$ Второй этап: $\begin{cases} x_{i+1} = x_i + hx, \\ y_{i+1} = y_i + hx(z_i + z1_i)/2, \\ z_{i+1} = z_i + hx(f(x_i, y_i, z_i) + f(x_i, y1_i, z1_i))/2 \end{cases}$



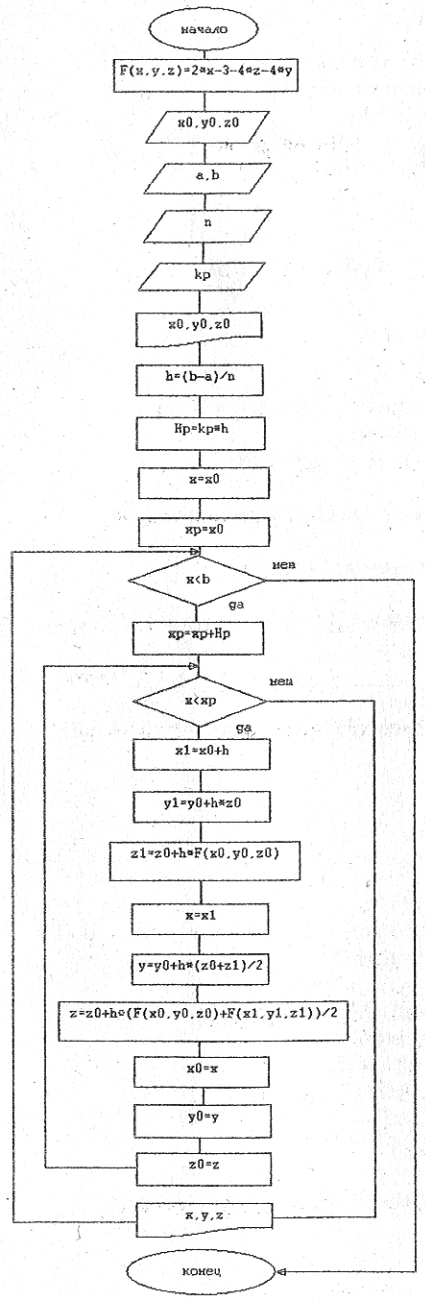
Метод Эйлера с выбором шага печати

```
#include <iostream.h>
#include <iomanip.h>
#include <math.h>

double F(double x,double y,double z)
{return (2*x-3-4*z-4*y);}
int main()
{
        double x0,y0,z0,x,y,z,xp,h,Hp,a,b;
        int kp,n;

        cout.precision(5);
        cout.setf(ios::left);
        cout<<" Введите н.у.  x0,y0,z0"<<endl;
         cin>>x0>>y0>>z0;
        cout<<"Введитеа и b"<<endl;
        cin>>a>>b;
        cout<<"Введите число
разбиений"<<endl;
        cin>>n;
        cout<<"Шаг печати"<<endl;
        cin>>kp;
 cout<< "x"<<setw(10)<<"y"<<setw(10)<<"z";
cout<<endl<<"_____"<<en
dl;
cout<<
x0<<setw(10)<<y0<<setw(10)<<z0<<endl;
        h=(b-a)/n;
        Hp=kp*h;

        x=x0;
        xp=x0;
        while(x<b)
  {
     xp=xp+Hp;
       while(x<xp){
             x=x0+h;
            y=y0+h;
            z=z0+F(x0,y0,z0)*h;
            x0=x;
            y0=y;
            z0=z; }
cout<<x<<setw(10)<<y<<setw(10)<<z<<endl;
   }

return 0;}
```

Первая модификация метода Эйлера

```
#include <iostream.h>
#include <iomanip.h>
#include <math.h>
double F(double x,double y,double z)
{return (2*x-3-4*z-4*y);}

int main()
{
double x0,y0,z0,x1,y1,z1,x,y,z,xp,h,Hp,a,b;
int kp,n;

cout.precision(5);
cout.setf(ios::left);

cout<<"Введите n.y. x0,y0,z0"<<endl;
cin>>x0>>y0>>z0;
cout<<"Введите a и b"<<endl;
cin>>a>>b;
cout<<"Введите число разбиений"<<endl;
cin>>n;
cout<<"Шаг печати"<<endl;
cin>>kp;
cout<<"x"<<setw(5)<<"y"<<setw(5)<<"z"<<end
l;
cout<<"_____"<<end
l;
cout<<x0<<setw(5)<<y0<<setw(5)<<z0<<endl;

h=(b-a)/n;
Hp=kp*h;
 x=x0;
xp=x0;
while(x<b)
   {   xp=xp+Hp;
       while(x<xp)
          {x1=x0+h/2;
           y1=y0+(h/2)*z0;
          z1=z0+(h/2)*F(x0,y0,z0);
           x=x1+h/2;
           y=y0+h*z1;
           z=z0+h*F(x1,y1,z1);
            x0=x;
           y0=y;
           z0=z; }
 cout<<x<<setw(5)<<y<<setw(5)<<z<<endl;
}
return 0;}
```



Вторая модификация метода Эйлера

```
#include <iostream.h>
#include <iomanip.h>
#include <math.h>
double F(double x,double y,double z)
{return (2*x-3-4*z-4*y);}
int main()
{
        double
x0,y0,z0,x1,y1,z1,x,y,z,xp,h,Hp,a,b;
        int kp,n;
        cout.precision(5);
        cout.setf(ios::left);
        cout<<"Введите x0,y0,z0"<<endl;
         cin>>x0>>y0>>z0;
        cout<<"Введите a и b"<<endl;
        cin>>a>>b;
        cout<<"Введите число
разбиений"<<endl;
        cin>>n;
        cout<<"Шаг печати"<<endl;
        cin>>kp;
 cout<<
"x"<<setw(10)<<"y"<<setw(10)<<"z";
cout<<endl<<"_____"<<
endl;
cout<<
x0<<setw(10)<<y0<<setw(10)<<z0<<endl;
        h=(b-a)/n;
        Hp=kp*h;
        x=x0;
        xp=x0;
        while(x<b)
        { xp=xp+Hp;
         while(x<xp)
         {x1=x0+h;
         y1=y0+h*z0;
         z1=z0+h*F(x0,y0,z0);
         x=x1;
         y=y0+h*(z0+z1)/2;
         z=z0+h*(F(x0,y0,z0)+F(x1,y1,z1))/2;
          x0=x;
         y0=y;
         z0=z;  }
cout<<
x<<setw(10)<<y<<setw(10)<<z<<endl; }
return 0;}
```


Список литературы

1. Павловская Т.А. С/С++Программирование на языке высокого уровня: учебник для вузов/ Т.А. Павловская. СПб. Изд.-во «Питер», 2007.
2. Павловская Т.А. С/С++Структурное программирование:практикум/ Т.А.Павловская, Ю.В.Щупак. СПб.: Изд.-во «Питер», 2007.
3. Турчак Л.И. Основы численных методов/ Л.И.Турчак. Изд - во «Наука» Главная редакция Физико-математической литературы, 1987
4. Подбельский В.В. Программирование на языке Си./ В.В.Подбельский, С.С.Фомин. М.:Изд.-во «Финансы и статистика»,2007.
5. Хабибуллин И.В. Программирование на языке высокого уровня: С/С++ / И.В.Хабибуллин. СПб.BHV-Санкт-Петербург, 2006.

Содержание