МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Белорусский национальный технический университет

Кафедра «Металлургия черных и цветных сплавов»

И. В. Рафальский
А. В. Арабей
П. Е. Лущик

ПРИКЛАДНАЯ ИНФОРМАТИКА

Пособие

Минск
БНТУ
2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Белорусский национальный технический университет

Кафедра «Металлургия черных и цветных сплавов»

И. В. Рафальский
А. В. Арабей
П. Е. Лущик

ПРИКЛАДНАЯ ИНФОРМАТИКА

Пособие
для студентов специальности 1-42 01 01
«Металлургическое производство и материалообработка
(по направлениям)»

Рекомендовано учебно-методическим объединением по образованию в области металлургического оборудования и технологий

Минск
БНТУ
2021

УДК 004(075.8)
ББК 32.97я7
Р26

Р е ц е н з е н т ы:
профессор кафедры ЭВМ УО «Белорусский государственный университет информатики и радиоэлектроники»,
д-р техн. наук *М. М. Татур*;
кафедра «Материаловедение и проектирование технических систем» Белорусского государственного технологического университета,
зав. кафедрой, канд. техн. наук, доцент *Д. В. Куис*

Рафальский, И. В.

Р26 Прикладная информатика : пособие для студентов специальности 1-42 01 01 «Металлургическое производство и материалообработка (по направлениям)» / И. В. Рафальский, А. В. Арабей, П. Е. Лущик. – Минск : БНТУ, 2021. – 38 с.

ISBN 978-985-583-672-9.

Пособие предназначено для закрепления и углубления теоретической базы знаний студентов при изучении дисциплин «Информатика», «Прикладная информатика», а также приобретения практических навыков работы с современными электронно-вычислительными и программными средствами, алгоритмами и компьютерными методами поиска, использования, хранения и обработки информации.

УДК 004(075.8)
ББК 32.97я7

ISBN 978-985-583-672-9

1. СРЕДА РАЗРАБОТКИ LAZARUS

Цель работы: изучить основные компоненты среды разработки Lazarus для создания графических приложений.

Теоретическая часть

Lazarus – интегрированная среда разработки (англ. Integrated Development Environment – IDE) программного обеспечения (ПО) на основе компилятора Free Pascal с открытым исходным кодом, которая благодаря широкому функционалу и обширной библиотеке компонентов LCL (Lazarus Component Library) позволяет создавать программные приложения с графическим интерфейсом, в том числе для работы с базами данных, и Интернет-приложения.

IDE Lazarus (рис. 1.1) имеет удобный графический интерфейс для быстрой разработки программ и включает текстовый редактор исходного программного кода, кроссплатформенный компилятор (поддерживает Linux, Windows, MacOS и др.), средства автоматизации сборки проекта, отладчик.

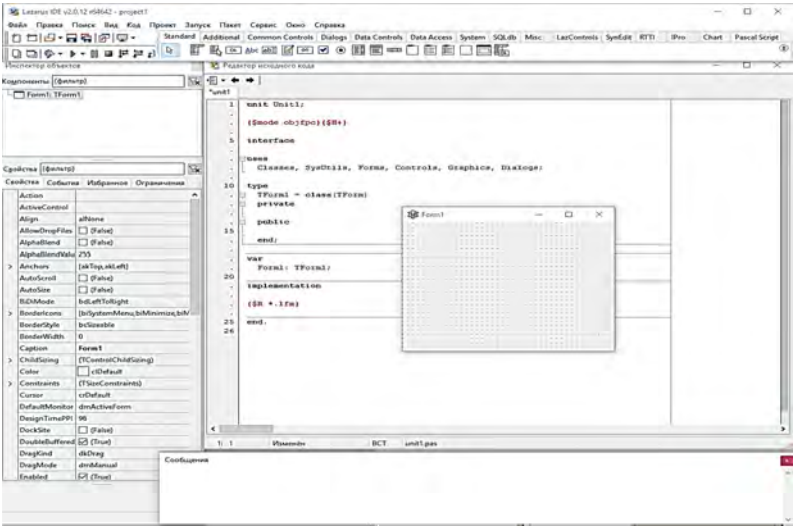


Рис. 1.1. Внешний вид IDE Lazarus

Среда разработки ПО Lazarus состоит из нескольких основных окон:

– главное окно, обеспечивающее процесс разработки ПО (управление файлами, редактирование, компиляцию и т. д., рис. 1.2);

Рис. 1.2. Главное окно IDE Lazarus

– редактор исходного программного кода (рис. 1.3);

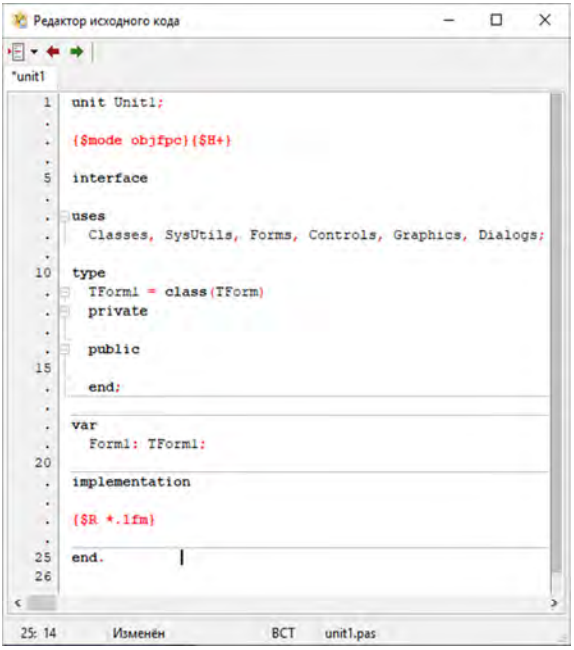


Рис. 1.3. Окно редактора исходного кода

– инспектор объектов, обеспечивающий возможность просмотра и редактирования свойств и методов используемых объектов (рис. 1.4);

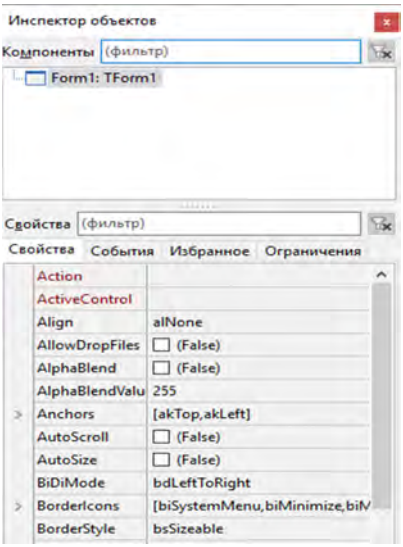
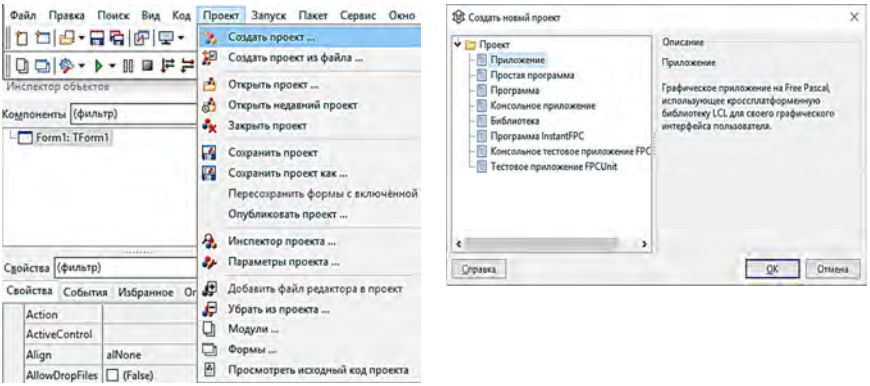


Рис. 1.4. Инспектор объектов

– окно сообщений, обеспечивающее вывод сообщений компилятора, компоновщика и отладчика.

Порядок создания нового приложения в среде Lazarus представлен на рис. 1.5.



а *б*

Рис. 1.5. Создание нового приложения в среде Lazarus:
а – главное меню; *б* – окно выбора типа проекта

Пример создания приложения для построения графика функции cos(2x).

При создании нового проекта (приложения) появляется пустая форма (рис. 1.6).

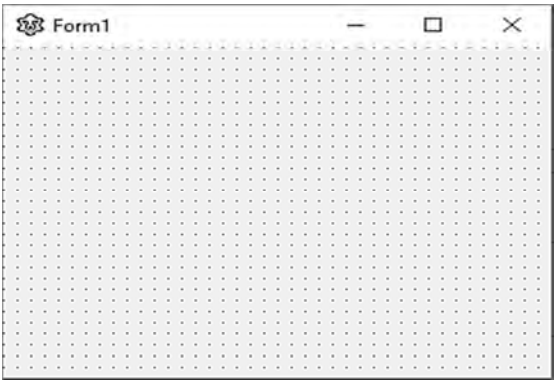


Рис. 1.6. Форма нового проекта

В процессе визуального программирования из набора компонентов библиотеки LCL на форму переносятся и настраиваются нужные визуальные компоненты, формируя дизайн программы. Свойства и методы используемых компонентов отображаются в инспекторе объектов. Поместим на форму нового проекта компонент TAChart из палитры компонентов на странице Chart (рис. 1.7, 1.8).

Рис. 1.7. Палитра компонентов Chart

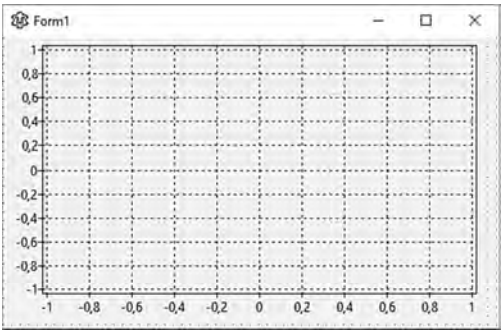


Рис. 1.8. Форма проекта с помещенным компонентом TAChart

Двойным кликом мыши на помещенном в форму компоненте TAChart вызовем окно редактора диаграмм, добавим диаграмму по математическому выражению (рис. 1.9), а с помощью Инспектора объектов (рис. 1.10) зададим математическое выражение для построения графика функции cos(2*x*), минимальное и максимальное значения аргумента.

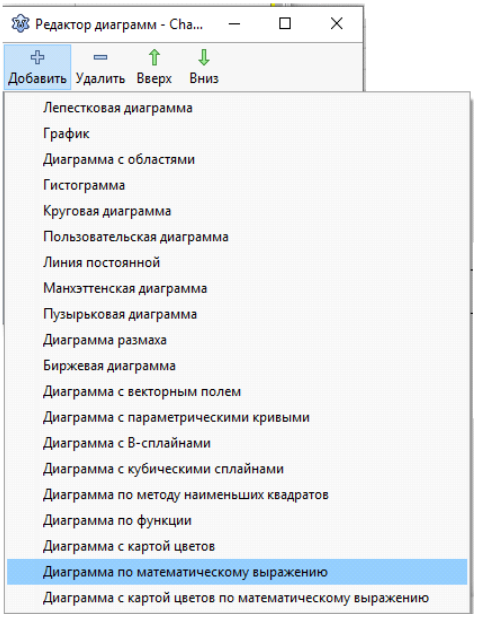


Рис. 1.9. Редактор диаграмм

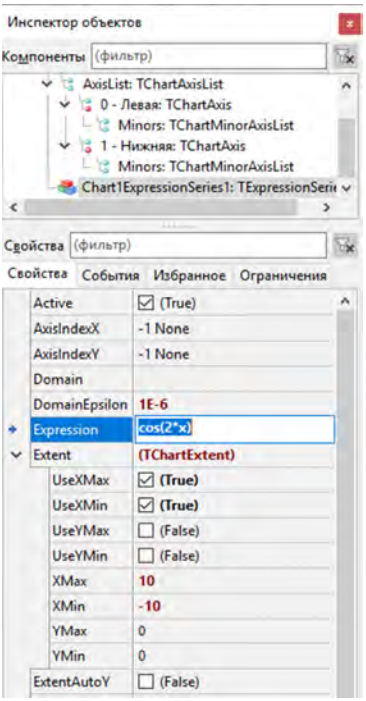


Рис. 1.10. Инспектор объектов

После сохранения файлов проекта и его компиляции получаем исполняемое приложение, окно которого показано на рис. 1.11.

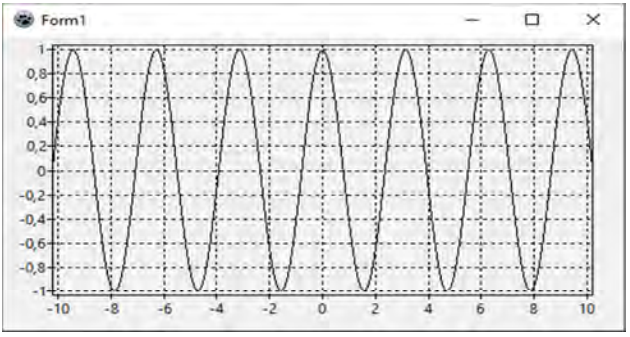


Рис. 1.11. Окно приложения с графиком функции cos(2*x*)

Задание для самостоятельной работы

1. Разработать программу, обеспечивающую построение графика заданной функции (табл. 1.1).

2. Определить число корней уравнения $f(x) = 0$.

3. Составить отчет о работе.

Таблица 1.1

Номер варианта	$f_1(x)$	x_0	x_k
1	$2 - \exp(\|x - 1\|)$	–1	1
2	$\cos(1 - e^x)$	–2	2
3	$\sin(1 - e^x)$	–1	2
4	$\sin(0{,}2x - 1)$	4	6
5	$\ln(\|x - 2\|)$	–1	1,5
6	$\ln(\|0{,}5x - 1{,}1\|)$	–1	1
7	$\cos(0{,}1 - 1/e^x)$	–2	0
8	$\ln(1 - \cos(x))$	–2	–0,5
9	$\ln(0{,}8 - \sin(x))$	–2	1
10	$\ln(1{,}2 - \sin^2(x))$	–2	0
11	$0{,}1\cos(x) - \|x\|^{2/3}$	–2	1
12	$0{,}5 - 0{,}1/e^x$	–2	0
13	$0{,}1/e^x - 0{,}6$	–2	1
14	$0{,}8 + \sin(0{,}2x - 1)$	–2	1
15	$0{,}5 + \cos(x - 1)$	–2	1
16	$0{,}5 - \|\sin(x - 1)\|^{2/3}$	–1	1
17	$0{,}5 - \|\cos(x - 1)\|^{2/3}$	–2	1
18	$2 - \|x^2 + e^x\|$	–2	0
19	$\|x^3 + e^{0{,}5x}\| - 3$	–2	1
20	$0{,}1x^2 - e^{0{,}2x}$	–1	1

2. ВЫЧИСЛЕНИЕ КОРНЕЙ УРАВНЕНИЯ

Цель работы: изучить численные (приближенные) методы вычисления корней уравнений.

Теоретическая часть

Приближенное (численное) определение корней уравнений проводится в два этапа:

1. Отделение корней, т. е. установление достаточно малых отрезков, в каждом из которых содержится только один корень уравнения.

2. Уточнение приближенного значения корней до некоторой заданной степени точности.

Любое уравнение можно представить в виде $f(x) = 0$. Процесс отделения корней начинается с установления знаков функции в граничных точках заданного интервала x_0 и x_k. Затем выделяются отрезки, на границе которых функция меняет знак на противоположный. Выделенные отрезки содержат только один корень данного уравнения.

Уточнение приближенного значения корня можно проводить различными методами, например методом деления отрезка пополам (метод бисекций), методом касательных и др.

Рассмотрим решение задачи методом деления отрезка пополам (рис. 2.1). Пусть выделен отрезок $[a, b]$, на котором находится корень x^*, т. е. $a < x^* < b$. В качестве начального приближения корня x_0 принимаем середину расчетного отрезка $x_0 = (a + b) / 2$. Далее исследуем значения функции: если $f(x_0) = 0$, то x_0 является корнем уравнения, т. е. $x^* = x_0$. Если $f(x_0) \neq 0$, то выбираем одну из половин отрезка $[a, x_0]$ или $[x_0, b]$, на концах которой функция $f(x)$ имеет противоположные знаки, т. е. содержит искомый корень, поэтому его принимают в качестве нового расчетного отрезка $[a, b]$. Вторая половина отрезка, на концах которого знак $f(x)$ не меняется, отбрасывается. Отрезок, на концах которого функция $f(x)$ име-

ет противоположные знаки, вновь делится пополам и вычисляется новое приближение корня и т. д. Итерационный процесс продолжается до тех пор, пока длина отрезка после k-й итерации не станет меньше некоторого заданного малого числа (погрешности) ε, т. е. $|b - a| \le \varepsilon$. За искомое значение корня принимают полученное приближение $x^* = x_k$ (говорят, что решение заданного уравнения найдено с точностью ε).

Анализ функции. В основе алгоритма отделения корней лежит проверка условия знакопеременности функции на концах интервала. На основе анализа графика заданной функции выбирается интервал, на концах которого функция имеет разный знак.

Уточнение значений корней. Структура исходного программного кода для уточнения значений корней уравнения $\cos(2x) = 0$, $-1 < x < 0$ на языке Паскаль приведена ниже:

```
Var
{a, b: границы расчетного интервала}
{Eps: заданная точность вычислений}
a,b,eps, x,s,y: real;
Function f(x: real): real;
Begin
f:=cos(2*x); {записывается выражение для функции f(x)}
End;
Begin
a:=-1;
b:=0;
Eps:=0,0001;
Repeat
x:=(a+b)/2;
y:=f(x);
s:=f(a);
If y*s<0 then b:=x else a:=x;
Until (y=0.0) or (abs(a-b)<=Eps);
End;
```

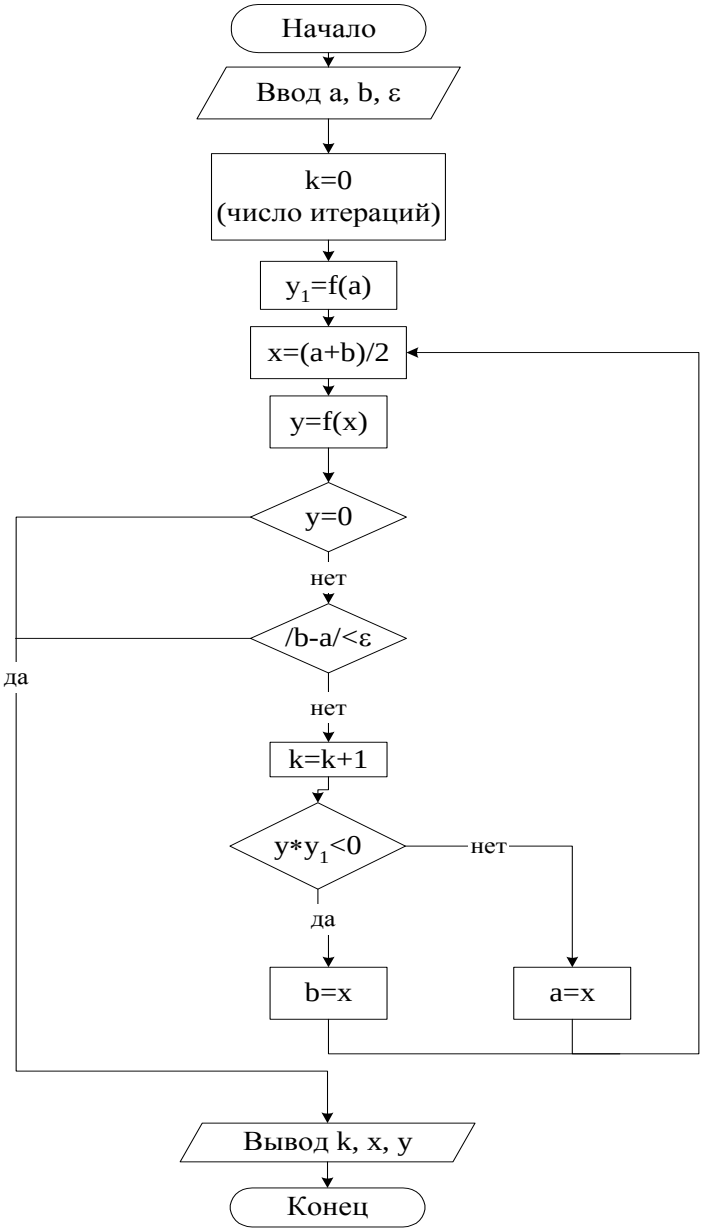


Рис. 2.1. Блок-схема алгоритма метода деления отрезка пополам

Рассмотрим пример создания графического приложения для определения корней уравнения методом деления отрезка пополам. Создадим новый проект и поместим на форму компоненты TButton, TEdit (страница Standart) и TAChart, как показано на рис. 2.2.

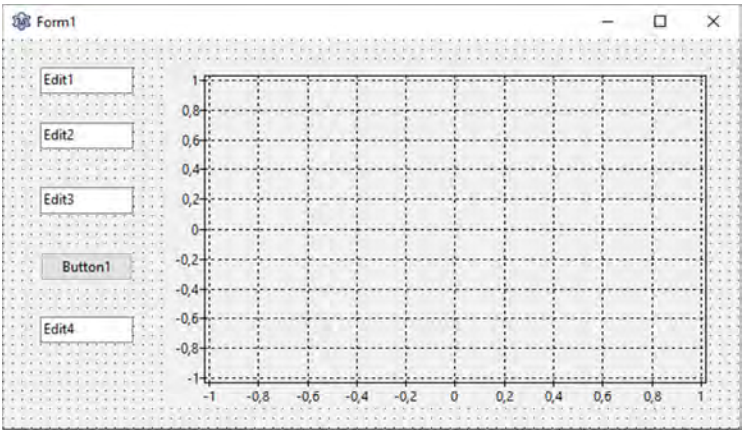


Рис. 2.2. Форма проекта с помещенными на нее компонентами TButton, TEdit, TAChart

Двойным кликом мыши на помещенном в форму компоненте TAChart вызовем окно редактора диаграмм, добавим диаграмму по математическому выражению, а с помощью Инспектора объектов зададим математическое выражение для построения графика заданной функции, минимальное и максимальное значения аргумента.

Компоненты TEdit используются для организации ввода и вывода данных. Сигналом завершения ввода и начала обработки введенных данных служит нажатие кнопки Button1.

Компонент TButton (кнопка Button1) является элементом управления, предназначенным для запуска вычислений. Двойной щелчок по кнопке мышью вызывает событие OnClick в обработчике событий и генерирует программный код процедуры вычислений:

```
procedure TForm1.Button1Click(Sender: TObject);
begin

end;
```

Вставляем программный код в процедуру TForm1.Button1Click, присваивая переменным *a*, *b*, *Eps* значения StrToFloat(Edit1.Text), StrToFloat(Edit2.Text), StrToFloat(Edit3.Text), а свойству Text компонента Edit4. – значение FloatToStr(x):

```
procedure TForm1.Button1Click(Sender: TObject);
Var
a,b,eps,x,s,y: real;
k: integer;
Function f(x: real): real;
Begin
{записывается выражение для функции f(x)}
f:=cos(2*x);
End;
Begin
a:=StrToFloat(Edit1.Text);
b:=StrToFloat(Edit2.Text);
Eps:=StrToFloat(Edit3.Text);
k:=0;
Repeat
inc(k);
x:=(a+b)/2;
y:=f(x);
s:=f(a);
If y*s<0 then b:=x else a:=x;
Until (y=0.0) or (abs(a-b)<=Eps);
Edit4.Text:=FloatToStr(x);
End;
```

Сохраните файлы проекта, скомпилируйте и запустите программу. Введите входную информацию. Щелчок по кнопке мышью приведет к обработке данных и выводу результата вычислений (рис. 2.3).

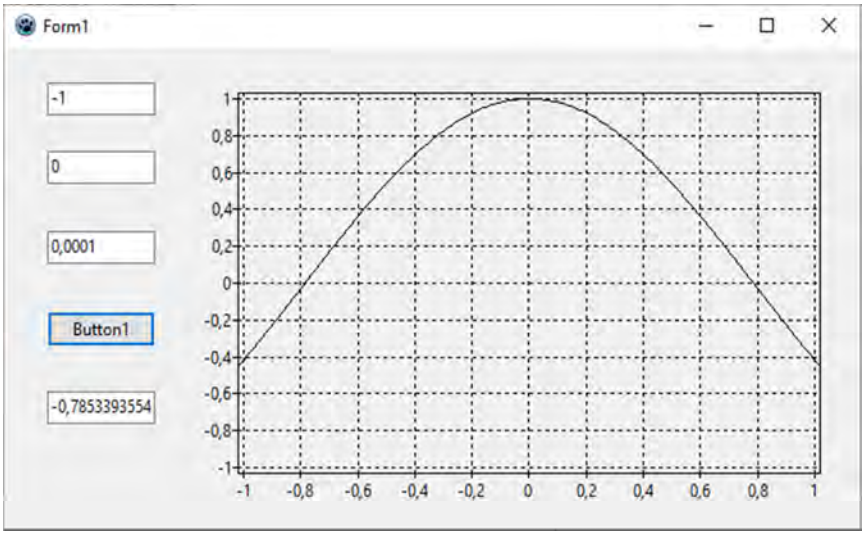


Рис. 2.3. Окно приложения с графиком функции
и результатом вычислений

Задание для самостоятельной работы

1. Определить количество корней исходного нелинейного уравнения графическим методом и построить график функции.
2. Составить программы, реализующие алгоритмы численных методов вычисления корней уравнений.
3. Решить уравнение с точностью ε = 0,001 в соответствии с заданным вариантом (табл. 2.1).
4. Составить отчет о работе.

Таблица 2.1

Варианты заданий для самостоятельной работы

Номер варианта	$f_1(x)$	$f_2(x)$	x_0	x_k	n	$E_{зад}$
1	$2 - \exp(\|x-1\|)$	$2\sin^2(e^x) - \sin(x^2)$	–1	1	100	0,01
2	$\cos(1-e^x)$	$\cos^2(e^x) - 0{,}2\sin(x)$	–2	2	50	0,001
3	$\sin(1-e^x)$	$\cos^3(3x) - 0{,}2\cos^2(x)$	–1	2	50	0,005
4	$\sin(0{,}2x-1)$	$\sin^3(3x) - 2\sin^2(2x)$	4	6	100	0,005
5	$\ln(\|x-2\|)$	$\cos^3(x) + \sin(x-0{,}2)$	–1	1,5	100	0,001
6	$\ln(\|0{,}5x-1{,}1\|)$	$\cos(e^x) - 0{,}5\sin(x^2)$	–1	1	100	0,001
7	$\cos(0{,}1-1/e^x)$	$\cos^3(e^x) - 0{,}2\cos(x)$	–2	0	50	0,001
8	$\ln(1-\cos(x))$	$\sin^3(3x) - \cos^2(x)$	–2	–0,5	100	0,005
9	$\ln(0{,}8-\sin(x))$	$0{,}2\cos^2(2x) + \sin(x)$	–2	1	100	0,005
10	$\ln(1{,}2-\sin^2(x))$	$\cos^3(x) + \sin(x-0{,}2)$	–2	0	100	0,001
11	$0{,}1\cos(x) - \|x\|^{2/3}$	$\sin^{0{,}2}(e^x) - 0{,}5\sin(x^2)$	–2	1	100	0,001
12	$0{,}5 - 0{,}1/e^x$	$0{,}5\cos^2(e^x) + \sin(x)$	–2	0	50	0,001
13	$0{,}1/e^x - 0{,}6$	$0{,}2\cos^2(x) + \cos(x)$	–2	1	50	0,001
14	$0{,}8 + \sin(0{,}2x-1)$	$0{,}2\sin^2(0{,}2x) + \cos(x)$	–2	1	50	0,001
15	$0{,}5 + \cos(x-1)$	$0{,}1\sin(x-0{,}1)$	–2	1	100	0,001
16	$0{,}5 - \|\sin(x-1)\|^{2/3}$	$\sin^2(e^x) - 0{,}1\sin(x^2)$	–1	1	100	0,001
17	$0{,}5 - \|\cos(x-1)\|^{2/3}$	$0{,}5\cos(0{,}1x^2)$	–2	1	100	0,001
18	$2 - \|x^2 + e^x\|$	$2\cos^2(x^2) + \sin(x^2)$	–2	0	100	0,001
19	$\|x^3 + e^{0{,}5x}\| - 3$	$2\sin(x^3) + \cos(x^2)$	–2	1	50	0,001
20	$0{,}1x^2 - e^{0{,}2x}$	$\sin^2(e^x) + 0{,}2\cos(x^2)$	–1	1	100	0,001
21	$0{,}1x - \ln\|1-x\|$	$\cos^2(e^x) - 0{,}2\sin(x)$	–1	2	50	0,001
22	$1 - e^{0{,}1x}$	$\sin^2(e^x) - 0{,}3\cos(x^2)$	–1	2	100	0,001
23	$1 - 0{,}2e^x$	$\sin^2(e^x) + 0{,}3\cos(x^2)$	–1	2	100	0,001
24	$0{,}1 - \sin(x)$	$\sin(e^x) - \cos(x^2)$	–1	2	100	0,001
25	$0{,}1x + \cos(x)$	$1 + \cos(e^x) - \ln(x^2)$	1	2	100	0,001

3. МЕТОДЫ ЧИСЛЕННОГО ДИФФЕРЕНЦИРОВАНИЯ ФУНКЦИЙ

Цель работы: изучить методы численного дифференцирования функций.

Теоретическая часть

При решении многих прикладных задач требуется вычислять производные определенного порядка от функции $f(x)$, аналитическое выражение которой неизвестно или невозможно, например в тех случаях, когда функция задана набором точек (таблично). В этих случаях используют приближенные (численные) методы дифференцирования функций.

Методы односторонней разности

По определению, первая производная есть предел отношения приращения функции к приращению независимой переменной при стремлении к нулю приращения независимой переменной:

$$f'(x_0) = \lim_{\mathrm{d}x \to 0} \frac{f(x_0 + \mathrm{d}x) - f(x_0)}{\mathrm{d}x}. \qquad (3.1)$$

Заменяя бесконечно малое приращение аргумента $\mathrm{d}x$ на малую, но конечную величину Δx, называемую шагом дифференцирования, получают выражение

$$f'(x_0) \approx \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}. \qquad (3.2)$$

Оценку ошибки вычисления производной получают разложением функции $f(x)$ в ряд Тейлора. Ее величина зависит от Δx: чем меньше шаг дифференцирования, тем меньше ошибка вычисления производной по формуле (3.2). Приведенная формула называется правой разностной схемой вычисления первой производной (рис. 3.1).

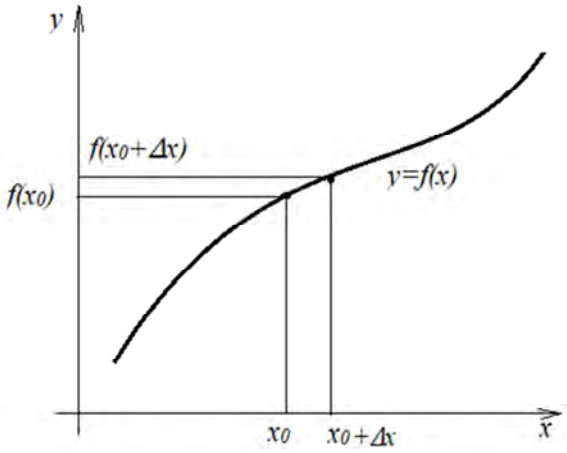


Рис. 3.1. Схема численного вычисления производной функции $f(x)$ в точке x_0 (метод правой односторонней разности)

Аналогично может быть записана левая разностная схема (рис. 3.2):

$$f'(x_0) \approx \frac{f(x_0) - f(x_0 - \Delta x)}{\Delta x}. \qquad (3.3)$$

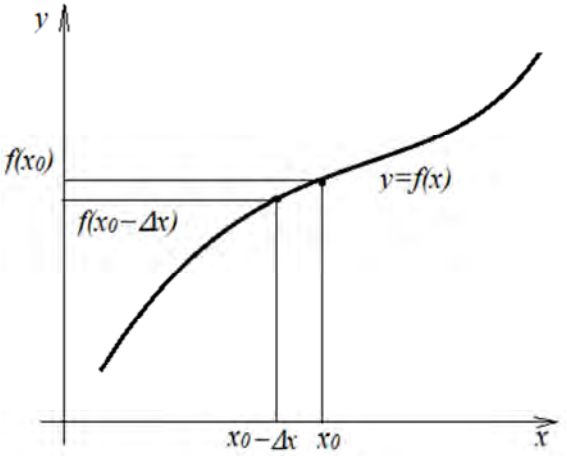


Рис. 3.2. Схема численного вычисления производной функции $f(x)$ в точке x_0 (метод левой односторонней разности)

Представленные выше численные методы вычисления производных называют в программировании двухточечными методами вычисления производных первого порядка.

Метод двусторонней разности

Методы левосторонней и правосторонней разностей позволяют получить значение производной с точностью, пропорциональной шагу дифференцирования Δx. Более точное значение можно получить с использованием метода двусторонней разности (рис. 3.3), в котором точность вычисления пропорциональна Δx^2:

$$f'(x_0) \approx \frac{f(x_0 + \Delta x) - f(x_0 - \Delta x)}{2 \cdot \Delta x}. \qquad (3.4)$$

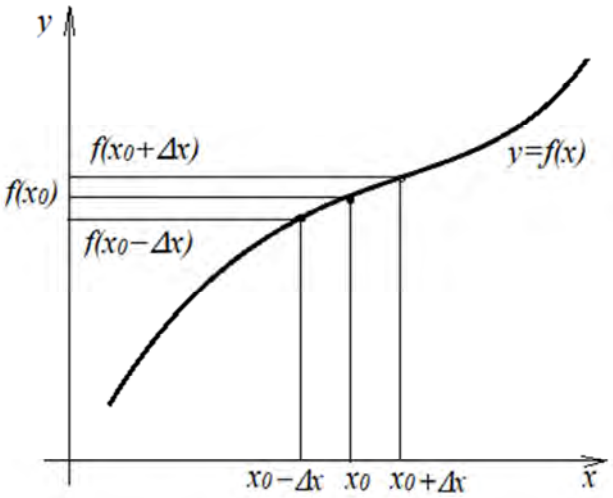


Рис. 3.3. Схема численного вычисления производной функции $f(x)$ в точке x_0 (метод двусторонней разности)

Метод двусторонней разности называют в программировании трехточечным методом вычисления производной.

Вычисление производных высоких порядков

Производная (n)-го порядка вычисляется как первая производная от производных (n – 1)-го порядка. Например, вторая производная функции $f(x)$ является первой производной от производных первого порядка, при этом выражение для ее численного вычисления имеет следующий вид:

$$f''(x_0) \approx \frac{f(x_0 + \Delta x) - 2 \cdot f(x_0) + f(x_0 - \Delta x)}{\Delta x^2}. \qquad (3.5)$$

Рассмотрим пример создания графического приложения для вычисления производных первого и второго порядков функции $y = x \cdot e^{-x}$ на заданном интервале $a < x < b$. Шаг изменения аргумента $\Delta x = (b - a) / N$, где $N = 100$. Принять $a = -1$, $b = 1$.

Создадим новый проект и поместим на форму компоненты TEdit, TLabel, TButton (страница Standart) и TAChart, как показано на рис. 3.4.

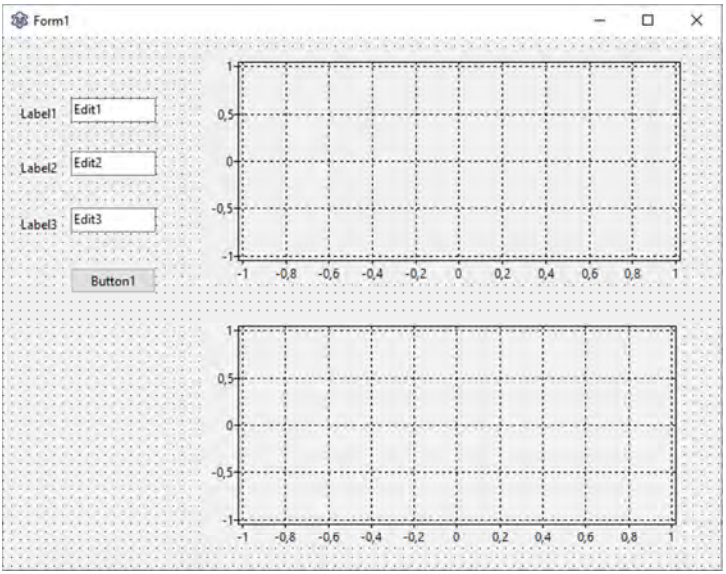


Рис. 3.4. Форма проекта с помещенными на нее компонентами TButton, TEdit, TLabel, TAChart

Двойным кликом мыши на помещенных в форму компонентах TAChart вызовем окно редактора диаграмм, добавим диаграмму типа «График».

Компоненты TEdit используются для организации ввода данных (*a, b, N*). Свойство Text измените последовательно во всех трех компонентах, соответственно, на –1, 1 и 100.

Компоненты TLabel используются для отображения необходимого текста на экране, для этого последовательно во всех трех компонентах свойство Caption измените, соответственно, на «*a*=», «*b*=», «*N*=».

Компонент TButton используется для организации ввода и начала обработки введенных данных. Свойство Caption измените на «Расчет». Двойной щелчок по кнопке мышью вызывает событие OnClick в обработчике событий и генерирует программный код процедуры вычислений.

Вставляем программный код в процедуру TForm1.Button1Click, присваивая переменным *a, b, N* значения StrToFloat(Edit1.Text), StrToFloat(Edit2.Text), StrToInt(Edit3.Text):

```
procedure TForm1.Button1Click(Sender: TObject);
var {раздел описания переменных}
a,b,x,dx,p1,p2: real;
N: integer;
function f(x:real):real;
begin
f:=x*exp(-x);
end;
{вычисление производных}
function df(x,dx:real):real;
begin
df:=(f(x+dx)-f(x-dx))/(2*dx);
end;
function dff(x,dx:real):real;
begin
```

```
dff:=(f(x+dx)-2*f(x)+f(x-dx))/sqr(dx);
end;
begin
a:=StrToFloat(Edit1.Text);
b:=StrToFloat(Edit2.Text);
N:=StrToInt(Edit3.Text);
dx:=(b-a)/N;
x:=a+dx;
while x<b do
begin
x:=x+dx;
p1:=df(x,dx);
p2:=dff(x,dx);
{используем тип диаграммы «График»}
Chart1LineSeries1.AddXY(x, p1);
Chart2LineSeries1.AddXY(x, p2);
end;
end;
```

Сохраните файлы проекта, скомпилируйте и запустите программу. Введите входную информацию. Щелчок по кнопке мышью приведет к обработке данных и выводу результата вычислений (рис. 3.5).

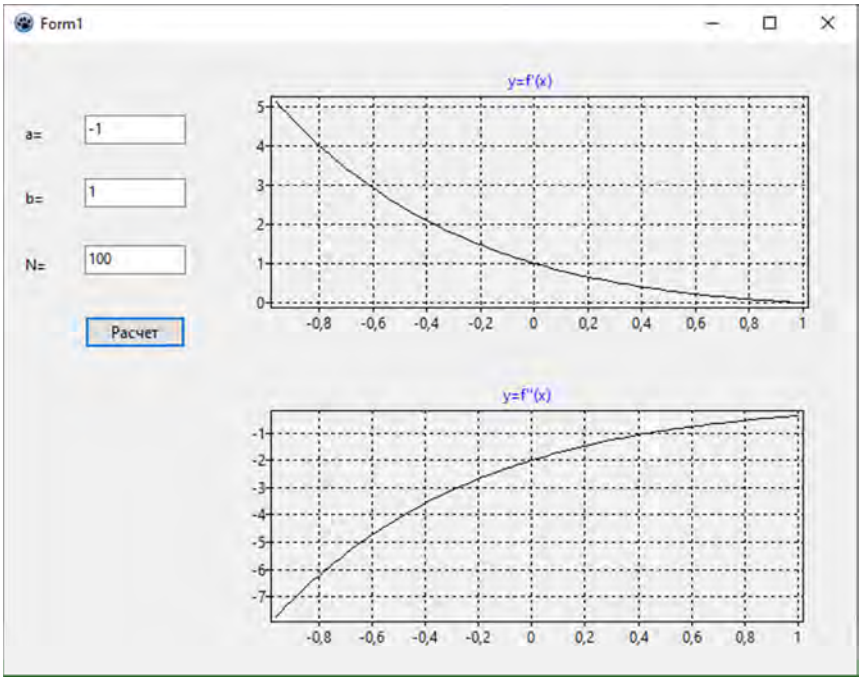


Рис. 3.5. Окно приложения с графиками первой и второй производных функции $f(x) = x \cdot e^{-x}$

Задание для самостоятельной работы

Используя данные табл. 2.1:

1. Разработать программу, реализующую алгоритм численных методов вычисления производных первого и второго порядков для заданных функций.
2. Составить отчет о работе.

4. РАСЧЕТ ТЕМПЕРАТУРНОГО ПОЛЯ МЕТОДОМ КОНЕЧНЫХ РАЗНОСТЕЙ

Численное моделирование процессов теплообмена играет исключительно важную роль в связи с тем, что обеспечивает достоверный прогноз таких процессов, экспериментальное изучение которых в лабораторных или натурных условиях является сложным, а в некоторых случаях невозможным [3].

Основным механизмом переноса тепла является теплопроводность, а процесс переноса тепла теплопроводностью описывается уравнением Фурье – Кирхгофа, которое устанавливает связь между временным и пространственным изменениями температуры в любой точке тела. В декартовой системе координат это уравнение имеет следующий вид:

$$c(T)\rho(T)\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(\lambda(T)\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\lambda(T)\frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(\lambda(T)\frac{\partial T}{\partial z}\right) + Q_w(x, y, z, t), \quad (4.1)$$

где T – температура;

λ – коэффициент теплопроводности;

c – удельная теплоемкость;

ρ – плотность;

t – время;

$Q_w(x, y, z, t)$ – мощность внутренних источников тепловыделения.

Геометрические условия определяют форму и размеры тела, в котором протекает изучаемый процесс. Физические условия определяют теплофизические характеристики тела λ, ρ, c. Начальные условия содержат распределение температуры в теле в начальный момент времени, граничные условия опреде-

ляют особенности протекания процесса на поверхности тела и могут быть заданы различными способами.

При решении дифференциального уравнения в частных производных наиболее часто используется метод конечных разностей (МКР). Идея МКР решения краевых задач весьма проста и видна уже из самого названия: вместо производных в дифференциальном уравнении используются их конечно-разностные аппроксимации.

При использовании МКР для задач теплопроводности твердое тело представляют в виде совокупности узлов. Аппроксимируя (заменяя) частные производные дифференциального уравнения (4.1) конечными разностями, получают систему линейных алгебраических уравнений для определения температуры как локальной характеристики в каждом узле сетки. Полученная система является незамкнутой, для ее замыкания используют разностное представление граничных условий. В результате получают замкнутую систему линейных алгебраических уравнений, которую решают численными методами с помощью ЭВМ.

Пример.

Рассмотрим теплопередачу через плоскую бесконечную пластину или изолированный стержень (рис. 4.1). На одной границе пластины поддерживается постоянная температура $T_л$, на другой границе – температура $T_п$. Начальная температура равна 0, источники тепловыделения внутри пластины отсутствуют.

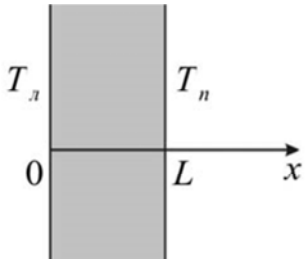


Рис. 4.1. Схема расчета

При заданных условиях температура будет изменяться только в направлениях, перпендикулярных границе пластины. Также предположим, что теплофизические характеристики не зависят от температуры. В связи с этим дифференциальное уравнение (4.1) преобразуется:

$$c\rho\frac{\partial T}{\partial t}=\lambda\frac{\partial^2 T}{\partial x^2},\ 0<x<L. \tag{4.2}$$

Начальные и граничные условия запишутся следующим образом:

$$\begin{aligned} &t=0:T=T_0,\ 0\le x\le L;\\ &x=0:T=T_{\text{л}},\ t>0;\\ &x=L:T=T_{\text{п}},\ t>0. \end{aligned} \tag{4.3}$$

Для того чтобы задать полное математическое описание рассматриваемой задачи, необходимо еще определить физические условия однозначности. Пусть материал пластины – сталь, для которой λ = 46 Вт/(м·ºС), ρ = 7800 кг/м^3, c = 460 Дж/(кг·ºС).

Эту задачу в полной математической постановке будем решать методом конечных разностей на равномерной сетке. Для этого разобьем пластину по толщине на $N-1$ равных промежутков, т. е. построим конечно-разностную сетку (рис. 4.2).

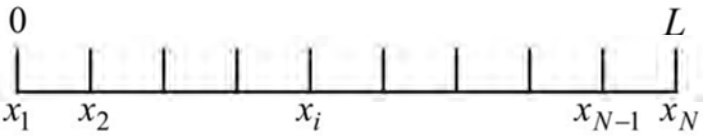


Рис. 4.2. Конечно-разностная (пространственная) сетка

Заменим дифференциальные операторы в (4.2) на их конечно-разностные аналоги, используя явную схему:

$$\frac{\partial T}{\partial t} = \frac{T_i^{n+1} - T_i^n}{\tau},$$
$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{h^2}. \qquad (4.4)$$

В результате аппроксимации частных производных соответствующими конечными разностями получаем следующее соотношение для определения поля температуры:

$$\rho c \frac{T_i^{n+1} - 2T_i^n}{\tau} = \lambda \left(\frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{h^2} \right), \; i = 2,.., N-1, \; n \geq 0, \quad (4.5)$$

из которого с учетом начальных и граничных условий получаем систему линейных алгебраических уравнений (СЛАУ) для нахождения распределения температуры в пластине в различные моменты времени:

$$\begin{cases} T_i^{n+1} = T_i^n + \dfrac{\lambda}{\rho c} \dfrac{\tau}{h^2} \left(T_{i+1}^n - 2T_i^n + T_{i-1}^n \right), \; i = 2,..., N-1, \; n \geq 0; \\ T_i^0 = T_0, \;\; i = 2,..., N-1; \\ T_1^n = T_{\text{л}}, \;\; n > 0; \\ T_N^n = T_{\text{п}}, \;\; n > 0. \end{cases} \qquad (4.6)$$

Таким образом, получили простую СЛАУ для нахождения распределения температуры в пластине в различные моменты времени. Аппроксимация дифференциальной задачи (4.2–4.3) конечно-разностной (4.5) выполнена с первым порядком по времени t и вторым по пространственной координате h.

Явная разностная схема является условно-устойчивой и требует специальных мероприятий по оценке возможности ее использования: чтобы решение конечно-разностной задачи (4.6)

сводилось к решению дифференциальной задачи, достаточно выполнение следующего условия (условие устойчивости разностной схемы):

$$\tau < \frac{\rho \cdot c \cdot h}{2\lambda}, \ 0 < x < L. \qquad (4.7)$$

Из условия (4.7) определяется шаг интегрирования по временной координате. Алгоритм решения задачи представлен на рис. 4.3 [3].

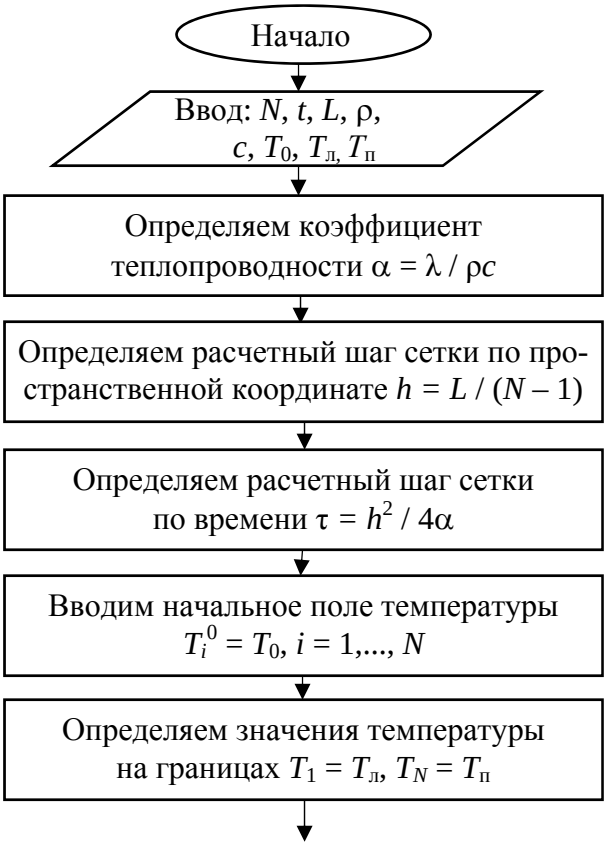
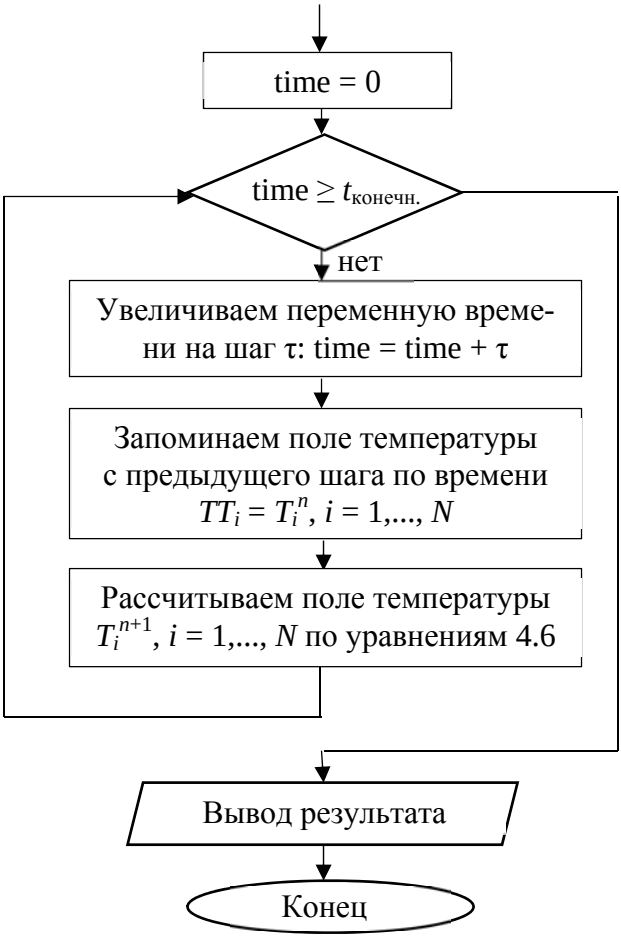


Рис. 4.3. Алгоритм решения задачи



Продолжение рис. 4.3

Создадим новый проект и поместим на форму компоненты TButton и TAChart.

Двойным кликом мыши на помещенном в форму компоненте TAChart вызовем окно редактора диаграмм, добавим диаграмму типа «График». Компонент TButton (кнопка Button1) является элементом управления, предназначенным для запус-

ка вычислений. Двойной щелчок по кнопке мыши вызывает событие OnClick в обработчике событий и генерирует программный код процедуры вычислений. Вставляем следующий программный код в процедуру TForm1.Button1Click для решения рассматриваемой задачи при L = 0,1 м, λ = 46 Вт/(м·ºС), ρ = 7800 кг/м^3, c = 460 Дж/(кг·ºС), T_0 = 20 °С, $T_л$ = 300 °С, $T_п$ = 100 °С:

```
procedure TForm1.Button1Click(Sender: TObject);
Const
mf = 2500;
N = 50; {количество узлов по пространственной координате}
type
 vector=array[1..mf] of real;
var {раздел описания переменных, которые мы будем использовать в программе}
i, j  : integer;
T, TT : vector;
a, lamda, ro, c, h, x, tau : real;
Tl, T0, Tr, L, t_end, time : real;
f, ff : text;
begin
{окончание по времени t_end };
t_end:=60;
{толщина пластины, L}
L:=0.1;
{коэффициент теплопроводности материала пластины lamda}
lamda:=46;
{плотность материала пластины, ro}
Ro:=7800;
{теплоемкость материала пластины, c}
C:=460;
{начальная температура, T0}
T0:=20;
```

```
{температура на левой границе x=0, Tl }
Tl:=300;
{температура на правой границе x=L, Tr }
Tr:=100;
{коэффициент температуропроводности}
a:=lamda/(ro*c);
{расчетный шаг сетки по пространственной координате}
h:=L/(N-1);
{расчетный шаг сетки по времени}
tau:=0.25*sqr(h)/a;
{поле температуры в начальный момент времени}
for i:= 2 to N-1 do
T[i]:=T0;
{определяем значения температуры на границе}
T[1]:=Tl;
T[N]:=Tr;
time:=0;
{используем цикл с предусловием}
while time<t_end do
begin
{увеличиваем переменную времени на шаг}
time:=time+tau;
{запоминаем поле температуры на предыдущем слое по времени}
for i:= 1 to N do
TT[i]:=T[i];
{определяем неизвестное поле температуры по соотношениям}
for i:= 2 to N-1 do
T[i]:=TT[i]+a*tau/sqr(h)*(TT[i+1]-2.0*TT[i]+TT[i-1]);
end;
{выводим результат в файл}
AssignFile (f,'res.txt');
Rewrite(f);
Writeln(f,'Толщина пластины L = ',L:6:4);
```

```
Writeln(f,'Расчетный шаг сетки по пространственной координате = ',h:6:4);
Writeln(f,'Коэффициент теплопроводности материала пластины lamda = ',lamda:6:4);  Writeln(f,'Плотность материала пластины ro = ',ro:6:4);
Writeln(f,'Теплоемкость материала пластины c = ',c:6:4);
Writeln(f,'Начальная температура  T0 = ',T0:6:4);
Writeln(f,'Температура на границе x = 0, Tl = ',Tl:6:4);
Writeln(f,'Температура на границе x = L, Tr = ',Tr:6:4);
Writeln(f,'Результат получен с шагом по пространственной координате x, h = ',h:6:4);  Writeln(f,'Результат получен с шагом по времени tau = ',tau:6:4);
Writeln(f,'Температурное поле в момент времени t = ',t_end:6:4);
CloseFile (f);
AssignFile (ff,'tempr.txt');
Rewrite(ff);
for i:=1 to N do
writeln(ff,'  ',h*(i-1):6:3,'   ',T[i]:8:5);
CloseFile (ff);
{выводим график распределения температуры}
for i:=1 to N do
begin
   x := h*(i-1);
  {используем тип диаграммы «График»}
   Chart1LineSeries1.AddXY(x, T[i]);
end;
end;
```

Сохраните файлы проекта, скомпилируйте и запустите программу. Щелчок по кнопке мышью приведет к обработке данных и выводу графика распределения температуры (рис. 4.4).

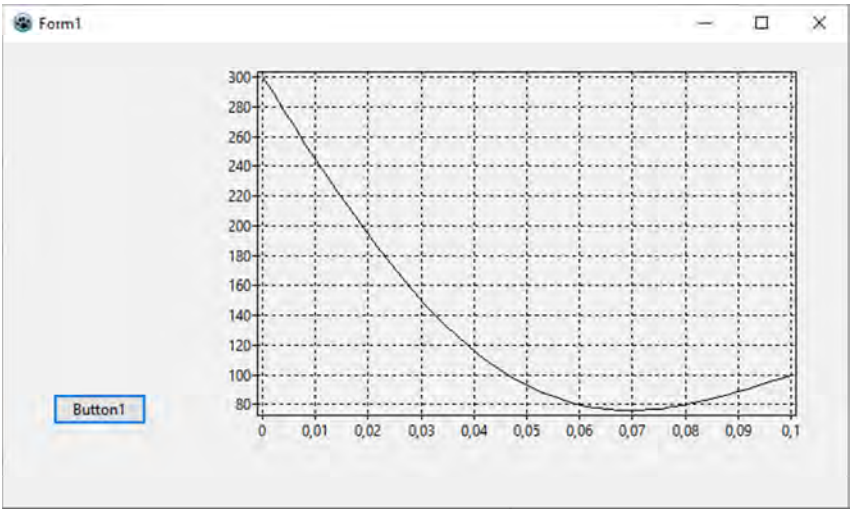


Рис. 4.4. Распределение температуры по толщине пластины в момент времени 60 с

Задание для самостоятельной работы

1. Выполнить расчет распределения температуры по толщине пластины в заданный момент времени t в соответствии с данными табл. 4.1.

2. Построить график распределения температуры по толщине пластины.

Таблица 4.1

Варианты заданий для самостоятельной работы

№ варианта	L, м	t, с	λ, Вт/(м·°С)	ρ, кг/м^3	c, Дж/(кг·°С)	T_0, °С	$T_л$, °С	$T_п$, °С
1	2	3	4	5	6	7	8	9
1	0,05	50	35	7650	420	0	100	800
2	0,06	45	40	7700	430	10	200	700
3	0,07	40	45	7750	440	20	300	600

Окончание табл. 4.1

1	2	3	4	5	6	7	8	9
4	0,08	35	50	7800	450	0	400	100
5	0,09	30	55	7650	460	10	500	200
6	0,10	25	35	7700	470	20	600	300
7	0,05	50	40	7750	480	0	700	100
8	0,06	45	45	7800	490	10	800	200
9	0,07	40	50	7650	420	20	900	300
10	0,08	35	55	7700	430	0	100	800
11	0,09	30	35	7750	440	10	200	700
12	0,10	25	40	7800	450	20	300	600
13	0,05	50	45	7650	460	0	400	100
14	0,06	45	50	7700	470	10	500	200
15	0,07	40	55	7750	480	20	600	300
16	0,08	35	35	7800	490	0	700	100
17	0,09	30	40	7650	420	10	800	200
18	0,10	25	45	7700	430	20	900	300
19	0,05	50	50	7750	440	0	100	500
20	0,06	45	55	7800	450	10	200	800

Вопросы для самостоятельной работы

1. Алгоритмы обработки данных: свойства, способы описания.
2. Основные алгоритмические конструкции.
3. Основные этапы разработки программы.
4. Структурное программирование: методология, основные положения.
5. Теорема Бема-Якопини.
6. Понятие типов данных в программировании.
7. Арифметические выражения в программировании.
8. Логические выражения в программировании.
9. Организация разветвляющихся вычислений в программировании.
10. Организация циклических вычислений в программировании.
11. Структурированные типы данных в программировании: массивы.
12. Структурированные типы данных в программировании: записи.
13. Понятие подпрограммы, типы подпрограмм. Обращение к подпрограмме.
14. Процедура. Формальные и фактические параметры процедуры.
15. Подпрограмма-функция. Формальные и фактические параметры подпрограммы-функции.
16. Стандартные подпрограммы для работы с файлами: объявление файлов в программе.
17. Стандартные подпрограммы для работы с файлами: вывод (запись) данных в файл, чтение данных из файла.
18. Модульный принцип программирования. Понятие модуля.
19. Структура модуля. Заголовок модуля и связь модуля с программой (язык программирования Паскаль).
20. Стандартные модули. Создание простейших графических изображений (язык программирования Паскаль).

21. Численные методы решения уравнений: основные понятия.

22. Теорема о существовании корней непрерывной функции.

23. Численные методы решения уравнений: алгоритм решения методом половинного деления.

24. Численные методы решения уравнений: метод секущих (хорд).

25. Численные методы решения уравнений: метод касательных.

26. Численные методы решения систем линейных алгебраических уравнений: основные понятия, способы решения.

27. Численные методы решения систем линейных алгебраических уравнений: метод обратной матрицы.

28. Численные методы решения систем линейных алгебраических уравнений: метод Крамера.

29. Приближение функций: методы аппроксимации и интерполирования.

30. Приближение функций: метод наименьших квадратов (постановка задачи).

31. Приближение функций: построение интерполяционных полиномов Лагранжа.

32. Численное дифференцирование. Конечно-разностные схемы аппроксимации производных первого порядка.

33. Численное дифференцирование. Конечно-разностные схемы аппроксимации производных второго порядка.

34. Численное интегрирование: алгоритм решения методом левых прямоугольников.

35. Численное интегрирование: алгоритм решения методом правых прямоугольников.

36. Численное интегрирование: алгоритм решения методом средних прямоугольников.

37. Численное интегрирование: алгоритм решения методом трапеций.

38. Численное интегрирование: алгоритм решения методом Симпсона.

Литература

1. Мансуров, К. Т. Основы программирования в среде Lazarus / К. Т. Мансуров. – 2010. – 772 с.

2. Алексеев, Е. Р. Free Pascal и Lazarus: учебник / Е. Р. Алексеев, О. В. Чеснокова, Т. В. Кучер. – М.: ALT Linux; Издательский дом ДМК-пресс, 2010. – 440 с.

3. Кузнецов, Г. В. Разностные методы решения задач теплопроводности: учебное пособие / Г. В. Кузнецов, М. А. Шеремет. – Томск: Изд-во ТПУ, 2007. – 172 с.

Оглавление

Учебное издание

РАФАЛЬСКИЙ Игорь Владимирович
АРАБЕЙ Анастасия Витальевна
ЛУЩИК Павел Евгеньевич

ПРИКЛАДНАЯ ИНФОРМАТИКА

Пособие
для студентов специальности 1-42 01 01
«Металлургическое производство и материалообработка
(по направлениям)»

Редактор *А. С. Мокрушников*
Компьютерная верстка *Н. А. Школьниковой*

Подписано в печать 01.10.2021. Формат 60×84 $^{1}/_{16}$. Бумага офсетная. Ризография.
Усл. печ. л. 2,27. Уч.-изд. л. 1,77. Тираж 100. Заказ 439.

Издатель и полиграфическое исполнение: Белорусский национальный технический университет.
Свидетельство о государственной регистрации издателя, изготовителя, распространителя
печатных изданий № 1/173 от 12.02.2014. Пр. Независимости, 65. 220013, г. Минск.