Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Елецкий государственный университет им. И. А. Бунина»

Губина Т.Н., Дякина В.А., Губин М.А.

**Язык программирования Паскаль: лабораторный практикум**

### Часть 1 Учебное пособие

ЕЛЕЦ — 2012

УДК 519.6+004.94+004.43

ББК 22.1

Г 93

Печатается по решению редакционно- издательского совета Елецкого государственного университета

им. И.А. Бунина от 12.04.2012 № 2

Рецензенты:

кандидат физико-математических наук, доцент Масина О.Н. кандидат педагогических наук, доцент Яшина Т.С.

### Губина Т.Н., Дякина В.А., Губин М.А.

**Г 93** Язык программирования Паскаль: лабораторный практикум. Часть 1. Учебное пособие. - Елец: ЕГУ им. И.А. Бунина, 2012. - 102с.

Учебное пособие содержит основные сведения о языке программирования Паскаль и его конструкциях; сведения о свободно распространяемых современных средах программирования, ориентированных на язык Паскаль; ряд лабораторных работ, в результате выполнения которых студенты приобретают необходимые знания и практические навыки программирования. Оно может быть использовано в рамках дисциплин «Алгоритмизация и программирование», «Основы информатики» и др. при подготовке бакалавров в учреждениях высшего и среднего профессионального образования, если государственным образовательным стандартом предусмотрено изучение раздела «Программирование». Оно также может быть полезным при изучении языка программирования Паскаль в старших классах общеобразовательных учреждений с информационно-технологическим профилем.

УДК 519.6+004.94+004.43

ББК 22.1

© Губина Т.Н., Дякина В.А., Губин М.А. 2012

© ЕГУ им. И.А. Бунина, 2012

**Оглавление**

[Предисловие 4](#_TOC_250024)

[Глава I. Основы программирования на языке Паскаль 5](#_TOC_250023)

[п. 1. Общие сведения о языке программирования Паскаль 5](#_TOC_250022)

[п. 2. Раздел описаний 7](#_TOC_250021)

[п. 3. Описание операций 9](#_TOC_250020)

[п. 4. Встроенные математические функции 10](#_TOC_250019)

[п. 5. Правила записи выражений 11](#_TOC_250018)

[п. 6. Операторы программ линейной структуры 11](#_TOC_250017)

[п. 7. Операторы программ разветвляющейся структуры 13](#_TOC_250016)

[п. 8. Операторы программ циклической структуры 14](#_TOC_250015)

[п. 9. Работа с массивами 16](#_TOC_250014)

[Глава II. Современные среды программирования, ориентированные на язык программирования Паскаль 20](#_TOC_250013)

[п. 1. Понятие «среда программирования» 20](#_TOC_250012)

[п. 2. Среда программирования PascalABC.NET 20](#_TOC_250011)

[п. 3. Среда программирования Geany 21](#_TOC_250010)

[п. 4. Среда программирования Free Pascal 23](#_TOC_250009)

[Глава III. Лабораторные работы 25](#_TOC_250008)

[п. 1. Лабораторная работа № 1. Программирование алгоритмов линейной структуры 25](#_TOC_250007)

[п. 2. Лабораторная работа № 2. Программирование алгоритмов разветвляющейся структуры 32](#_TOC_250006)

[п. 3. Лабораторная работа № 3. Программирование алгоритмов циклической структуры с заданным числом повторений 42](#_TOC_250005)

[п. 4. Лабораторная работа № 4. Программирование алгоритмов итерационной циклической структуры 50](#_TOC_250004)

[п. 5. Лабораторная работа № 5. Программирование алгоритмов со структурой вложенных циклов 62](#_TOC_250003)

[п. 6. Лабораторная работа № 6. Обработка одномерных массивов 70](#_TOC_250002)

[п. 7. Лабораторная работа № 7. Обработка двумерных массивов 78](#_TOC_250001)

Глоссарий 94

[Список использованной литературы 100](#_TOC_250000)

## ПРЕДИСЛОВИЕ

Настоящее пособие направлено на формирование у студентов практических навыков программирования на языке Паскаль, готовности использовать разнообразные приемы, методы и средства, необходимые в процессе разработки программного обеспечения.

В данном учебном пособии кратко приведены основные сведения о языке программирования Паскаль и его конструкциях; сведения о свободно распространяемых современных средах программирования, ориентированных на язык Паскаль; реализованы и подробно описаны программные коды, которые составляют основу решения большинства задач программирования.

В целях более эффективного усвоения учебного материала пособие построено по принципу лабораторных работ, разбитых по базовым типам алгоритмов и типам обрабатываемых данных.

Каждая лабораторная работа включает в себя: вопросы для подготовки к лабораторной работе, контрольные вопросы; учебные, учебно-исследовательские задания для самостоятельного решения, которым предшествует детальная проработка типовых задач и этапов их решения.

Выполнение лабораторных работ студентами позволит им закрепить знания и умения, полученные в школьном курсе информатики в области программирования, а также сформировать готовность осмысленно применять полученные знания и умения в будущей профессиональной деятельности.

В ходе изучения студентами учебного модуля «Язык программирования Паскаль: лабораторный практикум» предполагается выполнение практических заданий в компьютерной лаборатории.

## ГЛАВА I. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ ПАСКАЛЬ

## п. 1. Общие сведения о языке программирования Паскаль

**Паскаль** (англ. **Pascal**) — строго типизированный язык програм- мирования общего назначения.

Один из наиболее известных языков программирования, содержа- щий основные конструкции и способы обработки данных, присущие всем современным языкам программирования. Широко применялся в промышленном программировании, обучении программированию в выс- шей школе. Он является базой для таких языков, как Object Pascal, С++, Java, C#, Eiffel и др. Последние 2 года в индексе популярности языков про- граммирования TIOBE занимает 15 место[1](#_bookmark0).

Язык Паскаль был создан Никлаусом Виртом в 1968-69 годах после его участия в работе комитета разработки стандарта языка [Алгол-68](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D0%BB-68). Он был опубликован в 1970 году Виртом как небольшой и эффективный язык, способствующий хорошему стилю программирования, использую- щий структурное программирование и структурированные данные. Язык Паскаль поддерживает технологии объектно-ориентированного програм- мирования.

В языке Паскаль используются следующие *символы*:

* буквы латинского алфавита;
* цифры от 0 до 9;

• специальные символы: «:=»; «+»; «-»; «\*»; «/»; «>»; «<»; «=»; «()»; «[]»;

«.»; «,»; «{}»;

* ключевые слова: and, array, begin, const, do, end, then, var и др.;
* знаки операций: + - / \* и др..

В языке Паскаль выделяют следующие *категории лексем*:

• *специальные символы:* + - \* / < = > [ ] ( ) , : ; ^ $ := .. <= >= и др.;

* *зарезервированные (ключевые) слова;*
* *идентификаторы* – имена констант, типов, переменных, процедур, функций, программ, меток и др.;
* *метки*: числовые и символьные;
* *числа:* целые десятичные, целые шестнадцатеричные и веществен- ные десятичные;
* *строка символов;*
* *комментарий*.

В качестве *разделителей лексем* применяются пробел, символ табу- ляции и составной символ перехода в начало следующей строки (пара сим- волов «Возврат каретки» и «перевод строки»).

1 Режим доступа: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>

*Комментарии* представляют собой пояснительный текст, который включается в программу для лучшего ее понимания. Его можно писать в любом месте программы. Текст комментария ограничивается символами

{ } или (\* \*) и может содержать любые комбинации символов алфавита языка Паскаль.

Программа на языке Паскаль состоит из заголовка, разделов описа- ний и раздела операторов, последовательно расположенных друг за дру- гом.

|  |  |
| --- | --- |
| *Общая структура программы на языке Pascal* | *Пример программы* |
| { I. Заголовок программы } |  |
| program Имя\_Программы; | program summa; |
| { II. Раздел указания используемых модулей } |  |
| uses Список\_Используемых\_Модулей; | uses crt; |
| { III. Раздел описаний } |  |
| label Описание\_Меток; |  |
| const Описание\_Констант; | const n=3.5; |
| type Описание\_Типов; |  |
| var Описание\_Переменных; | var a, summa as real; |
| procedure Описание\_Процедур; |  |
| function Описание\_Функций; |  |
| exports Описания\_Экспортируемых\_имен; |  |
| { IV. Раздел операторов (операторный блок) } |  |
| begin | begin |
| Операторы | clrscr; read(a); summa:=a+n;  write('summa=', summa); |
| end. | end. |

Заголовок не является обязательной частью программы и может не указываться. Он состоит из зарезервированного слова program и идентификатора — имени программы, за которым следует точка с запятой.

При написании программ следует соблюдать *общие правила написа- ния идентификаторов:*

* идентификатор может начинаться только с буквы или знака подчер- кивания (исключение составляют метки, которые могут начинаться также и с цифры);
* идентификатор может состоять из букв, цифр и знаков подчеркива- ния (пробелы, точки и другие специальные символы недопустимы);
* между двумя идентификаторами должен быть по крайней мере один пробел;
* максимальная длина идентификатора составляет 127 символов, но значимыми являются только первые 63 символа;
* при записи идентификаторов можно использовать как прописные, так и строчные буквы. Компилятор не делает различий между ними, хотя они и имеют различные ASCII-коды.

## п. 2. Раздел описаний

Рассмотрим более подробно такую структурную единицу программы, как раздел описаний.

В программе на языке Паскаль все данные (переменные, константы, массивы, метки, подпрограммы пользователя) должны быть обязательно объявлены.

*Объявить* — значит определить тип и присвоить имя для того, чтобы программа при запуске смогла зарезервировать в оперативной памяти компьютера столько места, сколько необходимо для хранения перемен- ной, константы, массива, метки и пр.

*Тип данных* — фундаментальное понятие теории программирования. Тип данных определяет *множество значений*, *набор операций*, которые можно применять к таким значениям, и *способ реализации хранения значе- ний и выполнения операций*. Любые данные, которыми оперируют програм- мы, относятся к определённым типам.

Каждый язык программирования поддерживает один или несколько *встроенных типов данных* (базовых типов), кроме того, многие языки про- граммирования предоставляют программисту возможность описывать *собственные (пользовательские) типы данных*.

Приведем некоторые часто используемые *стандартные типы дан- ных* языка Паскаль:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Название типа** | **Идентифи- катор** | **Диапазон значений** | **Размер памяти** |
| Целые типы | Короткое целое со знаком | Shortint | -128…127 | 1 байт |
| Целое со знаком | Integer | -32768…32767 | 2 байта |
| Длинное целое со знаком | Longint | -2147483648 …  147483647 | 4 байта |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Название типа** | | **Идентифи- катор** | **Диапазон значений** | | **Размер памяти** |
|  | Целое без знака | | Word | 0…65535 | | 2 байта |
| Короткое без знака | целое | Byte | 0..255 | | 1 байт |
| Веществен  -ные типы | Вещественное | | Real | от 2.9\*10-39 1.7\*1038 | до | 6 байт |
| Вещественное двойной точности | | Double | от 5.0\*10-324 1.7\*10308 | до | 8 байт |
| Булевские типы | Булевское | | Boolean | True или false | | 1 байт |
| Символьн ые типы | Символьное | | Char | 1 символ таблицы ASCII | | 1 байт |
| Строковые типы | Строковое | | String | Последовательност ь символов ASCII длиной до 255 | | На 1  символ 1 байт |

Приведем некоторые часто используемые *пользовательские типы данных* языка Паскаль:

* *перечисления* — описывают новые типы данных, значения которых определяет сам программист. Список элементов этого типа заключается в круглые скобки;
* *интервальный тип* — диапазон значений какого-то порядкового типа, называемого базовым. Список допустимых значений этого типа задается через указание минимального и максимального значений из некоторого диапазона через две точки;
* *структурированный тип* служит для описания данных, которые могут иметь много значений: тип-массив, множество, запись, файловый тип.

Предложение описания переменных начинается с зарезервированно- го слова var (сокр. от англ. variable — переменная), далее, через запятую, указываются имена используемых переменных и через символ “:” тип переменных. Пример:

### var

**x,y:real; i,j:integer; s:string;**

**a:array[1..20] of integer;**

**b: 1..100; c=(1,2,3,4,5,6,7);**

Предложение описания констант начинается с зарезервированного слова const, далее через символ “=” указывается значение константы. При- мер:

### const

**m=100;**

**n='yes'; f=false;**

**a:array[1..5] of integer=(10, 20, 30, 40, 50);**

**b:set of 0..9=[1,2,3,4,5,6,7,8,9];**

## п. 3. Описание операций

*Операция —* конструкция в языках программирования, аналогичная по записи математическим операциям, то есть специальный способ записи некоторых действий[2](#_bookmark1). Наиболее часто применяются арифметические, ло- гические и строковые операции.

В отличие от функций, операции часто являются базовыми элемента- ми языка и обозначаются различными символами пунктуации, а не алфа- витно-цифровыми.

*Арифметические операции:*

\* умножение

/ деление

+ сложение

- вычитание

div — целочисленное деление

mod — остаток от целочисленного деления

*Операции отношения:*

= равно

<> не равно

<= меньше или равно

>= больше или равно

< меньше

> больше

*Логические операции:*

not логическое отрицание

and логическое умножение (и) or логическое сложение (или)

xor логическое исключающее или

*Строковая операция:*

+ конкатенация (сцепление) строк

2 [http://ru.wikipedia.org/wiki/Операция\_(программирование](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))

*Операции инкремента и декремента:*

Inc(A) — инкремент, увеличение значения А на единицу Dec(A) — декремент, уменьшение значения А на единицу Inc(A,n) — увеличение значения А на n

Deс(A,n) — уменьшение значения А на n

*Старшинство операций:*

1. приоритет: not
2. приоритет: \*, /, div, mod, and 3 приоритет: +, -, or, xor

4 приоритет: =,<>,<,>,<=, >=, in

## п. 4. Встроенные математические функции

*Функция* — в программировании — это поименованная часть про- граммы, которая может вызываться из других частей программы столько раз, сколько необходимо[3](#_bookmark2).

На вход функции поступают управляющие воздействия в виде значе- ний аргументов. На выходе функция возвращает результат, который может быть как скалярной величиной, так и векторным значением (структура, индексный массив и т.п.).

Различают встроенные функции и функции пользователя. Приведем часто используемые встроенные функции языка Паскаль, используемые при написании программ[4](#_bookmark3):

|  |  |
| --- | --- |
| abs(x) | возвращает абсолютное значение (модуль) x |
| sqr(x) | возвращает квадрат x |
| sqrt(x) | возвращает квадратный корень из x |
| sin(x) | возвращает синус x |
| cos(x) | возвращает косинус x |
| ln(x) | возвращает натуральный логарифм x |
| exp(x) | возвращает e в степени x (e=2.718281...) |
| arctan(x) | возвращает арктангенс x |
| power(x,y) | возвращает x в степени y |
| round(x) | возвращает результат округления x до ближайшего целого |
| trunc(x) | возвращает целую часть х |
| int(x) | возвращает целую часть х |
| frac(x) | возвращает дробную часть х |

1. [http://ru.wikipedia.org/wiki/Функция\_(программирование)](http://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))
2. Полный список встроенных функций можно найти в справочной системе среды программирования

|  |  |
| --- | --- |
| random(x) | возвращает случайное целое в диапазоне от 0 до x-1 |
| random | возвращает случайное вещественное в диапазоне [0..1) |

## п. 5. Правила записи выражений

В состав арифметического выражения на языке Паскаль могут вхо- дить числовые константы, имена переменных, знаки математических опе- раций, математические функции и функции, возвращающие число, откры- вающиеся и закрывающиеся круглые скобки. Правила построения выраже- ний напоминают математические с некоторыми уточнениями.

*Арифметические выражения* записываются по следующим правилам:

* + выражения записываются в строчку;
  + все знаки арифметических операция прописываются в явном виде; нельзя опускать знак умножения между сомножителями и ставить рядом два знака операций;
  + для обозначения переменных используются буквы латинского алфавита и символ подчеркивания;
  + аргументы всех встроенных функций пишутся в круглых скоб-

ках;

* + количество открывающихся и количество закрывающихся ско-

бок должно быть одинаковым;

* + операции выполняются в порядке старшинства: сначала вычис- ление функций, затем возведение в степень, потом умножение и деление и в последнюю очередь — сложение и вычитание;
  + для изменения порядка операций в выражении используются круглые скобки;
  + операции одного старшинства выполняются слева направо.

## п. 6. Операторы программ линейной структуры

*Программа линейной структуры* — программа, реализующая алго- ритм структуры «Следование» (рис. [6.1](#_bookmark5)), в котором блоки выполняются в указанном порядке, последовательно друг за другом. Для организации ли- нейной программы используют операторы ввода, вывода данных и опера- тор присваивания.

Самым простым действием над переменной является занесение в нее величины соответствующего типа — присвоение переменной конкретного значения.

Синтаксис оператора присваивания:

### Переменная:=Значение;[5](#_bookmark4)

Выражение, указанное справа от знака ":=" (символа присваивания), должно приводить к значению того же типа, какого и сама переменная, или типа, совместимого с пере- менной относительно команды присваивания. Выражение будет сначала вычислено, затем его результат будет поло- жен в ячейки памяти, отведенные для переменной.

Инструкции ввода и вывода позволяют ввести аргу- менты, параметры расчетов во время выполнения програм- мы, осуществить вывод рассчитанных данных в понятном человеку виде.

Синтаксис оператора ввода данных:

**Read(x1, x2, …, xn);** осуществляет ввод x1, x2, …, xn с клавиатуры.

**Readln(x1, x2, …, xn);** осуществляет ввод x1, x2, …, xn с клавиатуры и после выбора значения последней перемен- ной обеспечивает переход к началу новой строки.



Рисунок 6.1: Структура "следование"

**Readln;** обеспечивает пропуск одной строки и переход к началу но- вой строки.

Элементами списка ввода могут быть имена переменных, которые должны быть заполнены значениями, введенными с клавиатуры.

Выполнение операторов ввода происходит так: ход программы при- останавливается, на экран выводится курсор, пользователь с клавиатуры вводит необходимые значения в том порядке, в котором они требуются списком ввода, нажимает Enter[6](#_bookmark6). После этого набранные данные попадают в соответствующие им переменные и выполнение программы продолжа- ется.

Синтаксис оператора вывода данных:

**Write(x1, x2, …, xn);** выполняет вывод значений, соответствующих перечисленным именам, размещая выводимые значения в одной строке.

**Writeln(x1, x2, …, xn);** выполняет вывод значений x1, x2, …, xn, соот- ветствующих перечисленным именам, и после вывода последнего значе- ния осуществляет переход к новой строке.

**Writeln;** обеспечивает пропуск строки и переход к началу новой строки.

Элементами списка вывода могут являться имена переменных, выра- жения, константы. Прежде чем вывести на экран, компьютер вычислит значения выражений.

Задание формата выводимых данных:

* для вывода значений целого типа: Write(b:m);

Writeln(b:m);

1. Данные при вводе разделяются пробелами (в случае использования read).
   * для вывода действительного типа: Write(b:m:n);

Writeln(b:m:n);

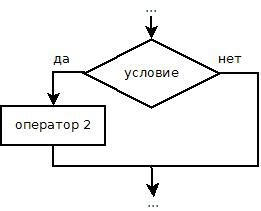
где b – имя выводимой переменной; m – поле, отводимое под значение и задаваемое константой или выражением целого типа; n – часть поля, отво- димого под дробную часть числа.

## п. 7. Операторы программ разветвляющейся структуры

*Программа разветвляющейся структуры —* программа, реализую- щая алгоритм структуры «Развилка» (рис. [7.1](#_bookmark32)-[7.3](#_bookmark35)), который предусматри- вает выбор одной из нескольких последовательностей операторов (вет- вей) в зависимости от некоторых условий. Для организации разветвления используют оператор перехода, условный оператор и оператор выбора.

*Оператор безусловного перехода* goto вызывает передачу управления оператору, которому предшествует метка, указанная в данном операторе перехода.

Синтаксис оператора безусловного пере-



хода:

### label n;

**…. ….. Goto n;**

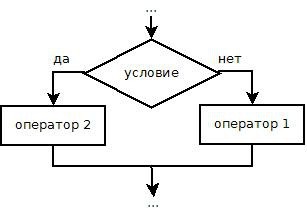
**…... …..**

**n: оператор;**

**…... …...**

где n – метка оператора.

Использовать оператор Goto следует только в редких случаях, например, для выхода к концу программы или процедуры в случае неправильного задания данных



или выхода из тела цикла.

*Условные операторы* позволяют выбрать для выполнения один из составных операторов (или не выбрать ни одного).

Синтаксис условного оператора:

**-** короткая форма условного опера-

Рисунок 7.1: Структура "развилка" "если-то"

тора:

ра:

### If условие then оператор;

* полная форма условного операто-

Рисунок 7.2: Структура "развилка" "если-то-иначе"

### If условие then оператор1 else оператор2;

Блочная форма условного оператора (пример):

### If условие then begin

**оператор;**

**if выражение then**

**оператор else оператор;**

**оператор; end**

**else begin**

**оператор; оператор;**

**end;**

*Оператор выбора* обеспечивает выполнение одного оператора из нескольких возможных. Переключатель (селектор) должен иметь порядковый тип (размером в байт или слово).

Синтаксис оператора выбора:

### Case Переключатель of

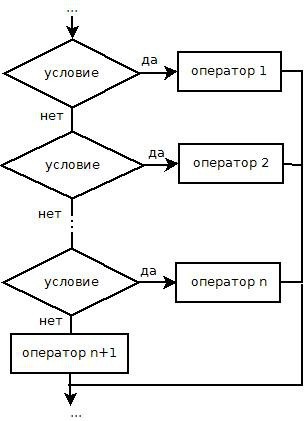
**Список констант1: Оператор1; Список констант2: Оператор2;**

**…**

**Список константN: ОператорN**

**Else ОператорЕ**

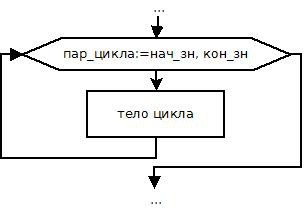
Рисунок 7.3: Структура "ветвление" "выбор"



### end;

**п. 8.** **Операторы программ циклической структуры**

*Программа циклической структуры —* программа, реализующая ал- горитм структуры «Цикл» (рис. [8.1](#_bookmark7)-[8.3](#_bookmark8)), в



котором происходит многократное по- вторение одного и того же участка про- граммы. Различают детерминированные циклы с заранее известным числом по- вторений (цикл с параметром) и итера- ционные циклы (циклы с предусловием и с постусловием), в которых число по- вторений заранее неизвестно.

*Цикл с параметром* (или со

счетчиком) используется, если до

Рисунок 8.1: Структура "цикл" "для"

выполнения цикла известны начальное и конечное значения счетчика

повторений цикла.

Синтаксис цикла с параметром:

### For переменная := нач\_значение To конеч\_значение Do Оператор; шаг +1

или

### For переменная := нач\_значение To конеч\_значение Do begin

**Оператор1;**

**Оператор2**; Если в цикле требуется выполнять более

**…. ….** одного оператора

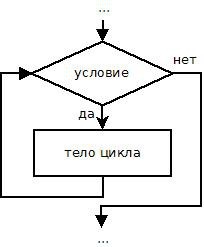
### ОператорN; end;

или

### For переменная := нач\_значение Downto конеч\_значение Do Оператор; шаг -1

*Цикл с предусловием* содержит в себе выражение (условие), которое управляет повторным выполнением оператора. Вычисление его производится до того, как внутренний оператор будет выполнен. Внутренний оператор выполняется повторно до тех пор, пока выражение принимает значение Тruе. Если выражение с самого начала принимает значение False, то оператор, содержащийся внутри оператора цикла с предусловием, не выполняется.

Синтаксис оператора цикла с предусловием:



### While условие do Оператор;

или

### While Условие do Begin

**Оператор1;**

**Оператор2;** Если в цикле требуется

выполнять более

**…. ….** одного оператора

### ОператорN;

**end;**

Рисунок 8.2: Структура "цикл с предусловием"

*Цикл с постусловием* — цикл, в котором условие продолжения цикла располагается в конце тела цикла. Выполняется до тех пор, пока условие окончания цикла ложно.

Синтаксис оператора цикла с постусловием:

### Repeat

**Оператор1;**

**Оператор2;**

**…. ….. ….. ОператорN**

**Until условие**

Операторы прерывания выполнения цикла:

* + **continue** – оператор завершения шага цикла. По этому оператору прекращается выполне- ние текущего шага цикла, и управление передается на следующий шаг;
  + **break** – оператор «выхода» из цикла. Пре- кращается выполнение цикла и управление пере- дается оператору, следующему за последним опера- тором цикла;
  + **exit** – оператор «выхода» из программы. По этому оператору выполнение программы завер-

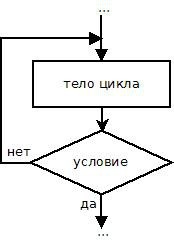


Рисунок 8.3: Структура "цикл с постусловием"

шается. Рекомендуется использовать для выхода из подпрограмм.

## п. 9. Работа с массивами

*Массив* — переменная структурированного типа. Массиву присваивается имя, посредством которого можно ссылаться на него, как на единое целое, так и на любую из его компонент.

Каждое из значений, составляющих массив, называется его

*компонентой* (или *элементом* массива).

Элементы массива располагаются в последовательных ячейках памяти, обозначаются именем массива и индексом.

*Размерность* массива — количество индексов элементов массива. По размерности массивы делятся на одномерные (линейные), двумерные, трёхмерные и т.д.

*Размер массива* — количество элементов в нем.

*Переменная с индексом* — идентификатор компоненты массива. Индекс может быть константой, переменной или выражением порядкового типа (целочисленный, логический, символьный, перечислимый, диапазон).

Примеры:

- одномерный массив — массив A, состоящий из компонент, каждая из которых является целым числом:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1-й элемент  12 | 2-й элемент  34 | 3-й элемент  98 | 4-й элемент  24 | 5-й элемент  1 | 6-й элемент  0 | ...  ... |

В одномерном массиве может хранится, например, информация о

количестве учащихся каждого класса в школе, температура воздуха за каждый день одного месяца и т. д.

- двумерный массив — массив B, состоящий из компонент, каждая из которых является вещественным числом:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1-й столбец | 2-й столбец | 3-й столбец | 4-й столбец | 5-й столбец | ... |
| 1-я строка | 1,2 | 4,78 | 0 | 0,23 | 67 | ... |
| 2-я строка | 2,3 | 3,45 | 9 | 0,12 | 0 | ... |
| 3-я строка | 1,12 | 2,1 | 5,89 | 3 | 0,25 | ... |
| 4-я строка | 0 | 0 | 5,45 | 4 | 1,56 | ... |
| 5-я строка | 78 | 2 | 2,45 | 2,1 | 7,8 | ... |
| ... | ... | ... | ... | ... | ... | ... |

В двумерном массиве может хранится, например, информация о температуре воздуха за каждый день всех месяцев года (номер строки совпадает с номером месяца, номер столбца — с номером дня месяца).

Чаще всего в программировании используются одномерные и двумерные массивы, реже — трехмерные массивы.

Массивы необходимо предварительно объявлять. *Объявление (описание) массива* определяет имя, размер массива и тип данных, которые будут храниться в массиве.

Формат объявления одномерного массива в разделе переменных: **var <имя массива>: array[<тип индекса>] of <базовый тип>** Пример:

### сonst n=100;

**var A: array[1..n] of real;** Формат объявления одномерного массива в разделе типов: **const n=<число>;**

### type <имя типа>=array[<тип индекса>] of <базовый тип>; var <имя массива>:<имя типа>;

Пример:

### const n=100;

**type mas=array[1..n] of integer; var A: mas;**

Формат объявления двумерного массива в разделе переменных:

### var <имя массива>: array[<тип индекса 1>, <тип индекса 2>] of

**<базовый тип>;**

Пример:

### const m=30; n=50

**var A: array[1..m, 1..n] of char;**

Формат объявления двумерного массива в разделе типов:

### type <имя типа>: array[<тип индекса 1>, <тип индекса 2>] of <ба- зовый тип>;

**var <имя массива>:<имя типа>;**

Пример:

### const m=30; n=30;

**type matr=array[1..m, 1..n] of longint; var A: matr;**

Объявление трехмерного массива (пример):

### const m=30; n=50; k=50;

**var A: array[1..m, 1..n, 1..k] of char;**

или

### const m=30; n=30; k=50;

**type mas=array[1..m, 1..n, 1..k] of longint; var A: mas;**

Для ввода и вывода элементов массива используется цикл с параметром, где в качестве параметра применяется индексная переменная. Для двумерного массива обычно первый индекс элемента в массиве связывают с номером строки, второй – с номером столбца.

Способы ввода (заполнения) массива данными:

1. С клавиатуры

**for i:=1 to 40 do read(a[i]);** или для двумерного массива **for i:=1 to 10 do**

### for j:=1 to 10 do read(a[i,j] );

1. По формуле

**for i:=1 to 40 do a[i]:=i\*i+5;** или для двумерного массива **for i:=1 to 10 do**

### for j:=1 to 10 do a[i,j]:=i\*j-4;

1. С помощью генератора случайного числа

### for i:=1 to 40 do a[i]:=random\*20;

или для двумерного массива

### for i:=1 to 10 do

**for j:=1 to 10 do a[i,j]:=random\*10;**

Вывод элементов массива на экран монитора:

### for i:=1 to 40 do write(a[i]);

или для двумерного массива (в виде матрицы)

### for i:=1 to 10 do begin

**for j:=1 to 10 do write(a[i,j] );**

**writeln; end;**

## ГЛАВА II. СОВРЕМЕННЫЕ СРЕДЫ ПРОГРАММИРОВАНИЯ, ОРИЕНТИРОВАННЫЕ НА ЯЗЫК ПРОГРАММИРОВАНИЯ ПАСКАЛЬ

## п. 1. Понятие «среда программирования»

*Среда программирования* или *интегрированная среда разработки*, ИСР (англ. IDE, Integrated development environment или integrated debugging environment) — система программных средств, используемая программистами для разработки программного обеспечения (ПО)[7](#_bookmark9) и обычно ориентируется на конкретный язык или несколько языков программирования (в этом случае языки, обычно, принадлежат одной языковой группе, например, Си-подобные).

Интегрированная среда разработки содержит в себе все необходимое для разработки программ:

* *текстовый редактор* с подсветкой синтаксиса конкретного языка программирования (в нем набирается программный код);
* *компилятор и/или интерпретатор* — транслирует программу, написанную на высокоуровневом языке программирования в машинный язык (машинный код), непосредственно понятный компьютеру;
* *отладчик* служит тестирования программы, выявления в ней логических ошибок и отладки программ;
* *средства автоматизации сборки*;
* *справочную систему*.

Иногда ИСР содержит также средства для интеграции с системами управления версиями и разнообразные инструменты для упрощения конструирования графического интерфейса пользователя.

## п. 2. Среда программирования PascalABC.NET

Система PascalABC.NET разработана на факультете математики, механики и компьютерных наук[8](#_bookmark10) Южного федерального университета.

PascalABC.NET — это бесплатная, простая и мощная среда разработки для языка Object Pascal, ориентированная на обучение программированию, опирающаяся на передовую платформу программирования Microsoft.NET. Официальный сайт, с которого можно скачать дистрибутив этой среды, расположен по адресу <http://pascalabc.net/>. Все права на систему программирования PascalABC.NET принадлежат PascalABCCompiler Team.

Система имеет удобный графический интерфейс (рис. [2.1](#_bookmark15)), имеется функция подсветки синтаксиса, поддерживает многооконный режим работы, результаты работы отображаются в нижней части окна. Предусмотрены возможности генерации исполняемых файлов, запуск

1. [http://ru.wikipedia.org/wiki/Интегрированная\_среда\_разработки](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8)
2. Официальный сайт: <http://mmcs.sfedu.ru/>

программы вне программной оболочки, установки точки прерывания программы, просмотра и редактирования текущих значений переменных, пошагового выполнения программы, вывода подсказки при наведении на объектах кода, автоформатирования кода и др.

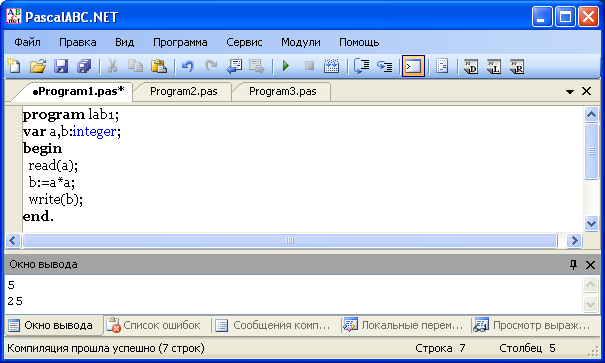


Рисунок 2.1: Графический интерфейс окна программы PascalABC.Net

Для запуска программы на исполнение можно воспользоваться кнопкой «Выполнить» на панели инструментов, или нажать клавишу F9.

Подробное описание системы PascalABC.NET, большое количество примеров программ можно найти на сайте: «PascalABC.Net. Обучение современному программированию» (режим доступа: <http://pascalabc.net/algoritmi-i-programmi-dlya-studentov>).

## п. 3. Среда программирования Geany

**Geany** — свободная среда разработки программного обеспечения, написанная с использованием библиотеки GTK2.

Она поддерживает много языков программирования, в том числе и Паскаль. Geany обладает всеми основными особенностями, присущими интегрированным средам разработки: подсветка синтаксиса, множество поддерживаемых языков — C, Java, PHP, Python, Паскаль и др., авто- закрытие XML и HTML-тегов, простое управление проектами, компиляция и выполнение кода программы, автоотступы, код-фолдинг (сворачивание блоков кода по щелчку на значке рядом с началом блока) и пр.

Среда позволяет выполнять настройку среды: сессий и сохранения файлов внутри проекта, настройку сообщений, расположение основных элементов интерфейса (редакторское окно, панель инструментов, список открытых файлов, список символов), установку типов файлов и кодировки

по умолчанию, основных параметров шаблонов, параметров печати и др.

Преимущество этой среды разработки в отличие от среды Pascal ABC.Net — возможность ее использования под управлением операционных систем и Windows, и Linux.

Официальный сайт, с которого можно скачать дистрибутив этой среды, расположен по адресу <http://www.geany.org/>.

Внешний вид окна программы непосредственно после ее запуска и выбора нового файла языка Паскаль приведен ниже (см. рис. [3.1](#_bookmark18)).

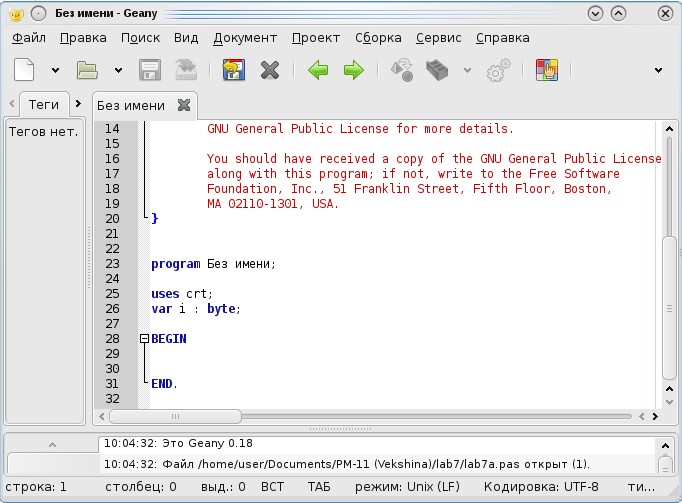


Рисунок 3.1: Графический интерфейс окна программы Geany

Как видим, программа распространяется по лицензии GNU GPL. Прежде чем запустить программу на исполнение (F5), ее требуется сохранить и выполнить компиляцию проекта (F8).

Подробное описание системы Geany можно найти на сайте «Geany: быстрая и легковесная среда разработки» по адресу <http://www.geany.org/manual/0.19_ru/index.html>.

Приведем пример окна среды разработки Geany с простейшей программой (см. рис. [3.2](#_bookmark20)).

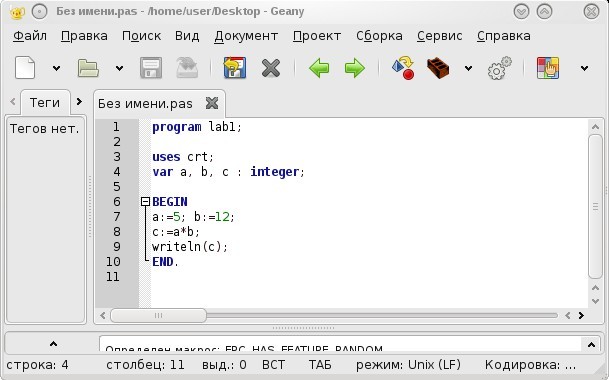


Рисунок 3.2: Вид окна среды разработки Geany и программы на языке Паскаль

Ввод данных, а также вывод результатов работы программы осуществляется в отдельном окне (см. рис. [3.3](#_bookmark22)).

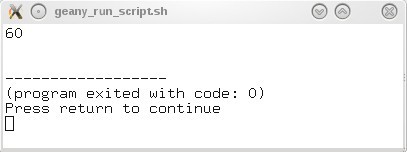


Рисунок 3.3: Окно ввода данных и вывода результатов в Geany

## п. 4. Среда программирования Free Pascal

Free Pascal (полное название Free Pascal Compiler, часто используется сокращение FPC) — свободно распространяемый компилятор языка программирования Pascal.

Официальный сайт, с которого можно скачать дистрибутив этой среды, расположен по адресу [http://www.freepascal.org/download.va](http://www.freepascal.org/download.var)r (если требуется установка под Windows, выбираем ссылку Win32, Win64 and WinCE, имя файла fpc-2.4.4.i386-win32.exe).

Информационный портал для разработчиков — <http://www.freepascal.ru/>.

Внешний вид окна программы приведен ниже (рис. [4.1](#_bookmark23)-[4.2](#_bookmark24)).

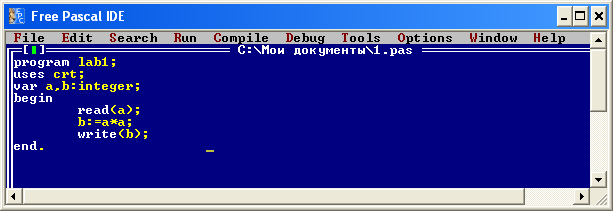


Рисунок 4.1: Графический интерфейс окна программы Free Pascal в ОС Windows

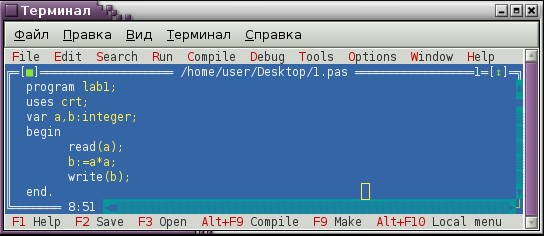


Рисунок 4.2: Графический интерфейс окна программы Free Pascal в ОС Linux

Прежде чем запустить программу на исполнение, требуется выполнить сохранение программы на диске и ее компиляцию (комбинация клавиш Alt+F9). Если при компиляции программы были обнаружены ошибки, то об этом будет выведено сообщение. Если же компиляция прошла успешно, то на экран будет выведено сообщение: *Compile successful. Press any key* (рис. [4.3](#_bookmark25)).

С помощью комбинации клавиш Ctrl+F9 программа запускается на исполнение.

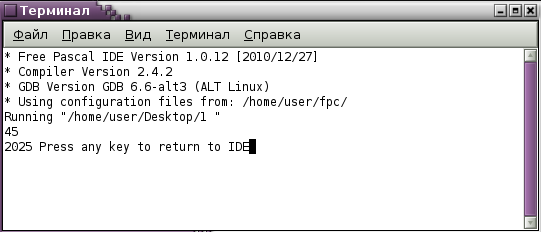


Рисунок 4.3: Вывод результата работы программы в строке терминала в OC Linux

## ГЛАВА III. ЛАБОРАТОРНЫЕ РАБОТЫ

## п. 1. Лабораторная работа № 1. Программирование алгоритмов линейной структуры

***Цель работы:*** знакомство с правилами подготовки и выполнения лабораторных работ; изучение структуры окна системы программирования, организации работы по набору, отладке, компиляции, запуска программы на исполнение; формирование совокупности знаний об операторах программ линейной структуры, умений анализировать исходные данные задачи, навыков разработки и программирования вычислительного процесса линейной структуры, навыков по отладке и тестированию программ, умений оформлять результаты своей деятельности в современном офисном пакете и аргументировать их, умений объяснять синтаксис операторов ввода, вывода, присваивания; развитие коммуникативных навыков взаимодействия в коллективе.

#### *Вопросы для подготовки к лабораторной работе:*

1. Что такое «алгоритм»? Каковы его свойства? Перечислите способы записи алгоритма.
2. Из каких элементов состоит блок-схема? Для чего предназначен каждый блок?
3. Что такое «язык программирования», «среда программирования»?
4. Какие языки называются компилируемыми, а какие интерпретируемыми? К какому типу относится язык программирования Паскаль?
5. Что такое «компиляция»?
6. Что такое «программа», «линейная программа»?
7. Какова структура программы, написанной на языке Паскаль?
8. Что такое «оператор»? Из каких операторов состоит линейная программа?
9. Для чего нужны комментарии в программе?
10. Запишите синтаксис оператора ввода, оператора вывода, оператора присваивания.
11. Для чего нужен раздел объявления переменных? Как объявляются переменные?
12. Что такое «тип данных»? Для чего нужен тип данных? Какие типы данных есть в Паскале?
13. Как записываются арифметические операции в языке Паскаль?
14. Что такое «переменная», «идентификатор»? Каковы правила записи идентификаторов?
15. Какие встроенные математические функции есть в Паскале?
16. Перечислите правила набора математических выражений на языке Паскаль.

#### *Ход работы:*

***Постановка задачи 1\_1:*** *выполнить пояснения к программе (выделить структурные единицы программы) и сформулировать задачу, для реше- ния которой программа написана. Набрать программу, проверить ее ра- ботоспособность. Сохранить в своей папке, создать exe-файл. Показать преподавателю.*

program digits; var

vklad:real; summa: real; procent:integer; begin

writeln('Введите сумму вклада'); readln(vklad);

writeln('Введите процент'); readln(procent); summa:=vklad+vklad/100\*procent; writeln('Сумма через год=', summa); end.

#### *Решение:*

Пояснения к программе:

|  |  |
| --- | --- |
| program digits; | **Заголовок программы** digits |
| var | **Раздел описания переменных** |
| vklad:real; | Сумма вклада: имя — vklad, тип – веще- ственный |
| summa: real; | Сумма через год: имя — summa, тип — ве- щественный |
| procent:integer; | Процентная ставка: имя — procent, тип — целый |
| begin | **Раздел операторов (основной блок)** |
| writeln('Введите сумму вклада'); | Вывод на монитор текстовой строки «Вве- дите сумму вклада» |
| readln(vklad); | Считывание с клавиатуры введенного зна- чения и сохранение его под именем vklad |
| writeln('Введите процент'); | Вывод на монитор текстовой строки «Вве- дите процент» |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| readln(procent); | Считывание с клавиатуры введенного зна- чения и сохранение его под именем procent | | | | |
| summa:=vklad+vklad/100\*pro cent; | Присваивание переменной summa значе- ния, вычисленного по формуле vklad+vklad/100\*procent | | | | |
| writeln('Сумма через год=', summa); | Вывод summa | на | экран | значения | переменной |
| end. | Завершение раздела операторов (текста программы) | | | | |

Формулировка задачи: вкладчик положил в банк некоторую сумму денег N под P процентов годовых. Какую сумму денег снимет вкладчик че- рез год?

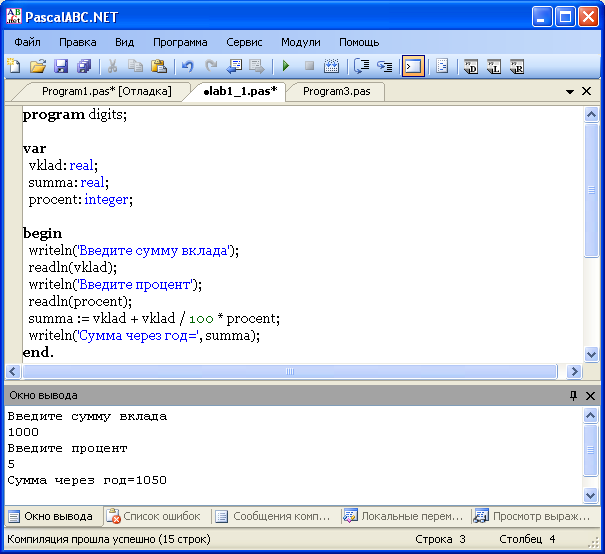


Рисунок 1.1: Проверка программы в системе PascalABC.Net

Проверку работоспособности программы можно проводить в любой системе программирования, например, в среде программирования

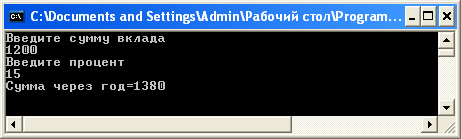
PascalABC.Net. В рабочем окне осуществим набор текста программы[9](#_bookmark12). Для улучшения читаемости программного кода можно выполнить автоформа- тирование кода (пункт меню «Сервис» → «Форматировать код»). Затем сохраним программу на диск под именем, например, lab1\_1.pas. Далее вы- полним компиляцию программы (меню «Программа» →

«Компилировать»). Если компиляция прошла успешно, то в строке состоя- ния появится сообщение «Компиляция прошла успешно (15 строк)»[10](#_bookmark13). Запустим программу на исполнение (меню «Программа» → «Выполнить»). Рабочее окно с программным кодом и результатами работы изображено на рис. [1.1](#_bookmark11)).

Перед повторным сохранением и перекомпиляцией файла в про- граммном коде прописать в конец программы команду readln;

В результате компиляции файла создается исполняемый файл lab1\_1.exe. Теперь, не загружая среду программирования, мы можем запу- стить программу на исполнение (двойной щелчок левой кнопки мыши по значку файла lab1\_1.exe).

Работа исполняемого файла:



**Учебное задание:** доработайте программу так, чтобы она вычис- ляла сумму денег на вкладе через два года. Создайте exe-файл. Продемонстрируйте работу программы преподавателю.

***Постановка задачи 1\_2:*** *вычислить значения переменных по заданным*

 *bt*

*расчетным формулам y*  *à*

sin(*at*  *b*) 

*s*  *b*2 tg(*t* 2 cos 4*t*)  *a*

*bt*  2*a , a*>0 . Разработать алгоритм

решения задачи в виде блок-схемы и программы. В*ывод вычисленных*

*значений переменных осуществить в соответствии со следующей раз- меткой строк:*

\*\*\*\* РЕЗУЛЬТАТЫ ВЫЧИСЛЕНИЙ \*\*\*\*

(пропуск 1-ой строки)

1. Для изменения размера шрифта и его начертания воспользуйтесь меню Сервис → Настройки, вкладка

«Редактор».

1. В случае наличия ошибок в программном коде, необходимо их устранить, сохранить исправленную программу и ее перекомпилировать.

имя переменной = значение

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* имя переменной = значение

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Вычислить значения y и s при а=0,1; b=2; t=0,78.*

#### *Решение:*

Воспользуемся логарифмическим тождеством: *a x* =*ex* ln *a* [11](#_bookmark14). Тем самым мы сведем наше выражение к использованию функций,

которые являются встроенными в языке Паскаль. Применительно к нашему заданию, получим: *a*−*bt* =*e*−*bt* ln *a* . Далее необходимо записать формулы для нахождения значений выражений по правилам языка Паскаль. Для этого воспользуемся разделами

«Описание операций» и

«Встроенные математические функции» из первой главы.

***Программа:* program** lab\_1\_2; **var** y, a, b, t, s :real; **begin**

write('Введите a, b, t: '); read(a, b, t);

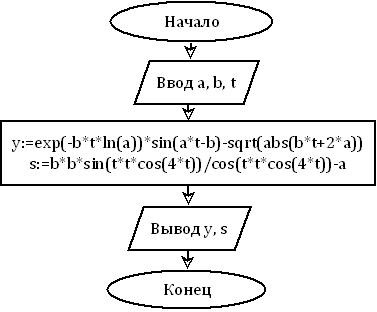


Рисунок 1.2: Блок-схема к задаче 1\_2

y := exp(-b \* t \* ln(a)) \* sin(a \* t - b) - sqrt(abs(b \* t + 2\*a)); s := b\*b \* sin(t \* t \* cos(4 \* t))/cos(t \* t \* cos(4 \* t))- a; writeln('\*\*\*\* РЕЗУЛЬТАТЫ ВЫЧИСЛЕНИЙ \*\*\*');

writeln; writeln('y=',y:4:4);

writeln('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'); writeln('s=',s:4:4); writeln('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'); **end.**

#### *Контрольный счет:*

1. Дело в том, что в языке Паскаль нет встроенной функции для возведения выражения в степень.

Введите a, b, t: 0.1 2 0.78

\*\*\*\* РЕЗУЛЬТАТЫ ВЫЧИСЛЕНИЙ \*\*\*

y=-35.4182

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* s=-2.8853

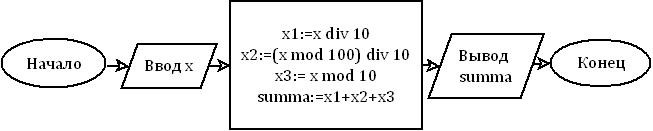
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Учебное задание:** измените программу так, чтобы она вычисляла площадь круга и длину окружности радиуса r.

***Постановка задачи 1\_3:*** *с клавиатуры вводится трехзначное число. На- писать программу, вычисляющую сумму цифр числа. Разработать блок- схему алгоритма. Выполнить трассировку программы для числа 152.*

#### *Решение:*

Введем обозначения: x — трехзначное число, x1 — первая цифра числа, x2 — вторая цифра числа, x3 — третья цифра числа, summa — сумма цифр числа x1+x2+x3. Для выделения цифр числа воспользуемся командами div и mod для нахождения целой части числа и остатка от целочисленного деления соответственно.



***Программа:* program** lab1\_3; **var**

Рисунок 1.3: Блок-схема к задаче 1\_3

x, x1, x2, x3, summa : integer;

### begin

writeln('Введите трехзначное число:'); readln(x);

x1:=x **div** 100;

x2:=(x **mod** 100) **div** 10; x3:= x **mod** 10; summa:=x1+x2+x3;

writeln('Сумма цифр числа=', summa);

**end**.

#### *Трассировка программы:*

Проведем пошаговое выполнение программы непосредственно в самой среде программирования. Для этого после сохранения и компиля- ции программы выбираем пункт меню «Программа» → «Шаг без входа в подпрограмму» (или клавиша F8). Далее последовательно нажимаем кла- вишу F8 для пошагового выполнения программы: при выполнении шага

«ввод х» с клавиатуры вводим трехзначное число, например, 152. Затем осуществляем пошаговый просмотр работы программы. После чего выби- раем пункт меню «Вид» → «Локальные переменные», которое содержит в себе значения переменных на каждом шаге выполнения программного кода.

Теперь составим таблицу трассировки:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Оператор** | **Значения переменных** | | | | |
| **x** | **x1** | **x2** | **x3** | **summa** |
| begin | пусто | пусто | пусто | пусто | пусто |
| readln(x); | 152 | пусто | пусто | пусто | пусто |
| x1:=x div 100; | 152 | 1 | пусто | пусто | пусто |
| x2:=(x mod 100) div 10; | 152 | 1 | 5 | пусто | пусто |
| x3:= x mod 10; | 152 | 1 | 5 | 2 | пусто |
| summa:=x1+x2+x3; | 152 | 1 | 5 | 2 | 8 |
| writeln('Сумма цифр числа=', summa); | Вывод на экран: Сумма цифр числа= 8 | | | | |
| end | Выход из программы | | | | |

#### *Контрольный счет:*

Введите трехзначное число: 152 Сумма цифр числа =8

**Учебное задание:** модифицируйте программу так, чтобы она на- ходила произведение цифр четырехзначного числа.

#### *Контрольные вопросы:*

1. Какова структура программы на языке Паскаль?
2. Перечислите операторы, используемые в линейных программах.
3. Запишите синтаксис операторов ввода и вывода данных.
4. Запишите синтаксис оператора присваивания.
5. Что такое «трассировка программы»?
6. Нарисуйте схематично блок-схему любой линейной программы.
7. Что значит «объявить переменную»? Объявите переменную x целочисленного типа данных.
8. Какого типа данных могут быть переменные s=1,7 и m=8?
9. Для чего нужен формат вывода данных? Как он задается?
10. Для чего предназначены операторы div и mod?
11. Запишите встроенные математические функции языка Паскаль.

### п. 2. Лабораторная работа № 2. Программирование алгоритмов разветвляющейся структуры

***Цель работы:*** формирование совокупности знаний об операторах программ разветвляющейся структуры; овладение практическими навыками разработки, программирования вычислительного процесса раз- ветвляющейся структуры, умений анализировать исходные данные задачи, умений оформлять результаты своей деятельности в современном офисном пакете и аргументировать их, умений объяснять синтаксис операторов ветвления; получение дальнейших навыков по отладке и тестированию программы.

#### *Вопросы для подготовки к лабораторной работе:*

1. Что такое «разветвляющаяся программа»? Из каких операторов состоит разветвляющаяся программа?
2. Запишите синтаксис оператора безусловного перехода.
3. Запишите синтаксис условного оператора (полная и неполная форма).
4. Запишите синтаксис оператора выбора.
5. Что такое «тестовый набор данных»?
6. Что такое «трассировка программы»?
7. Как выполнить компиляцию программного кода?

#### *Ход работы:*

***Постановка задачи 2\_1:*** *выполнить пояснения к программе (выделить структурные единицы программы) и сформулировать задачу, для решения которой программа написана. Набрать программу, проверить ее работо- способность (для этого подобрать тестовый набор данных). Сохранить в своей папке, создать exe-файл. Показать преподавателю.*

**program** krug;

**var** x,y:real; x0,y0,r,s:real;

**begin**

writeln('Введите координаты центра окружности');

readln(x0,y0); writeln('Введите радиус'); readln(r);

writeln('Введите координаты точки'); readln(x,y);

s:=sqrt((x0-x)\*(x0-x)+(y0-y)\*(y0-y));

**if** s<r **then**

writeln('Лежит внутри круга')

**else if** s=r **then**

writeln('Лежит на окружности')

**else**

writeln('Лежит вне круга');

**end**.

#### *Решение:*

Пояснения к программе:

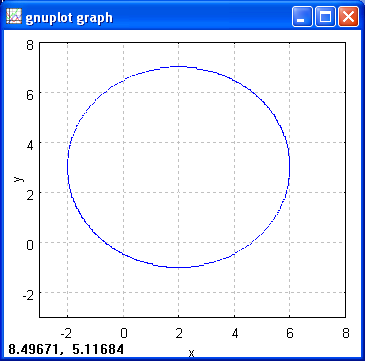
|  |  |
| --- | --- |
| program krug; | **Заголовок программы** krug |
| var | **Раздел описания переменных** |
| x,y:real; | Координаты точки на плоскости: имена — x, y, тип – вещественный |
| x0,y0,r,s:real; | Координаты центра окружности, радиус, рас- стояние: имена — x0, y0, r, s, тип — веществен- ный |
| begin | **Раздел операторов (основной блок)** |
| writeln('Введите координаты центра окружности'); | Вывод на экран строки комментария 'Введите координаты центра окружности' |
| readln(x0,y0); | Считывание с клавиатуры введенных значе- ний и сохранение под именами x0, y0 |
| writeln('Введите радиус'); | Вывод на экран строки комментария 'Введите радиус' |
| readln(r); | Считывание с клавиатуры введенного значе- ния и сохранения под именем r |
| writeln('Введите координаты точки'); | Вывод на экран строки комментария 'Введите координаты точки' |
| readln(x,y); | Считывание с клавиатуры значений перемен- ных x, y |
| s:=sqrt((x0-x)\*(x0-x)+(y0- | Вычисление расстояния от центра окружно- |

|  |  |
| --- | --- |
| y)\*(y0-y)); | сти до точки на плоскости, присваивание по- лученного значения переменной s |
| if s<r then | Проверка условия: s<r |
| writeln('Лежит внутри круга') | Если условие истинно, вывести на экран стро- ку 'Лежит внутри круга' |
| else if s=r then | Если условие ложно, то проверка условия: s=r |
| writeln('Лежит на окружности') | Если условие s=r истинно, то вывод на экран строки 'Лежит на окружности' |
| else  writeln('Лежит вне круга'); | Если ложно s<r и s=r (то есть s>r), то вывод на экран строки 'Лежит вне круга' |
| end. | **Завершение раздела операторов (текста программы)** |

Формулировка задачи: Определить, принадлежит ли точка с координа- тами (x, y) кругу с центром в точке (x0, y0) и радиусом r.

#### *Тестовый набор данных:*

Выполним построение окружности с центром в точке (2, 3) и радиуса r=4 (см. рис. [2.1](#_bookmark15)). Тогда подберем координаты точ- ки на плоскости:



1. вариант: (3, 5) — лежит внутри круга
2. вариант: (6, 3) лежит на окружности
3. вариант: (10,10) лежит вне круга

#### *Тестирование программы:*

* 1. вариант:

Введите координаты центра окружности 2 3

Введите радиус 4

Введите координаты точки 3 5 Лежит внутри круга

* 1. вариант:

Введите координаты центра окружности 2 3

Введите радиус 4

Введите координаты точки 6 3 Лежит на окружности

* 1. вариант:

Введите координаты центра окружности 2 3

Введите радиус 4

Введите координаты точки 10 10 Лежит вне круга

Рисунок 2.1: Окружность на плоскости

***Учебное задание:*** измените программу так, чтобы она определяла принадлежность точки квадрату со стороной x=4 с центром в нача- ле координат.

***Постановка задачи 2\_2:*** *пользователь ввел с клавиатуры значение пере- менной х. Определите, какое значение выведет компьютер на дисплей, если х равен:* 5; 2; -1; -6; 17.

**program** lab2\_2; **var** f, x: integer; **begin**

readln(x);

**case** x **of**

1, 3, 5, 7, 9 : f:=4\*x-1;

2, 4, 6, 8 : f:=x\*x;

10..100 : f:=x-10;

-3..-1 : f:=1;

**else** f:=0; **end**; writeln(f); **end**.

#### *Решение:*

Основной блок программы составляет оператор выбора по ключу x: если введенное значение x будет равно 1 или 3 или 5 или 7 или 9, то значе- ние переменной f будет вычисляться по формуле 4x+1, если x принадлежит отрезку от 10 до 100, то f=x-10 и т. д. Построим таблицу трассировки для зна- чения x=5.

|  |  |  |
| --- | --- | --- |
| **Оператор** | **Значения переменных** | |
| **x** | **f** |
| begin | пусто | пусто |
| readln(x); | 5 | пусто |
| **case** x **of** | Начало оператора выбора по ключу x=5 | |
| 1, 3, 5, 7, 9 : f:=4\*x-1; | x=5 входит в этот диапазон | 19 |
| 2, 4, 6, 8 : f:=x\*x; | x=5 не входит в этот диапазон | 19 |
| 10..100 : f:=x-10; | x=5 не входит в этот диапазон | 19 |
| -3..-1 : f:=1; | x=5 не входит в этот диапазон | 19 |
| **else** | Выбор если x не попал ни в один из диапазонов | |

|  |  |  |
| --- | --- | --- |
| f:=0; | 5 | 19 |
| **end**; | Конец оператора выбора | |
| writeln(f); | Вывод на экран: 19 | |
| end | Выход из программы | |

Таким образом, получили при x=5 значение f=19. Аналогично рассу- ждая, будем иметь при x=2 значение f=4, при x=-1 значение f=1, при x=-6 зна- чение f=0, при x=17 значение f=7.

***Постановка задачи 2\_3:*** *вычислить значение функции*

*Q*  

 *x*2  7 / *x*2 ,

 *ax*3  7 *x* ,

 ln(*x*  7 *x*  *a* ),



*x*  1,4

*x*  1,4 *. Осуществить вывод полученного значения функции. Реализовать*

*x*  1,4

*алгоритм в виде блок-схемы и на языке программирования. Выполнить программу и протестировать все ветви алгоритма.*

#### *Программа:*

В программе проведем вычис- ление значения функции q по форму- ле, соответствующей значению x<1,4. Если же x не попал в этот интервал, то с помощью условного оператора пересчитаем значение q по одной из двух других формул.

**program** lab2\_3;

### const

a=1.65;

### var

q, x:real;

**begin** write('Введите x: '); read(x);

q:=pi\*x\*x-7/(x\*x);

**if** x=1.4 **then** q:=a\*x\*x\*x+7\*sqrt(x); **if** x>1.4 **then** q:=ln(x+7\*sqrt(abs(x+a)));

writeln(' q = ',q:4:4);

**end**.

#### *Тестирование программы:*

1. вариант:

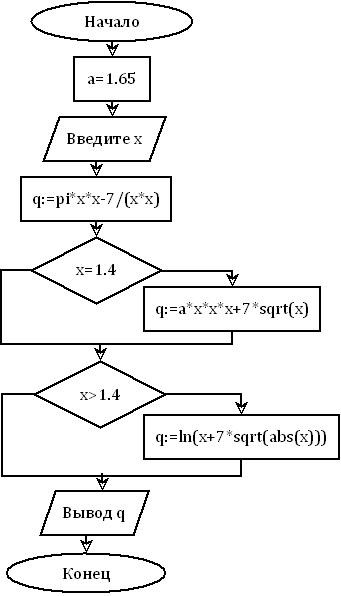


Рисунок 2.2: Блок-схема к заданию 2\_3

Введите x: 1

q = -3.8584

1. вариант:

Введите x: 1.4

q = 12.8101

1. вариант:

Введите x: 2

q = 2.7326

***Учебно-исследовательское задание:*** проанализируйте работу про- граммы. Найдите значение x, при котором программа работает не- корректно (функция при некотором x не определена). Устраните найденную ошибку и доработайте программу.

***Постановка задачи 2\_4:*** *Требовалось написать программу, которая реша- ет уравнение «ax+b=сx+d» относительно x для любых чисел a, b, c, d, введен- ных с клавиатуры. Все числа считаются действительными. Программист торопился и написал программу неправильно.*

**var** a, b, c, d: real; **begin** readln(a,b,c,d);

**if** d-b=0 **then** write('x= 0') **else**

**if** a-c=0 **then**

write('нет решений')

**else**

write('x =',(-b+d)/(a-c));

**end**.

*Предложить свой способ доработки программы, чтобы не было случаев ее неправильной работы. Реализовать алгоритм решения задачи в виде блок- схемы.*

#### *Решение:*

***Математическая модель:***

Перепишем уравнение в виде: (a-c)x=d-b. Рассмотрим возможные варианты значений переменных a, b, c, d:

* а-с=0 и d-b=0 — получаем уравнение 0x=0 – верно для любого х;
* а-c=0 и d-b<>0 — получаем уравнение 0x=d-b — нет решений;
* а-c<>0 и d-b — любое число (в том числе и 0) — получаем уравнение (a-c)x=d-b — решение х=(d-b)/(a-c).

В предложенном тексте программы не предусмотрен вариант решения «Любое число», поэтому в случае, если a=c, b=d программа будет

работать неверно. ***Исправленная программа:* var** a, b, c, d, x: real;

**begin** readln(a,b,c,d); **if** a-c=0 **then**

**if** d-b=0 **then**

write('x- любое число')

### else

write('Решений нет')

### else

write('x =',(-b+d)/(a-c));

**end**.

#### *Контрольный счет:*

1 вариант:

2 3 2 3

x- любое число 2 вариант:

2 1 2 3

Решений нет 3 вариант:

1 2 3 4

x =-1

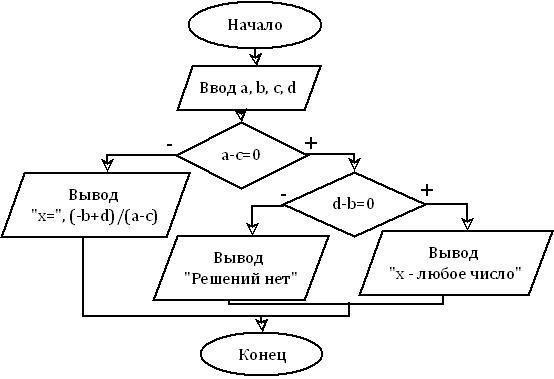


Рисунок 2.3: Блок-схема к заданию 2\_4

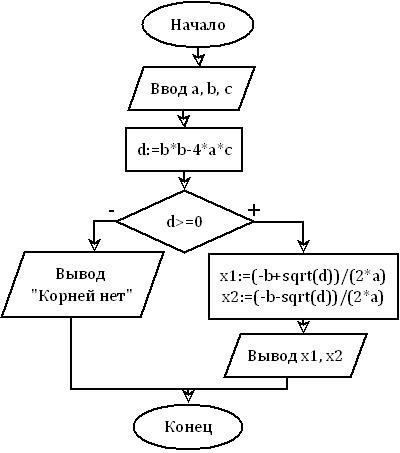
**Учебное задание:** модифицировать программу так, чтобы она ре- шала уравнение ax-b=cx-d для любых значений коэффициентов a, b, c, d.

***Постановка задачи 2\_5:*** *на основе блок-схемы (рис.* [*2.4*](#_bookmark16)*) реализовать про- грамму на языке Паскаль. Выполнить трассировку программы для a=1, b=2, c=-8.*

#### *Решение:*

Блок-схема является реализацией алгоритма решения квадратного уравнения вида *a x*2+*b x*+*c*=0 .

#### *Программа:*



**var**

a, b, c, d, x1, x2: real;

### begin

readln(a,b,c);

writeln ('Исходное уравнение:', a, 'x\*x+', b, 'x+', c, '=0');

d:=b\*b-4\*a\*c;

**if** d<0 **then**

write('Действительных корней нет')

### else begin

x1:=(-b-sqrt(d))/(2\*a);

x2:=(-b+sqrt(d))/(2\*a);

write('Решение: x1=', x1:2:2, ' x2=', x2:2:2);

### end end.

Рисунок 2.4: Блок-схема к заданию 2\_5

***Трассировка программы:*** *a=1, b=2, c=-8.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Оператор** | **Значения переменных** | | | | | |
| **a** | **b** | **c** | **d** | **x1** | **x2** |
| begin | пусто | пусто | пусто | пусто | пусто | пусто |
| readln(a,b,c); | 1 | 2 | -8 | пусто | пусто | пусто |
| d:=b\*b-4\*a\*c; | 1 | 2 | -8 | 36 | пусто | пусто |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| d<0 – ложь | 1 | 2 | -8 | 36 | пусто | пусто |
| x1:=(-b-  sqrt(d))/(2\*a); | 1 | 2 | -8 | 36 | -4 | пусто |
| x2:=(-  b+sqrt(d))/ (2\*a); | 1 | 2 | -8 | 36 | -4 | 2 |
| write('Решение  : x1=', x1:2:2, '  x2=', x2:2:2); | Вывод на экран: Решение: x1=-4, x2=2 | | | | | |
| end | Выход из программы | | | | | |

#### *Контрольный счет:*

1. вариант:

1 2 -8

Исходное уравнение: 1x\*x+2x+-8=0 Решение: x1=-4.00 x2=2.00

1. вариант:

1 2 3

Исходное уравнение: 1x\*x+2x+3=0 Корней нет

**Учебное задание:** модифицируйте программу так, чтобы она про- веряла коэффициент a на равенство нулю: если a=0, то вывести сообщение «Уравнение не является квадратным».

***Постановка задачи 2\_6:*** *на основе блок-схемы (рис.* [*2.5*](#_bookmark17)*) сформулировать условие задачи и реализовать программу на языке Паскаль.*

#### *Решение:*

Постановка задачи: разработать программу, которая по введенному символу компьютерного алфавита выводит одно из сообщений: это цифра, это буква, это точка. Для любого другого символа выводится сообщение «Это другой символ».

***Программа:* program** lab2\_6; **var** x:char;

### begin

writeln('Введите символ'); readln(x);

**case** x **of**

'0'..'9': writeln('Это цифра');

'a'..'z': writeln('Это буква'); '.': writeln('Это точка')

**else** writeln ('Это другой символ');

**end**; **end**.

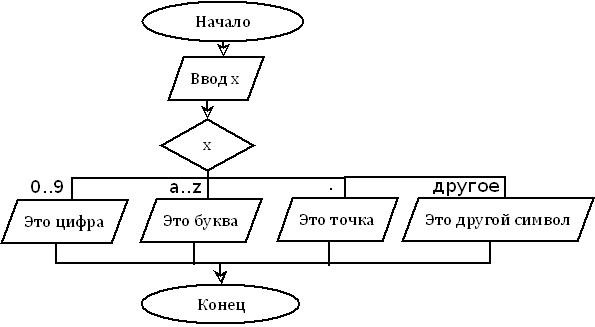


Рисунок 2.5: Блок-схема к заданию 2\_6

**Учебное задание:** модифицировать программу так, чтобы при вводе символов русского алфавита выводилось сообщение «Это русская буква», а при вводе символов латинского алфавита выво- дилось сообщение «Это латинская буква».

#### *Контрольные вопросы:*

* 1. Перечислите операторы, используемые в разветвляющихся программах.
  2. Запишите синтаксис операторов условного и безусловного перехода, выбора.
  3. Как правильно подобрать тестовый набор данных к программе, содержащей оператор ветвления?
  4. В каком случае необходимо заключать операторы в операторные скобки begin … end?
  5. Что может храниться в переменной типа char?
  6. Как записываются операции сравнения в языке Паскаль?
  7. Как записываются логические операции в языке Паскаль?
  8. Как правильно записать сложное условное выражение в языке Паскаль?
  9. Нарисуйте схематично блок-схему программы с использованием

полной (неполной) формы условного оператора.

* 1. Приведите пример задачи, алгоритм решения которой является разветвляющимся.

## п. 3. Лабораторная работа № 3. Программирование алгоритмов циклической структуры с заданным числом повторений

***Цель работы:*** формирование совокупности знаний об операторах программ с использованием цикла с параметром; овладение практическими навыками разработки, программирования вычислительного процесса циклической структуры с заданным числом повторений, умения объяснять синтаксис и назначение оператора цикла с параметром; получение дальнейших навыков по отладке и тестированию программы; умений представлять полученные результаты своей деятельности.

#### *Вопросы для подготовки к лабораторной работе:*

1. Как произвести пошаговое выполнение программы?
2. Что такое цикл? Какая программа называется циклической?
3. Запишите синтаксис оператора цикла с параметром.
4. Что такое «параметр цикла»? Что такое «тело цикла»?
5. С каким шагом перебираются значения параметра цикла? От чего это зависит?
6. Что требуется сделать для того, чтобы в теле цикла выполнялись несколько операторов?
7. Как найти, сколько раз выполняется цикл?
8. Приведите примеры задач, для решения которых используется структура цикл.
9. Объясните, как работает формула s:=s+i.

#### *Ход работы:*

***Постановка задачи 3\_1:*** *выполнить пояснения к программе (выделить структурные единицы программы) и сформулировать задачу, для решения которой программа написана. Набрать программу, проверить ее работоспособность. Сохранить в своей папке, создать exe-файл. Показать преподавателю.*

**program** lab3\_1; **var** i, n, s: integer; **begin**

writeln('Введите натуральное число n:'); readln(n);

s := 0;

**for** i := 1 **to** n **do**

s := s + i; writeln('Сумма =', s); **end**.

#### *Решение:*

Пояснения к программе:

|  |  |
| --- | --- |
| **program** lab3\_1; | **Заголовок программы** lab3\_1 |
| **var** | **Раздел описания переменных** |
| i, n, s: integer; | Параметр цикла: имя i, тип – целое число Натуральное число: имя n, тип – целое число Сумма натуральных чисел: имя s, тип — целое число |
| **begin** | **Раздел операторов (основной блок)** |
| writeln('Введите натуральное число n:'); | Вывод на экран строки комментария 'Введите натуральное число n:' |
| readln(n); | Считывание с клавиатуры значения натурально- го числа и сохранения его под именем n |
| s := 0; | Обнуляем значение суммы чисел |
| **for** i := 1 **to** n **do** | Организация цикла с параметром по переменной i, изменяющейся от 1 до n с шагом +1 |
| s := s + i; | Тело цикла: повторение операции добавления к сумме очередного натурального числа |
| writeln('Сумма =', s); | Вывод на экран строки «Сумма =» и значения, хранимого под именем s |
| **end.** | Завершение раздела операторов (текста про- граммы) |

Формулировка задачи:

***Контрольный счет:*** Введите натуральное число n: 5

Сумма = 15

***Учебное задание:*** модифицировать программу так, что в ней вы- числялось произведение первых N натуральных чисел.

найти сумму первых N натуральных чисел.

***Постановка задачи 3\_2:*** *определить, для решения какой задачи написа- на программа. Составить таблицу трассировки.*

**program** lab3\_2; **var** x:1..9; s:real; **begin**

s:=0;

**for** x:=1 **to** 9 **do** s:=s+sqrt(x); writeln(s);

**end**.

#### *Решение:*

Таблица трассировки:

|  |  |  |
| --- | --- | --- |
| **Оператор** | **Значения переменных** | |
| **x** | **s** |
| begin | пусто | пусто |
| s:=0; | пусто | 0 |
| **for** x:=1 **to** 9 **do** | Начало цикла с параметром: х меняется от 1 до 9 с шагом +1 | |
| s:=s+sqrt(x); | x=1 | *s*=0+√1=√1 |
|  | x=2 x=3 x=4 x=5 x=6 x=7 | *s*=√ 1+√ 2  *s*=√ 1+√ 2+√ 3  *s*=√ 1+√ 2+√ 3+√ 4  *s*=√ 1+√ 2+√ 3+√ 4+√ 5  *s*=√ 1+√ 2+√ 3+√ 4+√ 5+√ 6  *s*=√ 1+√ 2+√ 3+√ 4+√ 5+√ 6+√7  *s*=√ 1+√ 2+√ 3+√ 4+√ 5+√ 6+√7+√8 |
|  | x=8 | *s*=√ 1+√ 2+√ 3+√ 4+√ 5+√ 6+√7+√8+√ 9 |
|  | x=9 |  |
| writeln(s); | Вывод на экран: 19.3060005260357 | |
| end | Выход из программы | |

Таким образом, программа вычисляет сумму квадратных корней из чисел от 1 до 9.

***Учебное задание:*** модифицировать программу так, чтобы в ней находилась сумма квадратных корней из натуральных чисел до числа N.

***Постановка задачи 3\_3:*** *разработать программу вычисления числа*

*сочетаний по формуле*

*C* =

*m*

*n*

*n* !

*m* !( *n*−*m*)!

*, где и вводятся с клавиатуры.*

*n m*

*Разработать блок-схему алгоритма решения задачи.*

#### *Решение:*

*m* = *n* !

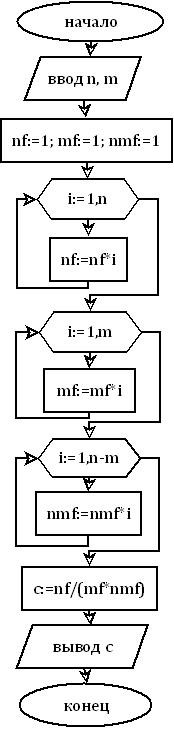
#### *Математическая модель: C*n

*m* !=1⋅2⋅3⋅... *m .*

***Программа:* program** lab3\_3; **var**

*m* !( *n*−*m*)! *, где n* !=1⋅2⋅3⋅... *n ,*

i, n, m:integer;



nf, mf, nmf:longint; c:real;

**begin**

writeln('Введите два натуральных числа'); readln(n,m);

nf:=1; mf:=1; nmf:=1;

**for** i:=1 **to** n **do**

nf:=nf\*i;

**for** i:=1 **to** m **do**

mf:=mf\*i;

**for** i:=1 **to** n-m **do**

nmf:=nmf\*i; c:=nf/(mf\*nmf);

writeln('Число сочетаний = ', c:8:0);

***end****.*

2 = 4 !

3⋅4

#### *Тестовый набор данных:* C4

2!(4−2)!= 1⋅2 =6 ,

4 4 ! 1

*C*4 = 4 !( 4−4)!= 0 !=1

***Контрольный счет:*** Введите два натуральных числа 4 2

Число сочетаний = 6

Введите два натуральных числа 4 4

Число сочетаний = 1

Рисунок 3.1: Блок- схема к заданию 3\_3

***Постановка задачи 3\_4:*** *используя численный метод трапеций, вычис-*



2

*лить значение интеграла z*  cos *xdx . Число разбиений отрезка интегри-*

0

*рования n*=50 *. Включить в программу вычисление точного значения ин- теграла. Вывести приближенное, точное значения интеграла и относи- тельную погрешность вычислений в процентах. Найти предельную по- грешность формулы трапеций. Разработать алгоритм в виде блок-схе-*

*мы для решения задачи.*

#### *Решение:*

***Математическая модель:***

Метод трапеций — метод численного интегрирования функции одной переменной, заключающийся в замене на каждом элементарном отрезке подынтегральной функции на многочлен первой степени, то есть линейную функцию. Площадь под графиком функции аппроксимируется прямоуголь- ными трапециями [(Рисунок 3.2](#_bookmark20)).

Промежуток интегрирования (*a ,b*) делим точками *x*1, *x*2, ... *xn*−1 на *n*

равных частей; длина каждой части *h*= *b*−*a* .

*n*

Для единообразия полагаем *a*= *x*0 *, b*= *xn* . Тогда имеем[12](#_bookmark21):

*b*

*a n* [ 2 ]

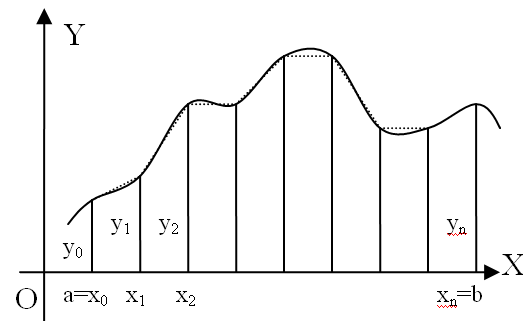
∫ *y* ( *x* ) *dx*≈ *b*−*a*

*y*0+ *y n*

+ *y* + *y* +...+ *y*

(1)

Это – формула трапеций. Она дает общую площадь трапе- ций, показанных на рис. [3.2](#_bookmark20).



Предельная погрешность формулы [(1)](#_bookmark19) составляет

(*b*−*a* )3

1 2 *n*−1

12 *n*2

*M* 2 , где *M* 2 — наи-

большее значение ∣*y ' '* ( *x*)∣ в промежутке (*a ,b*) .

Применительно к нашей задаче, для нахождения прибли- женного значения интеграла формула [(1)](#_bookmark19) примет вид:

Рисунок 3.2: Аппроксимация площади под графиком функции прямоугольными трапециями

*a n* [ 2 ]

*b*

∫cos *x dx*≈ *b*−*a* cos (*b*)+cos ( *a*)+cos ( *x* )+cos( *x* )+...+cos ( *x* ) .

1 2 *n* −1

Для нахождения точного значения интеграла воспользуемся формулой

*b b*

Ньютона-Лейбница ∫ *f* ( *x* ) *dx*=*F* (*b*)−*F* (*a* ) : ∫cos *x dx*=sin( *b*)−sin( *a*) .

*a a*

Для вычисления предельной погрешности формулы найдем

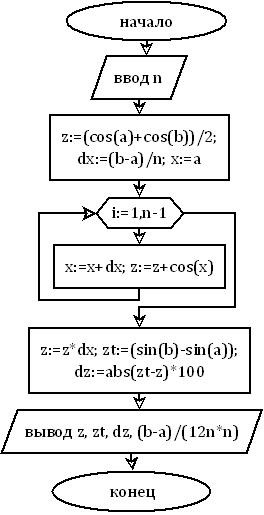
∣*y ' '* ( *x*)∣=∣(cos *x*) *' '*∣=∣−cos *x*∣ . На отрезке [ 0,π / 2] наибольшее значение второй производной достигается в точке *x*=0 *M* 2=1 .

***Программа:* program** lab3\_4; **const** a=0; b=pi/2; **var**

12 Выгодский М.А. Справочник по высшей математике. - М., 2000. - 864с. - С.480

x, z, zt, dz, dx:real; n, i:integer;

### begin



write('Введите n: '); read(n);

z:=(cos(a)+cos(b))/2;

dx:=(b-a)/n; x:=a; **for** i:=1 **to** n-1 **do begin**

x:=x+dx; z:=z+cos(x);

**end**;

z:=z\*dx; zt:=(sin(b)-sin(a)); dz:=abs(zt-z)\*100; writeln('Приближенное значение

z=',z:9:6);

writeln('Точное значение zt=',zt:9:6); writeln('Относительная погрешность',

dz:9:6);

writeln('Предельная погрешность:', (b- a)/(12\*n\*n)\*1:9:6);

**end**.

#### *Контрольный счет:*

Введите n: 50

Приближенное значение z= 0.999918 Точное значение zt= 1.000000 Относительная погрешность 0.008225

Предельная погрешность: 0.000052

***Учебное задание:*** модифицировать программу так, чтобы она вы-

π

числяла методом трапеций значение интеграла ∫sin *x dx* . Вычис-

0

лить предельную погрешность формулы трапеций.

Рисунок 3.3: Блок-схема к заданию 3\_4

***Постановка задачи 3\_5:*** *разработать программу вычисления значения числа* π *методом Монте-Карло. Выполнить прогонку программы по* 5 *раз при каждом из* n=100, n=1000, n=5000, n=50000*, где* n *— количество случайных чисел. На основе полученных результатов сделать вывод. Выделить структурные единицы программы и выполнить пояснения к ним. Разработать алгоритм в виде блок-схемы для решения задачи.*

#### *Решение:*

***Математическая модель:***

Вспомним из курса геометрии, что площадь круга вычисляется по фор-

муле: *S*=π *r*2 . Если мы возьмем круг радиуса *r* =1 , то площадь круга *S*=π . Та- ким образом, задача нахождения числа π сводится к задаче нахождения пло- щади круга. Применим метод Монте-Карло. Для этого поместим наш круг ра- диуса *r* =1 в квадрат со стороной *a*=2 . Будем наугад (случайным образом) бросать точки в этот квадрат. Естественно предполагать, что чем больше площадь фигуры, тем чаще в нее будут попадать точки: при большом числе точек, наугад выбранных внутри квадрата, доля точек, содержащихся в кру- ге, приближенно равна отношению площади круга к площади квадрата.

Если через tochki\_kvadrata обозначить число тех наугад выбранных то- чек ( *x , y* ) , которые содержатся в квадрате, а tochki\_kryg — те из них, которые

попали в круг, то площадь *S*= *tochki kryg a*2

*tochkikvadrata*

, или *S*=4 *tochkikryg* .

*tochki kvadrata*

Так как координаты центра окружности для нашей задачи не важны, то возьмем (для простоты расчетов) круг с центром в начале координат. Тогда:

* точка принадлежит квадрату, если −1≤*x*≤1 и −1≤ *y*≤1 ;
* точка принадлежит кругу, если *x*2+ *y*2≤1 .

|  |  |
| --- | --- |
| ***Программа:*** | ***Пояснения:*** |
| **Program** lab3\_5; | **Заголовок программы** |
| **var** | **Раздел описания переменных** |
| x,y:real; | Координаты точки: имена x, y |
| n,tochki\_kruga,tochki\_kvad:integer; s: real; | Количество точек: имя n  Количество точек, попавших в круг: имя tochki\_kruga  Параметр цикла: имя tochki\_kvad Значение числа пи: имя s |
| **begin** | **Операторный блок** |
| Writeln ('Вычисление числа ПИ'); | Вывод строки комментария на экран |
| rite ('Введите количество случайных | Вывод строки комментария на экран |
| чисел n='); |
| Read (n); | Считывание числа, введенного с кла- виатуры, сохранение его значения под именем n |
| tochki\_kruga:=0; | Обнуление переменной tochki\_kruga |
| **for** tochki\_kvad:=0 **to** n **do** | Оператор цикла с параметром: пере- бор n точек квадрата) |
| **begin** | Начало цикла |

W

|  |  |
| --- | --- |
| x:=(0.5-random)\*2; y:=(0.5- random)\*2; | Вычисление координат точки внутри квадрата со стороной 2: −1≤*x*≤1 и  −1≤ *y*≤1 |
| **if** x\*x+y\*y<=1 **then** | Проверка: если координаты x и y точ- ки удовлетворяют условию *x*2+ *y*2≤1 |
| tochki\_kruga:=tochki\_kruga+1; | Если условие истинно: увеличиваем количество точек внутри круга на 1 |
| **end**; | Конец цикла |
| s:=tochki\_kruga/(tochki\_kvad-1)\*4; | Вычисление значения пи |
| Writeln ('ПИ=', s:2:6); | Вывод на экран значения пи |
| **end**. | **Конец программы** |

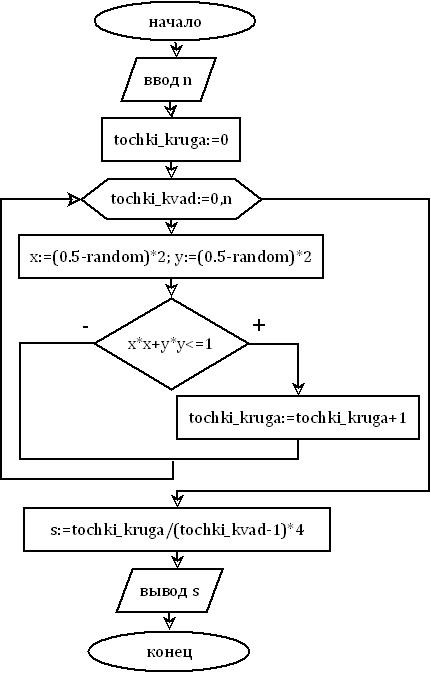


Рисунок 3.4: Блок-схема к заданию 3\_5

#### *Контрольный счет:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Количество случайных точек** | **Значение пи** | **Количество случай- ных точек** | **Значение пи** |
| n=100 | 2.868687 | n=5000 | 3.116623 |
| n=100 | 3.030303 | n=5000 | 3.179836 |
| n=100 | 3.111111 | n=5000 | 3.152631 |
| n=100 | 3.272727 | n=5000 | 3.166233 |
| n=100 | 3.151515 | N=5000 | 3.114223 |
| n=1000 | 3.139139 | n=50000 | 3.146623 |
| n=1000 | 3.215215 | n=50000 | 3.142623 |
| n=1000 | 3.151151 | n=50000 | 3.147183 |
| n=1000 | 3.203203 | n=50000 | 3.144383 |
| n=1000 | 3.191191 | n=50000 | 3.143663 |

***Вывод:*** наиболее близкий результат к точному значению числа π был получен при количестве случайных точек n=50000.

#### *Контрольные вопросы:*

1. Запишите синтаксис оператора цикла с параметром для случая, когда в цикле выполняется 1 оператор. Запишите синтаксис оператора цикла с параметром для случая, когда в цикле выполняется 3 оператора.
2. Задан цикл с параметром i, пробегающим значения от 0 до 10. Чему будет равен i при выходе из цикла?
3. Приведите пример объявления переменной, имеющей интерваль- ный тип данных.
4. Зачем нужны численные методы решения задач? В чем заключает- ся суть метода трапеций?
5. Запишите формулу трапеций. Как найти погрешность вычисления интеграла методом трапеций?
6. Что такое «предельная погрешность формулы трапеций»? Как ее найти?
7. Что такое «генератор случайного числа»? С помощью какой ко- манды можно сгенерировать число в интервале от 0 до 10?

## п. 4. Лабораторная работа № 4. Программирование алгоритмов итерационной циклической структуры

***Цель работы:*** формирование совокупности знаний об операторах программ с использованием циклов с предусловием и с постусловием; овладение практическими навыками разработки, программирования вычислительного процесса итерационной циклической структуры,

умения объяснять синтаксис и назначение операторов циклов с предусловием и с постусловием; получение дальнейших навыков по отладке и тестированию программы; умений представлять полученные результаты своей деятельности.

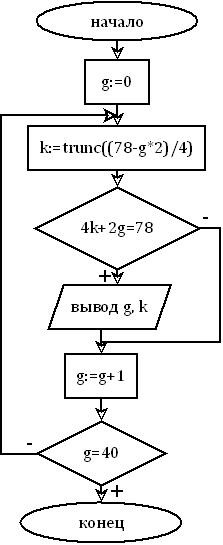
#### *Вопросы для подготовки к лабораторной работе:*

1. Что такое цикл? Какая программа называется циклической?
2. Запишите синтаксис оператора цикла с предусловием, с постусловием.
3. Что такое «тело цикла»?
4. Что требуется сделать для того, чтобы в теле цикла выполнялись несколько операторов?
5. Приведите примеры задач, для решения которых используется структура цикла с предусловием, с постусловием.
6. Что такое «итерация»?

#### *Ход работы:*

***Постановка задачи 4\_1.*** *У гусей и кроликов вместе 78 лап. Сколько могло быть кроликов и гусей (указать все сочетания, которые возможны)? Разра- ботать алгоритм в виде блок-схемы для решения задачи. Составить табли- цу трассировки.*

#### *Решение:*



***Математическая модель:*** пусть g — количе- ство гусей (2 лапы), k — количество кроликов (4 лапы). Тогда должно выполняться равенство 2g+4k=78. Минимально возможное количество гу- сей 0, а максимально возможное — 39. Формула для

нахождения количества кроликов: *k* = 78−2 *g*

4

(значе-

ние дроби должно быть целым числом).

***Программа:* program** lab4\_1; **var** k,g:integer; **begin**

g:=0;

**repeat**

k:=trunc((78-g\*2)/4);

**if** (k\*4+g\*2=78) **then**

writeln (g, ' гусей и ', k, ' кроликов'); g:=g+1;

**until** g=40;

**end**.

#### *Контрольный счет:*

Рисунок 4.1: Блок-схема к заданию 4\_1

1 гусей и 19 кроликов

3 гусей и 18 кроликов

5 гусей и 17 кроликов

7 гусей и 16 кроликов

9 гусей и 15 кроликов

11 гусей и 14 кроликов

13 гусей и 13 кроликов

15 гусей и 12 кроликов

17 гусей и 11 кроликов

19 гусей и 10 кроликов

21 гусей и 9 кроликов

23 гусей и 8 кроликов

25 гусей и 7 кроликов

27 гусей и 6 кроликов

29 гусей и 5 кроликов

31 гусей и 4 кроликов

33 гусей и 3 кроликов

35 гусей и 2 кроликов

37 гусей и 1 кроликов

39 гусей и 0 кроликов

#### *Таблица трассировки:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Оператор** | **Значения переменных** | | **Действия в цикле** | |
| **Условие**  (k\*4+g\*2=78) | **Вывод** |
| **g** | **k** |  |  |
| **begin** | пусто | пусто |
| g:=0; | 0 | пусто |
| **repeat** | Начало цикла с постусловием: выполнять до тех пор, пока условие g=40 ложно | |
| k:=trunc((78-  g\*2)/4); | g=0 g=1 g=2 g=3 g=4 g=5 g=6 g=7 g=8 | k=19 k=19 k=18 k=18 17  17  16  16  15 | ложно истинно ложно истинно ложно истинно ложно истинно ложно | отсутствует  1 гусей и 19 кроли- ков  отсутствует  3 гусей и 18 кроли- ков  отсутствует  5 гусей и 17 кроли- ков |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | g=9 | 15 | истинно | отсутствует |
| …............... | …...... | …......... | 7 гусей и 16 кроли- |
| g=38 | k=0 | ложно | ков |
| g=39 | k=0 | истинно | отсутствует |
|  |  |  | 9 гусей и 15 кроли- |
|  |  |  | ков |
|  |  |  | ….............. |
|  |  |  | отсутствует |
|  |  |  | 39 гусей и 0 кроли- |
|  |  |  | ков |
| **until** g=40; | g=40 — истинно — выход из цикла | | | |
| **end**. | Выход из программы | | | |

***Учебное задание:*** модифицировать программу так, чтобы количе- ство лап гусей и кроликов задавалось в диалоговом режиме. В том случае, если введенное число не соответствует целому количеству гусей и кроликов, должно выводиться об этом сообщение.

***Постановка задачи 4\_2:*** *с клавиатуры вводится целое число от 1 до 10000. Найти сумму цифр числа: если она равна 10, то вывести сообщение «Число удовлетворяет условию задачи», иначе вывести сообщение «Число не удовле- творяет условию задачи».*

#### *Решение:*

***Программа:***

**program** lab4\_2;

**var** n, s, x: integer; c: char;

### begin

n := 0; s := 0; x := 0;

writeln('Введите целое число от 1 до 10000'); read(c);

**while** c <> chr(13) **do begin**

n := (ord(c) - ord('0')); s := s + n; x := x \* 10 + n; read(c);

**end**;

**if** s = 10 **then**

writeln('Число ', x, ' удовлетворяет условию задачи')

### else

writeln('Число:', x, ' не удовлетворяет условию задачи');

**end**.

***Учебно-исследовательское задание:*** попробуйте ввести с кла- виатуры не число, а любую последовательность символов: убеди- тесь в том, что программа работает неверно. Предусмотрите оши- бочный ввод данных с клавиатуры, например, не целого числа или последовательности символов. С этой целью измените следу- ющий блок программы:

n := (ord(c) - ord('0'));

**if** (n >= 0) **and** (n <= 9) **then begin**

s := s + n; x := x \* 10 + n;

### end else begin

writeln('Было введено не число'); **exit**; **end**;

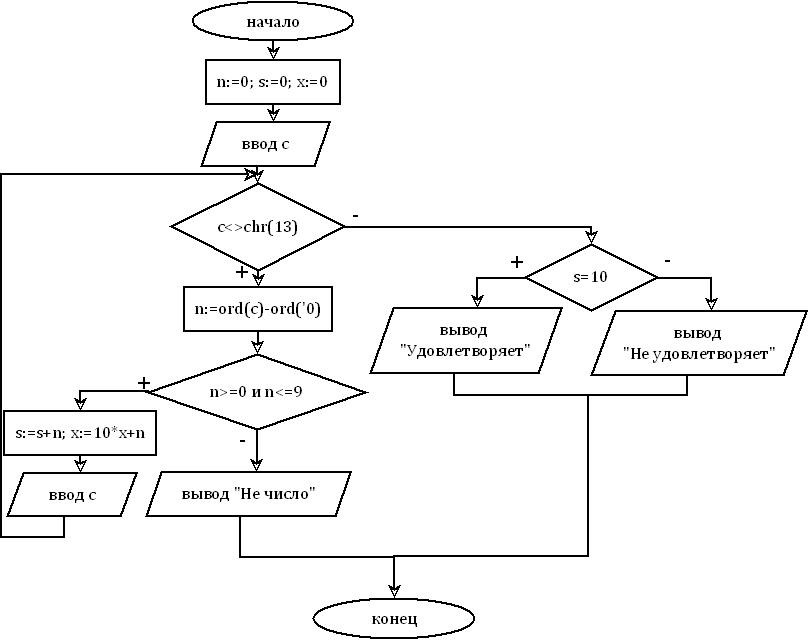


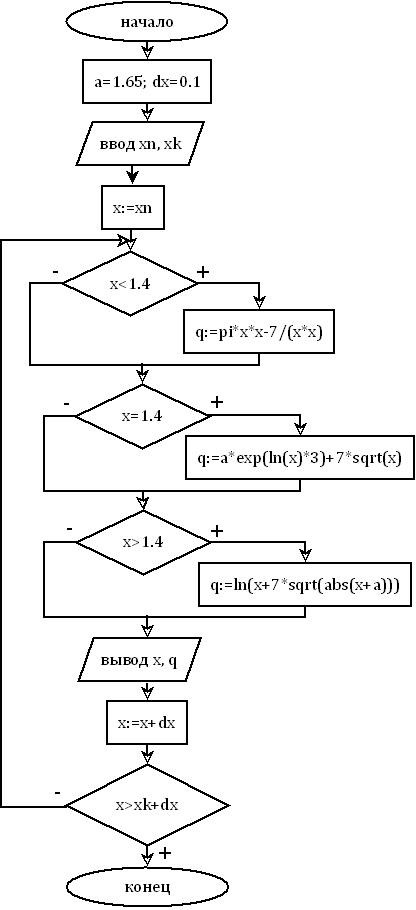
Рисунок 4.2: Блок-схема к заданию 4\_2

***Контрольный счет:*** Введите целое число от 1 до 10000 145kop

Было введено не число

Введите целое число от 1 до 10000 145

Число 145 удовлетворяет условию задачи Введите целое число от 1 до 10000



1256

Число:1256 не удовлетворяет условию задачи

***Постановка задачи 4\_3:*** *прота-*

*булировать*

*Q*={

π *x*2−7 / *x*2 *, ax*3+7 √ *x ,*

*функцию*

*x*<1,4

*при из-*

ln( *x*+7 √∣*x*+*a*∣) *,*

*x*=1,4

*x*>1,4

*менении аргумента в диапазоне*

*x* ∈[ 0,7 *;* 2]

*Δ x*=0,1 *.*

*значения*

*и c заданным шагом*

*Организовать вывод*

*аргумента*

*и*

*вы-*

*численного значения функции в*

*виде таблицы.*

***Решение:* program** lab4\_3; **const**

a=1.65; dx=0.1;

### var

q, x, xn, xk:real;

### begin

Write('Введите начальное и конечное значения x: '); read(xn, xk);

x:=xn;

writeln(' Таблица функции Q(X)'); writeln('|-------|-------------|');

### repeat

**if** x<1.4 **then** q:=pi\*x\*x-7/(x\*x); **if** x=1.4 **then**

q:=a\*exp(ln(x)\*3)+7\*sqrt(x);

**if** x>1.4 **then**

Рисунок 4.3: Блок-схема к заданию 4\_3

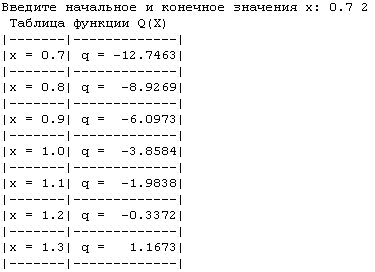
q:=ln(x+7\*sqrt(abs(x+a))); writeln('|x = ', x:3:1,'| q = ', q:8:4,'|'); writeln('|-------|-------------|');

x:=x+dx

**until** x>xk+dx;

**end**.

#### *Контрольный счет:*



***Учебное задание:*** модифицировать программу так, чтобы при зна- чении аргумента х=0 выводилось сообщение о том, что функция не определена. Измените шаг табулирования функции на 0,05.

***Постановка задачи 4\_4:*** *разработать программу, которая выводит та- блицу перевода температуры из градусов по шкале Цельсия (С) в градусы по шкале Фаренгейта (F) для значений от* 150 *С до* 310 *С с шагом* 20 *С.*

#### *Решение:*

***Математическая модель:*** для перевода температуры из шкалы Цельсия в шкалу Фаренгейта надо умножить исходное число на 9/ 5 и приба-

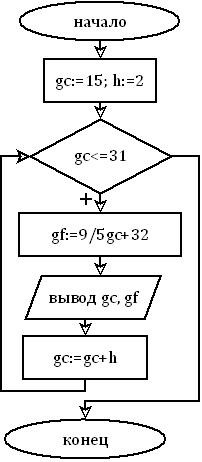
вить 32 : *gf* = *gc*∗9 .

5 +32

Если бы шаг изменения градусов был равен +1 или -1, то в этом случае следовало бы воспользоваться циклом с параметром, но в нашем случае шаг равен 2, поэтому воспользуемся циклом с предусловием.

***Программа:* program** lab4\_4; **var** gc, h:integer; gf:real;

### begin



gc:=15; h:=2;

writeln('Градусы Цельсия Градусы Фаренгейта');

**while** gc<=31 **do begin** gf:=gc\*9/5+32;

writeln(gc:10, ' ', gf:15); gc:=gc+h;

**end**; **end**.

#### *Контрольный счет:*

|  |  |
| --- | --- |
| Градусы Цельсия | Градусы Фаренгейта |
| 15 | 59 |
| 17 | 62.6 |
| 19 | 66.2 |
| 21 | 69.8 |
| 23 | 73.4 |
| 25 | 77 |
| 27 | 80.6 |
| 29 | 84.2 |
| 31 | 87.8 Рисунок 4.4: Блок-схема  к заданию 4\_4 |

***Постановка задачи 4\_5:*** *методом итераций вычислить корень нелинейно- го уравнения* 1  *x*  sin *x*  ln(1  *x*)  0 *, расположенный на отрезке* [ 0 *;* 1,5] *, c аб- солютной погрешностью ε=0.00001. Определить также число итераций, необходимое для нахождения корня.*

#### *Решение:*

***Математическая модель:*** при решении нелинейного уравнения ме- тодом итераций перепишем уравнение в виде *x*= *f* ( *x*) . Как видим, к такому виду уравнение можно привести несколькими способами. Мы должны полу- чить уравнение вида *x*= *f* ( *x*) , чтобы выполнялось условие сходимости мето- да итераций ∣ *f '* ( *x*)∣<1 .

Перепишем наше уравнение в виде: *x* :=1+sin *x*−ln(1+ *x*) . Проверим условие сходимости. Для этого вычислим производную функции *f* ( *x* ):=1+sin *x*−ln( 1+*x* ) : *f '* ( *x*):=cos *x*− 1

1 *x* . Выполним оценку модуля произ-

+

водной: функция cos *x* принимает значения от -1 до 1, то есть ∣cos *x*∣⩽1 ; функ-

ция 1

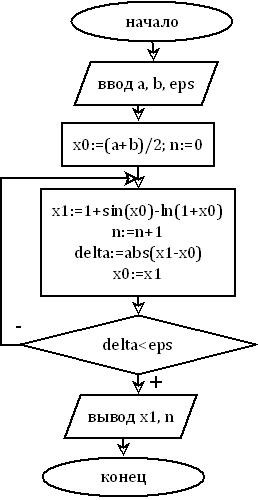
1+*x*

- убывающая, на отрезке [ 0 *;* 1,5] наибольшее значение *y*=1 прини-

1+*x*∣<1 для любого *x* ∈[ 0 *;* 1,5] .

За начальное значение аргумента *x*0 возьмем середину отрезка [ 0 *;* 1,5] . Первое приближение решения *x*1 находится из выражения *x*1= *f* ( *x*0) , второе

— из выражения *x*2= *f* ( *x*1) и т.д. В общем случае



*i*+1 приближение находится по формуле *xi*+1= *f* ( *xi*) . Указанную процедуру повторяем пока условие ∣ *f* ( *xi* )∣<*ε* ложно.

#### *Программа:*

**program** lab4\_5;

**var** a,b,x1,x0,delta,eps:real; n:integer;

### begin

write('Введите a, b, eps: '); read(a,b,eps);

x0:=(a+b)/2; n:=0;

### repeat

x1:=1+sin(x0)-ln(1+x0); n:=n+1; delta:=abs(x1-x0); x0:=x1;

**until** delta<eps; writeln('Корень =',x1:9:4); write('Число итераций =',n:5); **end**.

#### *Контрольный счет:*

Введите a,b eps: 0 1.5 0.00001

Корень = 1.1474 Число итераций = 5

Рисунок 4.5: Блок-схема к заданию 4\_5

***Учебное задание:*** модифицировать программу так, чтобы мето- дом итераций вычислялся корень нелинейного уравнения sin *x*+ *x*=0 на отрезке [−π / 2, π/ 2] .

***Постановка задачи 4\_6:*** *вычислить значение суммы бесконечного ряда*

*s*= 1 − 1 + 1 −...+(−1)*n* 1 +... *c заданной точностью ε=0.0001. Выве-*

*x*

3 *x*3 5 *x*5

( 2 *n*+1) *x*2 *n*+1

*сти значение суммы и число членов ряда, вошедших в сумму. Выполнить за-*

*дание 2 способами:*

*1)*

*2)*

*без использования рекуррентной формулы;*

*с использованием рекуррентной формулы.*

*Сравнить полученное значение суммы ряда со значением*

*s=0.58800260354757. Сравнить время выполнения каждой из программ.*

#### *Решение:*

Для вычисления времени работы программы воспользуемся встроенной функцией нахождения количества миллисекунд с момента начала работы программы. В среде PascalABC.NET эта команда называет- ся Milliseconds и расположена она в модуле Utils.

*n* 1

1. Воспользуемся накопительной формулой *s*=*s*+(−1)

( 2 *n*+1) *x*2 *n* +1

вну-

три цикла, *n*=0, 1, 2,... . В этом случае сложность заключается в том, как запи- сать (−1)*n* в языке Паскаль, поскольку встроенная операция возведения в степень отсутствует[13](#_bookmark26). Можно воспользоваться двумя встроенными функ- циями:

* функция odd(n) - возвращает логическое значение True, если n нечетно
* функция ord(n) - возвращает порядковый номер логического значения n (если n – четно, то ord(odd(n))=0, если n — нечетно, то ord(odd(n))=1).

Тогда будем вычислять (−1)*n* по формуле: 1-2\*ord(odd(n)). Действи- тельно, если n — четно, то получим 1-2\*0=1, а если нечетно — получим 1- 2\*1=-1.

***Программа:* program** lab4\_6; **uses** utils;

**const** s=0.58800260354757; x=1.5; eps=0.0001;

**var** a, s0, s1:real; n:integer;

### begin

a:=1/x;

s0:=0; s1:=1/x; n:=0;

**while** abs(s1-s0)>=eps **do begin**

n:=n+1;

a:=(1-2\*ord(odd(n)))/((2\*n+1)\*exp((2\*n+1)\*ln(x))); s0:=s1; s1:=s0+a;

**end**;

writeln('Сумма ряда =',s0:8:14); writeln('Число членов ряда =',n:4);

writeln('Результаты расходятся на ', abs(s-s0):8:14); Writeln('Время работы программы: ', milliseconds/1000,' секунд'); **end**.

***Контрольный счет:*** Сумма ряда =0.58795984618067 Число членов ряда = 8

1. В среде PascalABC.Net есть встроенная подпрограмма с именем power, которая позволяет возводить число в степень

Результаты расходятся на 0.00004275736690 Время работы программы: 0.016 секунд

В среде Geany для получения времени выполнения программы сле- дует воспользоваться командой Gettime.

uses dos;

var h, m, s, ms, h1, m1, s1, ms1 : word;

….... BEGIN

gettime(h,m,s,ms); Операторы; gettime(h1,m1,s1,ms1);

writeln('Time Pascal: ',h1-h, ':', m1-m, ':', s1-s,':', ms1-ms); END.

1. Выведем рекуррентную формулу для нахождения последующего

*an*+1

члена ряда через предыдущий. Для этого составим отношение и

*a*

*n*

упростим его:

*an* +1

(−1)*n*+1

(−1)*n*

( 2 *n*+1) *x*2 *n*+1

2 *n*+1

*k* = *a* ( 2( *n*+1)+1) *x*2( *n*+1)+1 (2 *n*+1) *x*2 *n*+1 ( 2( *n*+1)+1) *x*2( *n*+1)+1 (2 *n*+3) *x*2 .

= ÷ =− =−

*n*

Тогда получим рекуррентную формулу: *an*+1=*k*⋅*an* , или накопитель- ную формулу *a*=*a*⋅*k* .

#### *Программа:*

**program** lab4\_6;

**uses** utils;

**const** s=0.58800260354757; x=1.5; eps=0.0001;

**var** a,s0, s1:real; n:integer;

### begin

a:=1/x; s0:=0; s1:=1/x; n:=0;

**while** abs(s1-s0)>=eps **do begin**

a:=-a\*(2\*n+1)/((2\*n+3)\*x\*x); s0:=s1; s1:=s0+a;

n:=n+1;

**end**;

writeln('Сумма ряда =', s0:8:14); writeln('Число членов ряда =', n:4); writeln('Результаты расходятся на ',

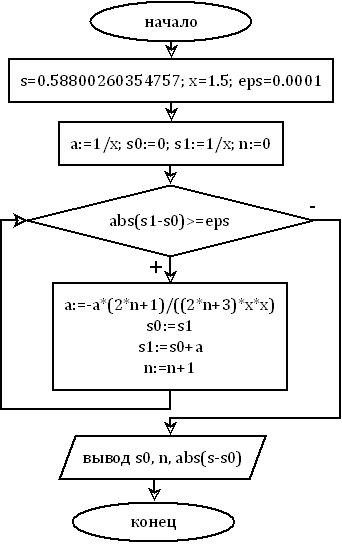


Рисунок 4.6: Блок-схема к заданию 4\_6

abs(s-s0):8:14);

writeln('Время работы программы: ', milliseconds/1000,' секунд');

**end**.

#### *Контрольный счет:*

Сумма ряда =0.58795984618067 Число членов ряда = 8

Результаты расходятся на 0.00004275736690 Время работы программы: 0.016 секунд

Таким образом, время работы программы с использованием рекуррентной формулы и без ее использования совпадает.

***Учебно-исследовательское задание:*** увеличьте точность вычис- лений и найдите то ее значение, при котором время работы про- грамм будет различным. Сделайте соответствующие выводы.

#### *Контрольные вопросы:*

1. Что такое «итерационный алгоритм»? Как найти число итераций?
2. Каков синтаксис оператора цикла с предусловием, с постусловием?
3. Что такое «условие выхода из цикла»?
4. Что такое «счетчик итераций»?
5. Каким должно быть условие продолжения цикла с предусловием (истинным или ложным)?
6. Каким должно быть условие продолжения цикла с постусловием (истинным или ложным)?
7. Запишите цикл с предусловием, цикл с постусловием на языке блок-схем.
8. Для чего предназначены встроенные функции ord, chr, odd, trunc в Паскале?
9. Какой оператор позволяет принудительно выйти из цикла (прервать выполнение цикла)?
10. Что значит «протабулировать функцию»?
11. Объясните суть метода итераций для нахождения корня нелинейного уравнения.
12. Что такое «рекуррентная формула»? Запишите прямую и рекуррентную формулы для нахождения факториала натурального числа N.
13. Какая команда в Паскале используется для вычисления времени работы программы?

## п. 5. Лабораторная работа № 5. Программирование алгоритмов со структурой вложенных циклов

***Цель работы:*** формирование совокупности знаний об особенностях структуры программ с использованием вложенных циклов, умений анализировать исходные данные задачи; овладение практическими навыками разработки, программирования вычислительного процесса циклической структуры; умений оформлять результаты своей деятельности в современном офисном пакете и аргументировать их, умений объяснять синтаксис операторов вложенных циклов и последовательность их выполнения; получение дальнейших навыков по отладке и тестированию программы; развитие коммуникативных навыков взаимодействия в коллективе.

#### *Вопросы для подготовки к лабораторной работе:*

1. Какие различают типы циклов? Запишите синтаксис каждого из цикла на языке Паскаль.
2. Задан цикл с параметром: for а:=1 to 10 do x:=x+a. Запишите на языке Паскаль этот циклический процесс с использованием цикла с предусловием, с постусловием.
3. Верно ли, что до начала итерационного цикла должны быть сделаны начальные установки переменных, управляющих условием цикла?
4. Верно ли, что в теле цикла должны присутствовать операторы, изменяющие переменные условия так, чтобы цикл через некоторое число итераций завершился?
5. В каком случае в теле цикла необходимо использовать составной оператор?
6. Какая конструкция называется вложенным циклом?
7. Что называется «внутренним циклом», «внешним циклом»? Что такое

«уровень вложенности»?

#### *Ход работы:*

***Постановка задачи 5\_1:*** *выполнить пояснения к программе (выделить структурные единицы программы) и сформулировать задачу, для решения которой программа написана. Набрать программу, проверить ее работо- способность. Сохранить в своей папке, создать exe-файл. Показать препо- давателю.*

**program** lab5\_1;

**var**

h, m, s: byte;

**begin**

**for** h := 0 **to** 23 **do for** m := 0 **to** 59 **do**

**for** s := 0 **to** 59 **do**

Writeln(h, ':', m, ':', s);

**end**.

#### *Решение:*

Пояснения к программе:

|  |  |
| --- | --- |
| **program** lab5\_1; | **Заголовок программы** lab5\_1 |
| **var** | **Раздел описания переменных** |
| h, m, s: byte; | Параметр цикла: имя h, тип – целое число Параметр цикла: имя m, тип – целое число Параметр цикла: имя s, тип — целое число |
| **begin** | **Раздел операторов (основной блок)** |
| **for** h := 0 **to** 23 **do** | Цикл с параметром: h пробегает значения от 0 до 23 (часы) |
| **for** m := 0 **to** 59 **do** | Цикл с параметром: m пробегает значения от 0 до 59 (минуты) |
| **for** s := 0 **to** 59 **do** | Цикл с параметром: s пробегает значения от 0 до 59 (секунды) |
| Writeln(h, ':', m, ':', s); | Вывод текущего значения переменных h, m, s |
| **end.** | **Завершение раздела операторов (текста програм- мы)** |

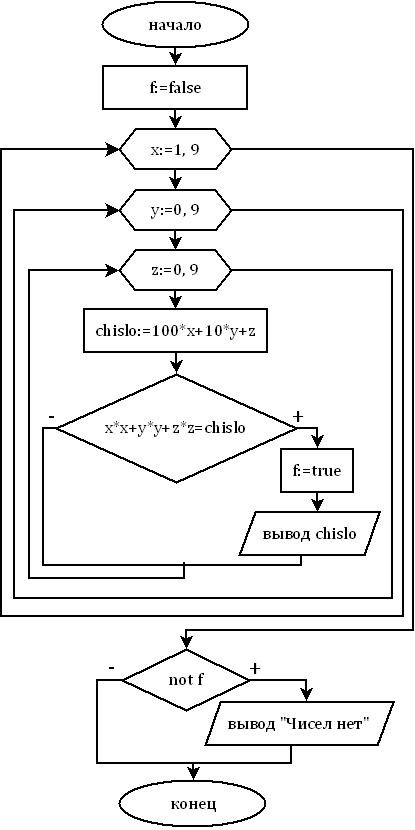
Цикл по переменной m (минуты) вложен в цикл по переменной h (часы), цикл по переменной s (секунды) вложен в цикл по переменной m (минуты). Таким образом, значение минут меняется только после того, как секунды «пробегут» все последовательные значения от 0 до 59. Часы изменяются только после того, как минуты «пробегут» все последова- тельные значения от 0 до 59.

Формулировка задачи: разработать программу, имитирующую ра- боту электронных часов, начиная с момента времени 0 часов, 0 минут, 0 секунд.

***Постановка задачи 5\_2:*** *найти все трехзначные числа, сумма квадра- тов цифр которых равна самому числу. Если таких чисел нет, то выве- сти об этом сообщение.*

#### *Решение:*

Введем обозначения: пусть трехзначное число состоит из цифр xyz. Тогда требуется найти все такие числа, для которых верно равенство: *x*2+ *y*2+*z*2 =100 *x*+20 *y*+ *z* . Первая



цифра числа может принимать значения в диапазон от 1 до 9, вторая и третья цифры — в диапазоне от 0 до 9. Для перебора всех возможных комбинаций цифр, а следовательно, всех трех- значных чисел воспользуемся тремя вложенными циклами с па- раметром.

***Программа:* program** lab5\_2; **var** x:1..9; y, z:0..9;

chislo:100..999; f:boolean;

### begin

f:=false;

**for** x:=1 **to** 9 **do for** y:=0 **to** 9 **do for** z:=0 **to** 9 **do begin**

chislo:=100\*x+10\*y+z;

**if** x\*x+y\*y+z\*z=chislo **then begin**

f:=true; writeln(chislo);

**end**; **end**;

**if not** f **then** writeln('Таких чисел нет');

**end**.

#### *Контрольный счет:*

Чисел нет

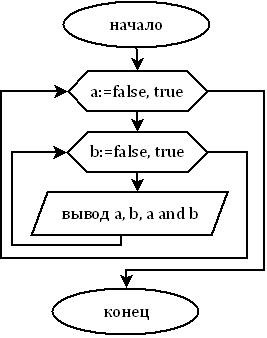
Рисунок 5.1: Блок-схема к заданию 5\_2

***Учебное задание:*** изменить программу так, чтобы она осуществляла поиск двухзначных чисел, равных разности квадратов (по модулю) своих цифр.

***Постановка задачи 5\_3:*** *написать программу, которая выводит табли- цу истинности для логической операции «and » для двух логических пере- менных a и b.*

#### *Решение:*

***Программа:* program** lab5\_3; **var** a, b :boolean; **begin**



writeln('| a | b | |a /\ b|'); writeln('--------------------------');

**for** a:=false **to** true **do for** b:=false **to** true **do begin**

writeln('|',a:6,'|', b:6,'| = |', a **and** b:6, '|'); writeln('--------------------------');

**end**; **end**.

#### *Контрольный счет:*

| a | b | |a /\ b|

------------------------------

| False| False| = | False|

------------------------------

| False| True| = | False|

------------------------------

| True| False| = | False|

------------------------------

| True| True| = | True|

------------------------------

Рисунок 5.2: Блок-схема к заданию 5\_3



***Постановка задачи 5\_4:*** *вычислить значение интеграла*

*z*  cos *xdx . На-*

2

0

*чальное число разбиений отрезка интегрирования n*=50 *, численный ме- тод решения — метод средних прямоугольников, точность вычислений e=0.0001. Включить в программу вычисление точного значения интегра- ла. На печать вывести приближенные значения интеграла, точное зна- чение интеграла, относительную погрешность вычисления в процентах, предельную погрешность формулы средних прямоугольников.*

***Задание:*** доработать программу, чтобы она выводила таблицу ис- тинности для логической операции «and» для трех логических переменных a, b и с.

#### *Решение: Математическая модель:*

Рассмотрим суть численного ме- тода средних прямоугольников для вы- числения значения определенного ин- теграла. Для начала разобъем криволи- нейную трапецию (она ограничена сле- ва и справа прямыми *x*=*a* и *x*=*b* , снизу осью Ox, сверху графиком функции *y*= *f* ( *x* ) ) на *n* равных прямоугольни- ков. Говорят: площадь под графиком функции аппроксимируется прямоуголь- никами (рис. [5.3](#_bookmark27)). Таким, образом, про- межуток интегрирования (*a ,b*) разде-

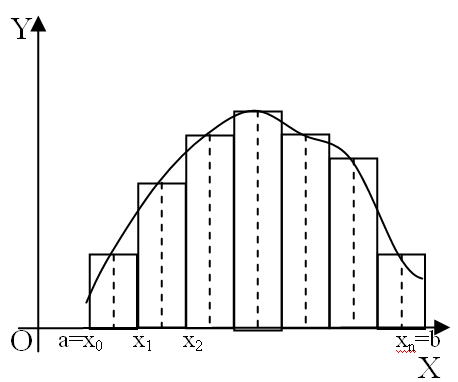


Рисунок 5.3: Метод средних прямоугольников

*b*−*a*

лился точками *x*1, *x*2, ... *xn*−1 на *n* равных частей; длина каждой *h*=

единообразия полагаем *a*= *x*0 *, b*= *xn* . Тогда имеем [14](#_bookmark29):

*n* . Для

*b n*−1

∫ *f* ( *x* ) *dx*≈*h*∑ *y* ( *xi*+*h*/ 2) (2)

*a i*=0

Это – формула средних прямоугольников. Она дает общую площадь прямоугольников, показанных на рис. [5.3](#_bookmark27).

Предельная погрешность формулы [(2)](#_bookmark28) составляет (*b*−*a* ) *M*

3

24 *n*2 2

, где *M* 2 —

наибольшее значение ∣ *f ' '* ( *x*)∣ в промежутке (*a ,b*) .

***Программа:* program** lab5\_4; **const**

a = 0; b = pi / 2; e = 0.0001;

### var

x, z, z0, eq, zt, dz, h: real; n, i: integer;

### begin

n := 50; z0 := 0;

### repeat

h := (b - a) / n; x := a + h / 2; z := cos(x);

**for** i := 1 **to** n - 1 **do begin**

x := x + h;

z := z + cos(x);

**end**;

1. Выгодский М.А. Справочник по высшей математике. - М., 2000. - 864с. - С.479

z := z \* h; eq := abs(z0 - z); z0 := z; n := n \* 2;

**until** eq < e;

zt := (sin(b) - sin(a)); dz := abs(zt - z) \* 100;

writeln('Приближенное значение z=', z:9:7); writeln('Точное значение zt=', zt:9:7);

writeln('Относительная погрешность', dz:9:7); writeln('Предельная погрешность:', (b - a) \* (b - a) \* (b - a) / (24 \* n \* n) \* 1:9:6);

**end**.

#### *Контрольный счет:*

Приближенное значение z=1.0000103

Точное значение zt=1.0000000 Относительная погрешность 0.0010281

Предельная погрешность: 0.000004

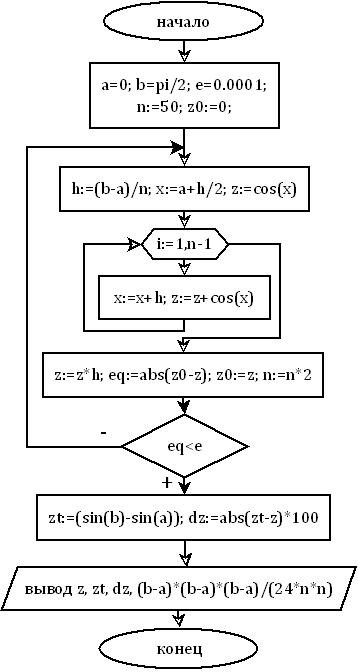


Рисунок 5.4: Блок-схема к заданию 5\_4

***Учебно-исследовательское задание:*** Добавить в программу вы- числение числа итераций, затраченных на вычисление значения интеграла с заданной точностью. Модифицировать программу так, чтобы она вычисляла значение интеграла методом левых прямоугольников. Сравнить число итераций, полученных в обоих методах. Сделать вывод.

***Постановка задачи 5\_5:*** *методом поразрядного приближения вычис- лить экстремум функции (максимум) y*  *x*1/ 3 (1 *x*)2 / 3 *на отрезке* [ 0,1 *;* 0,6] *c заданной точностью e=0,00001. Вывести вычисленное конечное значение экстремума и значение аргумента, при котором оно достигается. В ка- честве грубого значения шага взять h=0,3.*

#### *Решение:*

Убедимся в том, что данная функция имеет точку максимума на рассматриваемом отрез- ке. Для этого достаточно вспомнить алгоритм на- хождения экстремума функции через производ- ную функции и выпол- нить соответствующие вычисления. Мы же воспользуемся системой компьютерной математи- ки Maxima: загрузим си- стему, выберем пункт

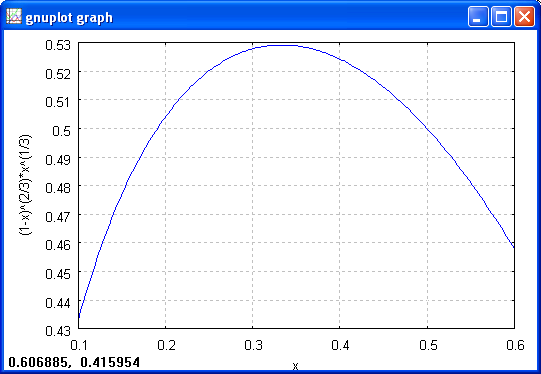


Рисунок 5.5: Вид графика функции на рассматриваемом отрезке

меню «Графики» → “Plot2d”, в диалоговом окне зададим функцию с помо- щью команды x^(1/3)\*(1-x)^(2/3), пределы изменения по оси абсцисс и нажмем на кнопку Ok (см. рис. [5.5](#_bookmark30)).

Запись команды на языке системы:

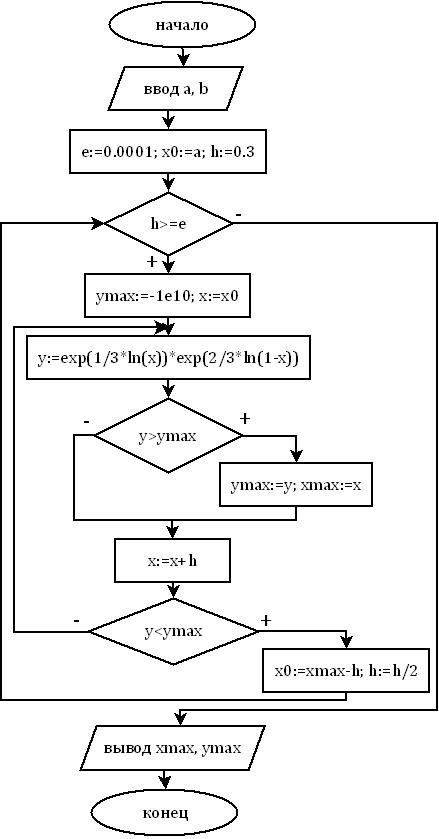
plot2d([x^(1/3)\*(1-x)^(2/3)], [x,0.1,0.6], [plot\_format, gnuplot], [gnuplot\_preamble, "set grid;"])$

Как видно из построения, на заданном отрезке [ 0,1 *;* 0,6] у функции действительно имеется точка максимума *x*≈0.33 .

***Математическая модель:*** рассмотрим суть метода поразрядного приближения для вычисления значения экстремума на заданном отрез- ке.

* 1. Задаем начальный шаг поиска *h* , точность *e* определения результата поиска (для переменной *x* ). Задаем начальную точку, равную левому концу отрезка.
  2. Пока значение шага перебора точек на отрезке больше или равно заявленной точности вычислений, берем в качестве максимума функции *ymax* очень большое отрицательное число и идем в цикл для вычисления значений функции *y* ( *x* ) с шагом *h* и сравнения найденного значения с *ymax* . Запоминаем наибольшее значение под именем *ymax* . Выполняем действия в цикле до тех пор, пока не перейдем точку максимума (условие *y*< *ymax* ).
  3. Если условие выполнилось, то отходим на один шаг *h* назад, уменьшаем шаг в два раза и повторяем пункт 2.
  4. Выводим на печать полученные значения *ymax* и *xmax* .

#### *Программа:*



**program** lab5\_5; **const** e=0.0001; **var**

a, b, x0, xmax, ymax, y, x, h: real;

### begin

**writeln**('Введите концы отрезка');

readln(a,b); x0:=a; h:=0.3;

**while** h>=e **do begin**

ymax:=-1e10; x:=x0;

### repeat

y:=exp(1/3\*ln(x))\*exp(2/3\*ln (1-x));

**if** y>ymax **then begin** ymax:=y; xmax:=x; **end**;

x:=x+h;

**until** y<ymax; x0:=xmax-h; h:=h/2; **end**;

writeln('xmax=', xmax:8:6, '

ymax =', ymax:8:6);

***end****.*

***Контрольный счет:*** Введите концы отрезка 0.1 0.6

xmax= 0.333350 ymax = 0.529134

Рисунок 5.6: Блок-схема к заданию 5\_5

***Задание:*** модифицировать программу так, чтобы она вычисляла минимум функции *y*=( *x*−2)2−1 на отрезке [1 *;* 3] .

#### *Контрольные вопросы:*

1. Приведите примеры задач, алгоритм решения которых основан на вложенных циклах.
2. Что такое «итерация»? Какие циклы называют итерационными?
3. Должны ли операторы внутреннего цикла полностью располагаться в теле внешнего цикла?
4. Вычислите, сколько раз выполнится тело цикла for i:=1 to 10 do for j:=1 to 10 do s:=s+i\*j. Поясните, как работает этот фрагмент программы.
5. С какого цикла начинается выполнение программы: с внешнего или с внутреннего?
6. Нарисуйте блок-схему алгоритма с двумя вложенными друг в друга циклами с параметром.
7. Нарисуйте блок-схему алгоритма, в которой в цикл с параметром вложен цикл с предусловием.
8. Изложите суть метода поразрядного приближения для вычисления значения экстремума на заданном отрезке.

## п. 6. Лабораторная работа № 6. Обработка одномерных массивов

***Цель работы:*** формирование совокупности знаний об особенностях организации работы с массивами; овладение практическими навыками разработки, программирования вычислительного процесса с использованием одномерных массивов; умений анализировать исходные данные задачи; получение дальнейших навыков по отладке и тестированию программы; умений оформлять результаты своей деятельности в современном офисном пакете и аргументировать их, умений объяснять синтаксис операторов организации работы с массивами; развитие коммуникативных навыков взаимодействия в коллективе.

#### *Вопросы для подготовки к лабораторной работе:*

1. Что такое «массив», «элемент массива», «индекс элемента массива»?
2. Почему массив является структурированным типом данных?
3. Какого типа могут быть элементы массива?
4. Какого типа могут быть индексы элементов массива?
5. Какие простые типы данных относятся к порядковым?
6. Что такое «размер» и «размерность» массива? Назовите виды массивов.
7. Как объявить одномерный целочисленный массив А из 100 элементов?
8. Как определить количество ячеек памяти, отводимой под массив?
9. Какие существуют способы заполнения одномерного массива?

Запишите фрагмент кода программы на языке Паскаль для заполнения массива 50 числами с клавиатуры.

1. Как организовать перебор всех элементов одномерного массива? Запишите фрагмент кода программы на языке Паскаль для перебора всех элементов массива и подсчета их суммы.

#### *Ход работы:*

***Постановка задачи 6\_1:*** *выполнить пояснения к программе (выделить структурные единицы программы) и сформулировать задачу, для решения которой программа написана. Набрать программу, проверить ее работо- способность. Сохранить в своей папке, создать exe-файл.*

**program** lab6\_1;

**const** nn=100;

**var** a: **array**[1..nn] **of** real; i,n:integer; s:real;

**begin**

writeln('Введите количество элементов массива'); readln(n); writeln('Введите элементы массива');

**for** i:=1 **to** n **do** read(a[i]); s:=0;

**for** i:=1 **to** n **do**

**if** a[i]>0 **then** s:=s+a[i]; writeln('Сумма =', s); **end**.

***Решение:***

Пояснения к программе:

|  |  |
| --- | --- |
| **program** lab6\_1; | **Заголовок программы** lab6\_1 |
| **const** nn=100; | **Раздел объявления констант**  имя nn: значение 100 |
| **var** | **Раздел описания переменных** |
| a: **array**[1..nn] **of** real; i,n:integer;  s:real; | Массив a: размер nn элементов, тип - веще- ственный  Параметр цикла: имя i, тип – целое число Размер массива: имя n, тип – целое число Сумма положительных чисел: имя s, тип — ве- щественное число |
| **begin** | **Раздел операторов (основной блок)** |
| writeln('Введите количество элементов | Вывод на экран строки комментария 'Введите количество элементов массива' |

|  |  |
| --- | --- |
| массива'); |  |
| readln(n); | Считывание с клавиатуры целого числа и сохранение его под именем n |
| writeln('Введите элементы массива'); | Вывод на экран строки комментария 'Введите элементы массива' |
| **for** i:=1 **to** n **do** | Организация цикла с параметром по перемен- ной i, изменяющейся от 1 до n с шагом +1 |
| read(a[i]); | Считывание элемента массива а с индексом i |
| s:=0; | Обнуление значения переменной s |
| **for** i := 1 **to** n **do** | Организация цикла с параметром по перемен- ной i, изменяющейся от 1 до n с шагом +1 |
| **if** a[i]>0 **then** s:=s+a[i]; | Тело цикла: если элемент массива a с индексом I больше нуля, то увеличение суммы положи- тельных чисел на значение элемента a[i] |
| writeln('Сумма =', s); | Вывод на экран строки «Сумма =» и значения, хранимого под именем s |
| **end.** | **Завершение раздела операторов (текста** |
| **программы)** |

Формулировка задачи: найти сумму положительных элементов од- номерного массива A размера N.

#### *Контрольный счет:*

Введите количество элементов массива 5

Введите элементы массива

-8 7 5 0 -1

Сумма =12

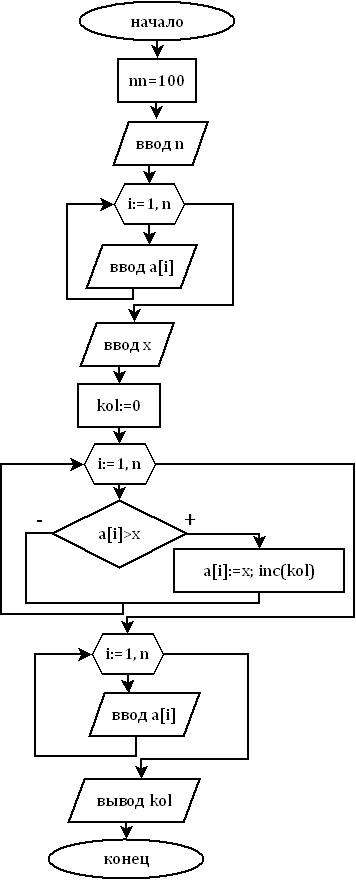
**Задание:** модифицируйте программу так, чтобы она выполняла вычисление произведения положительных и количество отрица- тельных элементов массива. Предусмотрите вывод элементов массива на экран в строку.

***Постановка задачи 6\_2:*** *дан массив действительных чисел А(n). Заменить все члены, большие заданного x, этим числом. Подсчитать количество за- мен.*

#### *Программа:*

**program lab6\_2;**

**const** nn=100;



**var** a: **array**[1..nn] **of** real; i,n, kol:integer;

x:real;

### begin

writeln('Введите количество элементов массива'); readln(n);

writeln('Введите элементы массива');

**for** i:=1 **to** n **do**

read(a[i]);

writeln('Введите значение x'); readln(x);

kol:=0;

**for** i:=1 **to** n **do if** a[i]>x **then begin**

a[i]:=x; inc(kol);

**end**;

writeln('Массив после замены');

**for** i:=1 **to** n **do**

write(a[i], ' '); writeln('Количество замен=', kol); **end**.

#### *Контрольный счет:*

Введите количество элементов мас- сива

8

Введите элементы массива 2 4 5 6 8 9 1 2

Введите значение x 4

Массив после замены 2 4 4 4 4 4 1 2

Количество замен=4

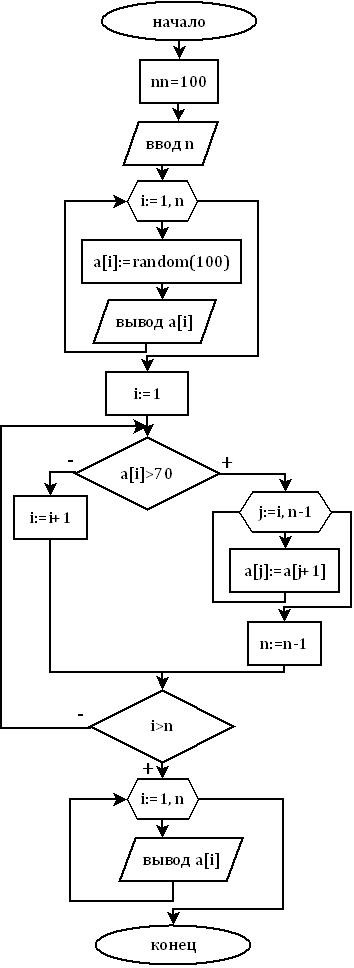
Рисунок 6.1: Блок-схема к задаче 6\_2

**Задание:** модифицируйте программу так, чтобы она выполняла подсчет количества элементов массива, лежащих в диапазоне от a до b.

***Постановка задачи 6\_3:*** *определить максимальный элемент массива В(50) и его порядковый номер. Предусмотреть вывод элементов массива на экран в строку.*

#### *Решение:*

Поскольку количество эле- ментов одномерного массива вели- ко, воспользуемся генератором слу- чайного числа для заполнения мас- сива элементами.



#### *Программа:*

**program** Lab6\_3;

**const** nn=50;

**var** n,i,imax:integer; bmax:real; b:**array**[1..nn] **of** integer;

### begin

write('Введите размер массива N<=50: ');

read(n); randomize;

**for** i:=1 **to** n **do** b[i]:=random(31); bmax:=b[1]; imax:=1;

**for** i:=2 **to** n **do**

**if** b[i]>bmax **then begin**

bmax:=b[i]; imax:=i;

**end**;

writeln('\*\*\*\*\*\*\*\*\*\*\*Rezultat\*\*\*\*\*\*\*\*\*

\*\*');

write('Elementy massiva:'); **for** i:=1 **to** n **do** write(b[i]:4);

writeln;

writeln('bmax = ',bmax:4); write('imax = ',imax);

**end**.

#### *Контрольный счет:*

Рисунок 6.2: Блок-схема к задаче 6\_4

Введите размер массива N<=50: 8

\*\*\*\*\*\*\*\*\*\*\*Rezultat\*\*\*\*\*\*\*\*\*\*\*

Elementy massiva: 26 5 29 25 2 28 9 21

bmax = 29

imax = 3

**Задание:** модифицируйте программу так, чтобы она находила минимальный элемент массива и его порядковый номер.

***Постановка задачи 6\_4:*** *имеется массив A(n), содержащий числа от 0 до 100 включительно. Требуется исключить из него все элементы, значения которых больше 70.*

#### *Решение:*

C помощью генератора случайного числа заполним массив А числами от 0 до 100 по формуле: A[i]:=random(100). Организуем просмотр всех эле- ментов массива и будем каждый из них сравнивать с числом 70: если a[i]>70, то сдвигаем элементы массива на одну позицию влево, значение n, обознача- ющее длину массива, уменьшаем на 1; если очередной элемент этому усло- вию не удовлетворяет, то переходим к просмотру следующего элемента (i:=i+1) и не уменьшаем размер массива.

#### *Программа:*

**program** lab6\_4; **const** nn = 100; **var**

a: **array**[1..nn] **of** real; i, j, n: integer;

### begin

readln(n);

**for** i := 1 **to** n **do begin**

a[i] := random(100);

write(a[i]:5:1);

**end**; i := 1;

writeln;

### repeat

**if** a[i] > 70 **then begin**

**for** j := i **to** n - 1 **do** a[j] := a[j + 1]; n := n - 1;

### end

**else** i := i + 1;

**until** i > n;

**for** i := 1 **to** n **do** write(a[i]:5:1);

**end**.

#### *Контрольный счет:*

10

32.0 15.0 2.0 76.0 84.0 57.0 79.0 5.0 65.0 56.0

32.0 15.0 2.0 57.0 5.0 65.0 56.0

**Задание:** модифицируйте программу так, чтобы она переписала все элементы, большие 70, в массив B. Исходный массив А оста- вить без изменений.

***Постановка задачи 6\_5:*** *проанализировать программу, определить, для решения какой задачи написана программа. Составить таблицу трасси- ровки.*

**program** massiv;

**const** n=30;

**var** a:**array**[1..n] **of** real; i, j:integer; t:real;

**begin**

**for** i:=1 **to** n **do** a[i]:=random(50); writeln('Исходный массив');

**for** i:=1 **to** n **do** write(a[i]:8:1); writeln;

**for** i:=1 **to** n-1 **do for** j:=i+1 **to** n **do begin**

**if** a[i]>a[j] **then begin**

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

**end**; **end**;

writeln('Получен массив'); **for** i:=1 **to** n **do** write(a[i]:8:1);

**end**.

#### *Решение:*

Пусть массив А состоит из следующих чисел: 45, 17, 5, 14, 39, 6, 37, 6,

1. Составим таблицу трассировки для следующего основного блока про- граммы:

**for** i:=1 **to** n-1 **do for** j:=i+1 **to** n **do begin**

**if** a[i]>a[j] **then begin**

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

**end**; **end**;

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **A[1]** | **A[2]** | **A[3]** | **A[4]** | **A[5]** | **A[6]** | **A[7]** | **A[8]** | **A[9]** |
| i=0, j=0 | **45** | **17** | 5 | 14 | 39 | 6 | 37 | 6 | 13 |
| **i=1,** j=2, a[1]>a[2] - верно | **17** | **45** | 5 | 14 | 39 | 6 | 37 | 6 | 13 |
| j=3, a[1]>a[3] - неверно | **5** | 45 | **17** | 14 | 39 | 6 | 37 | 6 | 13 |
| j=4, a[1]>a[4] - неверно | **5** | 45 | 17 | **14** | 39 | 6 | 37 | 6 | 13 |
| j=5, a[1]>a[5] - неверно | **5** | 45 | 17 | 14 | **39** | 6 | 37 | 6 | 13 |
| j=6, a[1]>a[6] - неверно | **5** | 45 | 17 | 14 | 39 | **6** | 37 | 6 | 13 |
| j=7, a[1]>a[7] - неверно | **5** | 45 | 17 | 14 | 39 | 6 | **37** | 6 | 13 |
| j=8, a[1]>a[8] - неверно | **5** | 45 | 17 | 14 | 39 | 6 | 37 | **6** | 13 |
| j=9, a[1]>a[9] - неверно | **5** | 45 | 17 | 14 | 39 | 6 | 37 | 6 | 13 |
| **i=2,** j=3, a[2]>a[3] - верно | 5 | **17** | 45 | 14 | 39 | 6 | 37 | 6 | 13 |
| j=4, a[2]>a[4] - верно | 5 | **14** | **45** | **17** | 39 | 6 | 37 | 6 | 13 |
| j=5, a[2]>a[5] - неверно | 5 | **14** | 45 | 17 | **39** | 6 | 37 | 6 | 13 |
| j=6, a[2]>a[6] - верно | 5 | **6** | 45 | 17 | 39 | 14 | 37 | 6 | 13 |
| j=7, a[2]>a[7] - неверно | 5 | **6** | 45 | 17 | 39 | 14 | **37** | 6 | 13 |
| j=8, a[2]>a[8] - неверно | 5 | **6** | 45 | 17 | 39 | 14 | 37 | **6** | 13 |
| j=9, a[2]>a[9] - неверно | 5 | **6** | 45 | 17 | 39 | 14 | 37 | 6 | **13** |
| **i=3,** j=4, a[3]>a[4] - верно | 5 | 6 | **17** | **45** | 39 | 14 | 37 | 6 | 13 |
| j=5, a[3]>a[5] - неверно | 5 | 6 | **17** | 45 | **39** | 14 | 37 | 6 | 13 |
| j=6, a[3]>a[6] - верно | 5 | 6 | **14** | 45 | 39 | **17** | 37 | 6 | 13 |
| j=7, a[3]>a[7] - неверно | 5 | 6 | **14** | 45 | 39 | 17 | **37** | 6 | 13 |
| j=8, a[3]>a[8] - верно | 5 | 6 | **6** | 45 | 39 | 17 | 37 | **14** | 13 |
| j=9, a[3]>a[9] - неверно | 5 | 6 | **6** | 45 | 39 | 17 | 37 | 14 | **13** |
| **i=4,** j=5, a[4]>a[5] - верно | 5 | 6 | 6 | **39** | **45** | 17 | 37 | 14 | 13 |
| j=6, a[4]>a[6] - верно | 5 | 6 | 6 | **17** | 45 | **39** | 37 | 14 | 13 |
| j=7, a[4]>a[7] - неверно | 5 | 6 | 6 | **17** | 45 | 39 | **37** | 14 | 13 |
| j=8, a[4]>a[8] - неверно | 5 | 6 | 6 | **17** | 45 | 39 | 37 | **17** | 13 |
| j=9, a[4]>a[9] - верно | 5 | 6 | 6 | **13** | 45 | 39 | 37 | 17 | **14** |
| **i=5,** j=6, a[5]>a[6] - верно | 5 | 6 | 6 | 13 | **39** | **45** | 37 | 17 | 14 |
| j=7, a[5]>a[7] - верно | 5 | 6 | 6 | 13 | **37** | 45 | **39** | 17 | 14 |
| j=8, a[5]>a[8] - верно | 5 | 6 | 6 | 13 | **17** | 45 | 39 | **37** | 14 |
| j=9, a[5]>a[9] - верно | 5 | 6 | 6 | 13 | **14** | 45 | 39 | 37 | **17** |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **A[1]** | **A[2]** | **A[3]** | **A[4]** | **A[5]** | **A[6]** | **A[7]** | **A[8]** | **A[9]** |
| **i=6,** j=7, a[6]>a[7] - верно | | 5 | 6 | 6 | 13 | 14 | **39** | **45** | 37 | 17 |
| j=8, a[6]>a[8] - верно | | 5 | 6 | 6 | 13 | 14 | **37** | 45 | **39** | 17 |
| j=9, a[6]>a[9] - верно | | 5 | 6 | 6 | 13 | 14 | **17** | 45 | 39 | **37** |
| **i=7,** j=8, a[7]>a[8] - верно | | 5 | 6 | 6 | 13 | 14 | 17 | **39** | 45 | 37 |
| j=9, a[7]>a[9] - верно | | 5 | 6 | 6 | 13 | 14 | 17 | **37** | 45 | **39** |
| **i=8,** j=9, a[8]>a[9]  неверно | - | 5 | 6 | 6 | 13 | 14 | 17 | 37 | **39** | **45** |

Из таблицы видно, элементы в массиве расположились по возраста- нию, т.е. программа предназначена для сортировки элементов одномер- ного массива (методом обмена) по возрастанию.

**Задание:** модифицируйте программу так, чтобы она выполняла сортировку элементов массива методом выбора[15](#_bookmark31).

#### *Контрольные вопросы:*

* 1. Что такое массив? Как объявить одномерный массив в языке Паскаль?
  2. Приведите примеры задач, в алгоритме решения которых используются одномерные массивы.
  3. Запишите блок операторов для заполнения массива 50 числами с клавиатуры.
  4. Пусть элементами массива A (A[1], A[2], A[3], A[4]) являются соответственно x, -x, x\*x, -x\*x. Чему будет равно значение выражения A[-A[A[3]-2]]+A[-A[A[3]]] при x=1?
  5. Опишите словесно алгоритм поиска максимального (минимального) элемента в одномерном массиве.
  6. Опишите словесно алгоритм сортировки элементов одномерного массива методом пузырька, выбора.

## п. 7. Лабораторная работа № 7. Обработка двумерных массивов

***Цель работы:*** формирование совокупности знаний об особенностях организации работы с массивами, умений анализировать исходные данные задачи; овладение практическими навыками раз- работки, программирования вычислительного процесса с использованием двумерных массивов; получение дальнейших навыков по отладке и тестированию программы; умений оформлять результаты своей деятельности в современном офисном пакете и аргументировать их.

1. Подробнее о методе сортировки см., например, [http://ru.wikipedia.org/wiki/Сортировка\_выбором](http://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D0%B2%D1%8B%D0%B1%D0%BE%D1%80%D0%BE%D0%BC)

#### *Вопросы для подготовки к лабораторной работе:*

* 1. Что такое «массив»? Что такое «размерность массива», «размер массива»?
  2. Какие бывают массивы?
  3. С чем сравнивают одномерный, двумерный массивы?
  4. Как объявить двумерный вещественный массив А из 50 элементов?
  5. Какие существуют способы заполнения двумерного массива? Запишите фрагмент кода программы на языке Паскаль для заполнения массива 50 числами с клавиатуры.
  6. Как организовать перебор всех элементов одномерного массива? Запишите фрагмент кода программы на языке Паскаль для перебора всех элементов массива и подсчета их суммы.

#### *Ход работы:*

***Постановка задачи 7\_1:*** *выполнить пояснения к программе (выделить структурные единицы программы) и сформулировать задачу, для решения которой программа написана. Набрать программу, проверить ее работо- способность. Сохранить в своей папке, создать exe-файл. Разработать блок- схему алгоритма.*

**program** lab7\_1;

**var** i, j, kol:integer; x:**array**[1..5,1..5] **of** integer;

**begin**

randomize;

**for** i:=1 **to** 5 **do for** j:=1 **to** 5 **do**

x[i,j]:=random(20)-10;

**for** i:=1 **to** 5 **do begin**

**for** j:=1 **to** 5 **do** write(x[i,j]:6); writeln

**end**; writeln('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'); kol:=0;

**for** i:=1 **to** 5 **do for** j:=1 **to** 5 **do**

**if** x[i,j]>0 **then** inc(kol); writeln('Количество положительных=', kol); **end**.

#### *Решение:*

Пояснения к программе:

**program lab7\_1;**

**Заголовок программы** lab7\_1

|  |  |
| --- | --- |
| **var** | **Раздел описания переменных** |
| **x:array[1..5,1..5] of integer;**  **i, j, kol:integer;** | Массив х: размер 5х5 элементов, тип — цело- численный  Параметры цикла: имена i, j, тип – целое число Сумма положительных чисел: имя kol, тип — целое число |
| **begin** | **Раздел операторов (основной блок)** |
| **randomize;** | Инициализация генератора случайного числа |
| **for i:=1 to 5 do for j:=1 to 5 do** | Организация вложенных циклов с параметрами i и j, изменяющимися в диапазоне от 1 до 5 с шагом +1 |
| **x[i,j]:=random(20)-10;** | Тело цикла: генерация случайного числа из от- резка [-10; 10] и сохранение его в массиве х на строке с номером i и столбце с номером j |
| **for i:=1 to 5 do** | Открываем внешний цикл по переменной i в диапазоне от 1 до 5( для вывода всех элементов двумерного массива в виде матрицы) |
| **begin** | **Начало действий во внешнем цикле** |
| **for j:=1 to 5 do** | Открываем внутренний цикл по переменной j в диапазоне от 1 до 5 |
| **write(х[i,j]:5);** | Тело внутреннего цикла: вывод элемента массива x с индексами [i ,j] в строку |
| **writeln** | Перевод каретки в начало новой строки |
| **end;** | **Конец действий во внешнем цикле** |
| **writeln('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');** | Вывод на экран строки из символов из \* |
| **kol:=0;** | Обнуление значения переменной kol: будем считать, что положительных элементов в мас- сиве нет |
| **for i:=1 to 5 do for j:=1 to 5 do** | Организация вложенных циклов с параметрами i и j, изменяющимися в диапазоне от 1 до 5 с шагом +1 |
| **if x[i,j]>0 then inc(kol);** | Тело цикла: если элемент массива x с индексами i, j оказался больше нуля, то увеличиваем значе- ние переменной kol на 1 |
| **writeln('Количество по-** | Вывод количества положительных элементов |

|  |  |
| --- | --- |
| **ложительных=', kol);** | массива |
| **end.** | **Завершение раздела операторов (текста про- граммы)** |

Формулировка задачи: найти количество положительных элемен- тов двумерного массива Х размера 5х5.

#### *Контрольный счет:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| -7 | 9 | 6 | -8 | -10 |
| 4 | -10 | 0 | -8 | 2 |
| 8 | -6 | 3 | 3 | -2 |
| -7 | -7 | 2 | -1 | -1 |
| -2 | -5 | -5 | -3 | 2 |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Количество положительных=9

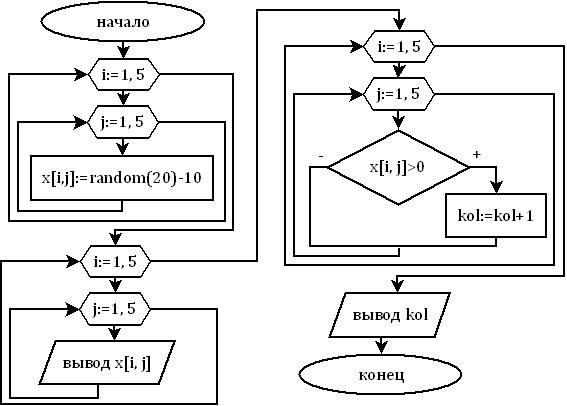


Рисунок 7.1: Блок-схема к заданию 7\_1

**Задание:** модифицируйте программу так, чтобы она выполняла вы- числение произведения модуля отрицательных и количества нуле- вых элементов массива. Размер массива NxN задавать в диалоговом режиме.

***Постановка задачи 7\_2:*** *проанализировать программу, определить, для*

*решения какой задачи написана программа. Составить таблицу трасси-*

*ровки.*

**program** lab7\_2;

**var**

i, j, k, n, t:integer; x:**array**[1..100,1..100] **of** integer; **begin**

write('Введите размер матрицы n<=100: '); read(n); write('Введите k<=n: '); read(k);

randomize;

**for** i:=1 **to** n **do for** j:=1 **to** n **do**

x[i,j]:=random(10); **for** i:=1 **to** n **do begin**

**for** j:=1 **to** n **do** write(x[i,j]:6); writeln

**end**; writeln('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'); **for** i:=1 **to** n **do**

**begin**

t:=x[i,i];

**end**;

x[i,i]:=x[i,k];

x[i,k]:=t;

**for** i:=1 **to** n **do**

**begin**

**for** j:=1 **to** n **do** write(x[i,j]:6); writeln

**end**; **end**.

#### *Решение:*

В программе сначала формируется массив X размера n, затем с помощью генератора случайного числа массив заполняется целыми числами из интервала (0, 10), сформированный массив выводится на экран.

Затем в цикле по переменной i элементы массива меняются местами (для этого используется прием переприсваивания). Проанализируем именно этот блок программы:

**for** i:=1 **to** n **do begin**

t:=x[i,i]; x[i,i]:=x[i,k]; x[i,k]:=t;

**end**;

Допустим, мы ввели размер массива n=6, k=4. Пусть массив был заполнен следующими числами:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | j=1 | j=2 | j=3 | j=4 | j=5 | j=6 |
| i=1 | **6** | **0** | **0** | **5** | **2** | **4** |
| i=2 | **7** | **5** | **1** | **2** | **2** | **8** |
| i=3 | **4** | **8** | **6** | **1** | **7** | **9** |
| i=4 | **1** | **9** | **5** | **1** | **6** | **6** |
| i=5 | **3** | **3** | **7** | **7** | **9** | **4** |
| i=6 | **1** | **8** | **7** | **6** | **5** | **8** |

#### *Таблица трассировки*[*16*](#_bookmark33)*:*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Оператор** | | **Значения** | | **переменных** | |
| **n** | **t** | **x[i,i]** | **x[i,k]** |
| **for i**:=1 **to n do** | | Начало цикла с параметром: i меняется от 1 до 6 с шагом +1 | | | |
| t:=x[i,i]; | x[i,i]:=x[i,k]; | i=1 | 6 | 5 | 6 |
| x[i,k]:=t; |  | i=2 | 5 | 2 | 5 |
|  |  | i=3 | 6 | 1 | 6 |
|  |  | i=4 | 1 | 1 | 1 |
|  |  | i=5 | 9 | 7 | 9 |
|  |  | i=6 | 8 | 6 | 8 |

Выделим в исходном массиве Х те его элементы, которые участвовали в обменах:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | i=1 | i=2 | i=3 | i,k=4 | i=5 | i=6 |
| i=1 | 6 | 0 | 0 | 5 | 2 | 4 |
| i=2 | 7 | 5 | 1 | 2 | 2 | 8 |
| i=3 | 4 | 8 | 6 | 1 | 7 | 9 |
| i=4 | 1 | 9 | 5 | 1 | 6 | 6 |
| i=5 | 3 | 3 | 7 | 7 | 9 | 4 |
| i=6 | 1 | 8 | 7 | 6 | 5 | 8 |

Из таблицы видно, что программа предназначена для перестановки местами элементов двумерного массива, стоящих на главной диагонали и в k-том столбце (в нашем примере k=4).

1. Обратите внимание, что при переборе элементов двумерного массива индекс j не задействован: индекс j в некоторых случаях совпадает с именем i (то есть равен ему), в некоторых – c именем k.

**Задание:** модифицируйте программу так, чтобы она выполняла перестановку местами элементов главной и побочной диагоналей.

***Постановка задачи 7\_3:*** *вычислить сумму и число положительных эле- ментов двумерного массива С* ( *N , N* ) *, находящихся над главной диагональю матрицы, где N принадлежит промежутку от 12 до 100*[*17*](#_bookmark34)*: если будет вве- дено другое значение N, то запросить размер массива заново. Выделить структурные единицы программы и выполнить пояснения к ним.*

#### *Решение: Программа:*

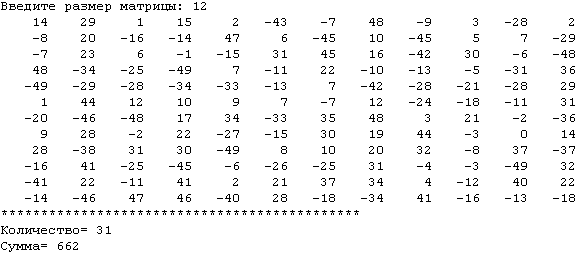
|  |  |
| --- | --- |
| **program** lab7\_3; | **Заголовок программы** |
| **label** vozvrat; | **Раздел объявления меток:**  имя: vozvrat — возврат в начало программы в случае , если N не принадлежит отрезку [12,100] |
| **var** | **Раздел объявления переменных** |
| **i**, j, k, n, summa : integer; | имена i, j: параметры циклов  имя k — количество положительных элементов над диагональю  имя n — размер массива  имя summa — сумма положительных элементов над диагональю |
| c:**array**[1..100,1..100] **of**  integer; | имя c — двумерный целочисленный массив |
| **begin** | **Начало программы** |
| vozvrat: | Метка vozvrat |
| write('Введите размер матрицы: '); | Вывод строки «Введите размер матрицы» |
| read(n); | Считывание введенного значения и сохранение под именем n |
| **if** (n<12) **or** (n>100) **then** | Проверка условия: если n не попало в отрезок [12, 100] |
| **begin** | **Тогда начать** |
| writeln('Введите число 12<=x<=100'); | Вывод строки «Введите число от 12 до 100» |

1. Замечание: на этапе разработки и тестирования программы рекомендуем взять размер массива <12

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **goto** vozvrat; | Перейти к строке с меткой vozvrat | | | |
| **end**; | **Конец действий если условие истинно** | | | |
| randomize; | Инициализация датчика псевдослучайных чисел | | | |
| k:=0; | Обнуление значения переменной k | | | |
| summa:=0; | Обнуление значения переменной summa | | | |
| **for** i:=1 **to** n **do** | Открываем два вложенных друг в друга цикла для перебора всех элементов двумерного массива | | | |
| **for** j:=1 **to** n **do** |
| **begin** | **Начало действий в циклах** | | | |
| c[i,j]:=random(101)-50; | Присвоить элементу массива С с индексами [i, j] значение, равное псевдослучайному числу из интервала (-50, 50) | | | |
| **if** c[i,j]>0 **then** | Если сформированное значение элемента массива c[i, j]>0, тогда | | | |
| **if** j>i **then** | Если для его индексов выполняется условие j>i (элемент стоит над диагональю), то | | | |
| **begin** | **Начало действий истинны)** | **(если** | **оба** | **условия** |
| inc(k); | Увеличить значение k на 1 | | | |
| summa:=summa+c[i,j]; | Увеличить значение summa на значение элемента массива c[i, j] | | | |
| **end**; | **Конец действий истинны)** | **(если** | **оба** | **условия** |
| **end**; | **Конец действий в циклах** | | | |
| **for** i:=1 **to** n **do** | Открываем два вложенных друг в друга цикла для вывода всех элементов двумерного массива | | | |
| **begin** |
| **for** j:=1 **to** n **do** |
| write(c[i,j]:5); | Вывод элемента массива с индексами [i ,j] в строку | | | |
| writeln | Перевод каретки в начало новой строки | | | |
| **end**; | **Конец действий в циклах** | | | |
| writeln('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*'); | Вывод строки «\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*» | | | |

|  |  |
| --- | --- |
| writeln('Количество= ',k); | Вывод количества положительных элементов |
| write('Сумма= ',summa); | Вывод суммы положительных элементов |
| **end**. | **Конец программы** |

#### *Контрольный счет:*



***Задание:*** модифицируйте программу так, чтобы она находила произведение положительных и количество нулевых элементов матрицы, расположенных под главной диагональю. Все элементы матрицы должны принадлежать диапазону [-10; 10].

***Постановка задачи 7\_4:*** *сформировать квадратную матрицу* А[n,n] *с помощью генератора случайных чисел. «Вырезать» в этой матрице подматрицу размером* [m,m] (m<n), *в которой сумма диагональных элементов наименьшая. Вывести координаты этой подматрицы – номера строки и столбца левого нижнего ее угла, а также построчно в форматном виде элементы подматрицы.*

***Решение: Программа:***

**program** lab7\_4;

**const** nn=50; mm=50;

**var** i, j, l, k, n, m : integer;

a, b:**array**[1..nn, 1..nn] **of** integer; s, min, nstroki, nstolbza :integer; **begin**

writeln('Введите размер матрицы'); readln(n); writeln('Введите размер подматрицы'); readln(m); randomize;

**for** i:=1 **to** n **do for** j:=1 **to** n **do**

a[i,j]:=trunc(random\*50);

writeln(' Сформированная матрица');

**for** i:=1 **to** n **do begin**

**for** j:=1 **to** n **do**

write(a[i, j]:6); writeln;

**end**;

**for** i:=1 **to** m **do for** j:=1 **to** m **do**

**if** i=j **then** s:=s+a[i, j];

min:=s; nstroki:=m; nstolbza:=1;

**for** l:=1 **to** n-m+1 **do for** k:=1 **to** n-m+1 **do begin**

**for** i:=1 **to** m **do for** j:=1 **to** m **do**

b[i, j]:=a[l+i-1, k+j-1]; s:=0;

**for** i:=1 **to** m **do for** j:=1 **to** m **do**

**if** i=j **then** s:=s+b[i, j];

**if** s<min **then begin**

min:=s; nstroki:=l+m-1; nstolbza:=k;

**end**; **end**;

writeln('Номер строки = ', nstroki, ' Номер столбца = ', nstolbza); writeln(' Искомая подматрица');

**for** i:=nstroki-m+1 **to** nstroki **do begin**

**for** j:=nstolbza **to** nstolbza+m-1 **do**

write(a[i, j]:6); writeln;

**end**;

**end**.

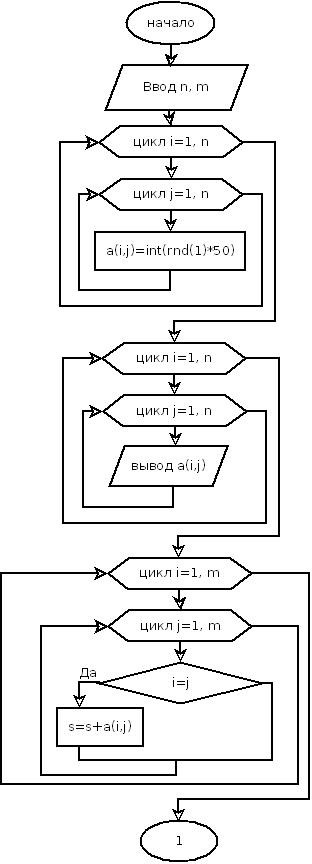
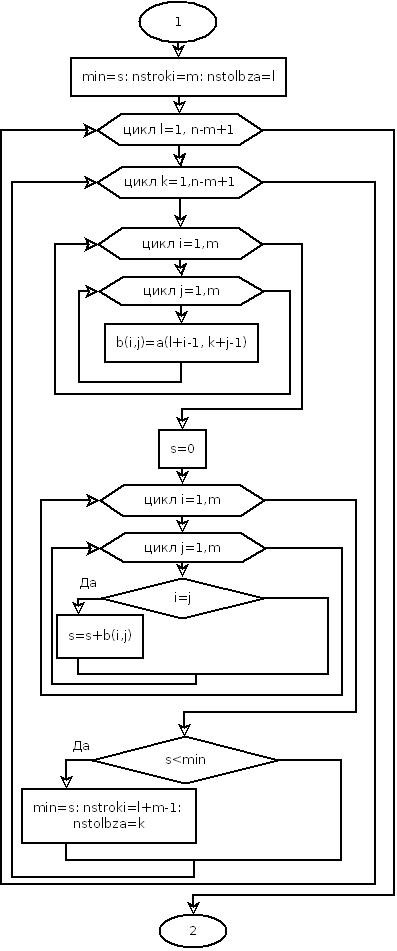
 

Рисунок 7.2: Блок-схема к заданию 7\_4

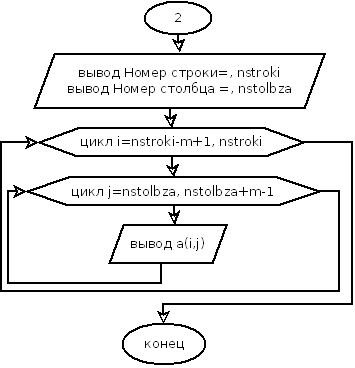
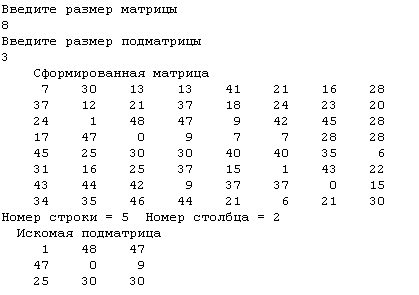


Рисунок 7.3: Блок-схема к заданию 7\_4 (продолжение)

#### *Контрольный счет:*



**Задание:** модифицировать программу так, чтобы она осуществляла поиск подматрицы размера n, в которой сумма элементов главной диагонали наибольшая.

***Постановка задачи 7\_5:*** *составить программу для решения системы*

2 *x*+ *y*+*z* =7

*линейных алгебраических уравнений* {

*x* +2 *y*+*z* =8

*x*+ *y*+2 *z* =9

*методом Крамера:*

*а) в предположении, что система совместна и имеет единственное решение;*

*б) в предположении, что система может быть как совместной, так и несовместной.*

#### *Решение: Математическая модель:*

Для системы трех уравнений с тремя неизвестными требуется [Выгодский, С.237-238.]:

* 1. Вычислить определитель системы Δ=∣ ∣ по формуле:

*a*31 *a*32 *a*33∣

*a*11 *a*12 *a*13 *a*21 *a*22 *a*23

*a*31 *a*32 *a*33

.

*a*21 *a*22 *a*23 =*a*11 *a*22 *a*33+*a*12 *a*23 *a*31+*a*13 *a*21 *a*32−*a*31 *a*22 *a*13−*a*32 *a*23 *a*11−*a*33 *a*21 *a*12

* 1. Вычислить определители, полученные поочередной заменой каждого из трех столбцов на столбец свободных членов:

*b*1 *a*12 *a*13

∣

*b*3 *a*32 *a*33

*a*11 *b*1 *a*13

, Δ *y*= *a*21 *b*2 *a*23

∣ ∣

*a*31 *b*3 *a*33

*a*11 *a*12 *b*1

, Δ *z*= *a*21 *a*22 *b*2 .

∣ ∣

*a*31 *a*32 *b*3

* 1. Проанализировать полученные значения:

a)определитель системы Δ≠0 : система совместна, единственное решение системы ищем по формулам: *x*= Δ *x , y*= Δ*y , z* = Δ*z* ;

Δ Δ Δ

b)определитель системы Δ=0 , при этом один из определителей

Δ *x ,* Δ *y ,* Δ*z* ≠0 : система несовместна, решений не имеет;

c) все полученные определители равны 0: система совместна, имеет бесчисленное множество решений.

#### *Программа:*

1. Приведем программу для решения задачи под цифрой 1. Коэффициенты системы уравнений будем хранить в константах — одномерном и двумерном массивах.

**program** lab7\_5;

### const

a: **array**[1..3, 1..3] **of** integer = ((2, 1, 1), (1, 2, 1), (1, 1, 2));

b: **array**[1..3] **of** integer = (7, 8, 9);

### var

i, j: integer;

delta, delta1, delta2, delta3, x1, x2, x3: real;

### begin

delta := a[1, 1] \* a[2, 2] \* a[3, 3] + a[1, 2] \* a[2, 3] \* a[3, 1] + a[1, 3] \* a[2, 1] \* a[3,

2] - a[3, 1] \* a[2, 2] \* a[1, 3] - a[3, 2] \* a[2, 3] \* a[1, 1] - a[3, 3] \* a[2, 1] \* a[1, 2];

delta1 := b[1] \* a[2, 2] \* a[3, 3] + a[1, 2] \* a[2, 3] \* b[3] + a[1, 3] \* b[2] \* a[3, 2] -

b[3] \* a[2, 2] \* a[1, 3] - a[3, 2] \* a[2, 3] \* b[1] - a[3, 3] \* b[2] \* a[1, 2];

delta2 := a[1, 1] \* b[2] \* a[3, 3] + b[1] \* a[2, 3] \* a[3, 1] + a[1, 3] \* a[2, 1] \* b[3] -

a[3, 1] \* b[2] \* a[1, 3] - b[3] \* a[2, 3] \* a[1, 1] - a[3, 3] \* a[2, 1] \* b[1];

delta3 := a[1, 1] \* a[2, 2] \* b[3] + a[1, 2] \* b[2] \* a[3, 1] + b[1] \* a[2, 1] \* a[3, 2] -

a[3, 1] \* a[2, 2] \* b[1] - a[3, 2] \* b[2] \* a[1, 1] - b[3] \* a[2, 1] \* a[1, 2];

x1 := delta1 / delta; x2 := delta2 / delta; x3 := delta3 / delta;

writeln('Найденное решение:'); writeln('x1=', x1, ' x2=', x2, ' x3=', x3); **end**.

#### *Контрольный счет:*

Найденное решение: x1=1 x2=2 x3=3

1. Модернизируем программу для системы линейных уравнений с тремя переменными с любыми целочисленными коэффициентами. Коэффициенты системы уравнений будем вводить с клавиатуры.

**program** lab7\_5;

**var** a: **array**[1..3, 1..3] **of** integer; b: **array**[1..3] **of** integer;

i, j: integer;

delta, delta1, delta2, delta3, x1, x2, x3: real;

### begin

writeln('Введите коэффициенты системы при неизвестных x1, x2, x3');

**for** i := 1 **to** 3 **do for** j := 1 **to** 3 **do** read(a[i, j]);

writeln('Введите столбец свободных членов');

**for** i := 1 **to** 3 **do** read(b[i]);

delta := a[1, 1] \* a[2, 2] \* a[3, 3] + a[1, 2] \* a[2, 3] \* a[3, 1] + a[1, 3] \* a[2, 1] \* a[3,

2] - a[3, 1] \* a[2, 2] \* a[1, 3] - a[3, 2] \* a[2, 3] \* a[1, 1] - a[3, 3] \* a[2, 1] \* a[1, 2];

delta1 := b[1] \* a[2, 2] \* a[3, 3] + a[1, 2] \* a[2, 3] \* b[3] + a[1, 3] \* b[2] \* a[3, 2] -

b[3] \* a[2, 2] \* a[1, 3] - a[3, 2] \* a[2, 3] \* b[1] - a[3, 3] \* b[2] \* a[1, 2];

delta2 := a[1, 1] \* b[2] \* a[3, 3] + b[1] \* a[2, 3] \* a[3, 1] + a[1, 3] \* a[2, 1] \* b[3] -

a[3, 1] \* b[2] \* a[1, 3] - b[3] \* a[2, 3] \* a[1, 1] - a[3, 3] \* a[2, 1] \* b[1];

delta3 := a[1, 1] \* a[2, 2] \* b[3] + a[1, 2] \* b[2] \* a[3, 1] + b[1] \* a[2, 1] \* a[3, 2] -

a[3, 1] \* a[2, 2] \* b[1] - a[3, 2] \* b[2] \* a[1, 1] - b[3] \* a[2, 1] \* a[1, 2];

**if** delta <> 0 **then**

### begin

x1 := delta1 / delta; x2 := delta2 / delta; x3 := delta3 / delta; writeln('Найденное решение:'); writeln('x1=', x1, ' x2=', x2, ' x3=', x3); **end**

### else

**if** (delta1 = 0) **and** (delta2 = 0) **and** (delta3 = 0) **then** writeln( 'Система имеет бесконечно много решений') **else**

writeln('Система решений не имеет');

**end**.

#### *Контрольный счет:*

*Тестовый набор 1 (единственное решение)*

Введите коэффициенты системы при неизвестных x1, x2, x3 3 4 2

5 -6 -4

-4 5 3

Введите столбец свободных членов 5 -3 1

Найденное решение: x1=1 x2=-2 x3=5

*Тестовый набор 2 (решений нет)*

Введите коэффициенты системы при неизвестных x1, x2, x3 1 1 1

1 -1 1

1 0 1

Введите столбец свободных членов 5 1 2

Система решений не имеет

*Тестовый набор 3 (бесконечно много решений)* Введите коэффициенты системы при неизвестных x1, x2, x3 1 1 1

3 -1 2

1 -3 0

Введите столбец свободных членов 0 0 0

Система имеет бесконечно много решений

**Учебно-исследовательское**

так,

чтобы

она

**задание:**

находила

модифицировать

программу

решение

системы

алгебраических уравнений 4 порядка с любыми вещественными

коэффициентами.

#### *Контрольные вопросы:*

1. Запишите синтаксис объявления двумерного массива размера 50х50 вещественного типа данных.
2. Как объявить массив контант?
3. Запишите программный блок для заполнения двумерного массива вещественными числами с клавиатуры.
4. Запишите программный блок для заполнения двумерного массива числами из диапазона от -20 до 20 с помощью генератора случайного числа.
5. Запишите программный блок для заполнения двумерного массива числами по формуле: *a* [ *i , j* ]=(*i*+ *j* )2 .
6. Запишите программный блок для вывода элементов двумерного массива в виде матрицы.
7. Какой блок позволяет осуществить перебор всех элементов двумерного массива?
8. Можно ли выполнять обход двумерного массива, организовав внешний цикл по столбцам, а внутренний — по строкам?
9. Целесообразно ли использовать вложенные циклы, если совершается обход только главной диагонали квадратной матрицы? одной строки матрицы? одного столбца матрицы?
10. При решении каких задач удобно использовать двумерные массивы? Приведите примеры.

### ГЛОССАРИЙ[18](#_bookmark36)

**Аппроксимация (приближение)** — научный метод, состоящий в замене одних объектов другими, в том или ином смысле близкими к исходным, но более простыми.

**Блок** — раздел описаний, после которого следуют операторы, заключенные в операторные скобки begin/end.

**Ввод данных** – чтение данных с устройства ввода (например, клавиатуры) в оперативную память компьютера. При этом данные из

«внешнего» представления кодируются в представление на уровне компьютера[19](#_bookmark37).

**Вывод данных** – отображение (запись) полученных результатов на устройстве вывода (например, дисплей). При этом результаты из представления на уровне компьютера преобразуются к виду, понятному человеку.

**Выражение** – конструкция языка для вычисления и изменения значений.

**Вычисление** – выполнение последовательности действий, которые необходимы для преобразования данных в результаты. Данная последовательность должна быть описана таким образом, чтобы ее мог выполнить процессор компьютера.

**Генератор псевдослучайных чисел** — [алгоритм](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC), генерирующий последо- вательность [чисел](http://ru.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%BE), элементы которой почти независимы друг от друга и подчиняются заданному распределению (обычно р[авномерному](http://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D0%B5_%D1%80%D0%B0%D0%B2%D0%BD%D0%BE%D0%BC%D0%B5%D1%80%D0%BD%D0%BE%D0%B5_%D1%80%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5)).

**Данные** с точки зрения программиста — это часть программы, совокупность значений определённых ячеек памяти, преобразование которых осуществляет код.

**Данные** с точки зрения компилятора, процессора, операционной системы, это совокупность ячеек памяти, обладающих определёнными свойствами (возможность чтения и записи (необяз.), невозможность исполнения).

**Диапазон массива** — количество допустимых значений каждого индекса массива.

**Динамический массив** — массив, размер которого может меняться во время исполнения программы.

**Запись** – это структурированный тип данных, состоящий из фиксированного числа компонентов разного типа.

**Идентификатор** – имена объектов языка программирования (типов, констант, переменных и др.).

1. Большинство определений были взяты из энциклопедии [http://progopedia.ru](http://progopedia.ru/) и [http://ru.wikipedia.org](http://ru.wikipedia.org/)/

19 Введение в программирование. Pascal и Delphi: лабораторный практикум / Л. В. Гурьянов, Л. С. Гурьянова, С. В. Самуйлов. – Пенза: Изд-во Пенз. гос. ун-та, 2006. – 168 с., С.9

**Идентификатор, ID** — это уникальный признак объекта, позволяющий различать объекты и/или объект по идентификатору.

**Императивная парадигма программирования** описывает процесс вычислений посредством описания управляющей логики программы, т.е. в виде последовательности отдельных команд, которые должен выполнить компьютер.

**Интерпретатор** (языка программирования) — программа или техническое средство, выполняющее интерпретацию; вид транслятора, осуществляющего пооператорную (покомандную) обработку и выполнение исходной программы или запроса (в отличие от компилятора, транслирующего всю программу без её выполнения)

**Исполнимый (исполняемый) модуль, исполнимый файл** — [файл](http://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB), содержащий программу в виде, в котором она может быть (после загрузки в память и настройки по месту) исполнена компьютером.

**Исходная программа** — обычный текстовый файл, содержащий запись операторов программы на языке программирования высокого уровня.

**Исходный код** — текст компьютерной программы на каком-либо языке программирования, который может быть прочтён человеком.

**Итерация** — это организация обработки данных, при которой действия повторяются многократно, не приводя при этом к вызовам самих себя (не путать с рекурсией).

**Ключевое слово** – идентификатор, используемый в качестве фиксированной части синтаксиса какой-либо конструкции языка (if, for, next и др.).

**Комментарий** – пояснение, включаемое в текст программы, но не обрабатываемое компилятором.

**Компиляция** — трансляция исходного кода компьютерной программы или отдельного программного модуля, составленных на языке программирования (исходная программа, исходный модуль) в программу или модуль на машинном языке или языке, близком к машинному (объектная программа, объектный модуль).

**Константа** – это данное, значение которого не изменяется при выполнении программы.

**Константа** — это ячейка (или несколько ячеек) оперативной памяти компьютера, имеющая определенное имя, в которой хранится неизменяемая величина.

**Лексема** *—* минимальная значимая единица текста программы.

**Массив** — структура данных, состоящая из фиксированного числа элементов, имеющих один и тот же тип и расположенных в памяти последовательно.

**Машинная программа** — программа на языке программирования низкого уровня (например, Assembler), являющаяся последовательностью

машинных команд, распознаваемая и выполняемая центральным процессором компьютера.

**Метод Монте-Карло** — общее название группы численных методов, осно- ванных на получении большого числа реализаций стохастического (слу- чайного) процесса, который формируется таким образом, чтобы его веро- ятностные характеристики совпадали с аналогичными величинами решае- мой задачи. Используется для решения задач в различных областях физи- к[и, химии](http://ru.wikipedia.org/wiki/%D0%A5%D0%B8%D0%BC%D0%B8%D1%8F), математики, экономики, оптимизации,теории управления и др. **Метод прямоугольников** — метод численного интегрирования функции одной переменной, заключающийся в приближённом вычислении площа- ди под графиком (подынтегральной функции) суммированием площадей конечного числа прямоугольников, ширина которых будет определяться расстоянием между соответствующими соседними узлами интегрирова- ния, а высота — значением подынтегральной функции в этих узлах.

**Метод трапеций** — метод численного интегрирования функции одной переменной, заключающийся в замене на каждом элементарном отрезке подынтегральной функции на многочлен первой степени, то есть линей- ную функцию. Площадь под графиком функции аппроксимируется прямо- угольными трапециями.

**Модуль** — это особым образом оформленная библиотека подпрограмм. **Набор символов** – буквы и цифры (например, символы ASCII, американского стандарта обмена информацией), а также ряд специальных символов.

**Оператор** – конструкция языка, необходимая для управления порядком действий по обработке информации.

**Операция** *—* конструкция в языках программирования, аналогичная по записи математическим операциям, то есть специальный способ записи некоторых действий.

**Описание (объявление) переменной** — выделение памяти для хранения ее значения и определение самого значения переменной.

**Отладка программы** – это поиск ошибок в программе (тестирование) и исправление ошибок.

**Парадигма программирования** — это комплекс концепций, принципов и абстракций, определяющих фундаментальный стиль программирования.

**Параметр (переменная цикла)** – величина, с изменением значения которой связано многократное выполнение цикла.

**Параметр-значение** — формальный параметр, который входит в группы, не содержащие слова var.

**Параметр-переменная** — формальный параметр, который входит в группы, содержащие слово var .

**Паскаль** (назван в честь Блеза Паскаля) — это чисто процедурный язык программирования, часто использующийся для обучения структурному

программированию.

**Переменная** – это данные, значения которых изменяются при выполнении программы.

**Переменная** — это ячейка (или несколько ячеек) оперативной памяти компьютера, имеющая определенное имя, в которой хранится изменяемая величина.

**Переменная** — это идентификатор, определяющий данные.

**Префикс** – обязательная предшествующая часть составного идентификатора для имен полей в структуре типа запись Паскаля.

**Программа** – последовательность операторов и других элементов языка, построенная в соответствии с определенными правилами и предназначенная для решения определенной задачи[20](#_bookmark38).

**Программирование** — процесс составления упорядоченной последова- тельности действий (программы) для ЭВМ; научная дисциплина, изучаю- щая программы для ЭВМ и способы их составления, проверки и улучше- ния.

**Простой оператор** – оператор, который не содержит в себе других опера- торов (оператор присваивания, оператор перехода и др.)

**Пространство имен —** область в программе, в которой имя должно иметь единственное описание (блок, модуль, класс).

**Процедурное программирование** — парадигма, основанная на использо- вании процедур. Процедурное программирование является частным слу- чаем императивной парадигмы.

**Размерность (ранг) массива** — количество индексов массива. **Рекуррентная формула** — формула вида *an*= *f* ( *n , an*−1 *, an*−2 *,*... *, an*− *p* ) *, n*≥ *p* , выражающая каждый член последовательности *an* ( *n*∈ *N* ) через *p* преды- дущих членов.

**Синтаксис языка программирования** – это система обозначений правильных последовательностей синтаксических элементов программы[21](#_bookmark39).

**Случайные и псевдослучайные числа** — числа или цифры, последова- тельность появления которых обладает теми или иными статистическими закономерностями. Случайные числа генерируются каким-либо стохасти- ческим устройством, псевдослучайные числа конструируются с помощью арифметических алгоритмов.

**Составной оператор** — оператор, объединяющий группу операторов в единое целое, состоит из последовательности объединяемых операторов, которые располагаются между ключевыми словами begin и end[22](#_bookmark40) .

1. Turbo Pascal / С.А. Немнюгин. - СПб.: Издательство «Питер», 2000. - 496с. - C.16
2. Введение в программирование. Pascal и Delphi: лабораторный практикум / Л. В. Гурьянов, Л. С. Гурьянова, С. В. Самуйлов. – Пенза: Изд-во Пенз. гос. ун-та, 2006. – 168 с., С.10
3. Программирование в среде Borland Pascal 7.0/Марченко А.И., Марченко Л.А.: Под ред. Тарасенко В.П. - К.:ЮНИОР, 1998. - 480 с. - С.87.

**Статический массив** — массив, размер которого не меняется во время ис- полнения программы.

**Строка** – упорядоченная последовательность символов.

**Строковая константа** – последовательность символов, заключенных в апострофы.

**Строковый тип данных** – структурированный тип в Паскале длиной от 0 до 255 символов (1 байт памяти).

**Структурный оператор** *—* оператор, строящийся из других операторов, порядок выполнения которых должен быть последовательным (составные операторы и операторы над записями), определяемым условной переда- чей управления (условные операторы) или повторяющимся (операторы цикла).

**Табулирование функции** — это вычисление значений функции при изме- нении аргумента от некоторого начального значения до некоторого ко- нечного значения с определенным шагом.

**Тело цикла** — последовательность инструкций, предназначенная для многократного исполнения.

**Тип** (сорт) — относительно устойчивая и независимая совокупность элементов, которую можно выделить во всём рассматриваемом множестве (предметной области).

**Тип данных** — определяет множество допустимых значений, набор операций, способ реализации хранения значений и выполнения операций. **Тип переменной** определяет вид информации, содержащейся в ячейке, набор преобразований, которые можно выполнять над этой переменной, и множество допустимых значений этой переменной[23](#_bookmark41).

**Транслятор** — программа или техническое средство, выполняющее трансляцию программы[**24**](#_bookmark42)**.**

**Трансляция программы** — преобразование программы, представленной на одном из языков программирования, в программу на другом языке и, в определённом смысле, равносильную первой.

**Трассировка программы** — выполнение программы по шагам.

**Условие выхода или условие окончания цикла** — выражение определяющее, будет в очередной раз выполняться итерация, или цикл завершится.

**Файл** – это структура данных, обладающая двумя основными свойствами: обычно она располагается на каком-либо внешнем носителе (например, магнитная лента, магнитный или оптический диск), и, следовательно, ее размер может быть значительно больше, чем

1. Turbo Pascal / С.А. Немнюгин. - СПб.: Издательство «Питер», 2000. - 496с. - C.14.
2. ГОСТ 19781-83 // Вычислительная техника. Терминология: Справочное пособие. Выпуск

1 / Рецензент канд. техн. наук Ю. П. Селиванов. — М.: Издательство стандартов, 1989. — 168 с.

размеры других типов структур данных; ее время жизни может значительно превышать время выполнения программы, создавшей ее.

**Файловая переменная** – переменная в программе, которая используется для описания структуры файла и логических операций с файлом (операций на уровне программы).

**Фактический параметр** — параметр, указываемый при вызове процедуры или функции при ее описании.

**Форма массива** — совокупность размерности и диапазонов массива. **Формальный параметр** — параметр, указываемый в заголовке процеду- ры или функции при ее описании.

**Цикл** — разновидность управляющей конструкции в высокоуровневых языках программирования, предназначенная для организации многократного исполнения набора инструкций.

**Цикл с предусловием** — цикл, который выполняется пока истинно некоторое условие, указанное перед его началом.

**Цикл с постусловием** — цикл, в котором условие проверяется после выполнения тела цикла.

**Цикл с выходом из середины** — наиболее общая форма условного цикла. Синтаксически такой цикл оформляется с помощью трёх конструкций: начала цикла, конца цикла и команды выхода из цикла.

**Цикл со счётчиком** — цикл, в котором некоторая переменная изменяет своё значение от заданного начального значения до конечного значения с некоторым шагом, и для каждого значения этой переменной тело цикла выполняется один раз.

**Шаг цикла** — значение, на которое изменяется параметр цикла при каждом повторении.

**Язык программирования** – формальная знаковая система, предназначенная для записи компьютерных программ. Язык программирования определяет набор лексических, синтаксических и семантических правил, задающих внешний вид программы и действия, которые выполнит исполнитель (компьютер) под её управлением[25](#_bookmark43).

25 Википедия. Режим доступа: [http://ru.wikipedia.org/wiki/Язык\_программирования](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)

### СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Златопольский Д.М. Сборник задач по программированию [Текст] / Д.М. Златопольский. - 2-е изд., перераб. и доп. - СПб. : БХВ-Петербург, 2007. - 240 с.
2. Адаменко А.Н. Pascal [Текст] : На примерах из математики / А. Н. Ада- менко. - СПб : БХВ-Петербург, 2005. - 416 с.
3. Сухарев М. Turbo Pascal 7.0: Теория и практика программирования [Текст] / М. Сухарев. - СПб : Наука и Техника, 2004.
4. Turbo Pascal [Текст]/ С.А. Немнюгин. - СПб.: Издательство «Питер», 2000. - 496с.
5. Введение в программирование. Pascal и Delphi: лабораторный прак- тикум [Текст]/ Л. В. Гурьянов, Л. С. Гурьянова, С. В. Самуйлов. – Пенза: Изд-во Пенз. гос. ун-та, 2006. – 168 с.
6. Программирование в среде Borland Pascal 7.0 [Текст]/Марченко А.И., Марченко Л.А.: Под ред. Тарасенко В.П. - К.:ЮНИОР, 1998. - 480 с.
7. Выгодский М.А. Справочник по высшей математике [Текст] / М.А. Вы- годский. - М., 2000. - 864с.

**Губина Т.Н., Дякина В.А., Губин М.А. Учебное издание**

**Язык программирования Паскаль:**

**лабораторный практикум**

**Часть 1 Учебное пособие**

Технический редактор – Безногих Н.П. Техническое исполнение – Гришин В.М. Компьютерная верстка – Губина Т.Н.

Лицензия на издательскую деятельность ИД № 06146. Дата выдачи 26.10.01

Формат 60х90/16 Усл. – печ.л. 6,4 Тираж 300 экз. (1-й завод 1-50 экз.) Уч. – изд.л. 6,5 Печать трафаретная Заказ № 56

Отпечатано с готового оригинал-макета на участке операторной полиграфии

Елецкого государственного университета им. И.А.Бунина

ФГБОУ ВПО «Елецкий государственный университет им. И.А. Бунина» 399770, г. Елец, ул. Коммунаров, 28