БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Г. А. Расолько, Е. В. Кремень,
Ю. А. Кремень, Д. В. Филимонов

ВЫЧИСЛИТЕЛЬНАЯ ПРАКТИКА

В двух частях

Часть 1
Методы программирования

Рекомендовано
Учебно-методическим объединением
по естественно-научному образованию
в качестве учебно-методического пособия
для студентов учреждений высшего образования,
обучающихся по специальности
«математика (по направлениям)»,
направление специальности «математика
(научно-педагогическая деятельность)»

Учебное электронное издание

Минск, БГУ, 2023

ISBN 978-985-881-480-9 (ч. 1)
ISBN 978-985-881-479-3

УДК 004.432.045C++(075.8)
ББК 32.973.2-018.1я73

А в т о р ы:
Г. А. Расолько, Е. В. Кремень, Ю. А. Кремень, Д. В. Филимонов

Р е ц е н з е н т ы:
кафедра алгебры, геометрии и математического моделирования
Брестского государственного университета имени А. С. Пушкина
(заведующий кафедрой кандидат физико-математических наук,
доцент *А. Н. Сендер*);
доктор физико-математических наук, доцент *А. С. Кравчук*

Вычислительная практика [Электронный ресурс] : учеб.-метод. пособие. В 2 ч. Ч. 1. Методы программирования / Г. А. Расолько [и др.]. – Минск : БГУ, 2023. – 1 электрон. опт. диск (CD-ROM). – ISBN 978-985-881-480-9.

Представлен учебный материал по курсу «Методы программирования», а также задания и задачи для отработки основных приемов программирования, не ориентированные на конкретный алгоритмический язык.

Предназначено для студентов учреждений высшего образования, обучающихся по специальности «математика (по направлениям)», направление специальности «математика (научно-педагогическая деятельность)».

Минимальные системные требования:
PC, Pentium 4 или выше;
RAM 1 Гб; Windows XP/7/10;
Adobe Acrobat

Оригинал-макет подготовлен в программе Microsoft Word

Ответственный за выпуск *Т. М. Турчиняк*
Технический редактор *В. П. Явуз*. Компьютерная верстка *Е. В. Анискевич*
Корректор *И. В. Сазонова*

Подписано к использованию 15.06.2023. Объем 2,1 МБ

Белорусский государственный университет
Управление редакционно-издательской работы
Пр. Независимости, 4, 220030, Минск
Телефон: (017) 259-70-70
email: urir@bsu.by
http://elib.bsu.by/

СОДЕРЖАНИЕ

Предисловие 4
Общие требования 5
Стиль программирования 24
Эффективность программ 28
Концепции разработки алгоритмов 33
Базовые алгоритмы 37
Задания 42
1. Табулирование непрерывной функции 42
2. Выстрел в мишень 44
3. Задачи целочисленной арифметики 46
4. Обработка динамических массивов и множества 51
5. Обработка динамических матриц 55
6. Арифметика многократной точности 57
7. Рекурсивные алгоритмы 60
8. Работа с типизированными файлами и модули 64
9. Файлы 71
Список литературы 73

ПРЕДИСЛОВИЕ

Вычислительная практика является обязательной частью подготовки специалистов. Данный вид учебной деятельности направлен на формирование и закрепление практических навыков и компетенций в процессе выполнения определенных видов работ, связанных с будущей профессиональной деятельностью.

Приведенный в данном издании материал предусматривает углубление и закрепление теоретических знаний, а также развитие практических навыков по избранной специальности. В качестве основных методов используется решение индивидуальных заданий. Умение решать практические задачи формируется на основе применения современных информационных технологий и изучения инженерии программного обеспечения.

В учебно-методическом пособии представлен минимальный теоретический материал и индивидуальные задания по темам:

- табулирование непрерывной функции;
- выстрел в мишень;
- задачи целочисленной арифметики;
- обработка динамических массивов и множества;
- обработка динамических матриц;
- арифметика многократной точности;
- рекурсивные алгоритмы;
- работа с типизированными файлами и модули;
- файлы.

ОБЩИЕ ТРЕБОВАНИЯ

Предлагается выполнить ряд заданий по вычислительной практике, которые позволят обучающимся закрепить теоретический материал курса «Методы программирования».

Каждый обучающийся получает в рамках заданной темы индивидуальную задачу и выполняет ее на ПК с использованием современных технологий программирования.

На последних занятиях защищается отчет в виде презентации по выполненным заданиям вычислительной практики.

ПЛАН РАЗРАБОТКИ АЛГОРИТМОВ И ПРОГРАММ

1. Переформулировка условия задачи на более алгоритмический вариант, если это необходимо.
2. Определение данных на входе и на выходе.
3. Четкое определение данных и их необходимое количество для решения задачи.
4. Определение ожидаемых результатов после выполнения программы.
5. Составление тестовых примеров, не забывая о «крайних случаях». Тесты должны отражать абсолютно все возможные варианты развития событий в программе.
6. Решение тестовых примеров вручную. Описание для каждого теста входных данных и ожидаемого результата. Составленный список впоследствии войдет в отчет.
7. Решение задачи на компьютере и проверка ее на всех тестовых примерах.
8. Просмотр кода программы на предмет соответствия общих требований к коду (см. раздел «Стиль программирования»), при необходимости – оптимизация и модернизация кода.
9. Просмотр кода программы с точки зрения его эффективности (см. раздел «Эффективность программ»), при необходимости – оптимизация и модернизация кода.
10. После отладки программы внесение в ее текст комментариев таким образом, чтобы сторонний программист мог легко понять код вашей программы.
11. Документирование решения задачи и оформление отчета.

ПРИМЕР ОФОРМЛЕНИЯ ОТЧЕТА ПО ЗАДАНИЮ

1. Постановка задачи.
2. Алгоритм.

3. Графическая схема иерархии подзадач или словесное описание.

4. Система тестов.

5. Листинг программы с комментариями.

6. Итоги тестирования.

Задача 1. В текстовом файле имеются целые числа, количество которых кратное трем. Читая из данного текстового файла по три числа, проверьте, существует ли треугольник со сторонами, соответствующими полученным числам. Если ответ положительный – определите тип треугольника.

Решение. Используя сведения из геометрии, получим следующий алгоритм.

Алгоритм

1. Организуем цикл на чтение данных из файла:

   • заполняем массив из трех чисел, считанных из файла;

   • вызываем подпрограмму `isTriangle`, чтобы проверить, выполняются ли для данных чисел неравенства треугольника (сумма любых двух сторон больше третьей стороны);

   • если `isTriangle` выдала истину, определяем тип треугольника (теорема косинусов!) при помощи подпрограмм `checkSides, checkAngles`;

   • в противном случае проверку не проводим.

2. Печатаем результат.

3. Конец цикла.

В программе используются следующие подпрограммы:

1) `function isTriangle (var s: TValues): boolean;`

Параметры: массив `s` трех величин. Возвращает `true`, если величины могут быть сторонами треугольника, и `false` – в противном случае. Здесь и далее

`type TValues = array[1..3] of integer;`

2) `function checkSides (var s: TValues): string;`

Параметры: массив `s` трех величин. Возвращает строку с описанием типа треугольника по отношению сторон;

3) `function checkAngles (var s: TValues): string;`

Параметры: массив `s` трех величин. Возвращает строку с описанием типа треугольника по наибольшему углу. Использует вспомогательную подпрограмму `sortSides`;

4) **procedure** sortSides (**var** s: TValues);

Параметры: массив s трех величин. Переставляет в третий элемент массива наибольшее значение. Использует вспомогательную подпрограмму swap;

5) **procedure** swap (**var** a, b: integer);

Меняет местами величины a и b.

Система тестов

1) **function** isTriangle (**var** s: TValues): boolean;

№	a	b	c	Результат	Пояснение
1	7	7	8	true	треугольник равнобедренный
2	5	6	5	true	треугольник равнобедренный
3	5	3	8	false	не существует треугольника с таким набором сторон
4	3	4	5	true	разносторонний прямоугольный треугольник
5	2	3	4	true	разносторонний тупоугольный треугольник

2) **procedure** sortSides (**var** s: TValues);

№	a	b	c	Результат
1	5	7	5	{5, 5, 7}
2	8	3	5	{5, 3, 8}

3) **function** checkSides (**var** s: TValues): string;

№	a	b	c	Результат
1	7	7	7	равносторонний
2	5	5	6	равнобедренный
3	7	10	12	разносторонний

4) **function** checkAngles (**var** s: TValues): string;

№	a	b	c	Результат
1	7	7	7	остроугольный треугольник
2	5	5	8	тупоугольный треугольник
3	3	4	5	прямоугольный треугольник

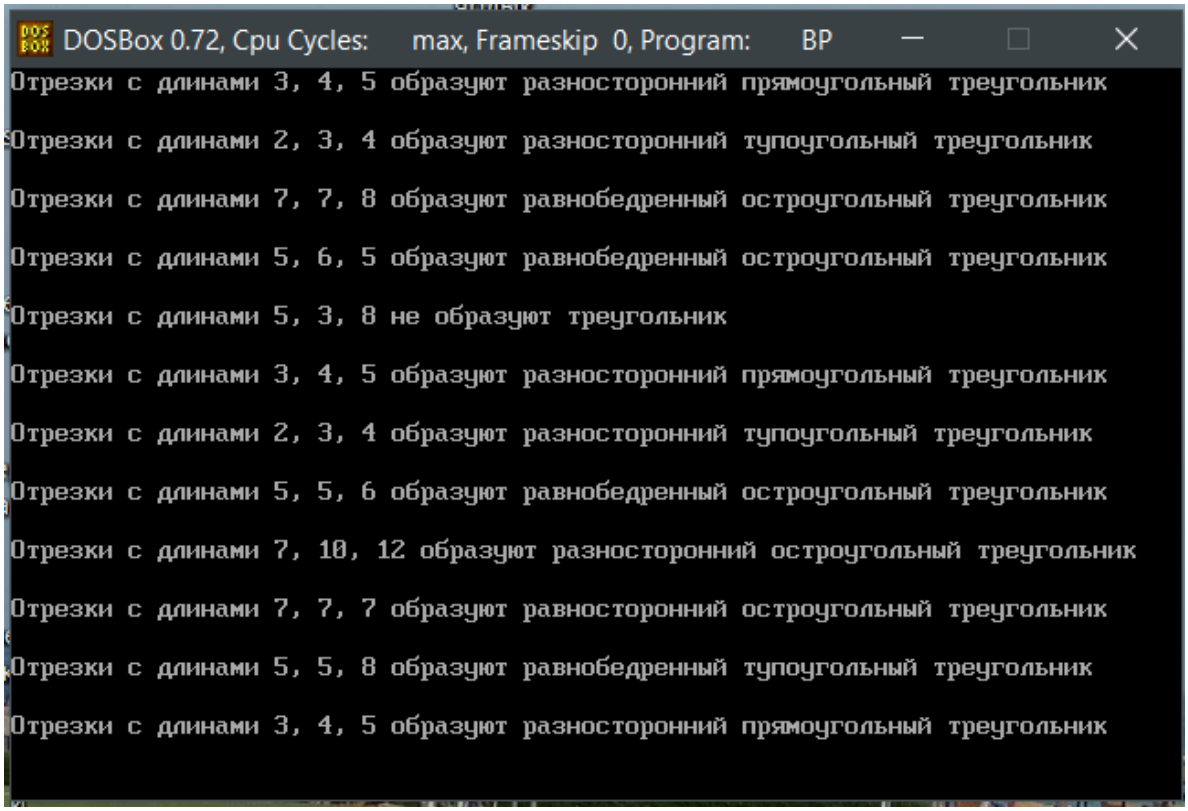
Листинг программы

```
program triangle;
uses crt;
type
```

```
        TValues = array [1..3] of integer;

   function isTriangle(var s: TValues): boolean;
    begin
     isTriangle:=(s[1]+s[2]>s[3]) and
(s[1]+s[3]>s[2]);
     isTriangle isTriangle and (s[3]+s[2]>s[1]);
    end;

   procedure swap(var a, b: integer);
   var tmp: integer;
    begin
    tmp := a;
    a   := b;
    b   := tmp;
    end;

   procedure sortSides(var s: TValues);
   begin
    if s[1]>s[3] then swap(s[1], s[3]);
    if s[2]>s[3] then swap(s[2], s[3]);
   end;

   function checkSides(var s: TValues): string;
   var AisB, AisC, BisC: boolean;
    begin
     AisB := s[1]=s[2];
     AisC := s[1]=s[3];
     BisC := s[2]=s[3];
     if AisB AND BisC then
checkSides:='равносторонний '
     else
          if AisB OR AisC OR BisC then
               checkSides :='равнобедренный '
                                 else
               checkSides :='разносторонний ';
    end;

   function checkAngles(var s: TValues): string;
```

```
var expr: integer;
 begin
    sortSides(s);
    expr := s[3]*s[3] - s[1]*s[1] - s[2]*s[2];
    if expr>0  then
       checkAngles := 'тупоугольный треугольник'
    else
     if expr<0 then
      checkAngles := 'остроугольный треугольник'
        else
      checkAngles := 'прямоугольный треугольник';
 end;

var
  s: TValues;
  filepath: string;
  t: text;
 begin
  clrscr;
  filepath:='testTriangle.txt';
  assign(t, filepath);
  reset(t);
  while not eof(t) do
   begin
    read (t, s[1], s[2], s[3]);
    if isTriangle(s) then
      begin
       write('Отрезки с длинами ',s[1],', ',
           s[2],', ',s[3],' образуют ');
       write (checkSides(s));
       writeln (checkAngles(s));
      end
    else writeln ('Отрезки с длинами ',
                  s[1],', ',s[2],', ',s[3],
                 ' не образуют треугольник');
    writeln;
   end;
   readln;
 end.
```

Результаты тестирования



Задача 2. В текстовом файле имеются вещественные числа, количество которых кратное трем. Читая из текстового файла по три числа и считая их коэффициентами уравнения степени не выше третьей, решите полученное уравнение. Рассчитайте статистические сведения о характере корней уравнений и их количестве.

Решение. Имеем уравнение вида $a_2x^2 + a_1x + a_0 = 0,$ у которого старший коэффициент может быть равен нулю. Используя известные сведения по решению уравнений второй и первой степеней, получим следующий алгоритм.

Алгоритм

1. Организуем цикл на чтение данных из файла:

• заполняем массив из трех чисел типа `TValues = array [0..2] of real`, считанных из файла;

• вызываем подпрограмму `checkEq`, которая распознает вариант ответа;

• вызываем подпрограммы получения информации по уравнению: `getNumOfRoots`, `getDetails`, `printSign`;

• печатаем результат.

2. Конец цикла.

3. Вызываем подпрограмму подсчета статистики по всем решенным уравнениям: `viewStatistics`.

4. Конец.

В программе используются следующие подпрограммы:

1) `function checkEq (var a: TValues): byte;`

Параметры: массив коэффициентов уравнения `a`. Возвращает код завершения, соответствующий типу уравнения и количеству его корней;

2) `procedure getDetails (var a: TValues; var x1, x2: real; var code: byte);`

Параметры: массив коэффициентов уравнения `a`; переменные `x1, x2` – для хранения корней (корня) уравнения, если таковые имеются; переменная `code` – для хранения типа уравнения. Использует функцию `checkEq` для получения сведений по уравнению с дальнейшим нахождением корней. Также использует вспомогательные функции `printSign, getNumOfRoots` для вывода информации на консоль;

3) `function printSign (var v: real): string;`

Параметры: величина `v`. Возвращает строку, состоящую из знака величины и самой величины для вывода на консоль;

4) `function getNumOfRoots (var hint: byte): string;`

Параметры: код завершения `hint`. Возвращает строку с указанием типа уравнения и количества его корней согласно значению `hint`;

5) `procedure viewStatistics (const codes: TCodes; cases: byte);`

Параметры: массив кодов `codes`, количество вариантов ответа при решении уравнения `cases`. Использует вспомогательную функцию `getNumOfRoots` для вывода на консоль типа уравнения, затем выводит количество уравнений соответствующего типа.

Программа тестируется на значениях, считываемых из текстового файла `testEquation.txt`.

Система тестов

1) `function checkEq (var a: TValues): byte;`

№	a_2	a_1	a_0	Результат (код завершения)
1	0	0	0	5
2	0	0	1	4
3	2	4	2	1
4	3	4	2	0

5	2	5	2	2
6	0	4	2	3
7	2	0	2	0
8	2	0	-2	2
9	2	4	0	2

2) `function getNumOfRoots (var hint: byte): string;`

№	Код	Результат (строка)
1	0	квадратное без корней
2	1	квадратное с равными корнями
3	2	квадратное с разными корнями
4	3	линейное с одним корнем
5	4	линейное без корней
6	5	с бесконечным числом корней

3) `procedure getDetails (var a: TValues; var x1, x2: real; var code: byte);`

№	a_2		a_1	a_0	Результат (строка)
1	0		0	0	уравнение $0x^2 + 0x + 0 = 0$ – с бесконечным числом корней
2	0		0	1	уравнение $0x^2 + 0x + 1 = 0$ – линейное без корней
3	2		4	2	уравнение $2x^2 + 4x + 2 = 0$ – квадратное с двумя одинаковыми корнями $x1 = -1$, $x2 = -1$
4	3		4	2	уравнение $3x^2 + 4x + 2 = 0$ – квадратное без корней
5	2		5	2	уравнение $2x^2 + 5x + 2 = 0$ – квадратное с двумя разными корнями $x1 = -1$, $x2 = -2$
6	0		4	2	уравнение $0x^2 + 4x + 2 = 0$ – линейное с одним корнем $x = -1$
7	2		0	2	уравнение $2x^2 + 0x + 2 = 0$ – квадратное без корней
8	2		0	-2	уравнение $2x^2 + 0x - 2 = 0$ – квадратное с разными корнями $x1 = 1$, $x2 = -1$
9	2		4	0	уравнение $2x^2 + 4x + 0 = 0$ – квадратное с разными корнями $x1 = 0$, $x2 = -2$

Примечание: значения переменной `code` соответствуют результатам работы функции `getNumOfRoots`.

4) **procedure** viewStatistics (**const** codes: TCodes;
cases: byte);

№	Сообщение	Результат (число случаев)
1	квадратное без корней	2
2	квадратное с двумя одинаковыми корнями	1
3	квадратное с разными корнями	3
4	линейное с одним корнем	1
5	линейное без корней	1
6	с бесконечным числом корней	1

Листинг программы

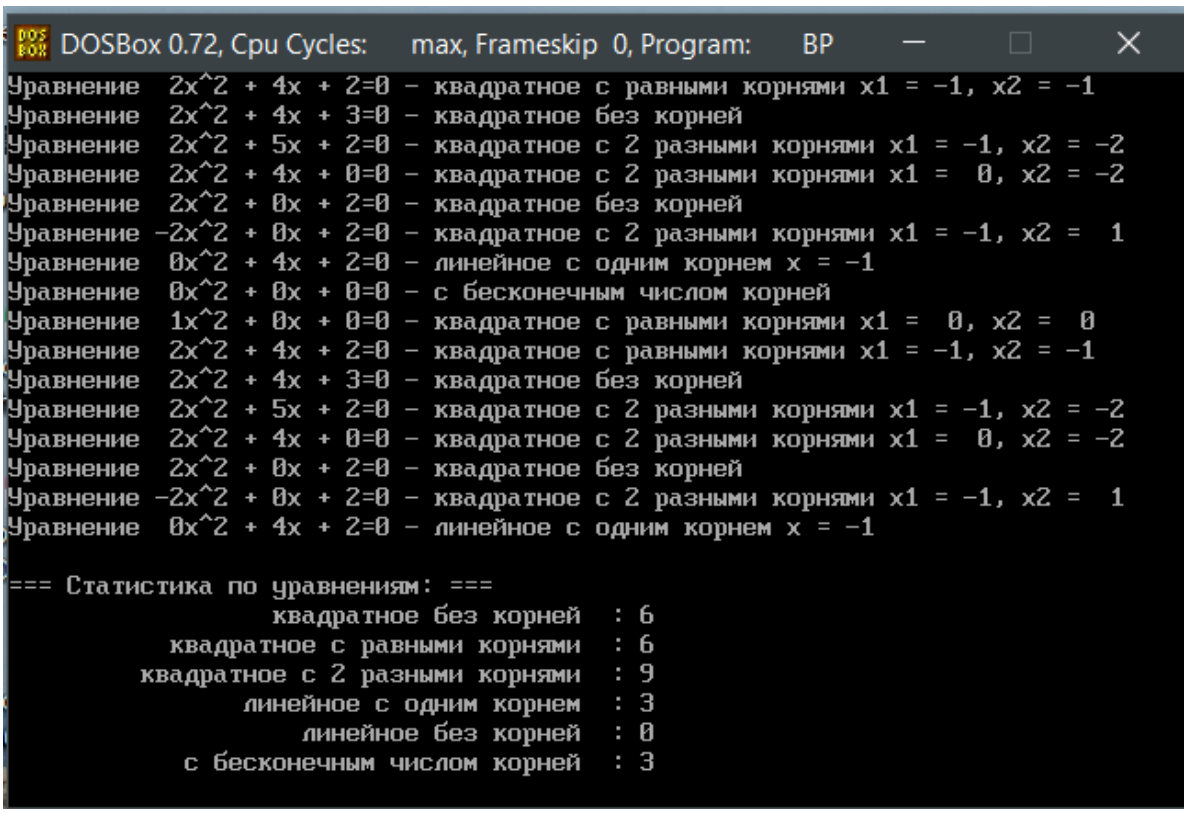
```
program equation;
uses crt;
const cases = 6;
type
 TValues = array [0..2] of real;
 TCodes = array [0..cases-1] of word;
 {тип для хранения возможных вариантов ответов
  при решении уравнения}
function checkEq (var a: TValues): byte;
var
 D: real;
 code: byte;
begin
    if a[2] <> 0 then
      begin
       D := a[1]*a[1] - 4*a[2]*a[0];
       if D>0.0 then code := 2      // 2 разных корня
       else
         if D = 0 then code := 1   // 2 равных корня
                  else code := 0;  // нет вещ. корней
       end
    else // линейное уравнение
       if a[1]<>0 then code := 3    // один корень
       else
         if a[0]<>0 then code := 4   // нет корней
         else code := 5;          // множество корней
 checkEq := code;
end;

function getNumOfRoots (hint: byte): string;
begin
```

```
 case hint of
  0: getNumOfRoots:='квадратное без корней ';
  1: getNumOfRoots:='квадратное с равными корнями ';
  2: getNumOfRoots:='квадратное с разными корнями ';
  3: getNumOfRoots:='линейное с одним корнем ';
  4: getNumOfRoots:='линейное без корней ';
  5: getNumOfRoots:='с бесконечным числом корней ';
 end;
end;

function printSign (var vel: real):string;
var
        s: string;
begin
     if vel<0 then
        begin
         str(abs(vel):2:0, s);
         printSign:= '-'+s;
        end
     else
        begin
         str(vel:2:0, s);
         printSign:= '+'+s;
        end;
 end;

procedure getDetails (var a: TValues;
           var x1, x2: real; var code: byte);
var
       roots : byte;
       info  : string;
       D     : real;
begin
    roots := checkEq(a);
    info  := getNumOfRoots(roots);
    code  := roots;
    write('Уравнение ',a[2]:2:0,'x^2 ',
      printSign(a[1]),'x ',printSign(a[0]),
      '=0 - ',info);
    if roots in [1, 2] then
      begin
       D := a[1]*a[1] - 4*a[2]*a[0];
       x1 := (-a[1]+sqrt(D))/(2.0*a[2]);
       x2 := (-a[1]-sqrt(D))/(2.0*a[2]);
```

```
        write ('x1 = ', x1:2:0, ', x2 = ', x2:2:0);
       end
      else
         if roots=3 then
          begin
            x1 := -a[0]/a[1];
            write ('x = ', x1:2:0)
          end
end;

procedure viewStatistics (const codes: TCodes;
                          cases: byte);
 var i: byte;
 begin
    writeln ('=== Статистика по уравнениям: ===');
 for i:=0 to cases-1 do
      begin
       write (getNumOfRoots(i));
       writeln (' : ', codes[i]);
      end;
  end;

var
  x1, x2   : real;
  coeffs   : TValues;
  code     : byte;
  filepath : string;
  t        : text;
  counters : TCodes;
  i        : byte;
begin
  //clrscr;
  filepath := 'testEquation.txt';
  assign(t, filepath);
  reset(t);
  while not eof(t) do
  begin
   read (t, coeffs[0], coeffs[1], coeffs[2]);
   getDetails(coeffs,x1,x2,code);
   inc(counters[code]);
   writeln;
  end;
```

```
    writeln;
    viewStatistics(counters, cases);
    readln;
end.
```

Результаты тестирования



Задача 3. Известно, что латинский квадрат – это матрица `n×n`, в каждой строке и столбце которой уникальным образом расположены числа от `1` до `n`. Проверьте, является ли некоторая квадратная матрица при `n=9` латинским квадратом. Если нет – выведите номер строки, где было замечено первое несоответствие.

Решение. Выделим вспомогательные подпрограммы, проверяющие принадлежность элементов матрицы указанному диапазону, затем уникальности элементов в каждой строке и каждом столбце и печати матрицы.

Алгоритм

Решение 1. В программе используются следующие процедуры и функции:

1) `function isInRange(const a:TMatr; n:integer; var k:integer): boolean;`

Параметры: a – матрица, n – размерность, k – переменная, в которую будет записан номер строки, если найдено первое несоответствие или значение 0. Под несоответствием понимается обнаружение элемента, лежащего вне диапазона допустимых значений от 1 до n. Функция возвращает true, если ни одной такой величины не было найдено, false – в противном случае. Здесь и далее

```
type
TMatr = array [1..n, 1..n] of word;
```

2) `function isLatin(const a: TMatr; n: integer; var k: integer): boolean;`

Параметры: a – матрица, n – размерность, k – переменная, в которой будет храниться номер первой строки или столбца, где было замечено несоответствие уникальности элементов матрицы. Возвращает false, если критерий латинского квадрата не выполнен, true – в противном случае. Используется вспомогательная функция yeasNo;

3) `function yeasNo(i:integer): boolean;`

Параметр: индекс i, указывающий на проверяемые строку и столбец. Возвращает false, если хотя бы один какой-то элемент строки или столбца встретился более одного раза, true – в противном случае;

4) `procedure init_f(var a:Tmatr; n: integer; name: string);`

Параметры: матрица a – возвращаемое значение, количество ее строк (столбцов) n и строка name, содержащая имя исходного текстового файла;

5) `procedure showmatrix(const a: Tmatr; n: integer);`

Параметры: матрица a и количество ее строк (столбцов) n для последующей печати элементов.

Система тестов

Программа тестируется:

1) на матрице-константе **const**

```
c:Tmatr=( (2,3,5,1,7,9,4,8,6),
          (6,4,8,2,5,3,1,9,7),
          (9,1,7,8,6,4,5,2,3),
          (7,9,2,4,3,8,6,5,1),
          (47,8,6,9,1,5,3,7,2),
          (3,5,1,7,22,6,8,4,9),
          (1,6,9,5,8,2,7,3,4),
          (5,2,3,6,4,7,9,1,8),
          (8,7,4,3,9,1,2,6,6) );
```

Ожидаемый результат: Не латинский квадрат (выход за диапазон в строке 5);

2) матрице из текстового файла latin1.txt:

```
2  3  5  1  7  9  4  8  6
6  4  8  2  5  3  1  9  7
9  1  7  8  6  4  5  2  3
7  9  2  4  3  8  6  5  1
4  8  6  9  1  5  3  7  2
3  5  1  7  2  6  8  4  9
1  6  9  5  8  2  7  3  4
5  2  3  6  4  7  9  1  8
8  7  4  3  9  1  2  6  5
```

Ожидаемые результаты: True;

3) матрице из текстового файла latin2.txt:

```
6  4  8  2  5  3  1  9  7
9  1  7  8  6  4  5  2  3
7  9  2  4  3  8  6  5  1
4  8  6  9  1  5  3  7  2
3  5  1  7  2  6  8  4  9
1  6  9  5  8  2  7  3  4
5  2  3  6  4  5  9  1  8
8  7  4  3  9  1  2  6  5
2  3  5  1  7  9  4  8  6
```

Ожидаемые результаты: FALSE (повторение элемента «5» в строке 6).

Листинг программы

```
program latins1;
uses crt;
const n =9;
type
TMatr = array [1..n, 1..n] of word;
TMas  = array [1..n] of word;
var
     a, b : Tmatr;
     k    : integer;
const
     c:Tmatr=( (2,3,5,1,7,9,4,8,6),
               (6,4,8,2,5,3,1,9,7),
               (9,1,7,8,6,4,5,2,3),
               (7,9,2,4,3,8,6,5,1),
               (47,8,6,9,1,5,3,7,2),
```

```
                (3,5,1,7,22,6,8,4,9),
                (1,6,9,5,8,2,7,3,4),
                (5,2,3,6,4,7,9,1,8),
                (8,7,4,3,9,1,2,6,6) );

procedure showmatrix(const a: Tmatr; n: integer);
var
     i, j: integer;
 begin
   for i := 1 to n  do
     begin
       for j := 1 to n do
          write(a[i,j]:4);
       writeln;
     end;
 end;

procedure init_f(var a:Tmatr; n:integer;
             name:string);
var
     t    : text;
     i, j : integer;
 begin
   assign(t, name);
   reset(t);
   for i := 1 to n do
     for  j := 1 to n do read(t, a[i,j]);
   close(t);
 end;

function isInRange(const a: TMatr; n: integer;
                   var k: integer): boolean;
var
    i, j : integer;
    flag : boolean;
 begin
   flag := true;
   i := 1;
   k := 0;
   while flag and (i <= n) do
     begin
         j := 1;
         while flag and (j <= n) do
           begin
```

```
            flag :=(a[i,j]>0) and (a[i,j] <= n);
            if not flag then k := i;
            j := j + 1;
           end;
        i := i + 1;
    end;
   Latin := flag;
end;

function isLatin(const a: TMatr; n: integer;
            var k: integer): boolean;
  function yeasNo(i:integer):boolean;
    var
        b1, b2 : Tmas;
        j      : integer;
        fl     : Boolean;
    begin
       k:=0;
       for j:=1 to n do
            begin
             b1[j]:=0;
             b2[j]:=0;
            end;
     j := 1;
     fl := true;
     while fl and (j <= n) do
          begin
inc(b1[a[i,j]]);
inc(b2[a[j,i]]);
if (b1[a[i,j]]>1) or (b2[a[j,i]]>1)
then begin
k:=j;
fl:=false;
end;
j := j + 1;
          end;
yeasNo := fl;
end;
var
    i    : integer;
    flag : boolean;
begin
    i:=1;
    k:=0;
```

```
    flag := true;
    while flag and (i <= n) do
        begin
            flag := yeasno(i);
            if not flag then k:=i;
            inc(i);
        end;
   isLatin:=flag;
end;

begin
clrscr;
writeln('---------------------');
init_f(a,n,'latin1.txt');
writeln('==========');
showmatrix(a,n);
if isInRange(a, n, k)
    then  writeln (isLatin(a, n, k))
    else writeln(' не латинский квадрат');
if k>0 then
    writeln('Несоответствие в строке ', k);
readln;
init_f(b,n,'latin2.txt');
writeln('==========');
showmatrix(b,n);
if isInRange(b, n, k)
   then writeln (isLatin(b, n, k))
   else writeln(' не латинский квадрат');
if k>0 then
   writeln('Несоответствие в строке ', k);
readln;
writeln('==========');
showmatrix(c,n);
if isInRange(c, n, k)
   then writeln (isLatin(c, n, k))
   else writeln(' не латинский квадрат');
if k>0
   then writeln('Несоответствие в строке ', k);
readln;
end.
```

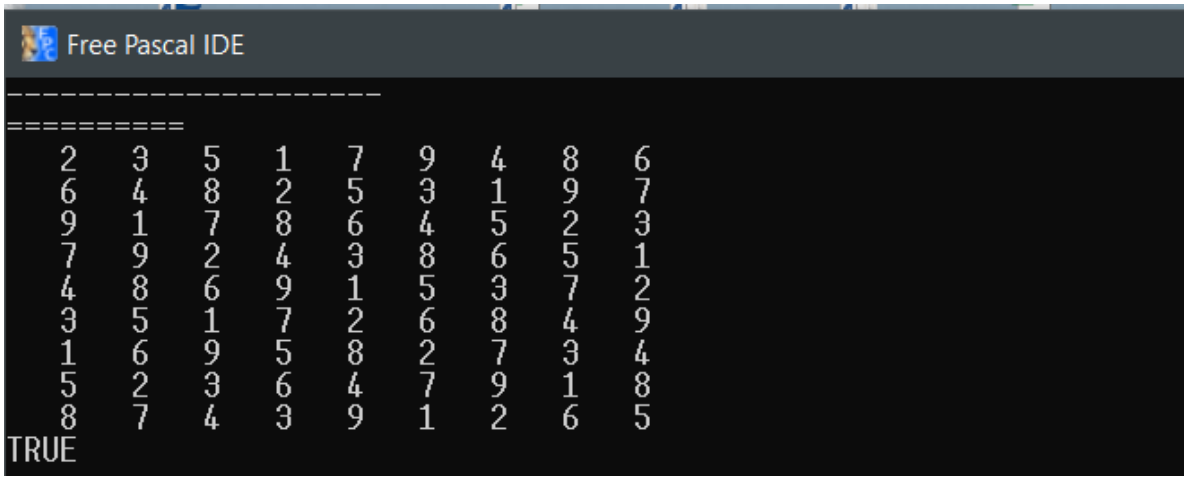
Замечание. Другим подходом к решению задачи является использование множеств. Тип `Set в Pascal` хранит только различные элементы, поэтому изначально пустые множество-строка и множество-столбец

в цикле заполняются элементами матрицы некоторой строки и столбца соответственно. Затем полученные множества сравниваются с эталонным множеством, заведомо содержащим все необходимые элементы. Ответом будет истина или ложь.

Решение 2. Листинг программы аналогичен листингу из решения 1, за исключением реализации подпрограммы `isLatin` и объявления типа `TSet`.

```
const n =9;
type
    TMat = array [1..n, 1..n] of integer;
    TSet = set of 1..n;
…
function isLatin(var a: TMat; n: integer): boolean;
var
    s, s1, s2 : TSet;
    i, j, k   : integer;
    flag      : boolean;
begin
    s := [1..n];     i := 1;    flag := true;
    while flag and (i<= n) do
        begin  s1:=[];  s2:=[];
            for j:=1 to n do
begin
s1 := s1 + [a[i,j]]; s2 := s2 + [a[j,i]];
end;
            flag := (s1=s) and (s2=s);
            inc(i);
        end;
   isLatin:=flag;
end;
```

Результаты тестирования



```
==========
   6   4   8   2   5   3   1   9   7
   9   1   7   8   6   4   5   2   3
   7   9   2   4   3   8   6   5   1
   4   8   6   9   1   5   3   7   2
   3   5   1   7   2   6   8   4   9
   1   6   9   5   8   2   7   3   4
   5   2   3   6   4   5   9   1   8
   8   7   4   3   9   1   2   6   5
   2   3   5   1   7   9   4   8   6
FALSE
stroka 6

==========
   2   3   5   1   7   9   4   8   6
   6   4   8   2   5   3   1   9   7
   9   1   7   8   6   4   5   2   3
   7   9   2   4   3   8   6   5   1
  47   8   6   9   1   5   3   7   2
   3   5   1   7  22   6   8   4   9
   1   6   9   5   8   2   7   3   4
   5   2   3   6   4   7   9   1   8
   8   7   4   3   9   1   2   6   6
 not latinas
stroka 5
```

СТИЛЬ ПРОГРАММИРОВАНИЯ

Стиль программирования – это способ построения программ, основанный на определенных принципах программирования, и выбор подходящего языка, который делает понятными программы, написанные в этом стиле.

Каждый стиль программирования имеет свою концептуальную базу и требует своего умонастроения и способа восприятия решаемой задачи.

Стиль программирования – это выражение опыта общения людей, занимающихся разработкой и использованием программ, выработанного в результате многолетней практики такого общения.

Рассмотрим общие рекомендации.

1. Программа должна быть спроектирована таким образом, чтобы существовала **возможность ее поддержки и расширения** возможностей при возникновении сопутствующих задач.

2. Все имена следует записывать по-английски, избегая транслитерации. Например, `fileName`, а `imyaFayla` не рекомендуется. Комментарии также рекомендуется писать на английском языке, тогда они будут легко переносимы с компьютера на компьютер.

3. **Комментарии** нужны только для человека. Комментарии – важнейший элемент стиля и мощный инструмент, позволяющий сделать программу действительно понятной.

4. Каждое **описание переменной**, смысл которой нельзя угадать по ее названию, должно сопровождаться комментарием, поясняющим не программистскую, а содержательную суть переменной. Например, «количество найденных совпадений» лучше отражает суть, чем «счетчик».

5. **Алгоритм** программы также нуждается в пояснениях. Следует комментировать логику программы.

6. Необходимо учитывать стандартный размер экрана, за которым работает программист. **Длина строк** программы не должна превышать ширины экрана (80 символов).

7. **Модульность.** Пока размер программы составляет несколько десятков строк, она достаточно легко обозревается и ее проще хранить в одном файле. С ростом объема программы просматривать ее становится тяжело. Если программа поделена на отдельные модули, то проще определить, в каком модуле ошибка.

8. **Регистр символов.** Если все имена в программе написаны в одном регистре, такая программа тяжело читается. Необходимо использовать

верхний, нижний и смешанный регистр для визуальной маркировки различных типов идентификаторов.

8.1. ***Зарезервированные слова*** языка Pascal должны быть записаны только маленькими буквами.

Например:

type, var, const, procedure, function, begin, end, if, then, else, repeat, until.

8.2. ***Директивы компилятора*** должны быть записаны буквами в верхнем регистре.

8.3. ***Имена, представляющие типы***, должны быть обязательно написаны в смешанном регистре, начиная с верхнего. К именам типов можно добавлять префикс **T**. Например,

```
type
TMatrix = array [1..10, 1..10] of integer;
```

8.4. ***Имена переменных.*** При решении математических задач желательно не отходить от математической нотации обозначения данных. Другие ***имена переменных*** должны быть записаны в смешанном регистре, начиная с нижнего.

Например, **line, savingsAccount.**

8.5. ***Имена функций и процедур*** должны быть написаны с большой буквы, так как вызов функции без параметров визуально не отличим от простой подстановки переменной. Чтобы избежать путаницы, надо переменные всегда писать с маленькой буквы, а процедуры и функции – с большой.

8.6. Именованные ***константы*** (включая значения перечислений) должны быть записаны в верхнем регистре с нижним подчеркиванием в качестве разделителя.

Например,

MAX_ITERATIONS, COLOR_RED – правильно,

COLOR_red – неправильно.

9. Следует **избегать** использования **глобальных переменных**, область видимости которых – весь файл.

10. В программе все постоянные значения, которые имеют некоторый смысл, следует оформлять как константы.

11. Ожидаемую часть (**наиболее вероятный положительный сценарий**) следует располагать в части **if,** исключение – в части **else**.

12. Необходимо строго избегать исполнимых **выражений в условиях**.

13. Следует **избегать** использования оператора **goto**, также ограничить использование подобных утверждений, таких как **exit**, **break**, **continue**. Часто для этого достаточно просто изменить условие выполнения цикла или использовать цикл другого типа.

14. Рекомендуется всегда писать символ – разделитель операторов «;» сразу же после оператора. Это позволит избежать ошибок при добавлении новых операторов.

15. **Процедуры и функции**.

15.1. Следует ***избегать дублирования*** кода. Код, используемый два и более раза оформлять в виде функции или подпрограммы, дублируемый код в классе выделять в отдельный метод.

15.2. Каждая функция, процедура ***должны решать только одну*** задачу. В этом случае ее будет легко протестировать.

15.3. ***Избегать длинного списка*** аргументов. Приближаясь к числу семь, список аргументов становится не воспринимаемым при чтении. Возможно, следует объединить группы аргументов в новый тип данных.

15.4. Процедура ***не должна быть большой***. Объем процедуры должен быть таким, чтобы ее было несложно охватить как единое целое, понять сразу всю логику ее работы, иначе есть смысл подумать о дальнейшем разбиении алгоритма.

16. **Форматирование кода.** От того, насколько удачно размещен текст программы, существенно зависит легкость ее восприятия.

17. Пишите **программы «лесенкой»**. «Лесенка» должна отражать структурную вложенность языковых конструкций.

Например:

```
sum := 0;
for i := 0 to n do
if not odd(i) then writeln('even')
else
begin
writeln('odd');
sum := sum + i
end;
writeln ('Sum of odd numbers = ', sum);
```

18. Правила написания конструкции **begin ... end**. Даже если оператор один, то предпочтительнее ставить операторные скобки **begin** ... **end**, поскольку в последствии код может измениться и понадобится поставить несколько операторов.

18.1. Необходимо использовать пустые строки для ***отделения логически обособленных блоков*** друг от друга, функции разделять одной пустой строкой.

18.2. Скобки не отделяются пробелами с внутренней стороны. Между функцией и ее аргументами пробел не ставится.

18.3. В списке аргументов перед запятой пробел не ставится, после запятой – ставится. Полезно ставить пробелы при знаках действий.

19. Один оператор – одна строка. Не надо размещать в строке несколько операторов – это затрудняет понимание.

20. Не следует располагать блок ***из нескольких инструкций*** на одной строке после **`if, while`** и т.д.

21. Все блоки описаний **`var`**, **`const`** и т. д. можно использовать не более одного раза.

ЭФФЕКТИВНОСТЬ ПРОГРАММ

Основной задачей программирования является создание правильных и эффективных программ.

Обычно стиль программы более важен, чем ее эффективность, так как хорошо структурированную программу легче править, модифицировать и использовать.

В тех случаях, когда программа или не помещается в памяти, или часто применяется, или долго выполняется, эффективность становится очень весомым фактором.

У каждой программы есть свои *критические части*, которые требуют много времени или памяти. Проверке на эффективность должны сразу подвергаться критические части. Такие фрагменты программы нужно оптимизировать, а потом снова проанализировать программу, чтобы найти другие критические части.

Оптимизировать программу можно на разных этапах ее написания. Рассмотрим некоторые положения.

ОПТИМИЗАЦИЯ ВО ВРЕМЯ КОМПИЛЯЦИИ

Некоторые компиляторы выполняют оптимизацию, которая заключается в уменьшении повторяющихся вычислений.

Программист может облегчить оптимизацию для компилятора, сделав некоторые действия на уровне входного языка.

Рассмотрим примеры:

1) **`x*y`** и **`y*x`** – для компилятора разные выражения;

2) **`2*x`** и **`2.0*x`** – для компилятора разные выражения;

3) **`a:=b*b*c*c`** – лучше записать как **`a:=(b*c)*(b*c);`**

4) **`a:=b-c;`**… **`d:=c-b;`** – лучше записать как **`d:=-(b-c)`**, поскольку оптимизирующему транслятору трудно обнаружить все повторяющиеся выражения.

Программисты должны записывать программу так, чтобы исключить, насколько возможно, повторяющиеся выражения.

Нужно помнить, что компилятор может оптимизировать только линейные участки программы, т. е. те, которые имеют один вход и один выход.

Компилятор может оптимизировать следующее:

```
k:=i/3.0*b;  p:=3+i/3.0*b;  a:=b*b-i/3.0*b;
```

Он не может оптимизировать такой фрагмент, ведь среди операторов появился оператор с меткой:

```
k:=i/3.0*b;  bl : p:=3+i/3.0*b;  a:=b*b-i/3.0*b;
```

Если второй оператор имеет метку, то компилятор не знает, какое влияние окажет переход на этот оператор из другого места. В таком случае программист должен сам позаботиться об оптимизации выражений.

Рассмотрим, как оптимизировать такую часть программы:

```
for i := 1 to 100 do
  if t then x[i] := a[i] + b[i]
       else x[i] := a[i] - b[i];
```

Поскольку переменная `t` внутри цикла не изменяется, но 100 раз выполняется проверка `if`, то лучше записать этот фрагмент иначе:

```
if t then
  for i := 1 to 100 do x[i]:=a[i] + b[i]
else
  for i := 1 to 100 do x[i] := a[i] - b[i];
```

ИНДЕКСАЦИЯ

Обращение по индексу в массиве удобно и наглядно, но в то же время требует дополнительного времени и памяти.

Рассмотрим, какие действия можно использовать, чтобы повысить эффективность программы.

1. Пусть подсчитывается

```
x := (a[i] + 1 / a[i]) * a[i]
```

Лучше сделать так:

```
ai := a[i];
 x := (ai + 1.0 / ai) * ai
```

2. Пусть подсчитывается

```
for i := 1 to n do
 for j := 1 to m do
  begin
   a[i, j] := 0;
   for k := 1 to L do
    a[i, j] := a[i, j] + b[j, k] * c[k, j];
  end
```

Здесь для каждого зафиксированного **`i, j`** нужно подсчитывать местоположение элемента **`a[i, j]`**, если **`k`** изменяется от **`1`** до **`L`**. Однако сумму можно накапливать в простой переменной `t`, а затем присвоить значение элементу **`a[i, j]`**.

Индексация обладает одной особенностью: чем больше индексов используется, тем менее эффективна программа, это значит, что массив **`a[72]`** более эффективен, чем массив **`a[12, 3, 2]`**. Посредством приведения типов можно наложить один массив на другой и работать с более эффективным.

Нужно избегать вычислений со сложной индексацией. Вместо

```
for i := 1 to 10 do
    x[3 * i + 4] := y[3 * i + 4] + c;
```

лучше запрограммировать

```
for i := 1 to 10 do
    begin
    ik := 3 * i + 4;
    x[ik] := y[ik] + c
    end;
```

ИСПОЛЬЗОВАНИЕ ЦИКЛОВ

Память экономится, если мы используем циклы вместо записи команд, которые повторяются. Циклы требуют некоторого дополнительного времени и памяти на инициирование, проверку, изменение параметра цикла и установку всех констант.

1. Нельзя использовать следующий цикл:

```
  p := a[1];
  for i := 1 to 4 do p := p * x + a[i];
```

лучше записать схему Горнера без оператора цикла:

```
p := ((a[1] * x + a[2]) * x + a[3]) * x + a[4];
```

2. Циклы можно объединять. Вместо

```
for i := 1 to 500 do x[i] := 0.0;
for j := 1 to 500 do y[j] := 0.0;
```

лучше написать

```
for i := 1 to 500 do
  begin
   x[i] := 0.0;   y[i] := 0.0
  end;
```

ОРГАНИЗАЦИЯ ЦИКЛОВ

Значительное количество времени затрачивается на инициирование и проверку параметра цикла. Тщательной организацией вложенных циклов можно сэкономить время.

Пример:

```
for k:=1 to 20 do
    {инициализация выполняется 1 раз (k:=1) }
 for j:=1 to 10 do
    {инициализация выполняется 20 раз (j:=1) }
  for l:=1 to 5 do
    {инициализация выполняется 200 раз (l:=1)}
     begin
     {  Тело цикла              }
     { операторы выполняются 20*5*10=1000 раз}
```

```
        end; {{по l} → завершение 1000 раз   }
        {{по j} → завершение 200 раз      }
        {{по k} → завершение 20 раз      }
```

Вывод. Инициализация выполняется 221 раз и завершение – 1220 раз.

Этот пример можно перепрограммировать таким образом:

```
 for l:=1 to 5 do
           {  инициализация 1 раз }
      for j:=1 to 10 do
           {  инициализация 5 раз }
          for k:=1 to 20 do
           {  инициализация 50 раз }
      begin
       { Тело цикла           }
      end;
     { {по k}  завершение 1000 раз = 5*10*20 }
     { {по j}  завершение 50  раз = 5*10  }
     { {по l}  завершение 5  раз      }
```

Вывод. Инициализация выполняется 56 раз и завершение – 1055 раз. Здесь явная экономия.

ОПТИМИЗАЦИЯ ЦИКЛОВ

Существуют некоторые общие приемы оптимизации циклов.

1. Вынесение инвариантных частей из тела цикла. Это те подвыражения, значения которых не изменяются внутри цикла.

2. Замена более продолжительных по времени подсчета операций на более быстрые.

Например,

Исходная операция в цикле	Оптимизированная операция в цикле
`i := i + 1`	`inc(i)`
`j := j – 2`	`dec(j, 2)`
`2 * i + 1`	`k,  k=1, 3, …`
`2 * x`	`x+x`
`x * x * x * x * x`	`sqr(sqr(x))*x`

но **x*x*x*x*x** более оптимально для подсчета x^5, чем **exp(5*ln(x))**

3. Одноразовое вычисление одинаковых подвыражений, или экономия выражений.

Пример. Если в цикле много раз используется выражение, то полезно ввести вспомогательную переменную, присвоить ей значение выражения и далее использовать эту переменную:

(x + 1) … **(x + 1)** … **(x + 1)** → **y := x + 1** и использовать **y**.

4. Исключение действий над константами в теле цикла.

Пример. Константные выражения **`1/4;  exp(-2)/5`** и т. д. лучше подсчитать до входа в цикл.

5. Использование результатов предыдущего шага цикла.

КОНЦЕПЦИИ РАЗРАБОТКИ АЛГОРИТМОВ

Первое, что требуется от алгоритма решения задачи, это правильно реализовать задачу. Второе – нужна такая реализация, которая является легкой для понимания, простой для доказательства правильности алгоритма и удобной для модификации.

Популярной методикой разработки алгоритма решения задачи является *структурное программирование сверху вниз*, когда задача осмысливается в целом, потом она разбивается на подзадачи, которые уже известно, как алгоритмизировать средствами структурного программирования.

Современные языки программирования имеют операторы, которые хорошо отображаются диаграммами Насси – Шнейдермана, или структурограммами. Блок-схемы, подчиненные таким же условиям, называются *структурными блок-схемами*. Они разрабатываются на базе шести структур управления.

Структурная блок-схема – это блок-схема, которая может быть выражена как композиция из шести элементарных схем, или структур управления.

Доказательство правильности алгоритма – это один из самых трудных и утомительных этапов программирования. Поэтому «прозрачность» отдельных действий разрешит легче проследить за правильностью всех действий по решению поставленной задачи.

Пользуясь управляющими конструкциями при разработке алгоритма, легче следить за правильностью его частей, чем доказывать правильность всего алгоритма. Это неизбежно приводит к обязательному требованию – разрабатывать (проектировать) программу так же, как проектируется любое устройство. Наиболее общая тактика разработки программ состоит в *декомпозиции* всей задачи на более простые подзадачи, которые приводят к решению всей задачи. Затем процесс декомпозиции распространяется на подзадачи, и так до тех пор, пока не получатся подзадачи, легко реализуемые на языке программирования.

Декомпозиция задачи – это итерационный процесс, который не исключает обращение назад. При разработке программы может оказаться, что на некотором шаге приняты неудовлетворительные решения или отдельные подзадачи почти не реализуемы на языке программирования. В этом случае, возможно, придется обратиться на несколько шагов назад и пересмотреть декомпозиции снова.

Описанный метод предложен Н. Виртом (1971) и называется методом *пошаговой детализации*.

Важной особенностью всех структур управления является то, что каждая из них имеет один вход и один выход. Значит и вся блок-схема будет иметь это свойство, поэтому мы с начала алгоритма выйдем на конец его.

Для каждой из предложенных конструкций в структурном алгоритмическом языке существуют определенные операторы:

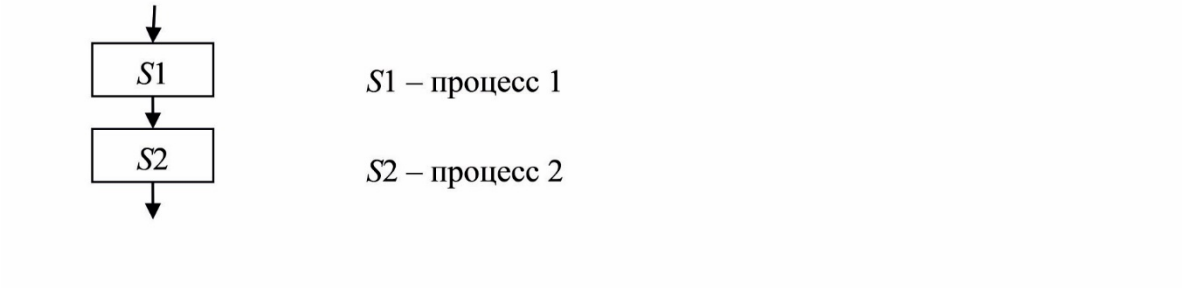
if … then … else,
if … then,
case … of … end,
while … **do,**
repeat … until,
for … to(downto) … do.

Отметим, что существует много языков структурного направления: Algol, Pascal, C, Ada, Modula и другие. Языки программирования Fortran-подобные, например Basic, не структурного направления, хотя Fortran-77 уже имеет элементы структурного программирования.

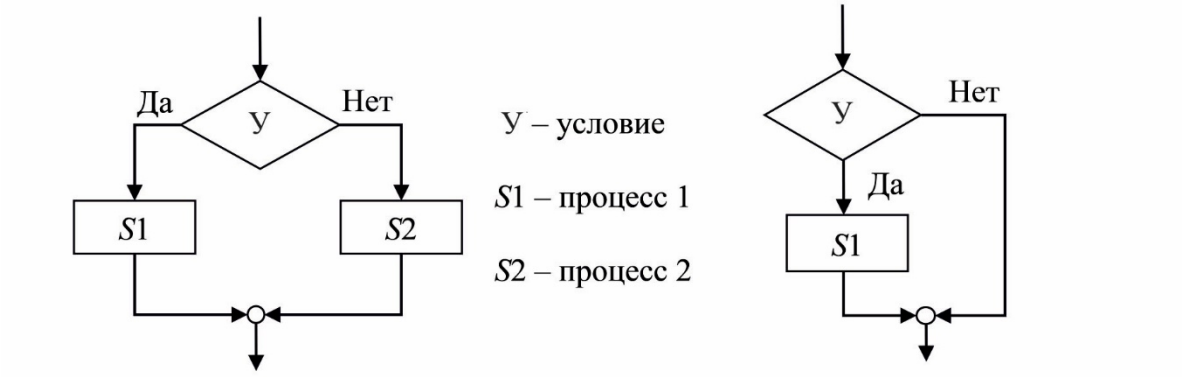
Используя предложенную методику структурного программирования, рассмотрим несколько задач, которые можно назвать базовыми, и *составим* структурные блок-схемы для них.

Основные конструкции структур управления

Следование

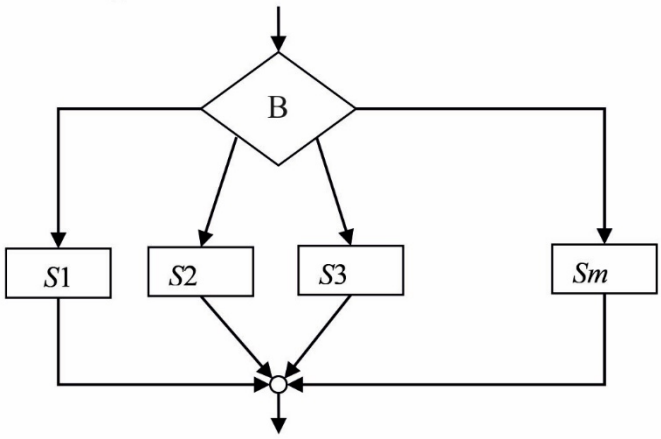


Альтернатива



(оператор **if … then … else**) (оператор **if … then**)

Выбор



B – вариант выбора

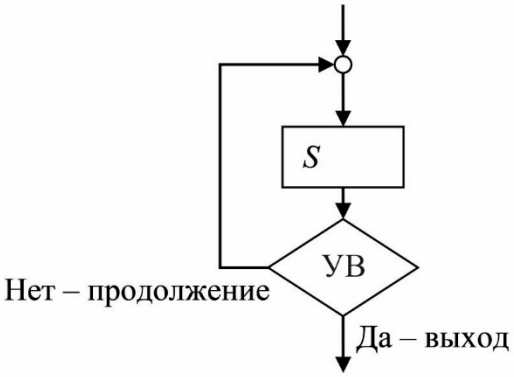
*S*1 – процесс 1

. . .

Sm – процесс *m*

(оператор **`case`** … **`of`** … **`end`**)

Повторение (цикл) с постусловием

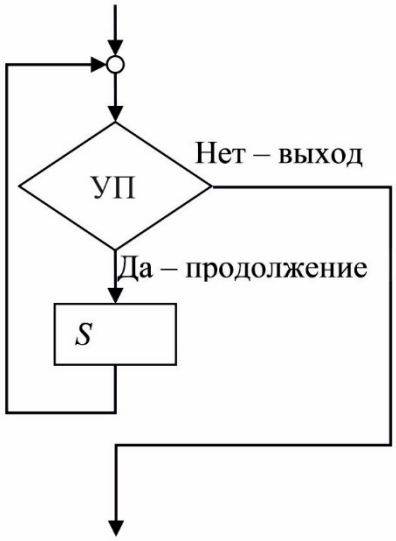


S – процесс

УВ – условие выхода из цикла

(оператор **`repeat … until`**)

Повторение (цикл) с предусловием



УП – условие продолжения цикла

S – процесс

(оператор **`while … do`**)

Цикл с перечислением (параметром)

(оператор **`for … do (downto)`**)

БАЗОВЫЕ АЛГОРИТМЫ

На стадии разработки алгоритмов и программ существенную помощь оказывают методы проектирования:

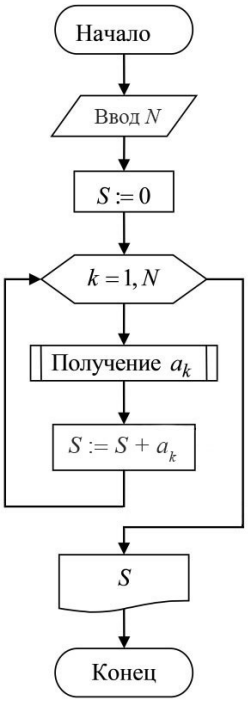
- *проектирование сверху вниз (нисходящее программирование);*
- *модульное программирование*;
- *проектирование снизу вверх* (*восходящее программирование*);
- *структурное кодирование.*

Примеры на прием «пошаговой детализации»

Задача 1. Найти одну из следующих сумм:
1) N первых нечетных чисел;
2) N первых случайных чисел;
3) N первых простых чисел;
4) N чисел a_k, где $a_k = \dfrac{2^k}{k!}$, $k = 1, 2, 3, \ldots, N$.

Решение. Во всех случаях нужно подсчитать $S = \sum_{k=1}^{N} a_k$.

Блок-схема получения суммы чисел следующая.



Однако при решении задачи 1) программа должна сразу дать ответ: N^2 (сумма последовательности чисел 1, 3, 5, …, $2N - 1$ есть $S = \dfrac{2N-1+1}{2} N = N^2$) и не организовывать циклический процесс.

Для решения задач 2), 3), 4) получим единую блок схему, в которой не конкретизировано, как получается слагаемое.

В задаче 2) слагаемые получаются при помощи вспомогательного алгоритма, возвращающего очередное случайное число.

В задаче 3) слагаемые получаются при помощи вспомогательного алгоритма, который возвращает очередное простое число. Такие алгоритмы нужно запрограммировать.

В задаче 4) нужно подумать, как эффективнее получить следующее слагаемое.

Дадим определение массива.

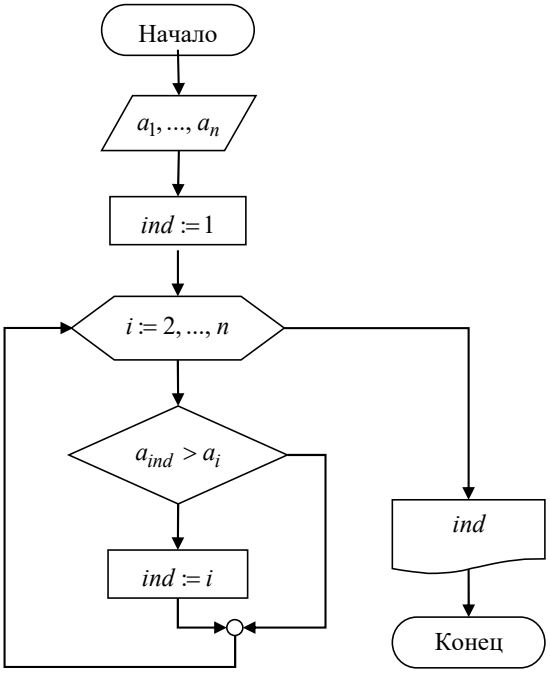
Массивом называется совокупность элементов одного типа, которые имеют одно имя и отличаются в совокупности один от другого своими номерами – индексами. Существуют одномерные массивы (векторы), двумерные массивы (матрицы), трехмерные массивы и так далее. Для описания массива используют служебные слова **`array`** и **`of`**.

Задача 2. Найдите индекс наименьшего элемента в одномерном массиве.

Алгоритм

Запоминаем индекс первого элемента $ind = 1$ и затем организуем цикл на количество оставшихся элементов, в котором сравниваем все очередные элементы с тем, который имеет индекс *ind*. Когда нашелся меньший элемент, то его номер запоминаем в переменной *ind*.

Блок-схема:

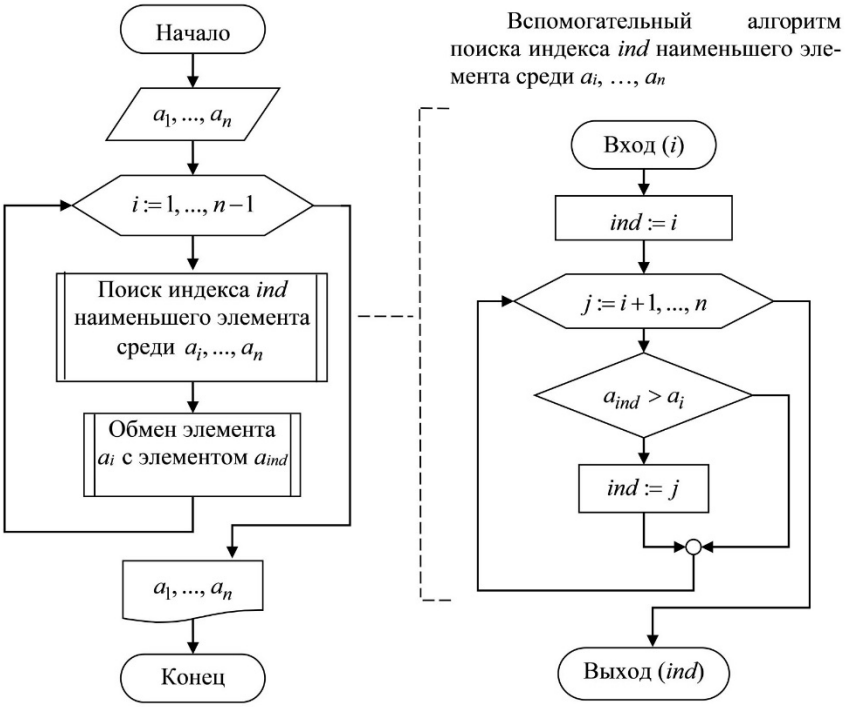


Задача 3. Упорядочите элементы массива по возрастанию.

Алгоритм

При разработке блок-схемы будем опираться на предыдущую задачу. Вызовем вспомогательный алгоритм поиска индекса наименьшего элемента среди a_i, …, a_n. Затем первый наименьший обменяем с a_1, второй наименьший – с a_2 и т. д. Такой алгоритм упорядочения называется *сортировкой методом прямого выбора*.

Блок-схема:



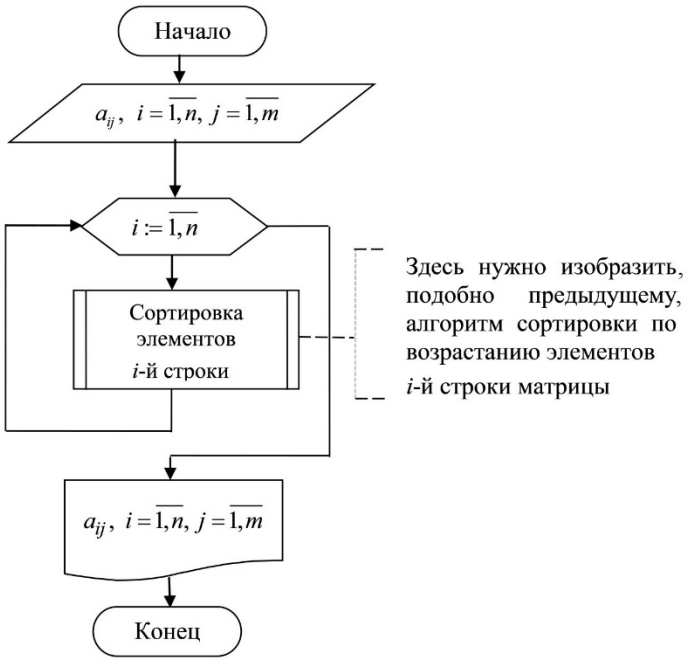
Задача 4. Упорядочите элементы каждой строки матрицы $A_{n,m}$ по возрастанию.

Алгоритм

Используем метод пошаговой детализации.

Так как каждая зафиксированная *i*-я строка матрицы представляет собой одномерный массив a_{i1}, a_{i2}, …, a_{im}, то используем предыдущий алгоритм упорядочения одномерного массива, блок-схему которого изобразим самостоятельно на месте комментария.

Блок-схема:



Задача 5. Найдите сумму $S=\sum_{k=0}^{\infty}\frac{(-2)^k}{k!}=1-\frac{2}{1!}+\frac{4}{2!}-\ldots;$ процесс накопления слагаемых закончите, когда получится слагаемое по модулю меньше чем 10^{-7}.

Решение. Рассмотрим сумму

$S=\sum_{k=0}^{\infty}a_k$, где $a_k=\frac{(-2)^k}{k!}$.

Алгоритм

$S:=0$; вычисляем первое слагаемое; если оно по модулю не меньше 10^{-7}, добавляем к сумме S; вычисляем следующее слагаемое; если оно по модулю больше чем 10^{-7}, добавляем его в сумму S; и т. д., пока очередное вычисленное слагаемое по модулю не станет меньше чем 10^{-7}.

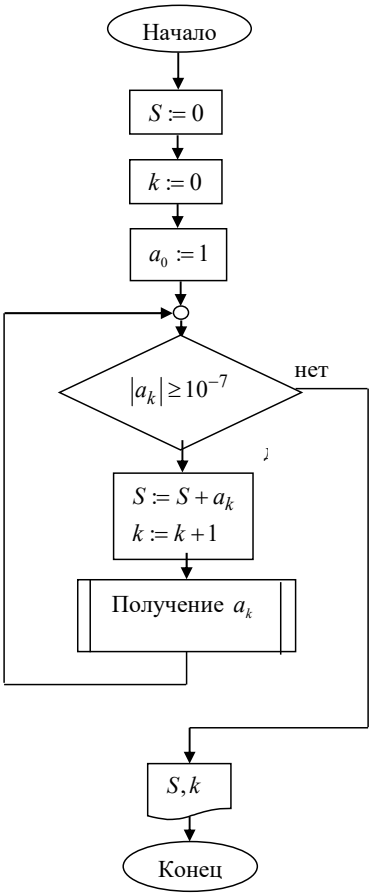
Получить a_k довольно просто.

Так как

$a_k=\frac{(-2)^{k-1}}{(k-1)!}\cdot\frac{(-2)}{k}$, то

$a_k=-2\frac{a_{k-1}}{k}$, где $a_0:=1$.

Блок-схема:



ЗАДАНИЯ

1. ТАБУЛИРОВАНИЕ НЕПРЕРЫВНОЙ ФУНКЦИИ

Постановка задания

Самостоятельно постройте на отрезке $[a;\ b]$ непрерывную функцию $y = f(x)$, которая складывается не менее чем из пяти фрагментов с использованием следующих линий:

- L_1 – линейной функции $y = kx + b$;
- L_2 – квадратичной функции $y = ax^2 + bx + c$;
- L_3 – части дуги окружности радиуса R:

$y = \pm\sqrt{R^2 - (x - x_0)^2} + y_0$;

- L_4 – тригонометрических функций $\alpha\sin(\beta x + \gamma)$ и $\alpha\cos(\beta x + \gamma)$.

Концевые точки отрезков и типы линий на них даны в таблице.

№	$[x_0;\ x_1]$, $x_0 = a$	$[x_1;\ x_2]$	$[x_3;\ x_4]$	$[x_4;\ x_5]$	$[x_5;\ x_6]$, $x_6 = b$
1	$[-6;\ -4]$, L_2	$[-4;\ 1]$, L_4	$[1;\ 4]$, L_1	$[4;\ 7]$, L_3	$[7;\ 9]$, L_2
2	$[-9;\ -5]$, L_4	$[-5;\ -1]$, L_3	$[-1;\ 2]$, L_2	$[2;\ 6]$, L_1	$[6;\ 9]$, L_3
3	$[2;\ 5]$, L_3	$[5;\ 9]$, L_2	$[9;\ 10]$, L_1	$[10;\ 15]$, L_3	$[15;\ 20]$, L_4
4	$[-4;\ 1]$, L_4	$[1;\ 5]$, L_3	$[5;\ 7]$, L_2	$[7;\ 10]$, L_3	$[10;\ 15]$, L_1
5	$[-2;\ 5]$, L_1	$[5;\ 7]$, L_2	$[7;\ 10]$, L_3	$[10;\ 13]$, L_1	$[13;\ 18]$, L_4
6	$[1;\ 6]$, L_4	$[6;\ 9]$, L_2	$[9;\ 10]$, L_1	$[10;\ 16]$, L_3	$[16;\ 19]$, L_1
7	$[-5;\ -1]$, L_4	$[-1;\ 0]$, L_3	$[0;\ 5]$, L_1	$[5;\ 7]$, L_2	$[7;\ 12]$, L_4
8	$[9;\ 15]$, L_4	$[15;\ 19]$, L_3	$[19;\ 24]$, L_2	$[24;\ 26]$, L_1	$[26;\ 33]$, L_3
9	$[12;\ 15]$, L_3	$[15;\ 22]$, L_2	$[22;\ 27]$, L_4	$[27;\ 30]$, L_3	$[30;\ 35]$, L_1
10	$[-9;\ -5]$, L_4	$[-5;\ -1]$, L_2	$[-1;\ 2]$, L_1	$[2;\ 6]$, L_2	$[6;\ 9]$, L_3
11	$[-17;\ -9]$, L_4	$[-9;\ -5]$, L_3	$[-5;\ 7]$, L_2	$[7;\ 12]$, L_3	$[12;\ 15]$, L_1
12	$[-12;\ -5]$, L_1	$[-5;\ -2]$, L_2	$[-2;\ 1]$, L_3	$[1;\ 7]$, L_4	$[7;\ 12]$, L_2
13	$[0;\ 5]$, L_3	$[5;\ 6]$, L_2	$[6;\ 9]$, L_3	$[9;\ 14]$, L_4	$[14;\ 18]$, L_1
14	$[2;\ 5]$, L_1	$[5;\ 9]$, L_2	$[9;\ 10]$, L_3	$[10;\ 15]$, L_4	$[15;\ 20]$, L_3
15	$[-12;\ -7]$, L_2	$[-7;\ -1]$, L_3	$[-1;\ 5]$, L_4	$[5;\ 7]$, L_1	$[7;\ 12]$, L_3
16	$[-9;\ -5]$, L_3	$[-5;\ -1]$, L_4	$[-1;\ 2]$, L_3	$[2;\ 6]$, L_4	$[6;\ 9]$, L_3

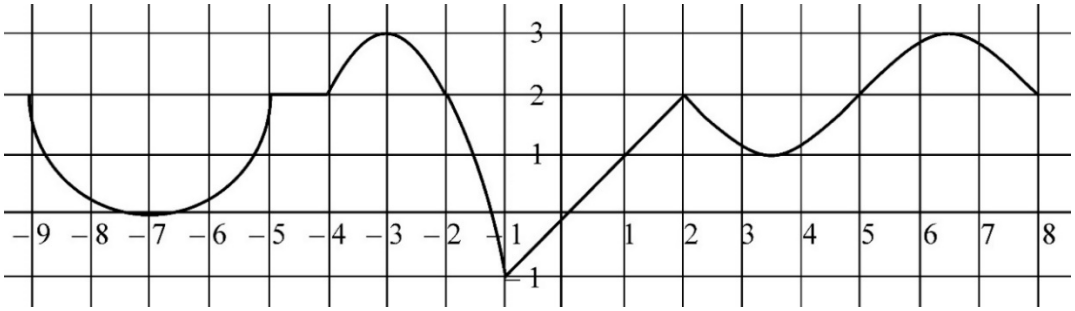
№	$[x_0;\ x_1]$, $x_0 = a$	$[x_1;\ x_2]$	$[x_3;\ x_4]$	$[x_4;\ x_5]$	$[x_5;\ x_6]$, $x_6 = b$
17	$[2;\ 5]$, L_3	$[5;\ 9]$, L_2	$[9;\ 10]$, L_1	$[10;\ 15]$, L_4	$[15;\ 20]$, L_1
18	$[-6;\ -2]$, L_3	$[-2;\ 1]$, L_1	$[1;\ 4]$, L_2	$[4;\ 9]$, L_4	$[9;\ 14]$, L_3
19	$[-9;\ -6]$, L_3	$[-6;\ -4]$, L_1	$[-4;\ -1]$, L_2	$[-1;\ 2]$, L_1	$[2;\ 8]$, L_4
20	$[-12;\ -7]$, L_3	$[-7;\ -1]$, L_2	$[-1;\ 4]$, L_2	$[4;\ 7]$, L_1	$[7;\ 12]$, L_4

Напечатать таблицу значений построенной непрерывной функции на отрезке $[\ a;\ b\ ]$ с шагом h.

Например, для задания

$[-9;\ -5]$, L_3	$[-5;\ -4]$, L_1	$[-4;\ -1]$, L_2	$[-1;\ 2]$, L_1	$[2;\ 8]$, L_4

один из возможных вариантов представления непрерывной функции $y = f(x)$ построен на рисунке.



Аналитически эту функцию можно представить в следующем виде:

$$y = \begin{cases} -\sqrt{4-(x+7)^2}+2, & -9 \le x \le -5, \\ 2, & -5 \le x \le -4, \\ -(x+3)^2+3, & -4 \le x \le -1, \\ x, & -1 \le x \le 2, \\ \sin\dfrac{(x-5)\pi}{3}+2, & 2 \le x \le 8. \end{cases}$$

2. ВЫСТРЕЛ В МИШЕНЬ

Постановка задания

Математическими объектами очерчивается область (мишень) D.

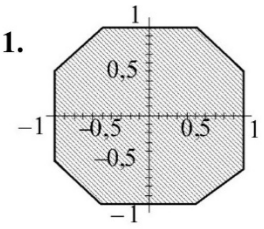
1. Генерируя случайными числами координаты точки $M(x;y)$, подсчитайте в процентах количество попаданий в мишень.

2. Подсчет закончите, когда встретится точка $M(x_0;y_0)$ или осуществится ровно n (n – константа) проб.

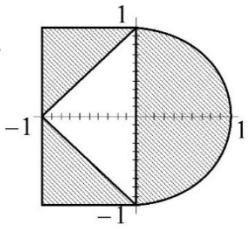
3. Предусмотрите вывод на печать протокола выстрелов.

Варианты задания

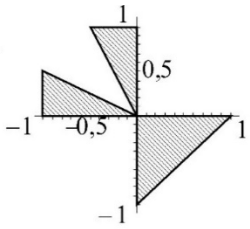
1.



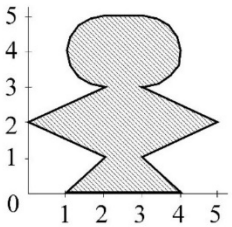
2.



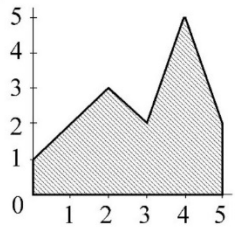
3.



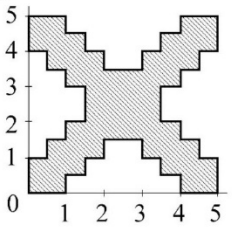
4.



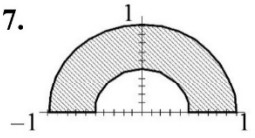
5.



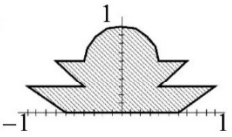
6.



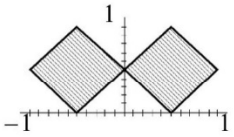
7.



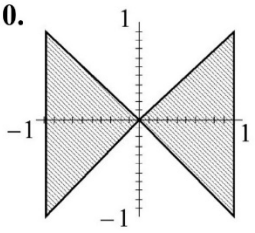
8.



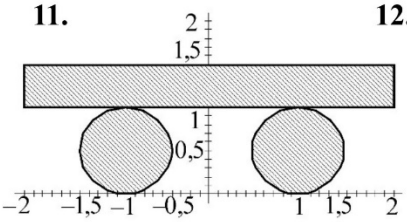
9.



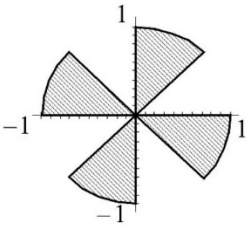
10.



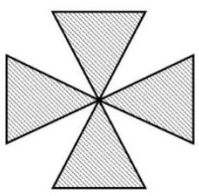
11.



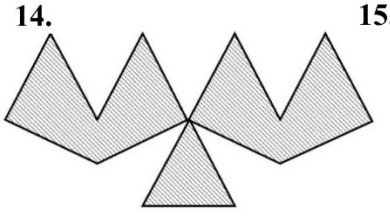
12.



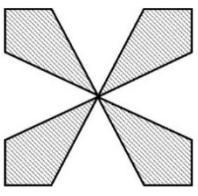
13.



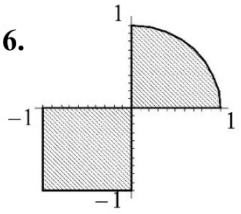
14.



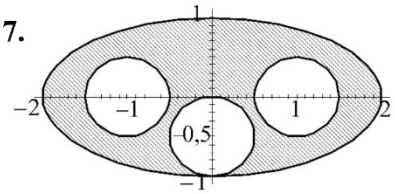
15.



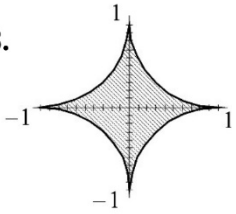
16.



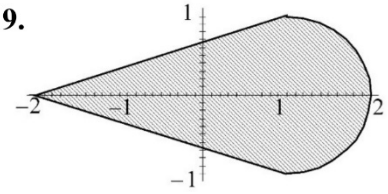
17.



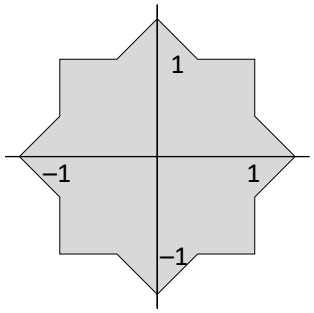
18.



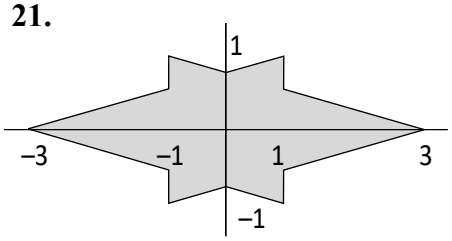
19.



20.



21.



3. ЗАДАЧИ ЦЕЛОЧИСЛЕННОЙ АРИФМЕТИКИ

Постановка задания

Известно, что если целое положительное число n, $n \neq 1$, не делится ни на одно простое число, не превышающее $\sqrt{n}$, то оно простое.

Количество простых чисел не превышает

$$\left[\frac{1{,}6n}{\ln n}+1\right], \text{ когда } n \le 200; \quad \left[\frac{n}{\ln n - 2}+1\right], \text{ когда } n > 200, \qquad (*)$$

где $[x]$ – целая часть числа x.

Для нахождения простых чисел, которые не превышают M, используют следующий алгоритм, полученный из алгоритма «решето Эратосфена».

В массив простых чисел включают сразу $P_1 = 2, P_2 = 3$. Дальше в цикле для переменной k от 3 до M с шагом 2 (по нечетным значениям) проверяют, делится ли без остатка число k на все простые, которые не превосходят $\sqrt{k}$. Если k не делится ни на один из этих простых, то k – очередное простое число. Иначе – переходим к следующему числу k.

Приведем на языке Pascal текст программы, которая реализует этот алгоритм.

```
PROGRAM Prime;
CONST n = ...;  {константу n выбираем, используя формулы (*)}
TYPE
  TMas = array [1..n] of Integer;
VAR
   M, k, s, i, j : Integer;
   Boo           : boolean;
   P             : TMas;
begin
Readln(M);
P[1] := 2;
P[2] := 3;
J := 2;          {номер последнего простого числа}
k := 3;          {   цикл для k:= 3(2)n}
WHILE k < M-1 do
 begin
  k := k + 2;
  s:= trunc(sqrt(k));
  i := 1; {цикл для проверки делимости на полученные простые}
  boo := true;
  WHILE (P[i] <= s) AND boo  do
```

```
      begin
      boo:=(k mod P[i]) <> 0;
      i := i + 1;
      end;
  if boo then
        begin
          j:= j + 1;
          P[j]:= k
        end
 end;
for i := 1 to j do
    Write (P[i]:4);
Writeln;
end.
```

На основании данной программы создайте процедуру, которая вычисляет массив простых чисел и функцию, которая проверяет число на простоту.

Выполните указанный вариант задания.

Вариант 1. Найдите все четверки простых чисел, которые принадлежат одному десятку. Например: 11, 13, 17, 19.

Вариант 2. Дано натуральное число *n*. Разложите его на простые множители и представьте результат в каноническом виде: $n = 2^{k_1} \cdot 3^{k_2} \cdot 5^{k_3} \cdot \ldots \cdot P_s^{k_s},\ k_j \geq 0,\ j = \overline{1,s}.$

Вариант 3. Подсчитайте, сколько среди простых чисел встречается параболических простых чисел. Параболическое простое число *p* имеет вид $p = q^2 + r + 1,$ где *q u r* – простые числа.

Вариант 4. Подсчитайте, сколько среди простых чисел встречается простых чисел Кэрола. Простое число Кэрола имеет вид $p = (2^n - 1)^2 - 2, n \in \mathbb{N} \setminus \{1\}.$

Вариант 5. Подсчитайте, сколько среди простых чисел встречается центральных гептагональных простых чисел. Центральное гептагональное простое число имеет вид $p = (7n^2 - 7n + 2)/2, n \in \mathbb{N}.$

Вариант 6. Подсчитайте, сколько среди простых чисел встречается центральных декагональных простых чисел. Центральное декагональное простое число имеет вид $p = 5(n^2 - n) + 1, n \in \mathbb{N}.$

Вариант 7. Подсчитайте, сколько среди простых чисел встречается центральных квадратичных простых чисел. Центральное квадратичное простое число имеет вид $p = n^2 + (n+1)^2, n \in \mathbb{N}$.

Вариант 8. Подсчитайте, сколько среди простых чисел встречается краниальных простых чисел. Краниальное простое число имеет вид $p = 2^n \pm n, n \in 2\mathbb{N} + 1$.

Вариант 9. Подсчитайте, сколько среди простых чисел встречается центральных триангулярных простых чисел. Центральное триангулярное простое число имеет вид $p = (3n^2 + 3n + 2)/2, n \in \mathbb{N}$.

Вариант 10. Подсчитайте, сколько среди простых чисел p встречаются такие числа, что $(p^2 + 1)/2$ есть квадрат.

Вариант 11. Подсчитайте, сколько среди простых чисел встречается кубических простых чисел типа 1. Кубическое простое число p типа 1 имеет вид $p = (n+1)^3 - n^3,\ n \in \mathbb{N}$.

Вариант 12. Подсчитайте, сколько среди простых чисел встречается кубических простых чисел типа 2. Кубическое простое число p типа 2 имеет вид $p = ((n+2)^3 - n^3)/2, n \in \mathbb{N} \setminus \{1\}$.

Вариант 13. Подсчитайте, сколько среди простых чисел встречается простых чисел Каллена. Простое число Каллена имеет вид $p = n \cdot 2^n + 1, n \in \mathbb{N}$.

Вариант 14. Подсчитайте, сколько среди простых чисел встречается ранних простых чисел. Раннее простое число имеет вид $p = n^2 - 79n + 1601,\ n \in \{0, 1, 2, \ldots, 31\}$.

Вариант 15. Подсчитайте, сколько среди простых чисел встречается квадратных чисел. Квадратное простое число – это простое число, которое представимо в виде $n^2 + 1, n \in \mathbb{N}$.

Вариант 16. Подсчитайте, сколько среди простых чисел встречается четвертичных чисел. Четвертичное простое число – это простое число p, при котором числа $p + 2,\ p + 6,\ p + 8$ являются простыми.

Вариант 17. Подсчитайте, сколько среди натуральных чисел встречается радикальных чисел. Радикальное натуральное число – это число $n \in \mathbb{N}$, при котором число $n^2 + 1$ является простым.

Вариант 18. Подсчитайте, сколько среди простых чисел встречается благополучных чисел. Благополучное (safe) простое число – это такое простое число $p > 2,$ при котором число $(p-1)/2$ является простым.

Вариант 19. Подсчитайте, сколько среди простых чисел встречается стабильных декагональных чисел. Стабильное декагональное простое число – это простое число p с $n > 1,$ при котором число $(p_{n+10} - p_n)/2$ является простым.

Вариант 20. Подсчитайте, сколько среди простых чисел встречается тернарных чисел. Тернарное простое число является суммой последовательных трех простых чисел.

Вариант 21. Подсчитайте, сколько среди простых чисел встречается простых чисел Вагштафа. Простое число Вагштафа (*Wagstaff*) – это простое число $p,$ которое может быть записано в форме $p = (2^q + 1)/3,$ где q – простое число.

Вариант 22. Подсчитайте, сколько среди простых чисел встречается трансферабельных чисел. Трансферабельное простое число – это такое простое число $q,$ для которого имеется $n \in \mathbb{N}, n > 1,$ с условием $q = p_n + 2n.$

Вариант 23. Найдите натуральные числа, в которых цифры числа в какой-нибудь системе счисления с основанием p образуют симметричную последовательность, т. е. они читаются одинаково слева направо и наоборот. Примером может быть число 012210.

Натуральное число называется ***палиндромом***, если его запись читается одинаково слева направо и справа налево.

Вариант 24. Найдите все натуральные k-разрядные числа, цифры в которых в какой-нибудь системе счисления с основанием p образуют строго возрастающую последовательность.

Вариант 25. Получите в порядке возрастания M первых натуральных чисел, которые не делятся ни на какие простые числа, кроме 2, 3 и 5.

Вариант 26. Найдите все числа, которые являются палиндромами, и среди них найдите те, которые при возведении в квадрат также дают палиндром.

Вариант 27. Найдите все совершенные и дружелюбные числа.

Натуральное число называется ***совершенным***, когда оно равно сумме всех своих делителей, включая 1. Например, $28 = 1 + 2 + 4 + 7 + 14.$

Два натуральных числа M и N называются ***дружелюбными***, когда сумма всех делителей каждого из них равна другому числу. Например:

220: $1+2+4+5+10+11+20+22+44+55+110=284$.

284: $1+2+4+71+142=220$.

Вариант 28. Тройку чисел (a, b, c) назовем Героновой тройкой, если эти числа натуральные и площадь треугольника тоже натуральное число. Выведите n Героновых троек.

Вариант 29. Распределите натуральные числа от 1 до M на плоскости по спирали, отмечая звездочками простые числа. Напечатайте картинку, которая содержит звездочки и число 1.

Вариант 30. У многочлена степени n с целыми случайными коэффициентами, взятыми из интервала от $-M$ до M, у которого коэффициент при наибольшей степени равен 1. Найдите целые корни.

Вариант 31. Найдите все натуральные k-разрядные числа Армстронга $(k \geq 2)$. Натуральное k-разрядное число N называется ***числом Армстронга***, когда $N = \sum_{i=1}^{k} a_i^k$, где a_i – цифры числа N. Например, $153 = 1^3 + 5^3 + 3^3$.

Вариант 32. Найдите все числа Мерсенна в интервале от 1 до M. Простое число называется ***числом Мерсенна***, если его можно представить в виде $2^p - 1$, где p – простое число.

Вариант 33. В старояпонском календаре был принят 60-годовой цикл, который состоит из пяти 12-годовых подциклов. Подциклы обозначались названиями цвета: зеленый, красный, желтый, белый, черный. Внутри каждого подцикла годы носили названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, курицы, собаки и свиньи (1984 год – год зеленой крысы – был началом очередного цикла).

По заданному году подсчитайте его название по старояпонскому календарю.

4. ОБРАБОТКА ДИНАМИЧЕСКИХ МАССИВОВ И МНОЖЕСТВА

Постановка задания

Основные этапы работы с массивом – ввод с файла, инициализация случайными числами, вывод на экран, обработка массива по условию задачи − оформить в виде подпрограмм. Массивы должны быть расположены в динамической памяти.

Далее используйте следующие определения.

Параболическое простое число p имеет вид $p = q^2 + r + 1,$ где q *и* r – простые числа.

Простое число Кэрола имеет вид $p = (2^n - 1)^2 - 2, n \in \mathbb{N} \setminus \{1\}$.

Центральное гептагональное простое число имеет вид $p = (7n^2 - 7n + 2)/2, n \in \mathbb{N}.$

Центральное декагональное простое число имеет вид $p = 5(n^2 - n) + 1, n \in \mathbb{N}.$

Центральное квадратичное простое число имеет вид $p = n^2 + (n+1)^2, n \in \mathbb{N}.$

Краниальное простое число имеет вид $p = 2^n \pm n, n \in 2\mathbb{N} + 1.$

Центральное триангулярное простое число имеет вид $p = (3n^2 + 3n + 2)/2, n \in \mathbb{N}.$

Кубическое простое число *p* типа 1 имеет вид $p = (n+1)^3 - n^3,\ n \in \mathbb{N}.$

Кубическое простое число *p* типа 2 имеет вид $p = ((n+2)^3 - n^3)/2, n \in \mathbb{N} \setminus \{1\}$.

Простое число Каллена имеет вид $p = n \cdot 2^n + 1, n \in \mathbb{N}.$

Квадратное простое число – это простое число, которое представимо в виде $n^2 + 1, n \in \mathbb{N}$.

Четвертичное простое число – это такое простое число *p*, что числа $p + 2,\ p + 6,\ p + 8$ являются простыми.

Радикальное натуральное число – это такое число $n \in \mathbb{N},$ что число $n^2 + 1$ является простым.

Благополучное (safe) простое число – это такое простое число $p > 2,$ что число $(p-1)/2$ является простым.

Тернарное простое число является суммой последовательных трех простых чисел.

Простое число Вагштафа (*Wagstaff*) – это простое число *p,* которое может быть записано в форме $p = (2^q + 1)/3,$ где q − простое число.

Раннее простое число имеет вид $p = n^2 - 79n + 1601,\ n \in \{0, 1, 2, \ldots, 31\}$.

Простое число называется *числом Мерсенна*, если его можно представить в виде $2^p - 1$, где p – простое число.

Натуральное число называется *совершенным*, когда оно равно сумме всех своих делителей, включая 1. Например, $28 = 1 + 2 + 4 + 7 + 14$.

Выполните указанный вариант задания.

Вариант 1. По массиву чисел получите подмассив, каждый элемент которого представляет собой параболическое простое число.

Вариант 2. По массиву чисел получите подмассив, каждый элемент которого представляет собой простое число Кэрола.

Вариант 3. По массиву чисел получите подмассив, каждый элемент которого представляет собой центральное гептагональное простое число.

Вариант 4. По массиву чисел получите подмассив, каждый элемент которого представляет собой центральное декагональное простое число.

Вариант 5. По массиву чисел получите подмассив, каждый элемент которого представляет собой центральное квадратичное простое число.

Вариант 6. По массиву чисел получите подмассив, каждый элемент которого представляет собой краниальное простое число.

Вариант 7. По массиву чисел получите подмассив, каждый элемент которого представляет собой центральное триангулярное простое число.

Вариант 8. По массиву чисел получите подмассив, каждый элемент которого представляет собой кубическое простое число типа 1.

Вариант 9. По массиву чисел получите подмассив, каждый элемент которого представляет собой кубическое простое число типа 2.

Вариант 10. По массиву чисел получите подмассив, каждый элемент которого представляет собой простое число Каллена.

Вариант 11. По массиву чисел получите подмассив, каждый элемент которого представляет собой квадратное простое число.

Вариант 12. По массиву чисел получите подмассив, каждый элемент которого представляет собой четвертичное простое число.

Вариант 13. По массиву чисел получите подмассив, каждый элемент которого представляет собой радикальное натуральное число.

Вариант 14. По массиву чисел получите подмассив, каждый элемент которого представляет собой благополучное (safe) простое число.

Вариант 15. По массиву чисел получите подмассив, каждый элемент которого представляет собой простое число Вагштафа (Wagstaff).

Вариант 16. По массиву чисел получите подмассив, каждый элемент которого представляет собой раннее простое число.

Вариант 17. По массиву чисел получите подмассив, каждый элемент которого представляет собой тернарное простое число.

Вариант 18. По массиву чисел получите подмассив, каждый элемент которого представляет собой простое число Мерсенна.

Вариант 19. По массиву чисел получите подмассив, каждый элемент которого представляет собой совершенное число.

Вариант 20. По массиву чисел получите подмассив чисел, порядковые номера которых в исходном массиве есть числа Фибоначчи. Числа Фибоначчи получаются с помощью следующих рекуррентных соотношений: $f_1 = f_2 = 1,\ f_n = f_{n-1} + f_{n-2},\ n = 3, 4, \ldots$.

Вариант 21. По массиву чисел получите подмассив уникальных чисел, оставив из повторяющихся элементов их первое вхождение.

Вариант 22. Заданы два одномерных массива, состоящие из n вещественных элементов.

Проверьте, все ли элементы первого массива содержатся во втором и наоборот.

Вариант 23. Существует массив, составленный из 22 целых чисел интервала от 0 до 66, которые представляют собой условные обозначения костей домино (например, число 42 – обозначение кости домино «4–2», число 33 – кости «3–3» и т. д.).

Определите, соответствует ли последовательность элементов массива ряду костей домино, которые выложены по правилам этой игры.

Вариант 24. Задан одномерный массив, состоящий из n вещественных элементов.

Осуществите, используя вспомогательный массив, циклический сдвиг элементов массива $T(n)$ на m позиций влево так, чтобы получился массив $t_{m+1}, \ldots, t_n, t_1, \ldots, t_m$.

Вариант 25. Задан одномерный массив, состоящий из n целочисленных элементов.

Осуществите, не используя вспомогательный массив, циклический сдвиг элементов массива $T(n)$ на m позиций влево так, чтобы получился массив $t_{m+1}, \ldots, t_n, t_1, \ldots, t_m$.

Вариант 26. Заданы три упорядоченных по возрастанию одномерных массива, состоящие из n целочисленных элементов. Известно, что все элементы массивов различны, кроме одного, который встречается во всех массивах.

Найдите общий для всех массивов элемент за наименьшее число сравнений.

5. ОБРАБОТКА ДИНАМИЧЕСКИХ МАТРИЦ

Постановка задания

Исходная матрица читается из текстового файла. Структура файла следующая: число строк, число столбцов, а затем идут элементы матрицы. Инициируйте ее случайными числами.

Выполните указанный вариант задания.

Вариант 1. Из заданной квадратной матрицы удалите строку и столбец, на пересечении которых находится максимальный по модулю элемент, переместив элементы вверх и влево.

Вариант 2. В заданной прямоугольной матрице найдите характеристики строк. Характеристикой строки матрицы назовем сумму модулей ее отрицательных элементов.

В матрице отсортируйте строки в зависимости от характеристик.

Вариант 3. В заданной прямоугольной матрице найдите характеристики столбцов. Характеристикой столбца матрицы назовем сумму модулей его элементов.

В матрице отсортируйте столбцы в зависимости от характеристик.

Вариант 4. В заданной прямоугольной матрице найдите характеристики строк. Характеристикой строки матрицы назовем сумму квадратов ее элементов.

В матрице отсортируйте строки в зависимости от характеристик.

Вариант 5. В заданной прямоугольной матрице найдите характеристики столбцов. Характеристикой столбца матрицы назовем сумму квадратов его элементов.

В матрице отсортируйте столбцы в зависимости от характеристик.

Вариант 6. Известно, что в каждой строке и каждом столбце квадратной матрицы имеется единственный отрицательный элемент.

Переставьте строки матрицы так, чтобы в полученной матрице отрицательные элементы находились на главной диагонали.

Вариант 7. Известно, что в каждой строке и каждом столбце квадратной матрицы имеется единственный отрицательный элемент. Переставьте строки матрицы так, чтобы в полученной матрице отрицательные элементы находились на побочной диагонали.

Вариант 8. Проверьте, является ли некоторая квадратная матрица магическим квадратом.

Вариант 9. Определите, является ли квадратная матрица ортонормированной, т. е. такой, в которой скалярное произведение каждой пары разных строк равно нулю, а скалярное произведение каждой строки на себя равно единице.

Вариант 10. По заданной матрице постройте матрицу, полученную перестановкой строк (первой с последней, второй с предпоследней и т. д.).

Вариант 11. По заданной матрице постройте матрицу, полученную перестановкой столбцов (первого с последним, второго с предпоследним и т. д.).

Вариант 12. Расположите строки матрицы в порядке убывания модулей сумм элементов строк.

Вариант 13. Задайте функцию, которая подсчитывает величину $x_1x_n + x_2x_{n-1} + \cdots + x_nx_1$, где $x_i -$ максимальный элемент i-й строки матрицы.

Вариант 14. Задана квадратная матрица. Отразите ее зеркально относительно горизонтальной оси симметрии.

Вариант 15. Задана квадратная матрица. Отразите ее зеркально относительно вертикальной оси симметрии.

Вариант 16. Задана квадратная матрица. Отразите ее зеркально относительно главной диагонали.

Вариант 17. Задана квадратная матрица. Отразите ее зеркально относительно побочной диагонали.

Вариант 18. В заданной прямоугольной матрице удалите полностью нулевые строки.

Вариант 19. В заданной прямоугольной матрице удалите полностью нулевые столбцы.

Вариант 20. Проверьте, является ли некоторая квадратная матрица латинским квадратом.

Вариант 21. Постройте квадратную матрицу, являющуюся латинским квадратом.

Вариант 22. Постройте квадратную матрицу, являющуюся магическим квадратом.

6. АРИФМЕТИКА МНОГОКРАТНОЙ ТОЧНОСТИ

Постановка задания

Поскольку диапазон целых чисел, которые можно задать в компьютере, ограничен, то для работы с числами, превышающими максимально допустимые величины, требуется применение специальных приемов.

Например, 13! = 7 005 398 400 > MaxLongint.

Для задания многоразрядного числа можно каждую его цифру хранить в памяти компьютера как элемент массива: самую младшую цифру – в нулевом элементе, следующую – в первом и так далее. Это касается не только 10-й системы счисления, но, например, и 16-й системы счисления. Очевидно, что системы счисления с основанием меньше чем 10-я приведут к массиву цифр большей длины, чем десятичной.

Распечатать такое число – значит напечатать в нужном порядке цифры числа – элементы массива.

Задачи обработки многоразрядных чисел в случае если число есть массив своих цифр, сводятся к элементарным алгоритмам типа «сложение», «вычитание», «умножение». При этом надо учитывать, что сумма (произведение) двух цифр может быть больше, чем наибольшая цифра системы счисления. Тогда необходимо сделать перенос в следующий старший разряд.

Определите нужный тип данных в 10-й системе счисления можно, например, объединяя массив цифр числа и их количество в единую запись:

```
CONST Nmax = 2000;
TYPE Digit = 0..9;
   Chislo = Array[1..Nmax] Of Digit;
   DlChislo = RECORD
          Len : WORD;
          A  : Chislo;
         end;
```

Определим нужный тип данных в 16-й системе счисления, например, так:

```
CONST Nmax = 2000;
TYPE Digit = 0..15;
   Chislo = Array[1..Nmax] Of Digit;
   DlChislo = RECORD
          Len : WORD;
          A  : Chislo;
         end;
```

Различие в этих объявлениях незначительное, но при использовании 16-й системы счисления в представлении данных в случае необходимости печати числа в заданной системе счисления необходимо верно изображать цифры больше чем 9. Этот алгоритм можно найти в лекциях.

В следующих задачах использовать нестандартные типы данных. При помощи соответствующих ресурсов, собранных с модуле, решить поставленные задачи.

Выполните указанный вариант задания.

Указание. В вариантах 1–9 используйте представление данных с использованием 10-й системы счисления, а в варинтах 10–18 – с использованием 16-й системы счисления.

Вариант 1. Найдите все такие *N*, что в числе $N!$ сумма цифр будет квадратом некоторого целого числа.

Вариант 2. Определите вхождение в число $N!$ каждой из цифр, из которых он состоит.

Вариант 3. Определите вхождение в число $N!$ каждой четной цифры.

Вариант 4. Определите вхождение в число $N!$ каждой нечетной цифры.

Вариант 5. Найдите все такие *N*, что в числе $N!$ сумма цифр будет простым числом.

Вариант 6. Найдите все такие *N*, что в числе $N!$ сумма цифр будет делиться на 11.

Вариант 7. Найдите все такие *N*, что в числе $N!$ сумма цифр будет делиться на 15.

Вариант 8. Найдите все такие *N*, что в числе $N!$ сумма цифр будет делиться на 7.

Вариант 9. Найдите $2^n + 3^n + 6^n$ и $5^n + 7^n + 35^n$, $n > 100$.

Вариант 10. Найдите все такие *N*, что в числе $N!$ сумма цифр будет квадратом некоторого целого числа.

Вариант 11. Определите вхождение в число $N!$ каждой из цифр, из которых он состоит.

Вариант 12. Определите вхождение в число $N!$ каждой четной цифры.

Вариант 13. Определите вхождение в число $N!$ каждой нечетной цифры.

Вариант 14. Найдите все такие N, что в числе $N!$ сумма цифр будет простым числом.

Вариант 15. Найдите все такие N, что в числе $N!$ сумма цифр будет делиться на 11.

Вариант 16. Найдите все такие N, что в числе $N!$ сумма цифр будет делиться на 15.

Вариант 17. Найдите все такие N, что в числе $N!$ сумма цифр будет делиться на 7.

Вариант 18. Найдите $2^n + 3^n + 6^n$ и $5^n + 7^n + 35^n$, $n > 100$.

Вариант 19. Докажите, что любое натуральное число n можно единственным способом представить при помощи некоторых целых неотрицательных d_0, d_1, …, d_s в виде $d_s(s+1)! + d_{s-1} \cdot s! + \ldots + d_1 \cdot 2! + d_0 = n$ при условии, что $0 \le d_i < i+1$, $i = 0, \ldots, s$, $d_s \ne 0$.

Найдите соответствующие d_0, …, d_s для заданного многоразрядного числа n. Поиск осуществляется от старшей цифры к младшей.

Вариант 20. Любое целое число n единственным образом может быть представлено в виде суммы $a_s p^s + a_{s-1} p^{s-1} + \ldots + a_1 p + a_0$, где a_i – десятичная цифра, p – основа системы счисления. Получите запись многоразрядного числа n в системе счисления с основанием «–10».

Вариант 21. Найдите два k-значные натуральные числа, последние k цифр квадрата которых совпадают с самими числами.

Вариант 22. Реализуйте три арифметических действия $(+, -, \times)$ над рациональными числами вида p/q, где p и q – многоразрядные числа.

Вариант 23. Реализуйте три арифметических действия над многоразрядными числами с фиксированной точкой (вида $p.q$).

Вариант 24. Посчитайте 2^n или 3^n $(n > 100)$, используя запись числа n в двоичной системе счисления (быстрая степень).

Вариант 25. Посчитайте F_k – k-е число Фибоначчи ($F_0 = 1$, $F_1 = 1$, $F_n = F_{n-2} + F_{n-1}$, n = 2, 3, …), переведите его из десятичной системы счисления в систему счисления с основанием p (p = 2, 8, 16).

Вариант 26. Реализуйте операцию извлечения квадратного корня из n-разрядного числа (n < 40).

Вариант 27. Дробная часть бесконечной десятичной дроби хранится в массиве цифр. Проверьте, не является ли эта дробь периодической. Если это так, превратите ее в правильную дробь вида m / n.

7. РЕКУРСИВНЫЕ АЛГОРИТМЫ

Выполните указанный вариант задания.

Вариант 1. По схеме Горнера вычислите значения многочлена с комплексными коэффициентами

$$(a_n + ib_n)(x+iy)^n + (a_{n-1} + ib_{n-1})(x+iy)^{n-1} + \ldots + (a_0 + ib_0)$$

Вариант 2. Найдите $\min\{a_1, \ldots, a_n\}$, если числа $a_1, \ldots, a_n$ заданы последовательностью своих десятичных цифр, которые хранятся в одномерных массивах.

Вариант 3. Найдите $\sum_{i=1}^{n} a_i$, где a_i – комплексные числа.

Вариант 4. Найдите $\prod_{i=1}^{n} a_i$, где a_i – комплексные числа.

Вариант 5. Найдите наибольшее по модулю комплексное число среди чисел $a_1, \ldots, a_n$.

Вариант 6. Найдите наибольшее рациональное число среди чисел $a_1, \ldots, a_n$.

Вариант 7. Вычислите $n!\ (n \gg 13)$.

Вариант 8. Найдите $2^n\ \ (n \gg 31)$.

Вариант 9. Найдите число Фибоначчи F_N ($N \approx 100$), где $F_0 = 1$, $F_1 = 1$, $F_n = F_{n-2} + F_{n-1}$, $n = 2, 3, \ldots, N$.

Имеет место асимптотическое равенство (для больших k)

$$F_k \approx \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^k.$$

Например, для $k = 100$ получаем $F_k \approx 3{,}54 \cdot 10^{20} >$ longint.

Вариант 10. Для заданных неотрицательных целых чисел n, m подсчитайте $A(n,m)$, где

$$A(n,m) = \begin{cases} m+1, & n = 0; \\ A(n-1,1), & n \neq 0, \quad m = 0; \\ A(n-1, A(n, m-1)), & n > 0, \quad m > 0. \end{cases}$$

Вариант 11. Для заданных неотрицательных целых чисел n, x и y вычислите $A(n,\ x,\ y)$, где

$$A(n,x,y)=\begin{cases} x+1, & n=0; \\ x, & n=1, \quad y=0; \\ 0, & n=2, \quad y=0; \\ 1, & n=3, \quad y=0; \\ 2, & n\geq 4, \quad y=0; \\ A(n-1,\ A(n,\ x,\ y-1),\ x), & n\neq 0, \quad y\neq 0. \end{cases}$$

Вариант 12. Найдите наибольший общий делитель элементов массива A_{20} целых чисел, используя рекурсивную функцию для определения наибольшего общего делителя двух чисел $G(m,\ n)$:

$$G(m,\ n)=\begin{cases} m, & n=0; \\ G(n,\ m\ \bmod\ n), & n\neq 0. \end{cases}$$

Вариант 13. Найдите все числа из заданной последовательности целых чисел, взаимно простые с заданным числом p. Взаимно простые числа – это два числа, для которых НОД (наибольший общий делитель) равен единице. Нахождение НОД оформите как рекурсивную функцию nod, которая возвращает НОД чисел x и y.

Вариант 14. Опишите рекурсивно процедуру, которая переводит целое число из десятичной системы счисления в систему счисления с основанием p.

Вариант 15. По схеме Горнера подсчитайте значение многочлена степени n в точке x.

Вариант 16. Для заданных границ интегрирования a и b вычислите значение интеграла следующего вида:

$$\int x^n\ \cos \alpha x\ dx=\begin{cases} \dfrac{x^n}{\alpha}\sin \alpha x+\dfrac{n}{\alpha}\int x^{n-1}\ \sin \alpha x\ dx, & n\geq 1, \\ \dfrac{\sin \alpha x}{\alpha}, & n=0; \end{cases}$$

$$\int x^n \sin \alpha x\ dx=\begin{cases} -\dfrac{x^n}{\alpha}\cos \alpha x+\dfrac{n}{\alpha}\int x^{n-1}\ \cos \alpha x\ dx, & n\geq 1, \\ -\dfrac{\cos \alpha x}{\alpha}, & n=0. \end{cases}$$

Вариант 17. Для заданных границ интегрирования a и b вычислите значение интеграла следующего вида:

$$\int \cos^n x\ dx = \begin{cases} \dfrac{\cos^{n-1} x \sin x}{n} + \dfrac{n-1}{n}\int \cos^{n-2} x\ dx, & n > 2, \\ \dfrac{x}{2} + \dfrac{1}{4}\sin 2x, & n = 2, \\ \sin x, & n = 1. \end{cases}$$

Вариант 18. Для заданных границ интегрирования a и b вычислите значение интеграла следующего вида:

$$\int \sin^n x\ dx = \begin{cases} -\dfrac{\sin^{n-1} x \cos x}{n} + \dfrac{n-1}{n}\int \sin^{n-2} x\ dx, & n > 2, \\ \dfrac{x}{2} - \dfrac{1}{4}\sin 2x, & n = 2, \\ -\cos x, & n = 1. \end{cases}$$

Вариант 19. Вычислите по заданному вещественному числу x и целому n величину x^n согласно формуле

$$x^n = \begin{cases} x\left(x^{n-1}\right), & n > 0, \\ 1 \Big/ x^{|n|}, & n < 0, \\ 1, & n = 0. \end{cases}$$

Вариант 20. Для заданных границ интегрирования a и b вычислите значение интеграла следующего вида:

$$\int \ln^n x\ dx = \begin{cases} x \cdot \ln^n x - n\int \ln^{n-1} x\ dx, & n > 1, \\ x \cdot \ln x - x, & n = 1. \end{cases}$$

Вариант 21. Для заданных границ интегрирования a и b вычислите значение интеграла следующего вида:

$$\int \mathrm{x}^m \ln^n x\ dx = \begin{cases} \dfrac{\mathrm{x}^{m+1}}{m+1}\ln^n x - \dfrac{n}{m+1}\int \mathrm{x}^m \cdot \ln^{n-1} x\ dx, & n > 1, \\ \mathrm{x}^{m+1}\left(\dfrac{\ln x}{m+1} - \dfrac{1}{(m+1)^2}\right), & n = 1. \end{cases}$$

Вариант 22. Для заданных границ интегрирования a и b вычислите значение интеграла следующего вида:

$$\int \operatorname{tg}^n x\ dx = \begin{cases} \dfrac{\operatorname{tg}^{n-1} x}{n-1} - \int \operatorname{tg}^{n-2} x\ dx, & n \geq 2, \\ -\ln\left|\cos x\right|, & n = 1, \\ x, & n = 0. \end{cases}$$

Вариант 23. Для заданных границ интегрирования a и b вычислите значение интеграла следующего вида:

$$\int \operatorname{ctg}^n x\ dx = \begin{cases} -\dfrac{\operatorname{ctg}^{n-1} x}{n-1} - \int \operatorname{ctg}^{n-2} x\ dx, & n \geq 2, \\ \ln\left|\sin x\right|, & n = 1, \\ x, & n = 0. \end{cases}$$

Вариант 24. Опишите рекурсивно функцию, которая расставляет между заданными n цифрами знаки сложения, вычитания, умножения и деления без остатка так, чтобы после проведения операций получилось заданное число.

8. РАБОТА С ТИПИЗИРОВАННЫМИ ФАЙЛАМИ И МОДУЛИ

Постановка задания

Используйте модульный принцип разработки программ, т. е. подпрограммы.

Все необходимые данные должны передаваться подпрограммам в качестве параметров; все величины, используемые только внутри подпрограмм, должны быть описаны как локальные. Использование глобальных переменных в подпрограммах не допускается.

Задачи данного раздела предусматривают необходимость создания исходного текстового файла, печати его элементов, получения на его основе типизированного файла, печати итогового файла. Предусмотрите возможность пополнения файла, замены данных в файле, удаления данных в файле. Эти ресурсы оформите отдельным модулем.

Для выполнения данного задания по работе с типизированными файлами в главной программе создайте меню.

В зависимости от запроса пользователя в меню выбирается пункт, отвечающий за конкретное действие, например:

- получает указанную в задании информацию по пункту 1;
- получает указанную в задании информацию по пункту 2;
- получает указанную в задании информацию по пункту 3;
- получает указанную в задании информацию по пункту 4;
- выводит на экран информацию и т. п.

Выполните указанный вариант задания.

Вариант 1. Задан файл, содержащий информацию о туристическом агентстве. О каждом туре известно:

- страна;
- город;
- место проживания (отель или дом отдыха);
- характер тура (коммерческий, оздоровительный и т. д.);
- цена путевки;
- количество свободных мест.

Предусмотрите следующие действия:

- выберите коммерческие туры;
- выберите туры отдыха;
- выберите три самых дешевых тура.

Вариант 2. Задан файл, содержащий информацию о картинах галереи. О каждой картине известно:

- фамилия художника;

• год создания картины;
• жанр;
• название.

Предусмотрите следующие действия:

• выберите все картины конкретного художника;
• выберите все картины заданного жанра;
• если картине более 100 лет, то включите ее в список на реставрацию.

Вариант 3. Задан файл, содержащий информацию о видеотехнике, находящейся на складе. О каждом экземпляре известно:

• функциональное название (телевизор, видеомагнитофон, видеокамера и т. п.);
• заводской номер;
• год выпуска;
• страна-производитель.

Предусмотрите следующие действия:

• выберите всю технику заданной страны-производителя;
• выберите всю технику заданного года выпуска;
• выберите все видеомагнитофоны и поставьте их по возрастанию года выпуска.

Вариант 4. Задан файл, содержащий следующую информацию:

• марка автомобиля;
• код ремонта;
• стоимость ремонта;
• данные клиента;
• дата взятия заказа.

Предусмотрите следующие действия:

• выберите автомобили данной марки;
• найдите информацию о ремонте автомобиля по данным клиента;
• выберите автомобили по заданному коду ремонта.

Вариант 5. Задан файл, содержащий информацию о лекарствах в аптеке. О каждом из лекарств известно:

• название;
• страна-производитель;
• дата выпуска;
• срок хранения;
• цена.

Предусмотрите следующие действия:

• выберите лекарства одной страны-производителя;

• если срок хранения истек, то информацию о таких лекарствах удалите из файла;

• выберите самые дорогие и самые дешевые лекарства.

Вариант 6. Задан файл, содержащий информацию о зарегистрированных в ГАИ автомобилях. О каждом из автомобилей известно:

• марка;

• цвет;

• год выпуска;

• государственный номер;

• владелец;

• адрес владельца.

Предусмотрите следующие действия:

• выберите все автомобили заданной марки и цвета;

• найдите среди них те, номер которых соответствует шаблону (шаблон вводится с клавиатуры, например 7971MI);

• выберите все ведомственные автомобили.

Вариант 7. Задан файл, содержащий информацию об автомобилях, зарегистрированных в гаражном кооперативе. О каждом из автомобилей известно:

• государственный номер;

• марка;

• цвет;

• фамилия владельца и его домашний адрес;

• сведения об оплате членских взносов.

Предусмотрите следующие действия:

• выберите все автомобили заданной марки;

• выберите все автомобили заданного цвета;

• если задолженность составляет 3 месяца и более, то подготовьте список таких лиц на исключение из кооператива.

Вариант 8. Задан файл, содержащий информацию о жителях интерната. О каждом из них известно следующее:

• фамилия, имя, отчество;

• год рождения;

• серия и номер паспорта;

• номер комнаты, первая цифра которого обозначает номер этажа.

Предусмотрите следующие действия:

• выберите всех жителей, проживающих на данном этаже;

• выберите всех жителей, проживающих на данном этаже, сгруппировав списки по комнатам;

• выберите всех жителей заданного возраста.

Вариант 9. Задан файл, содержащий информацию об ассортименте кондитерского магазина:

• наименование товара;

• цена за 1 кг;

• дата изготовления, срок хранения;

• фирма-производитель.

Предусмотрите следующие действия:

• выберите всю продукцию, выпускаемую одной фирмой;

• найдите кондитерские изделия, срок хранения которых истекает в данном месяце;

• найдите кондитерские изделия, срок хранения которых достаточен.

Вариант 10. Задан файл, содержащий информацию о результатах соревнований по плаванию:

• фамилия спортсмена;

• название спортивного общества;

• дистанция;

• результат.

Предусмотрите следующие действия:

• выберите спортсменов заданного спортивного общества;

• получите список призеров по каждой дистанции;

• получите список спортсменов, принявших участие в соревнованиях.

Вариант 11. Задан файл, содержащий информацию о результатах соревнований по пулевой стрельбе:

• фамилия спортсмена;

• название спортивного общества;

• вид программы;

• результат.

Предусмотрите следующие действия:

• выберите спортсменов заданного спортивного общества;

• получите список спортсменов, принявших участие в соревнованиях;

• получите список спортивных обществ в порядке возрастания количества спортсменов, которые стали призерами соревнований.

Вариант 12. Задан файл, содержащий информацию об успеваемости студентов:

- фамилия имя, отчество студента;
- курс;
- группа;
- отметки в последнюю сессию.

Предусмотрите следующие действия:

- получите список студентов заданного курса и группы;
- получите список студентов-отличников с указанием курса и группы;
- получите список студентов с указанием курса и группы, которые имеют задолженности.

Вариант 13. Задан файл, содержащий сведения о выпускниках средней школы:

- фамилия, имя, отчество;
- класс;
- год окончания школы;
- места учебы или работы.

Предусмотрите следующие действия:

- получите список выпускников заданного года;
- определите учебные заведения, где учатся выпускники школы;
- определите предприятия, где работают выпускники школы.

Вариант 14. Файл содержит сведения о клиентах ателье проката:

- фамилия клиента;
- название предмета проката;
- инвентарный номер;
- стоимость проката за месяц;
- срок возврата предмета.

Предусмотрите следующие действия:

- найдите три наиболее дорогих предмета проката;
- получите список клиентов, у которых еще не истек срок пользования предметами проката;
- получите список клиентов, у которых истек срок пользования предметами проката.

Вариант 15. В файле находятся сведения об ассортименте хозяйственного магазина:

- название товара;
- фирма-производитель;

• дата изготовления;
• дата поступления в магазин;
• стоимость;
• ассортиментный код.

Предусмотрите следующие действия:

• определите количество ассортиментных единиц по коду;
• получите список всех фирм-производителей;
• определите товары, которые продаются в магазине наиболее продолжительное время.

Вариант 16. Задан файл, содержащий информацию о студентах:

• фамилия, имя, отчество;
• место жительства (общежитие, съемная квартира, дом);
• курс, факультет, отделение, группа;
• оценки, полученные во время последней сессии;
• форма обучения: платная или бесплатная.

Предусмотрите следующие действия:

• выберите студентов, которые учатся на данном факультете, получив полную информацию о них;
• выберите тех, кто учится на данном факультете платно;
• напечатайте список тех, кто живет в общежитии.

Вариант 17. Задан файл, содержащий информацию о студентах:

• фамилия, имя, отчество;
• место жительства (общежитие, съемная квартира, дом);
• номер студенческого билета;
• курс, факультет, отделение, группа;
• оценки, полученные во время последней сессии;
• форма обучения: платная или бесплатная.

Предусмотрите следующие действия:

• выберите студентов, которые учатся в данной группе, получив полную информацию о них;
• выберите тех, кто учится на данном факультете бесплатно;
• напечатайте список тех, кто живет дома.

Вариант 18. Задан файл, содержащий информацию о студентах:

• фамилия, имя, отчество;
• место жительства (общежитие, съемная квартира, дом);
• номер студенческого билета;
• курс, факультет, отделение, группа;

• оценки, полученные во время последней сессии;

• форма обучения: платная или бесплатная.

Предусмотрите следующие действия:

• выберите студентов, которые учатся на данном отделении, получив полную информацию о них;

• выберите тех, кто учится на данном факультете платно;

• напечатайте список тех, кто живет на квартире.

Вариант 19. В файле находятся сведения об ассортименте хозяйственного магазина:

• название товара;

• фирма-производитель;

• дата изготовления;

• дата поступления в магазин;

• стоимость;

• ассортиментный код.

Предусмотрите следующие действия:

• определите количество ассортиментных единиц по коду;

• получите список всех фирм-производителей;

• определите товары, которые находятся в магазине наиболее продолжительное время.

Вариант 20. Задан файл, содержащий информацию о результатах соревнований по плаванию:

• фамилия спортсмена;

• название спортивного общества;

• дистанция;

• результат.

Предусмотрите следующие действия:

• выберите спортсменов заданного спортивного общества;

• получите список призеров по каждому обществу;

• получите список спортсменов, принявших участие в соревнованиях.

9. ФАЙЛЫ

Постановка задания

При выполнении заданий этого раздела продемонстрируйте умение создавать исходные файлы, добавлять информацию в файл, корректировать и печатать существующие файлы. Размер файлов должен быть оптимальным.

В тех случаях, когда файл изменяется (добавляется или корректируется информация), сделайте его копию, проведите обработку файла, а затем удалите копию. При создании копии желательно пользоваться процедурами BlockRead и BlockWrite, которые обеспечивают быстрый обмен информацией и работают с любыми нетипизированными файлами.

При поиске записи в файле предусмотрите ситуацию, когда файл является пустым.

Пользуясь исходными файлами, сделайте корректировку заданных файлов или создайте новые файлы и напечатайте их в виде соответствующей таблицы.

Выполните указанный вариант задания.

Вариант 1. Файл F содержит сведения о шифре научной темы и стоимости ее разработки, а файл G – о шифре темы и ее названии. Создайте файл H, содержащий сведения о шифре, названии и стоимости разработки темы.

Вариант 2. Файл F содержит сведения о шифре и количестве ткани, а файл G – о шифре, названии и цене за один метр ткани. Создайте файл H, который содержит сведения о шифре ткани, ее названии, количестве и общей стоимости.

Вариант 3. Файл F содержит сведения о номере зачетки студента, его фамилии и шифре НИРС (научно-исследовательской работы студента), а файл G – о шифре НИРС и ее названии. Создайте файл H, который содержит сведения о номере зачетки, фамилии студента, шифре и названии НИРС.

Вариант 4. Файл F содержит сведения о номере цеха, табельном номере работника, коде и количестве произведенных им деталей. Пользуясь файлом-справочником H нормативных расценок продукции, создайте файл G, который содержит сведения о номере цеха, табельном номере работника, сумме начисления за работу. Важно учесть, что один работник может производить несколько видов продукции.

Вариант 5. Файл F содержит фамилию и инициалы абонента телефонной станции, его адрес и номер телефона. В файле G собираются сведения о междугородних телефонных разговорах: номер телефона, продолжительность разговора (мин), стоимость минуты разговора (руб.). Напечатайте сообщение на оплату для абонента, номер телефона которого вводится с клавиатуры.

Вариант 6. Файл F содержит сведения о номере зачетки и четырех отметках, полученных студентом во время сессии, а файл G – о номере зачетки, фамилии и инициалах студента. Создайте файл H, который содержит сведения о номере зачетки, фамилии и инициалах студента и его среднем балле.

Вариант 7. Файл F содержит сведения о фамилии и инициалах студента, курсе, группе, отметках, которые студент получил во время двух последних сессий. Создайте файл H, который получает сведения о тех студентах, средний балл которых стал выше.

Примечание. В заданиях 8–13 файл F содержит сведения об отделении, курсе, группе, номере зачетки, отметках, которые студент получил во время последней сессии, файл G – номер зачетки, ФИО студента.

Вариант 8. Пользуясь файлами F и G, напечатайте для выбранной группы список студентов в следующем порядке: сначала отличников, потом тех, кто имеет оценки «отлично» и «хорошо», затем студентов, которые имеют удовлетворительные оценки, и, наконец, тех, кто имеет неудовлетворительные оценки. Список сохраните в соответствующем файле.

Вариант 9. Пользуясь файлами F и G, напечатайте для избранной группы список студентов, которые не имеют неудовлетворительных оценок, в порядке убывания их среднего балла. Список сохраните в соответствующем файле.

Вариант 10. Пользуясь файлами F и G, создайте и напечатайте файл, который упорядочивает общий список студентов по отделениям, курсам и группам, а в каждой группе – по алфавиту.

Вариант 11. Пользуясь файлами F и G, составьте и напечатайте ведомость на получение студентами стипендии по итогам последней сессии, сохранить ее в соответствующем файле.

Вариант 12. Пользуясь файлами F и G, создайте и напечатайте файл H, содержащий сведения об отделении, курсе, группе, номере зачетки, фамилии и инициалах студента, размере его стипендии (если студент не имеет стипендии, то ее размер – 0).

Вариант 13. Скорректировать файл F после пересдачи экзаменов студентами факультета. Для каждого курса сначала напечатайте в алфавитном порядке список студентов, которые пересдали два экзамена, потом список тех, кто пересдал один экзамен. Для этих студентов напечатайте все предыдущие и новые отметки.

СПИСОК ЛИТЕРАТУРЫ

Основной

Аляев, Ю. А. Практикум по алгоритмизации и программированию на языке Pascal : учеб. пособие / Ю. А. Аляев, В. П. Гладков, О. А. Козлов. – М. : Финансы и статистика, 2004.

Долинский, М. С. Алгоритмизация и программирование на Turbo Pascal: от простых до олимпиадных задач / М. С. Долинский. – СПб. : Питер, 2005.

Расолько, Г. А. Теория и практика программирования на языке Pascal / Г. А. Расолько, Ю. А. Кремень. – Минск : Выш. шк., 2022.

Расолько, Г. А. Теория и практика программирования на Pascal / Г. А. Расолько, Ю. А. Кремень. – Минск : Выш. шк., 2015.

Ускова, О. Ф. Программирование алгоритмов обработки данных / О. Ф. Ускова. – СПб. : БХВ-Петербург, 2003.

Дополнительный

Серпинский, В. Что мы знаем и чего не знаем о простых числах / В. Серпинский. – М. ; Л. : Физматгиз, 1963.

Ширяев, В. М. Простые числа и канонические разложения натуральных чисел [Электронный ресурс] / В. М. Ширяев. – Минск : БГУ, 2013. – Режим доступа: https://elib.bsu.by/handle/123456789/107015. – Дата доступа: 24.02.2023.