**Московский авиационный институт**

(национальный исследовательский университет)

Факультет радиоэлектроники

Кафедра 403



**Разработка алгоритмов и программ решения**

**алгебраических задач численными методами**

Расчётно-графическая работа

по дисциплине «Информатика»

|  |  |
| --- | --- |
|  | Выполнил:  *студент группы М4О-103Б-17*  Парамонов Н. М. |
|  | Принял:  *преподаватель кафедры 403*  Кошелькова Л. В. |

Москва

2017 г.

**Оглавление**

[1. Условие задачи 3](#_Toc498562031)

[2. Анализ задания 3](#_Toc498562032)

[3. Теоретические сведения 4](#_Toc498562033)

[4. Схемы алгоритмов 5](#_Toc498562034)

[5. Описание Delphi-программы 11](#_Toc498562035)

[6. Текст программы 12](#_Toc498562036)

[7. Пример выполнения программы: 16](#_Toc498562037)

[8. Набор тестов 16](#_Toc498562038)

[9. Выводы 19](#_Toc498562039)

[10. Список использованной литературы 20](#_Toc498562040)

# **Условие задачи**

Вариант №79

Разработать схему алгоритма, составить Delphi-проект вычисления таблицы значений функции:

Аргумент X принимает N значений от Xn с шагом Dx, а параметр А принимает значения от An до Аk с шагом Da. Параметр B принимает значение, численно равное интегралу:

вычисленному при заданных пределах интегрирования и с погрешностью .

# **Анализ задания**

Входные данные:

1. Xn – начальное значение аргумента, тип – вещественный;
2. Dx – шаг изменения аргумента, тип – вещественный;
3. N – число значений аргумента, тип – целый;
4. An – начальное значение параметра A, тип – вещественный;
5. Ak – конечное значение параметра A, тип – вещественный;
6. Da – шаг изменения параметра A, тип – вещественный;
7. C – нижняя граница интегрирования, тип – вещественный;
8. D – верхняя граница интегрирования, тип – вещественный;
9. Eps – погрешность вычисления интеграла, тип – вещественный;
10. Km - максимальное количество итераций, тип – целый;

Выходные данные:

1. Mx – массив (одномерный) значений аргумента X, тип – вещественный;
2. My – массив (двумерный) значений функции Y, тип – вещественный;
3. Ma – массив (одномерный) значений параметра A, тип – вещественный;
4. B – численное значение интеграла, тип – вещественный;
5. Er – массив (двумерный) признака ошибки при вычислении функции, тип – целый;
6. Err – признак ошибки при вычислении определённого интеграла, тип – целый.

В алгоритме выполняются следующие функции:

1. Ввод исходных данных;
2. Вычисление значения определённого интеграла;
3. Вычисление таблицы значений функции;
4. Проверка значения подкоренного выражения и формирование признака ошибки, если оно имеет отрицательный знак;
5. Вывод результатов вычислений.

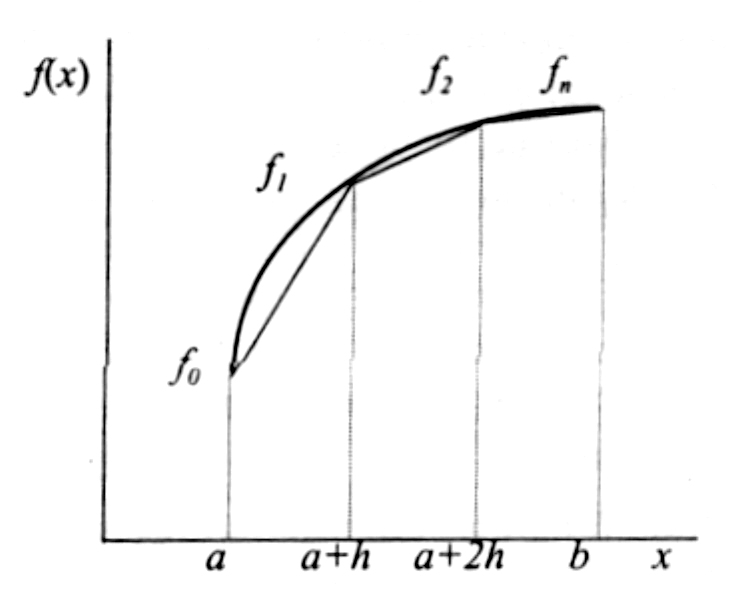
# **Теоретические сведения**

***Метод трапеций для оценки определенного интеграла.***

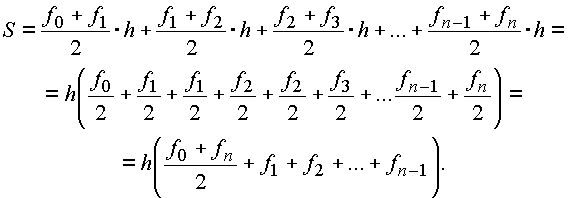
Величина определенного интеграла численно равна площади фигуры, образованной графиком функции и осью абсцисс (геометрический смысл определенного интеграла).

Следовательно, найти интеграл – это значит оценить площадь фигуры, ограниченной перпендикулярами, восстановленными к графику подынтегральной функции  f(x)  из точек a и b, расположенных на оси аргумента x.

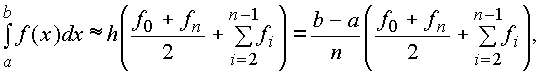
Для решения задачи разобьем интервал [a, b] на n одинаковых участков. Длина каждого участка будет равна h=(b-a)/n (см. рис.).



Восстановим перпендикуляры из каждой точки до пересечения с графиком функции f(x). Если заменить полученные криволинейные фрагменты графика функции отрезками прямых, то тогда приближенно площадь фигуры, а следовательно и величина определенного интеграла оценивается как площадь всех полученных трапеций. Обозначим последовательно значения подынтегральных функций на концах отрезков f0, f1, f2, ..., fn и подсчитаем площадь трапеций

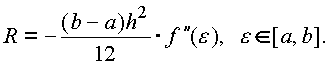


В общем случае формула трапеций принимает вид



где fi – значение подынтегральной функции в точках разбиения интервала (a,b) на равные участки с шагом h; f0, fn – значения подынтегральной функции соответственно в точках a и b.

Остаточный член пропорционален длине интервала [a,b] и квадрату шага h



Согласно рис. и формуле остаточного члена, точность вычисления определенного интеграла повышается с уменьшением шага h(увеличением числа отрезков n).

Метод трапеций можно реализовать в виде процедуры или даже функции, поскольку результат вычисления определенного интеграла – скалярная величина. Параметрами программного модуля являются границы интервала (a и b) и число шагов разбиения на малые интервалы n.

# **Схемы алгоритмов**

В соответствии с принципами структурного программирования каждый функционально законченный фрагмент программы оформ­лен в виде подпрограммы. В результате программа включает глав­ную программу и набор подпрограмм, предназначенных соответ­ственно для табулирования функции (Tab), вычисления интеграла (Trap), вывода результатов выполнения программы (RezOut).

Схема алгоритма главной программы представлена на рис. 1, а таблица обозначения переменных главной программы – в табл. 1.

Главная программа начинается с ввода значений входных данных.

*Таблица 1*

|  |  |  |
| --- | --- | --- |
| Обозначение в задании | Обозначение в алгоритме | Наименование |
| Xn | Xn | Начальное значение аргумента, вещественный тип |
| Dx | Dx | Шаг изменения аргумента, вещественный тип |
| N | N | Число значений аргумента, целый тип |
| An | An | Начальное значение параметра A, вещественный тип |
| Ak | Ak | Конечное значение параметра A, вещественный тип |
| Da | Da | Шаг изменения параметра A, вещественный тип |
| C, D | C, D | Верхняя и нижняя границы интегрирования, вещественный тип |
|  | Eps | Погрешность вычисления интеграла, вещественный тип |
|  | Km | Максимальное количество итераций, целый тип |
|  | Mx | Массив значений аргумента X, вещественный тип |
|  | My | Массив значений функции Y, вещественный тип |
|  | Ma | Массив значений параметра A, вещественный тип |
| B | B | Параметр функции, вещественный тип |
|  | Er | Массив признака ошибки при вычислении функции, целый тип |
|  | Err | Признак ошибки при вычислении определённого интеграла, целый тип |
| X | X | Аргумент, вещественный тип |
| Y | Y | Функция, вещественный тип |
|  | Ka | Количество значений параметра A, целый тип |
|  | I, J | Счётчики числа повторений циклов, целый тип |

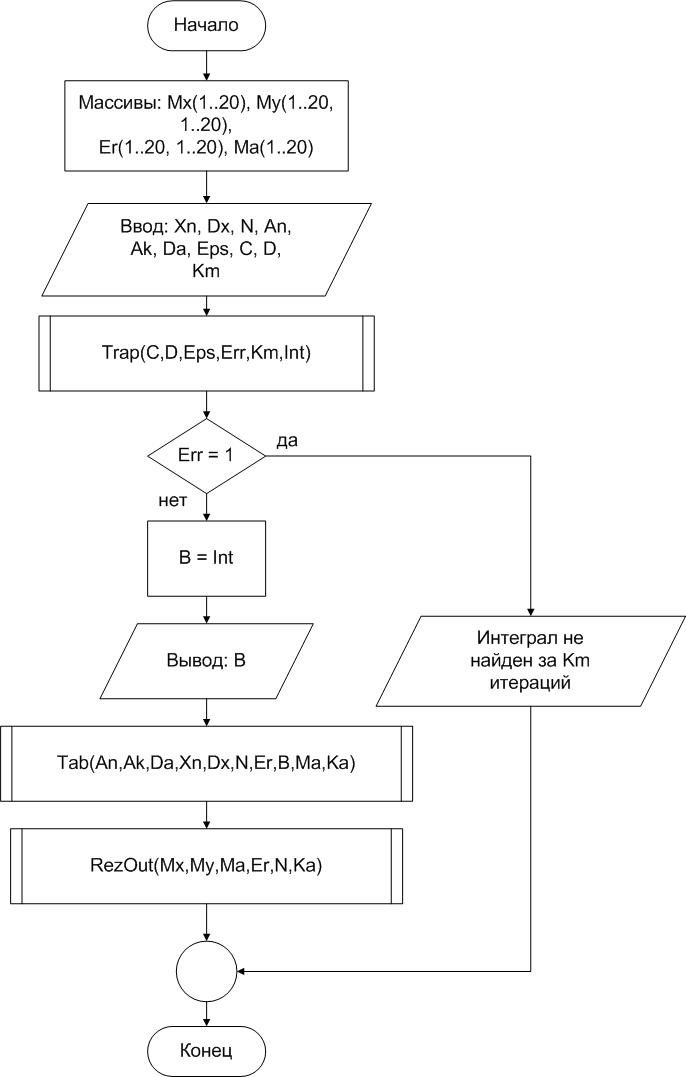


Рис. 1. Схема алгоритма главной программы

Вычисление значения определённого интеграла производится путём обращения к ПП **Trap**, формирующей также признак ошибки в случае, если значение интеграла не найдено за предельно допустимое число итераций Km. При Err = 1 выводится диагностическое сообщение «Интеграл не найден за Km итераций», иначе происходит табулирование функции (ПП **Tab**) и вывод результатов выполнения программы (ПП **RezOut**). Значение Ka определяет количество значений аргумента функции.

Схемы алгоритмов подпрограмм, используемых в данной программе, с указанием их назначения и списков формальных параметров приведены на рис. 2 – 5.

Подпрограмма-функция **F** (рис. 2) предназначена для вычисления значения подынтегральной функции, представляет собой один оператор присваивания и используется в ПП вычисления интеграла Trap.

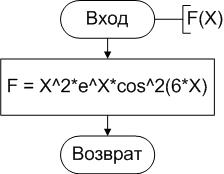


Рис. 2. Схема алгоритма подпрограммы-функции F

Подпрограмма-функция F предназначена для вычисления значения функции F.

Список формальных параметров: X.

Входные параметры:

X – аргумент функции, тип – вещественный.

Подпрограмма-процедура вычисления определённого интеграла **Trap** (рис. 3) реализуется циклом итерационного типа, который завершается при условии, что разность двух последовательных приближений меньше заданной погрешности (Del < Eps). Во избежание «зацикливания» программы при неудачном выборе начального приближения (или ошибках в данных) в алгоритме предусмотрено задание предельно допустимого количества повторений цикла Km и выполнение соответствующего арифметического цикла с проверкой окончания J <= Km.

Подпрограмма-процедура табулирования **Tab** (рис. 4), выполненная в виде двойного цикла, определяет функциональную зависимость вида Y = f(X) при различных значениях параметров, поэтому внутренний цикл должен быть связан с изменением аргумента X, а внешний – с изменением параметра A. Во внутреннем цикле имеются две развилки: одна из них обусловлена тем, что функция задаётся разными формулами на разных участках изменения аргумента (проверка условия X < 1), вторая – проверяет знак подкоренного выражения и при условии A\*X+B2 > 0 вычисляет значение функции, в противном случае – формирует признак ошибки Er[I, J] = 1.

Подпрограмма-процедура **RezOut** (рис. 5) выводит результаты выполнения программы. По структуре она построена аналогично подпрограмме Tab: представляет собой двойной цикл арифметического типа, но в отличие от ПП Tab она получает количество значений параметра A в качестве входных данных.

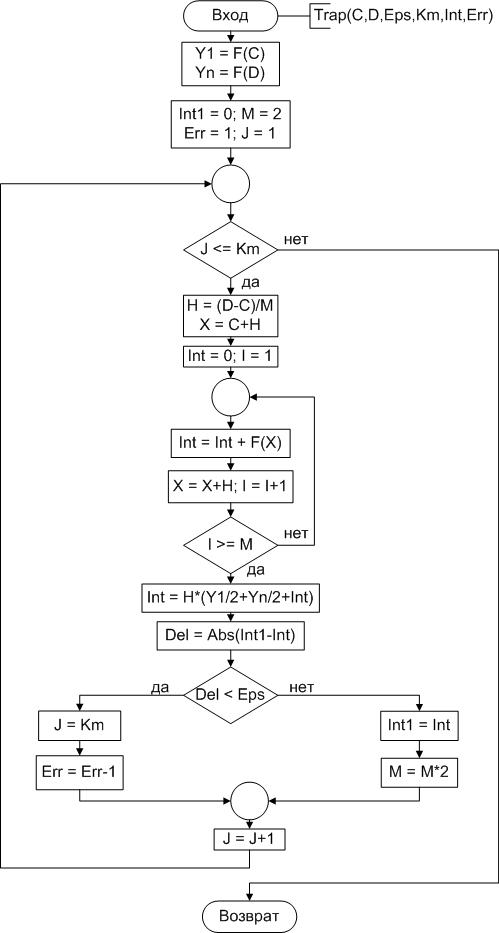


Рис. 3. Схема алгоритма подпрограммы-процедуры Trap

Подпрограмма-процедура Trap предназначена для вычисления значения определённого интеграла с заданной погрешностью методом трапеций.  
Список формальных параметров: C, D, Eps, Km, Int, Err.

Входные параметры:

C, D – пределы интегрирования, тип – вещественный;

Eps – погрешность вычисления интеграла, тип – вещественный;

Km – предельно допустимое количество итераций, тип – целый.

Выходные параметры:

Int – значение интеграла, тип – вещественный;

Err – признак ошибки, тип – целый.

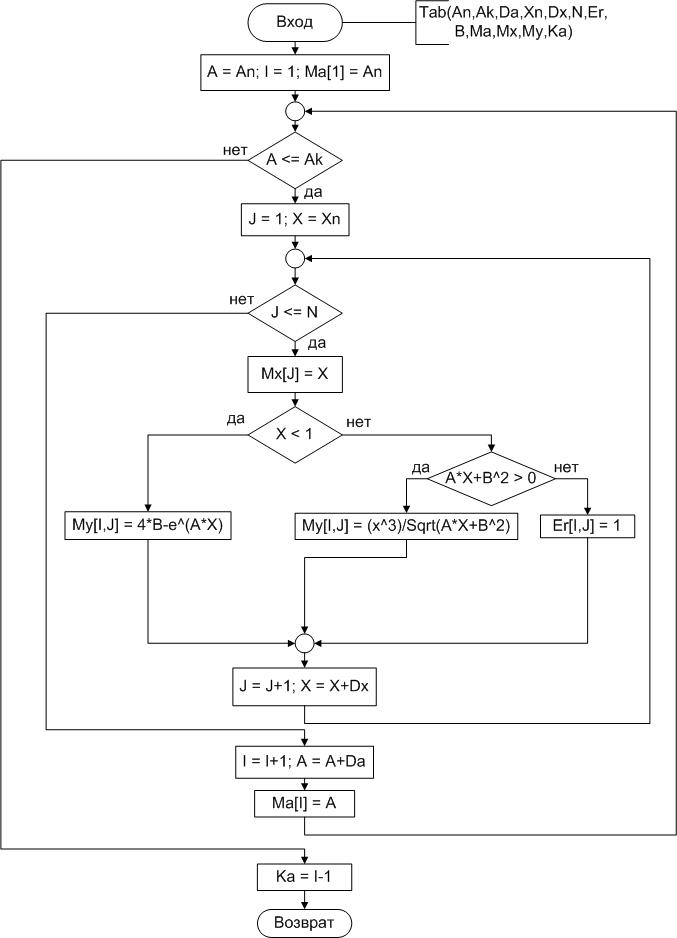


Рис. 4. Схема алгоритма подпрограммы-процедуры Tab

Подпрограмма-процедура Tab предназначена для вычисления таблицы значений функции Y.  
Список формальных параметров: An, Ak, Da, Xn, Dx, N, B, Er, Mx, My, Ma, Ka.

Входные параметры:

An – начальное значение параметра A, тип – вещественный;

Ak – конечное значение параметра A, тип – вещественный;

Da – шаг изменения параметра A, тип – вещественный;

Xn – начальное значение аргумента, тип – вещественный;

Dx – конечное значение аргумента, тип – вещественный;

N – количество значений аргумента, тип – целый;

B – параметр функции, тип – вещественный.

Выходные параметры:

Er – массив признака ошибки, тип – целый;

Mx – массив значений аргумента X, тип – вещественный;

My – массив значений функции Y, тип – вещественный;

Ma – массив значений параметра A, тип – вещественный;

Ka – количество значений параметра A, тип - целый.

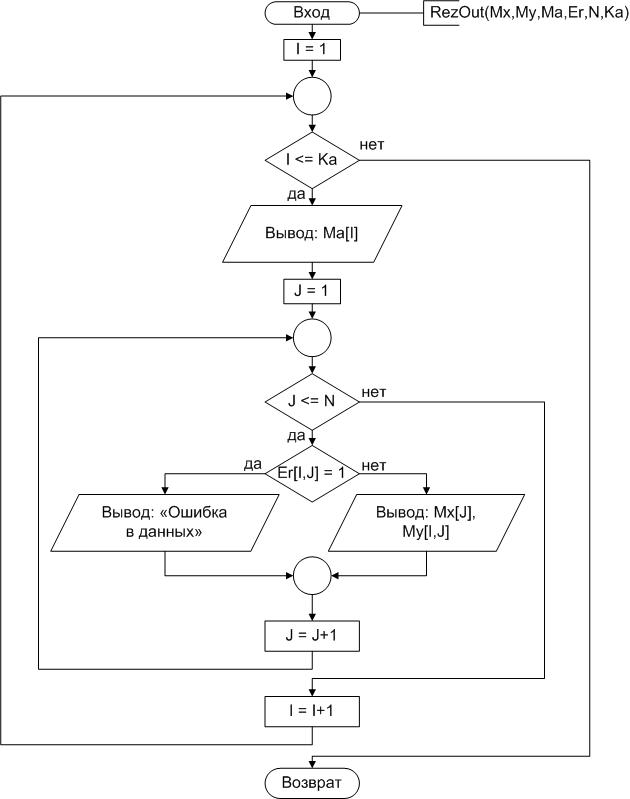


Рис. 5. Схема алгоритма подпрограммы-процедуры RezOut

Подпрограмма-процедура RezOut предназначена для вывода результатов выполнения программы на внешние носители информации.  
Список формальных параметров: Mx, My, Ma, Er, N, Ka.

Входные параметры:

Mx – массив значений аргумента X, тип – вещественный;

My – массив значений функции Y, тип – вещественный;

Ma – массив значений параметра A, тип – вещественный;

Er – массив признака ошибки, тип – целый;

N – количество значений аргумента, тип – целый;

Ka – количество значений параметра A, тип - целый.

# **Описание Delphi-программы**

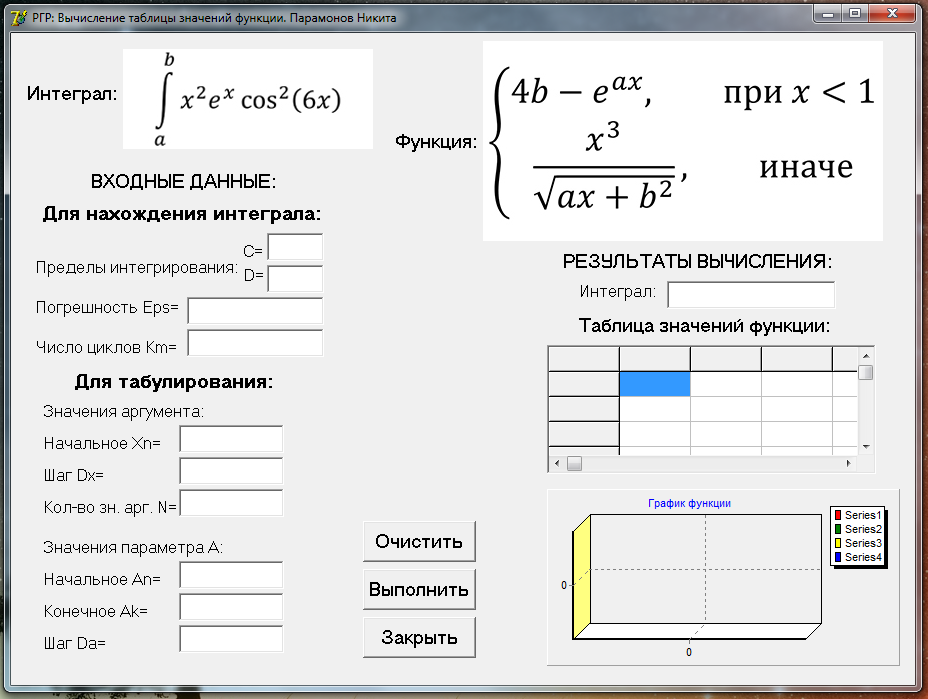


Рис. 6. Форма Delphi-приложения

Разработка приложения в визуальной среде программирования Delphi включает два этапа:

* разработка интерфейса приложения;
* определение функциональности приложения, т.е. написание кода.

Интерфейс определяет способ взаимодействия пользователя и приложения, т.е. внешний вид формы при выполнении приложения и то, каким образом пользователь управляет приложением.

Разработка интерфейса состоит в создании главного окна, т.е. в расположении на форме необходимых компонентов редактирования, отображения и управления. Внешний вид формы для задачи табулирования функции представлен на рисунке 6. На форме расположены следующие визуальные компоненты: Label, Edit, Button, StringGrid, Chart, Image.

Функциональность приложения определяется процедурами, которые выполняются при возникновении определенных событий. Структура Delphi-проекта соответствует рассмотренным в предыдущем разделе схемам алгоритмов, но дополнительно включает процедуры или функции преобразования данных символьного типа в арифметические при вводе и обратного преобразования арифметических данных в строковые – при выводе. Текст модуля формы представлен в следующем пункте.

Обработчик кнопки «Выполнить» по событию OnClick реализует процедуры (Trap, Tab, RezOut), необходимые для выполнения задачи. Обработчик включает в себя: функции преобразования входных данных типа String, полученных из компонентов Edit, в вещественные числа типа real или целые числа типа integer; вызов процедур Trap, Tab, RezOut; заполнение таблицы StringGrid – вывод данных; функции преобразования выходных данных типа real (значение интеграла) в данные типа String для вывода в компонент Edit; вывод сообщений об ошибках, если они присутствуют; вывод графика с использованием компонента Chart.

Обработчик кнопки «Очистить» по событию OnClick включает в себя: очистку компонентов Edit, используемых для получения входных (выходных – для значения интеграла) данных, очистку серий компонента Chart, очистку компонента StringGrid в цикле.

Обработчик кнопки «Закрыть» по событию OnClick включает в себя метод Close, обеспечивающий закрытие приложения.

# **Текст программы**

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, ExtCtrls, TeeProcs, TeEngine, Chart, Grids, Series,

jpeg;

type

TForm1 = class(TForm)

Label1: TLabel;

Label2: TLabel;

Label5: TLabel;

Label6: TLabel;

Label7: TLabel;

Label8: TLabel;

Label9: TLabel;

Label10: TLabel;

Edit1: TEdit;

Edit2: TEdit;

Label11: TLabel;

Edit3: TEdit;

Label12: TLabel;

Edit4: TEdit;

Label13: TLabel;

Label14: TLabel;

Label15: TLabel;

Edit5: TEdit;

Label16: TLabel;

Edit6: TEdit;

Label17: TLabel;

Edit7: TEdit;

Label18: TLabel;

Label19: TLabel;

Edit8: TEdit;

Label20: TLabel;

Edit9: TEdit;

Label21: TLabel;

Edit10: TEdit;

Button1: TButton;

Button2: TButton;

Button3: TButton;

StringGrid1: TStringGrid;

Label22: TLabel;

Chart1: TChart;

Label23: TLabel;

Edit11: TEdit;

Series1: TLineSeries;

Image1: TImage;

Image2: TImage;

Series2: TLineSeries;

Series3: TLineSeries;

Series4: TLineSeries;

procedure Button3Click(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

implementation

{$R \*.dfm}

procedure TForm1.Button3Click(Sender: TObject); //Кнопка "Закрыть"

begin

close;

end;

procedure TForm1.Button1Click(Sender: TObject); //Кнопка "Очистить"

var I,J: integer;

begin

Edit1.Text := '';

Edit2.Text := '';

Edit3.Text := '';

Edit4.Text := '';

Edit5.Text := '';

Edit6.Text := '';

Edit7.Text := '';

Edit8.Text := '';

Edit9.Text := '';

Edit10.Text := '';

Edit11.Text := '';

for I := 0 to 20 do

for J := 0 to 20 do

Form1.StringGrid1.Cells[I,J] := '';

for I := 0 to 3 do Form1.Chart1.Series[I].Clear;

end;

procedure TForm1.Button2Click(Sender: TObject); // Кнопка "Вычислить"

const Nmax = 20;

type

Tmy = Array[1..Nmax, 1..Nmax] of real;

Tmx = Array[1..Nmax] of real;

var

My, Er: Tmy;

Mx, Ma: Tmx;

Xn,Dx,An,Ak,Da,Eps,C,D,Int,B: real;

N,Km,Err,Ka: integer;

procedure Trap(C,D,Eps:real; Km:integer; var Int:real; var Err:integer); //Процедура вычисления интеграла

function F(X: real): real;

begin

F := X\*X\*Exp(X)\*cos(6\*X)\*cos(6\*X);

end;

var

Y1,Yn,Int1,X,H,Del: real;

M,I,J: integer;

begin

Y1 := F(C);

Yn := F(D);

Int1 := 0;

M := 2;

Err := 1;

J := 1;

while J <= Km do

begin

H := (D-C)/M;

X := C + H;

Int := 0;

repeat

Int := Int + F(X);

X := X + H;

I := I + 1;

until I >= M;

Int := H\*(Y1/2 + Yn/2 + Int);

Del := Abs(Int1-Int);

if Del < Eps then

begin

J := Km;

Err := Err - 1;

end

else

begin

Int1 := Int;

M := M\*2;

end;

J := J + 1;

end;

end;

procedure Tab(An,Ak,Da,Xn,Dx,B:real; N:integer; var My, Er: Tmy; var Mx, Ma: Tmx; var Ka: integer); //Процедура табулирования функции

var

A,X :real;

I,J:integer;

begin

A := An;

I := 1;

while A <= Ak do

begin

X := Xn;

for J := 1 to N do

begin

Mx[J] := X;

if x < 1 then My[I,J] := 4\*B-exp(A\*X)

else

if A\*X+B\*B > 0 then My[I,J] := (x\*x\*x)/(sqrt(A\*X+B\*B))

else Er[I,J] := 1;

X := X + Dx;

end;

Ma[I] := A;

I := I + 1;

A := A + Da;

end;

Ka := I-1;

end;

procedure RezOut(var Mx,Ma: Tmx; var My,Er: Tmy; Ka,N:integer); //Процедура вывода результатов

var

I,J:integer;

begin

//Формирование заголовков таблицы

StringGrid1.Cells[0,0] := 'X/A';

for I := 1 to Ka do

begin

StringGrid1.Cells[I,0]:=('A['+IntToStr(I)+']='+FloatToStr(Ma[I]));

Form1.Chart1.Series[I-1].Title:='A['+IntToStr(I)+']';

Form1.Chart1.Series[I-1].Clear;

for J := 1 to N do

begin

with Form1.StringGrid1 do

begin

Cells[0,J]:=('X['+IntToStr(J)+']='+FloatToStr(Mx[J]));

if Er[I,J] = 1 then Cells[I,J]:= 'Err' else Cells[I,J]:=FloatToStrF(My[I,J],ffGeneral,6,5);

end;

Form1.Chart1.Series[I-1].AddXY(Mx[J],My[I,J]);

end;

end;

end;

begin

//Исходные данные для вычисления интеграла

C:= StrToFloat(Edit1.Text);

D:= StrToFloat(Edit2.Text);

Eps:= StrToFloat(Edit3.Text);

Km:= StrToInt(Edit4.Text);

//Исходные данные для табулирования

Xn:= StrToFloat(Edit5.Text);

Dx:= StrToFloat(Edit6.Text);

N:= StrToInt(Edit7.Text);

An:= StrToFloat(Edit8.Text);

Ak:= StrToFloat(Edit9.Text);

Da:= StrToFloat(Edit10.Text);

Trap(C,D,Eps,Km,Int,Err);

if Eps >= 1 then

begin

ShowMessage('Погрешность Eps должна быть < 1');

exit;

end

else

if Err = 1 then

begin

ShowMessage('Интеграл не найден за '+IntToStr(Km)+' итераций');

exit;

end

else

begin

B := Int;

Edit11.Text := FloatToStrF(B,ffGeneral,10,6);

Tab(An,Ak,Da,Xn,Dx,B,N,My,Er,Mx,Ma,Ka);

RezOut(Mx,Ma,My,Er,Ka,N);

end;

end;

end.

# **Пример выполнения программы:**

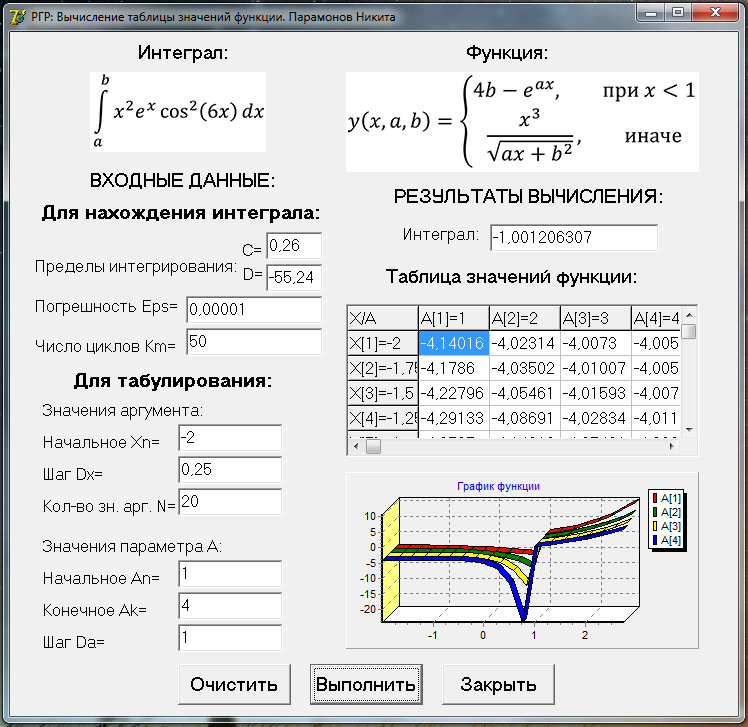


Рис. 7. Форма Delphi-приложения с результатами выполнения программы

# **Набор тестов**

Для проверки правильности алгоритмов составим тесты для всех возможных путей вычислений и выполним контрольные просчеты пользуясь независимыми от Delphi-среды вычислительными средствами (Калькулятор, WolframAlpha и др.). В данном случае воспользуемся системой «WolframAlpha».

***Тест 1.*** *Проверка правильности вычисления определённого интеграла.*

Пусть входные данные имеют следующие значения: C=0,26; D=-55,24; Eps=0,00001; Km=50. Результат вычисления Delphi-приложения показан на рис. 7 и равен -1,001206307. Результат вычисления в WolframAlpha показан на рис. 8 и равен -1,00121, а значит при погрешности Eps=0,00001 результат верен.

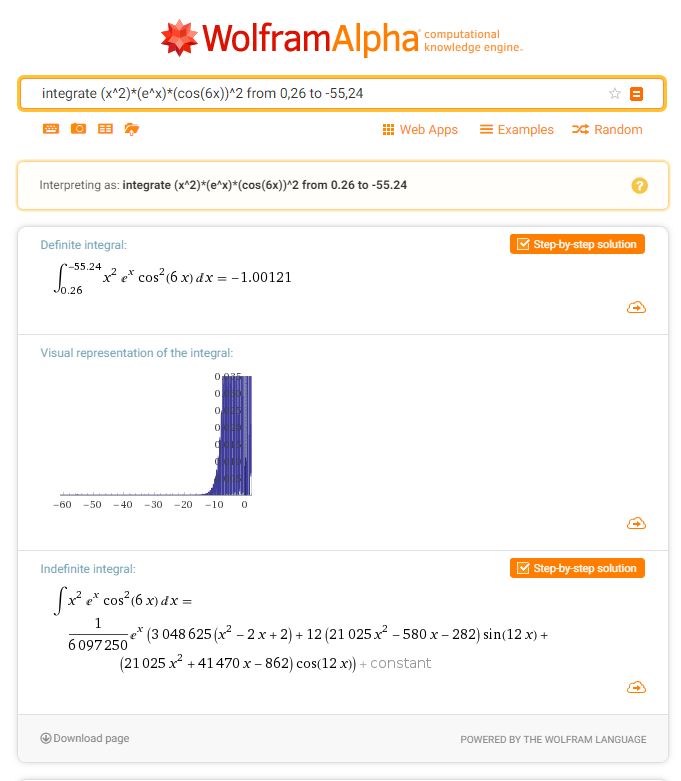


Рис. 8. Результат вычисления интеграла в WolframAlpha

***Тест 2.*** *Проверка ветви, вычисляющей значение функции при X < 1.*

Пусть входные данные имеют следующие значения: C=0,26; D=-55,24; Eps=0,00001; Km=50; X=-2; A=1. Результат вычисления Delphi-приложения показан на рис. 7 и равен -4,14016. Результат вычисления в WolframAlpha показан на рис. 9 и равен -4,14014. Следовательно, значение функции при таких входных параметрах вычислено правильно.

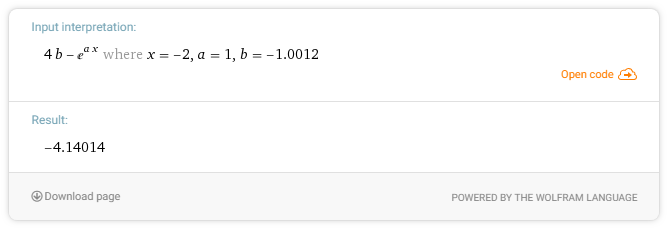


Рис. 9. Результат вычисления функции при X=-2 и A=1 в WolframAlpha

***Тест 3.*** *Проверка ветви, вычисляющей значение функции при X >= 1.*

Пусть входные данные имеют следующие значения: C=0,26; D=-55,24; Eps=0,00001; Km=50; X=2,5; A=2. Результат вычисления Delphi-приложения равен 6,3776. Результат вычисления в WolframAlpha показан на рис. 10 и равен 6,3776. Следовательно, значение функции при таких входных параметрах вычислено правильно.

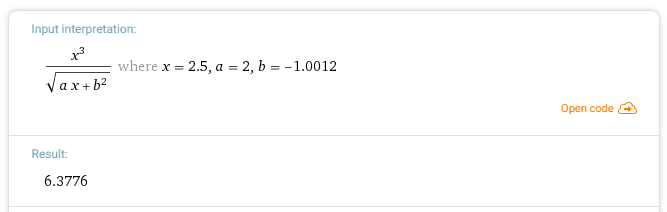


Рис. 10. Результат вычисления функции при X=2,5 и A=2 в WolframAlpha

**При вводе неверных значений Delphi-программа выводит сообщение об ошибке.**

При недостаточном числе итераций Km выводится сообщение об ошибке (рис. 11):

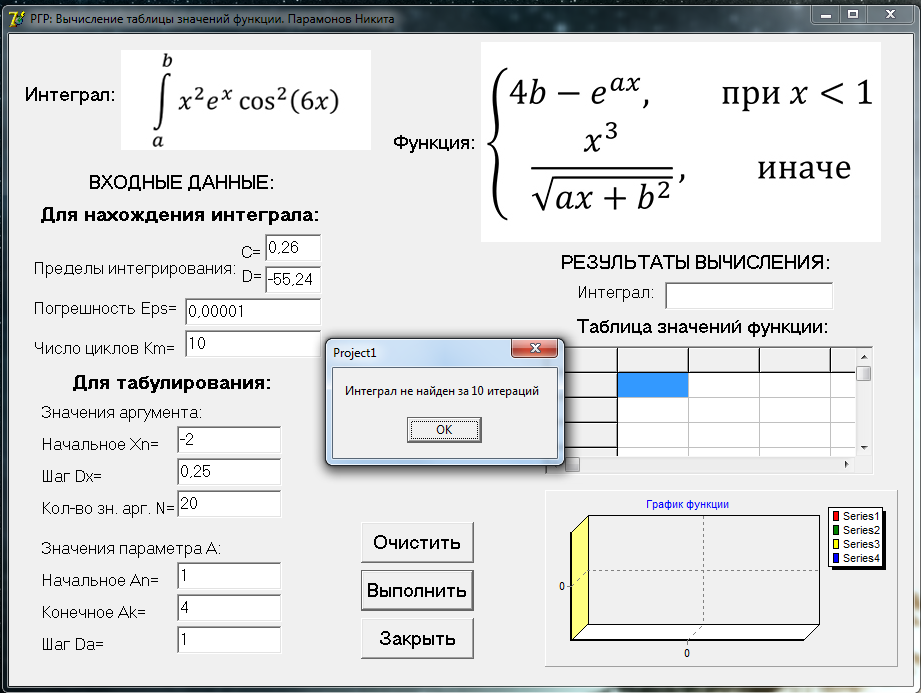


Рис. 11. Сообщение об ошибке при недостаточном числе итераций

Сообщение об ошибке также выводится при погрешности Eps большей или равной 1(рис. 12):

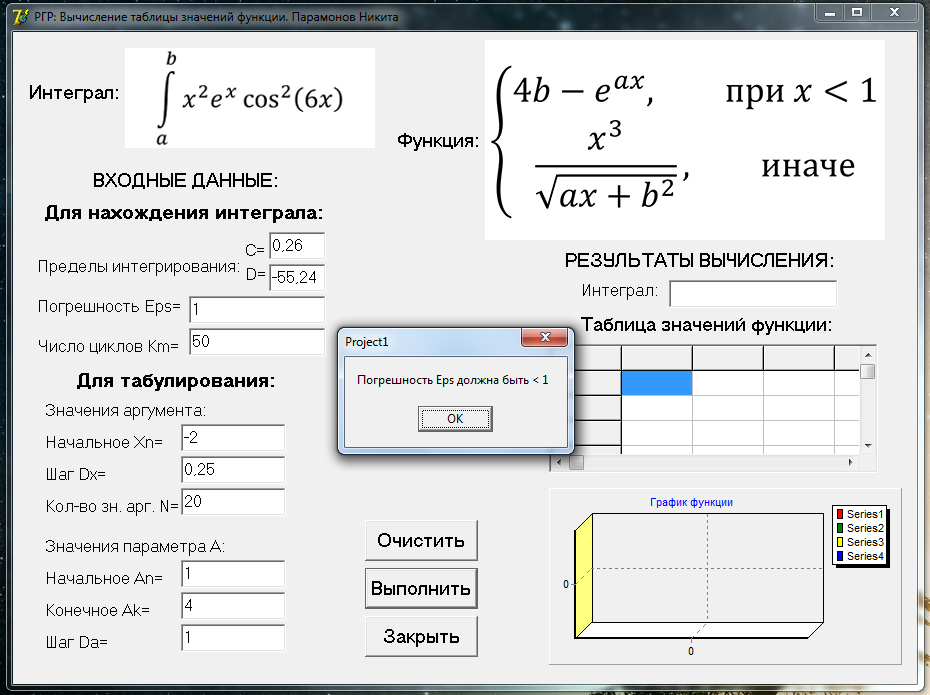


Рис. 12. Сообщение об ошибке при погрешности Eps большей или равной 1

Если при определённых значениях аргумента X и параметра A подкоренное выражение становится меньше 0, программа выводит строку «Err» в таблице значений функции (рис. 13), сообщающую о наличии ошибки в вычислениях (выход за область определения функции):



Рис. 13. Ошибка в вычислениях, если подкоренное выражение меньше 0

# **Выводы**

Анализ распечатки результатов выполнения программы показывает, что полученные значения функции приближенно совпадают с результатами, полученными для контрольных тестовых примеров с помощью WolframAlpha, что подтверждает работоспособность программы. Следовательно, программа правильно вычисляет заданную функцию по всем ветвям алгоритма и может быть использована для других значений аргументов и параметров функции.

# **10. Список использованной литературы**

* Учебное пособие «Решение алгебраических задач численными методами в среде Delphi» Авторы: Л.В. Кошелькова, А.И. Заковряшин
* Учебное пособие «Программное обеспечение, алгоритмизация и программирование». Авторы: Ю.И. Волощенко, Л.В. Кошелькова
* Онлайн справочник "Основы Delphi" (http://delphibasics.ru/)