ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)

Кафедра прикладной математики

А.А. Егорова

АЛГОРИТМИЧЕСКИЕ ЯЗЫКИ И ПРОГРАММИРОВАНИЕ

ЧАСТЬ II. МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ

Учебно-методическое пособие
по выполнению лабораторных работ

для студентов I курса
направления 01.03.04
очной формы обучения

Москва
ИД Академии Жуковского
2020

УДК 004.42
ББК 6Ф6.5
Е30

Рецензент:
Коновалов В.М. – канд. техн. наук, доцент

Егорова А.А.
Е30 Алгоритмические языки и программирование. Часть II. Модульное программирование [Текст] : учебно-методическое пособие по выполнению лабораторных работ / А.А. Егорова. – М.: ИД Академии Жуковского, 2020. – 52 с.

Данное учебно-методическое пособие издается в соответствии с рабочей программой учебной дисциплины «Алгоритмические языки и программирование» по учебному плану для студентов I курса направления 01.03.04 очной формы обучения.

Рассмотрено и одобрено на заседаниях кафедры 29.08.2020 г. и методического совета 29.08.2020 г.

УДК 004.42
ББК 6Ф6.5

В авторской редакции

Подписано в печать 16.12.2020 г.
Формат 60x84/16 Печ. л. 3,25 Усл. печ. л. 3,02
Заказ № 690/1008-УМП10 Тираж 90 экз.

Московский государственный технический университет ГА
125993, Москва, Кронштадтский бульвар, д. 20

Издательский дом Академии имени Н. Е. Жуковского
125167, Москва, 8-го Марта 4-я ул., д. 6А
Тел.: (495) 973-45-68
E-mail: zakaz@itsbook.ru

ПРЕДИСЛОВИЕ

Настоящее пособие содержит задания на выполнение лабораторных работ по дисциплине «Алгоритмические языки и программирование», выполняемых студентами I курса направления подготовки «Прикладная математика» во втором семестре.

Навыки, приобретенные в процессе выполнения лабораторных работ, необходимы студентам в процессе дальнейшей подготовки практически по всем дисциплинам, а также для самостоятельной работы и самоподготовки.

В пособии отражены организационно-методические аспекты выполнения лабораторных работ, особенности их проведения, цели, достигаемые в процессе выполнения лабораторных работ.

Пособие охватывает дисциплину не полностью, а только второго семестра и содержит цель, задание на выполнение каждой лабораторной работы, требования к их выполнению, краткие теоретические сведения, примеры выполнения и контрольные вопросы.

Теоретический материал содержит информацию по способам разработки и использования процедур, функций и модулей, классические алгоритмы поиска и сортировки данных. Теоретический материал не претендует на полноту, а содержит только необходимую для выполнения работы информацию. Требования к выполнению работ являются обязательными в рамках выполнения работ по дисциплине «Алгоритмические языки и программирование», сформулированы в соответствии с методикой преподавания дисциплины.

Список рекомендуемой литературы содержит актуальные ссылки на Интернет-источники.

Настоящее пособие может быть использовано и как справочник при самостоятельной работе при программировании прикладных задач, в том числе по другим дисциплинам.

Пособие имеет прикладной характер, что способствует формированию у студентов компетенций в соответствии с требованиями, содержащимися в рабочей программе по дисциплине «Алгоритмические языки и программирование», и в целом соответствующие модели компетенций по направлению подготовки «Прикладная математика».

Оглавление

1. Введение

Модульное программирование - способ разработки программы, которая строится из нескольких относительно независимых друг от друга частей – модулей. Понятие модуля является одним из центральных при разработке программного обеспечения.

При разработке программ в парадигме структурного программирования чаще всего используется метод, который называется "программирование сверху-вниз" или "пошаговая детализация". Суть метода: исходная задача разбивается на относительно независимые друг от друга подзадачи. Если полученные подзадачи просты, то для них разрабатывается алгоритм, иначе для каждой подзадачи снова происходит разделение на подзадачи. Каждую из полученных в итоге подзадач реализуют в виде отдельного относительно самостоятельного программного фрагмента, называемого модулем. Таким образом, **модуль** - это функционально законченный фрагмент программы. В программном коде модули реализуются в виде процедур и функций.

Модули задачи могут быть написаны как на одном языке программирования, так и на разных, и в этом случае говорят, что используется многоязыковая система программирования.

Настоящее пособие, предназначенное для выполнения лабораторных работ по дисциплине «Алгоритмические языки и программирование», призвано, в первую очередь, привить студентам навыки грамотного использования программных инструментов для разработки модулей, будь то процедуры и функции, или личные «библиотеки», предназначенные для использования в ряде разрабатываемых программ.

Целью проведения лабораторных работ является как закрепление основных теоретических положений, изложенных в лекциях, так и получение практических навыков по реализации модульного подхода в программировании как для числовой, так и нечисловой обработки данных, необходимых студентам в процессе изучения других дисциплин ИТ-направления.

Пособие необходимо студентам на всех этапах, начиная от подготовки до оформления отчета и защиты лабораторной работы.

2. Организационно-методические рекомендации

2.1. Отчет по лабораторной работе

По окончании лабораторной работы студент оформляет отчет, который включает:

1. номер работы,
2. название работы,
3. цель работы,
4. задание на выполнение работы,
5. вариант выполнения работы,
6. алгоритмы выполнения каждой процедуры за исключением небольших линейных (допускается «от руки»),
7. программа (листинг, распечатка),
8. распечатки исходных данных и результатов работы (или файлы, или скриншоты в соответствии с заданием).

Если лабораторная работа предполагает выполнение нескольких заданий, то п.5-8 группируются по каждому заданию.

Скриншоты делать не полного экрана, а фрагмента, захватив актуальную информацию. Распечатка программы должна содержать комментарии: фамилию студента и другие, необходимые для удобочитаемости программы. Размеры букв в распечатке должны быть такими, чтобы текст было удобно читать, не поднимая лист со стола (например, не

менее 10 пт для шрифта Times New Roman). Схема алгоритма, выполненная «от руки» должна быть аккуратной, использование линейки обязательно.

Номер работы и ее название, а также фамилия и группа студента могут быть оформлены на титульном листе, где также указывается дисциплина. Но можно и сделать общий титульный лист на все лабораторные работы семестра.

2.2. Защита лабораторной работы

После сдачи выполненной работы преподавателю и оформления отчета студент защищает работу (на следующем занятии). В процессе защиты преподаватель проверяет правильность оформления отчета (в первую очередь алгоритмов), соответствие программы заданию, исходных данных и результатов программе. Студент отвечает на вопросы преподавателя как по выполнению работы, так и по теоретической части работы (цель работы и контрольные вопросы - ориентир).

3. Лабораторная работа №5

Разработка программ обработки арифметических массивов данных с использованием процедур

3.1. Цель работы

Целью работы является освоение:

- строения процедур,
- назначения и состава формальных и фактических параметров, правил их записи и согласования,
- правил вызова процедур,
- приемов отладки программ с процедурами,
- правил определения области действия имен переменных и констант.

3.2. Задание на выполнение работы

Разработать программу для обработки двумерных арифметических массивов, в которой для всех вариантов:

1. границы массива объявить в разделе констант;
2. значения исходных данных сформировать в файле; элементы массивов должны иметь мантиссы от 2 до 6 разрядов, быть положительными и отрицательными, целыми, дробными и в виде смешанных дробей; должны быть подобраны таким образом, чтобы продемонстрировать все возможности программы.
3. С помощью отдельных процедур с параметрами выполнить:

- ввод исходных данных,
- вывод массива исходных данных (в виде матрицы с выровненными столбцами),
- формирование массива и обработку данным (допустимо использовать 1, 2 или 3 процедуры),
- вывод результатов.

3.3. Требования к выполнению

- Для вывода результатов использовать стандартный файл OUTPUT, для исходных – нестандартный текстовый файл;

- Все, что «дано», вводить из текстового файла;
- Программы должны быть хорошо структурированы, содержать комментарии.
- Алгоритмы всех процедур и основной программы должны быть представлены в виде графических схем.

3.4. Краткие теоретические сведения

Процедуры и функции в языке Pascal являются подпрограммами. Подпрограмма – это именованная часть программы, представляющая собой некоторое собрание операторов, структурированных аналогично основной программе. Подпрограммы могут быть встроенными и пользовательскими. Первые изначально присутствуют в системе, а вторые создаются программистом.

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>(<формальные параметры>: <тип данных>);

Здесь:

Имя_процедуры – идентификатор, по которому она вызывается;

Формальные параметры необязательны.

Формальные параметры могут оформляться в виде списка. При этом если тип данных у всех параметров один, то параметры перечисляются через запятую. Например,

Procedure OBR (M, N, K : integer);

Если типы у параметров разные, то разные блоки параметров разделяются точной с запятой. Например,

Procedure OBR (N: integer; M:real; K : byte);

Параметры используются для передачи данных. Параметры можно разделить на передаваемые и возвращаемые. Передаваемые параметры – это переменные, которые подпрограмма получает из главной части программы (или из другой подпрограммы, которая в этом случае называется вызывающей), а возвращаемые – отдает (возвращает) ей.

Различают формальные и фактические параметры. Те, что используются при вызове процедуры или функции называются фактическими, а формальные описываются в заголовке подпрограммы, и принимают значения фактических параметров при вызове.

При этом важно помнить **Правило согласования параметров**.

Фактические параметры должны соответствовать формальным по количеству, типу и порядку следования:

1. Количество формальных и фактических параметров должно быть одинаково.
2. Типы формальных и соответствующих им фактических параметров должны совпадать или быть согласованными (аналогично правой и левой частям оператора присваивания).
3. Последовательность формальных параметров и соответствующих им фактических должна быть одинакова (например, если в заголовке процедуры первый параметр – массив исходный, а второй – массив формируемый, то и в фактических параметрах они должны быть представлены именно в таком порядке, даже если они одного типа).

Обратите внимание, что согласование по имени не предусмотрено (имена могут быть разными). А это значит, что с помощью одной процедуры могут обрабатываться данные с разными именами (но одного типа).

В Паскале различают формальные параметры в первую очередь по механизму их передачи:

- по адресу – формальный и фактический параметры имеют общую область памяти, т.е. фактически передается информация о месте нахождения переменной в оперативной памяти (ОП);
- по значению – формальный параметр вычисляется (при необходимости) и заносится в дополнительную область памяти, выделяемую под формальный параметр (т.е. значение дублируется).

Таким образом, в первом случае вызывающая программа видит результат работы подпрограммы, а во втором нет.

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 (будут более подробно рассмотрены в следующей лабораторной работе).

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

В Pascal, как и в большинстве языков программирования, переменные по отношению к программе делятся на локальные и глобальные. Те, которые объявляются в основной части, называются глобальными, а в процедуре или функции – локальными. Локальные переменные в отличие от глобальных могут использоваться лишь внутри подпрограммы, к которой они принадлежат.

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

Пример процедуры без параметров, вычисляющей минимальное значение массива X:

```
Procedure OBR;
// Вычисление минимального значения массива X
   Var  J : byte;
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;
```

Эта же процедура, но с параметрами:

```
Procedure OBR (Var B: TMAS;  N: integer; Var M:real; Var K : byte);
// Вычисление минимального значения массива B
   Var  J : byte;
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 данных примерах переменная J – локальная рабочая переменная (параметр цикла). Остальные переменные в первом примере – глобальные. Во втором примере – формальные параметры. При этом переменные M и K является результатами, поэтому использование **Var** обязательно (использование параметра переменной), а массив B – не результат, не изменяется, для экономии оперативной памяти лучше использовать параметр-переменную или параметр-константу. Но параметр-значение также допустим.

Форма вызова процедуры:

Имя_процедуры (список фактических параметров);

В первом случае для обращения к процедуре достаточно просто написать в вызывающей процедуре (в том числе в главной) Obr; , а во втором – еще и перечислить параметры. Например, Obr (X, N, M, L). При этом переменные X, N, M, L должны быть объявлены в вызывающей программе. Тогда фактически процедура OBR и во втором случае будет вычислять минимальное значение массива X.

Принципиальна последовательность размещения подпрограмм в программе: вызываемая программа должна располагаться до вызывающей. Избежать этого помогает опережающее описание подпрограмм: заголовки подпрограмм выносятся в начало программы в декларативный блок (блок описания данных) с ключевым словом ***Forward***. При косвенной рекурсии использование слова ***Forward*** является обязательным.

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 (см. рисунок 1). Блок так и называется «процедура» (или типовой процесс). Он имеет один вход и один выход.

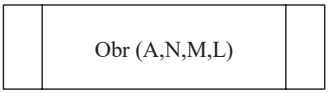


Рисунок 1. Блок для обозначения вызова процедуры в схеме алгоритма

3.5. Пример выполнения работы

Рассмотрим выполнение лабораторной работы на следующем примере.

Дан двумерный массив X из M строк и N столбцов. Требуется ввести и вывести исходный массив, а также

1) Сформировать массив из индексов максимальных значений каждой строки массива;
2) Поменять местами в исходном массиве строки с заданными номерами;
3) Определить сумму и количество положительных элементов заданного столбца массива.

Очевидно, что для рассматриваемого примера необходимы 5 процедур (вывод элементов исходного массива, формирование нового массива, его вывод, процедуру для смены строк местами и для вычисления суммы и количества). Можно добавить процедуру ввода исходного массива.

```
PROGRAM LR5;
 CONST  M = 5; N = 6;

 TYPE     TMAS = ARRAY [1..M, 1..N] OF REAL;
            TR = ARRAY [1..M, 1..2] OF BYTE;

 VAR         X : TMAS;
```

```
            R : TR;
     K1,K2,I,J,Kol : BYTE;
          Sum : REAL;
           FI : TEXT;

{ --------Процедура вывода--------------------}
     PROCEDURE PRINT (B : TMAS; M, N : INTEGER);
     VAR I,J : BYTE;
     BEGIN
            FOR I := 1 TO M DO begin
                    FOR J := 1 TO N DO WRITE ( B[I,J] : 8 :1);
                 writeln;       end;
     END;

{ --------Процедура вывода нового массива-------------------}
     PROCEDURE PRINT2 (B : TR; M : INTEGER);
     VAR I : BYTE;
     BEGIN
            FOR I := 1 TO M DO  WRITELN ( B[I,1] : 3,  B[I,2] : 3);
                 writeln;
     END;

//--------Процедура формирования массива-------------------
PROCEDURE OBRABOT (VAR B: TMAS; VAR R : TR; M,N: INTEGER);
   VAR  I, J, JM : BYTE;
BEGIN
     FOR I := 1 TO M DO BEGIN
         JM :=1;
         FOR J := 1 TO N DO
             IF B[I,J] > B[I,JM] THEN JM := J;
         R[I,1] := I;  R[I,2] := JM; END;
 END;

 { --------Процедура смены строк местами-------------------}
     PROCEDURE CHANGE (VAR B : TMAS; M, K1, K2 : INTEGER);
     VAR I : BYTE; TEMP : REAL;
     BEGIN
            FOR I := 1 TO M DO BEGIN
                  TEMP := B[I,K1];
                  B[I,K1]:= B[I,K2];
                  B[I,K2]:= TEMP;
                     END;
     END;

{ --------Процедура обработки столбца-------------------}
     PROCEDURE SUMKOL (VAR B : TMAS;  M, K1  : INTEGER; VAR S: REAL; VAR K: BYTE);
     VAR I : BYTE;
     BEGIN
            FOR I := 1 TO M DO
               IF B[I,K1] > 0 THEN BEGIN
                  S := S + B[I,K1];
                  K := K + 1;
                                  END;
     END;

 {-------------- Основная программа--------------}
BEGIN   ASSIGN (FI, 'ffff.dat');  RESET(FI);

 // --------ввод исходного массива
  WRITELN (' Исходный массив ' );
     FOR I := 1 TO M DO
          FOR J := 1 TO N DO READ ( FI, X[I,J] );
     PRINT ( X, M, N );

  OBRABOT ( X, R,  M, N );WRITELN;
  WRITELN (' Результат работы ' );
  WRITELN (' Сформированный массив: ' );
  PRINT2 ( R, M );

  WRITE (' Введите номера строк, которые нужно поменять местами ' );
  READLN (K1, K2); CHANGE (X, M, K1, K2 );
  WRITELN (' Измененный исходный массив ' );
```

```
  PRINT ( X, M, N );    WRITELN;

  WRITE (' Введите номера столбца, который нужно обработать ' );
  READLN (K1);
  SUMKOL (X, M, K1, Sum, Kol);
  WRITELN (' сумма и произведение ', K1, '-го столбца :' );
  WRITELN ( ' Sum = ', Sum : 5:1 );
  WRITELN ( ' Kol = ', Kol: 3 );
END.
```

Результат работы программы представлен на листинге:

```
Окно вывода
Исходный массив
     6.0      4.0      5.0      3.0      7.0      9.0
    16.0    -11.0     12.0     13.0    -14.0     15.0
    21.0     26.0     23.0     24.0     25.0    -22.6
    31.0    -32.0     33.0    -34.2     35.0     30.0
    41.0     42.0     43.0     48.0     45.0     46.0

Результат работы
Сформированный массив:
 1  6
 2  1
 3  2
 4  5
 5  4

Введите номера строк, которые нужно поменять местами 1 6
Измененный исходный массив
     9.0      4.0      5.0      3.0      7.0      6.0
    15.0    -11.0     12.0     13.0    -14.0     16.0
   -22.6     26.0     23.0     24.0     25.0     21.0
    30.0    -32.0     33.0    -34.2     35.0     31.0
    46.0     42.0     43.0     48.0     45.0     41.0

Введите номера столбца, который нужно обработать 2
сумма и произведение 2-го столбца :
Sum =  72.0
Kol =   3
```

3.6. Варианты на выполнение работы №5

1. Дан двумерный массив B из M строк и N столбцов.
   1) Сформировать массив из максимальных значений каждой строки массива и их индексов;
   2) Найти и поменять местами два максимальных значения сформированного массива;
   3) определить сумму и количество положительных элементов нечетных столбцов исходного массива.

2. Дан двумерный массив A из M строк и N столбцов.
   1) Сформировать массив из минимальных значений каждого столбца и их индексов;
   2) Найти и поменять местами минимальное и максимальное значения сформированного массива;
   3) определить произведение и количество отрицательных элементов четных строк исходного массива.

3. Дан двумерный массив R из M строк и N столбцов.
Даны V и W.

1) Сформировать массив из первых попавшихся элементов каждого нечетного столбца массива C, находящихся в пределах: V < C[i, j] < W их индексов в исходной матрице; если их нет, результат должен быть равен 0;
2) определить произведение отобранных элементов и их количество;
3) определить минимальное и максимальное значения из отобранных элементов и их координаты и поменять местами столбцы исходной матрицы, в которых они найдены, если номера столбцов разные.

4. Дан двумерный массив C из M строк и N столбцов.
1) Сформировать массив из первых попавшихся отрицательных элементов каждой строки и их индексов в исходной матрице; если отрицательных элементов нет, результат должен быть равен 0;
2) определить сумму отобранных элементов массива и их количество;
3) определить минимальное и максимальное значения из элементов исходного массива и их координаты и поменять местами столбцы исходной матрицы, в которых они найдены, если номера строк разные.

5. Дан двумерный массив Z из M строк и N столбцов.
1) Сформировать массив из отрицательных элементов четных строк и их индексов;
2) Определить количество таких элементов в каждой нечетной строке.
3) Определить максимальное значение из отобранных элементов и поменять его местами с последним элементом исходного массива, если индексы отличаются.

6. Дан двумерный массив Y из M строк и N столбцов.
1) Сформировать массив из положительных элементов нечетных столбцов и их индексов;
2) Определить количество таких элементов в каждом нечетном столбце.
3) Определить минимальное значение из отобранных элементов и поменять его местами с первым элементом исходного массива, если индексы отличаются.

7. Дан двумерный массив B из N строк и N столбцов.
1) Сформировать массив из отрицательных элементов диагоналей массива и их индексов;
2) Определить сумму каждого нечетного столбца исходного массива;
3) Поменять в исходном массиве элементы прямой диагонали (первый с последним, второй с предпоследним и т.д.).

8. Дан двумерный массив C из N строк и N столбцов.
1) Сформировать массив из положительных элементов диагоналей массива и их индексов;
2) Определить произведение каждой четной строки исходного массива;
3) Поменять в исходном массиве элементы обратной диагонали (первый с последним, второй с предпоследним и т.д.).

9. Дан двумерный массив X из M строк и N столбцов.
Даны K и F.
1) Сформировать массив из:
а) элементов каждой нечетной строки, находящихся в пределах:
K < X[i, j] <= F
б) индексов этих элементов;
2) Определить минимальный элемент каждого столбца;

3) В исходном массиве поменять местами в каждом столбце минимальные элементы с первым, если номера строк разные.

10. Дан двумерный массив X из M строк и N столбцов.
Даны K и F.
1) Сформировать массив из:
а) элементов каждого четного столбца, находящихся в пределах:
K < X[i, j] <= F
б) индексов этих элементов;
2) Определить максимальный элемент каждой строки;
3) В исходном массиве поменять местами в каждой строке максимальные элементы с последним, если номера строк разные.

11. Дан двумерный массив Y из M строк и N столбцов.
1) Сформировать массив из двух минимальных значений каждой нечетной строки матрицы и их индексов;
2) Поменять местами значения двух минимальных в каждой нечетной строке исходной матрицы;
3) Определить сумму и произведение отрицательных элементов нового массива.

12. Дан двумерный массив D из M строк и N столбцов.
1) Сформировать массив из двух максимальных значений каждого нечетного столбца матрицы и их индексов;
2) Поменять местами 2 максимальных значения в каждом нечетном столбце исходной матрицы;
3) определить сумму и произведение положительных максимальных элементов нового массива.

13. Дан двумерный массив A из M строк и N столбцов.
1) Сформировать массив из:
а) сумм отрицательных элементов каждого четного столбца;
б) количества таких элементов в каждой четной строке;
2) Определить произведение сумм сформированного массива и общее количество положительных элементов четных строк;
3) Определить минимальное и максимальное значения сумм и поменять местами строки исходной матрицы, в которых они найдены, если номера строк разные.

14. Дан двумерный массив C из M строк и N столбцов.
1) Сформировать массив из:
а) произведений положительных элементов каждой нечетной строки; если их нет, результат должен быть равен 0;
б) количества таких элементов в каждой нечетной строке;
2) Определить сумму произведений сформированного массива и общее количество положительных элементов строк;
3) Определить минимальное и максимальное значения произведений и поменять местами строки исходной матрицы, в которых они найдены, если номера строк разные.

15. Дан двумерный массив B из N строк и N столбцов.

1) Сформировать массив из квадратов, элементов и квадратных корней положительных элементов диагоналей массива;
2) Определить сумму квадратов и элементов из сформированного массива;
3) Определить минимальное и максимальное значения элементов исходной матрицы и поменять местами строки, в которых они найдены, если номера строк разные.

16. Дан двумерный массив Q из M строк и N столбцов.
1) Сформировать массив из минимальных значений каждого нечетного столбца массива и их индексов;
2) Найти произведение элементов каждого нечетного столбца массива;
2) Определить минимальное произведение и в исходном массиве поменять столбец, в котором он найден, с первым столбцом, если номера столбцов разные.

17. Дан двумерный массив T из M строк и N столбцов.
1) Сформировать массив из максимальных значений каждой четной строки массива и их индексов;
2) Найти сумму элементов каждой четной строки массива;
2) Определить максимальную сумму и в исходном массиве поменять строку, в которой она найдена, со второй строкой, если номера строк разные.

18. Дан двумерный массив B из N строк и N столбцов.
1) Сформировать массив из сумм, произведений и количеств отрицательных элементов диагоналей массива;
2) Определить минимальный элемент из нечетных столбцов исходного массива и его индексы;
3) Поменять в исходном массиве элементы прямой и обратной диагоналей.

19. Дан двумерный массив C из M строк и N столбцов.
Даны R и S.
1) Сформировать массив из:
а) последних попавшихся элементов каждой строки массива C, находящихся в пределах: $R < C[i, j] < S$;
б) их индексов в исходной матрице;
(если их нет, результат должен быть равен 0;)
2) определить произведение отобранных элементов и их количество;
3) определить минимальное и максимальное значения из отобранных элементов и их координаты и поменять их местами в исходной матрице.

20. Дан двумерный массив A из M строк и N столбцов.
1) Сформировать массив из минимальных и максимальных значений каждого четного столбца и их индексов;
2) Поменять местами минимальное и максимальное значения в каждом четном столбце исходной матрицы;
3) определить произведение отрицательных минимальных значений и сумму положительных максимальных.

21. Дан двумерный массив D из M строк и N столбцов.
1) Сформировать массив из:

а) произведений отрицательных элементов каждого четного столбца; если их нет, результат должен быть равен 0;
б) количества таких элементов в каждом четном столбце;
2) определить сумму произведений сформированного массива и общее количество отрицательных элементов строк;
3) определить минимальное и максимальное значения произведений и поменять местами столбцы исходной матрицы, в которых они найдены, если номера столбцов разные.

22. Дан двумерный массив X из M строк и N столбцов.
Даны K и F.
1) Сформировать массив из:
а) сумм элементов каждой строки, находящихся в пределах:
K < X[i, j] <= F
б) количества таких элементов в каждой строке;
2) определить произведение сумм сформированного массива и общее количество отобранных элементов строк;
3) определить минимальное и максимальное значения сумм и поменять местами строки исходной матрицы, в которых они найдены, если номера строк разные.

Варианты повышенной сложности

31. Дан двумерный массив C из M строк и N столбцов.
1) Сформировать трехмерный массив из всех отрицательных значений четных строк и их индексов в исходной матрице;
2) определить сумму отобранных элементов массива и их количество;
3) определить минимальное и максимальное значения из отобранных элементов и их координаты и поменять местами столбцы исходной матрицы, в которых они найдены, если номера столбцов разные.

32. Дан двумерный массив C из M строк и N столбцов.
1) Сформировать трехмерный массив из всех положительных значений нечетных столбцов и их индексов в исходной матрице;
2) определить произведение отобранных элементов массива и их количество;
3) определить минимальное и максимальное значения из отобранных элементов и их координаты и поменять местами строки исходной матрицы, в которых они найдены, если номера строк разные.

33. Дан двумерный массив C из M строк и N столбцов.
Даны Z и X.
1) Сформировать трехмерный массив из всех элементов матрицы, находящихся в пределах: Z < C[i, j] < X ;
и их индексов в исходной матрице;
2) определить сумму отобранных элементов массива и их количество;
3) определить минимальное и максимальное значения из отобранных элементов и их координаты и поменять местами столбцы исходной матрицы, в которых они найдены, если номера столбцов разные.

34. Дан двумерный массив C из M строк и N столбцов.
Даны A и Y.

1) Сформировать трехмерный массив из всех элементов нечетных строк матрицы, находящихся в пределах: A < C[i, j] < Y ;
и их индексов в исходной матрице;
2) определить произведение отобранных элементов массива и их количество;
3) определить минимальное и максимальное значения из отобранных элементов и их координаты и поменять местами строки исходной матрицы, в которых они найдены, если номера строк разные.

3.7. Контрольные вопросы

1. Что такое процедура и каково ее назначение?
2. Поясните параметры процедуры.
3. Объясните механизм передачи параметров.
4. Поясните, что общего и в чем отличие параметров-значений, параметров-переменных и параметров-констант.
5. Правила согласования параметров.
6. Какие бывают объекты в подпрограммах?
7. Что такое область видимости объекта?
8. Что такое опережающее описание подпрограмм и для чего оно применяется?
9. Рабочие переменные подпрограмм, что это и в каких случаях используются.
10. Поясните состав параметров (что должно передаваться в параметрах, что нельзя передавать в параметрах, что рекомендуется/не рекомендуется передавать в параметрах).
11. Поясните вызов процедур.

4. Лабораторная работа №6

Вычисление значений сумм ряда с помощью функции

4.1. Цель работы

Целью данной лабораторной работы является изучение и освоение:

- строения функций;
- правил формирования и использования процедурных параметров;
- рекурсивных функций;
- правил вызова функций;
- отличий функций от процедур;
- использования операторов цикла с предусловием и с постусловием.

4.2. Задание на выполнение работы

Разработать программу для вычисления значений суммы ряда в соответствии со своим вариантом для десяти значений X шестью способами:

1. с использованием стандартной функции (возможно, потребуются дополнительные математические преобразования);
2. с использованием рекуррентной формулы и цикла с предусловием;
3. с использованием рекуррентной формулы и цикла с постусловием;
4. с использованием передаваемой функции для вычисления каждого члена ряда и цикла с предусловием;
5. с использованием передаваемой функции для вычисления каждого члена ряда и цикла с постусловием;
6. с использованием функции рекурсивного суммирования.

4.3. Требования к выполнению работы

- В программе необходимо использовать процедуры-функции с параметрами.
- Исходное значение X и шаг изменения X вводить.
- Точность можно задать с помощью константы или в операторе присваивания.
- Использование процедурного параметра обязательно.
- Для вычисления факториала и X в степени N использовать рекурсивную функцию.
- Алгоритмы в виде схем должны быть разработаны по всем процедурам, кроме рекурсивных. Для рекурсивных использовать словесное описание.

4.4. Краткие теоретические сведения

Функция

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. Функции обычно применяются в расчетных задачах, когда необходимо выполнить ряд вычислений и передать их значения основной программе.

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

Function Имя_функции (список формальных параметров) : тип_функции;

Здесь:

Имя_функции – идентификатор, по которому функция вызывается.

Список формальных параметров так же как и у процедуры 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, или оператор Result, который также может использоваться для передачи результата.

Пример функции с параметрами, вычисляющей факториал числа N:

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

Функция и процедура: принципиальные отличия

1. Заголовок процедуры не содержит тип идентификатора, у функции содержит.
2. В списке формальных параметров функции нет параметров-переменных (нет необходимости и не рекомендуется, можно использовать параметры-константы).
3. В теле функции есть оператор, присваивающей ей значение (идентификатору) для возвращения результата (или оператор Result).
4. Результат функции простого типа (скаляр).
5. Вызов процедуры – отдельный оператор, функция вызывается из другого оператора (например, Write, присваивания, условного и т.п.)
6. Для процедуры в схеме существует специальный блок.

Рекурсия

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.

Рекурсивная 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;
```

Здесь Halt – процедура прерывания программы. Необходима для «отсечения» недопустимых исходных данных, переданных в функцию. В зависимости от условия задачи она может быть заменена на процедуру Exit – выход из процедуры, при котором вся программа не прерывается.

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:

```
Function SUM ( N: integer; X: TMAS): real;
 Var i : byte;
 Begin
     S:=0;
     For  i:= 1 to N do S:= S + X[i];
 End;
```

и через рекурсивную функцию

```
Function SUM ( N: integer; X: TMAS): real;
      Var i : byte;
     Begin
```

```
        if N < 1 then SUM := 0
                 else  SUM := X[N] + SUM ( N-1, X );
    End;
```

Вычисление переменной X в степени N можно записать в следующем виде:

```
Function STEP (X: real; N: integer) : real;
    Begin
        if N<0 then begin
            Writeln (' Значение N отрицательно' ); Halt; end;
       if X = 0 then STEP := 0
                 else if N = 0 then STEP := 1
                               else if N = 1 then STEP := X
                                             else
                               STEP := X * STEP ( X, N-1 );
    end;
```

Тогда суммирование N членов бесконечного сходящегося ряда может выглядеть, например, 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.

Процедурный тип

Тип, предназначенный для хранения ссылок на процедуры или функции, называется процедурным, а переменная такого типа - процедурной переменной. Основное назначение процедурных переменных - хранение и косвенный вызов действий (функций) в ходе выполнения программы и передача их в качестве параметров.

Описание процедурного типа совпадает с заголовком соответствующей процедуры или функции без имени. Например:

```
type
  Proc = procedure (i: integer);
  Func = function (x,y: integer): integer;
```

Процедурной переменной можно присвоить процедуру или функцию с совместимым типом, например:

```
function Proiz(x,y: integer): integer;
    begin
         Result := x*y;
    end;

var f: Func := Proiz;
```

Аналогично происходит присваивание при передаче параметров.

4.5. Пример выполнения работы

Рассмотрим вычисление шестью способами десяти значений функции e^x. Формула для вычисления через сходящийся ряд имеет вид:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots для |x| < \infty$$

Диапазон значений x выберем от -1 с шагом 0,2 (до 0,8). Для вычислений используется встроенная функция exp (x) и пять пользовательских функций. При этом две из них выполняют вычисление на основе рекуррентной формулы, никаких других функций не вызывают. А другие три используют дополнительную функцию для определения очередного члена ряда (FunExp), которую получают в параметрах. FunExp в свою очередь вызывает две другие функции для вычисления x^N и факториала N. Обе функции являются рекурсивными. При этом Func5 сама является рекурсивной. Такое количество вызовов функций может приводить к переполнению стека. В этом случае рекомендуется увеличить значение точности вычислений функции (члена ряда функции).

```
program riad;
   Type Func = Function (X: real; N: integer) : real;
   Var X,Eps, DX, Y:real;
                  i:integer;

    // Вычисление факториала
    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;

      // Вычисление X в степени N
      Function STEP (X: real; N: integer) : real;
      Begin
           if N<0 then begin
            Writeln (' Значение N отрицательно' ); Halt; end;
        if X = 0 then STEP := 0
                 else if N = 0 then STEP := 1
                               else if N = 1 then STEP := X
                                             else
                               STEP := X * STEP ( X, N-1 );
      end;

      // Вычисление очередного члена ряда
     Function FunExp (X: real; N: integer) : real;
     begin
          FunExp := STEP (X,N)/FACT (N);
     end;

     {  Функция с использованием рекуррентной формулы и   while }
     Function  Func1 (x,e: real): real;
       var s,a : real;
           i: byte;
       begin
            a:=1; s:=0; i:=1;
            while abs (a) > e do begin
                s:=s+a;
                a:=a*x/i;
                i:=i+1;
                              end;
```

```
            func1:= s + a;
        end;

   {  Функция с использованием рекуррентной формулы и repeat  }
     Function  Func2 (x,e: real): real;
       var s,a : real;
           i: byte;
       begin
            a:=1; s:=0; i:=1;
            repeat
                s:=s+a;
                a:=a*x/i;
                i:=i+1;
            until abs (a) < e;
             func2:= s + a;
        end;

     {  суммирование с while и процедурным параметром }
     Function  Func3 (i: integer; x,e: real; F:Func): real;
       var s,a : real;
       begin
            s:=0; a:= F(X,i);
            while abs (a) > e do begin
                s:=s+a;
                i:=i+1;
                a:= F(X,i);
                                end;
            func3:= s + a;
        end;

     {  суммирование с repeat и процедурным параметром }
     Function  Func4 (i: integer; x,e: real; F:Func): real;
       var s,a : real;
       begin
            s:=0; a:= F(X,i);
            Repeat
                s:=s+a;
                i:=i+1;
                a:= F(X,i);
            Until   abs (a) < e ;
            func4:= s + a;
        end;

      {  рекурсивное суммирование с процедурным параметром }
     Function  Func5 (i: integer; x,e: real; F:Func): real;
       var a : real;
       begin
            a:= F(X,i);
            If abs (a) <= e then Func5 := a
                            else Func5 := a +Func5 (i+1, x, e, f);
        end;

       {  Основная программа }
       begin
        x:=-1; eps:= 1E-4; dx:=0.2;
        writeln
( '        i    x     While     Repeat    RecW      RecR      SumR     exp(x)');
        writeln
( '      ------------------------------------------------------------------');
       for i:= 1 to 10 do begin
            Y :=Func1 (x, eps);
            writeln (i:9,  x:6:1, y:10:5,
                     Func2 (x, eps):10:5,
                     Func3 (0,x, eps, FunExp):10:5,
                     Func4 (0,x, eps, FunExp):10:5,
                     Func5 (0,x, eps, FunExp):10:5,
```

```
                    exp(x):10:5);
        x:=x+dx;
                end;
end.
```

Результат работы программы представлен на листинге далее. Здесь i – номер итерации цикла, в котором вызываются все функции.

Окно вывода

i	x	While	Repeat	RecW	RecR	SumR	exp(x)
1	-1.0	0.36788	0.36788	0.36788	0.36788	0.36788	0.36788
2	-0.8	0.44933	0.44933	0.44933	0.44933	0.44933	0.44933
3	-0.6	0.54882	0.54882	0.54882	0.54882	0.54882	0.54881
4	-0.4	0.67031	0.67031	0.67031	0.67031	0.67031	0.67032
5	-0.2	0.81873	0.81873	0.81873	0.81873	0.81873	0.81873
6	0.0	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
7	0.2	1.22140	1.22140	1.22140	1.22140	1.22140	1.22140
8	0.4	1.49182	1.49182	1.49182	1.49182	1.49182	1.49182
9	0.6	1.82211	1.82211	1.82211	1.82211	1.82211	1.82212
10	0.8	2.22554	2.22554	2.22554	2.22554	2.22554	2.22554

4.6. Варианты на выполнение работы №6

1. $arctg\ x = \sum_{n=0}^{\infty} (-1)^n * \frac{x^{2*n+1}}{2*n+1} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$ Для $|X| < 1$

2. $arctg\left(\frac{1}{x}\right) = \sum_{n=0}^{\infty} (-1)^n * \frac{1}{(2*n+1)*x^{2*n+1}}$
$= \frac{1}{x} - \frac{1}{3*x^3} + \frac{1}{5*x^5} - \frac{1}{7*x^7} + \cdots$ Для $|X| > 1$

3. $arcth\ x = \sum_{n=0}^{\infty} \frac{x^{2*n+1}}{2*n+1} = x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \cdots$ Для $|X| < 1$

4. $arcth\ x = \sum_{n=0}^{\infty} \frac{1}{(2*n+1)*x^{2*n+1}} = \frac{1}{x} + \frac{1}{3*x^3} + \frac{1}{5*x^5} + \frac{1}{7*x^7} + \cdots$ Для $|X| > 1$

5. $\ln x = \sum_{n=1}^{\infty} (-1)^{n+1} * \frac{(x-1)^n}{n}$
$= \frac{(x-1)^1}{1} - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \frac{(x-1)^4}{4} + \cdots$ Для $0 < X < 2$

6. $\ln(1+x) = \sum_{n=0}^{\infty} (-1)^{n+1} * \frac{x^n}{n} = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots$ Для $-1 < X < 1$

7. $\ln(1-x) = -\sum_{n=0}^{\infty} \frac{x^n}{n} = -\left(x + \frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{4} + \cdots\right)$ Для $|X| < 1$

8. $\ln\left(\frac{1+x}{1-x}\right) = 2 * \sum_{n=0}^{\infty} \frac{x^{2*n+1}}{2*n+1} = 2 * \left(x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \cdots\right)$ Для $|X| < 1$

9. $\ln\left(\frac{x+1}{x-1}\right) = 2 * \sum_{n=0}^{\infty} \frac{1}{(2*n+1)*x^{2*n+1}}$

$= 2 * \left(\frac{1}{x} + \frac{1}{3*x^3} + \frac{1}{5*x^5} + \frac{1}{7*x^7} + \cdots\right)$ Для $|X| > 1$

10. $e^x * (1+x) = \sum_{n=0}^{\infty} \frac{x^n * (n+1)}{n!} = 1 + \frac{2*x}{1!} + \frac{3*x^2}{2!} + \frac{4*x^3}{3!} + \cdots$ Для $|X| < 2{,}4$

11. $e^{-x^2} = \sum_{n=0}^{\infty} (-1)^n * \frac{x^{2*n}}{n!} = \frac{x^0}{0!} - \frac{x^2}{1!} + \frac{x^4}{2!} - \frac{x^6}{3!} + \cdots$ Для $X < 1$

12. $\ln x = \sum_{n=1}^{\infty} \frac{(x-1)^n}{n*x^n} = \frac{x-1}{x} + \frac{(x-1)^2}{2*x^2} + \frac{(x-1)^3}{3*x^3} + \cdots$ Для $X > 0.5$

13. $\sin x = \sum_{n=1}^{\infty} (-1)^{n-1} * \frac{x^{2*n-1}}{(2*n-1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$ Для $|X| < \infty$

14. $\cos x = \sum_{n=0}^{\infty} (-1)^n * \frac{x^{2*n}}{(2*n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \cdots$ Для $|X| < \infty$

15. $sh\ x = \sum_{n=1}^{\infty} \frac{x^{2*n-1}}{(2*n-1)!} = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \cdots$ Для $|X| < 3$, $sh\ x = \frac{e^x - e^{-x}}{2}$

16. $ch\ x = \sum_{n=0}^{\infty} \frac{x^{2*n}}{(2*n)!} = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \frac{x^8}{8!} + \cdots$ Для $|X| < 3$

17. $\sin^2 x = \sum_{n=1}^{\infty} (-1)^{n+1} * \frac{2^{2*n-1} * x^{2*n}}{(2*n)!}$

$= \frac{2^1 * x^2}{2!} - \frac{2^3 * x^4}{4!} + \frac{2^5 * x^6}{6!} - \frac{2^7 * x^8}{8!} + \cdots$ Для $|X| < 1$

18. $\cos^2 x = 1 - \sum_{n=1}^{\infty} (-1)^{n+1} * \frac{2^{2*n-1} * x^{2*n}}{(2*n)!}$

$= 1 - \left(\frac{2^1 * x^2}{2!} - \frac{2^3 * x^4}{4!} + \frac{2^5 * x^6}{6!} - \frac{2^7 * x^8}{8!} + \cdots\right)$ Для $|X| < 1$

19. $\sin^3 x = \frac{1}{4} \sum_{n=1}^{\infty} (-1)^{n+1} * \frac{3^{2*n+1} - 3}{(2*n+1)!} * x^{2*n+1}$

$= \frac{1}{4} * \left(\frac{3^3 - 3}{3!} * x^3 - \frac{3^5 - 3}{5!} * x^5 + \frac{3^7 - 3}{7!} * x^7 - \cdots\right)$ Для $X < 1$

20. $\cos^3 x = \frac{1}{4} \sum_{n=0}^{\infty} (-1)^n * \frac{3^{2*n} + 3}{(2*n)!} * x^{2*n}$

$= \frac{1}{4} * \left(\frac{3^0 + 3}{0!} * x^0 - \frac{3^2 + 3}{2!} * x^2 + \frac{3^4 + 3}{4!} * x^4 - \cdots\right)$ Для $X < 1$

21. $arctg\ x = \frac{\pi}{2} + \sum_{n=0}^{\infty} (-1)^{n+1} * \frac{x^{2*n+1}}{2*n+1} = \frac{\pi}{2} - \left(x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots\right)$ Для $|X| < 1$

22. $arctg\ x = \frac{\pi}{2} + \sum_{n=0}^{\infty} (-1)^{n+1} * \frac{1}{(2*n+1)*x^{2*n+1}}$

$= \frac{\pi}{2} - \left(\frac{1}{x} - \frac{1}{3*x^3} + \frac{1}{5*x^5} - \frac{1}{7*x^7} + \cdots\right)$ Для $|X| > 1$

23. $arctg\ x = -\frac{\pi}{2} + \sum_{n=0}^{\infty} (-1)^{n} * \frac{1}{(2*n+1)*x^{2*n+1}}$

$= -\frac{\pi}{2} + \left(\frac{1}{x} - \frac{1}{3*x^3} + \frac{1}{5*x^5} - \frac{1}{7*x^7} + \cdots\right)$ Для $X < -1$

4.7. Контрольные вопросы

1. Что такое функция и каково ее назначение?
2. Поясните параметры функции.
3. Как передается результат функции?
4. Каким образом программировать рекуррентную формулу?
5. Каким образом программировать сумму бесконечного сходящегося ряда?
6. Поясните состав параметров функции.
7. Поясните разницу между процедурой и функцией.
8. Что общего между процедурой и функцией?
9. Вызов функции.
10. Что такое рекурсивная функция?
11. Типы рекурсивных функций.
12. Поясните использование процедурного типа.
13. Поясните передачу функций в параметрах.
14. В чем ключевое отличие между циклами с предусловием и с постусловием?
15. Поясните обозначение функции на схеме алгоритма.

5. Лабораторная работа №7

Разработка программ для обработки массивов записей

5.1. Цель работы

Целью лабораторной работы является освоение:

- описание записей и массивов записей;
- обращения к элементам записей и массивов записей;
- оператора With;
- типизированных массивов строк;
- алгоритмов поиска данных;
- опережающего описания функций.

5.2. Задание на выполнение работы

Разработать программу, которая должна включать процедуры для:

1. вывода шапки таблицы;
2. вывода данных одной строки таблицы;
3. ввода данных массива записей;
4. вывода данных массива записей;

5. фильтрации строковых данных, удаляющую пробелы в начале и в конце строки (можно использовать функцию);
6. арифметической обработки одного из полей массива записей (в соответствии со своим вариантом) (можно использовать функцию);
7. смену местами двух элементов массива записей (строк) по запросу пользователя (номера строк вводить с экрана);
8. поиска и вывода требуемых элементов массива записей (строки массива) по заданному варианту поиска:
   - по одному заданному уникальному признаку;
   - по любому сочетанию нескольких заданных поисковых признаков (поиск не уникальный).

5.3. Требования к выполнению

- Состав шапки таблицы и исходные данные для заполнения таблицы определяете самостоятельно в соответствии с вариантом.
- В массиве должно быть 5-7 записей.
- Критерии для поиска выбираете самостоятельно.
- Шапка таблицы задается с использованием типизированной константы.

5.4. Краткие теоретические сведения

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

```
VAR MZ : array [ 1 .. M ] of TZ;
```

Обратиться полям записи можно через составное имя (иногда называют «оператор точка»). Например, в нашем случае для ввода всех полей записи имя записи указывается перед именем каждого поля:

```
Readln ( Z.NOM , Z.RS ,  Z.PAS , Z.FIO , Z.SB );
```

В случае же с массивом записей необходимо также указывать и номер элемента (как в обычном массиве), поэтому запись еще удлиняется:

```
Readln ( Z[i].NOM , Z[i].RS ,  Z[i].PAS , Z[i].FIO , Z[i].SB );
```

Если же поле записи само является массивом, то могут появиться два и более индексов. При этом размещение индексов может быть, например, таким: Z[i].FIO[1] (первая буква фамилии), а может и применяться группировка Z.FIO [i,1].

Обращение к полям записи получается довольно громоздким, поэтому в случае многократного повторения имени записи (возможно и номера элемента массива) удобнее обращаться с использованием оператора присоединения with.

Пример ввода массива записей с использованием оператора присоединения:

```
for i :=1 to M do
          with MZ [ i ] do begin
               readln (  NOM , RS , PAS ,  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 выглядит следующим образом:

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

Записи обычно вводят из файла, зачастую их удобно размещать на одной строке. Рекомендуется для удобства формирования файла с данными и работы с ним между полями записи оставлять хотя бы один пробел. При этом в случае с арифметическими данными пробел ни на что не влияет, а строковые и символьные данные будут вводить пробел и его сохранять (т.е. нужно объявлять на один символ больше). Рекомендуется использовать символьную переменную для ввода пробела. Оператор ввода примет вид:

```
readln ( MZ [ i ].NOM , MZ [ i ].RS , b , MZ [ i ].PAS , b ,
         MZ [ i ].FIO , MZ [ i ].SB );
```

В данном примере b - символьная переменная, которая в дальнейшем нигде не используется.

Над полями записи можно производить манипуляции в соответствии с объявленным типом поля. Например, арифметические складывать, перемножать и т.п.

Тип поля может быть любого типа, в том числе и сложным. Объявлять такие записи нужно, начиная с простых (вложенных типов). При этом рекомендуется по возможности больше использовать типы, определенные пользователем (массивы, вложенные записи).

Поиск

Одно из наиболее часто встречающихся в программировании действий – поиск. Существует несколько основных алгоритмов поиска и целый ряд алгоритмов позволяющих сделать этот процесс наиболее эффективным.

При дальнейшем рассмотрении исходим из допущения, что группа данных, в которой необходимо отыскать заданный элемент, фиксирована. Процесс поиска заключается в выборе элемента из последовательности элементов в соответствии с ключом. Если ключ входит в состав данных, среди которых производится поиск, то он называется внутренним, в противном случае – внешним. Ключ внешний - относительная позиция записи внутри файла или таблица ключей. Ключ может быть простым или сложным (составным).

К основным методам поиска относятся:

- Линейные поиск (последовательный),
- Бинарный поиск (двоичный, делением пополам),
- Индексно-последовательный поиск,
- Поиск строки,
- Поиск по дереву.

Поиск может быть внутренний и внешний. Критерием является размещение данных, среди которых производится поиск: если они размещены в оперативной памяти, то поиск внутренний, а если на внешнем носителе – внешний.

Алгоритм поиска – это алгоритм, который воспринимает некоторый аргумент A и пытается локализовать некоторую запись, ключ которой равен A.

Успешный поиск называется извлечением.

В настоящем пособии рассматривается только линейный поиск.

Если нет никакой дополнительной информации о разыскиваемых данных, то очевидный подход – простой последовательный просмотр массива с увеличением шаг за шагом той его части, где желаемого элемента не обнаружено. Такой метод называется линейным поиском.

Линейный поиск применяется к таблице, которая организована или как массив, или как связанный список (или к бинарному файлу при внешнем поиске).

Алгоритм линейного поиска – это простой последовательный просмотр массива с увеличением шаг за шагом той его части, в которой не оказалось желаемого элемента.

Условия окончания поиска таковы:

- элемент найден,
- весь массив просмотрен и совпадений не обнаружено.

Это дает нам алгоритм:

```
For i:=1 to N do
      if A[i]=key then begin
           Write (A[i]); Break;
                        end;
```

Если искомый элемент найден, то он будет напечатан (или может быть выполнено другое необходимое действие). Если искомый элемент не уникален, то прерывать работу цикла нельзя (т.е. использовать Break), а в некоторых случаях приходится использовать специальный признак (PR), который говорил бы о том, что поиск был успешен:

```
PR:=false;
For i:=1 to N do
      if A[i]=key then begin
          Write (A[i]); PR:=true;
                        end;
```

Таким образом можно говорить о том, что существуют два типа линейных поисков: с уникальным поисковым признаком (поиск прерывается после обнаружения первого же элемента) и неуникальным (поиск продолжается до конца таблицы, а найденные элементы фиксируются (подсчитывается их количество, выводятся на печать, сохраняются в другой структуре и т.п.), или успешность поиска отражается в специальном поисковом признаке.

Рассмотрим соответствующие алгоритмы на примере массива записей, объявленном в начале раздела 5.4.

Поиск по уникальному признаку

Для поиска по уникальному признаку выберем поле «FIO», предположив, что именно оно повторяться не может. Безусловно, это «натяжка» для примера, но в реальных задачах уникальные поля встречаются очень часто. На рисунке 2 представлен алгоритм, где видно, что после того, как произошло извлечение (вызвана процедура печати одной строки P(I);), цикл поиска прерывается. Если необходимо поиск повторить, то все операторы могут быть вызваны повторно (например, из цикла до конца файла с данными для поиска).

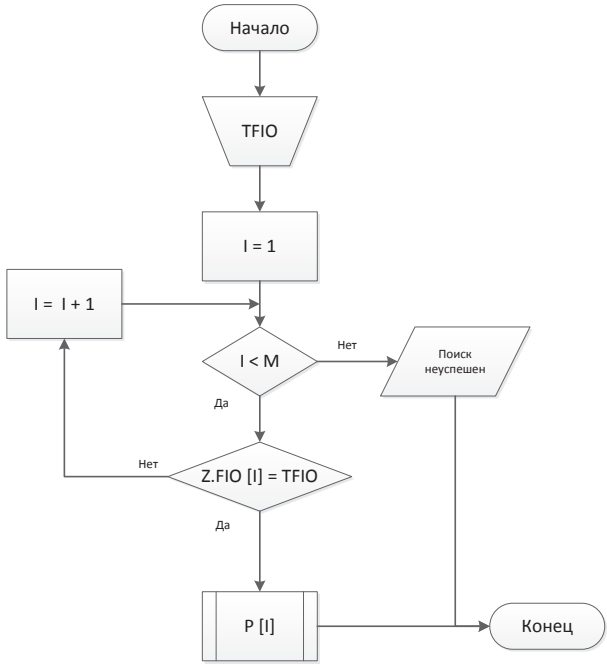


Рисунок 2. Алгоритм поиска по уникальному поисковому признаку

Поиск по неуникальному признаку

Для поиска по неуникальному признаку также выберем поле «FIO», предположив, что оно повторяться может. На рисунке 3 представлен алгоритм, где видно, что после того, как произошло извлечение (вызвана процедура печати одной строки P(I);), цикл поиска не прерывается. Здесь PR – признак успешности поиска. После того, как произошло извлечение, признак получает значение TRUE. По окончании цикла поиска признак анализируется. Если значение TRUE он не принял, значит ничего найдено не было.

Вместо признака можно использовать счетчик. Начальное значение счетчика равно 0, с каждой найденной записью он увеличивается на 1, по окончании цикла мы знаем количество извлечений (найденных записей). Если он остался равен 0, то поиск был неуспешен.

Поиск универсальный (по сочетанию поисковых признаков)

Частным случаем поиска по неуникальному признаку является поиск по любому сочетанию поисковых признаков. Теоретически он может быть и уникальным, но как правила, в этом случае лишено смысла его применение.

Рассмотрим на примере двух полей: фамилии и номера рейса. Количество полей может быть любым разумным, с каждым полем увеличивается сложность логического выражения в условных операторах.

Сначала проверим наличие данных для поиска. Проверка в этом случае необходима, что далее будет видно из программы в разделе 5.5.

```
if ( tfio = '' ) and ( trs = 0 ) then
                  writeln ( FR , ' Нет данных для поиска .' , #10);
```

Далее в цикле поиска логическое выражение в условном операторе примет вид:

```
if (( tfio = '' ) or ( tfio = FIO )) and
                        (( trs = 0  ) or ( trs = RS ))
                       then …
```

В этом выражении строковый поисковый признак либо не учитывается (строка пуста), либо проверяется на соответствие полю записи; также и арифметический поисковый признак, но он пустым быть не может, в качестве критерия его отсутствия выбирается какое-то значение, например, 0. В остальном алгоритм процедуры не отличается от представленного на рисунке 3.

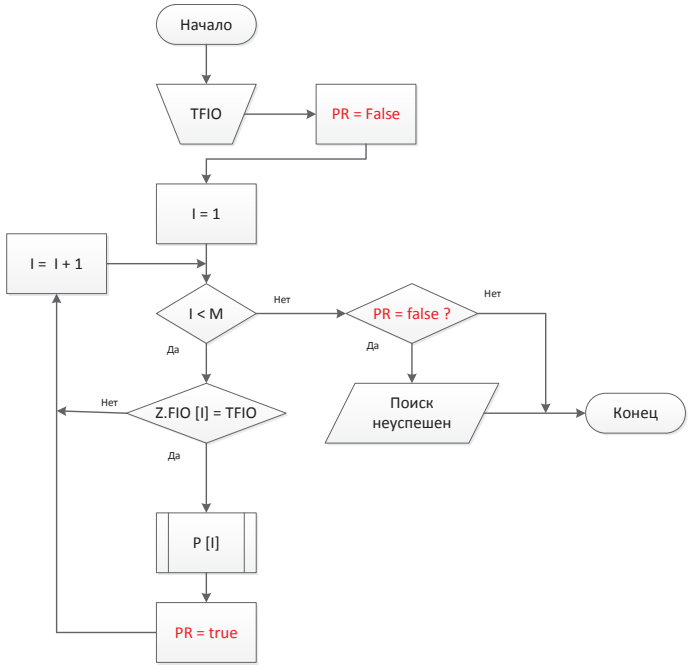


Рисунок 3. Алгоритм поиска по неуникальному поисковому признаку

5.5. Пример выполнения работы

Рассмотрим пример для массива записей, объявленного в начале раздела 5.4. Данные в файле размещены построчно:

Program1.pas* | C_MODUL_POSOB.pas | •LR7_Posobie.pas* | LR7.RES | LR7.DAT

```
 1    1 1625 77 45 639187      Иванов И.И.       99.53
 2    4 1341 99 46 173855      Петров П.П.        102.14
 3    9 2100 88 47 250000      Егорова А.А.         94.36
 4   25 2100 44 49 439608      Сидоров С.С.       115.08
 5   31 3141 11 48 304815      Кузнецов В.Л.        127.00
```

Программа:

```
     { ----- Полная программа лабораторной работы ----- }
PROGRAM LR7;
{$R+}
CONST M = 5;
TYPE TSH = array [ 1 .. 7 ] of string [ 76 ];
     S18 = string [ 18 ];
```

```
      TZ  = record
               NOM : byte;                { номер билета }
               RS : word;                 { рейс }
              PAS : string [ 15 ];        { паспорт }
              FIO : string [18];          { фио пассажира  }
               SB : real;                 { стоимость билета }
              end;
       ttz= array [ 1 .. M ] of TZ;
 VAR i : byte;
     Z : TZ;
     MZ : ttz;
     S : real;
     FI, FR : text;
 CONST SH : TSH = (
'                  Заголовок таблицы                                          ',
'============================================================================ ',
'|   Номер  |   Рейс  |     Паспорт     |       Фамилия         | Стоимость  | ',
'|пассажира |         |     пассажира   |      пассажира        | билета, $  | ',
'============================================================================',
'----------------------------------------------------------------------------
',
'============================================================================'
) ;
       { ----- Шапка таблицы ----- }
 PROCEDURE PSH;
 var i : byte;
 Begin
 for i := 1 to 5 do writeln ( FR , SH [ i ] );
 End;
       { -----Печать ----- }
 PROCEDURE PRINT (i: integer);
 Begin with MZ [ i ] do
            writeln ( FR , '|' , NOM : 8 , '|' : 3 , RS : 7 , '|' : 3 ,
                   PAS : 15 , '|' : 3 , ' ' : 2 , FIO ,
                      '|' : 21 - length ( FIO ) , SB : 10 : 2 , '|' : 3 );
 End ;

              { ----- Фильтр ----- }
 FUNCTION FILTR ( st : s18 ) : s18;   FORWARD;    { будет далее }

      { ----- Ввод и сумма ----- }
 PROCEDURE VVOD;
 var b : char; i : integer;
 Begin assign ( FI , 'lr7.dat' ); reset ( FI );
       PSH;
       S := 0;
       for i :=1 to M do begin
           with MZ [ i ] do begin
                readln ( FI , NOM , RS , b , PAS , b , FIO , SB );
                S := S + SB;
                FIO := FILTR ( FIO );
                PRINT (i);
                if i = M then writeln ( FR , SH [ 7 ] )
                         else writeln ( FR , SH [ 6 ] );
                   end; end;
       writeln ( FR , ' суммарная стоимость = ',
                      S : 10 : 2 , ' $ ' );
       close ( FI );
 End;

 { ----- Поиск по фамилии ----- }
     PROCEDURE POISK;
 var
     tfio : S18;
       i : integer;
 Label MF;
```

```
Begin writeln ( FR , ' Поиск по фамилии ' );
      assign ( FI ,'lr7poisk-F.dat' ); reset ( FI );
      repeat
         readln ( FI , tfio  );
         tfio := FILTR ( tfio );
         if ( tfio = '' ) then begin
                       writeln ( FR , ' Нет данных для поиска ');
                       goto MF;
                                  end;
         writeln ( FR , ' Найти фамилию : ', '''' ,tfio,'''');
            for i := 1 to M do
                     with MZ [ i ] do
                           if (tfio = FIO)  then begin
                                    PRINT (i); goto MF;
                                      end;
         writeln ( FR , ' Фамилии ', tfio, ' нет ' );
      MF:until eof ( FI );
      close ( FI );
      writeln ( FR ,  ' Поиск по фамилии окончен. ' );
End ;

{ ----- Поиск по номеру рейса ----- }
     PROCEDURE POISK_RS;
var  trs : word;
      pr : boolean;
       i : integer;

Begin writeln ( FR , ' Поиск по номеру рейса ' );
      assign ( FI ,'lr7poisk-RS.dat' ); reset ( FI );
      repeat
         readln ( FI , trs  );
         pr := false;
      writeln ( FR , ' Найти пассажиров с рейса : ', trs);
            for i := 1 to M do
                     with MZ [ i ] do
                           if trs = RS then begin
                                    PRINT (i);  pr := true;
                                      end;
        if not pr then writeln ( FR , ' Рейса ', trs, ' нет ' );
      until eof ( FI );
      close ( FI );
      writeln ( FR ,  ' Поиск по номеру рейса окончен. ' );
End ;
          { ----- Универсальный поиск  ----- }
PROCEDURE POISK_U;
var  trs : word;
    tfio : S18;
      pr : boolean;
    i, R : integer;
Begin writeln ( FR ,  ' Поиск сведений ', #10 ,
              ' по любому сочетанию поисковых признаков ' );
      assign ( FI ,'lr7poisk.dat' ); reset ( FI );
      repeat
            readln ( FI , tfio , trs );
            tfio := FILTR ( tfio );
            writeln ( FR , ' Найти : ', #10 ,
                        ' требуемый рейс  : ', trs , #10 ,
                        ' фамилия пассажира  : ', '''', tfio , '''' );
    { Анализ наличия данных для поиска }
            if ( tfio = '' ) and ( trs = 0 ) then begin
               writeln ( FR , ' Нет данных для поиска .' , #10);
               continue;
                                                  end;
            pr := false; R:=0;
            for i := 1 to M do
              with MZ [ i ] do
```

```
                if (( tfio = '' ) or ( tfio = FIO )) and
                       (( trs = 0  ) or ( trs = RS ))
                       then begin
                                          PRINT (i);
                            pr := true;
                            R:=R+1;
                            end;
          if not pr then
               writeln ( FR , ' Ничего не найдено. ', #10 )
                     else
                  writeln ( FR , ' Найдено  ', R, ' записей.', #10 );
       until eof ( FI );
       close ( FI );
       writeln ( FR , ' Поиск завершен. ', #10 );
End ;

FUNCTION FILTR( st : s18 ) : s18;
Var i , j , l : integer;
Begin
     l := length ( st );
     for i := 1 to l do
         if st [ i ] <> ' ' then
            for j := l downto i do
                if st [ j ] <> ' ' then begin
                   FILTR := copy ( st , i , j - i + 1 );
                   exit;                  end;
     FILTR := '' ;
End;

       { ----- Основная программа  ----- }
BEGIN
      Assign ( FR ,'lr7.res' ); Rewrite ( FR );
          VVOD; writeln ( FR );
          POISK; writeln ( FR );
          POISK_RS; writeln ( FR );
          POISK_U; writeln ( FR );
      { ----- Смена строк местами  ----- }
      Z := MZ [ 1 ]; MZ [ 1 ] := MZ [ M ]; MZ [ M ] := Z;
      Writeln ( FR ,  ' После смены строк : ' );
      PSH;
      For i := 1 to M do  begin
          PRINT (i);
          if i = M then writeln ( FR , SH [ 7 ] )
                   else writeln ( FR , SH [ 6 ] );
                        end;
      Close ( FR );
END.
```

Данные для поиска представлены в трех файлах:

Поиск по фамилии	Егорова А.А. Упырев Ю.Ю. Кузнецов В.Л.
Поиск по номеру рейса	2100 777 3141
Поиск по сочетанию признаков	Егорова А.А. 2100 2100 0 Петров П.П. 0 Упырев Ю.Ю. 3141

В данных предусмотрены и варианты неуспешных поисков.

Результат работы программы (без смены строк местами):

Program1.pas* | C_MODUL_POSOB.pas | •LR7_Posobie.pas* | **LR7.RES** | LR7.DAT

```
                    Заголовок таблицы
=============================================================================
|   Номер   |   Рейс   |      Паспорт      |        Фамилия          | Стоимость  |
|пассажира  |          |     пассажира     |       пассажира         | билета, $  |
=============================================================================
|       1   |   1625   |77 45 639187       |  Иванов И.И.            |      99.53 |
-----------------------------------------------------------------------------
|       4   |   1341   |99 46 173855       |  Петров П.П.            |     102.14 |
-----------------------------------------------------------------------------
|       9   |   2100   |88 47 250000       |  Егорова А.А.           |      94.36 |
-----------------------------------------------------------------------------
|      25   |   2100   |44 49 439608       |  Сидоров С.С.           |     115.08 |
-----------------------------------------------------------------------------
|      31   |   3141   |11 48 304815       |  Кузнецов В.Л.          |     127.00 |
=============================================================================
 суммарная стоимость =      538.11 $

 Поиск по фамилии
 Найти фамилию : 'Егорова А.А.'
|       9   |   2100   |88 47 250000       |  Егорова А.А.           |      94.36 |
 Найти фамилию : 'Упырев Ю.Ю.'
 Фамилии Упырев Ю.Ю. нет
 Нет данных для поиска
 Найти фамилию : 'Кузнецов В.Л.'
|      31   |   3141   |11 48 304815       |  Кузнецов В.Л.          |     127.00 |
 Поиск по фамилии окончен.

 Поиск по номеру рейса
 Найти пассажиров с рейса : 2100
|       9   |   2100   |88 47 250000       |  Егорова А.А.           |      94.36 |
|      25   |   2100   |44 49 439608       |  Сидоров С.С.           |     115.08 |
 Найти пассажиров с рейса : 777
 Рейса 777 нет
 Найти пассажиров с рейса : 3141
|      31   |   3141   |11 48 304815       |  Кузнецов В.Л.          |     127.00 |
 Поиск по номеру рейса окончен.

 Поиск сведений
 по любому сочетанию поисковых признаков
 Найти :
 требуемый рейс  : 2100
 фамилия пассажира  : 'Егорова А.А.'
|       9   |   2100   |88 47 250000       |  Егорова А.А.           |      94.36 |
 Найдено  1 записей.

 Найти :
 требуемый рейс  : 2100
 фамилия пассажира  : ''
|       9   |   2100   |88 47 250000       |  Егорова А.А.           |      94.36 |
|      25   |   2100   |44 49 439608       |  Сидоров С.С.           |     115.08 |
 Найдено  2 записей.

 Найти :
 требуемый рейс  : 0
 фамилия пассажира  : ''
 Нет данных для поиска .

 Найти :
 требуемый рейс  : 0
 фамилия пассажира  : 'Петров П.П.'
|       4   |   1341   |99 46 173855       |  Петров П.П.            |     102.14 |
 Найдено  1 записей.

 Найти :
 требуемый рейс  : 3141
 фамилия пассажира  : 'Упырев Ю.Ю.'
 Ничего не найдено.

 Поиск завершен.
```

5.6. Варианты на выполнение работы №7

1. Даны сведения о N книгах известных фантастов в составе:
   — Название книги,
   — Автор,
   — Год выхода книги,
   — Тираж,
   — Название фильма (если была экранизация),
   — Дата выхода фильма.

Найти книгу, имеющую самый большой тираж.

2. Даны сведения о N модельерах в составе:
   — ФИО,
   — Страна,
   — Дом моды,
   — Количество «недель моды», в которых участвовал,
   — Средняя стоимость одного изделия.

Найти модельера, который участвовал в наибольшем количестве «недель моды».

3. Даны сведения о N немецких автомобилях в составе:
   — Марка,
   — Модель,
   — Объем двигателя,
   — Количество дверей,
   — Тип кузова,
   — Стоимость в минимальной комплектации.

Определить самый дорогой автомобиль.

4. Даны сведения о N роботах в составе:
   — Название,
   — Область применения,
   — Страна,
   — Год производства,
   — Производитель,
   — Стоимость.

Найти самого старого робота.

5. Даны сведения о N великих людях России в составе:
   — ФИО,
   — Годы жизни,
   — Место рождения,
   — Кто он,
   — Чем знаменит.

Найти человека, прожившего наиболее долгую жизнь.

6. Даны сведения о N книгах библиотеки в составе:
   — Название,
   — Автор,
   — Жанр,

— Библиотечный шифр,
— Количество экземпляров,
— Стоимость одного экземпляра.

Найти общую стоимость по каждой книге (Стоимость одного экземпляра* Количество экземпляров).

7. Даны сведения о N великих шахматистов в составе:
— ФИО,
— Годы жизни,
— Страна,
— Максимальное достижение (чемпион мира, страны и т.п.),
— В какой части партии наиболее силен.

Найти продолжительность жизни (или возраст) каждого шахматиста.

8. Даны сведения о N административных округах Москвы в составе:
— Административный округ (название),
— Площадь, км²,
— Население, чел.,
— ФИО главы,
— Количество районов.

Найти плотность населения (чел./км²) каждого округа.

9. Даны сведения о N современных хореографах в составе:
— Фамилия, имя
— Страна,
— Стиль, в котором работает,
— Год рождения,
— Возглавляемый коллектив (если есть).

Определить средний возраст хореографов.

10. Даны сведения о N фотоаппаратах в составе:
— Производитель;
— Модель,
— Стоимость,
— Матрица (параметры),
— Фокусное расстояние,
— Страна-производитель.

Найти фирму с наибольшим объемом продукции.

11. Даны сведения о N художниках в составе:
— ФИО,
— Годы жизни,
— Жанр, в котором работал,
— Количество работ,
— Самая знаменитая работа,
— Ее место нахождения (музей).

Найти художника, прожившего самую короткую жизнь.

12. Даны сведения о N спортсменах-легкоатлетах в составе:

— ФИО,
— Год рождения,
— Специализация (бег, прыжки и т.п.),
— ФИО тренера,
— Команда,
— Место в последнем чемпионате,
— Годовой бюджет .

Найти самого молодого и самого возрастного спортсмена.

13. Даны сведения о N компьютерных играх в составе:
— Название,
— С какого года проводится,
— Производитель,
— Количество игроков,
— Жанр.

Определить самую старую игру.

14. Даны сведения о N программах компьютерной графики в составе:
— Название,
— Фирма-производитель,
— Современная версия,
— Тип графики,
— Стоимость.

Определить среднюю стоимость программ (бесплатные не учитывать).

15. Даны сведения о N музыкальных группах в составе:
— Название,
— Год создания,
— Солист,
— Количество человек в группе,
— Стиль.

Найти самую старую группу.

16. Даны сведения о N вестернах в составе:
— Название,
— Режиссер,
— Год выпуска,
— Бюджет,
— Кассовый сбор.

Найти фильм с самой маленькой разницей между бюджетом и кассовым сбором.

17. Даны сведения о N поэтах 19 века в составе:
— Фамилия, Имя,
— Годы жизни (год рождения, год смерти)
— Страна,
— Самое известное произведение.

Найти поэта, прожившего самую долгую жизнь.

18. Даны сведения о N карнавалах в составе:

— Место проведения (город, страна),
— С какого года проводится,
— Месяц (когда проводится),
— Продолжительность.

Определить самый продолжительный фестиваль.

19. Даны сведения о N мотоциклах в составе:
— Наименование (модель),
— Производитель (завод),
— Цена,
— Мощность,
— Объем бензобака.

Найти самый дорогой мотоцикл.

20. Даны сведения о N горах мира в составе:
— Название,
— Название самой высокой вершины,
— Высота самой высокой вершины,
— Материк,
— Площадь.

Найти горы с самой высокой вершиной.

21. Даны сведения о N картинах Третьяковской галереи в составе:
— Название,
— Автор,
— Год написания,
— Размеры,
— Стоимость.

Найти среднюю стоимость картин.

22. Даны сведения о N архитектурных шедеврах в составе:
— Название,
— Место (город и страна),
— Год построения,
— Архитектор.

Определить самый старый архитектурный шедевр.

23. Даны сведения о N странах Европы в составе:
— Название,
— Столица,
— Площадь,
— Население,
— Язык.

Найти страну с самой большой плотностью населения.

5.7. Контрольные вопросы

1. Что такое структурированные типы данных (записи)?
2. Как объявить запись на Паскале?

3. Как объявить массив записей на Паскале?
4. Как обратиться к элементам записи (массива записей)?
5. Что такое типизированная константа?
6. Как объявить типизированную константу – массив?
7. Поясните назначение и использование оператора WITH.
8. Что такое линейный поиск данных?
9. В чем разница между алгоритмами поиска по уникальному и неуникальному поисковым признакам?
10. Что такое поиск по сочетанию признаков и в чем его универсальность?
11. Как называется успешный результат поиска?

6. Лабораторная работа №8

Разработка программ сортировки данных

6.1. Цель работы

Целью работы является освоение методов внутренней сортировки данных.

6.2. Задание на выполнение работы

Разработать процедуры, которые должны:

1. выполнять сортировку двумерного арифметического массива, созданного в лабораторной работе №5 первым методом (в соответствии со своим вариантом):
   - четные варианты – по возрастанию,
   - нечетные варианты – по убыванию.

2. выполнить сортировку этого же массива вторым методом по строкам или столбцам (сортировать элементы строк или столбцов в соответствии со своим вариантом):
   - четные варианты – по строкам,
   - нечетные варианты – по столбцам.

3. выполнять сортировку массива записей, созданного в лабораторной работе №7 третьим методом по одному из строковых полей (в соответствии со своим вариантом):
   - четные варианты – по убыванию,
   - нечетные варианты – по возрастанию.

6.3. Требования к выполнению

- Все сортировки оформляются в виде процедур; при этом для второго задания могут использоваться две процедуры, так как вызывать процедуру сортировки придется несколько раз (для каждой строки или столбца);
- Вывод результатов сортировки должен находиться в вызывающей программе;
- Программы должны содержать вывод результатов сортировки (при этом после первой сортировки массив должен вернуться в исходное состояние);
- Программы должны быть хорошо структурированы, содержать комментарии.
- Алгоритмы должны быть представлены в виде графических схем.

6.4. Краткие теоретические сведения

Методы сортировки

Сортировка – это упорядочивание элементов. В случае, когда элемент – запись (имеет несколько полей), поле, служащее критерием порядка, называется ключом сортировки.

Существует множество алгоритмов сортировки, и регулярно появляются новые или модифицируются уже известные алгоритмы.

Результатом сортировки может быть как упорядоченный массив (множество записей упорядоченных по ключам), так и вектор порядковых номеров элементов множества, расстановка по которым даст сортированное множество.

Критериями оценки различных методов сортировки могут быть:

- Количество сравнений и пересылок записей,
- Время сортировки заданного объема данных,
- Требуемый объем памяти для сортировки,
- Сложность алгоритмов и программ сортировки.

Сортировки могут быть внутренние (массивов) и внешние (сортировка данных, размещенных на внешнем носителе информации, например, сортировка данных типизированного файла).

Алгоритмы внутренней сортировки можно отнести к различным типам:

Сортировка выборкой,

Сортировка включением,

Обменные сортировки,

Сортировка распределением,

Сортировка подсчетом,

сортировка слиянием.

В данном пособии рассматриваются не все алгоритмы, а только шесть классических алгоритмов внутренней сортировки, которые предлагаются в качестве варианта на выполнение лабораторной работы №8. На полноту рассмотрения всех существующих алгоритмов сортировки и тенденций в этой области настоящее пособие не претендует.

Рассмотрим алгоритмы подробнее. Программа, содержащая рассматриваемые алгоритмы, представлена далее. Методы сортировки оформлены в виде процедур с параметрами.

Простого выбора

Алгоритм простого выбора является одним из самых простых алгоритмов сортировки массива. Смысл его в том, чтобы идти по массиву, и каждый раз искать максимальны (минимальный) элемент массива, обменивая его с начальным элементом неотсортированной части массива.

Алгоритм:

1. Находим максимальный/минимальный элемент в массиве.
2. Меняем местами максимальный/минимальный и последний/первый элемент местами.
3. Снова ищем максимальный/минимальный элемент в неотсортированной части массива
4. Меняем местами максимальный/минимальный и последний/первый элемент местами из неотсортированной части массива.
5. Так продолжаем, пока массив не будет отсортирован до конца.

Пузырькового включения

Классический алгоритм сортировки, построенный на разделении массива на отсортированную и неотсортированную части. Перемещение первого элемента остатка (неотсортированной части) на принадлежащее ему место в итоге делается путем выбора его места в отсортированной части (включения в последовательность отсортированных

элементов). Причем сначала отсортированной частью считается первый элемент. К нему «включается» второй и т.д. до конца исходного массива.

Метод Шелла

Алгоритм назван так в честь американского ученого Дональда Шелла. Смысл алгоритма заключается в том, чтобы сравнивать не только элементы, стоящие рядом друг с другом, но и на некотором удалении. Сначала выбирается Шаг - некий промежуток, через который будут сравниваться элементы массива на каждой итерации.

Обычно его определяют так:

для первой итерации Шаг равен степени двойки, равной или большей (ближайшей) половине количества элементов массива. На каждой итерации шаг уменьшается вдвое. Когда Шаг будет равен 1 произойдет последнее сравнение и массив будет отсортирован.

Пузырькового всплытия

Самый известный алгоритм, применяемый в учебных целях, но не для практического применения, поскольку является слишком медленным. Относится к обменным сортировкам. Смысл алгоритма в том, что самые "легкие" элементы массива как бы "всплывают", а самые "тяжелые" "тонут". Отсюда и название.

Алгоритм:

1. Каждый элемент массива сравнивается с последующим и если элемент[i] > элемент[i+1] происходит замена. Таким образом, самые "легкие" элементы "всплывают" - перемещаются к началу списка, а самые тяжелые "тонут" - перемещаются к концу.
2. Повторяем Шаг 1 n-1 раз, где n – количество элементов массива.

Для уменьшения количества проверок используется признак PR. Если он процессе проверки не изменится (останется в значении true), то проверку можно заканчивать, так как массив упорядочен.

Метод подсчета

Алгоритм применяется для формирования вектора порядковых номеров элементов множества, расстановка по которым даст сортированное множество.

Алгоритм:

1. Сравниваются все элементы массива друг с другом. При этом порядковый номер большего элемента увеличивается на единицу. Порядковые номера элементов массива формируются в новом массиве. (В примере в листинге далее таким массивом является массив B).
2. Выполняется расстановка элементов исходного массива в новом массиве в соответствии с номерами массива B.

Простого слияния

Алгоритм применяется для сортировки упорядоченных массивов (часто используется как часть алгоритма внешней сортировки). Суть его – два упорядоченных массива соединяются в третий массив.

Алгоритм:

1. Сравнить первые элементы массивов A и B. Меньший элемент (например, из массива A) поместить в новый массив C, сдвинув в массивах A и C указатель записи (счетчик) на единицу.
2. Если в массиве A еще остались элементы, то продолжить выполнение п.1.
3. Если в массиве A элементов не осталось, то переслать оставшиеся в массиве B элементы в массив C. Получили отсортированный массив.

(Замечание. Меньший элемент может оказаться как в массиве A, так и в массиве B, поэтому в п.1-3 алгоритма практически везде можно вместо A написать B).

```
program Sort;
Type TA = array [1..10] of byte;
var A : TA; i: byte;

//Вывод массива
procedure Vivod (A : TA; N : byte);
    var  i: byte;
begin
   for i :=1 to N do write (a[i]:4);
   writeln;
End;

// Сортировка методом простого выбора
procedure Sort1 (var A : TA; N : byte);
    var  i, j, M, AX : byte;
begin
   for i := N downto 2 do begin  // номера упорядочиваемых элементов
     M:=1;
     for j := 2 to i do // поиск максимума среди неупорядоченных
          if A[j] > A[M] then M :=j;
     AX := A[j]; A[j] := A[M]; A[M] := AX //Смена строк местами
                          End;
End;

// Сортировка методом пузырькового включения
procedure Sort2 (var A : TA; N : byte);
    var  i, j, M, B : byte;
begin
     for i := 2 to N do begin  // номера упорядочиваемых элементов
          B:= A[i]; M:=i; // упорядочиваемый элемент и его номер
          for j := i-1 downto 1 do begin
               if A[j] <= B then break; // найдено место для элемента
               A[M] := A[j]; M:= j; end;  // сдвиг больших элементов
          A[M] :=B;       End;
End;

// Сортировка методом пузырькового всплытия
procedure Sort3 (var A : TA; N : byte);
  var  i, j, B : byte; Pr : boolean;
begin
  for j := 1 to N-1 do begin  // номера упорядочиваемых элементов
     Pr :=true;
     for i := N downto j+1 do
          if A[i-1] > A[i]  then begin // проверка сортированности
               Pr := false;
               B:=A[i]; A[i] := A[i-1]; a[i-1]:= B;// смена местами элементов
                                  end;
     if pr then break;   End;
End;

// Сортировка методом Шелла
procedure Sort4 (var A : TA; N : byte);
    var  i,j,k,M,B,D : byte;
    Label Met;
begin
     D:=1; While D < N div 2 do D:=D*2;// шаг и количество групп
 Met: for k := 1 to D do // перебор групп
 // сортировка элементов группы
    i := k+D;
    While i <= N do  begin
          B:= A[i]; M:=i; // упорядочивамый элемент и его номер
          j := i-D;
```

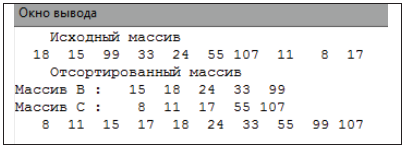
```
          While j >= k do begin
               if A[j] <= B then break;
               A[M] := A[j]; M:= j;
               j:= j-D;     end;
          A[M] :=B; i:=i+D;
                    End;
    if D>1 then begin D:=D div 2; goto Met; end;
End;

// Сортировка методом подсчета
procedure Sort5 (var A : TA; N : byte);
    var  i,j : byte; B,C : TA;
begin
     // определение номеров мест элементов
     for i := 1 to N do B[i]:=1;
     for i := N downto 2 do
          for j := i-1 downto 1 do
               if A[i] <= A[j] then B[j]:=B[j]+1
                               else B[i]:=B[i]+1;
     // Расстановка элементов массива в новый массив
     for i := 1 to N do C[B[i]]:=A[i];
     // Возврат массива в исходный массив
     for i := 1 to N do A[i]:= C[i];
End;

// Сортировка методом простого слияния
procedure Sort6 (var A : TA; N : byte);
Const N2=N div 2;
    var  i,j,k : byte; B, C :TA;
    Label Met;
begin // Подготовка массивов для дальнейшего упорядочивания
  for i :=1 to N2 do B[i]:= A[i];  Sort1(B,5); //Первая половина массива A
        write ('Массив B : '); Vivod (B,5);
  for i :=N2 + 1 to N do C[i-N2]:= A[i]; Sort1(C,5); //Вторая половина массива A
        write ('Массив C : '); Vivod (C,5);
   i:=1; j:=1; k:=1; //начальные значения массивов B, C, A
   Met: if B[i]<C[j] then begin
           A[k] := B[i]; i:=i+1; k:=k+1; // пересылка в массив A из B
           if i <=N2 then goto Met;
           for j:=j to N2 do begin // пересылка остатков из массива C
               A[k] := C[j]; k:=k+1; end;
               exit; end //сортировка закончена
                    else begin
           A[k] := C[j]; j:=j+1; k:=k+1; // пересылка в массив A из C
           if j <=N-N2 then goto Met;
           for i:=i to N-N2 do begin // пересылка остатков из массива B
               A[k] := B[i]; k:=k+1; end;
               exit; end; //сортировка закончена
End;

// Основная программа
begin
   assign (input, 'Sortdan.txt'); Reset (input);
   Writeln ('    Исходный массив');
   for i :=1 to 10 do read (A[i]);
   Vivod (A, 10);
   Writeln ('    Отсортированный массив');
   //Sort1 (A, 10);
   // Sort2 (A, 10);
   // Sort3 (A, 10);
   // Sort4 (A, 10);
   // Sort5 (A, 10);
    Sort6 (A, 10);
   Vivod (A, 10);
   Close (input);
End.
```

Результат выполнения сортировки методом простого слияния:



Результат работы программы при сортировке именно методом простого слияния представлен здесь потому, что отличается от остальных выводом двух исходных упорядоченных массивов B и C.

6.5. Пример выполнения работы

Выполнить сортировку двумерного арифметического массива целых чисел по возрастанию двумя способами:

1. Сортировать весь массив методом пузырькового включения
2. Сортировать строки массива методом пузырькового всплытия

```
program Sort;

type
  TA = array [1..4, 1..5] of byte;
  TB = array [1..20] of byte;
var
    A : TA;
    B : TB;
  i, j: byte;

//Ввод массива
procedure Vvod(var A: TA; M, N: byte);
var
  i, j: byte;
begin
  for i := 1 to M do
    for j := 1 to N do read(A[i, j]);
end;

//Вывод массива
procedure Vivod(A: TA; M, N: byte);
var
  i, j: byte;
begin
  for i := 1 to M do
  begin
    for j := 1 to N do write(a[i, j]:4);
    writeln; end;
end;

// Сортировка методом пузырькового включения
procedure Sort2(var A: TB; N: byte);
 var  i, j, M, C : byte;
begin
     for i := 2 to N do begin  // номера упорядочиваемых элементов
          C:= A[i]; M:=i; // упорядочивамый элемент и его номер
          for j := i-1 downto 1 do begin
               if A[j] <= C then break; // найдено место для элемента
               A[M] := A[j]; M:= j; end;  // сдвиг больших элементов
          A[M] :=C;       End;
end;
```

```
// Сортировка методом пузырькового всплытия
procedure Sort3(var A: TA; K, N: byte);
var  i, j, B : byte; Pr : boolean;
begin
  for j := 1 to N-1 do begin  // номера упорядочиваемых элементов
     Pr :=true;
     for i := N downto j+1 do
           if A[k,i-1] > A[k,i]  then begin // проверка сортированности
               Pr := false;
               B:=A[k,i]; A[k,i] := A[k,i-1]; a[k,i-1]:= B;// смена местами
элементов
                                        end;
     if pr then break;   End;
end;

// Основная программа
begin
  assign(input, 'Sortdan.txt');Reset(input);
  Writeln(' Исходный массив');
  Vvod (A, 4, 5);
  Vivod (A, 4, 5);
  Writeln(' Отсортированный  полный массив');
  // Переводим двумерный массив в одномерный
  for i := 1 to 4 do
       for j := 1 to 5 do B[(i-1)*5 + j] := A[i,j];
   Sort2 (B, 4*5);
   // Переводим отсортированный одномерный массив в двумерный
   for i := 1 to 4 do
       for j := 1 to 5 do A[i,j] := B[(i-1)*5 + j];
  Vivod(A, 4, 5);
  Close(input);Reset(input);
  Writeln(' Снова исходный массив');
  Vvod (A, 4, 5);
  Vivod (A, 4, 5);
  Writeln(' Массив с отсортированными строками');
  for i := 1 to 4 do Sort3 (A, i, 5);
     Vivod(A, 4, 5) ;
  Close(input);
end.
```

Результат работы программы:

```
 Исходный массив
  18  15  99  33  24
  55 107  11   8  17
   9  66  71 117 224
  85   4   7 250  50
 Отсортированный  полный массив
   4   7   8   9  11
  15  17  18  24  33
  50  55  66  71  85
  99 107 117 224 250
 Снова исходный массив
  18  15  99  33  24
  55 107  11   8  17
   9  66  71 117 224
  85   4   7 250  50
 Массив с отсортированными строками
  15  18  24  33  99
   8  11  17  55 107
   9  66  71 117 224
   4   7  50  85 250
```

Обратите внимание, что в данной программе при сортировке всего массива двумерный массив предварительно переведен в одномерный, а после сортировки – обратно. Это действие обязательным не является, приведено здесь для простоты работы и иллюстрации работы процедуры сортировки. Вполне можно было использовать и два индекса. При выполнении работы можно использовать любой способ. Но в качестве полезного упражнения все-таки рекомендую реализовать сортировку именно двумерного массива.

Во втором случае перевод в двумерный массив не требуется, так как выполняется сортировка только одной строки, номер которой передается в процедуру как параметр. Вызов процедуры осуществляется из цикла.

Исходный массив введен два раза из одного и того же файла. Допускается вместо этого использовать пересылку исходного массива в другой массив для временного хранения.

6.6. Варианты на выполнение работы №8

№ вар.	первый метод сортировки	второй метод сортировки	третий метод сортировки
1.	Простого выбора	Метод подсчета	Простого слияния
2.	Пузырькового включения	Пузырькового всплытия	Простого выбора
3.	Метод Шелла	Простого выбора	Метод подсчета
4.	Пузырькового всплытия	Простого слияния	Метод Шелла
5.	Метод подсчета	Простого слияния	Простого выбора
6.	Простого слияния	Пузырькового включения	Метод подсчета
7.	Метод Шелла	Метод подсчета	Простого слияния
8.	Пузырькового включения	Простого слияния	Пузырькового всплытия
9.	Метод подсчета	Пузырькового всплытия	Метод Шелла
10.	Простого слияния	Простого выбора	Пузырькового включения
11.	Простого выбора	Метод подсчета	Простого слияния
12.	Пузырькового включения	Пузырькового всплытия	Метод подсчета
13.	Метод Шелла	Простого выбора	Пузырькового всплытия
14.	Пузырькового всплытия	Простого слияния	Простого выбора
15.	Метод подсчета	Простого слияния	Метод Шелла
16.	Простого слияния	Пузырькового включения	Метод подсчета
17.	Метод Шелла	Метод подсчета	Простого слияния
18.	Пузырькового включения	Простого слияния	Простого выбора
19.	Метод подсчета	Пузырькового всплытия	Пузырькового включения
20.	Простого слияния	Простого выбора	Метод подсчета
21.	Простого выбора	Метод подсчета	Простого слияния
22.	Пузырькового включения	Пузырькового всплытия	Метод Шелла

23.	Метод Шелла	Простого выбора	Простого слияния
24.	Пузырькового всплытия	Простого слияния	Простого выбора

6.7. Контрольные вопросы

1. Что такое сортировка?
2. Что такое внутренние сортировки?
3. Что такое внешние сортировки?
4. Какие существуют критерии для оценки (выбора) методов сортировки?
5. Что может быть результатом сортировки?
6. Поясните суть сортировки методом простого выбора.
7. Поясните суть сортировки методом пузырькового включения.
8. Поясните суть сортировки методом Шелла.
9. Поясните суть сортировки методом пузырькового всплытия.
10. Поясните суть сортировки методом подсчета.
11. Поясните суть сортировки методом простого слияния.

7. Лабораторная работа №9

Разработка программ с использованием модулей

7.1. Цель работы

Целью данной лабораторной работы является освоение:

- правил формирования модулей;
- приемов отладки процедур, содержащихся в модуле;
- формирования и отладки программ, использующих модули.

7.2. Задание на выполнение работы

1. Сформировать модуль, содержащий все процедуры и функции из лабораторных работ 5-8 (при этом ВСЕ процедуры и функции из основных программ должны быть исключены).
2. Выполнить компиляцию модуля.
3. Подготовить 4 программы, использующие модуль.
   a. Программу, выполняющую лабораторную работу №5, а также обе процедуры сортировки арифметических массивов из лабораторной работы №8.
   b. Программу, выполняющую лабораторную работу №5, но с использованием одной или двух функций (также добавленных в модуль) в зависимости от варианта (например, для поиска минимума, суммы, произведения и т.п.), обязательно с учетом смысла задачи и логики ее решения.
   c. Программу, выполняющую лабораторную работу №6 (ввод значения X и шаг его изменения осуществлять в программе, а точность вычислений – в модуле).
   d. Программу, выполняющую задание лабораторной работы №7, а также процедуру сортировки записей из лабораторной работы №8.

7.3. Требования к выполнению

Для удобства выполнения работы допустимо формировать 3 модуля: отдельно по каждой лабораторной работе. В этом случае использование полноценной исполняемой части обязательно. В обязательную часть могут быть включены операторы по определению файлов (хотя бы для вывода результатов), а также константы, являющиеся общими для всех подпрограмм (например, точность).

7.4. Краткие теоретические сведения

Модуль Паскаля – это автономно компилируемая программная единица, включающая в себя различные компоненты раздела описаний (типы, константы, переменные, процедуры и функции) и, возможно, некоторые исполняемые операторы инициирующей части. Модульное программирование – это организация программы как совокупности небольших независимых блоков, называемых модулями, структура и поведение которых подчиняются определенным правилам.

Модуль Паскаля имеет следующую структуру:

Unit <имя_модуля>;
interface <интерфейсная часть>;
implementation < исполняемая часть >;
begin
<инициирующая часть>;
End.

В интерфейсной части, которая открывается словом ***interface*****,** содержатся объявления всех глобальных объектов модуля (типов, констант, переменных и подпрограмм), которые должны быть доступны основной программе и (или) другим модулям Паскаля. При объявлении глобальных подпрограмм в интерфейсной части указывается только их заголовок. В части ***implementation*** содержатся подпрограммы (процедуры и функции). В инициирующей части будут операторы, которые выполняются при подключении модуля. Инициирующая часть может быть вырожденной, когда используется только ***End.***

Модуль к программе подключается с использованием ключевого слова ***Uses,*** которое должно быть после слова ***Program.***

Модуль может использовать подпрограммы других модулей. При этом к модулю модуль подключается после слова ***interface.***

Необходимо помнить, что при подключении модуля к программе объекты модуля (типы, константы, переменные, подпрограммы) – внешние переменные - фактически становятся для программы глобальными переменными, а переменные, объявленные в основной программе – локальными. Это приводит к тому, что переобъявленные объекты в основной программе не будут видны внутри модуля (в том числе в заголовках подпрограмм), а значит, появится ошибка несогласованности типов. Например, объявленный дважды тип

```
TYPE    TMAS = ARRAY [1..M, 1..N] OF REAL;
```

будет считаться переобъявленным. Поэтому при выполнении работы нужно обратить особое внимание на области видимости программ и локализацию мест объявления (определения) переменных.

Для того, чтобы использовать сформированные модули, необходимо выполнить их компиляцию, т.е. сформировать из Pas-файлов объектные модули, сохранив их на диск. В PascalABC.Net сохранение на диск выполняется автоматически, некоторые системы требуют дополнительной настройки. Важно, что Имя_модуля (которое записано после слова **Unit)** должно совпадать с именем Pas-файла, в котором модуль находится!

В результате компиляции модуля в текущем каталоге появляются файлы с именем таким же как и Pas-файл, но с расширением pcu (в некоторых версиях Паскаля tpu). Можно разместить модули и в другом каталоге, но в этом случае требуется дополнительная настройка среды.

7.5. Пример выполнения работы

Выполнение программы рассмотрим только на примере одной лабораторной работы - №5. При этом добавим две функции (вместо процедуры) для вычисления суммы и количества положительных элементов заданного столбца (как того требует задание №3b).

Программный код модуля имеет вид:

```
UNIT MODUL_LR5;
Interface
 CONST  M = 5; N = 6;
 TYPE    TMAS = ARRAY [1..M, 1..N] OF REAL;
          TR = ARRAY [1..M, 1..2] OF BYTE;
 VAR        X : TMAS;
            R : TR;
     K1,K2,I,J,Kol : BYTE;
          Sum : REAL;
           FI : TEXT;

 PROCEDURE PRINT (B : TMAS; M, N : INTEGER);
 PROCEDURE PRINT2 (B : TR; M : INTEGER);
 PROCEDURE OBRABOT (VAR B: TMAS; VAR R : TR; M,N: INTEGER);
 PROCEDURE CHANGE (VAR B : TMAS; M, K1, K2 : INTEGER);
 PROCEDURE SUMKOL (VAR B : TMAS;  M, K1  : INTEGER; VAR S: REAL; VAR K: BYTE);
 FUNCTION SUMMA ( VAR B : TMAS;  M, K1  : INTEGER) :REAL;
 FUNCTION KOLVO ( VAR B : TMAS;  M, K1  : INTEGER) :BYTE;

Implementation
{ --------Процедура вывода--------------------}
     PROCEDURE PRINT (B : TMAS; M, N : INTEGER);
     VAR I,J : BYTE;
     BEGIN
            FOR I := 1 TO M DO begin
                    FOR J := 1 TO N DO WRITE ( B[I,J] : 8 :1);
                writeln;      end;
     END;

{ --------Процедура вывода нового массива-------------------}
     PROCEDURE PRINT2 (B : TR; M : INTEGER);
     VAR I : BYTE;
     BEGIN
            FOR I := 1 TO M DO  WRITELN ( B[I,1] : 3,  B[I,2] : 3);
                writeln;
     END;
//--------Процедура формирования массива-------------------
PROCEDURE OBRABOT (VAR B: TMAS; VAR R : TR; M,N: INTEGER);
   VAR  I, J, JM : BYTE;
BEGIN
     FOR I := 1 TO M DO BEGIN
         JM :=1;
         FOR J := 1 TO N DO
             IF B[I,J] > B[I,JM] THEN JM := J;
         R[I,1] := I;  R[I,2] := JM; END;
 END;

 { --------Процедура смены строк местами-------------------}
     PROCEDURE CHANGE (VAR B : TMAS; M, K1, K2 : INTEGER);
```

```
     VAR I : BYTE; TEMP : REAL;
     BEGIN
            FOR I := 1 TO M DO BEGIN
                  TEMP := B[I,K1];
                  B[I,K1]:= B[I,K2];
                  B[I,K2]:= TEMP;
                     END;
     END;

{ --------Процедура обработки столбца--------------------}
     PROCEDURE SUMKOL (VAR B : TMAS;  M, K1  : INTEGER; VAR S: REAL; VAR K:
BYTE);
     VAR I : BYTE;
     BEGIN
            FOR I := 1 TO M DO
               IF B[I,K1] > 0 THEN BEGIN
                  S := S + B[I,K1];
                  K := K + 1;
                                 END;
     END;

     FUNCTION SUMMA ( VAR B : TMAS;  M, K1  : INTEGER) :REAL;
      VAR I : BYTE; S : REAL;
     BEGIN
            FOR I := 1 TO M DO
               IF B[I,K1] > 0 THEN S := S + B[I,K1];
            SUMMA := S;
      End;

     FUNCTION KOLVO ( VAR B : TMAS;  M, K1  : INTEGER) :BYTE;
         VAR I : BYTE; S : BYTE;
       BEGIN
            FOR I := 1 TO M DO
               IF B[I,K1] > 0 THEN S := S + 1;
            KOLVO := S;
       End;

 {-------------- Инициирующая часть модуля--------------}
BEGIN   ASSIGN (FI, 'ffff.dat');  RESET(FI);

END.
```

Здесь в инициирующей части модуля оставлены операции по работе с файлом исходных данных. В общем случае это не очень хорошо, так как снижает гибкость программы. Допустимо использовать инициализацию файла результатов. Но в данном случае оставлено просто для демонстрации возможностей. В инициирующей части может быть вообще оставлено только слово End с точкой. Кроме того, в этом модуле размещена процедура для вычисления количества и суммы положительных элементов заданного столбца двумерного массива, и функции. Для решения конкретной задачи в этом может не быть необходимости, но учитывая, что модуль – это библиотека, имеет смысл оставить оба варианта, которые можно использовать при необходимости.

Программа с использованием функций примет вид:

```
PROGRAM LR5;
 Uses MODUL_LR5;
 Var i,j : byte;
 {-------------- Основная программа--------------}
BEGIN   ASSIGN (FI, 'ffff.dat');  RESET(FI);
 // --------ввод исходного массива--------------------
  WRITELN (' Исходный массив ' );
     FOR I := 1 TO M DO
```

```
          FOR J := 1 TO N DO READ ( FI, X[I,J] );
      PRINT ( X, M, N );
      OBRABOT ( X, R,  M, N );WRITELN;
  WRITELN (' Результат работы ' );
  WRITELN (' Сформированный массив: ' );
  PRINT2 ( R, M );
  WRITE (' Введите номера строк, которые нужно поменять местами ' );
  READLN (K1, K2); CHANGE (X, M, K1, K2 );
  WRITELN (' Измененный исходный массив ' );
  PRINT ( X, M, N );    WRITELN;
  WRITE (' Введите номера столбца, который нужно обработать ' );
  READLN (K1);
 // SUMKOL (X, M, K1, Sum, Kol);
  WRITELN (' сумма и произведение ', K1, '-го столбца :' );
  WRITELN ( ' Sum = ', SUMMA (X, M, K1) : 5:1 );
  WRITELN ( ' Kol = ', KOLVO (X, M, K1) : 3 );
END.
```

Результат не отличается от результата, полученного в лабораторной работе №5. Приведем его здесь еще раз с другими заданными строками:

Окно вывода

```
Исходный массив
     6.0      4.0      5.0      3.0      7.0      9.0
    16.0    -11.0     12.0     13.0    -14.0     15.0
    21.0     26.0     23.0     24.0     25.0    -22.6
    31.0    -32.0     33.0    -34.2     35.0     30.0
    41.0     42.0     43.0     48.0     45.0     46.0

Результат работы
Сформированный массив:
 1  6
 2  1
 3  2
 4  5
 5  4

Введите номера строк, которые нужно поменять местами 2 3
Измененный исходный массив
     6.0      5.0      4.0      3.0      7.0      9.0
    16.0     12.0    -11.0     13.0    -14.0     15.0
    21.0     23.0     26.0     24.0     25.0    -22.6
    31.0     33.0    -32.0    -34.2     35.0     30.0
    41.0     43.0     42.0     48.0     45.0     46.0

Введите номера столбца, который нужно обработать 6
сумма и произведение 6-го столбца :
Sum = 100.0
Kol =   4
```

7.6. Контрольные вопросы

1. Что такое модуль?
2. Поясните правила формирования модулей.
3. Поясните использование ключевых слов Interface и Implementation.
4. Что такое внешние переменные? Поясните правила их использования.

5. Области видимости объектов модуля.
6. Как подключить модуль к программе?
7. Что такое стандартные модули?
8. Поясните структуру модуля.
9. Какие процедуры/функции следует помещать в модули?
10. Для чего нужна исполняемая часть модуля?
11. В какой момент выполняются операторы исполняемой части модуля?

8. Применение модульного программирования

Преимущества модульного программирования

1. Выполнять детализацию задачи для удобства отладки и работы с кодом.
2. Повторно использовать подпрограммы не только в рамках одной программы, но и в разных.
3. Возможность писать модули на разных языках программирования.
4. Повторно использовать имена объектов программы (модуль является естественной единицей локализации имен).
5. Локализовывать места ошибки: обычно исправление ошибки внутри одного модуля не влечет за собой исправление других модулей.

Недостатки модульного программирования

1. Модульность требует большей дополнительной работы. При написании модульных программ программист больше времени тратит на этапе проектирования программной разработки и должен быть значительно более аккуратным в процессе программирования.
2. Модульный подход обычно требует большего времени центрального процессора.
3. В модульном подходе скорее всего потребуется несколько больший объем памяти.

Рекомендации по применению модулей

Основной рекомендацией по применению модулей является разумный подход к их использованию и организации. К этому можно отнести следующее:

1. Объем модулей должен быть не слишком большой, но и не слишком маленький, с завершенной функциональностью.
2. Обязательное использовать модульную структуру при повторяющейся однотипной обработке данных.
3. Использование параметров и минимизация использования глобальных переменных.
4. Использование «говорящих» имен и по возможности «сквозных» во всех модулях.

9. Список рекомендуемой литературы

1. Егорова А.А.	Пособие по проведению практических занятий по дисциплине «Алгоритмические языки и программирование» - М.: МГТУ ГА, 2013 – 36 стр.
1. Егорова А.А.	Алгоритмические языки и программирование. Учебно-методическое пособие по выполнению лабораторных работ. Часть 1. Основы программирования. - М.: МГТУ ГА, 2020 – 52 стр.
3. Климова Л.М.	Pascal 7.0. Практическое программирование. Решение типовых задач. КУДИЦ-ОБРАЗ., 4-е издание

4. Кремень Ю., Расолько Г.	Теория и практика программирования на языке Pascal / Москва / Литрес/ 2015 г., 449 с.
5. Осипов А.В.	PascalABC.NET: Введение в современное программирование.– Ростов-на-Дону , 2019 – 572 с. :
6.	https://pascal-abc.ru.net/uchebnik/
7.	http://pascalabc.net/downloads/OsipovBook/%D0%9A%D0%BD%D0%B8%D0%B3%D0%B0%D0%94%D0%BB%D1%8F%D0%A1%D0%B0%D0%B9%D1%82%D0%B0.pdf
8.	http://pascalabc.net/

10. Заключение

Поскольку лабораторные работы, а соответственно и данное учебное пособие имеют практическую направленность и предназначены в первую очередь для отработки навыков, то и защита каждой лабораторной работы проводится с демонстрацией полученных навыков. А значит, их необходимо довести если не до автоматизма, то хотя бы до такого уровня, при котором можно выполнить задание преподавателя за ограниченное время, не гладя в пособие. Это требует определенной практики. Поэтому выполнение каждой работы – это не просто четкое выполнение задания, но и практическое изучение материала, связанного с заданием. Вариативность выполнения каждого задания (несколькими способами) позволит лучше понять механизм функционирования систем и закрепить навык.

В рамках лабораторной работы приветствуется расширение работы (использование не указанных в пособии функций, операций, выполнение дополнительных заданий и т.п.).