Содержание

[Введение 4](#_Toc27344711)

[1. Описание метода решения задачи 5](#_Toc27344712)

[1.1. Численное интегрирование 5](#_Toc27344713)

[1.2. Метод парабол (Симпсона) 5](#_Toc27344714)

[2. Разработка структур данных 7](#_Toc27344715)

[3. Разработка алгоритмов 9](#_Toc27344716)

[4. Тестирование программы 13](#_Toc27344717)

[5. Разработка документации 17](#_Toc27344718)

[5.1. Назначение и условия применения программы 17](#_Toc27344719)

[5.2 Характеристика программы 17](#_Toc27344720)

[5.3 Обращение к программе 17](#_Toc27344721)

[5.4 Входные и выходные данные 18](#_Toc27344722)

[Заключение 19](#_Toc27344723)

[Библиографический список 20](#_Toc27344724)

[Приложение А. Листинг программы 21](#_Toc27344725)

Введение

Целью курсового проекта является разработке программной системы для решения математической задачи численного интегрирования методом парабол (метод Симпсона).

Задачами курсового проектирования является:

* изучение метода решения задачи;
* разработка алгоритма и структур данных;
* разработка программы, позволяющей решить задачу численного интегрирования;
* выполнить контрольный расчет в ручном режиме;
* провести тестирование работы программы на контрольном примере;
* выполнить тестирование работы программы на нескольких тестовых примерах;
* разработать документацию к программе.

Пояснительная записка к курсовому проекту состоит из следующих разделов:

* введение;
* описание метода решения задачи;
* разработка структур данных;
* разработка алгоритмов;
* тестирование программы;
* разработка документации;
* заключение;
* библиографический список;
* приложение (листинг программных модулей).

# Описание метода решения задачи

## Численное интегрирование

Для заданной функции f (x) вычислить значение определенного интеграла:

Желательно, чтобы метод численного интегрирования обладал следующими свойствами:

* Универсальность. Функция f (x) может быть задана в виде «черного ящика», как способ вычисления f (x) по данному x.
* Экономичность. Количество вычислений функции f (x) по возможности должно быть сведено к минимуму.
* Хорошая обусловленность. Неустранимые погрешности ∆f в значениях f (x) не должны приводить к значительной итоговой ошибке ∆I.

Численное интегрирование может применяться для:

* интегрирования функций, известных только в некоторых точках, например, полученных в результате измерений;
* интегрирования сложных выражений, не имеющих элементарных первообразных, либо имеющих слишком громоздкие выражения для них;
* построения методов численного решения уравнений в обыкновенных и частных производных (методы конечных элементов, интегро-интерполяционные методы).

## Метод парабол (Симпсона)

График подынтегральной функции приближается не ломаной линией, а маленькими параболами. Сколько промежуточных отрезков – столько и маленьких парабол. Если взять те же три отрезка, то метод Симпсона даст ещё более точное приближение, чем метод прямоугольников или метод трапеций.

Рассмотрим определенный интеграл http://www.mathprofi.ru/h/formula_simpsona_metod_trapecij_clip_image017_0000.gif, где http://www.mathprofi.ru/h/formula_simpsona_metod_trapecij_clip_image019_0000.gif – функция, непрерывная на отрезке http://www.mathprofi.ru/h/formula_simpsona_metod_trapecij_clip_image021_0001.gif.  Проведём разбиение отрезка http://www.mathprofi.ru/h/formula_simpsona_metod_trapecij_clip_image021_0002.gif на **чётное** количество **равных** отрезков. Чётное количество отрезков обозначают через http://www.mathprofi.ru/h/formula_simpsona_metod_trapecij_clip_image120.gif.

На практике отрезков может быть:

**два**: http://www.mathprofi.ru/h/formula_simpsona_metod_trapecij_clip_image122.gif

**четыре**: http://www.mathprofi.ru/h/formula_simpsona_metod_trapecij_clip_image124.gif

**восемь**: http://www.mathprofi.ru/h/formula_simpsona_metod_trapecij_clip_image126.gif

**десять**: http://www.mathprofi.ru/h/formula_simpsona_metod_trapecij_clip_image128.gif

**двадцать**: http://www.mathprofi.ru/h/formula_simpsona_metod_trapecij_clip_image130.gif и так далее.

Итак, наше разбиение имеет следующий вид:

http://www.mathprofi.ru/h/formula_simpsona_metod_trapecij_clip_image135.gif

Точки http://www.mathprofi.ru/h/formula_simpsona_metod_trapecij_clip_image137.gif называют **узлами**.

**Формула Симпсона** для приближенного вычисления определенного интеграла имеет следующий вид:

Общая формула Симпсона, где:

http://www.mathprofi.ru/h/formula_simpsona_metod_trapecij_clip_image141.gif – длина каждого из маленьких отрезков или **шаг**;

http://www.mathprofi.ru/h/formula_simpsona_metod_trapecij_clip_image037_0001.gif – значения подынтегральной функции в точках http://www.mathprofi.ru/h/formula_simpsona_metod_trapecij_clip_image137_0000.gif.

Детализируя это нагромождение, разберу формулу подробнее:

http://www.mathprofi.ru/h/formula_simpsona_metod_trapecij_clip_image143.gif – сумма первого и последнего значения подынтегральной функции;

http://www.mathprofi.ru/h/formula_simpsona_metod_trapecij_clip_image145.gif – сумма членов с чётными индексами умножается на 2;

http://www.mathprofi.ru/h/formula_simpsona_metod_trapecij_clip_image147.gif – сумма членов с нечётными индексами умножается на 4.

# Разработка структур данных

Для решения задачи необходимо определить типы данных, которые будут использоваться для хранения исходных и промежуточных данных, а также результатов вычислений.

Для реализации подсчета числа итераций необходимы данные целого типа. Язык *Free Pascal* предоставляет возможность использования следующих целочисленных типов (Таблица 1):

Таблица 1 – Целые типы данных

|  |  |  |
| --- | --- | --- |
| **Тип** | **Диапазон значений** | **Размер, байт** |
| Byte | 0 .. 255 | 1 |
| Shortint | -128 .. 127 | 1 |
| Smallint | -32768 .. 32767 | 2 |
| Word | 0 .. 65535 | 2 |
| Integer | smallint или longint | 2 или 4 |
| Cardinal | longword | 4 |
| Longint | -2147483648 .. 2147483647 | 4 |
| Longword | 0..4294967295 | 4 |
| Int64 | -9223372036854775808 .. 9223372036854775807 | 8 |
| QWord | 0 .. 18446744073709551615 | 8 |

Диапазона значений типов *Byte*, *ShortInt* может не хватить для подсчета количества итераций. Диапазон значений типов *Cardinal*, *Longint*, *Longword*, *Int64*, *QWord* явно избыточен для решения поставленной задачи. Количество итераций не может быть отрицательным, поэтому тип *Word* предпочтительнее, чем *Smallint*.

Для реализации математических вычислений необходимы данные вещественного типа. Язык *Free Pascal* предоставляет возможность использования вещественных типов данных:

Таблица 2 – Вещественные типы данных

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип** | **Диапазон значений** | **Значимых разрядов** | **Размер, байт** |
| Real | Зависит от платформы | Зависит от платформы | 4 или 8 |
| Single | 1.5E-45 .. 3.4E38 | 7-8 | 4 |
| Double | 5.0E-324 .. 1.7E308 | 15-16 | 8 |
| Extended | 1.9E-4932 .. 1.1E4932 | 19-20 | 10 |
| Comp | -2E64+1 .. 2E63-1 | 19-20 | 8 |
| Currency | -922337203685477.5808 | 922337203685477.5807 | 8 |

Тип данных *Real* зависит от платформы, и использовать его нежелательно. Тип *Currency* предназначен для операций с числами с фиксированной точностью (операции с денежными величинами). Тип *Comp* трактуется как вещественное число без дробной части.

Точность данных типа *Single* недостаточна для решения задач вычислительной математики, поэтому предпочтение следует отдать типам *Double* или *Extended*.

# Разработка алгоритмов

Процесс численного интегрирования является итерационным процессом, поэтому за основу в качестве обобщенной схемы возьмем общую схему алгоритма итерационного процесса, представленную на рисунке (Рисунок 3.1).



Рисунок 3.1 – Общая схема итерационного процесса

Для метода численного интегрирования методом Симпсона входными данными алгоритма будут:

* максимальное количество итераций *MaxIterCount*;
* пределы интегрирования данной функции *a*, *b*;
* точность вычисления *EPS*.

Блок подготовки итерационного процесса заключается в том, что значение счетчика итераций становится равным 2.

Тело цикла будет включать в себя следующие действия:

* определение длины шага разбиения: *h=abs((b-a)/(n))*;
* вычисление значения функции: *f=(h/3)\*(Calc(a)+Calc(b)+2\*Calc2(a,b,h,true)+4\*Calc2(a,b,h,false))*

Условием завершение цикла является:

* либо достижение максимального количества разбиений (*MAX\_ITER\_COUNT=n*);
* либо достижение заданной точности по правилу Рунге:

На рисунке (Рисунок 3.2) представлен алгоритм вычисления значения суммы чётных или нечётных индексов точек.

На рисунке (Рисунок 3.3) представлен алгоритм вычисления значения функции в данной точке.

Общий алгоритм (Рисунок 3.4) будет заключаться в следующем:

* ввод исходных данных (*a,b,h,isEven*);
* проверка исходных данных;
* выполнение ;
* вывод результатов.

Исходные данные корректные, если:

* число итераций положительно: *MaxIterCount* > 0;
* левая граница интеграла a, правая граница - b;



Рисунок 3.2 – Алгоритм вычисления значения суммы чётных или нечётных индексов точек.



Рисунок 3.3 – Алгоритм вычисления значения функции в данной точке



Рисунок 3.4 – Общая схема алгоритма решения задачи

# Тестирование программы

Тестирование программного обеспечения — процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом

Тестирование – это процесс выполнения программы с целью нахождения ошибок.

Для тестирования программы нахождения значения интеграла методом Симпсона необходимо:

* выбрать интеграл, который легко найти и провести тестирования работы программы;
* выбрать интеграл, найти его значение при помощи другого программного средства и проверить найденное решение с тем, что будет выдано программой.

Проведем тестирование программы для интеграла sin(x). Значение функции на интервале равно 0.4596976941318602, а для интервала - 0.956449142415282.

Зададим предел интегрирования и выполним тестовый запуск программы:

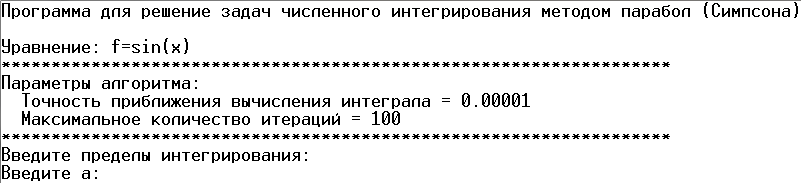


Рисунок 4.1 – Запуск программы

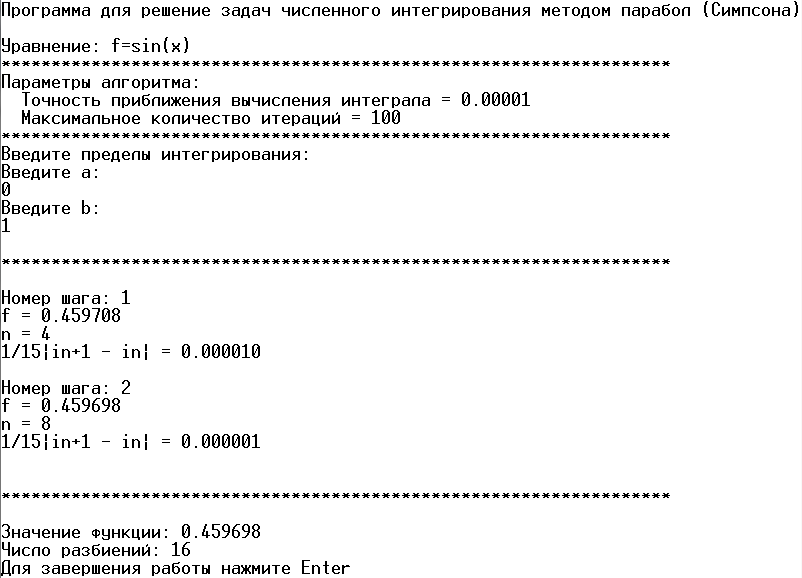


Рисунок 4.2 – Результаты выполнения программы

В результате запуска требуемый корень был найден с заданной точность. Повторим тестирование для промежутка .

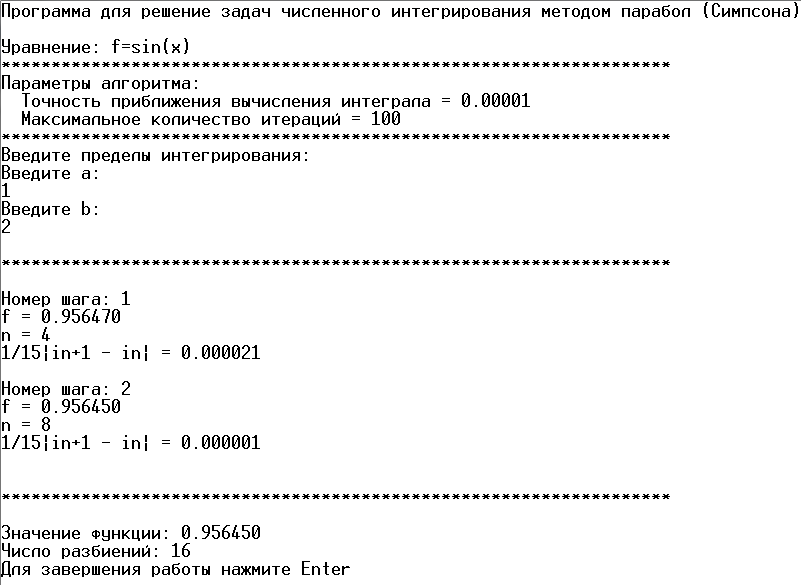


Рисунок 4.2 – Результаты выполнения программы для другого предела интегрирования

В результате запуска интегралы были найдены с заданной точность.

Проведем тестирование для функции, интеграл которой нельзя найти. Для этого построим график функции и таблицу ее значений в определённых точках.

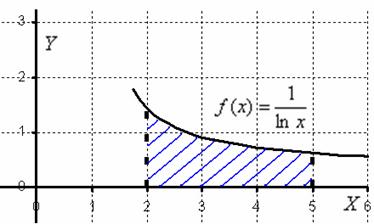


Рисунок 4.4 – График функции

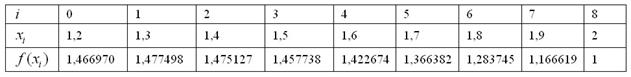


Рисунок 4.5 – Таблица значений функции

Выберем пределы интегрирования и выполним тестовый запуск программы:

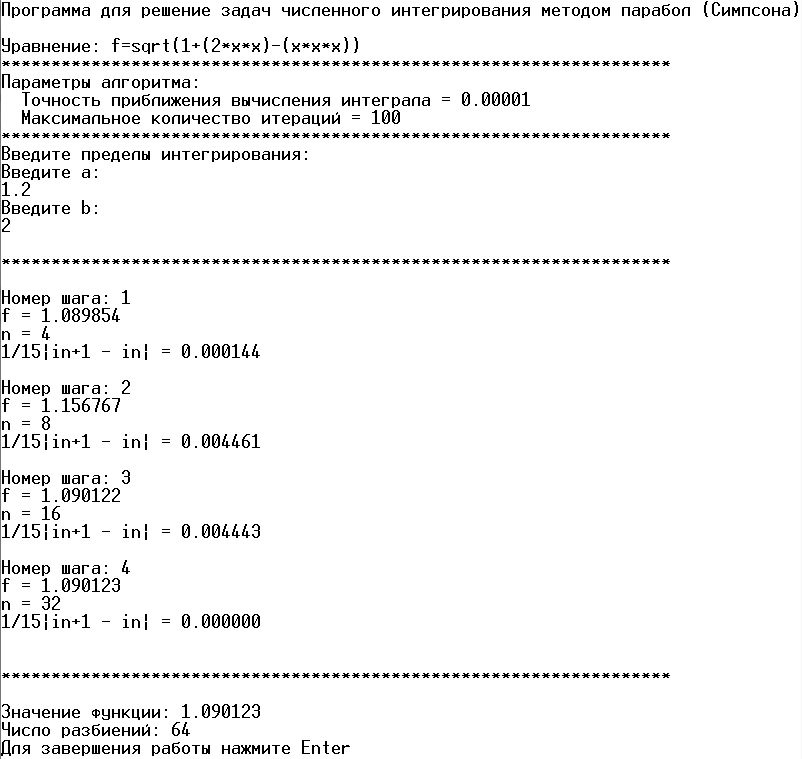


Рисунок 4.2 – Результаты выполнения программы для интеграла

Результаты запуска показывают, что интеграл найден с заданной точностью.

По результатам испытаний считаем, что программа работает корректно.

# Разработка документации

В данном разделе приведено руководство программиста по использованию программы численного интегрирования методом парабол *Simpson*.

## 5.1. Назначение и условия применения программы

Программа *Simpson* предназначена для численного интегрирования методом парабол. Программа позволяет задавать пределы интегрирования и может быть применена для решения любых интегралов.

Для выполнения программы необходимо наличие операционной системы *Windows*. Особые требования к составу периферийных устройств не предъявляются.

Персонал, использующий программу (программист) должен обладать практическими навыками работы с текстовым пользовательским интерфейсом.

## 5.2 Характеристика программы

Программа *Simpson* является независимым программным обеспечением и обеспечивает решение задач вычислительной математики – численного интегрирования методом парабол.

Работа программы осуществляется в консольном режиме.

Размер исполняемого файла *Simpson*.*exe* – 133 961 байт.

## 5.3 Обращение к программе

Запуск программы осуществляется стандартным для операционной системы способом – двойным щелчком левой кнопки мыши на исполняемом файле *Simpson*.*exe* или выполнением команды в *Simpson* в командной строке в каталоге программы.

После запуска в окне консоли (Рисунок 5.1) появляется следующая информация:

* назначение программы;
* параметры алгоритма.

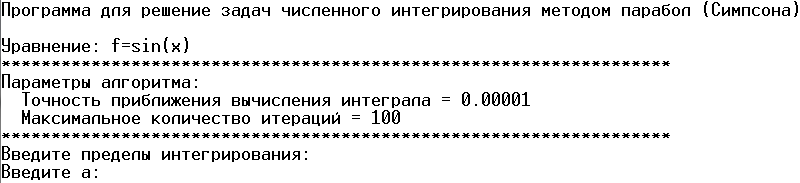


Рисунок 5.1 – Результат запуска программы Simpson

Пользователь должен ввести сначала начало придела интегрирования, затем его конец.

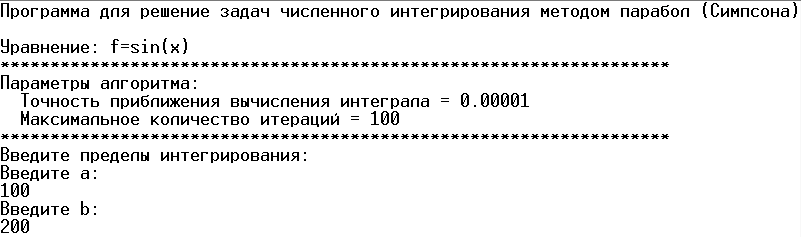


Рисунок 5.2 – Ввод приделов интегрирования в программе Simpson

Если введены корректные данные, то осуществляется расчет численного интегрирования методом парабол.

## 5.4 Входные и выходные данные

Входными данными программы являются:

* исходная функция;
* нижний предел интегрирования;
* верхний предел интегрирования;
* точность вычисления;

Исходное уравнение задается в виде функции. Точность вычисления задаётся при помощи констант. Верхняя и нижняя границы интегрирования вводятся пользователем в процессе работы программы.

Выходными данными программы являются:

* значение интеграла*;*
* число разбиений.

Заключение

Библиографический список

1. ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов.
2. ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов.
3. ГОСТ 19.104-78\* ЕСПД. Основные надписи.
4. ГОСТ 19.105-78\* ЕСПД. Общие требования к программным документам.
5. ГОСТ 19.106-78\* ЕСПД. Общие требования к программным документам, выполненным печатным способом.
6. ГОСТ 19.504-79\* ЕСПД. Руководство программиста. Требования к содержанию и оформлению.
7. ГОСТ 19.604-78\* ЕСПД. Правила внесения изменения в программные документы, выполненные печатным способом.
8. Самарский А.А., Гулин А.В. Численные методы. - М.: 1989.
9. Калиткин Н.Н. Численные методы. - М.: Наука, 1978.
10. Демидович Б.П., Марон И.А. Основы вычислительной математики. - М.: 1970.
11. Копченова Н.В., Марон И.А. Вычислительная математика в примерах и задачах . - М.: Наука, 1972.
12. Мудров А.Е. Численные методы для ПЭВМ на языках Бейсик, Фортран, Паскаль. – Томск: МП «Раско», 1991.
13. Майерс Г.[en] Надежность программного обеспечения. М: Мир, 1980

Приложение А. Листинг программы

// Курсовая работа, Решение задач численного интегрирования методом парабол

// Выполнил: ст. гр. 848 Федотов М.Ф.

// Проверил: доц. каф. КТ Наумов Д.А.

// Дата написания: 12 декабря 2019

{\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}

{ }

{ Решение задач численного интегрирования методом парабол }

{ Copyright (c) 2019 ФГБОУ РГРТУ им. В.Ф. Уткина }

{ ФВТ/кафедра КТ }

{ }

{ Разработчик: ст. гр. 848 Федотов М.Ф. }

{ Модифицирован: 12 декабря 2019 }

{ }

{\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}

program Simpson;

//установка кодовой страницы для отображения русских символов

{$codepage utf8}

type

//тип данных для вычислений

TValue = Double;

const

EPS = 0.00001; //точность вычисления

MAX\_ITER\_COUNT = 100; //максимальное количество итераций

var

a,b,h,f,f2: TValue;

n: Integer; //число разбиений

//функция для вычисления значения при заданном x

// x - аргумент функции

// результат - значение sin(x)

function Calc(x:TValue):TValue;

begin

Calc:=sin(x);

end;

//функция для вычисления значения суммы чётных или нечётных индексов точек

// x - аргумент функции

function Calc2(a,b,h:TValue; isEven:Boolean):TValue;

var x:TValue;

begin

Calc2:=0;

if isEven then //если суммируем только чётные индексы точек

x:=a+(2\*h)//начинаем вычисения с точки с нечётным индексом

else//если суммируем только нечётные индексы точек

x:=a+h;//начинаем вычисения с точки с чётным индексом

while x<b do//пока не дойдём до конца предела интегрирования

begin

Calc2:=Calc2+Calc(x);//складываем значение функции в точке

x:=x+(2\*h);//переходим на следующюю точку с чётным (нечётным) индексом

end;

end;

begin

writeln('Программа для решение задач численного интегрирования методом парабол (Симпсона)');

writeln('Уравнение: f=sin(x)');

writeln('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

writeln('Параметры алгоритма:');

writeln(' Точность приближения вычисления интеграла = ', EPS:0:5);

writeln(' Максимальное количество итераций = ', MAX\_ITER\_COUNT);

writeln('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

writeln('Введите пределы интегрирования:');

n:=2;//начинаем разбиение с 2

writeln('Введите a:');

readln(a);

writeln('Введите b:');

readln(b);

writeln;

writeln('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

writeln;

f:=EPS;

repeat

f2:=f;

h:=abs((b-a)/(n));//длина каждого из маленьких отрезков или шаг

//формула Симпсона для приближенного вычисления определенного интеграла

f:=(h/3)\*(Calc(a)+Calc(b)+2\*Calc2(a,b,h,true)+4\*Calc2(a,b,h,false));

if ln(n)/ln(2)>1 then//выводим номер шага в цикле

begin

writeln('Номер шага: ', ln(n)/ln(2)-1:0:0);

writeln('f = ', f:6:6);

writeln('n = ', n);

writeln('1/15|in+1 - in| = ', 1/15\*abs(f-f2):6:6);

writeln;

end;

n:=n\*2;//увеличиваем значение в 2 раза

until (1/15\*abs(f-f2)<EPS) or (MAX\_ITER\_COUNT<n);//Правило Рунге для метода Симпсона

writeln;

writeln('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

writeln;

writeln('Значение функции: ', f:6:6);

writeln('Число разбиений: ', n);

writeln('Для завершения работы нажмите Enter');

readln;

end.