1. Введение

Целью проведения практических занятий по дисциплине «Программирование для ЭВМ» является закрепление основных теоретических положений, изложенных в лекциях. В соответствии с учебной программой продолжительность практических занятий на 1 курсе – 10 часов.

В процессе работы студенты должны научиться составлять алгоритмы и программы на алгоритмическом языке высокого уровня, используя для решения задач приемы и теоретические положения в соответствии с лекционным материалом. Все примеры, представленные в данном пособии, приведены на алгоритмическом языке Паскаль.

Каждому практическому занятию должна предшествовать лекция по соответствующей теме. Материалы практических занятий выносятся на зачеты и экзамен по дисциплине.

Настоящее пособие представляет собой задачник с кратким справочным материалом, необходимым для решения задач, и примеры их решения.

2. Организационно-методические рекомендации

В соответствии с учебным планом подготовки студентов по направлению 231300 «Прикладная математика» (бакалавриат) и рабочей программой по дисциплине «Программирование для ЭВМ» и изложенными в них требованиями к уровню подготовки инженеров для работы в организациях ГА, студенты должны обладать практическими навыками и компетенциями в решении задач, связанных с использованием различных сценарных языков и технологий.

Данное пособие способствует формированию у студентов компетенций в соответствии с государственным стандартом, поскольку пособие имеет прикладной характер.

2.1. Компетенции обучающегося, формируемые в результате освоения дисциплины «Программирование для ЭВМ»

Студент:

- владеет культурой мышления, способностью к обобщению, анализу, восприятию информации, постановке цели и выбору путей ее достижения (ОК-1);
- способен оформлять, представлять и докладывать результаты выполненной работы (ОК-14);
- способен использовать для решения коммуникативных задач современные технические средства и информационные технологии (ОК-16);
- способен использовать современные прикладные программные средства и осваивать современные технологии программирования (ПК-2);

- способен и готов демонстрировать знания современных языков программирования, операционных систем, офисных приложений, Интернета, способов и механизмов управления данными; принципов организации, состава и схемы работы операционных систем (ПК-5);

- способен самостоятельно изучать новые разделы фундаментальных наук (ПК-14).

В результате выполнения практических заданий по дисциплине «Программирование для ЭВМ» студент должен уметь:

- составлять алгоритмы решаемых задач;
- составлять программы на алгоритмическом языке высокого уровня;
- готовить (формализовать) исходные данные, используемые программами;
- самостоятельно отлаживать программу на персональном компьютере в среде программирования.

2.2. Перечень тем практических и семинарских занятий

1 семестр (6 часов)

ПЗ - 1. Запись арифметических и логических выражений на языке высокого уровня и реализация простейших вычислительных алгоритмов.

ПЗ - 2. Реализация разветвляющихся и циклических алгоритмов и программ арифметической обработки массивов переменных вещественного типа.

ПЗ - 3. Программирование с использованием двумерных арифметических массивов и их обработка.

2 семестр (4 часа)

ПЗ - 4. Реализация программ с использованием процедур и функций.

ПЗ - 5. Программы создания и обработки данных типизированных файлов.

2.3. Содержание занятий

Занятие 1.

Цель занятия:

- освоение правил записи арифметических, логических и смешанных выражений на языке программирования высокого уровня (Паскаль);
- освоение оператора присваивания;
- формирование программ линейной структуры для вычисления выражений различного типа.

Задания (выполняются в соответствии с вариантом):

- записать алгебраическую формулу … в виде арифметического выражения;

- записать логическое выражение, которое будет истинно при условии, что…;
- записать оператор присваивания для вычисления выражения…;
- сформировать схему алгоритма и программу линейной структуры для вычисления выражения… (исходные данные задавать в программе с использованием констант и переменных).

Занятие 2.

Цель занятия:

- освоение правил записи условного оператора;
- освоение циклических алгоритмов с предусловием, в постусловием и цикла с параметром;
- решение простейших циклических задач обработки одномерных арифметических массивов.

Задания (выполняются в соответствии с вариантом):

- записать фрагмент схемы алгоритма и оператор для вычисления заданной переменной с использованием условного оператора;
- записать фрагмент схемы алгоритма и оператор для вычисления суммы, произведения, нахождения минимального, максимального элементов одномерного массива;
- записать фрагмент схемы алгоритма и оператор для вычисления суммы ряда двумя способами.

Занятие 3.

Цель занятия:

- освоение правил записи вложенных циклов;
- овладение навыками работы с двумерными арифметическими массивами.

Задания (выполняются в соответствии с вариантом):

- записать фрагмент схемы алгоритма и фрагмент программы для ввода и вывода двумерного массива;
- записать фрагмент схемы алгоритма и фрагмент программы для вычисления суммы, произведения, нахождения минимального, максимального элементов двумерного арифметического массива (четных/нечетных строк/столбцов двумерного арифметического массива);
- записать фрагмент схемы алгоритма и фрагмент программы для формирования нового массива из элементов двумерного арифметического массива, выбираемых по заданному правилу.

Занятие 4.

Цель занятия - освоение правил:

- записи процедур и функций;
- формирования и использования параметров процедур и функций;

- вызова процедур и функций.

Задания (выполняются в соответствии с вариантом):

- написать схему алгоритма и программу, использующую процедуру для обработки двумерного арифметического массива;
- написать схему алгоритма и программу, использующую процедуру для обработки массива записей;
- написать схему алгоритма и программу, использующую функции для вычисления арифметических скаляров, в том числе при обработке массивов.

Занятие 5.

Цель занятия:

- освоение правил объявления и использования типизированных файлов;
- освоение операций для обработки типизированных файлов;
- применение функций для обработки типизированных файлов.

Задание (выполняется в соответствии с вариантом):

написать схему алгоритма и программу для создания чтения, дополнения типизированного файла и корректировки его записей.

3. Методические указания и теоретический материал

3.1. Арифметические выражения

Арифметические выражения – это аналог алгебраических выражений в математике. Они используются:

- в операторе присваивания;
- в списках данных операторов вывода;
- в качестве фактических параметров процедур и функций;
- в заголовках циклов.

Приоритеты при выполнении арифметических выражений (от высшего к низшему):

- действия в скобках;
- функции (аргумент всегда в скобках);
- одноместные операции (унарные, с одним операндом);
- мультипликативные операции: *, /, div, mod;
- аддитивные операции +, -.

Пример унарной операции: - A.

Мультипликативные – это обычные операции умножения и деления (соответственно * и /), а также целочисленное деление div и остаток от целочисленного деления mod. Операции div и mod отделяются пробелами от операндов, которые всегда целочисленные.

Результат выполнения операций:

7 div 2 = 3, 7 mod 2 = 1.

Наиболее часто используемые функции, которые можно применить к арифметическим данным, приведены в табл. 1.

Таблица 1

Функции для арифметических типов данных

Функция	Результат
Abs (X)	Модуль аргумента
Arctan (X)	Арктангенс (угол в радианах)
Cos (X)	Косинус (угол в радианах)
Dec (X [,n])	Уменьшение X на 1 или на n, если оно есть
Exp (X)	Экспонента
Inc (X [,n])	Увеличение X на 1 или на n, если оно есть
Int (X)	Целая часть числа
Ln (X)	Логарифм натуральный
Pi	Число Pi с точностью до 20 знака
Round (X)	Округляет X до ближайшего целого значения
Sin (X)	Синус (угол в радианах)
Sqr (X)	Квадрат аргумента
Sqrt (X)	Квадратный корень аргумента
Tan (X)	Тангенс (угол в радианах)
Trunc (X)	Целую часть Ч формирует как длинное целое

Функция тангенс появилась только в последних реализациях языка Паскаль, поэтому нужно проверять (например, по «Подсказке») возможность ее использования.

В Паскале нет возведения в степень, поэтому приходится использовать математические преобразования. Так, возведение переменной X в степень Y можно записать как Exp (D * Ln (C)).

Пример

Записать оператор присваивания соответствующий следующей алгебраической формуле (в правой части - арифметическое выражение):

$$X = \sqrt{\frac{ctg(A^4 + 7{,}5 \cdot 10^{-3})}{A + B}} - 2 \cdot \sin^2(\sqrt{B + 0{,}14} - A)$$

X := Sqrt (Cos (Sqr (Sqr (A)) + 7.5E-3)/ Sin (Sqr (Sqr (A)) + 7.5E-3)/ (A+B)) – 2* Sqr (Sin (Sqrt (B+ 0.14) –A));

При необходимости переноса части оператора на другую строку никакие знаки не дублируются. Пробелы можно ставить в любом месте (как и переносы), кроме разделения констант и ключевых слов.

В данном примере используются константы, записанные в двух формах: с фиксированной точкой 0.14 и с плавающей 7.5E-3 (аналог 7.5 *10^{-3}). Во втором случае 7.5 – мантисса числа, а -3 – порядок. В константах с плавающей точкой пробелы не допустимы. Недопустимо также использовать пробел между знаком числа и его значением (в случае отрицательных чисел).

Упражнения

1.

$$Y = A^{-BT}\sin(A+TB) - \sqrt{|AT-B|}$$

2.

$$S = B^2 \bullet ctg^2(X+B)^3 + \frac{A}{\sqrt{\sin(X+B)}}$$

3.

$$R = Z^{2(Y-1)} / B - tg^2(Z+A) + \sqrt{|Z \cdot B / A + \sin(Z)^3|}$$

4.

$$Y = C \bullet ctg^2 X^A \frac{A}{\cos^2(X/A) + 1{,}2 \bullet 10^{-7}} + e^{\sqrt{|5 \cdot A \ln C|}}$$

5.

$$Z = \log_2(C + X^2) + \frac{\sqrt{C+B}}{e^{5-C}}$$

6.

$$Z = \sqrt[5]{a \cdot x \cdot \cos(3a) + e^{4 \cdot x}(x+a)}$$

7.

$$Z = \frac{b^2 X + e^{-X} \sin^5(b \cdot X)}{\ln(A+b) - \log_5(b-x)}$$

8.

$$Y = \frac{1{,}23 \cdot 10^{12} \cdot \log_{10}(X-C)}{\sqrt{arctg(X + \frac{C}{14})}}$$

9.

$$X = arctg \frac{2{,}345 \cdot 10^9}{\sin^2(a \cdot \cos b)} + \ln b - \ln^2 a$$

10.

$$Y = \sqrt{tg(X^2 + \frac{A+B}{X})} - arctgX + \sin^2 X$$

3.2. Логические выражения

Логические выражения – это средства алгебры логики, которые применяются для определения истинности или ложности определенных ситуаций. Строятся из допустимых операндов и логических операций.

Логические операции можно использовать в:

- условных операторах;
- операторах присваивания;
- списках данных оператора вывода данных;
- качестве фактических параметров.

Приоритеты логических операций (от высшего к низшему):

- одноместные операции (унарные, с одним операндом) NOT;
- мультипликативные операции: AND, SHR, SHL;
- аддитивные операции OR, XOR.

NOT - логическое отрицание; дает результат, обратный исходному.

AND – логическое умножение;

OR – логическое сложение;

XOR – исключающее ИЛИ (или сложение по модулю 2).

Результатом логической операции является логическое значение «истина» - True или «ложь» - False.

Логические операции, примененные к логическим операндам, дают результат, приведенный в табл. 2.

Таблица 2

Результат логических операций

A	B	A and B	A or B	A xor B
False	False	False	False	False
False	True	False	True	True
True	False	False	True	True
True	True	True	True	False

Логические операции могут применяться и к целым операндам. В этом случае результат логической операции также целое число. Биты результата формируются из битов операндов поразрядно по тем же правилам, что и в табл. 2.

Например, результаты операций:

7 and 2 = 2,

7 or 2 = 7,

7 xor 2 =5.

Здесь нужно принять во внимание двоичное представление целых чисел:

7 -> 111,

2 -> 010,

5 -> 101.

К логическим операциям в Паскале относят и две сдвиговые операции

SHR – сдвиг вправо (k Shr n – сдвиг числа k вправо на n разрядов);

SHL – сдвиг влево (k Shl n – сдвиг числа k влево на n разрядов).

При этом нужно иметь в виду двоичное представление чисел (левого операнда). Фактически сдвиг влево на 1 разряд является умножением числа на 2, сдвиг вправо на 1 разряд – целочисленным делением на 2.

Например, результаты операций:

7 Shr 2 = 1,

7 Shl 2 = 28.

Здесь представление целых чисел:

7 -> 00111,

1 -> 00001,

28 ->11100.

К логическим выражениям в Паскале иногда относят и выражения сравнения, так как у них логический результат. Операндами выражений сравнения могут быть выражения любого простого типа, типы операндов должны быть совместимы. Используются там же, где и логические. Операции в выражениях сравнения называются операциями отношения и все имеют одинаковый самый низший приоритет.

Операции отношения:

- равно =,
- не равно <>,
- меньше <,
- меньше или равно <=,
- больше >,
- больше или равно >=.

Если необходимо в одном выражении объединить два и больше сравнений, нужно использовать логические операции, при этом выражения сравнения всегда в скобках.

Например, выражение, позволяющее определить, попадает ли переменная X в заданный интервал между A и B, выглядит следующим образом:

(A < X) and (X < B).

Примеры

1. Напишите логическое выражение, которое должно быть верно при условии, что обе переменные A и C ложные или обе истинны, а переменная K кратна 5.

Not (A xor C) and (K mod 5 = 0).

2. Напишите логическое выражение, которое должно быть верно при условии, что хотя бы одна из переменных A и K истинна, а переменная T - четная, либо переменные B и C больше 100, но при этом C больше B.

(A or K) and (T mod 2 = 0) or (B > 100) and (C > 100) and (C>B).

Упражнения

1. Напишите логическое выражение, которое должно быть верно при условии, что истинны переменные A, B и C или переменные K и T равны.

2. Напишите логическое выражение, которое должно быть верно при условии, что переменные A, B и C или все истинны или все ложны.

3. Напишите логическое выражение, которое должно быть верно при условии, что переменные A и B истинны, а сумма C и K больше T.

4. Напишите логическое выражение, которое должно быть верно при условии, что переменные B и C ложны, а сумма переменных K и T не равна 0.

5. Напишите логическое выражение, которое должно быть верно при условии, что переменные B и C ложны, а произведение переменных K и T не меньше A.

6. Напишите логическое выражение, которое должно быть верно при условии, что переменные K и C либо обе ложные, либо обе верные, а сумма переменных A и T больше их произведения.

7. Напишите логическое выражение, которое должно быть верно при условии, что хотя бы одна из переменных A, B и C верна и переменная K больше T.

8. Напишите логическое выражение, которое должно быть верно при условии, что только одна из переменных A, B и C верна и переменная K больше 0.

9. Напишите логическое выражение, которое должно быть верно только при условии, что одна из переменных A, B и C верна, а две другие ложные.

10. Напишите логическое выражение, которое должно быть верно при условии, что только одна из переменных A, X и C ложная или переменная K - четная.

3.3. Оператор присваивания

Оператор присваивания определяет процесс вычисления нового значения с помощью выражения и запоминание полученного выражения в оперативной памяти (ОП) переменной.

Форма оператора присваивания:

A := B;

где : = - знак оператора (пробел между знаками : и = недопустим);

A – имя переменной, в которой запоминается полученное выражение;

B – вычисляемое выражение.

Менять местами правую и левую части КАТЕГОРИЧЕСКИ запрещено.

Результат выражения должен быть совместим с типом переменной A.

Например, в операторе присваивания
A := Not (B xor C) and (K mod 5 = 0);
тип переменной A должен быть логическим, а в операторе
X := Sqrt (Cos (Sqr (Sqr (A)) + 7.5E-3); - вещественным.

3.4. Основные фигуры схем алгоритмов

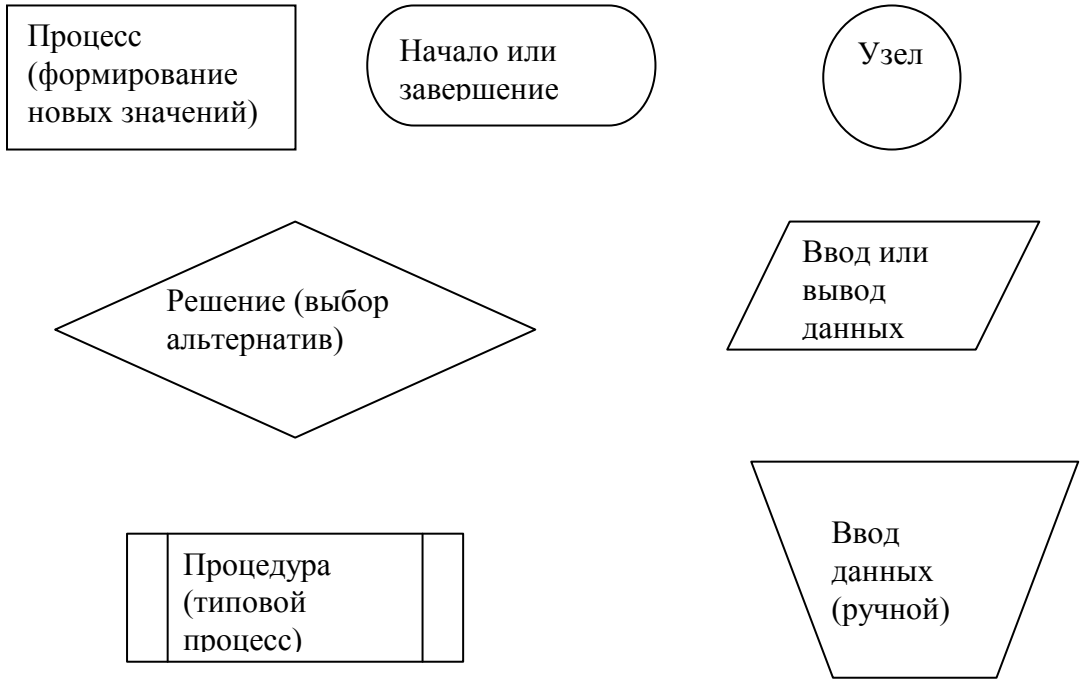


Рис. 1

3.5. Условный оператор

Полная форма условного оператора:

```
If условие    then оператор1
              else оператор2;
```

Пример: если переменная A отрицательна, то переменной X присвоить значение sin (Y), а иначе X присвоить cos (Y).

```
If A<0  then X:= sin (Y)
        else X:= cos (Y);
```

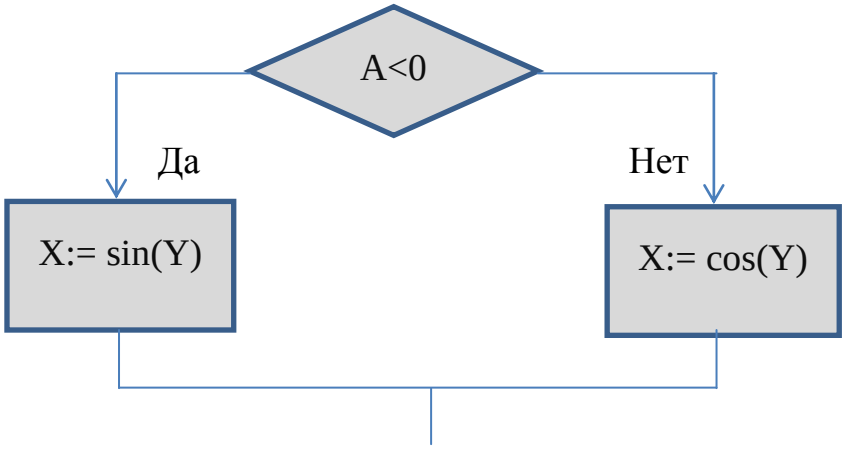


Рис. 2

Сокращенная форма условного оператора:

If условие then оператор;

Пример. Если переменная A отрицательна, то переменной X присвоить противоположное по знаку значение.

If A<0 then X:= -X.

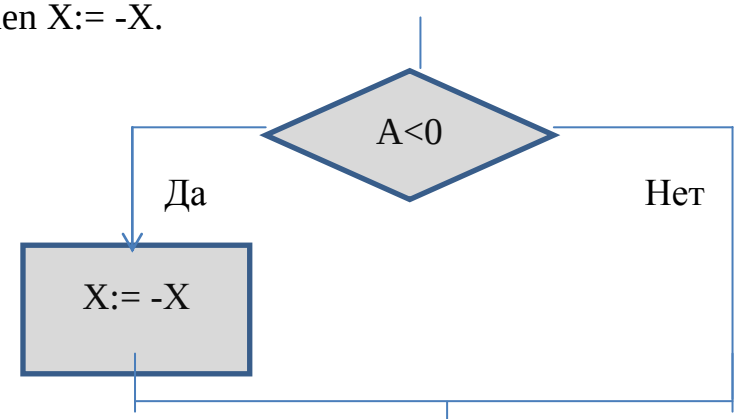


Рис. 3

Вложенные условные операторы:

```
If условие1   then оператор1
              else if условие2   then оператор2
                                 else оператор3;
```

Пример. Если переменная A отрицательна, то переменной X присвоить значение sin (Y), если положительна, то – cos (Y), а если равно 0, то – exp (Y).

```
If A<0  then X:= sin (Y)
        else If A>0  then X:= cos (Y)
                     else X:= exp (Y);
```

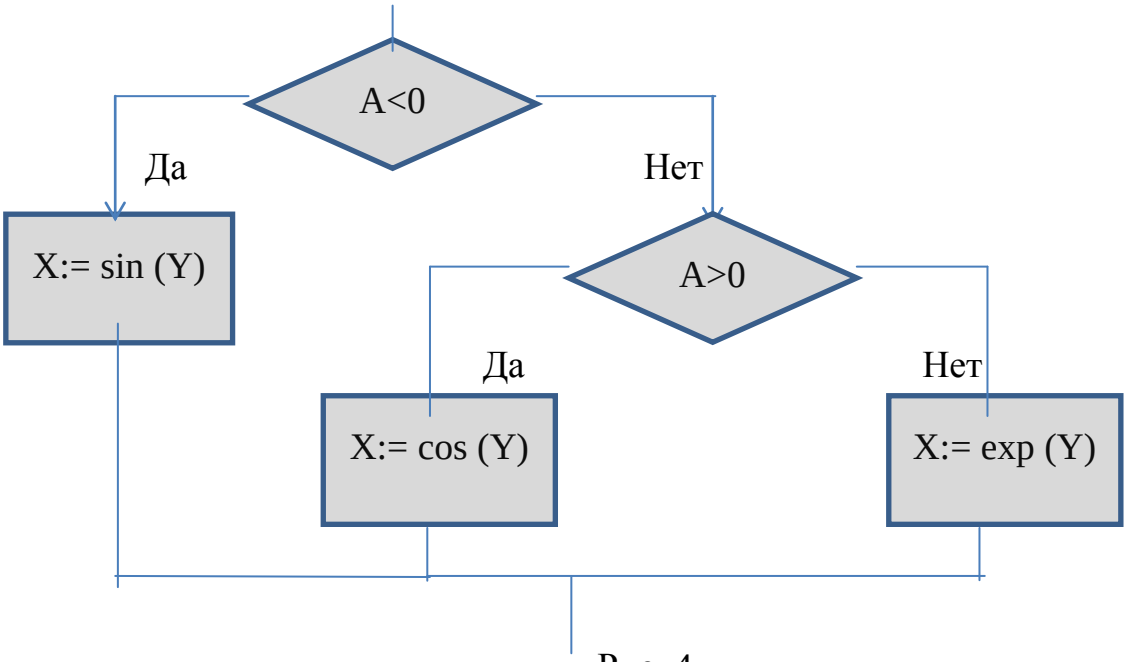


Рис. 4

Если при выполнении (не выполнении) условия необходимо выполнить не один оператор, а несколько, то используется составной оператор (несколько операторов, объединенных операторными скобками).

Begin оператор1; оператор2; …операторN end;

Пример. Если переменная A отрицательна, то переменной X присвоить значение sin (Z), а Y присвоить cos (Z), а иначе - наоборот.

```
If A<0 then begin X:= sin (Z);Y:= cos (Z) end
       else begin X:= cos (Z); Y:= sin (Z) end;
```

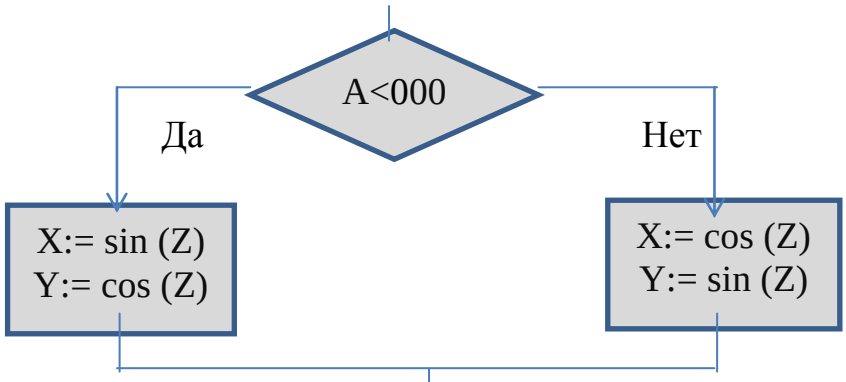


Рис. 5

Упражнения

1. Напишите оператор для вычисления наибольшей и наименьшей переменной из трех переменных A, B и C.

2. Напишите оператор для вычисления значений переменных A и B:

A= sin (x) и B= cos (x), если сумма C и D > 0 и D>C,
A=B=1 - в противном случае.

3. Напишите оператор для вычисления A и C:
A и C истинны, если B ложно, при этом частное от деления C на K>0,
A и C ложны - в противном случае.

4. Напишите оператор для вычисления K:
K = большему из C и D, если A>0 и B>0,
K = меньшему из C и D - в противном случае.

5. Напишите оператор для вычисления переменных B, C, D:
при K>1 и в не равном C исходные B, C и D возвести в квадрат, в противном случае – в третью степень.

6. Напишите оператор для вычисления Y:
Y= X при X<0,
Y = 0 при 0< X < 1,
Y = ln X в остальных случаях.

3.6. Циклы

В Паскале существует три типа циклов:

- цикл с параметром,
- цикл с предусловием,
- цикл с постусловием.

For – цикл с параметром (или с шагом). Используется, если количество повторений цикла определено до его начала.

Форма оператора цикла For:

For i := in to ik do s; или
For i := in downto ik do s;
где:

- i – параметр цикла, идентификатор переменной ординального типа;
- in и ik – начальное и конечное значения параметра цикла;
- to – определяет шаг изменения цикла равным 1;
- downto - определяет шаг изменения цикла равным -1;
- s – оператор (тело цикла), который может быть простым или составным (когда используются операторные скобки).

Схема оператора цикла:

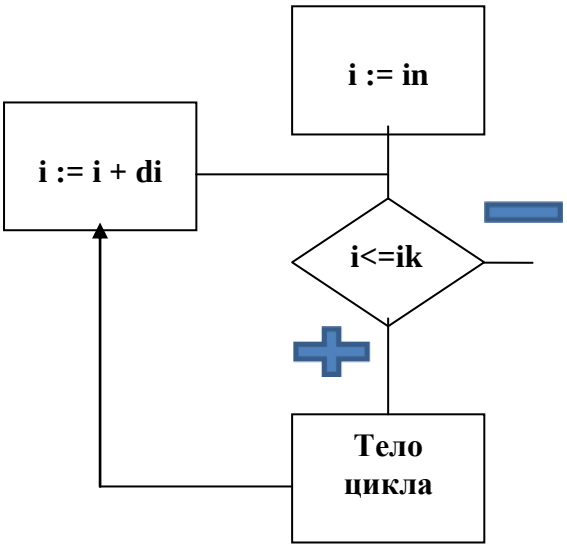


Рис. 6

Например, ввести и напечатать одномерный массив B:

```
For i := 1 to 7 do begin
                    readln  (B[i]);
                    writeln (B[i]);
                end;
```

При этом схема алгоритма будет выглядеть следующим образом:

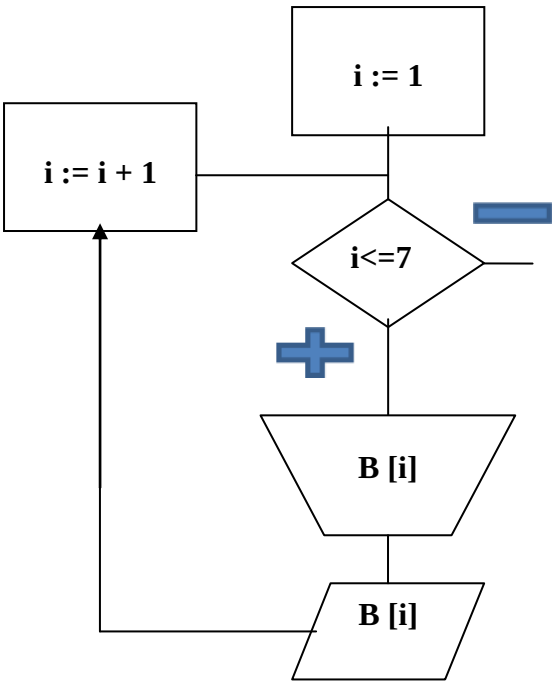


Рис. 7

Важно:

параметр i, а также его начальное и конечное значения не должны изменяться в теле цикла!

Шаг изменения цикла не может по модулю отличаться от 1, поэтому если требуется, чтобы он был иным, необходимо использовать специальный счетчик или использовать его в выражении, формируя нужное значение с помощью математических формул.

Например:

Найти сумму нечетных элементов массива B.

Решение задачи можно выполнить двумя способами:

```
s:=0; k:=1;
For i := 1 to 4  do    begin
                       s := s + B[k]; k := k + 2;
                       end;
```

или

```
s:=0; For i := 1 to 4  do  s:=s+B [i*2 -1];
```

Схемы алгоритма аналогичны схеме, представленной на рис. 7.

На первый взгляд второе решение предпочтительнее (короче), но иногда, в зависимости от задачи, первый подход (через дополнительную переменную) является единственно возможным.

Важно:

- в первом случае параметр цикла в теле цикла не используется, что является совершенно нормальной ситуацией;
- во втором случае индексом массива является выражение, которое должно быть ординального типа (если приходится использовать деление, то операцию div или округляющие функции);
- и в первом, и во втором случае количество повторений цикла в заголовке уменьшено вдвое (4 вместо 7); в любом случае необходимо следить, чтобы значение индекса массива не превышало количество объявленных элементов в массиве.

  В общем случае конечное значение параметра цикла можно определить по формуле: (ik +1) div 2, причем единица нужна для нечетных значений.

Упражнения

Дан одномерный массив A, в котором N элементов.

Требуется:

- ввести исходный массив из текстового файла;
- вывести исходный массив в текстовый файл.

Найти:

- найти сумму и произведение элементов исходного массива;
- найти сумму и произведение отрицательных элементов исходного массива и их количество;
- минимальный и максимальный элемент исходного массива;
- минимальный и максимальный элемент исходного массива и его номер.

Цикл с предусловием While используется, когда количество повторений операторов цикла заранее неизвестно и определяется в процессе выполнения цикла. При определенных условиях операторы тела цикла могут не выполниться ни разу.

Схема оператора цикла:

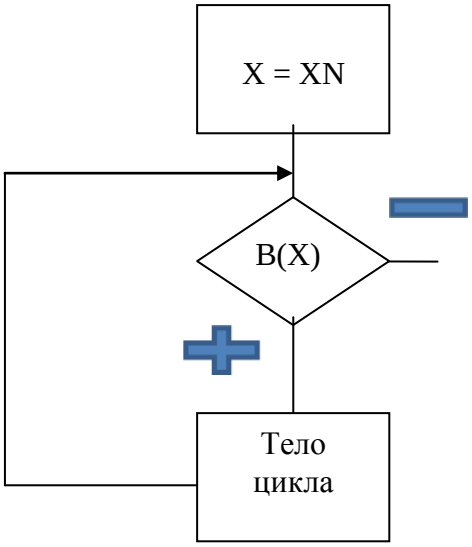


Рис. 8

Форма оператора цикла While:

While B (X) do S;

где:

- B (X) – логическое выражение – условие выполнения цикла;
- s – оператор (тело цикла), который может быть простым или составным (когда используются операторные скобки).

Условие выполнения цикла должно быть определено до начала цикла, а в теле цикла должен быть механизм, приводящий к достижению этого условия (один или несколько операторов). Иначе программа зациклится.

Цикл с постусловием Repeat также используется, когда количество повторений операторов цикла заранее неизвестно и определяется в процессе выполнения цикла. Но операторы тела цикла должны выполниться хотя бы

один раз, так как проверка условия окончания цикла осуществляется после выполнения операторов цикла.

Форма оператора цикла Repeat:

Repeat S

Until B (X);

- B (X) – логическое выражение – условие завершения цикла;
- s – операторы цикла, которых может быть много (операторные скобки не требуются).

Схема оператора цикла:

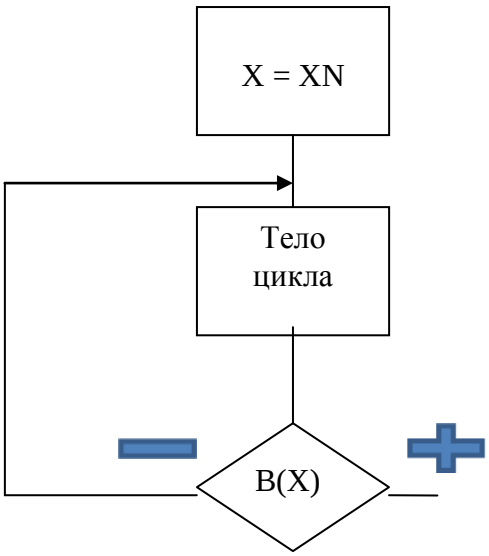


Рис. 9

В этом случае условие завершения цикла может быть не определено до начала цикла, а формироваться уже в его теле. Однако все подготовительные операции предварительно должны быть завершены (переменные определены).

Важно:

Цикл While выполняется, пока истинно условие B (X), а оператор цикла Repeat – пока условие ложно, т.е. ДО ТЕХ ПОР, ПОКА УСЛОВИЕ НЕ СТАНЕТ ИСТИННО.

Вложенные циклы могут быть любых типов. Каждый внутренний цикл должен быть полностью вложен во все внешние (пересечения не допускаются). Рассмотрим вложенные циклы For с уровнем вложенности 2, которые чаще всего используются для обработки двумерных массивов на примере ввода и вывода двумерного массива A, содержащего 4 строки и 5 столбцов.

```
For i := 1 to 4 do begin
        For j := 1 to 5 do begin
```

```
                Read (A[I, J] );
                Write (A[I, J] );
                end;
        Writeln; end;
```

В данном случае begin и end во внешнем цикле необходимы для обеспечения перехода на другую строку при выводе данных (writeln). Иначе их можно было бы опустить. В простейшем случае (только для ввода) вложенные циклы могут выглядеть следующим образом:

```
For i := 1 to 4 do
        For j := 1 to 5 do Read (A[I, J] );
```

Схема вложенных циклов:

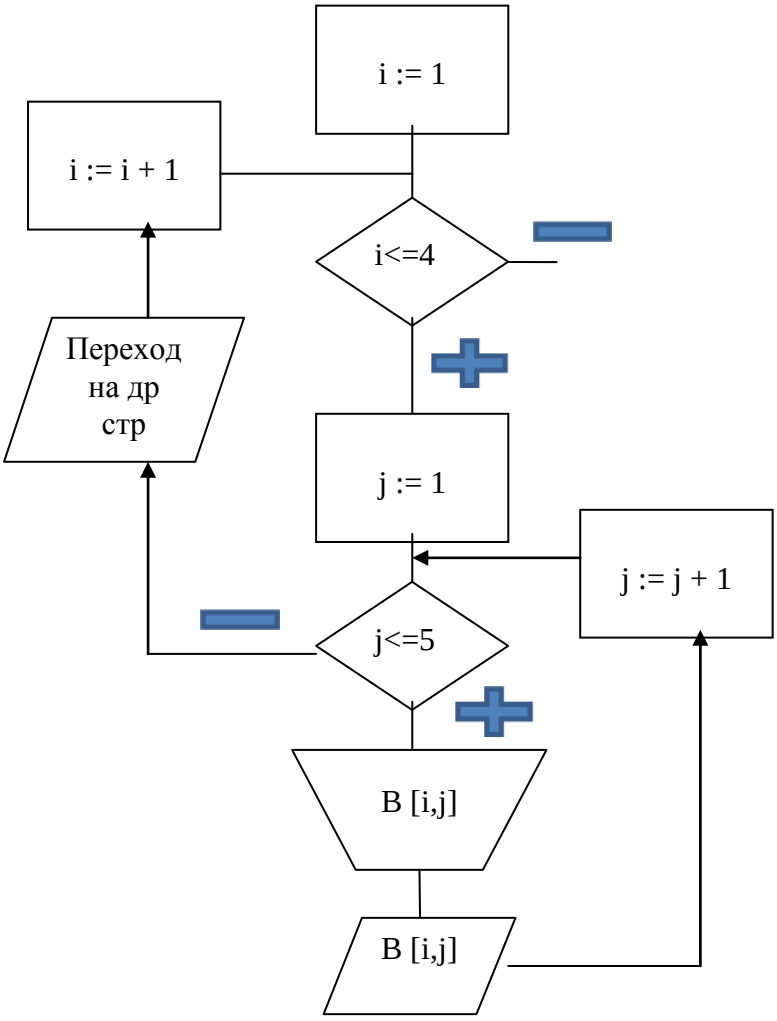


Рис. 10

Пример

Дан двумерный массив A, в котором 4 строки и 5 столбцов.

Написать фрагмент схемы алгоритма и программу, позволяющую сформировать массив B из сумм и максимальных положительных элементов каждого столбца массива A.

```
For j := 1 to 5 do begin
        S := 0; MX:= A [1,j];
        For i := 1 to 4 do
              If A [i,j] > 0 then begin
                          S := S + A [i,j];
                          If A [i,j] > MX then MX :=(A[I, J] );
                          end;
        B[j,1] := S; B[j,2] := MX; end;
```

В данном случае в новом массиве будет 5 строк (по числу столбцов массива A) и 2 столбца (в первом – сумма, во втором – максимальное значение).

Схема алгоритма представлена на рис. 11.

В данном случае оператор MX:= A [1,j] необходим для определения переменной, с которой потом будет происходить сравнение в цикле (предполагаем, что минимальным будет первый элемент в j-м столбце). Следует обратить внимание на следующий момент: если необходимо найти максимальный элемент среди отрицательных, то такое начальное значение корректным не будет, так как в случае, если первый элемент будет положительным или даже нулем, то он и будет максимальным. Поэтому необходимо либо производить проверку на отрицательность и присваивать MX первый отрицательный элемент в столбце, либо (что проще) присваивать ему очень маленькое значение (например, MX:= -1E30), которое заведомо не будет максимальным и на первом же шаге изменится. Если же значение не изменится, то это означает, что отрицательных элементов в столбце нет. Это значение можно оставить или, в зависимости от условия задачи, заменить нулем.

Рассмотрим на схеме алгоритма три области.

1. Тело цикла по j (внешний цикл), но цикл по i еще не обозначен (внутренний цикл). В этом месте находятся операторы S := 0;
   и MX:= A [1,j]; Использовать в операторах переменную i нельзя.
2. Тело цикла по i, вложенного в цикл по j; здесь можно использовать в операторах обе переменные – параметры обоих циклов.
3. Цикл по i закончен, но по j еще нет, следовательно, обработку столбцов можно продолжать, но не используя номера строк.

Для тренировки попробуйте сделать аналогичную «перевернутую» задачу: написать фрагмент схемы алгоритма и программу, позволяющую сформировать массив B из сумм и максимальных положительных элементов каждой строки массива A.

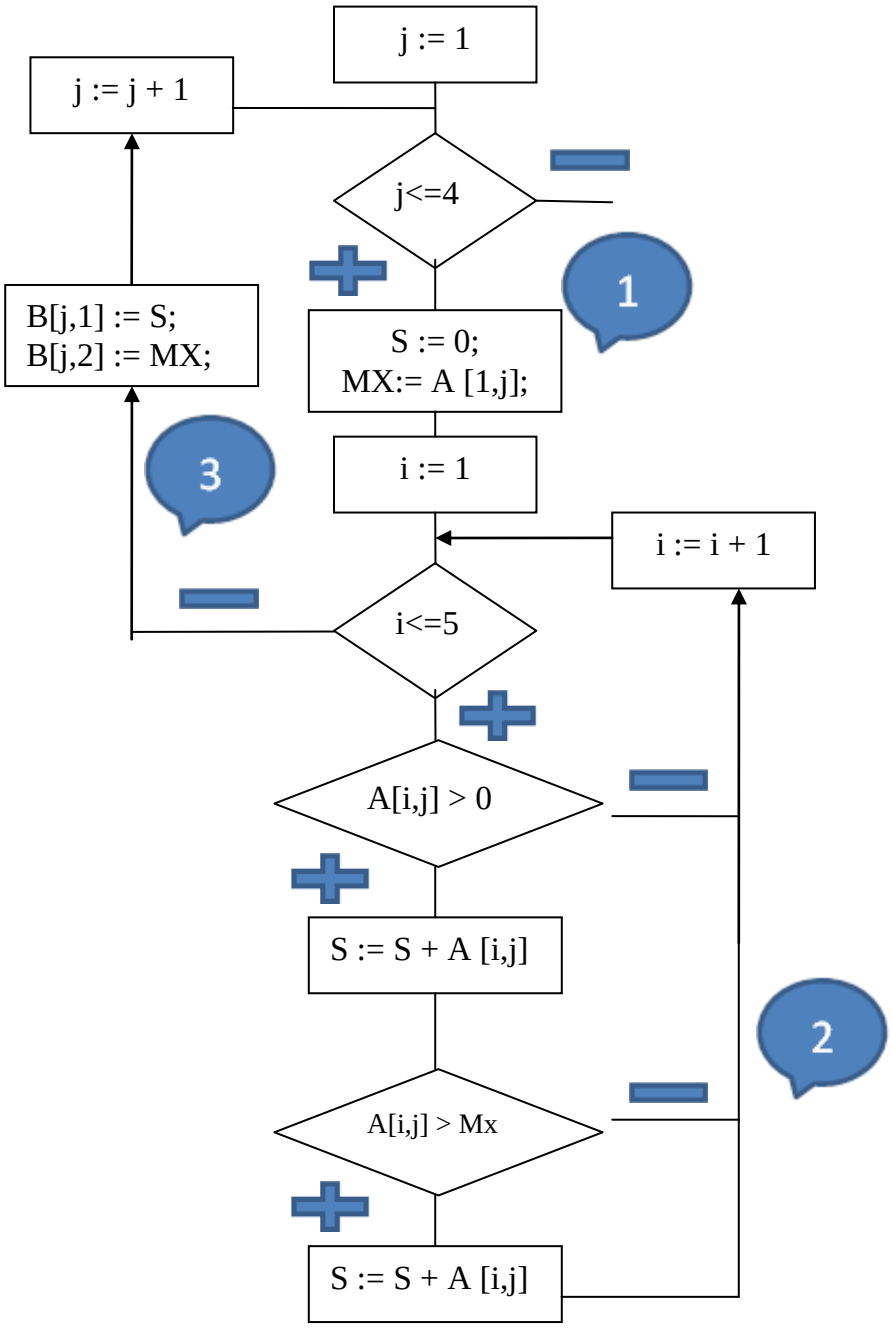


Рис. 11

Упражнения

Дан двумерный массив A, в котором M строк и N столбцов.

Требуется:

- ввести исходный массив из текстового файла;
- вывести исходный массив в текстовый файл.

Найти:

1) сумму и произведение элементов исходного массива;
2) сумму и произведение отрицательных элементов исходного массива и их количество;
3) минимальный и максимальный элемент исходного массива;
4) минимальный и максимальный элемент исходного массива и их индексы;
5) выполнить п.1-4 для каждой строки исходного массива;
6) выполнить п.1-4 для каждого столбца исходного массива;
7) выполнить п.1-4 для каждой нечетной (четной) строки исходного массива;
8) выполнить п.1-4 для каждого нечетного (четного) столбца исходного массива.

Сформировать массив из:

9) нечетных/четных строк/столбцов массива;
10) сумм и произведений нечетных/четных строк/столбцов массива;
11) минимальных и максимальных элементов исходного массива (четных/нечетных строк/столбцов);
12) минимальных и максимальных элементов исходного массива (четных/нечетных строк/столбцов) и их индексов;
13) отрицательных (положительных) элементов исходного массива (четных/нечетных строк/столбцов) и их индексов.

3.7. Процедуры

Процедура – это относительно самостоятельный фрагмент программы для функционально законченной обработки данных, который вызывается как отдельный оператор.

Процедура имеет такое же строение, как и блок программы, но заголовок отличается:

Procedure Имя_процедуры (список формальных параметров);

Имя_процедуры – идентификатор, по которому она вызывается.

Список формальных параметров необязателен. Но если он есть, то ему при вызове должен соответствовать список фактических параметров.

Форма вызова процедуры:

Имя_процедуры (список фактических параметров);

При этом фактические параметры должны соответствовать формальным по типу, количеству и порядку следования (правило согласования параметров).

Схема процедуры также самостоятельна, а не вписана в схему основной программы. Вместо слово «начало» внутри первого блока пишется имя процедуры, а в схеме вызывающей программы (процедуры) должен быть отдельный специальный блок (рис. 1).

Пример процедуры без параметров, вычисляющей минимальное значение массива X:

```
Procedure OBR;
  Var  J : byte;    R : real;
Begin  MIN :=  X[1];
      K := 1;
    For J := 1 to M do
       If X[J] < MIN then begin
            MIN := X[J]; K := J; end;
   X[K] := 0;
 end;
```

Она же, но с параметрами:

```
Procedure OBR (Var B: TMAS;  N: integer; Var M:real; Var K : byte);
  Var  J : byte;    R : real;
Begin  M :=  B[1];
      K := 1;
    For J := 1 to N do
       If B[J] < M then begin
            M := B[J]; K := J; end;
   B[K] := 0;
 end;
```

В списке параметров могут использоваться:

- параметры-переменные (используют для передачи результатов выполнения процедуры в вызывающую),
- параметры-значения (используются для передачи данных для обработки, можно передавать константы, выражения),
- параметры-константы (изменение их в процедурах запрещено),
- процедурные параметры (в данном пособии не рассматриваются).

Параметры-переменные и параметры-константы передаются по адресу, а параметры-значения – по значению.

В процедурах можно использовать локальные (объявленные внутри процедуры) и глобальные (объявленные в вызывающей процедуре) объекты. При этом рабочие переменные должны быть локальными.

В первом случае для обращения к процедуре достаточно просто написать Obr; , а во втором – еще и перечислить параметры. Например, Obr (A,N,M,L).

Упражнения

1. Дано: Массив C, в котором M строк и N столбцов.

Надо:

- ввести исходные данные;

- отпечатать исходные данные;
- сформировать массив из сумм положительных и отрицательных значений каждой строки массива C и количества таких элементов;
- отпечатать сформированный массив.

При программировании использовать процедуры с параметрами для каждого пункта задания.

2. Дано:

A и B – одномерные массивы.

N, M – скаляры,

где N <= 30 - количество элементов массива A,

M <= 40 – количество элементов массива B.

Надо:

- ввести исходные данные,
- отпечатать исходные данные,
- определить и отпечатать значение X

$$X = \frac{2^{KA} \bullet 7^{KB}}{SA + SB}$$

где SA и SB – суммы положительных элементов массивов A и B,

KA и KB – количество положительных элементов массивов A и B.

При программировании использовать процедуру с параметрами для определения значений суммы и количества.

3. Можно выполнить с использованием процедур любое задание из п. 3.6.

3.8. Функции

Функция – это относительно самостоятельный фрагмент программы для функционально законченной обработки данных, который вызывается как переменная. Используется в том случае, если результат должен быть получен в виде одного значения определенного типа.

Функция имеет такое же строение, как и блок программы, но заголовок отличается:

Function Имя_функции (список формальных параметров) : тип_функции;

Имя_функции – идентификатор, по которому она вызывается.

Список формальных параметров необязателен. Но если он есть, то ему при вызове должен соответствовать список фактических параметров. Правила согласования параметров те же, что и для процедур, но в списке параметров не должно быть результата, т.е. скорее всего параметры-переменные не используются. Тип функции – простой. Для формирования результата в теле

функции должен быть оператор, присваивающий имени функции значение результата.

Пример

```
Function PR(N: word): longint;
  var s:longint;
  begin s:=1;
          For J := 1 to N do
            s:= s*J;
      PR:=s;
  end;
```

Упражнения

1. Дано:

A, B, N, M – скаляры,
где N <= 30 - количество элементов массива X,
M <= 40 – количество элементов массива Y;
X и Y – одномерные массивы.

Надо:

- ввести исходные данные,
- отпечатать исходные данные,
- определить и отпечатать значение T

$$T = \frac{\prod_{i=1}^{N} x_i^{A-1} + A \bullet \prod_{j=1}^{M-2} Y_j^{B+3}}{\prod_{i=1}^{N-1} X_i} + A \bullet B$$

При программировании использовать функцию с параметрами для определения значения произведения.

2. Дано:

A и B – одномерные массивы.
N, M – скаляры,
где N <= 30 - количество элементов массива A,
M <= 40 – количество элементов массива B.

Надо:

- ввести исходные данные,
- отпечатать исходные данные,
- определить и отпечатать значение X

$$X = \frac{2^{KA} \bullet 7^{KB}}{SA + SB}$$

где SA и SB – суммы положительных элементов массивов A и B,

КА и КВ – количество положительных элементов массивов A и B.

При программировании использовать функции с параметрами для определения значений сумм и количеств.

3. Дано: A, B, C, D - скаляры.

Надо:

- ввести исходные данные,
- отпечатать исходные данные,
- определить и отпечатать значение Y

$$Y = \frac{(A+D)!+B!}{C!\bullet D!} - (A+B)!$$

При программировании использовать функцию с параметрами для определения значения факториала.

3.9. Рекурсии

Рекурсивным называется объект, частично состоящий или определяемый с помощью самого себя. Рекурсивные определения представляют мощный аппарат в математике. Общеизвестные примеры: натуральные числа, деревья, некоторые функции (факториал).

Функция N! (для N>0):

- 0!=1
- N!=N*(N-1)!

Если процедура содержит явную ссылку на саму себя, то ее называют прямо рекурсивной, а если она обращается к другой процедуре, которая в свою очередь содержит ссылку на нее, то такую функцию называют косвенно рекурсивной.

Функция для вычисления факториала выглядит следующим образом:

```
Function FACT ( N: integer): longint;

Begin        if N<0 then begin
     Writeln (‘ Значение N отрицательно’ ); Halt;       end;
     if (N = 0) or (N = 1) then FACT := 1
                              else FACT := N * FACT ( N-1 );      end;
```

С помощью конечной рекурсивной программы можно описать бесконечное вычисление, причем программа не будет содержать явных повторений.

Рекурсивные процедуры могут приводить к незаканчивающимся вычислениям, поэтому на эту проблему необходимо обращать особое внимание. Кроме того, каждая рекурсивная активация процедуры требует памяти для размещения ее переменных. Также нужно сохранять текущее состояние вычислений, чтобы можно было вернуться в него по окончании новой активации. Таким образом, в практических применениях рекурсии важно убедиться, что максимальная глубина рекурсий не только конечна, но и достаточно мала.

При построении рекурсивной процедуры нужно помнить, что она должна содержать условие выхода из рекурсии (в данном случае (N = 0) or (N = 1)) и механизм достижения этого условия (в данном случае N-1 в операторе присваивания FACT := N * FACT (N-1);.

Рекурсия может заменить цикл. Например, сумму N элементов одномерного массива X можно посчитать двумя способами:

```
S:=0;
For  i:= 1 to N do S:= S + X[i];
```

и через рекурсивную процедуру

```
Function SUM ( N: integer; X: TMAS): real;
    Begin
        if N < 1      then SUM := 0
                      else  SUM := X[N] + SUM ( N-1, X );
    end;
```

Суммирование N членов бесконечного сходящегося ряда выглядит следующим образом:

```
Function SUM ( N: integer; X, EPS: real): real;
  Var F : real;
    Begin
        F:=STEP ( X, N) / FACT (n);
        if abs (F) >= EPS then SUM := F
                          else  SUM := F + SUM ( N+1, X, EPS );
    end;
```

Считается, что там, где можно обойтись без рекурсии, ее нужно исключить. Однако есть ряд алгоритмов, где использование рекурсии не только оправдано, но и является единственно возможным решением.

Упражнения

Разработать программу, использующую рекурсивную процедуру для:

- вычисления произведения элементов массива вещественных чисел;
- вычисления X в степени N;
- для ввода строки и печати ее в обратном порядке;

- печати всех целых четных чисел в интервале от N до M;
- всех степеней тройки в интервале от N до M;
- ввода одномерного массива вещественных чисел и печати его в обратном порядке;
- печати всех квадратов целых чисел в интервале от N до M;
- определения минимального среди последних K элементов одномерного массива;
- нахождения корня уравнения f(x)=0 методом деления отрезка пополам на интервале от A до B;
- печати всех перестановок N натуральных целых чисел;
- печати N натуральных целых чисел, являющихся последовательностью Фибоначчи;
- печати всех делителей целого числа N.

3.10. Записи

Запись – это структура данных, состоящая из фиксированного количества элементов, называемых полями. Они могут быть (и чаще всего бывают) разного типа. Ключевое слово для объявления записи RECORD, в конце обязательно END.

Например, объявление записи, содержащей 5 полей:

```
TYPE        TZ  = record
                    NOM : byte;
                    RS : word;
                    PAS : string [ 13 ];
                    FIO : string [ 18 ];
                    SB : real;
                    end;
VAR Z : TZ;
```

Соответственно массив записей - MZ : array [1 .. M] of TZ;

Обращение к массиву записей – по полям напрямую или через оператор присоединения with.

Пример ввода массива записей с использованием оператора присоединения:

```
    for i :=1 to M do
        with MZ [ i ] do begin
            readln ( FI , NOM , RS , b , PAS , b , FIO , SB );
                        end;
```

Операторные скобки в данном случае не являются обязательными, а лишь иллюстрируют, что оператор with может распространяться более чем на один оператор.

Без использования оператора присоединения ввод массива записей:

```
for i :=1 to M do
        readln ( FI , MZ [ i ].NOM , MZ [ i ].RS , b , MZ [ i ].PAS , b ,
                MZ [ i ].FIO , MZ [ i ].SB );
```

Над полями записи можно производить манипуляции в соответствии с объявленным типом поля.

Тип поля может быть любого типа, в том числе и сложным. Объявлять такие записи нужно, начиная с простых (вложенных типов).

Например, требуется объявить запись, содержащую сведения о вузе, в котором 10 факультетов, на каждом из которых не более 30 групп, в каждой из которых не более M студентов. При этом нужно хранить названия факультетов, групп и сведения о студентах.

```
TYPE
    S18 = string [ 18 ];
    TZ  = record                    {сведения о студенте}
          NOM : word;               { номер зачетки }
           RS : word;               { размер стипендии }
          FIO : S18;                { фамилия студента}
           SB : real;               { рейтинг }
          end;
    MZ : array [ 1 .. M ] of TZ; {сведения о M студентах группы}
    TP  =  record
           NG : string [6];         {название группы}
           GR : MZ;
           end;
    TF  =  array [1..30] of TP;    {сведения о 30 группах}
    TI  =  record
           F  : string [10];       {название факультета}
           All: TF;
           end;
 VAR
   Z : array [1..10] of TI; {сведения 10 факультетах}
```

Упражнения

1. Даны сведения о K библиотеках города в составе:

- № библиотеки,
- Название,
- Адрес (район),
- телефон,
- количество наименований книг,
- количество читателей.

Написать схему алгоритма и программу для ввода, вывода исходного массива и определения:

- библиотеки с наибольшим количеством книг и наименьшим количеством читателей;
- общего количества книг;
- библиотеки с максимальным количеством читателей;
- телефона заданной по номеру библиотеки;
- номеров всех библиотек в заданном районе.

2. Даны сведения о Kol театрах города в составе:

- название,
- адрес,
- год основания,
- режиссер,
- количество спектаклей в текущем репертуаре,
- сведения о спектаклях текущего репертуара в составе:
  - название,
  - автор пьесы,
  - режиссер,
  - продолжительность,
  - дата премьеры;
- сведения труппе (К артистах) в составе:
  - ФИО,
  - год рождения,
  - звание,
  - амплуа.

Объявить массив записей и написать фрагменты программы для его ввода и вывода.

3.11. Типизированные файлы

Типизированные файлы используются для хранения данных типа компонентов файла, определенных при его объявлении. Объявление файла:

Var F: file of тип; где тип – любой тип, кроме файлового.

Процедура Assign (F, ‘имя_файла_на_диске’) ничем не отличается от текстового файла, так же, как и Close (F); Отличие состоит в открытии файлов и чтении/записи.

Типизированный файл можно открыть только Rewrite(F) (для записи) и Reset(F) (для чтения и дозаписи). Append использовать нельзя!

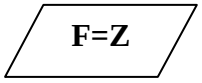
При чтении и записи в типизированный файл переменные в списке данных должны иметь тот же тип, что и компоненты файла.

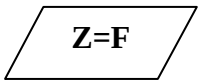
Например, если объявлено

```
TYPE
    TZ  = record
          NOM : byte;            { номер рейса }
          PAS : string [ 13 ];    { паспорт }
          FIO : string [18];           { фамилия}
          end;
    FL = file of TZ;
 VAR    Z : TZ;  F: FL; Fi : text;
```

то чтение типизированного файла Read (F, Z); в то время как текстового Read (Fi, z.nom, z.pas, z.fio).

На схеме алгоритма для работы с типизированным файлом используется один блок, но направление обмена данными указывается по аналогии с оператором присваивания:

Запись в файл – Write (F, Z); 

Чтение из файла – Read (F, Z); 

Вся работа с файлом строится с использованием пяти абстрактных операций:

- создание файла,
- чтение файла,
- дополнение файла,
- модификация (корректировка) записей файла,
- удаление записей из файла (в настоящем пособии не рассматривается).

Остальные операции рассмотрим подробнее.

Запись в файл заключается в последовательном вводе данных из текстового файла и их пересылка в типизированный.

```
PROCEDURE SOZ;
 var fid : text;
Begin  assign ( fid , 'lr8soz.dat'); reset ( fid ); rewrite ( F );
         while not seekeof ( fid ) do
             with Z do begin
                 readln ( fid , NOM , PAS , FIO);
                 write ( f , Z );
                     end;
       close ( fid );  close ( F ); end;
```

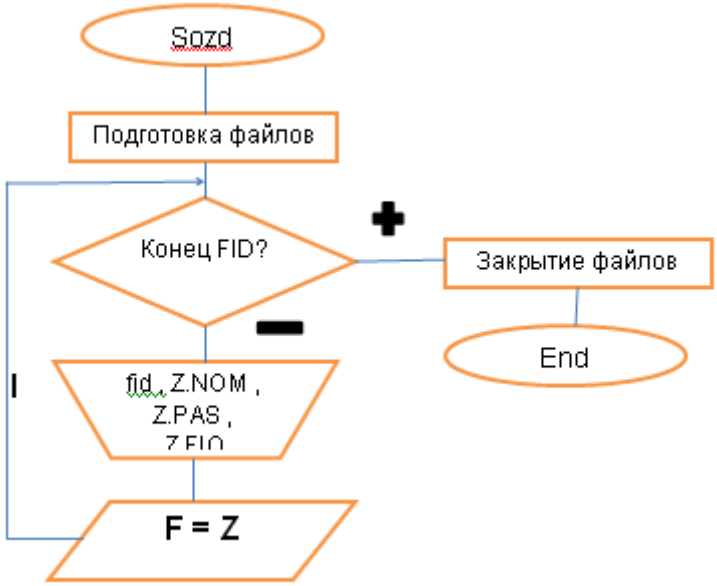


Рис. 12

Дополнение файла выполняется так же, как создание, но файл уже существует. Поэтому его нужно открыть для дозаписи Reset (F) и перейти в конец файла с помощью процедуры Seek (F, n). Здесь n – указатель записи в файле, на которую нужно перейти. filesize (F) – функция, которая определяет количество записей в файле. Записи нумеруются с 0, поэтому переход в конец файла – это переход на запись с номером, определяющим количество элементов в файле.

```
PROCEDURE DOP;
var fid : text;
Begin  assign ( fid , 'lr8dop.dat'); reset ( fid ); reset ( F );
      seek ( F , filesize ( F ) );
         while not seekeof ( fid ) do
              with Z do begin
                   readln ( fid , NOM , PAS , FIO); write ( f , Z );
                              end;
       close ( fid );  close ( F ); end;
```

Схема алгоритма аналогична схеме представленной на рис. 12.

Чтение файла – практически противоположная созданию операция – из типизированного файла последовательно до конца файла считываем записи и выводим на экран или в текстовый файл.

```
PROCEDURE CHT;
Begin
      reset ( f );
      repeat
            read ( f , Z ); PRINT;
      until eof ( f );
      close ( f ); end;
```

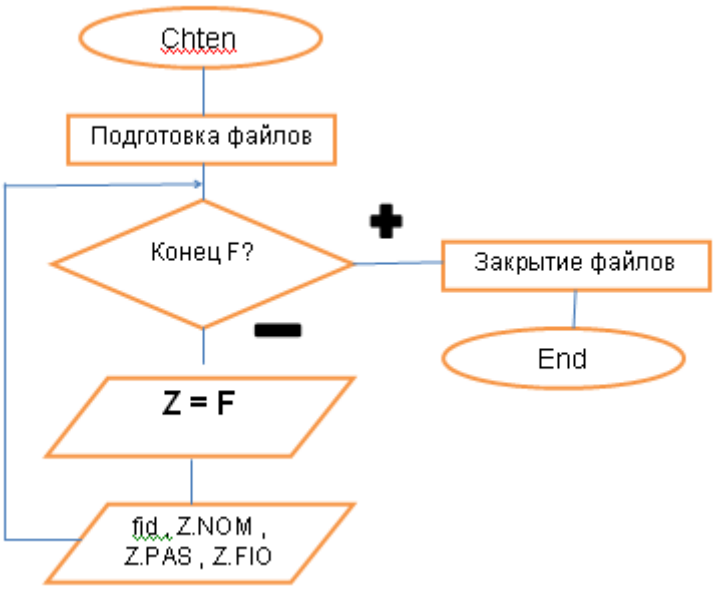


Рис. 13

Модификация записей в файле заключается в поиске записи, которую нужно изменить, внесением в нее новой информации (изменение полей записи) и запись ее на старое место. Для того, чтобы записать на старое место необходимо вернуться на одну запись назад, поскольку после чтения записи указатель переместится за нее. Это можно сделать с помощью процедуры:

```
seek ( F , filepos ( F ) - 1 );
```

где filepos (F) – функция, возвращающая номер текущей активной записи в файле.

```
PROCEDURE KOR;
Label MK;
Var trs : word;  tfio : S18;
 Begin   Assign ( FI ,'lr8kor.dat' ); Reset ( FI );  Reset ( F );
      repeat
           readln ( FI , tfio , tnm );
           tfio := FILTR ( tfio );
```

```
        if tfio = '' then begin
          writeln ( 'Нет фамилии для поиска.' );
          continue;      end;
        seek ( F , 0 );
        repeat
             read ( F , Z );
             if tfio = Z.FIO then begin
               Z.NOM := tnm;
               seek ( F , filepos ( F ) - 1 );
               write ( F , Z );
               goto MK;
                          end;
        until eof ( F );
        writeln ( 'Фамилия не найдена' );
 MK : until eof ( FI );
    Close ( FI );   Close ( F ); End ;
```

Упражнения

- Дан текстовый файл, содержащий ряд целых чисел. Создать два типизированных файла. В один поместить все четные, а в другой все нечетные числа.
- Создан типизированный файл, содержащий целые числа. Удалить из файла все числа, находящиеся в заданных пределах, определить их сумму и количество.
- Создан типизированный файл, содержащий вещественные числа. Заменить все отрицательные числа в файле их модулями.
- Созданы два типизированных файла, содержащих вещественные числа. Удалить из второго файла все те числа, которые содержатся в первом.
- Созданы два типизированных файла, содержащих вещественные числа. Сравнить длины файлов. В более длинном файле оставить столько же чисел, сколько в более коротком, а оставшиеся числа переписать в третий файл.
- Создан типизированный файл, содержащий строки, длиной не более 9 символов. Создать файл, содержащий те же строки, но расположенные в обратном порядке. Создать файл, в который поместить слова максимально возможной длины. Определить количество таких слов.

- Создан типизированный файл, содержащий строки, длиной не более 15 символов. Удалить из файла ряд заданных слов. Создать массив, содержащий удаленные слова и количество их появлений в тексте.
- Созданы два типизированных файла, содержащих строки, длиной не более 12 символов. Удалить из второго файла все слова, которые содержатся в первом. Определить количество удаленных слов.
- Создан типизированный файл, содержащий строки, длиной не более 8 символов. В текстовом файле содержатся слова, которые необходимо найти в файле и заменить пробелами (каждое слово – на новой строке). Определить количество найденных слов.
- Дан ряд массивов вещественных чисел. Известно количество элементов в них. В каждом массиве не более 10 элементов. Ввести и напечатать исходные данные. Сформировать типизированный файл, содержащий сумму и произведение элементов каждого массива.

4. Список рекомендуемой литературы

1. Вирт Н. Алгоритмы и структуры данных. – М.: ДМК Пресс, 2010.
2. Грызлов В.И., Грызлова Т.П. Рекомендации по записи программ Турбо Паскаль 7.0. – М.: ДМК Пресс, 2008.
3. Егорова А.А Пособие по дисциплине «Структуры данных и методы обработки информации». – М.: МГТУ ГА, 2011.
4. Климова Л.М. Pascal 7.0. Практическое программирование. Решение типовых задач. - 4-е изд. – М.: КУДИЦ-ОБРАЗ, 2003.

СОДЕРЖАНИЕ