Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Амурский государственный университет»
Кафедра «Информационных и управляющих систем»

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ДИСЦИПЛИНЫ

«ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ»

Для специальности 230201.65 Информационные системы и технологии
(код и наименование специальности по Общероссийскому классификатору специальностей по образованию)

Благовещенск 2012

УМКД разработан к.т.н. Соловцова Л.А.
(степень, звание, фамилия, имя, отчество разработчиков)

_____

_____

Зав. кафедрой _____ /А.В. Бушманов /
(подпись) (фамилия, имя, отчество)

Протокол заседания кафедры № _____ от «____» _____ 200__ г.

СОГЛАСОВАНО:

Протокол заседания УМС специальности 230201.65 Информационные системы и технологии
(указывается название специальности (направление подготовки))

№ _____ от «____» _____ 201__ г.

Председатель УМСС _____ /В.В. Еремина /
(подпись) (фамилия, имя, отчество)

ОГЛАВЛЕНИЕ

1. ВЫПИСКА ИЗ ГОСУДАРСТВЕННОГО ОБРАЗОВАТЕЛЬНОГО СТАНДАРТА ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

Направление подготовки дипломированного специалиста

654600 – Информатика и вычислительная техника

Специальность

071900 (230201) – Информационные системы и технологии.

Квалификация – *инженер*

Индекс	*Наименование дисциплин и их основные разделы*	Всего часов
ОПД.Ф.12	**Технология программирования** Основные этапы решения задач на ЭВМ; критерии качества программы; диалоговые программы; дружественность, жизненный цикл программы; постановка задачи и спецификация программы; способы записи алгоритма; программа на языке высокого уровня; стандартные типы данных. Представление основных структур программирования: итерация, ветвление, повторение; процедуры; типы данных, определяемые пользователем; записи; файлы; динамические структуры данных. Списки: основные виды и способы реализации; программирование рекурсивных алгоритмов; способы конструирования программ; модульные программы; основы доказательства правильности.	102

2. РАБОЧАЯ ПРОГРАММА

по дисциплине "Технология программирования" для специальности 230201 "Информационные системы и технологии"

курс 1 семестр 1

Лекции 36 (час.) Экзамен 5

Практические занятия 18 (час.)

Лабораторных занятий 36(час)

самостоятельная работа _56__ (час.)

Всего часов _110___ час.

Составитель: ст.преподаватель Соловцова Л.А..

Факультет Математики и информатики

Кафедра Информационных и управляющих систем

1. Цели и задачи дисциплины.

1.1. Цели и задачи дисциплины

Целью курса является подготовка студентов к самостоятельной деятельности в области проектирования и сопровождения программных продуктов.

Задачами курса являются освоение технологии проектирования программ, их тестирование, изучение и использование методов сопровождения программных продуктов.

1.2. Требования к уровню освоения содержания дисциплины

В результате изучения дисциплины студент должен:

- знать принципы построения модульных программ, структурное программирование;
- освоить язык программирования Turbo Paskal
- уметь создавать коллективные программы, документировать программы.

2. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ.

2.1. Федеральный компонент.

Дисциплина «Технология программирования» является дисциплиной , входящей в блок общеобразовательных дисциплин федерального компонента для специальности 230201 «Информационные системы и технологии».

Государственный стандарт – СД.04

2.2.Наименование тем их содержание, объем в лекционных часах.

Тематический план лекционных занятий

№	Наименование темы	Кол-во часов
1	Начальные понятия программирования	2
2	Язык программирования как средство конструирования алгоритмов.	2
3	Основные элементы алгоритмических языков	2
4	Операторы алгоритмического языка.	2
5	Процедуры и функции	2

6	Массив как структура данных.	2
7	Обработка строк в языках программирования	2
8	Запись – структурный тип данных	2
9	Файлы.	2
	Итого	18

Тема 1. Начальные понятия программирования (2 часа)

Основные определения. Этапы программирования. Средства описания алгоритма. Алгоритмический язык программирования. Спецификация программы. Проверка правильности программы.

Тема 2. Язык программирования как средство конструирования алгоритмов.(2 часа)

Обзор языков программирования. Структура языка. Синтаксис и семантика.

Тема 3. Основные элементы алгоритмических языков.(2 часа)

Основные символы алгоритмического языка. Данные скалярные типы данных. Стандартные типы данных. Выражения.

Тема 4. Операторы алгоритмического языка. (4 часа)

Операторы ввода-вывода. Оператор присваивания. Составной оператор. Условные операторы. Селективный оператор. Операторы цикла.

Тема 5. Процедуры и функции.(2 часа)

Основные понятия. Параметры подпрограмм. Процедуры без параметров. Локальные и глобальные переменные.

Тема 6. Массив как структура данных.(2 часа)

Физическое представление массивов. Объявление одномерного и двумерного массивов. Использование массивов как параметров. Методы сортировки массивов.

Тема 7. Обработка строк в языках программирования. (2 часа)

Физическое представление строк. Объявление строк. Функции для их обработки.

Тема 8. Запись – структурный тип данных. (2 часа)

Объявление записей. Записи с вариантами. Обработка записей.

Тема 9. Файлы. (2 часа)

Основные понятия. Основные действия при использовании файлов.Создание файла. Обработка файлов. Добавление записей в файл. Замена записей в файле. Текстовые файлы.

2.3. Практические занятия, их содержание и объем в часах

Тематический план практических занятий.

№	Наименование темы	К-во час.
1	Операторы ввода-вывода	2
2	Условные операторы	2
3	Операторы цикла	4
4	Процедуры и функции	2
5	Одномерный массив	2
6	Двумерный массив	2
7	Обработка строк	2
8	Обработка записей	2
9	Работа с файлами	2
	Итого	18

2.4. Лабораторные занятия, их содержание и объем в часах.

Тематический план лабораторных занятий.

№	Наименование темы	К-во час.
1	Линейный информационный процесс	2
2	Условные операторы. Использование селективного оператора.	2
3	Написание процедур и функций.	2
4	Обработка одномерного массива	2

5	Обработка двумерного массива	2
6	Обработка строк	2
7	Работа с записями	2
8	Обработка файлов	4
	Итого	18

2.5. Самостоятельная работа студентов.

Для самостоятельной работы предлагается следующая тема "Изучение динамических структур данных". В рамках этой темы рассмотреть следующие подтемы:

- «Списки»;
- «Стек»;
- «Очередь»;
- «Двусвязный список»;
- «Деревья».

При изучении уделить внимание следующим вопросам:

- физическое и логическое представление в памяти ЭВМ;
- основные приемы работы с данными;
- написание программы для создания и обработки данных.

2.6. Вопросы к экзамену

1. Основные определения программирования.
2. Этапы программирования.
3. Средства описания алгоритма.
4. Характеристика алгоритмического языка программирования.
5. Составление спецификации программы.
6. Проверка правильности программы.
7. Основные символы алгоритмического языка.
8. Данные скалярные типы данных.

9. Стандартные типы данных.
10. Выражения.
11. Операторы ввода-вывода.
12. Оператор присваивания.
13. Составной оператор.
14. Условные операторы.
15. Селективный оператор.
16. Операторы цикла.
17. Основные понятия процедур и функций.
18. Параметры подпрограмм.
19. Процедуры без параметров.
20. Локальные и глобальные переменные.
21. Физическое представление массивов.
22. Объявление одномерного и двумерного массивов.
23. Использование массивов как параметров.
24. Методы сортировки массивов.
25. Физическое представление строк.
26. Объявление строк.
27. Функции для обработки строк.
28. Объявление записей.
29. Записи с вариантами.
30. Обработка записей.
31. Основные действия при использовании файлов.
32. Создание файла.
33. Обработка файлов.
34. Добавление записей в файл.
35. Замена записей в файле.
36. Текстовые файлы.

2.8.Виды контроля

Текущий контроль за аудиторной и самостоятельной работой обучаемых осуществляется во время проведения аудиторных занятий посредством устного опроса , проведения контрольных работ. Промежуточный контроль осуществляется два раза в семестр на основе анализа аудиторной и самостоятельной работы студента. Итоговый контроль осуществляется после успешного прохождения текущего и промежуточного контроля в виде зачета и устного или письменного экзамена при ответах на два вопроса в билете и дополнительные вопросы экзаменатора.

2.10. Требования к знаниям студентов, предъявляемые на экзамене

Студент, сдающий экзамен по данному предмету, должен показать знания по всем лекционным темам.

Знания студента оцениваются как отличные при полном изложении теоретического материала экзаменационного билета и ответах на дополнительные вопросы со свободной ориентацией в материале и других литературных источниках, а также при правильном написании программы на языке Turbo Paskal для решения предложенной задачи.

Оценка «хорошо»ставится при твердых знаниях студентом всех разделов курса, но в пределах конспекта лекций и обязательных заданий по самостоятельной работе с литературой, а также при небольших неточностях в написанной на языке Turbo Paskal программы.

Оценку «удовлетворительно» студент получает, если дает неполные ответы на теоретические вопросы билета, показывая поверхностное знание учебного материала, владение основными понятиями и терминологией, при неверном ответе на билет дает ответы на наводящие вопросы, а также при неверном использовании операторов в написанной на языке Turbo Paskal программы.

Оценка «неудовлетворительно» выставляется за незнание студентом разделов курса. Студент не дает полные ответы на теоретические вопросы билета, показывая лишь фрагментарное знание учебного материала, незнание основных понятий и терминологии, наводящие вопросы остаются без ответа, а также для реализации задачи выбран неверный алгоритм и есть затруднения в написании кода программы.

3. Учебно-методические материалы по дисциплине

3.1 Перечень обязательной (основной) литературы.

1. Н.И. Минакова, Е.С. Невская, А.А. Чекулаева Методы программирования. М.: Вузовская книга, 2000
2. Кнут Д. Искусство программирования. – М.: ИНФРА-М, 2004
3. Лавров С.С. Введение в программирование – М.: Вузовская книга, 2002
4. Турский В. Методология программирования. – М.:Высшая школа, 2001
5. Вирт Н. Алгоритмы и структуры. – М: Вузовская книга, 1999
6. Полякова А.Б., Круглов И.Ю. Программирование в среде Турбо Паскаль. – М.: Высшая школа, 2001.

3.1 Перечень дополнительной литературы.

1. Майер Б., Бодуэн К. Методы программирования. – М.: Мир, 1982
2. Прайс Д. Программирование на языке Паскаль. Практическое руководство. – М.:Мир, 1987
3. Алферова З.В. Теория алгоритмов. – М.: Статистика, 1989

3. ГРАФИК САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ

Содержание	Объем в часах	Сроки и форма контроля
3.1. Контрольная работа по теме «Конечные автоматы»	2 час.	Собеседование (6 неделя)
3.2. Контрольная работа по теме «Автомат с магазинной памятью»	2 час.	Собеседование (14 неделя)
3.3 Контрольная работа теме «Контекстно-свободные грамматики»	2 час.	Собеседование (18 неделя)

4. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ПРОВЕДЕНИЮ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ

Для самостоятельной работы предлагается следующая тема "Изучение динамических структур данных". В рамках этой темы рассмотреть следующие подтемы:

- «Списки»;
- «Стек»;
- «Очередь»;
- «Двусвязный список»;
- «Деревья».

При изучении уделить внимание следующим вопросам:

- физическое и логическое представление в памяти ЭВМ;
- основные приемы работы с данными;
- написание программы для создания и обработки данных.

5. ПЕРЕЧЕНЬ УЧЕБНИКОВ, УЧЕБНЫХ ПОСОБИЙ

5.1. Перечень основной литературы

1. Н.И. Минакова, Е.С. Невская, А.А. Чекулаева Методы программирования. М.: Вузовская книга, 2000
2. Кнут Д. Искусство программирования. – М.: ИНФРА-М, 2004
3. Лавров С.С. Введение в программирование – М.: Вузовская книга, 2002
4. Турский В. Методология программирования. – М.:Высшая школа, 2001
5. Вирт Н. Алгоритмы и структуры. – М: Вузовская книга, 1999
6. Полякова А.Б., Круглов И.Ю. Программирование в среде Турбо Паскаль. – М.: Высшая школа, 2001.

5.2 Перечень дополнительной литературы.

1. Майер Б., Бодуэн К. Методы программирования. – М.: Мир, 1982
2. Прайс Д. Программирование на языке Паскаль. Практическое руководство. – М.:Мир, 1987
3. Алферова З.В. Теория алгоритмов. – М.: Статистика, 1989

6. КОНСПЕКТ ЛЕКЦИЙ

Лекция №3

Операторы алгоритмического языка

Условные операторы

Условные операторы служат для организации ветвлений в программах. Условный оператор на основании истинности одного или нескольких условий выбирает некоторую последовательность операторов. Логические условия определяются на основе анализа постановки задачи. Синтаксические формы условных операторов включают условие и разделители, которые выделяют , последовательности операторов, выполняемые при истинности/ложности заданных условий. В задании сложного условного оператора (каждый из них, в свою очередь, может включать условные операторы) необходимо придерживаться принципа структурирования — разбиения сложных условий на простые.

В языке Паскаль принято следующее определение условного оператора:

<условный оператор>::=<полный условный оператор>|
<неполный условный оператор> <полный условный
оператор>::=if<логическое выражение>
then<оператор>else<оператор>
<неполный условный оператор>::=if<логическое выражение>
then<оператор>

Семантику условного оператора можно представить в виде схем (рис.).

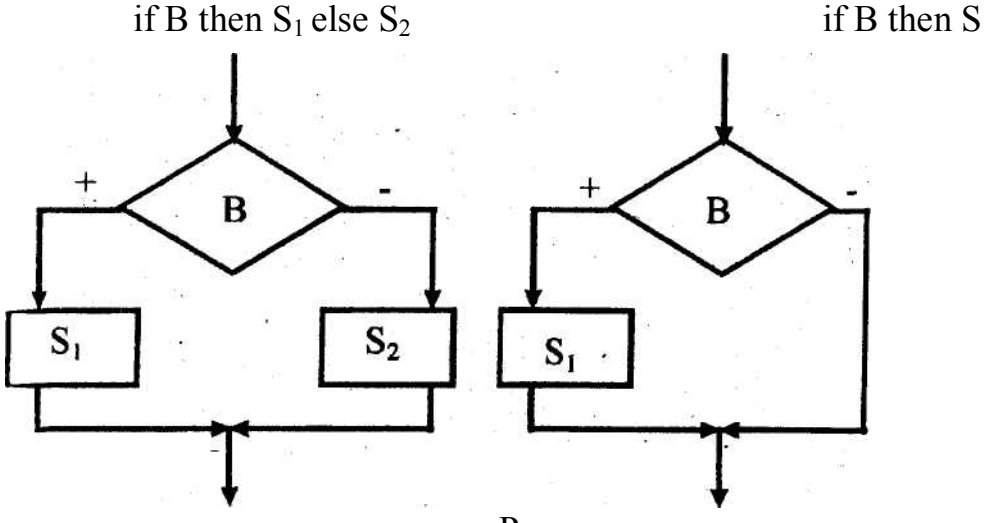


Рис.

Здесь B — логическое выражение, S1, S_2 — операторы. Вторую форму можно рассматривать как сокращенную запись оператора в том случае, когда отсутствует альтернатива S_2.

Пример**.** Вычислить значение функции

$$y=\begin{cases}\sqrt{x^2+1}, & если \quad A\geq 1\\ x+1, & если \quad A\prec 1\end{cases}$$

Программа имеет вид:

```
program PRP ;
{простое ветвление}
var A, x, y : real;
begin
     writeln ('введите значения x, A');
      readln(x,A) ;
      if A>=1 then y: =SQRT(x*x+l)
                 else y:=x+l;
      writeln('x=' ,x:б:3, 'y=' ,y:б:3, 'A=' ,A:6:3)
end.
```

Данная программа имеет логическую структуру типа ветвление.

Реализация разветвлений на несколько ветвей сводится к описанию разветвлений на две ветви. С помощью логического выражения, проверяется условие реализации одной ветви. В случае невыполнения первого условия, необходимо обратиться вновь к проверке условия с целью выбора одной из оставшихся ветвей и т.д.

Пример. Вычислить значение функции

$$z=\begin{cases}\ln x, & если \quad x\geq 1\\ 1, & если \quad -1\prec x\prec 1\\ e^x, & если \quad x\leq -1\end{cases}$$

```
program PRS; {разветвление на несколько ветвей}
var X, z: real ;
begin
   writeln('введите X');   readln(X); if X>=1 then  z:=LN(X)
else if X>-1 then z:=1
else z:=EXP(X) ;
writeln('X=',X:10:3,'z=',z:10:3) end.
```

В условном операторе после символов then и else по синтаксису может присутствовать один оператор. Если же при выполнении (невыполнении) условия необходимо выполнить последовательность операторов, то их надо объединить в единый составной оператор.

Например,

```
if x<y then begin r:=x; x:=y; y:=r end;
```

По правилам Паскаля символ else соответствует ближайшему символу then. В связи с этим условный оператор вида:

if B1 then if B_2 then S1 else S_2 надо трактовать как неполный условный оператор: if B1 then begin if B_2 then S1 else S_2 end

Селективный оператор

Селективный оператор, или оператор выбора, выполняет одно из перечисленных в программе действий в зависимости от значения выражения — селектора.

Оператор выбора в Паскале имеет следующий формат:

case i of
  L_1:S_1;
  L_2:S_2;
  …
  L_N:S_N;
 else S
end;

Семантику оператора CASE можно представить в виде схемы (рис.).

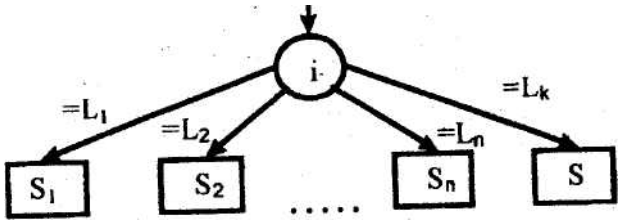


Рис.

Здесь L_k — списки констант выбора, состоящие из произвольного количества значений или диапазонов, отделенных друг от друга запятыми. Выражение (селектор) i имеет значение упорядоченного типа, кроме строкового, i и L_k принадлежат одному типу, $L_i \neq L_j$ при $i \neq j$.

В случае, если выражение i принимает значение L_k, выбирается и выполняется оператор $S_к$.

Если в списке выбора не будет найдена константа, соответствующая вычисленному значению селектора, управление передается оператору S.

Структура типа CASE (выбор) представляет интерес, т.к. позволяет довольно естественно описывать логические ветвления программы (например, выбор одной из многих возможностей).

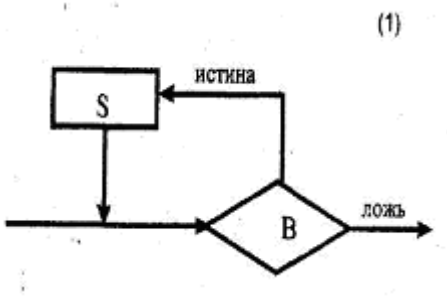
Операторы цикла

Для описания алгоритмов циклической структуры применяются операторы цикла.

Операторы цикла указывают, что определенную совокупность операторов следует выполнить последовательно несколько раз. В общем случае оператор цикла состоит из двух компонентов: заголовка цикла и тела цикла. Заголовок цикла задает правила, определяющие, сколько раз нужно выполнить тело цикла. Применяются различные способы определения заголовка цикла. Тело цикла — это последовательность операторов, ограниченная слева и справа специальными символами языка. Эти ограничители позволяют структурировать основной цикл включением в его состав других операторов цикла. Если в теле цикла S_o имеется оператор цикла S1, то оператор цикла S_o

называется объемлющим для оператор S1, а оператор S1- вложенным в цикл S_o. Любая пара операторов цикла должна быть непересекающейся, или один из пары циклов должен целиком вкладываться в другой.
В языке Паскаль циклические операторы с предусловием и с постусловием имеют соответственно следующий вид (рис.).
Здесь S — оператор, B — логическое выражение.
(1) — оператор цикла с предусловием. Здесь S выполняется нуль и более раз до тех пор, пока B имеет значение истина. Очевидно, S должно модифицировать B так, чтобы через конечное число шагов оно приняло значение FALSE (ложь).

while B do S;



repeat S until B;

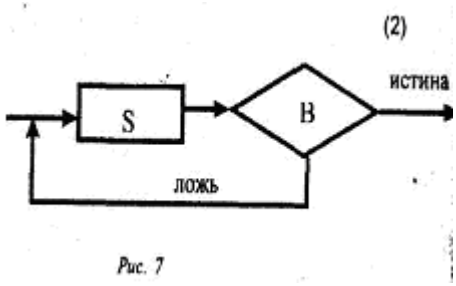


Рис. 7

(2) — оператор цикла с постусловием. Здесь S выполняется до вычисления логического выражения B, в результате тело цикла выполняется по крайней мере один раз.
Пример. Написать оператор цикла для вычисления четных степеней переменной x, начиная с x^2: x^2, x^4,..., до тех пор пока очередное значение степени x не станет больше 10^8. Используем оператор цикла с предусловием:

```
……
readln(x);
while x<=1E08 do
begin
  x:=SQR(x);
writeln(x)
end ;
……
```

Если применить оператор цикла с постусловием, то фрагмент программы будет иметь вид:

```
……
readln(x) ;
repeat
X:=SQR(x);
writeln(x)
 until x>1E08;
……
```

Кроме того, в Паскале имеет место оператор цикла с параметром:

for i:=A1 to A_2 do S; (3)

for i:=A1 downto A_2 do S; (4)

Здесь i — управляющая переменная (параметр цикла) задается в конструкции оператора цикла своими начальным и конечным значениями, определяемыми соответственно выражениями A1 и A_2.

Тип управляющей переменной и тип выражений совпадают и относятся к любому скалярному типу, кроме вещественного типа.

S — оператор, называемый телом цикла. Оператор S может быть составным.

Значение параметра цикла изменяется от начального до конечного по правилу i:=succ(i) в случае конструкции (3) и i =pred(i) в случае (4).

При нормальном завершении оператора значение параметра не определено.

Таким образом, оператор цикла (3) равносилен следующей последовательности операторов:

```
R1:=A1; R2:=A2 ;
if R1<=R2 then
  begin
    i:=Rl;S;
    while i<>R2 do
    begin
       i:=SUCC(i);S;
    end
 end ;
```

Оператор (4) может быть представлен аналогично:

```
R1:=A1; R2:=A2 ;
if in>=ik then
begin
i:=in;S;
while i<>R2 do
begin
  i:=pred(i);S;
end
end;
```

Переменные Rl и R2 имеют тот же тип, что и управляющая переменная i. Они создаются компилятором и к ним нет доступа из программы.

Управление параметром цикла i тоже осуществляется компилятором, поэтому не рекомендуется изменять этот параметр внутри цикла.

Пример. Изменить порядковые номера строчных букв латинского алфавита в соответствии с правилом: kn = $(ks)^2$+ 1, где kn, ks — соответственно старый и новый номера, соответствующие конкретной реализации алфавита. Program PR ;

```
{оператор цикла с параметром}
var l: char ;
kn:integer;
begin
for l : = 'a' to 'z' do
begin kn: =SQR (ORD (l) )+l;
writeln('letter-′,l,' ordnew=',kn)
end
end.
```

Пример. Вычислить значение логической функции $F = \neg a \wedge (y > 0) \vee (y > z)$ для двух значений a : «истина» и «ложь».

```
Program Logic ;
{оператор цикла с параметром}
var F,a:boolean;
z,y:integer;
begin
writeln('z,y?') ; readln(z, y) ;
for a:=TRUE downto FALSE do
begin F:=NOT a AND (y>0) OR (y>z) ;
writeln('a=',a,'F=',F)
end
end.
```

Необходимо заметить, что оператор цикла с предусловием (1) представляет собой управляющую структуру — повторение типа делать пока. Дополнительные типы операторов цикла (2)-(4) предусмотрены, чтобы упростить описание некоторых структур алгоритмов. Однако, их использование не нарушает хорошей структурированности программ, т.к. они имеют один вход и один выход . Синтаксис операторов цикла (1)—(4) обеспечивает возможность вложения управляющих структур друг в друга, т.е. операторы определяются рекурсивно.

Лекция №4

Процедуры и функции

Подпрограмма определяется в разделе описания процедур и функций программы.

<описание программы>::=<заголовок программы><блок>;

< заголовок программы>::=<заголовок программы>< заголовок функции>;
< заголовок процедуры>::=procedure<имя>[(< описание формальных параметров>)];
< заголовок функции>::=function<имя>[(< описание формальных параметров>)]:<тип>;

Здесь <имя> - имя подпрограммы (идентификатор); < описание формальных параметров> - последовательность имён формальных параметров, их типов и способов подстановки. Описания параметров отделяются друг от друга точкой с запятой; <тип> - тип возвращаемого функцией результата.

Описание формальных параметров может отсутствовать.

Блок (или тело программы) имеет ту же структуру, что и блок, являющийся телом программы.

Для вызова процедуры в нужном месте программы служит оператор процедуры:

< оператор процедуры>::=<имя процедуры>|<имя процедуры> (<список фактических параметров>)

Первый вид записи используется в тех случаях, когда процедура не имеет параметров. Фактические параметры разделяются в списке запятой.

Вызов функции:

<указатель функции>::=<имя функции>|<имя функции>(<список фактических параметров>)

При описании обращений к процедуре и к функции необходимо обеспечить соответствие фактических и формальных параметров по количеству, типу и порядку следования.

Без учёта контекста трудно различить указатель функции и оператор процедуры.

Пример. Если заголовок процедуры, вычисляющей $S=\sum_{i=0}^{n} y^i/i!$ задан следующим образом:

procedure SUM (y: real; N: integer; var S: real);

то обращение к ней имеет вид:

SUM(y, N, S);

Если пользоваться функцией с заголовком

function SUM (y: real; N: integer): real;

То обращение к ней может выглядеть следующим образом:

```
I: =SUM(y, N);
```

Итак, указатели функции используются в выражениях в качестве операндов, имеющих определённое значение. В соответствии с этим требуется, чтобы в теле функции присутствовал хотя бы один оператор присваивания с именем этой функции в левой части. Если в процессе работы функции не было выполнено ни одного такого присваивания, результат функции считается неопределенным.

Функция может возвращать в качестве результата значение скалярного, строкового ил ссылочного типа.

Пример.

Процедура, определяющая возможность построения треугольника по трём сторонам a, b и c и вычисляющая его площадь, может быть описана следующим образом:

```
procedure TRIANGL (a, b, c: real; var S: real) ;
var p: real;
begin
        p: =(a + b + c)/2;
        p:=p*(p-a)*(p-b)*(p-c);
        if p>0 then S:=SQRT(p)
                else S:=0
end;
```

Тогда значение площади треугольника может быть использовано в алгоритме, например, следующим образом:

```
TRIANGL (2, 3; 4, Q);
If Q<>0 then P: =5.7+2*Q;
```

Воспользуемся функцией:

```
function TRIANGLE (a, b, c: real): real;
var p: real;
begin
        p: =(a + b + c)/2;
        p: =p*(p-a)*(p-b)*(p-c);
        if p>0 then TRIANGLE:=SQRT(p)
                else TRIANGLE:=0
end;
```

Тогда два оператора в разделе действий программы примут вид:

```
Q: =TRIANGLE (2, 3, 4);
If Q<>0 then P: =5.7+2*Q;
```

Параметры подпрограмм

Заголовок подпрограммы содержит информацию о формальных параметрах. Здесь, помимо спецификации типа формальных параметров, необходимо также указать желаемый способ подстановки.

Принято различать два способа подстановки параметров:

- подстановка значения (параметр-значение);
- подстановка переменной (параметр-переменная).

Параметры – значения передаются основной программой в подпрограмму через стек в виде их копий, поэтому фактический параметр подпрограммой изменяться не может.

Параметры, которые называют параметрами-переменными, указываются заданием зарезервированного слова **var** перед их идентификаторами в списке формальных параметров. При передаче параметров-переменных в подпрограмму фактически через стек передаются их адреса в порядке, объявленном в заголовке подпрограммы. Следовательно, подпрограмма имеет доступ к этим параметрам и может их изменять.

Входные параметры подпрограммы могут быть как параметрами-значениями, так и параметрами-переменными. Выходные (и модифицируемые) – только параметрами-переменными.

Фактическими параметрами, соответствующими параметрам-значениям, могут быть имена переменных, константы, выражения. Фактическими параметрами, соответствующим параметрам-переменным – только имена переменных.

Пример**.** Отпечатать таблицу значений суммы

$$S=\sum_{i=0}^{m} 1/i \text{ для } m=1, 2, \ldots, 1024.$$

Введём процедуру вычисления суммы $S=\sum_{i=1}^{m} 1/i$.

```
Program SUMK;
const n=1024;
var x: real; m: integer;
procedure SUM (n: integer; var S: real);
var i: integer;
begin
```

```
        S: =0;
        for i: =1 to n do S: =S+1/i
end;
begin
        for m: =1 to n do
          begin
          SUM(m, x); write(m, x)
end;
end.
```

Здесь, в списке параметров процедуры SUM, n – параметр-значение, S – параметр-переменная.

Лекция №5

Массив как структура данных.

Массивы относят к структурированным типам данных. Все элементы (компоненты) массива принадлежат одному типу. Тип компонент массива может быть любым, число компонент массива задаётся при его описании и в дальнейшем не изменяется.

Массив — структура с так называемым случайным доступом, все его компоненты могут выбираться произвольно и являются одинаково доступными. К любому элементу массива можно обратиться, задав индекс (индексы), который однозначно определяет относительную позицию элемента в массиве. Тип индексов задаёт тип значений, которые используются для обращения к отдельным элементам массива. Он неявно определяет число элементов в массиве. Тип индекса может быть одним из упорядоченных типов, т.е. любым скалярным типом, кроме real. Следовательно, элементы массива всегда упорядочены.

В качестве индекса при обращении к элементу массива может быть использовано выражение.

Массив с одним индексом называют одномерным (часто его называют также вектором). Массив с двумя индексами называют двумерным массивом или матрицей. Первому индексу двумерного массива можно поставить в соответствие номер строки матрицы, второму — номер столбца. В Паскале допустимы массивы, содержащие три и более измерений.

Физическое представление массивов.

Для размещения одномерного массива ему выделяется сплошная область памяти. Таким образом, вектор можно представить в виде ряда смежных

позиций памяти, каждая из которых способна хранить одно значение, тип которого определяется при описании массива.
Под доступом к элементу массива понимают процедуру вычисления адреса этого элемента по его индексам.
Если предположить, что каждый элемент массива занимает p байт памяти, то адресом элемента с индексом i одномерного массива будет:

$$x + (i-b_1)p,$$

где x — адрес первого элемента массива, b_1 — нижняя граница индексов в массиве $x[b_1...B_1]$.
Для массивов с более чем одним измерением этот метод можно обобщить. Размещение называют построчным или поколонным в зависимости от того, меняются ли более быстро первые или последние индексы. Если предположить построчное размещение, то адрес элемента $x[i_1, i_2, i_n]$ в массиве x $[b_1..B_1, b_2..B_2, b_n..B_n]$ будет адресом первого элемента $x[b_1, b_2, b_n]$увеличенным на $(...((i_1 - b_1)(B_2 - b_2 + 1) + i_2 - b_2)(B_3 - b_3 + 1) + ... + i_n - b_n)p.$
В Паскале используется построчное размещение в памяти элементов матрицы.
Приведённые формулы вычисления адреса содержат обычные арифметические операции. При неправильном значении индекса (или индексов) будет получен неправильный адрес. Программист должен предусматривать возможность возникновения таких ошибок, строго доказывая правильность алгоритма или включая проверки правильности значений индексов (проверку нахождения индекса внутри границ диапазона). Такие дополнительные проверки могут быть вставлены автоматически, если использовать режим {$R+} .
Синтаксис описания массива:
TYPE<ИМЯ ТИПА>=ARRAY[<список индексных типов>]
OF<тип компонент>;
VAR<идентификатор массива>:<имя типа>;
или без использования раздела типов:
var<идентификатор массива>:ARRAY[<список индексных типов>]
OF<тип компонент>.
Здесь
<имя типа>,<идентификатор массива> — правильные идентификаторы;
<список индексных типов> — список одного или нескольких индексных типов, разделённых запятыми;
<тип компонент> — любой тип Турбо-Паскаля.
Пример**.** Описать массив, состоящий из 100 элементов типа real:

```
const   n100=100;
type    r100 = array [1. .n100] of real ;
var     a :  r100;
```

или, проще:

```
var a   : array [1. . 100] of real;
```

В основном, работа с массивом ведётся покомпонентно. Для компонент допустимы все операции, которые определяются их типом. Компонента

обозначается с помощью переменной с индексами . Например, с помощью оператора

a[10] := 7.9;

десятому элементу массива присваивается значение, равное 7.9.

Приведём пример описания двумерного массива:

```
const n10 = 10;
type tmatr = array[1. .n10,l. .n10] of integer;
var a   : tmatr;
```

Двумерный массив можно интерпретировать как вектор, каждый элемент которого в свою очередь является вектором. Выше описанный двумерный массив можно описать иначе:

```
Const  n10 = 10;
type tvect = array[1. .n10] of integer;
var a : array[1..n10] of tvect;
```

Оба описания позволяют интерпретировать массив *a* как поле размером 10 x 10. Для обращения к элементу массива можно использовать обозначение

a [i] [j] или a[i,j].

Оба варианта равноправны.

Использование массивов как параметров.

Если процедуре (функции) будет передан массив, желательно описать в ней массив как параметр с атрибутом var, даже в том случае, если значение массива внутри процедуры (функции) не изменяется. Это нужно для того, чтобы не тратить время и память на размещение копии массива внутри процедуры. Параметры-массивы обязательно должны быть описаны типом, имеющем имя. Например, заголовок процедуры

procedure pr (var a:array [1. .10] of real; n : integer; var s,p : real)
;

является неправильным. Правильная форма записи следующая:

```
type tvect = array[1. .10] of real;
procedure pr (var a: tvect; n : integer; var s,p: real);
```

Приведём несколько примеров использования одномерных массивов и реализации алгоритмов на Паскале.

Пример 1. Задан массив *A* [1 ... *N]* с компонентами целого типа и целое значение *X.* Найти в массиве наименьший индекс компоненты с заданным значением.

Алгоритм содержит 2 пункта:

— ввод исходных данных;

— поиск в массиве заданного значения. Вариант программы:

```
{метод линейного поиска}

CONST N=30;
TYPE   T=ARRAY[1. .N] OF INTEGER;
VAR     A:T;
X, I : INTEGER;
BEGIN
FOR I :=1 TO N DO
READ(A[I]);
READ(X);
I:=0;
REPEAT
I:=I+1
UNTIL (A[I]=X) OR (I=N) ;
IFA[I]<>X
THEN   WRI TELN (' элемент отсутствует')
ELSE     WRITELN('I=',I)
END.
```

Заметим, что цикл поиска продолжается при выполнении двух условий: элемент не найден и не весь массив просмотрен. Поэтому по окончанию цикла уточняется условие, при котором произошёл выход из цикла.
Используя распространённый метод барьера (или фиктивного элемента), можно упростить условие окончания цикла. Для размещения «барьерного» элемента необходимо при описании массива зарезервировать дополнительную компоненту (в начале или в конце массива). Разместим барьерный элемент в конце массива. Алгоритм поиска примет вид:

```
I:=0; A[N+1] :=X;
REPEAT
I:=I+1
UNTIL A [I] =X;
IF I=N+1 THEN
WRITELN( ' элемент отсутствует ')
ELSE
WRITELN('I= ' ,I) ;
```

Число сравнений элементов при реализации линейного по иска в среднем равно *N/2,* если отыскиваемый элемент содержится в массиве, и равно *N,* если элемент со значением *X* отсутствует в массиве.
Рассмотрим проблему поиска в упорядоченном массиве. В этом случае можно применить метод половинного деления (бинарный поиск): интервал индексов делится пополам, пока не будет найден нужный элемент или левая граница окажется больше правой. При этом интервал индексов компонент, среди которых ведётся поиск, постоянно сужается.
Пример 2. Бинарный поиск значения элемента в массиве.

```
CONST N=30;
TYPE T=ARRAY [1..N] OF INTEGER;
VAR A:T;
X,I,J,K :INTEGER;
BEGIN
FOR I: =1 TO N DO
READ(A[I]);
READ(X);
I: = 1; J: =N; {I -левая граница интервала, J - правая}
REPEAT
K: = (I+J) DIV 2; {K-середина интервала}
IF X>A[K] THEN
I: =K+1 {изменяем левую границу}
ELSE
J: =K-1 {изменяем правую границу}
UNTIL (A[K]=X) OR (I>J) ;
IF A [K] =X THEN
WRITELN (' найденный номер =', к)
ELSE
WRITELN (' элемент отсутствует')
END.
```

Заметим, что число требуемых сравнений при реализации двоичного поиска не более, чем $\log_2 n$.

В следующих примерах программу не будем записывать полностью.

Пример 3. Определить номер первого элемента, большего *M* и сумму элементов, ему предшествующих.

```
S:=0; I:=l;
WHILE (I<=N) AND (X[I]<=M) DO
BEGIN
S:=S+X[I] ; I:=I+1 END; IF I=N+1 THEN
WRITELN (S,' нет элементов больших I')
ELSE
WRITELN(S,I)
```

Пример 4**.** Вычислить значение полинома л-ой степени в заданной точке. Если мы хотим применить эффективный алгоритм для этой задачи, то используем вычисление по схеме Горнера:

$y = (...((0 + A[N]) * X + A[N-1]) * X + ...A[1]) * X + A[0]$, или на Паскале :

```
Y:=0;
FOR I: =N DOWNTO 0 DO
Y:=Y*X+A[I] .
```

Пример 5. Осуществить циклический сдвиг элементов вектора на одну позицию влево.

```
X: =A [1]; {первый элемент запоминаем во вспомогательной
            переменной}
FOR I:=1 TO N-1 DO {каждыйследующийэлемент}
A [I] :=A[I+1] ;       {переставляем на место левого соседа}
A [N] : =X;       {Первый элемент массива становится последним}
```

Для выполнения сдвига на *L* позиций (L — любое число ≥ 0) вначале найдём фактическую константу сдвига (L< N):

L:=L mod N,

а затем можно использовать цикл:

```
for J:=1 to L do
begin
X:=A[1];
FOR I:=1 TO N-1 D0 A[I] :=A[I+1] ;
A[N] :=X end;
```

Пример 6. Заданы два массива *X[1 ..N], Y[1 ..N].* Составить программу, которая присваивает логической переменной *EQ* значение результата проверки условия: равны ли поэлементно данные два массива. Пустые массивы (N=0) считаются равными.

Введём переменную *j,* присвоим переменной *EQ* значение высказывания: «среди первых пар не обнаружено отличий», и запишем два варианта программы.

```
{первый вариант}
j:=0; EQ:=true;
while j<>N do
begin j : = j +1; EQ: =EQ and
X[j]=Y[j] end;
writeln('EQ=',EQ);

{второй вариант}
j:=0; EQ:=true;
while (j <>N) and EQ do
begin j :=j+l; EQ:= X[j] =Y[j] end;
writeln('EQ=',EQ)
```

Программа (2) отличается от программы (1) тем, что в программе (2) повторения заканчиваются, как только обнаружено отличие в какой-то паре. Число повторений в этих двух программах будет совпадать только в случае, если массивы поэлементно равны. Второй алгоритм эффективнее первого для неравных массивов большой длины.

Пример 9. Составить программу вычисления среднеарифметических значений положительных элементов каждой строки матрицы A[1..N,1..M], результаты поместить в вектор B[1..N].

Алгоритм:

1) ввод исходных данных;
2) вычисление компонент вектора;
3) вывод результатов.

Алгоритм пункта 2:

Цикл по номерам строк {I:=l...N}
{ S:=0;K:=0;{S — сумма>0 элементов, К — количество} Цикл по номерам элементов внутри строки {J:=1..M} Подсчёт суммы и количества положительных элементов в строке;
Найти среднее арифметическое и поместить результат в вектор.

Алгоритм на Паскале:

```
CONST NM=30; MM=50;
            {NM и MM обозначают максимальные длины столбцов и строк,
здесь                    они взяты произвольно}
TYPE VEKT = ARRAY[1..NM] OF REAL;
MATR = ARRAY[1..NM,1..MM] OF REAL;
VAR  A:MATR;
B:VEKT;
S:REAL;
I,J,K,N,M:INTEGER;
{N, M - фактические размеры матрицы}
BEGIN
{ввод}
READLN(N,M);
FOR I :=1 TO N DO
FOR J:=l TO M DO
READLN(A[I,J]);
{вычисление}
FOR I:=l TO N DO
BEGIN
S:=0; K:=0;
FOR  J:=l TO M DO
IF A[I,J]>0 THEN
BEGIN S:=S+A[I,J] ; K:=K+1 END;
IF K<>O THEN B [I] : =S/K ELSE B [I] : =0
END;
{вывод}
FOR I :=1 TO N DO
WRITE(B[I])
END.
```

Лекция №6

Обработка строк в языках программирования

Строка — последовательность символов, принадлежащих конечному множеству символов, или алфавиту.

Как и любая другая информация, символьные данные хранятся в памяти ВМ в двоичном виде: каждому символу ставится в соответствие некоторое неотрицательное число, называемое кодом символа (код символа занимает один байт).

Максимальная длина строки составляет 255 символов. Строки называют динамическими, поскольку они могут изменять длину в пределах объявленных границ.

Синтаксис описания переменных строкового типа:

type <имя типа>= string[<максимальная длина строки>];

var < идентификатор имя типа>;

или без описания типа:

var <идентификатор>: string[<максимальная длина строки>];

Указание максимальной длины строки при её описании не является обязательным, при отсутствии её максимальная длина строки принимается равной 255.

Пример.

```
type    tstr20=string[20];
var       str20   : tstr20;
          str255 : string;
          strl0   : string[10];
```

Переменная str20 может хранить строку не более 20 символов, str255 — не более 255 символов, strl0 — не более 10.

Текущая длина строки может изменяться от 0 до указанной в описании максимальной длины строки .

Например, после выполнения оператора

```
str20 := 'block'  ;
```

строка str20 будет иметь длину 5 символов.

Если количество символов, определяющих строковое значение, превышает максимально допустимую длину строки, излишек строки отсекается справа.

Например, после выполнения оператора:

```
str10 : = ' переменные символьного типа' ;
```

строка strl0 получит значение: ' переменные'.

Тип STRING без указания длины является базовым строковым типом, и он совместим со всеми производными строковыми типами.

К любому символу в строке можно обратиться по номеру (индексу) символа в строке, например, str20[3].

В самом начале строки расположен дополнительный байт. Он имеет индекс 0 и содержит код, равный числу символов в строке (текущей длине). Значение текущей длины строки (например, строки s) можно получить, используя функцию ORD:

```
L:=ord(s[0] ) ;
```

В некоторых языках программирования символ длина строки отсутствует, вместо него вводится символ «конец строки» (в Си — это нуль-символ). Он является последним элементом строки. Наличие символа конец строки или длина строки означает, что Количество ячеек строки должно быть на одну больше, чем число символов, которое необходимо разместить в памяти. Нуль-символ так же, как и символ длина строки, программисту надо самостоятельно размещать в конец или начало строки. Эту задачу выполняет процедура (или функция) — читать вводимую строку (readln в Паскале или scanf в Си).

Над строковыми данными допустимы следующие операции::
сцепление (+) — применяется для сцепления нескольких строк в одну;
сравнение — применяется для сравнения строк друг с другом.

Для выполнения сравнения строк используются операции отношения: =, <>, <, >, <=, >=. Сравнение строк происходит посимвольно слева направо. Строки равны, если их длины одинаковы и элементы строк посимвольно совпадают.

Для работы со строками в языке Паскаль имеются стандартные процедуры и функции.

Функции:

LENGTH(S) -функция типа INTEGER, выдаёт текущую длину строки S;

CONCAT (S1 , S2, ..., SN) — функция типа STRING, возвращает строку, представляющую собой конкатенацию (слияние) строк Sl, S2,..., Sn;

COPY(S, L, N) — функция типа STRING, возвращает подстроку длины N, начинающуюся с позиции L в строке S;

POS (S1, S) — функция типа INTEGER, определяет номер начальной позиции первого вхождения подстроки S1 в строку S; если подстрока не найдена, возвращает значение нуль.

Процедуры:

DELETE (S, L, N) — удаляет из S подстроку длины N, начинающуюся с позиции L в строке;

INSERT (S1, S, L) — вставляет в S подстроку S1, начиная с позиции L;

STR (X, S) — преобразует числовое значение X типа REAL или INTEGER в строку символов S; параметр X может содержать форматы преобразования (аналогично их использованию в процедуре WRITE или WRITELN);

VAL (S, X) — преобразует строковое значение S (строку цифр) в значение числовой переменной X.

Пример . Заменить в строке S каждую группу символов S1 на S2.

```
var s, sl, s2 : string;
    n, k      : integer;
begin
```

```
 readln (s);  readln (s1) ;   readln (s2) ;
n:=length(s1);
 k:=pos (s1, s) ; {k - позиция первого вхождения s1 в s}
while k<>0 do         {повторять пока s1 входит в s }
 begin
     delete (s, k, n) ; {удалить в строке s с k-той позиции n символов}
     insert(s2,s,k); {вставить s2 в s с k-ой позиции}
     k:=pos (s1, s) {k - позиция первого вхождения s1 в s}
end;
writeln (s)
end.
```

В процедурах и функциях, работающих со строками, надёжнее всего описывать формальные параметры типом STRING. Если тип параметра-строки является производным от типа string, то необходимо использовать тип, сконструированный в разделе описания типов.

Например,

```
type tstr20 = string [20] ;
…
procedure prov (var s1, s1: tstr20) ;
```

Лекция №7

Запись – структурный тип данных

Запись — конечное множество элементов, состоящее в общем случае из компонент разного типа (размера) и представляющих логически единое целое. Элементами записи могут быть скалярные переменные, массивы, записи низших уровней, массивы записей. Элементы записи называют *полями.* Поля именуются, доступ к полям осуществляется по именам. Записи

используются для представления иерархически упорядоченных совокупностей данных, состоящих из разнотипных элементов.

Запись — обобщение понятия массива, при котором не требуется однородности элементов.

В оперативной памяти элементы записи располагаются последовательно в том порядке, в котором они описаны. Объём памяти, занимаемой записью, складывается из длин её полей. При выделении памяти каждая переменная выравнивается на определённую границу памяти. Например, переменная типа integer или longint размещается в памяти с границы, кратной 2 или 4 байтам соответственно. Естественно, что при таком размещении памяти могут появиться неиспользованные участки памяти. Чтобы сократить их количество, нужно позаботиться о том, чтобы при описании записей непосредственно друг за другом не следовали элементы с различными границами выравнивания (если, конечно, допустимо изменение порядка следования элементов записи). Адрес какой-либо компоненты записи относительно начального адреса записи называется смещением компоненты.

Синтаксис определения типа записи (комбинированного типа):

<комбинированный тип>::= RECORD

<последовательность описаний> END

Последовательность описаний может содержать: 1) фиксированную часть, 2) фиксированную и вариантную части, 3) вариантную часть. Фиксированная часть состоит из перечисления списка полей и их типов.

Запись может быть элементом массива. Массив, элементами которого являются записи, называют массивом записей.

Например,

```
type rec = record
                a : integer;
                b : real;
                c : string[10]
```

```
        end;
    var elem : rec;
      tabl   : array [1..100] of rec;
```

Обращение к элементам записи осуществляется с использованием составных имён. Составное имя начинается с имени записи и содержит список имён полей (разделённых точками), входящих в цепь, ведущую к требуемому элементу.

Например, к полям записи можно обратиться по именам:

elem.a, elem.b, elem.c, elem.c[5],

а к полям *i*-й компоненты массива записей — по именам : tabl[i].a, tabl[i].b, tabl[i].c, tabl[i].c[l].

Для сокращения обозначений полей записи (когда ведётся работа с несколькими полями одной и той же записи) используется оператор присоединения:

WITH <переменная-запись>DO<оператор>.

Внутри оператора, входящего в оператор присоединения, компоненты записи обозначаются с помощью только имён полей (имя переменной-записи перед ними не указывается). Заголовок операторов может содержать список переменных-записей:

WITH<переменная-запись> {,<переменная-запись>}DO<оператор>.

Например:

```
WITH elem DO
BEGIN
      a: =1998; b:=37.5; c: = 'А.И.Иванов'
END;
```

или:

```
FOR I: =l TO 10 DO
WITH tabl[i] DO
BEGIN READ (a, b); READLN(c) END
```

Записи удобны при создании информационных систем, при работе с файлами. При передаче процедуре и функции большого числа различных параметров их удобно объединять в запись.

Записи с вариантами.

Записи не обязательно должны содержать одни и те же компоненты. Если запись объединяет поля с частично совпадающими компонентами, то применяется «запись с вариантами». Запись с вариантами, кроме фиксированной части (которая может отсутствовать) содержит вариантную часть. Вариантная часть предусматривает возможность задания типа, содержащего определения нескольких вариантов структуры. Различие может касаться как числа компонент, так и их типа. Каждый вариант характеризуется задаваемым в скобках списком описаний присущих ему компонент. Списку предшествует одна или несколько меток. Перед всей группой списков стоит заголовок варианта. Он задаёт особое поле записи — поле признака. Эта компонента записи указывает, о каком варианте идёт речь.

Синтаксис определения вариантной части таков:

<вариантная часть>: = CASE<поле признака>: <имя типа> OF<вариант>
{<вариант>}

<вариант>::=<список меток варианта>: (<список полей>) I <пусто>

<список меток варианта>::=<метка варианта>{<метка варианта>}

<метка варианта>::=<константа>

<поле признака>::=<имя>: I <пусто>

В любое время доступными полями являются поля только одно го из всех возможных вариантов, описанных в типе. При хранении все варианты размещаются на одном участке памяти. Размер этого участка определяется вариантом, имеющим наибольшую длину.

Пример 15. Вычислить площадь и периметр одной из геометрических фигур: квадрата, прямоугольника или треугольника. Будем использовать запись, в состав которой входят постоянные переменные P и S, а также одна,

две или три варьируемые переменные (в зависимости от признака фигуры) для размещения длин сторон . Значение признака используем в дальнейшем для выбора расчётных формул. Характеристики фигур помещаются в массив записей.

Алгоритм на Паскале:

```
CONST NM= 30;
 TYPE REC=RECORD
                    S, P:REAL;   {s - площадь, P - периметр фигуры}
                    CASE PR: 1. . 3 OF
                      1: (A: REAL);   {длина стороны квадрата}
                      2: (B, C: REAL); {длина стороны прямоугольника};
                      3: (D, E, F: REAL)   {длина стороны треугольника}
                   END;
VAR X: ARRAY [I. .NM] OF REC;
    I, N: INTEGER; {N- фактический размер массива}
      Q.: REAL;       {Q - полупериметр}
BEGIN
    READLN (N);
    FOR I: =1 TO N DO
    WITH X [I] DO
    BEGIN
          READLN (PR);
               CASE PR OF
                  1: BEGIN READLN (A); S: =A*A; P: =4*A END;
                  2: BEGIN READLN (B, C); S: =B*C; P: =2*(B+C) END;
                  3: BEGIN READLN (D, E, F); P: =D+E+F; Q: =P/2;
                        S: =SQRT (Q*(Q-D)*(Q-E)* (Q-F))
                END;
     END {CASE}
END; {FOR}
```

```
 FOR I: =l TO N DO
WITH X [I] DO
BEGIN
        CASE PR OF
             1: WRITE (A);
             2: WRITE (B, C);
             3: WRITE (D, E, F)
         END;
         WRITELN(S, P)
       END
END.
```

В заголовке варианта поле признака может отсутствовать, т.е. заголовок может использовать только тип меток (тип, на основании которого мы различаем варианты), например:

```
        type t=l..2;
         var x: record
                        a: integer;
                         case t of {t -тип меток}
                                  1:   (b1, b2: real);
                                  2:   (c: real; d: char)
        end;
q: t; {q - селектор, имеет тот же тип, что и тип меток}.
```

Приведём пример операций над описанными данными

```
x. a:= 5;
with x do
        for q:=l to 2 do
                case q of
                1: begin b1: =0 .1; b2: =0. 2;
```

```
                writeln (q, '  ', b1, b2)
            end;
        2: begin c:=0.3;     d:='*';
                writeln (q, '   ', c, '  ', d )
    end
end
```

При использовании записей следует помнить, что:

—все имена полей должны быть различными, даже если они встречаются в различных вариантах;

—если поле с меткой M в списке пустое, то оно записывается так M:() ;

—любой список полей может иметь только одну вариантную часть, и она должна следовать за фиксированной частью (или частями);

—вариантная часть сама может содержать варианты (допускаются вложенные варианты).

Лекция №8

Файлы.

Файл — поименованная область памяти на внешнем носителе, предназначенная для хранения информации. В файлах могут храниться программы, данные, тексты документов, закодированные изображения. Файл может быть входным и выходным.

К достоинствам файла можно отнести следующее: 1) данные, организованные в виде файла, могут использоваться в нескольких программах; 2) файл сохраняет свои значения по окончанию работы программы, создавшей файл; 3) файл — единственный способ размещения данных очень большого объёма (если оперативная память не позволяет этого сделать).

Различают стандартные файлы и файлы, созданные пользователем. Стандартные файлы имеют стандартные имена, не подлежащие изменению. Каждому файлу пользователя должно быть присвоено уникальное имя. Имя

состоит из собственно имени (1-8 символов) и расширения, необязательного типа (3 символа), он отделяется от первой части имени точкой. Тип файла может характеризовать хранимую в нём информацию. В соответствии с установленными соглашениями можно применять следующие стандартные расширения файлов:

1) pas, asm, c — содержит текст программы на языках Паскаль, ассемблер, Си;

2) exe, com — содержит текст откомпилированной программы, которая может быть выполнена;

3) txt, doc — текстовые файлы;

4) pcx, pic — файлы картинок;

5) sys — служебные системные файлы.

Файл содержит компоненты одного типа. Длина файла не указывается при объявлении файла.

Для того, чтобы использовать файл в программе, необходимо выполнить следующие действия:

1. описать переменную файлового типа одним из способов;
   f:file of<тип>;{типизированный файл}
   f:text; {текстовый файл}
   f:file; {файл без типа}
2. поместить имя файла в переменную символьного типа (например, name);
3. связать файловую переменную с именем файла;
   assign(f,name);
4. открыть файл для чтения /записи оператором:
   reset(f) или rewrite(f);
5. читать/писать запись из файла/в файл, используя переменную (например, zap):
   read(f,zap) или write(f.zap).
6. закрыть файл по концу работы:
   close(f).

Для обнаружения конца файла используется функция eof(f).

Одновременно могут быть открыты несколько файлов. В ходе выполнения программы один и тот же файл может быть открыт для записи, а затем использован для чтения. Открытый на запись файл изначально является пустым, он содержит лишь маркер конца файла. Каждый оператор write или writeln осуществляет добавление новой информации, после чего маркер сдвигается к новому концу файла. Оператор writeln (в отличие от write) добавляет в файл литеру конца строки. По смыслу, маркер конца файла — это следующая доступная компонента, в которую будет помещён следующий символ (если он есть).

При работе с файлами автоматически проверяются ошибки ввода-вывода. Если проверка возникновения ошибки ввода-вывода включена (по умолчанию или с помощью директивы компилятора {$I+}), то при возникновении ошибки выполнение программы автоматически завершается с

выдачей сообщения об ошибке. Если проверка возникновения ошибки ввода-вывода отключена директивой компилятора {$I-}, то при возникновении ошибки программа продолжает свою работу и может перехватывать проверку наличия или отсутствия ошибок ввода-вывода в свои руки. Функция IORESULT проверяет результат выполнения последней операции ввода-вывода на наличие ошибок и возвращает значение 0 или номер ошибки.

Типизированные файлы.

Типизированными (двоичными) файлами называются дисковые файлы, состоящие из нумерованной последовательности записей (компонент) одинакового типа. Тип записей в файле задаётся при его объявлении. Длина каждой записи постоянна. Можно определить позицию каждой записи в файле и напрямую считать (или записать) эту запись.

В типизированном файле, в отличие от текстового файла, информация хранится во внутренней (двоичной) форме. Отсутствие необходимости в преобразовании данных позволяет их записывать и считывать с большей скоростью.

В типизированном файле, также в отличие от текстового файла, возможен прямой доступ к записям файла, наряду с последовательным способом доступа.

Типизированные файлы полезны для временного хранения информации в процессе выполнения программы или для передачи большого объёма промежуточных данных, полученных в одной программе, другой программе.

Создание файла

Рассмотрим пример.

Пример 22. Создать файл, состоящий из записей с полями: ф.и.о. студента, номер курса и номер группы. Признаком конца вводимых записей будем считать пустую строку (пустая фамилия).

Программа на Паскале

```
type rec=record
                fio : string[20];
      kurs, group : integer
         end;
var zap : rec;
      f : file of rec;
    name: string;
begin
    writeln (' задайте имя файла');
    readln (name);
    assign (f, name); {привязка файла к конкретному имени}
    rewrite (f) ;       {создаём новый пустой файл}
    write ('введите фамилию и.о. =>...');
    readln (zap.fio);
    while zap.fio<>' ' do
    begin
```

```
            write ('курс и группа =>...');
            readln (zap.kurs, zap.group);
            write (f, zap); (занесение содержимого записи zap в файл в
двоичном коде}
            write ('фамилия и.о. =>...');
            readln(zap.fio)
            end;
            close (f);
            writeln (‘файл создан’)
        end
```

Обработка файла

После того как файл создан, он пригоден для обработки (в частности, для просмотра). Обработка (просмотр) осуществляется строго последовательно, начиная с первой компоненты файла. При этом на каждом шаге доступна лишь одна компонента файла.

Выполним просмотр ранее созданного файла, выдав на экран те записи, фамилии которых начинаются и заканчиваются на одну и ту же букву.

Приведём фрагмент программы:

```
reset (f) ;    {открыть файл для работы с ним}
while not eof (f) do{проверить, не достигнут ли конец файла}
  begin
  read (f, zap); {читать запись из файла в переменную zap}
        with zap do
         begin
              n:=length(fio);
              if fio[l]=fio[n] then
                   writeln (fio:20,' ',kurs:3,group:3)
         end
end;
```

Добавление записей в файл (в конец файла)

Для добавления записей в файл будем использовать следующие стандартные процедуры и функции:

seek(f,n) — установить указатель файла на компоненту с номером n.

Указатель перемещается к компоненте с номером n, начиная счёт с нуля, т.е. первая компонента файла имеет номер 0, вторая -1,..., последняя — n.

filesize(f) — определить количество компонент в файле.

Выполним дополнение ранее созданного файла.

```
reset (f);
seek (f, f ilesize (f)) ; (установить указатель за последней компонентой файла}
writeln ('задайте фамилию и.о. =>...');
readln(zap.fio);
```

```
while zap.f.io<>' ' do
begin
        writeln {'курс и группа =>. ..'} ;
        readln(zap.kurs, zap.group);
        writeln(f,zap);
        writeln('фамилия и.о. =>...');
        reafln (zap,fio)
end ;
close (f)) ;
```

Замена записей в файле

Если известен номер (или содержимое) записи, которую необходимо заменить, то это можно сделать, используя оператор seek или последовательный просмотр записей в файле. Пусть необходимо поменять значениями первую и последнюю записи в файле. Это можно выполнить следующим образом:

```
reset(f);
read(f,zap);
seek(f,filesize(f) -1);
read(f,zapl);
seek(f,0); write(f,zap1);
seek(f,filesize (f)-1); write (f, zap);
```

Оператор seek (f,k) позволяет организовать прямой доступ к записям файла. Прямой доступ в файле предоставляет много возможностей для организации работы с ним. Одним из методов является использование индексных файлов.

Пример 23. Дан файл записей, каждая из которых содержит ведения о студенте и состоит из полей: личный номер студента, Фамилия и инициалы, номер курса, номер группы, средний балл. Отсортировать данный файл по полю «средний балл».

Для выполнения сортировки файла воспользуемся подходом, который был использован в примере 14 под названием *индексная сортировка.*

Введём два вспомогательных файла: IND — индексный файл (файл номеров записей), KEY — файл ключей доступа к записям исходного файла.

Файл ключей инициируется просмотром исходного файла.

Индексный файл формируется в процессе сортировки файла ключей (используется метод «сортировка включением», см. раздел 3.2). При этом файл ключей не изменяется. Номера записей в создаваемом индексном файле задают порядок чтения записей в исходном файле, отсортированных по заданному ключу.

Программа на Паскале:

```
TYPE REC=RECORD
                NOM : BYTE;
                  FIO : STRING[20];
  KURS,GROUP : BYTE;
```

```
                SR :REAL
    END;
   st=string[12];
VAR namef : st;
{индексная сортировка файла}
PROCEDURE SORTF_IND (VAR name :St);
VAR:    F:FILE OF REC;              {исходный файл}
       IND:FILE OF INTEGER;         {файл  номеров записей}
       KEY:FILE OF REAL;            {файл ключей записей}
        zap:rec;
    n,m,i,j:integer;
          x:real ;
{функция чтения из индексного файла записи с номером к}
function out__ind (к: integer):integer;
   var 1: integer ;
begin
   seek(ind,k);
   read{ind,1);
   out_ind:=l
end;
{функция чтения из файла ключей записи с номером к}
function out_key(k:integer):real;
    var x: real;
begin
       seek (key, k) ;
       read (key, x);
       out_key: =x
       end;
BEGIN
       assign (f, name) ; reset (f) ;
       assign (ind,'find') ; rewrite (ind);
       assign (key, 'fkey'); rewrite (key);
       while not eof (f) do
       begin
           read (f, zap) ;
           write (key, zap. sr); {создание файла ключей}
       end;
       close(key);
       reset(key);
       m:=0;
       write (ind, m) ;
       n: = filesize (f) ;
{создание файла индексов, соответствующего}
{отсортированному порядку следования записей}
(в файле ключей}
```

```
      for j :=1 to n-1 do
      begin
          x:=out_key (j);
           i:=j-l;
          while (i<>-1) and (out_key (out_ind (i))>x) do
          begin
              m: =out__ind(i);
              seek (ind, i + 1);
              write (ind, m)
               i:=i-1;
          end;
          seek (ind. i+1);
          write (ind, j)
       end;
       for i:=0 to n-1 do
       begin
            j:=out__ind (i);
     seek(f,j);
     read(f,zap);
     with zap do
     BEGIN
       WRITE(N0M:3,' ');
       WRITE(FIO);
                FOR J:=LENGTH (FIO)+1 TO 20 DO WRITE (' ');{*}
                WRITELN(KURS:5, GROUP:5 , SR:7:3)
            END;
       END;
     close(f);
     close(kew);
     close(ind)
end;
begin
   readln{namef};
   sortf_ind(namef)
end.
```

Строка, помеченная символом «*», содержит цикл для дополнения значений FIO пробелами до 20 символов.

Описание типа записи (типа **rec)** можно внести в описание процедуры в данном случае.

Программу можно сделать независимой от типа записей файла, если поместить тип **rec** и тип ключа сортировки в отдельный модуль, содержащий типы конкретной задачи, и подключить его оператором **uses** (см. главу 5).

Приведём пример использования файлов для размещения входной и выходной информации в задаче о преобразовании в ПОЛИЗ (пользовательскую индексную запись) арифметического выражения.

Пример 24. Дано арифметическое выражение, Преобразовать его в ПОЛИЗ.

ПОЛИЗ — постфиксная запись арифметического выражения, в которой операнды предшествуют знакам операций.

Например, a*b/c-d*e преобразуется к виду ab*c/de*-,

(a + b)*(c — d) — к виду a b + c d — *;

a + b * (c — d) — "к виду a b c d — * +

Используем следующий алгоритм: просматривая арифметическое выражение, операнды помещаем в ПОЛИЗ. Знаки операций помещаем в стек, если он пуст. Если стек не пуст, переписываем из стека в ПОЛИЗ все операции с приоритетом, большим либо равным приоритету текущей операции. Затем знак текущей операции помещаем в стек. Открывающую скобку всегда заносим в стек. Закрывающая скобка выталкивает из стека в ПОЛИЗ все операции, вплоть до открывающей скобки. При этом открывающая скобка удаляется из стека, но в ПОЛИЗ не записывается. Закрывающая скобка в стек не заносится. По окончании входного текста содержимое стека переписывается в ПОЛИЗ.

В качестве операндов арифметического выражения будем использовать однобуквенные идентификаторы, в качестве знаков операций - ' + ', '-', '*', '/'. Входной текст будем считывать посимвольно из файла F. ПОЛИЗ (выходной текст) записываем в файл G.

Введем приоритеты для скобок и операций.

Операция	Приоритет
(	0
)	1
+, -	2
*, /	3

Рис. 14

Опишем функцию «приоритет». Процедуры и функции для работы со стеком описаны в разделе 3.7.2. В программе укажем только их заголовки.

```
type link= ^rec;
      rec = record
              inf:char;
              next: link
              end;
var   F, G:file of char;
    c:char;
    start: link;
    {функция определения приоритета}
function pr(x:char):integer;
begin
  case x of
      '(':    pr:=0;
      ')':    pr:=l;
```

```
  '+', '-':      pr:=2;
  '*', '/':      pr:=3
  end
end ;
{ занесение элемента в стек}
procedure IN_STACK(ch:char);
…
{ Выбор элемента из стека}.
function OUT_STACK: char;
…
{Получение значения вершины стека}.
function TOP_STACK:char;
…
{ текст операторной части программы.}
begin
  assign(F,'namef'); reset(F);
  assign(G,'nameg');rewrite(G);
  START := nil ;
  while not eof { F ) do
   begin
   read {F, c) ;
   if c in ['a' . . 'z', 'A' .. '-Z'] then {операнд}
      write { G, c }
   else
      if c = ' (' then {открывающая скобка}
         IN__STACK ( c )
      else
       if c in ['+', '-', '*', '/', ')'] then
              {знак операции или ) }
       begin
            while (START<>nil)and
                  (PR(TOP_STACK ) >= PR(c) ) do
              begin cl: = OUT_STACK;
                  write ( G, cl)
              end;
       if c<>')' then {завершение работы со знаком операции}
          IN_STACK ( c )
      else {завершение работы с символом ')' }
           if   (START <> nil ) and (TOP_STACK = ' ( ' )
           then
              c: = OUT_STACK
           else
             begin
               write { ' Нет соответствия скобок') ;
               HALT
```

```
            end
        end                                                    .
        else
          begin
            write (' В арифметическом выражении - ' );
            write ('недопустимый символ', c) ,
       HALT
  end
end;
while START <> nil do
  begin c:= OUT_STACK; write (G, c) end;
close ( F );
close ( G );
end.
```

Пример 25. Даны два непустых отсортированных по неубыванию файла, состоящих из компонент типа real. Выполнить слияние этих файлов в один выходной, отсортированный по неубыванию файл.

Воспользуемся переменными old1, old2 для обозначения входных файлов, newf — для обозначения выходного файла. Вспомогательные логические переменные stop 1, stop2 будут использованы для проверки условия: указатель файла не указывает на конец файла (old1 или old2).

Опишем процедуры inl, in2 для чтения из входных файлов (чтению предшествует проверка условия «не конец файла»),

```
        var old1,old2,newf :file of real;
        name1,name2, name3:string;
        a,b :real;
  stop,stop1,stap2:boolean;
procedure in1; {процедура чтения числа из файла old1}
begin
  stop1: =not eof (old1);
  if stop1 then read (old1, a)
end;
procedure in2; (процедура чтения числа из файла old2}
begin
  stop2 :=not eof(old2);
  if stop2 then read(old2,b)
end;
begin
  readln(name1); reset(old1,name1);
  readln(name2);reset(old2,name2);
  readln(name3);reset(newf,name3};
  in1; in2 ;
  stop :=stop1 and stop2;
  while stop do
  begin
```

```
 if a<>b then
            begin write {newf, a) ; in1 end
       else
            begin write (newf ,b) ;in2 end;
          stop:          =stop1          and          stop2
  end;
   while stop1 do begin write (newf, a) ; in1 end;
   while stop2 do begin write (newf ,b) , in2 end;
   close(old1); close(old2); close(newf)
end.
```

Текстовые файлы

Текстовые файлы состоят из символов, объединённых в строки. Длина строки текстового файла переменная (от 0 до 255 символов). В конце каждой строки файла размещается признак конца строки: это последовательность кодов ASCII — 13 (CR) и 10 (LF). В конце всего файла — признак конца файла: код ASCII - 26 (CTRL-Z).

Текстовый файл является последовательным файлом, и этим он отличается от типизированного файла, для которого доступ может быть выполнен к любой записи файла.

Текстовый файл может быть открыт на запись, чтение или дополнение.

Для доступа к записям текстового файла, используются процедуры: READ, READLN, WRITE, WRITELN. В них можно указывать переменное число параметров. Параметры могут иметь тип: integer, real, char, string, boolean (тип boolean может использоваться только в процедурах вывода).

Элементы текстового файла преобразуются во внутреннюю форму при выполнении процедуры READ (READLN) и из внутренней формы в цепочку литер — при выполнении процедуры WRITE (WRITELN). Это преобразование выполняется в соответствии с типом используемого параметра.

Синтаксис определения операторов ввода-вывода:

```
read[ln] ( [f,] <список ввода>);
write[ln](   [f,] <список вывода>);
```

здесь f — имя файловой переменой.

Элементы, заключённые в квадратные скобки, не являются обязательными в операторах ввода-вывода.

Если файловая переменная указана, осуществляется обращение к дисковому файлу или к логическому устройству. Примеры логических устройств: CON — клавиатура или экран дисплея; PRN — принтер.

Если файловая переменная не указана, происходит обращение к стандартным файлам INPUT или OUTPUT (что соответствует вводу с клавиатуры или выводу на экран дисплея).

Пример.

```
var     a : real;
        b : integer;
```

```
        c : char;
        s : string;
         f : text;
…
{использование дискового файла}      {использование стандартных файлов}
{запись  в файл f }                  {запись в файл output }
 writeln(f, a, b, c);                writeln(a, b, c);
{или write(f,a);write(f,b);writeln(f,c)}   {или write(a,b);writeln(c)}
{чтение из файла f)                  {чтение из файла input}
readln(f,s)                          readln(s)
```

Для работы с текстовыми файлами используются следующие стандартные логические функции:

EOLN(f) — возвращает значение true, если в файле достигнут маркер конца строки, false — в противном случае;

SEEKEOLN(f) — пропускает пробелы и знаки табуляции до маркера конца строки или до первого значащего символа и возвращает значение true, если маркер обнаружен, false - в противном случае;

SEEKEOF(f) — пропускает все пробелы, знаки табуляции и маркеры конца строки файла до маркера конца файла или до первого значащего символа и возвращает значение true, если маркер обнаружен, false — в противном случае.

Пример. Выдать на печать построчно, а внутри строки посимвольно содержимое текстового файла, например, такого:

1 2 3 4 5
6 7 8 9 0.

Для выдачи на печать воспользуемся библиотечным модулем PRINTER. В нём описана переменная LST, которая связывается с логическим устройством печати (PRN, LPT1 или LPT2).

```
uses printer;
var f : text ;
     c : char;
begin
    assign(f,'namef);
    reset(f);
```

{первый вариант}	{второй вариант}

while not eof(f) do
begin
while not eoln(f) do
begin
read(f,c);
write(lst,c)
end;
readln(f);
writeln(lst)
end
	while not seekeof (f) do
	begin
	while not seekeoln(f) do
	begin
	read(f, c);
	write(1st,c)
	end;
	readln(f);
	writeln(lst)
	end

```
end.
                    {результаты работы}
{первый вариант}    {второй вариант — пробелы пропущены }
1 2 3 4 5           12345
6 7 8 9 0           67890
```

Пример**.** Выдать на печать в виде одной строки содержимое текстового файла (пробелы опустить),

Фрагмент программы:

```
  while not seekeof (f) do
   begin
         read(f,c);
        write(lst,c)
  end;
```

Результаты выдачи на печать: 1234567890.

Нетипизированные. файлы

Нетипизированные файлы совместимы со всеми типами файлов и позволяют организовать высокоскоростной обмен данными между диском и памятью. Информация из нетипизированного файла считывается (или записывается в файл) блоками записей. Длина записи 1 может быть указана при инициализации файла в процедурах reset, rewrite в качестве второго параметра (или принимается по умолчанию равной 128 байт):

```
reset( f, 1 );   rewrite( f, 1 ).
```

Для организации обмена данными между файлом и памятью используются следующие процедуры:

```
blockread( f, buf, n [, count ] );
blockwrite( f, buf, n [, count ] ),
```

где f — файловая переменная, buf — имя переменной, которая будет участвовать в обмене данными с файлом, n — количество записей, которое надо прочитать/записать, count ~- фактическое количество переданных записей (необязательный параметр). За одно обращение к процедурам может быть передано n*1 байт.

7. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

Практическое занятие №1

Программа на языке Pascal состоит из следующих разделов:

1. Описание меток
2. Описание констант
3. Описание типов
4. Описание переменных
5. Описание процедур и функций
6. Инструкции

Структура программы в общем виде:

```
Label
{описание меток}
Const
{описание констант}
Type
{описание типов}
Var
{описание переменных}
{описание процедур и функций}
Begin
{инструкции программы}
End.
```

Инструкции WRITE и WRITELN

Инструкция WRITE предназначена для вывода на экран монитора сообщений и значений переменных.

Синтаксис инструкции:

WRITE(список переменных:формат);

Кроме имен переменных в список можно включить сообщение – текст заключенный в апострофы.

Для переменных типа INTEGER формат – это целое число, определяющее ширину поля вывода. (количество позиций на экране). Например инструкция WRITE(d:4) показывает, что для вывода значения переменной d используется 5 позиций. Неиспользуемые позиции заполняются пробелами, а само изображение выравнивается по правой границе поля.

Для переменной типа REAL формат представляет собой 2 целых числа, разделенных двоеточием. Первое число определяет ширину поля вывода, второе – число цифр, стоящих справа от десятичной точки. Если задать только ширину поля, то на экране появится число, представленное в формате с плавающей точкой.

Пусть переменные x1, x2 типа REAL имеют значения 13.25, -0.3401, тогда в результате выполнения инструкции

WRITE(‘x1=’, x1:5:2, ‘ x2=’, x2:12);

на экране будет выведено

x1=13.25 x2=-3.40100E-01

Если ширины поля, указанной в формате, недостаточно для вывода значения переменной, то выводится число в формате с плавающей точкой и десятью цифрами после запятой(17 позиций-все число).

После выполнения инструкции WRITE курсор остается в той позиции экрана, в которой он находился после вывода последнего сообщения этой инструкции. Следующая инструкция WRITE начинает вывод именно с этой позиции.

Инструкция WRITELN отличается от инструкции WRITE тем, что после вывода сообщения или значений переменных курсор переводится в начало следующей строки.

Инструкция READ предназначена для ввода с клавиатуры значений переменных (исходных данных). Синтаксис инструкции:

Read(переменная1, переменная2, ……переменнаяN);

Задание

1. Написать инструкции для ввода значений переменных вещественного, целого и символьного вида.
2. Написать инструкции для вывода на экран

```
**************************************
*        результаты работы программы        *
**************************************
```

3. Написать различные инструкции для вывода на экран значений переменных вещественного типа X, Y и как будет выглядеть вывод на экране.
4. Имеется переменная X=-4.561, что будет выведено на экран при выполнении инструкции

– WRITE (‘Значение переменной X=’, X:7:4);
– WRITELN (‘Значение переменной X=’, X:10:4);
– WRITE (‘Значение переменной X=’, X:7);
– WRITELN (X:7:4);

Контрольные вопросы

1. Из каких разделов состоит программа на языке Pascal?
2. какие инструкции предназначены для ввода информации?
3. Что представляет собой формат для вывод а вещественных значений?
4. Что такое формат с плавающей точкой?
5. Чем отличается инструкция WRITELN от инструкции WRITE?

Практическое занятие №2
Условные операторы

Пример. Вычислить значение функции

$$z=\begin{cases}\ln x, & если \quad x\geq 1\\ 1, & если \quad -1\prec x\prec 1\\ e^{x}, & если \quad x\leq -1\end{cases}$$

```
program PRS; {разветвление на несколько ветвей}
var X, z: real ;
begin
   writeln('введите X');   readln(X); if X>=1 then  z:=LN(X)
else if X>-1 then z:=1
else z:=EXP(X) ;
writeln('X=',X:10:3,'z=',z:10:3) end.
```

Пример.

```
program VIBOR ;
{ПРОГРАММА СООБЩАЕТ МЕНЮ: СПИСОК ТРЕХ
ВСТРОЕННЫХ ФУНКЦИЙ ПО ПЕРВОЙ БУКВЕ ИМЕНИ
ФУНКЦИИ НА ЭКРАН ВЫВОДИТСЯ ПОЛНОЕ НАЗВАНИЕ ЭТОЙ
ФУНКЦИИ И ЕЕ ЗНАЧЕНИЕ ПРИ ЗАДАННОМ АРГУМЕНТЕ}
{V - ПРИЗНАК ВЫХОДА ИЗ ПРОГРАММЫ}
var REJ: char ;
          X: real;
          ch: char;
begin
       clrscr;
      writeln(' введите значение аргумента') ;
      readln(X) ;
      writeln{'X=',x);
      while TRUE do begin
      writeln (' Укажите нужный режим :');
      writeln (' A-abs     ');
      writeln (' S-sin     ');
      writeln (' C-cos     ');
      writeln (' T-arctg   ');
      writeln (' V -выход  ');
      readln(REJ) ;
      case REJ of
      'A','a' :writeln{'абсолютная величина abs(x) =' ,abs(x)) ;
      'S','s' :writeln('синус угла sin(x)=',sin(x>);
      'C','c' : writeln ('косинус угла cos (x) =', cos (x) );
      'T','t' :writeln ('арктангенс угла arctg (x) =', arctg(x));
      'V','v' :begin
```

```
    writeln (' Программа закончила обработку ) ;
    writeln Сданного значения аргумента') ;
    writeln (' Вы хотите продолжить работу') ;
    writeln (' с другим значением аргумента y/n) ?') ;
    readln(ch);
    if <ch='n') or (ch='N') then HALT
    else writeln (' Введите значение аргумента') ;
    readln(X);
   end
  else writeln (' Указанного имени нет в меню')
 end
end
end.
```

Пример. Вычислить значение логической функции $F = \neg a \wedge (y > 0) \vee (y > z)$ для двух значений a : «истина» и «ложь».

```
Program Logic ;
{оператор цикла с параметром}
var F,a:boolean;
z,y:integer;
begin
writeln('z,y?') ; readln(z, y) ;
for  a:=TRUE  downto  FALSE  do
begin F:=NOT a AND (y>0) OR (y>z) ;
writeln('a=',a,'F=',F)
end
end.
```

Задание

1. Записать условие для определения зимнего месяца.
2. Решить квадратное уравнение.
3. Из трех чисел выбрать максимальное.
4. Начислить заработную плату работнику, который отработал X рабочих дней по У часов, заданы расценки за час. За сверхурочные он получает двойной тариф. Норма рабочего времени – 8 час.
5. Из пункта А в пункт В(расстояние X) в 10 часов выехал автомобиль со скоростью 60км/час. Определить время прибытия автомобиля в город В.

Контрольные вопросы

1. Какие виды условных операторов существуют?
2. Перечислить логические операции?
3. Какие операции используются для записи условия?
4. Какие значения принимает логическая переменная?
5. Какой оператор используется для построения меню?

Практическое занятие №3
Операторы цикла

Пример. Изменить порядковые номера строчных букв латинского алфавита в соответствии с правилом: kn = $(ks)^2$+ 1, где kn, ks — соответственно старый и новый номера, соответствующие конкретной реализации алфавита. Program PR ;

```
{оператор цикла с параметром}
var l: char ;
kn:integer;
begin
for l : = 'a' to 'z' do
begin kn: =SQR (ORD (l) )+l;
writeln('letter-′,l,' ordnew=',kn)
end
end.
```

Пример. Вычислить значение логической функции $F = \neg a \wedge (y > 0) \vee (y > z)$ для двух значений a : «истина» и «ложь».

```
Program Logic ;
{оператор цикла с параметром}
var F,a:boolean;
z,y:integer;
begin
writeln('z,y?') ; readln(z, y) ;
for a:=TRUE downto FALSE do
begin F:=NOT a AND (y>0) OR (y>z) ;
writeln('a=',a,'F=',F)
end
end.
```

Пример. Написать оператор цикла для вычисления четных степеней переменной x, начиная с x^2: x^2, x^4,..., до тех пор пока очередное значение степени x не станет больше 10^8. Используем оператор цикла с предусловием:

```
……
readln(x);
while x<=1E08 do
begin
  x:=SQR(x);
writeln(x)
end ;
……
```

Если применить оператор цикла с постусловием, то фрагмент программы будет иметь вид:

```
……
readln(x) ;
repeat
X:=SQR(x);
writeln(x)
 until x>1E08;
……
```

Задание

1. Найти сумму четных чисел, выбрав их из 20-ти введенных.
2. Выполнить табулирование функции f = cos (x+1.6) + sin (x*2.8), x принадлежит интнрвалу [1, 25], шаг табулирования – 0.25. Написать программу с использованием различных видов цикла.
3. Вывести все простые числа в заданном интервале.
4. Найти сумму всех делителей для заданного числа.
5. Найти наименьший общий делитель для заданных чисел.

Контрольные вопросы

1. Что такое циклический алгоритм?
2. Какие циклы существуют в программировании?
3. Что такое параметр цикла?
4. Чем отличаются циклы с предусловием и с постусловием?
5. В каком случае необходимо использовать цикл с предусловием?

Практическое занятие №4
Процедуры и функции

Пример**.** Отпечатать таблицу значений суммы

$$S=\sum_{i=0}^{m} 1/i \text{ для } m=1, 2, \ldots, 1024.$$

Введём процедуру вычисления суммы $S=\sum_{i=1}^{m} 1/i$.

```
Program SUMK;
const n=1024;
var x: real; m: integer;
procedure SUM (n: integer; var S: real);
var i: integer;
begin
        S: =0;
        for i: =1 to n do S: =S+1/i
end;
begin
```

```
        for m: =1 to n do
          begin
          SUM(m, x); write(m, x)
end;
end.
```

Здесь, в списке параметров процедуры SUM, n – параметр-значение, S – параметр-переменная.

Задание

1. Написать функцию для определения площади треугольника, и программу, использующую функцию для определения площади нескольких треугольников.
2. Написать функцию для определения является ли число простым.
3. Написать процедуру для вывода на экран шапки таблицы.
4. Написать процедуру для определения длины окружности и площади круга, определяемого этой окружностью.

Контрольные вопросы

1. Чем отличается функция и процедура?
2. В каком случае следует использовать функцию, а в каком процедуру?
3. Что такое фактические и формальные параметры?
4. Что такое глобальные переменные и как они объявляются?
5. Какие существуют формы передачи параметров в процедуры ?

Практическое занятие №5

Одномерный массив

Пример 7. Подсчитать количество вхождений в текст каждой из букв 'A' .. 'Z'. Текст будем вводить посимвольно, последним символом строки будем считать символ «точка». В этой задаче удобно использовать массив счётчиков с символьным типом индексов. Программу запишем полностью.

```
VAR К: ARRAY [' A' . . ' Z' ] OF INTEGER; { массив счетчиков}
CH:CHAR; BEGIN
FOR CH: = 'A' то ' Z' DO {обнуление массива счётчиков}
К[CH] :=0;
REPEAT                    {ввод и анализ символов}
READ(CH);
IF (CH>='A') AND  (CH<='Z') THEN
к[сн] :=к[сн]+1 {добавление 1 к счётчику}
UNTIL CH= ' . ' ;
FOR I: = 'A' то ' Z' DO {вывод символов и их количеств}
WRITELN{I, ' - ',K[CH] :3) END.
```

Пример 8**.** Даны два упорядоченных но неубывающих массива A[1 .. N], B[1 .. M]. Объединить их в один упорядоченный массив C[1 ..N+M].
Алгоритм:
1) просматривать с начала массивы A[1..N) и B[1..M], сравнивать их элементы, меньший элемент помещать в массив C[1..N+M];
2) поместить в массив *C* оставшиеся элементы из массива A или B.
Будем использовать следующие обозначения: I, J, К — индексы элементов в массивах *A, B* или *C* соответственно. Алгоритм на Паскале:

```
I:=l; J:=l; K:=l;
WHILE (I<=N) AND (J<=M) DO
      BEGIN
       IF A[I]<B[J] THEN
                        BEGIN
                          C[K] :=A[I] ;  I: =I + 1
                        END
                      ELSE
                        BEGIN
                          C[K] :=B[J] ; J:=J+1
                        END;
        K:=K+1
      END;
FOR I:=I TO N DO
  BEGIN
    C[K] :=A[I] ;
     K:=K+1
  END;
FOR J : = J  TO  M  DO
  BEGIN
   C [K] :=B[J] ;
   K:=K+1
 END;
```

Пересылку оставшихся элементов из массива *A* (или *B)* можно было бы осуществить с помощью вспомогательной переменной (например, *L)* в качестве параметра цикла:
FOR L:=I TO N DO . . . ;
FOR L: = J TO M DO . . . ;
Оба варианта заголовков цикла равноправны.
Приведём примеры использования двумерных массивов.

Задание

1. В массиве A(50) найти максимальный элемент и его местоположение.
2. В массиве A(100) найти сумму положительных и отрицательных чисел?

3. Выполнить сортировку массива B(150).
4. В отсортированном массиве A(70) найти местоположение числа X, используя бинарный поиск.
5. В массиве B(50) найти среднеарифметическое всех элементов массива, заменить все нулевые элементы массива найденным средним.

Контрольные вопросы

1. Какие операции можно выполнить над одномерным массивом?
2. Чем определяется местоположение каждого элемента массива?
3. Какие операторы используются для обработки массивов?
4. В чем суть бинарного поиска?
5. Какие методы используются для сортировки массивов?

Практическое занятие №6
Двумерный массив

Пример. Разделить каждый столбец матрицы на максимальный элемент в этом же столбце. Представим преобразование матрицы в виде процедуры. Воспользуемся типом MATR:

```
PROCEDURE OBR_MATR (VAR A : MATR;   M, N : integer) ;
VAR I,J : INTEGER;
BEGIN
FOR J: =1 то м DO {цикл по номерам столбцов}
BEGIN
MAX : = A [1 , J]; {первый элемент столбца}
FOR I  : =2  TO N DO
{сравнение со всеми остальными в столбце}
IF A[I,J]>MAX THEN MAX;=A[I, J];
IF MAX<>0 THEN
FOR I :=1 TO N DO
A[I,J] :=A[I,J]/MAX
END
END;
```

Пример **.** Проверить, имеет ли матрица X[1..N, 1..M] две одинаковые строки.

При составлении данной программы воспользуемся идеей из примера 6. Введём следующие переменные: i — для обозначения номера строки, которую сравниваем; *j* — для обозначения номера строки, с которой сравниваем строку i; *к* — номер сравниваемой пары в строках i,j

```
NOT_FOUND:=true; i:=0;
while (i<=N-l) and NOT_FOUND do
begin
i:=i+l; j :=i;
while (j<=N) and NOT_FOUND do
```

```
begin
j:=j+l; k:=0;
EQ:=true;
while (k<=M) and EQ do
begin
к:=k+l;
EQ:=A[i,k]=A[j,k] ;
end ;
NOT_FOUND: =not EQ
end;
end;
writeln( EQ) ;
```

Задание

1. В двумерном массиве A(10X5) определить значение минимального элемента и его местонахождение?
2. В двумерном массиве B(10X10) определить среднеарифметическое всех элементов, расположенных на главной диагонали.
3. Преобразовать двумерный массив B(5X20) в одномерный массив.
4. В двумерном массиве найти количество элементов кратных 7.

Контрольные вопросы

1. Как выполняется обработка элементов двумерного массива?
2. Чем определяется местонахождение элемента в двумерном массиве
3. Какие операции выполняют над элементами двумерного массива?
4. Как используется массив в качестве параметра подпрограммы?

Практическое занятие №7
Обработка строк

Пример . Заменить в строке S каждую группу символов S1 на S2.

```
var s, sl, s2 : string;
    n, k      : integer;
begin
 readln (s);  readln (s1) ;   readln (s2) ;
n:=length(s1);
```

```
 k:=pos (s1, s) ; {k - позиция первого вхождения s1 в s}
while k<>0 do         {повторять пока s1 входит в s }
 begin
     delete (s, k, n) ; {удалить в строке s с k-той позиции n символов}
     insert(s2,s,k); {вставить s2 в s с k-ой позиции}
     k:=pos (s1, s) {k - позиция первого вхождения s1 в s}
end;
writeln (s)
end.
```

Пример. Описать функцию определения длины самой длинной цифровой последовательности, входящей в строку.

```
Function MAX_DL (var s : string) : integer ;
const m=['0' . . '9'];
var k, kmax, i, n : integer ;
begin
  n:=length(s);
  kmax:=0;
  i:=1;
  b:=true;
  while b do
begin
  k: = 0;
  while (i< = n) and (s[i] in m) do {подсчёт количества подряд идущих цифр}
begin
  i: = i+l;    k: = k+l
end;
if k>kmax then kmax: = k;
while (i < = n) and not ( s [i] in m) do {пропустить не цифры}
          i=i+1;
```

```
if i = n + l then
                b: = false
end;
MAX_DL: = kmax
end;
```

Задание

1. В массиве строк выявить все вхождения заданной подстроки.
2. В строке определить количество слов.
3. В массиве строк найти самое длинное предложение.
4. В массиве строк заменитьвсе вхожденмя заданной подстроки на «XXX».
5. Из строки удалить все вхождения заданной подстроки.

Контрольные вопросы

1. Как выполняется объявления строки?
2. Как выполняется ввод и вывод строк?
3. Перечислить процедуры и функции для обработки строк?
4. Можно ли выполнять посимвольную обработку строки?

Практическое занятие №8
Обработка записей

Пример. Запись содержит сведения о студенте и состоит из полей: личный номер студента, фамилия и инициалы, номер курса, номер группы, средние оценки за каждый год учёбы. Вычислить средний балл каждого студента и отсортировать список студентов по успеваемости.

Алгоритм содержит пункты:

— ввод исходных данных;

— вычисление среднего балла для каждого студента;

— сортировка списка студентов (массива записей) по успеваемости;

— выдача результатов.

Уточним пункт — сортировка.

При сортировке массива записей по выбранному полю (его обычно называют ключевым полем или ключом) происходит обмен значений всех полей массива записей. Это приводит к увеличению времени выполнения

программы. Для повышения эффективности программы обмен всех полей записи заменяют обменом номеров элементов. Вводят два вспомогательных массива: массив ключей и массив индексов (номеров). Первоначальное значение массива индексов: 1, 2, 3, ... Сортировка массива записей по ключевому полю сопровождается изменением порядка следования номеров в массиве индексов. Результатом сортировки будет не отсортированный массив записей, а вспомогательный массив индексов (номеров). Значения компонент массива индексов будут указывать на порядок следования компонент в исходном массиве записей после выполнения сортировки по выбранному ключу (исходный порядок компонент в массиве записей сохраняется). Такой подход к процессу сортировки называют индексной сортировкой.

Введём два вспомогательных массива: IND — индексный массив целого типа, KEY — массив ключей вещественного типа (в нашей задаче сортировка должна быть выполнена по полю SR типа REAL).

Программа на Паскале:

```
CONST NMAX=30;
TYPE ZAP=RECORD
                    NOM: BYTE;
                    FIO: STRING [20];
                    KURS, GROUP: BYTE;
                    OC: ARRAY[1. .5] OF REAL;
                    SR: REAL
      END;
   V= ARRAY [1. .NMAX] OF ZAP;
   TKEY = ARRAY [1. . . NMAX] OF REAL;
   TIND = ARRAY [1. .NMAX] OF INTEGER;
   REC = RECORD
                KEY: TKEY;
                IND: TIND END;
```

```
VAR N, I, J, K: INTEGER;
        LIST: V;
        KI: REC;
{индексная сортировка}
PROCEDURE SORT_IND (VAR A: REC; L: INTEGER);
VAR I, J, K: INTEGER;
        BOUND: BOOLEAN;
BEGIN
         BOUND: =TRUE; I: =l;
         WHILE BOUND DO
         BEGIN
            BOUND: =FALSE;
            FOR J: =l TO L-I DO
                WITH A DO
                        IF KEY [IND [J] ] >KEY [IND [J+l] ]
                                THEN BEGIN
                                        K: =INDY [J];
                                        INDY [J]:=IND [J+1];
                                        IND [J+1]:=K;
                                        BOUND: =TRUE
                        END;
        IF BOUND THEN I: =I+1
    END
END;


BEGIN
{ввод исходных данных}
WRITE (' задайте количество студентов в списке   =>...');
READLN (N);
```

```
FOR I: =1 TO N DO
        WITH LIST [I] DO
            BEGIN
             WRITE ('задайте личный номер студента =>...');
                READLN (NOM);
             WRITE (' задайте фамилию студента =>...');
                READLN (FIO);
             WRITE (' курс и группа =>...');
                READLN (KURS, GROUP);
             WRITE (' средние оценки за каждый год =>...');
               FOR j: =1 TO 5 DO
                    READ (OC[j]);
       READLN
END;
{вычисление   среднего балла для каждого студента}
 FOR I: =1 TO N DO
       WITH LIST [I] DO
        BEGIN
            SR: =0;
            FOR J: =l TO 5 DO
                    SR: =SR+OC [J];
            SR: =SR/5
END;



{занесение значений в массив ключей и в массив индексов}
{массив ключей содержит средний балл}
{массив индексов содержит   порядковые номера записей}
       FOR I: =l TO N DO
```

```
          WITH KI DO
           BEGIN
             KEY [I]:=LIST [I].SR;
          END;
{сортировка}
        SORT_IND (K1, N);
{выдача результатов}
       FOR I: =l TO N DO
            WITH LIST [K1.IND [I]] DO
            BEGIN
               WRITE (NOM: 3, ' ‘)
               WRITE (FIO);
               FOR J: = LENGTH (FIO) +1 TO 20 DO WRITE ( ' ' ); {*}
               WRITELN (KURS: 5, GROUP: 5, SR: 7:3)
  END;
END.
```

Строка, помеченная символом «*», содержит цикл для дополнения значений FIO пробелами до 20 символов.

Задание

1. Имеется массив записей с полями артикул, название товара, количество, цена, стоимость. Выполнить расчет стоимости. Определить товар, имеющий максимальную стоимость.
2. Имеется массив записей с полями идентификационный номер, ФИО, разряд, тарифная ставка. Выполнить сортировку записей по разряду.
3. Имеется массив записей с полями индекс, название СМИ, издательство, тираж, цена. Выбрать все СМИ, имеющие заданнуюцену.

Контрольные вопросы

1. Что такое запись в языках программирования?
2. Как объявляется запись?
3. Как выполняется доступ к полям записи?
4. Как выполнить сортировку записей?
5. Как передаются записи в подпрограммы?
6. Что представляет собой вложенная запись?

Практическое занятие №9
Работа с файлами

Пример. Перенести содержимое файла записей, в текстовый файл.

```
type rec = record
                fio : string[20] ;
                kurs,group : integer
           end;
var zap :rec;
      f : file of rec;
      t : text; n1,nl: string;
 begin
     readln(nl) ; assign(f,n1) ; reset(f) ;
     readln(n2); assign(t,n2); rewrite(t)
    while not eof (f) do
     begin
          read(f,zap);
          with zap do
                writeln(f,fio:20,' ',kurs:3,group:5)
     end;
close(f); close(t)
end.
```

Пример. Организовать программный просмотр текстового файла с выдачей содержимого на экран дисплея.

Фрагмент программы:

```
Reset (t);
  while not eof (t) do
  begin
        readln(t,nl);
           writeln (n1)
    end;
  close(t);
```

.

Для добавления записей в текстовый файл он должен быть открыт оператором append(f), после чего записи дописываются оператором write (writeln). Например,

```
append(t);
writeln (t,'____конец файла_____') ;
  close (t)
```

Пример. Имеется файл С, содержащий вещественные числа. Организовать сортировку блоков файла С и их запись поочерёдно в файлы А и В, Для чтения/записи блоками опишем файлы А, В, С как нетипизированные.

Нетипизированный файл С совмещается с ранее созданным файлом типа file of real.

```
const n =500; {число записей для считывания/записи}
type vector = array [l..n] of real;
var A,B,C:file;
    x : vector; {массив для обмена с файлом блоками}
    up : boolean; {переключатель записи в файл А или В}
    1,count:integer;
    na, nb, nc : string;
    begin                                                    .
        1 :=sizeof (real) ; {определяем размер записи 1}
        readln(na); assign(A,na) ; rewrite(A,1);
        readln(nb) ; assign{B,nb); rewrite(B,l) ;
        readln(nc) ; assign (C,nc) ; reset (C,1);
        up:=true;
        blockread_(C,x,n, count);{ читать из файла}
{l* count - количество информации в байтах, фактически считываемое за
одно обращение к файлу}
        while count<> 0 do
      begin
      {сортируем массив X обменами}
       sort_exchange (x, count) ;
      if up then
           blockwrite(A,x,count)    {записать в файл А}
      else
             blockwrite (B,x, count); {записать в файл В}
       up:=not up;                    {изменитьпереключатель}
       blockread(C, x, n, count) ,     {читать из файла}
     end;
     close(A) ; close(B); close(C)
end.
```

Цикл завершает работу, если количество прочитанных из файла записей равно нулю.

Задание

1. В текстовом файле найти самое длинное предложение.
2. Заменить в файле с целыми числами все числа, равные 1, на 0.
3. Из файла удалить все числа, равные 0.
4. В текстовом файле заменить все буквы «а» на «о».

Контрольные вопросы

1. Какие операции необходимо выполнять при работе с файлами?
2. В каких случаях необходимо использовать файлы?
3. Перечислить функции, необходимые для работы с файлами?
4. Как выполнить пользовательскую обработку ошибок при работе с файлами?

8. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Лабораторная работа №1

Линейный информационный процесс

Задание

1. Вариант №1. Рассчитать площадь круга с заданным радиусом.
2. Вариант №2. Рассчитать длину окружности с заданным радиусом.
3. Вариант №3. Найти площадь треугольника по трем сторонам.
4. Вариант №4. В треугольнике с заданными двумя сторонами и углом между ними найти третью сторону.
5. Вариант №5. Для треугольника найти радиус вписанной окружности.
6. Вариант №6. Для треугольника найти радиус описанной окружности.
7. Вариант №7. В прямоугольном треугольнике найти высоту к гипотенузе.

Лабораторная работа №2

Условные операторы. Использование селективного оператора.

Задание

1. Выполнить табулирование функции. Задать область значений x, y, z и шаги табулирования по x, y, z.

Вариант 1

$$F(x, y, z)= \begin{cases} Ln(x/2.5)+y^2/(z-1) & \text{если } 0<=x<7,\ -5<=y<0,\ z=1 \\ \sqrt{x/y}+z & \text{если } x<0,\ y<-5,\ z<1 \\ (x+5.8*y)/(z-3) & \text{иначе} \end{cases}$$

Вариант 2.

$$F(x, y, z)= \begin{cases} \exp(x/z)+y^2*(x-1.5) & \text{если } 1<=x<5,\ -5<=y<1,\ z<1 \\ x/y+\sqrt{z} & \text{если } x<0,\ y<-5,\ 2>z>1 \end{cases}$$

$(1/x-5.8*y)/\sqrt{(z-3)}$ иначе

Вариант3

$$F(x,y,z)=\begin{cases}\sin(x*y)/(z-1) & \text{если } -1<=x<3,\ -5<=y<5,\ z>=3\\ (\overline{x+y})/z & \text{если } x<-2,\ y<-5,\ z<1\\ Ln(z-3/(x-1)) & \text{иначе}\end{cases}$$

Вариант 4

$$F(x,y,z)=\begin{cases}Ln(1/x)+\sin(1/(z-y)) & \text{если } 0<=x<7,\ -5<=y<0,\ z=1\\ \sqrt{z}+1/(x*y) & \text{если } x<0,\ y<-5,\ z<1\\ x/(z-y) & \text{иначе}\end{cases}$$

Вариант 5

$$F(x,y,z)=\begin{cases}1/\underline{x+x/\sin(z}-y) & \text{если } 0<=x,\ -5<=y<0,\ z=1\\ \sqrt{z+4.5*x}+1/(x*y) & \text{если } x<-3.5,\ y<-5,\ z<1\\ x/Ln(z-y) & \text{иначе}\end{cases}$$

Вариант 6

$$F(x,y,z)=\begin{cases}z/\underline{x+(2-y)/s}\sin(z/y) & \text{если } 0<=x,\ -3<=y<0,\ z>1\\ \sqrt{z+4.5*x}+1/(x*y) & \text{если } x<-1,\ y<-5,\ z<5\\ x/Ln(z-y) & \text{иначе}\end{cases}$$

Вариант 7

$$F(x,y,z)=\begin{cases}1/\underline{(x-y)/x/s}\sin(z-y) & \text{если } -2<=x,\ -1<=y<0,\ z<1\\ \sqrt{z-5*x}+1/(x*y) & \text{если } x<0,\ -5<y<-1,\ z>5\\ x/(z-y) & \text{иначе}\end{cases}$$

Вариант 8

$$F(x,y,z)=\begin{cases}\sqrt{1/x+x/\sin(z-y)} & \text{если } 0<=x,\ -3<=y<-1,\ z>1.5\\ z/(4.5-x)+1/(z*x-y) & \text{если } x<-1,\ y<-5,\ z<1\\ (x-1)/\sin(z/y) & \text{иначе}\end{cases}$$

Вариант 9

$$F(x,y,z)=\begin{cases}1/(x-1/y)+x/(z-y) & \text{если } -1<=x,\ -5<=y<1,\ z>3\\ \sqrt{\sin(z+4.5*x*y)} & \text{если } x<-7,\ y<-5,\ z<1\\ Ln(x/(z-y)) & \text{иначе}\end{cases}$$

Вариант 10

$$F(x, y, z)= \begin{cases} Exp(1/x)+x/(1-y) & \text{если } 0<=x,\ 0<=y,\ z>3 \\ \sqrt{z+4.5/x}+1/Ln(x*y) & \text{если } -3<x<-1,\ y<-5,\ z<1 \\ \sqrt{x/(z-y)} & \text{иначе} \end{cases}$$

Лабораторная работа №3

Написание процедур и функций

Задание

Вариант 1.

1.Написать процедуру и функцию для решения представленной ниже задачи, написать фрагмент программы с ее использованием.
Определить является ли число простым. Формальные параметры для процедуры – число и признак, который =1, если число простое и 0 – иначе.

Вариант 2.

1.Написать процедуру и функцию для решения представленной ниже задачи, написать фрагмент программы с ее использованием.
Найти наименьшее общее кратное двух заданных чисел.

Формальные параметры для процедуры – три числа, первое и второе заданные числа, третье – их наименьшее общее кратное.

Вариант 3.

1.Написать процедуру и функцию для решения представленной ниже задачи, написать фрагмент программы с ее использованием.
Заданы три положительных числа. Определяют ли эти числа стороны треугольника и если да, то какого равностороннего, равнобедренного или простого.

Формальные параметры для процедуры – три положительных числа – стороны треугольника и признак, который равен 0 – если числа не определяют треугольник, 1 – простой треугольник, 2 – равнобедренный, 3 – равносторонний.

Вариант 4.

1. Написать процедуру и функцию для решения представленной ниже задачи, написать фрагмент программы с ее использованием.
Определить являются ли два целых числа взаимно простыми. (Два числа называются взаимно простыми, если они не имеют общих делителей).

Формальные параметры для процедуры – два заданных целых числа и признак, который равен 1 – если числа взаимно простые, 0 – иначе.

Вариант 5.

Написать процедуру и функцию для решения представленной ниже задачи, написать фрагмент программы с ее использованием.

Найти наибольший общий делитель двух заданных чисел. Формальные параметры для процедуры – три числа, первое и второе заданные числа, третье – их наибольший общий делитель.

Вариант 6

Написать процедуру и функцию для решения представленной ниже задачи, написать фрагмент программы с ее использованием.

Определить является ли число совершенным. Совершенным называется такое натуральное число, которое равно сумме всех своих делителей, за исключением самого числа, например: 28=1+2+4+4+14.

Формальные параметры для процедуры – заданное для проверки число и признак, который равен 1 если число совершенно, 0 – в противоположном случае.

Вариант 7.

1.Написать процедуру и функцию для решения представленной ниже задачи, написать фрагмент программы с ее использованием.

Решить квадратное уравнение. Формальные для процедуры параметры – три числа, определяющие коэффициенты квадратного уравнения, два – корни этого уравнения.

Вариант 8.

1.Написать процедуру и функцию для решения представленной ниже задачи, написать фрагмент программы с ее использованием.

Заданы четыре положительных числа. Могут ли они определять четырехугольник, квадрат, прямоугольник, ромб, параллелограмм.

Формальные параметры для процедуры – 4 числа, определяющие стороны четырехугольника и признак, принимающий значение 0 – если числа не определяют четырехугольник, 1 – параллелограмм или прямоугольник, 2– ромб или квадрат.

Вариант 9.

1.Написать процедуру и функцию для решения представленной ниже задачи, написать фрагмент программы с ее использованием.

Задать номер года и определить является ли го високосным (високосные года – это те года , которые делятся на 400, и те, у которых номер делится на 4, но не делится на 100).

Формальные параметры для процедуры – № года, признак, равный 1 если год високосный, 0– иначе.

Вариант 10.

1.Написать процедуру и функцию для решения представленной ниже задачи, написать фрагмент программы с ее использованием.

Найти сумму всех простых делителей для заданного числа. Формальные параметры для процедуры – заданное число и сумма простых делителей числа.

Лабораторная работа №4

Обработка одномерного массива

Задание

Вариант 1

1. Найти самую длинную последовательность подряд идущих элементов массива, равных нулю.

2. Выполнить сортировку элементов массива методом прямого выбора. Найти сумму n элементов отсортированного массива, начиная с k-того.

Вариант 2.

1.Найти количество различных чисел в массиве.

2. Выполнить сортировку элементов массива методом прямого выбора. В вызывающей функции найти сумму n наибольших значений элементов отсортированного массива.

Вариант 3.

1.Найти самую длинную последовательность подряд идущих равных между собой элементов массива.

2. Выполнить сортировку элементов массива методом прямого выбора. Найти сумму n наименьших элементов отсортированного массива.

Вариант 4.

1.Найти среднее значение максимального и минимального элементов массива и замены максимальных и минимальных элементов найденным средним.

2. Выполнить сортировку элементов массива методом прямого выбора. Найти количество максимальных элементов.

Вариант 5.

1.Найти количество элементов массива равных максимальному и минимальному элементу и замены этих элементов нулевыми значениями.

2. Выполнить сортировку элементов массива методом прямого выбора. Найти количество минимальных элементов.

Вариант 6.

1.Найти среднее значение всех элементов массива, начиная с первого минимального и до конца массива и замены этим средним всех минимальных элементов массива.

2. Написать функцию для выполнения сортировки элементов массива методом прямого обмена. Выполнить замену всех наименьших значений нулевыми значениями.

Вариант 7.

1.Найти среднее значение всех элементов массива, начиная с первого до первого максимального и замены всех максимальных этим средним значением.

2. Написать функцию для выполнения сортировки элементов массива методом прямого обмена. Выполнить замену всех наибольших значений средним значением максимального и минимального.

Вариант 8.

1.Найти самую длинную последовательность подряд идущих элементов массива, равных какому либо заданному.

2. Выполнить сортировку элементов массива методом прямого обмена. Найти элемент, который расположен в середине массива и определить количество таких элементов в массиве.

Вариант 9.

1.Найти среднее значение элементов массива, расположенных между максимальным и минимальным элементом массива и замены всех нулевых значений на это среднее.

2. Написать функцию для выполнения сортировки элементов массива методом прямого обмена. Найти сумму n элементов отсортированного массива, начиная с k-того.

Вариант 10

1.Найти среднее значение элементов массива, расположенных между первым максимальным и последним минимальным элементом массива, и замены элементов равных 1 найденным средним.

2. Выполнить сортировку элементов массива методом прямого обмена. Найти количество наименьших и наибольших элементов массива.

Лабораторная работа №5

Обработка двумерного массива

Задание

Вариант 1.

1. Удалить из двумерной матрицы все строки, начинающихся 0.
2. Найти среднеарифметическое минимальных элементов столбцов матрицы.

Вариант 2.

1. Преобразования двумерную матрицу в одномерный столбец.
2. Найти произведение максимальных элементов строк матрицы.

Вариант 3.

1. Удалить из двумерного массива всех строки, в которых находятся максимальные элементы.
2. Найти максимум среди минимальных элементов столбцов матрицы.

Вариант 4.

1. Преобразовать одномерный массив в двумерный. (количество строк задается пользователем, при нехватке элементов одномерного массива – дополнить одномерный массив нулями)
2. Найти минимум среди максимальных элементов столбцов матрицы.

Вариант 5.

1. Дополнить двумерный массив строкой, элементы которой максимальные элемента массива.
2. Найти максимум среди минимальных элементов строк матрицы.

Вариант 6.

1. Удалить из двумерного массива всех столбцы, в которых находятся максимальные элементы.
2. Найти минимум среди максимальных элементов строк матрицы.

Вариант 7.

1. Удалить из двумерного массива среднего столбца и средней строки.

2. Найти среднеарифметическое всех максимальных значений столбцов матрицы.

Вариант 8.

1. удалить из двумерного массива все строки с минимальными и максимальными элементами
2. Найти среднеарифметическое всех минимальных значений столбцов матрицы.

Вариант 9.

1. Выполнить сравнение двух двумерных массивов и среди отличных элементов двух матриц найти максимальный.
2. Найти среднеарифметическое всех максимальных значений столбцов и максимальных значений строк матрицы.

Вариант 10.

1. Удалить из двумерного массива все столбцы, в которых находятся минимальные элементы.
2. Найти среднеарифметическое всех максимальных значений столбцов и минимальных значений строк матрицы.

Лабораторная работа №6

Обработка строк

Задание

Вариант 1. З1. Написать программу для удаления из массива строк заданной подстроки.

Вариант 2. З1. Написать программу для подсчета в массиве строк гласных букв.

Вариант 3. З1. Написать программу для подсчета в массиве строк количества слов, разделителем слов считать пробел и ;.

Вариант 4. З1. Написать программу для определения местоположения различных символов массива строк.

Вариант 5. 31. Написать программу для удаления из массива строк одинаковых слов, разделителем слов является пробел.

Вариант 6. 31. Написать программу для определения количества вхождений заданной подстроки в массив строк.

Вариант 7. 31. Написать программу для выделения и вывода на экран предложений из массива строк, ограничитель предложения – точка справа.

Вариант 8. 31. Написать программу для выделения всех слов из массива строк и сортировки этих слов по алфавиту.

Вариант 9. 31. Написать программу для выделения всех слов из массива строк, начинающихся с заданной подстроки.

Вариант 10. 31. Написать программу для выделения всех слов из массива строк, заканчивающихся заданным символом.

Вариант 11. 31. Написать программу для выделения всех слов из массива строк, содержащих заданную подстроку.

Вариант 12. 31. Написать программу для нахождения предложения максимальной длины в массиве строк.

Вариант 13. 31. Написать программу для нахождения длин всех предложений из массива строк.

Лабораторная работа №7

Работа с записями

Задание

Вариант 1. Написать программу для обработки массива записей, содержащей следующие поля:

- название музыкального альбома;
- год выпуска;
- жанр;
- страна-изготовитель;
- цена.

Выдать на экран все записи отсортированные по первому полю в виде таблицы. Массив должен содержать не менее 10 записей. Выполнить инициализацию полей записей.

Вариант 2. Написать программу для обработки массива записей, содержащей следующие поля:

- название газеты или журнала;
- вид СМИ (газета или журнал);
- город, где издается СМИ;
- тираж;
- цена.

Выдать на экран все записи отсортированные по третьему полю в виде таблицы. Массив должен содержать не менее 10 записей. Выполнить инициализацию полей записей.

Вариант 3. Написать программу для обработки массива записей, содержащей следующие поля:

- наименование товара;
- производитель;
- год выпуска;
- партия;
- количество;
- цена.

Выдать на экран все записи отсортированные по второму полю в виде таблицы. Массив должен содержать не менее 10 записей. Выполнить инициализацию полей записей.

Вариант 4. Написать программу для обработки массива записей, содержащей следующие поля:

- название фильма;
- год выпуска;
- киностудия-производитель;
- жанр фильма;
- режиссер.

Выдать на экран все записи отсортированные по третьему полю в виде таблицы. Массив должен содержать не менее 10 записей. Выполнить инициализацию полей записей.

Вариант 5. Написать программу для обработки массива записей, содержащей следующие поля:

- марка автомобиля;
- страна-изготовитель;
- вид двигателя;

– год выпуска;
– цена.

Выдать на экран все записи отсортированные по пятому полю в виде таблицы. Массив должен содержать не менее 10 записей. Выполнить инициализацию полей записей.

Вариант 6. Написать программу для обработки массива записей, содержащей следующие поля:

– название книги;
– автор;
– год издания;
– тираж;
– цена.

Выдать на экран все записи отсортированные по второму полю в виде таблицы. Массив должен содержать не менее 10 записей. Выполнить инициализацию полей записей.

Вариант 7. Написать программу для обработки массива записей, содержащей следующие поля:

– код специальности;
– название специальности;
– название факультета;
– время обучения;
– размер платы за обучение.

Выдать на экран все записи отсортированные по пятому полю в виде таблицы. Массив должен содержать не менее 10 записей. Выполнить инициализацию полей записей.

Вариант 8. Написать программу для обработки массива записей, содержащей следующие поля:

– номер группы;
– специальность;
– факультет;
– плановое количество студентов;
– количество студентов, обучающихся в настоящее время.

Выдать на экран все записи отсортированные по третьему полю в виде таблицы. Массив должен содержать не менее 10 записей. Выполнить инициализацию полей записей.

Вариант 9. Написать программу для обработки массива записей, содержащей следующие поля:

– название телепередачи;
– вид телепередачи;

– автор;
– время показа;
– рейтинг.

Выдать на экран все записи отсортированные по второму полю в виде таблицы. Массив должен содержать не менее 10 записей. Выполнить инициализацию полей записей.

Вариант 10. Написать программу для обработки массива записей, содержащей следующие поля:

– № заявки на учебную дисциплину;
– название дисциплины;
– специальность;
– курс;
– количество часов.

Выдать на экран все записи отсортированные по третьему полю в виде таблицы. Массив должен содержать не менее 10 записей. Выполнить инициализацию полей записей.

Вариант 11. Написать функцию для обработки массива записей, содержащей следующие поля:

– название футбольной команды;
– страна;
– тренер;
– количество проведенных игр;
– количество очков.

Выдать на экран все записи отсортированные по первому полю в виде таблицы. Массив должен содержать не менее 10 записей. Выполнить инициализацию полей записей.

Вариант 12. Написать программу для обработки массива записей, содержащей следующие поля:

– идентификационный номер;
– ФИО;
– Название банка;
– Шифр на зачисление денежных средств;
– сумма.

Выдать на экран все записи отсортированные по второму полю в виде таблицы. Массив должен содержать не менее 10 записей. Выполнить инициализацию полей записей.

Вариант 13. Написать программу для обработки массива записей, содержащей следующие поля:

– ФИО;

– серия паспорта;
– номер паспорта;
– кем выдан ;
– когда выдан.

Выдать на экран все записи отсортированные по четвертому полю в виде таблицы. Массив должен содержать не менее 10 записей. Выполнить инициализацию полей записей.

Лабораторная работа №8

Обработка файлов

Задание

Вариант №1

1. В файле целых чисел заменить все нулевые значения максимальным значением.
2. В текстовом файле посчитать количество заданных букв.

Вариант №2

1. В файле целых чисел среднеарифметическое.
2. В текстовом файле посчитать количество.

Вариант №3

3. В файле целых чисел заменить все нулевые значения среднеарифметическим всех значений из файла.
4. В текстовом файле найти самое длинное предложение.

Вариант №4

1. В файле целых чисел найти максимальное значение.
2. В текстовом файле посчитать количество предложений.

Вариант №5

1. В файле целых чисел найти сумму положительных и отрицательных значений.
2. В текстовом файле посчитать количество вхождений подстроки.

Вариант №6

1. В файле целых чисел заменить все значения равные максимальному нулевыми значениями.
2. Из текстовом файле удалить заданную подстроку.

Вариант №7

1. В файле целых чисел посчитать количество нулевых значения.
2. В текстовом файле посчитать количество предложений.

Вариант №8

1. В файле целых чисел заменить все нулевые значения максимальным значением.
2. В текстовом файле посчитать количество заданных букв.

Вариант №9

1. Из файла целых чисел удалить все нулевые значения.
2. В текстовом файле посчитать процентное отношение гласных и согласных букв.

Вариант №10

3. В файле целых чисел найти сумму четных и нечетных чисел.
4. В текстовом файле удалить все гласные буквы.

9. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОРГАНИЗАЦИИ МЕЖСЕССИОННОГО КОНТРОЛЯ ЗНАНИЙ СТУДЕНТОВ

1. Межсессионная аттестация студентов проводится дважды в семестр на 7 и 13 неделях 1-го семестра семестра.

2. Аттестационная оценка складывается из оценок, полученных за контрольные работы по результатам промежуточного тестирования.

3. Организация аттестации студентов, проводится в соответствии с положением АмГУ о курсовых экзаменах и зачетах.

10. ТЕСТОВЫЕ ЗАДАНИЯ ДЛЯ ОЦЕНКИ КАЧЕСТВА ЗНАНИЙ ПО ДИСЦИПЛИНЕ

Вариант №1

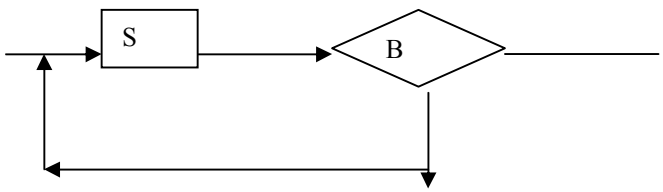
1. Результат сравнения двух строк “петров” и ”Петров
   a) строки равны
   b) строка 1 больше строки 2
   c) строка 2 больше строки 1

2. Имеется функция вычисления кубического корня числа
   ```
   Function Cub(a:real):real;
     Begin
       Cub:=a*a*a;
     End;
   ```
   Какие из инструкций использования этой функции неверны?
   a) x:=cub(y)+0.5;
   b) cub(7.2);
   c) Writeln(x, cub(x));

3. Что не является критерием качества алгоритма?
   a) правильность;
   b) прозрачность;
   c) дискретность;
   d) эффективность.

4. К какому типу языков относится алгоритмический язык Паскаль?
   a) непроцедурный;
   b) функциональный;
   c) операторного типа;

5. Какой тип данных не относится к стандартным?
   a) целый;
   b) перечислимый
   c) вещественный;
   d) символьный;
   e) логический.

6. 

   a) блок-схема while B do S;
   b) блок-схема repeat S until B;

c) блок-схема if (B) then S;

7. Конечное множество элементов состоящее из компонентов разного типа и представляющих логически единое целое
   a) строки;
   b) массив;
   c) запись.

8. Какой процедурой необходимо открыть файл для чтения и записи
   a) rewrite (f);
   b) reset(f);
   c) append(f);

9. Какой из фрагментов демонстрирует использование интервального типа для объявления массивов
   a) type index=1..100;
      var tabl: array[index] of integer;
   b) const n=1;
      const m=100;
      var tabl: array[n..m] of integer;
   c) var tabl: array[1..100] of integer;

10. Инструкция With используется в связи с
    a) циклическими конструкциями
    b) инструкцией case
    c) записями.

Вариант №2

1. Какой тип имеет значение, возвращаемое функцией EOF
   a) boolean;
   b) string;
   c) integer.

2. Если объект описан в некоторой подпрограмме, то область его видимости является
   a) главная программа и все подпрограммы
   b) подпрограмма и вложенные в нее процедуры и функции;
   c) вложенные в подпрограмму процедуры и функции.

3. Какие различают способы подстановки параметров в процедуры 1) подстановка значения; 2) подстановка переменной; 3) отсутствие параметров
   a) 1,2
   b) 1,2,3
   c) 2,3

4. Логическое отрицание not относится к
   a) аддитивным операциям;
   b) мультипликативным;
   c) унарным;
   d) операциям отношений.

5. Определить результат выполнения инструкций
   St:=’студент Иванов’;
   F=copy(st, 9, 6);
   a) ‘Иванов’
   b) ‘нт Иванов’
   c) ‘студен’

6. Программа и процедура имеет вид
```
Var a,b,c: real;
Procedure S(x,y,z:real);
  Begin
      Z:=x+y;
   End;
Begin
    a:=7;
    b:=9;
    c:=1;
    S(a,b,c);
    Writeln(c);
End;
```
   Чему равно с по окончанию работы программы?
   a) 16
   b) 0
   c) 1

7. Если файл открыт с помощью процедуры append(f), то он должен быть объявлен следующим способом
   a) var f: file of char;
   b) var f: file of string;
   c) var f: text;

8. <имя типа> = set of <базовый тип>

a) синтаксис определения пользовательского типа
b) синтаксис определения типа множества
c) синтаксис определения перечислимого типа

9. Какой блок описаний на языке Паскаль приведен не на своем месте?
   Label <описание меток>
   Type <описание типов>
   Const <описание констант>
   Var < описание переменных>
   <описание процедур и функций>
   a) label, type
   b) type, const
   c) const, var

10. Какие действия не используются для описания алгоритмов решения любой задачи?
    a) присваивание
    b) рекурсия
    c) разветвление
    d) цикл
    e) составное действие

Вариант №3

1. Ошибки связанные с недостатком знаний или пониманием программистом языка программирования относятся к
   a) семантические
   b) синтаксические

2. Описание идентификатора с помощью формулы
   <идентификатор>::=<буква>|<идентификатор><буква>|<идентификатор><цифра>
   a) синтаксическая диаграмма
   b) нормальная форма Бэкуса
   c) продукция

3. Для организации ветвлений в программе предназначены
   a) Циклические операторы
   b) Условные операторы
   c) Операторы перехода

4. Какая из процедур предназначена для вставки подстроки в строку

a) Val
b) Delete
c) Insert

5. Функция вычисления скорости свободного падения тела имеет вид
```
Function  VP(t:real): integer;
 Begin
    Vp:= 10*t;
  End;
```
Какая ошибка допущена при написании функции?
   a) В конце функции отсутствует инструкция присваивания значения функции
   b) Неверно указан тип функции
   c) Отсутствует блок Var в функции

6. Назначение процедуры Assign
   a) открытие файла
   b) объявление файла
   c) связывание файловой переменной с именем файла

7. Какое из описаний является правильным описанием массива?
   a) var a:array [1..n] of real;
   b) const n100=100;
      type r100:array [1..n100] of real;
       var a:r100;
   c) const n=100;
       var a=array [1..n] of real;

8. Тип, определяемый следующими инструкциями
```
Type Day=(Mon, Tue, Wed, Thu, Fri, Sat, Sun);
 Var today: day;
```
   a) множество;
   b) перечисляемый;
   c) интервальный

9. «Последним пришел – первым вышел» - принцип работы с
   a) стеком
   b) записями
   c) интервальным типом

10. Какой из операторов выполняет одно из перечисленных в программе действий в зависимости от значения выражения
   a) условный оператор
   b) операторы цикла
   c) селективный оператор

11. БИЛЕТЫ ДЛЯ ПРОВЕДЕНИЯ ЭКЗАМЕНА

Экзаменационный билет №1

1. Общий вид структуры программы на языке Паскаль.
2. Написать программу для решения следующей задачи. Найти сумму всех простых чисел в интервале [0, 1000].

Экзаменационный билет №2

1. Понятие жизненного цикла программного изделия.
2. Написать программу для решения следующей задачи. Выполнить сортировку по методу прямого выбора одномерного массива A(100).

Экзаменационный билет №3

1. Основные определения программирования.
2. Написать программу для решения следующей задачи. Выполнить бинарный поиск в отсортировнном одномерном массиве A(100).

Экзаменационный билет №4

1. Последовательные файлы. Логическая организация и представление в памяти.
2. Написать программу для решения следующей задачи. Выполнить сортировку одномерного массива A(100) методом прямого обмена.

Экзаменационный билет №5

1. Понятие файла и основные процедуры и функции для работы с файлами.
2. Написать программу для решения следующей задачи. Определить количество элементов равных максимуму в двумерном массиве A(10X10).

Экзаменационный билет №6

1. Операторы ввода-вывода в Паскале.
2. В массиве строк выявить строку, имеющую максимальное количество заданных символов. Из всех строк удалить все заданные символы.

Экзаменационный билет №7

1. Массив как структура данных. Логическое и физическое представление.
2. Написать процедуру для решения следующей задачи. Найти максимум в одномерном массиве. Использовать эту процедуру для нахождения максимального значения в массивах A(10), B(10), C(10).

Экзаменационный билет №8

1. Процедуры и функции.
2. Написать программу для решения следующей задачи. В массиве строк удалить все подстроки ‘def’.

Экзаменационный билет №9

1. Операторы цикла. Виды. Синтаксические конструкции. Использование.
2. Написать программу для решения следующей задачи. Заменить все максимальных и минимальных элементы двумерного массива нулями.

Экзаменационный билет №10

1. Строки. Логическое и физическое представление.
2. Написать функцию для решения следующей задачи. Нахождение среднеарифметического всех четных, положительных элементов двумерного массива. Использовать функцию для обработки массивов A(5X5), B(5X5), C(5X5).

Экзаменационный билет №11

1. Записи. Логическое и физическое представление.
2. Написать программу для решения следующей задачи. Каждый элемент в столбце двумерного массива разделить на максимальный элемент в этом столбце.

Экзаменационный билет №12

1. Понятие файла и основные процедуры и функции для работы с файлами.
2. Написать процедуру для решения следующей задачи. Найти сумму минимальных значений каждой строки двумерного массива. Использовать процедуру для обработки трех двумерных массивов.

Экзаменационный билет №13

1. Данные. Стандартные типы данных.
2. Написать программу для решения следующей задачи. Дан файл целых чисел ‘dan.dat’. найти среднеарифметическое всех чисел файла.

Экзаменационный билет №14

1. Пользовательский тип данных.
2. Написать программу для решения следующей задачи. Записи содержат сведения о студентах и состоят из полей: ФИО, номер группы, средний балл за каждый год учебы. Вычислить средний балл за все время учебы.

Экзаменационный билет №15

1. Оператор присваивания и составной оператор.
2. Написать программу для решения следующей задачи. В файле вещественных чисел ‘dan.dat’ заменить все отрицательные на 0.

Экзаменационный билет №16

1. Обзор языков программирования. Сравнение языков программирования.
2. Написать процедуру для решения следующей задачи. Найти среднеарифметическое минимальных значений строк двумерного массива. Использовать процедуру для обработки трех двумерных массивов.

Экзаменационный билет №17

1. Записи. Работа с записями.
2. Написать программу для решения следующей задачи. В массиве строк выявить строки, содержащие подстроку ‘def’. Сформировать из этих строк новый массив.

Экзаменационный билет №18

1. Множество. Описание. Операции над множествами.
2. Написать процедуру для решения следующей задачи. Определение значения $S=\Sigma \cos^{1}(x)$ i=1…m. Использовать процедуру для нахождения S для m=3, 10, 25.

Экзаменационный билет №19

1. Этапы программирования.
2. Написать функцию для решения следующей задачи. Нахождение суммы элементов двумерной матрицы, расположенных выше главной диагонали. Использовать ее для обработки трех двумерных матриц.

Экзаменационный билет №20

1. Условный оператор. Селективный оператор.
2. Написать процедуру для решения следующей задачи. Печать на экране определенного количества заданных символов. Использовать процедуру для вывода на экран фигуры *****

* *
*****

Экзаменационный билет №21

1. Одномерный массив данных.
2. Написать функцию для решения следующей задачи. Вычислить $S=\Sigma\ 1/i,\ i=1,2,\ldots m$. Использовать функцию для вычисления суммы для m=5, 7, 15.

Экзаменационный билет №22

1. Двумерный массив данных.
2. Написать программу для решения следующей задачи. Даны два непустых отсортированных файла целых чисел. Выполнить слияние этих файлов в один выходной отсортированный.

Экзаменационный билет №23

1. Строки. Процедуры и функции для работы со строками.
2. Написать программу для решения следующей задачи. Протабулировать функцию f=(x+y)/x*y x Є [1,10], y Є[2 ,5], hx=0.5,hy=0.1.

Экзаменационный билет №24

1. Операторы циклов WHILE и REPEAT UNTIL. Общее и различия.
2. Написать программу для решения следующей задачи. Напечатать таблицу умножения.

Экзаменационный билет №25

1. Типизированные файлы.
2. Написать программу для решения следующей задачи. В одномерном массиве определить номер первого элемента, большего M и сумму элементов ему предшествующих.

12. КАРТА КАДРОВОЙ ОБЕСПЕЧЕННОСТИ ДИСЦИПЛИНЫ

Лектор – старший преподаватель Соловцова Любовь Александровна

Руководитель практических работ – старший преподаватель Соловцова Любовь Александровна

Руководитель лабораторных работ – старший преподаватель Соловцова Любовь Александровна