МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**»

УТВЕРЖДАЮ
Директор ИнЭО
_____ С.И. Качин

«____»_____ 2015 г.

ИНФОРМАТИКА 1.2

Методические указания к выполнению лабораторных работ
для студентов ИнЭО, обучающихся по направлению
09.03.01 «Информатика и вычислительная техника»

Составители
О.С. Токарева, А.В. Лепустин, Д.В.Погребной

Издательство
Томского политехнического университета
2015

УДК 681.32

Информатика 1.2: метод. указ. к выполнению лабораторных работ для студентов ИнЭО, обучающихся по напр. 09.03.01 «Информатика и вычислительная техника» / сост. О.С. Токарева, А.В. Лепустин; Томский политехнический университет. – Томск: Изд-во Томского политехнического университета, 2015. – 20 с.

Методические указания к выполнению лабораторных работ рассмотрены и рекомендованы к изданию методическим семинаром кафедры вычислительной техники
«____»_____2015 г., протокол №____.

Зав. кафедрой ВТ
профессор, доктор техн. наук _____ Н.Г. Марков

Аннотация

Методические указания к выполнению лабораторных работ по дисциплине «Информатика 1.2» предназначены для студентов ИнЭО, обучающихся по направлению 09.03.01 «Информатика и вычислительная техника». Данные лабораторные работы выполняются в первом семестре.

ОГЛАВЛЕНИЕ

ОБЩИЕ МЕТОДИЧЕСКИЕ УКАЗАНИЯ

В соответствии с учебным графиком для студентов, обучающихся по направлению 09.03.01 «Информатика и вычислительная техника» предусмотрено выполнение двух лабораторных работ.

Отчет о выполнении лабораторной работы оформляется в печатном виде и должен включать титульный лист и состоять из разделов:

- цель работы;
- постановка задачи;
- схема алгоритма;
- листинг программы с комментариями основных действий;
- результаты работы программы и ручного тестирования;
- выводы по проделанной работе.

Схемы алгоритмов должны быть составлены в соответствии с ГОСТ 19.701–90.

Текстовая часть отчета оформляется в свободной форме. В отчете необходимо привести листинг программы. Оформление титульного листа приведено в прил. 1.

Студент выбирает тот вариант задания, который соответствует двум последним цифрам его зачетной книжки (НЗК). Если НЗК превышает 20, то необходимо из НЗК отнимать 20 до тех пор, пока полученное число не попадет в диапазон от 1 до 20.

ЛАБОРАТОРНАЯ РАБОТА № 1
«АЛГОРИТМЫ СОРТИРОВКИ НЕУПОРЯДОЧЕННЫХ ДАННЫХ»

Цель работы. Научиться составлять алгоритмы и программы сортировки неупорядоченных данных. Освоить алгоритм поиска элемента методом деления пополам.

Задания

1. Задан массив $A[12]$ из вещественных чисел. Выполнить сортировку элементов массива, т.е. расположить в порядке убывания (или возрастания) методом «пузырька» (сортировка обменом).

2. Задан массив $A[12]$ из вещественных чисел. Выполнить сортировку элементов массива, т.е. расположить элементы в порядке убывания (или возрастания), используя метод сортировки выбором.

3. Задан массив $A[15]$ из вещественных чисел, первые элементы которого упорядочены, последние – не упорядочены. Упорядочить весь массив методом вставок.

4. Даны два массива вещественных чисел. Один массив упорядочен, другой – не упорядочен. Составить из двух массивов один упорядоченный массив методом вставок.

5. Из двух отсортированных массивов $A[10]$ и $B[10]$ вещественных чисел методом слияния создать один отсортированный массив в порядке убывания или возрастания элементов.

6. Даны два массива вещественных чисел $A[12]$, $B[12]$. Оставить в первом массиве только те элементы, которых нет во втором массиве.

7. Из массива $Z[15]$ вещественных чисел удалить все повторяющиеся элементы.

8. Дана матрица из вещественных элементов $A[6,6]$. Переставить строки и столбцы матрицы так, чтобы максимальный элемент оказался первым элементом матрицы.

9. Даны координаты точек на плоскости. Найти наибольшее расстояние между парами этих точек.

10. Дан массив вещественных чисел $A[20]$. Определить количество различных элементов в массиве, повторяющиеся элементы считать один раз.

11. Дан массив $A[20]$ из вещественных чисел. Преобразовать массив так, чтобы в начале массива стояли положительные числа и нули, в конце массива – отрицательные элементы, порядок следования элементов в группах сохранить.

12. Дан массив $A[20]$ из вещественных чисел. Преобразовать массив так, чтобы из повторяющихся подряд чисел осталось только одно.

13. Компоненты вектора $a=\{a_1,..., a_{10}\}$ расположены в порядке возрастания по абсолютной величине. Построить (наиболее оптимальным способом) и вывести на печать вектор $b=\{b_1, ..., b_{10}\}$ исключая из вектора k-ю компоненту и вставляя на нужное место вещественное число r так, чтобы компоненты вектора B оказались также расположенными в порядке возрастания по абсолютной величине.

14. Упорядочить матрицу $A[6,6]$ по элементам первого столбца. При выполнении указанной сортировки необходимо выполнить перестановку строк матрицы таким образом, чтобы элементы первого столбца были упорядочены методом выбора минимума.

15. Упорядочить матрицу $A[6,6]$ по элементам первой строки. При выполнении сортировки необходимо выполнить перестановку столбцов таким образом, чтобы элементы первой строки были упорядочены методом простых вставок.

16. Упорядочить матрицу $A[6,6]$ по элементам третьей строки. При выполнении сортировки необходимо выполнить перестановку столбцов таким образом, чтобы элементы третьей строки были упорядочены методом выбора максимума.

17. Даны действительные числа $a_1,..., a_n, b_1,..., b_m$ ($a_1 \le a_2 \le ... \le a_n$). Получить натуральные числа $k_1,..., k_m$ такие, что k_i – это решение задачи поиска места b_i среди $a_1,..., a_n$ ($i=1,..., m$). Применить алгоритм деления пополам.

18. Даны действительные числа $a_1,..., a_n, p$, натуральное число k ($a_1 \le a_2 \le ... \le a_n$, $k \le n$). Удалить из $a_1,..., a_n$ элемент с номером k (т.е. a_k) и вставить элемент, равный p, так, чтобы не нарушилась упорядоченность. Действия выполнять оптимальным способом.

19. Пусть дан массив $a_1,..., a_n$. Требуется переставить $a_1,..., a_n$ так, чтобы вначале массива шла группа элементов, больших того элемента, который в исходном массиве располагался на первом месте, затем – сам этот элемент, потом – группа элементов, меньших или равных ему. Число сравнений и перемещений, каждое в отдельности, не должно превышать n-1.

20. Последовательным просмотром чисел $a_1,..., a_n$ найти наименьшее i такое, что $a_i > a_{i+1}$. Поменять a_i и a_{i+1} местами и возобновить просмотр с начала массива. Когда не удастся найти такое i, массив будет упорядочен нужным образом. Написать программу, реализующую этот алгоритм.

Методические указания

Сортировка представляет собой процесс упорядочивания элементов в массиве по возрастанию или убыванию. Различают внутренние и внешние сортировки

Внутренняя сортировка – сортировка данных, полностью помещающихся в оперативной памяти.

Внешняя сортировка – сортировка данных с ленты или диска, т.е. из внешней памяти.

Обращение к оперативной памяти требует меньше времени, поэтому внутренние сортировки эффективнее.

Сортировка называется устойчивой, если при ее применении положение записей с равными ключами (или элементов массива с одинаковыми значениями) не изменяется.

Существует большое количество алгоритмов сортировки, базирующихся на трех основных методах сортировки:

- обменом;
- выбором;
- вставкой.

Рассмотрим подробнее перечисленные алгоритмы сортировки массива. Пусть на вход алгоритма поступают:

1. Количество элементов массива – переменная *n*.
2. Целочисленный массив – переменная *A*. Номера элементов в массиве – от 1 до *n*.

Сортировка методом простого обмена (пузырьковая сортировка) является наиболее известной, является устойчивой.

В упорядоченном по возрастанию массиве для любой пары рядом стоящих элементов должно выполняться условие:

$A[i] <= A[i+1]$.

Пусть алгоритм последовательно выполняет сравнения:

$A[i] > A[i+1]$.

При каждом сравнении возможны два исхода:

1) $A[i] > A[i+1]$, результат сравнения истинен, в этом случае элементы массива $A[i]$ и $A[i+1]$ должны обменяться своими значениями;

2) $A[i] <= A[i+1]$, результат сравнения ложен, обмен делать не нужно.

Сравнения нужно выполнить последовательно от начала массива для всех пар соседних элементов. После однократного прохода по массиву самый большой элемент попадет на последнее место (пузырек всплывет). Он больше не будет участвовать в сравнениях, это следует учесть при записи цикла. Для того чтобы каждый элемент попал на свое место, нужно выполнить (n – 1) проходов по массиву.

Алгоритм пузырьковой сортировки приведен на рис. 1*a*.

Сортировка выбором. Алгоритм сортировки выбором относится к неустойчивым алгоритмам. Будем сортировать массив в порядке возрастания:

1. Найдем в текущем массиве элемент с минимальным значением.
2. Произведем обмен этого значения со значением на первой позиции в текущем массиве.
3. Сортируем хвост массива, исключив из процедуры сортировки уже отсортированный первый элемент.

На каждом проходе граница массива сдвигается вправо на 1 элемент, таким образом, на втором проходе не рассматривается первый элемент, на третьем – первый и второй и т.д.

Алгоритм сортировки выбором приведен на рис. 1*b*.

Существует двунаправленный алгоритм сортировки выбором, когда на каждом шаге ищется максимальный и минимальный элемент и устанавливается минимальный в начало, максимальный – в конец массива (при сортировке по возрастанию). Граница рассматриваемого массива на каждом проходе сдвигается вправо и влево на один элемент.

Сортировка вставкой. Будем сортировать массив в порядке возрастания. Сортировка вставкой заключается в том, что сначала сортируются два первых элемента массива в порядке возрастания. Затем делается вставка третьего элемента в соответствующее положение по отношению к первым двум, так, чтобы три элемента были отсортированы по возрастанию. Процесс повторяется, пока все элементы массива не будут отсортированы.

Метод сортировки вставкой прост в реализации, эффективен при небольших наборах и для частично отсортированных данных, относится к устойчивым сортировкам. Используя метод сортировки вставкой можно сортировать список по мере его поступления. Алгоритм сортировки вставкой приведен на рис. 1*c*.

Поиск элемента с заданным значением методом деления пополам. Дан упорядоченный целочисленный массив A, содержащий n элементов, и некоторое числовое значение p. Требуется найти такой номер i элемента массива, для которого $A[i] = p$, или определить, что такого номера нет.

Эта задача может быть решена просмотром элементов массива в цикле до первого совпадения значения элемента $A[i]$ с p. Поскольку после нахождения искомого значения просматривать массив дальше нецелесообразно, в алгоритме следует использовать цикл while. Этот цикл завершится, когда будет достигнут конец массива или найдено искомое значение. Если при этом окажется, что параметр цикла стал больше, чем длина массива, то искомого значения в массиве нет. В противном случае элемент со значением p находится на месте i.

Поиск можно значительно ускорить, если использовать свойство упорядоченности массива.

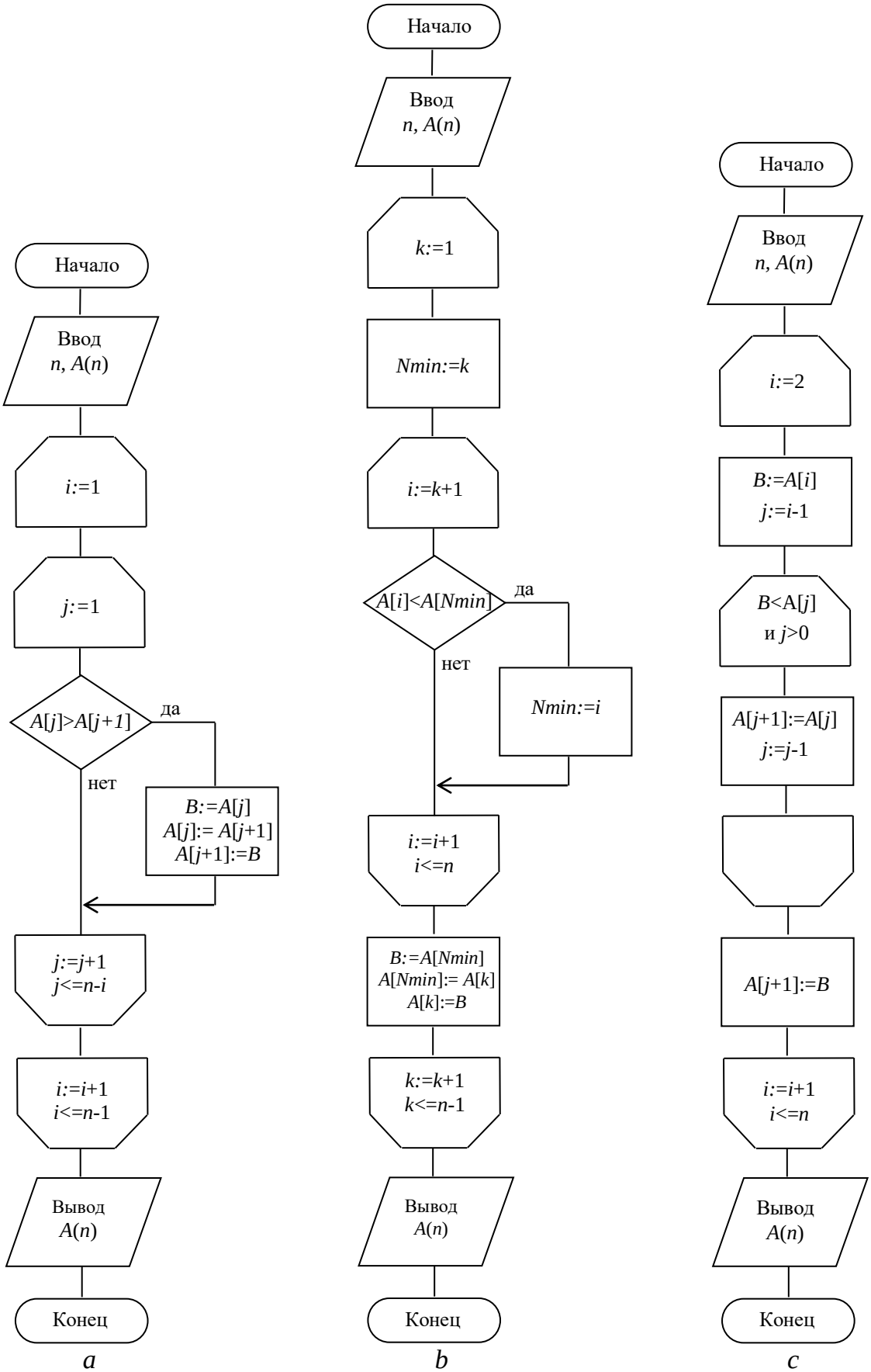

Рис. 1. Схемы сортировок: *a* – пузырьковая, *b* – выбором, *c* – вставками

Идея быстрого алгоритма поиска состоит в том, что в массиве выделяется область «подозрительных» элементов, т.е. таких, что если искомый элемент в массиве существует, он обязательно будет внутри этой области. Каждое сравнение выполняется так, чтобы область сокращалась вдвое, поэтому алгоритм получил название ***дихотомического (деление пополам)***. Для этого на каждом шаге цикла производится сравнение p с элементом $A[c]$, расположенным в середине "подозрительной" области. При этом возможны два случая:

1) $A[c] < p$, тогда из области «подозрительных» элементов отбрасываются все, начиная с начала области до номера с включительно;

2) $A[c] >= p$, тогда из области «подозрительных» элементов отбрасываются все, начиная с номера $c + 1$ до конца области.

ЛАБОРАТОРНАЯ РАБОТА № 2
«ПРОЦЕДУРЫ И ФУНКЦИИ»

Цель работы. Научиться составлять программы с использованием процедур и функций.

Задания

Заданы две матрицы $A(6,6)$ и $B(6,6)$. Для матрицы, которая отвечает заданному условию (1) выполните преобразования (2) в соответствии с вариантом задания, представленным в табл. 1.

Условие (1) проверить в подпрограмме Function, преобразование выполнить в подпрограмме Procedure.

Таблица 1

№	Задание
1	1 – количество отрицательных элементов больше 2 – просуммировать положительные элементы в каждой строке
2	1 – количество нулевых элементов больше 2 – просуммировать отрицательные элементы в каждой строке
3	1 – количество ненулевых элементов больше 2 – найти сумму всех элементов главной диагонали
4	1 – больше строк начинается с нуля 2 – вывести все положительные элементы в одномерный массив
5	1 – больше строк начинается с нуля 2 – поменять между собой соответствующие элементы в строках, если они не равны друг другу
6	1 – больше столбцов начинается с положительного элемента 2 – поменять между собой соответствующие элементы в строках, если они разных знаков
7	1 – больше строк начинается с положительного элемента 2 – из каждой пары соседних элементов в каждой строке больший вывести в одномерный массив
8	1 – больше столбцов начинается с нуля 2 – найти сумму и количество всех отрицательных элементов после главной диагонали
9	1 – больше сумма положительных элементов 2 – найти сумму элементов после главной диагонали в каждой строке и заменить этим значением элемент на главной диагонали
10	1 – больше сумма отрицательных элементов 2 – в каждой строке, если она начинается с нуля, перемножить все ненулевые элементы и заменить этим значением первый элемент строки

11	1 – больше сумма элементов на главной диагонали 2 – в каждом столбце найти максимальный элемент и номер столбца, в котором он расположен
12	1 – больше количество нулевых элементов на главной диагонали 2 – в каждой строке найти максимальный элемент и место, где он расположен
13	1 – больше количество положительных элементов на главной диагонали 2 – в каждом столбце найти максимальный элемент и место, где он расположен
14	1 – больше сумма отрицательных элементов на главной диагонали 2 – в каждой строке найти минимальный элемент и место, где он расположен
15	1 – больше сумма положительных элементов в четных строках 2 – найти максимальный элемент и место, где он расположен
16	1 – больше сумма положительных элементов в четных строках 2 – найти первый нулевой элемент и место где он расположен
17	1 – больше количество нулевых элементов в нечетных столбцах 2– найти первый положительный элемент и место где он расположен
18	1 – больше количество нулевых элементов в четных столбцах 2 – найти в каждой строке первый положительный элемент
19	1 –больше количество нулевых элементов в четных столбцах и нечетных строках 2 – найти в каждой строке первый отрицательный элемент
20	1 – больше количество нулевых элементов в четных столбцах четных строках 2 – транспонировать матрицу

Методические указания

В языке Паскаль имеется два вида подпрограмм – ***процедуры и функции.***

Процедуры и функции помещаются в раздел описаний программы. Для обмена информацией между процедурами и функциями и другими блоками программы существует механизм входных и выходных параметров. Входными параметрами называют величины, передающиеся из вызывающего блока в подпрограмму (исходные данные для подпрограммы), а выходными – передающиеся из подпрограммы в вызывающий блок (результаты работы подпрограммы).

Одна и та же подпрограмма может вызываться неоднократно, выполняя одни и те же действия с разными наборами входных данных. Параметры, использующиеся при записи текста подпрограммы в разде-

ле описаний, называют ***формальными***, а те, что используются при ее вызове – ***фактическими***.

Структура описания процедур и функций похожа на структуру программы: у них также имеются заголовок, раздел описаний и исполняемая часть. Раздел описаний содержит те же подразделы, что и раздел описаний программы: описания констант, типов, меток, процедур, функций, переменных. Исполняемая часть содержит собственно операторы процедур.

Формат описания процедуры имеет вид:

```
procedure имя процедуры (формальные параметры);
    {раздел описаний процедуры}
begin
    {исполняемая часть процедуры}
end;
```

Формат описания функции:

```
function имя функции (формальные параметры): тип результата;
    {раздел описаний функции}
begin
    {исполняемая часть функции}
end;
```

Формальные параметры в заголовке процедур и функций записываются в виде:

```
var имя параметра: имя типа
```

и отделяются друг от друга точкой с запятой. Ключевое слово `var` может отсутствовать. Если параметры однотипны, то их имена можно перечислять через запятую, указывая общее для них имя типа. При описании параметров можно использовать только стандартные имена типов, либо имена типов, определенные с помощью команды `type`. Список формальных параметров может отсутствовать.

Вызов процедуры производится оператором, имеющим следующий формат:

```
имя процедуры(список фактических параметров);
```

Список фактических параметров – это их перечисление через запятую. При вызове фактические параметры как бы подставляются вместо формальных, стоящих на тех же местах в заголовке. Таким образом происходит передача входных параметров, затем выполняются операторы исполняемой части процедуры, после чего происходит возврат в вызывающий блок. Передача выходных параметров происходит непосредственно во время работы исполняемой части.

Вызов функции в Турбо Паскаль может производиться аналогичным способом, кроме того имеется возможность осуществить вызов внутри какого-либо выражения. В частности имя функции может стоять в правой части оператора присваивания, в разделе условий оператора `if` и т.д.

Для передачи в вызывающий блок выходного значения функции в исполняемой части функции перед возвратом в вызывающий блок необходимо поместить следующую команду:

```
имя функции := результат;
```

При вызове процедур и функций необходимо соблюдать следующие правила:

- количество фактических параметров должно совпадать с количеством формальных;
- соответствующие фактические и формальные параметры должны совпадать по порядку следования и по типу.

Заметим, что имена формальных и фактических параметров могут совпадать. Это не приводит к проблемам, так как соответствующие им переменные все равно будут различны из-за того, что хранятся в разных областях памяти. Кроме того, все формальные параметры являются временными переменными – они создаются в момент вызова подпрограммы и уничтожаются в момент выхода из нее.

Рассмотрим использование процедуры на примере программы поиска максимума из двух целых чисел.

```
var x,y,m,n: integer;
procedure MaxNumber(a,b: integer; var max: integer);
begin
    if a>b then max:=a else max:=b;
end;
begin
    write('Введите x,y ');
    readln(x,y);
    MaxNumber(x,y,m);
    MaxNumber(2,x+y,n);
    writeln('m=',m,'n=',n);
end.
```

Аналогичную задачу, но уже с использованием функций, можно решить так:

```
var x,y,m,n: integer;

function MaxNumber(a,b: integer): integer;
```

```
    var max: integer;
begin
    if a>b then max:=a else max:=b;
    MaxNumber := max;
end;

begin
    write('Введите x,y ');
    readln(x,y);
    m := MaxNumber(x,y);
    n := MaxNumber(2,x+y);
    writeln('m=',m,'n=',n);
end.
```

Передача параметров. В стандарте языка Паскаль передача параметров может производиться двумя способами – по значению и по ссылке. Параметры, передаваемые по значению, называют параметрами-значениями, передаваемые по ссылке – параметрами-переменными. Последние отличаются тем, что в заголовке процедуры (функции) перед ними ставится служебное слово `var`.

При первом способе (передача по значению) значения фактических параметров копируются в соответствующие формальные параметры. При изменении этих значений в ходе выполнения процедуры (функции) исходные данные (фактические параметры) измениться не могут. Поэтому таким способом передают данные только из вызывающего блока в подпрограмму (т.е. входные параметры). При этом в качестве фактических параметров можно использовать и константы, и переменные, и выражения.

При втором способе (передача по ссылке) все изменения, происходящие в теле процедуры (функции) с формальными параметрами, приводят к немедленным аналогичным изменениям соответствующих им фактических параметров. Изменения происходят с переменными вызывающего блока, поэтому по ссылке передаются выходные параметры. При вызове соответствующие им фактические параметры могут быть только переменными.

Выбор способа передачи параметров при создании процедуры (функции) происходит в соответствии со сказанным выше: входные параметры нужно передавать по значению, а выходные – по ссылке. Практически это сводится к расстановке в заголовке процедуры (функции) описателя `var` при всех параметрах, которые обозначают результат работы подпрограммы. Однако, в связи с тем, что функция возвращает

только один результат, в ее заголовке использовать параметры-переменные не рекомендуется.

Локальные и глобальные идентификаторы. Использование процедур и функций в Паскале тесно связано с некоторыми особенностями работы с идентификаторами (именами) в программе. В частности, не все имена всегда доступны для использования. Доступ к идентификатору в конкретный момент времени определяется тем, в каком блоке он описан.

Имена, описанные в заголовке или разделе описаний процедуры или функции называют локальными для этого блока. Имена, описанные в блоке, соответствующем всей программе, называют глобальными. Следует помнить, что формальные параметры процедур и функций всегда являются локальными переменными для соответствующих блоков.

Основные правила работы с глобальными и локальными именами можно сформулировать так:

1. Локальные имена доступны (считаются известными, «видимыми») только внутри того блока, где они описаны. Сам этот блок, и все другие, вложенные в него, называют областью видимости для этих локальных имен.

2. Имена, описанные в одном блоке, могут совпадать с именами из других, как содержащих данный блок, так и вложенных в него. Это объясняется тем, что переменные, описанные в разных блоках (даже если они имеют одинаковые имена), хранятся в разных областях оперативной памяти.

3. Глобальные имена хранятся в области памяти, называемой сегментом данных (статическим сегментом) программы. Они создаются на этапе компиляции и действительны на все время работы программы.

4. В отличие от них, локальные переменные хранятся в специальной области памяти, которая называется стек. Они являются временными, так как создаются в момент входа в подпрограмму и уничтожаются при выходе из нее.

5. Имя, описанное в блоке, «закрывает» совпадающие с ним имена из блоков, содержащие данный. Это означает, что если в двух блоках, один из которых содержится внутри другого, есть переменные с одинаковыми именами, то после входа во вложенный блок работа будет идти с локальной для данного блока переменной. Переменная с тем же именем, описанная в объемлющем блоке, становится временно недоступной и это продолжается до момента выхода из вложенного блока.

Рекомендуется все имена, которые имеют в подпрограммах внутреннее, вспомогательное назначение, делать локальными. Это предохраняет от изменений глобальные объекты с такими же именами.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Истомин Е.П. Информатика и программирование: учебник/ Е.П. Истомин, С.Ю. Неклюдов, В.И. Романченко. – СПб: ООО «Андреевский издательский дом», 2008. – 248 с.

2. Костюк Ю.Л. Основы алгоритмизации: учеб. пособие. – Томск: Изд-во ТГУ, 1996. – 124 с.

3. Токарева О.С. Информатика: учеб. пособие/ О.С. Токарева, А.В. Лепустин. – Томск: Изд-во Томского политехнического университета, 2011.

4. Турбо Паскаль 7.0. – Режим доступа: http://pascal.guti.ru/, вход свободный.

5. Фаронов В.В. Turbo Pascal: учеб. пособие. – СПб: Питер, 2007. – 376 с.

ПРИЛОЖЕНИЯ

Приложение 1

Образец оформления титульного листа отчета по лабораторной работе

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт дистанционного образования

Направление 09.03.01 «Информатика и вычислительная техника»

Кафедра Вычислительной техники

НАЗВАНИЕ РАБОТЫ – ПРОПИСНЫМИ БУКВАМИ

Отчет по лабораторной работе № ___
по курсу «Информатика 1.2»
Вариант № ___

Выполнил студент гр._____ _____ _____ _____

Подпись Дата И.О. Фамилия

Проверил _____ _____ _____ _____

должность Подпись Дата И.О. Фамилия

Томск – 201_

Учебное издание

ИНФОРМАТИКА 1.2

Методические указания к выполнению лабораторных работ

Составители

ТОКАРЕВА Ольга Сергеевна
ЛЕПУСТИН Алексей Владимирович

Рецензент
кандидат технических наук
доцент кафедры ВТ ИК

Е.А. Мирошниченко

Компьютерная верстка *В.П. Зимин*

Национальный исследовательский Томский политехнический университет
Система менеджмента качества
Издательства Томского политехнического университета сертифицирована
NATIONAL QUALITY ASSURANCE по стандарту BS EN ISO 9001:2008

ИЗДАТЕЛЬСТВО ТПУ. 634050, г. Томск, пр. Ленина, 30.
Тел./факс: 8(3822)56-35-35, www.tpu.ru