Федеральное государственное образовательное бюджетное
учреждение высшего образования
«Финансовый университет при Правительстве Российской Федерации»
(Финуниверситет)

Самарский финансово-экономический колледж
(Самарский филиал Финуниверситета)

УТВЕРЖДАЮ
Заместитель директора по учебно-методической работе
_____Л.А Косенкова
«_21_» _февраля_ 20_22_ г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОРГАНИЗАЦИИ И ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ ЗАНЯТИЙ ПО ДИСЦИПЛИНЕ «ОП.04 ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ»

СПЕЦИАЛЬНОСТЬ: 09.02.07 ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ПРОГРАММИРОВАНИЕ

Самара – 2022

Методические указания по организации и выполнению практических занятий разработаны на основе рабочей программы по дисциплине «Основы алгоритмизации и программирования» и в соответствии с федеральным государственным образовательным стандартом среднего профессионального образования по специальности 09.02.07 Информационные системы и программирование, утвержденного приказом Министерства образования науки Российской Федерации от 09.12.2016 года № 1547, с учетом Профессионального стандарта, утвержденного приказом Министерства труда и социальной защиты Российской Федерации от 11 февраля 2014 г. № 647н «Об утверждении профессионального стандарта 06.011 Администратор баз данных» (зарегистрирован Министерством юстиции Российской Федерации 24 ноября 2014 г., регистрационный № 34846)
Присваиваемая квалификация: администратор баз данных

Разработчики:

Платковская Е.А. _____ Преподаватель Самарского филиала Финуниверситета

Методические указания по организации и выполнению практических занятий рассмотрены и рекомендованы к утверждению на заседании предметной (цикловой) комиссии естественно-математических дисциплин

Протокол от «24» января 2022 г. № 5

Председатель ПЦК _____ М.В. Писцова

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Методические указания по выполнению практических занятий по предмету ОП.04 Основы алгоритмизации и программирования разработаны с целью оказания помощи студентам специальности 09.02.07 Информационные системы и программирование и преподавателям по организации практических занятий по изучаемой дисциплине, в соответствии с требованиями федерального государственного стандарта среднего профессионального образования.

Методические разработка включает в себя краткие теоретические сведения, указания по выполнению практических работ, контрольные вопросы, формы контроля.

В соответствии с учебным планом на практические занятия для студентов отводится **36** часов.

Учебная дисциплина «ОП.04 Основы алгоритмизации и программирования» относится к обязательной части общепрофессионального цикла.

Учебная дисциплина «ОП.04 Основы алгоритмизации и программирования» обеспечивает формирование профессиональных и общих компетенций по всем видам деятельности ФГОС по специальности 09.02.07 Информационные системы и программирование.

Методические указания направлены на формирование и развитие у студентов общих и профессиональных компетенций:

ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.

ОК 02. Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.

ОК 04. Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.

ОК 09. Использовать информационные технологии в профессиональной деятельности.

ОК 10. Пользоваться профессиональной документацией на государственном и иностранном языке.

ПК 1.1. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.

ПК 1.2. Разрабатывать программные модули в соответствии с техническим заданием.

ПК 1.3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК 1.4. Выполнять тестирование программных модулей.

ПК 1.5. Осуществлять рефакторинг и оптимизацию программного кода.

ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.

ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования.

ПК.4.1 Осуществлять инсталляцию, настройку и обслуживание программного обеспечения компьютерных систем.

ПК.4.2 Осуществлять измерения эксплуатационных характеристик программного обеспечения компьютерных систем.

ПК 7.1. Выявлять технические проблемы, возникающие в процессе эксплуатации баз данных и серверов.

ПК 7.2. Осуществлять администрирование отдельных компонент серверов.

ПК 7.3. Формировать требования к конфигурации локальных компьютерных сетей и серверного оборудования, необходимые для работы баз данных и серверов.

ПК 7.4. Осуществлять администрирование баз данных в рамках своей компетенции.

ПК 7.5. Проводить аудит систем безопасности баз данных и серверов с использованием регламентов по защите информации.

В результате освоения учебной дисциплины обучающийся **должен**

иметь практический опыт: посторенние алгоритмов для конкретных программ; работать в языках программирования; выполнять проверку, отладку программного кода, оформлять программный код в соответствии со стандартом.

уметь:

– разрабатывать алгоритмы для конкретных задач;
– использовать программы для графического отображения алгоритмов;
– определять сложность работы алгоритмов;
– работать в среде программирования;
– реализовывать построенные алгоритмы в виде программ на конкретном языке программирования;
– оформлять код программы в соответствии со стандартом кодирования.

знать:

– понятие алгоритмизации, свойства алгоритмов, общие принципы построения алгоритмов, основные алгоритмические конструкции;
– эволюцию языков программирования, их классификацию, понятие системы программирования;
– основные элементы языка, структуру программы, операторы и операции, управляющие структуры, структуры данных, файлы, классы памяти;
– подпрограммы, составление библиотек подпрограмм;
– объектно-ориентированную модель программирования, основные принципы объектно-ориентированного программирования на примере алгоритмического языка: понятие классов и объектов, их свойств и методов, инкапсуляция и полиморфизма, наследования и переопределения;
– выполнять проверку, отладку кода программы.

Характерная черта практических занятий – индивидуальное выполнение заданий, самостоятельное приобретение знаний. В связи с этим предусмотрены работы по всем основным разделам курса. Перед выполнением практической работы обучающийся получает опережающее теоретическое домашнее задание. На занятии объясняются вопросы, уточняются определения, которые помогают выполнению заданий. Обучающийся может просмотреть запись объяснения любой примерной работы по всем темам. И только после этого обучающийся приступает к выполнению практической работы.

При выполнении работы обучающийся должен самостоятельно изучить методические рекомендации по проведению практической работы, подготовить ответы на контрольные вопросы. Все практические задания выполняются за компьютером, теоретические вопросы сдаются устно или письменно.

После выполнения работы обучающийся должен представить отчет о проделанной работе с полученными результатами и в устной форме защитить.

При отсутствии по неуважительной причине обучающийся выполняет работу самостоятельно во внеурочное время и защищает на консультации по расписанию.

Структура практических работ:

1. Тема.
2. Цель.
3. Теоретическое обоснование.
4. Ход работы.
5. Контрольные вопросы.
6. Содержание отчета.

При изучении дисциплины необходимо постоянно обращать внимание студентов на ее прикладной характер, показывать, где и когда изучаемые теоретические положения, и практические навыки могут быть использованы в будущей профессиональной деятельности.

Объем учебной дисциплины и виды учебной работы

Вид учебной работы	**Объем в часах**
Объем образовательной программы учебной дисциплины	180
Объем работы обучающихся во взаимодействии с преподавателем	134
в том числе:	
теоретическое обучение	78
практические занятия	36
курсовое проектирование	20
самостоятельная работа	34
Промежуточная аттестация в форме экзамена	12
В т.ч. консультации	2
экзамен	10

ПЕРЕЧЕНЬ ПРАКТИЧЕСКИХ РАБОТ

Практическая работа №1. Знакомство со средой программирования.
Практическая работа №2. Составление программ линейной структуры.
Практическая работа №3. Составление программ циклической структуры Обработка одномерных массивов.
Практическая работа №4. Обработка двумерных массивов.
Практическая работа №5. Работа со строками.
Практическая работа №6. Работа с данными типа множество.
Практическая работа №7. Файлы последовательного доступа.
Практическая работа №8. Типизированные файлы.
Практическая работа №9. Применение рекурсивных функций.
Практическая работа №10. Программирование модуля.
Практическая работа №11. Использование указателей для организации связанных списков.
Практическая работа №12. Создание проекта с использованием компонентов для работы с текстом.
Практическая работа №13. Создание процедур на основе событий.
Практическая работа №14. Разработка функциональной схемы работы приложения.
Практическая работа №15. Разработка оконного приложения с несколькими формами
Практическая работа №16. Тестирование, отладка приложения.
Практическая работа №17. Классы ООП: виды, назначение, свойства, методы, события. Объявления класса.
Практическая работа №18. Создание наследованного класса.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ СТУДЕНТОВ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

Практическая работа №1. Знакомство со средой программирования.
Цель занятия:

- научиться применять основные операторы и редактировать программы с линейной структурой;
- сформировать навыки работы в среде программирования;

Оснащение:

- ПК;
- Программное обеспечение: Pascal ABC.

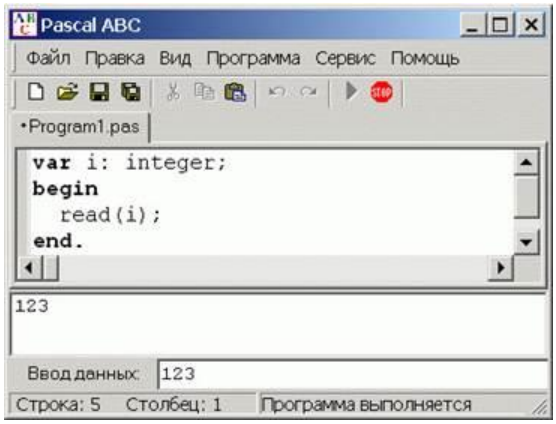
Краткие теоретические сведения

Основными операторами являются:

· Read, Readln – оператор ввода данных;
· : = – оператор присваивания;
· Write, Writeln - оператор вывода.

Program - зарезервированное слово для обозначения заголовка программы.
Var - зарезервированное слово для обозначения раздела переменных.
Begin и End - служебные слова, заключающие между собой раздел операторов.

Структура программы с линейной структурой



Окно программы Pascal ABC состоит из окна редактора кода, окна ввода и окна вывода.

Окно редактора кода

Большую часть рабочей области, её верхнюю часть занимает окно редактора кода. В него вводится исходный текст программы.

Горячие клавиши, которые можно использовать при работе с текстом программы:

F2, Ctrl-S- сохранить файл.

F3, Ctrl-O- загрузить файл.

F12- сохранить файл под новым именем.

Ctrl-Shift-S- сохранить все открытые файлы.

Ctrl-Tab, Ctrl-Shift-Tab – перейти к следующему/предыдущему окну редактора.

Ctrl-Shift-I- увеличить отступ выделенного блока.

Ctrl-Shift-U – уменьшить отступ выделенного блока.

Под окном редактора расположено окно вывода. Оно предназначено для вывода данных процедурами write и writeln, а также для вывода сообщений об ошибках и предупреждений во время работы программы.

Окно вывода может быть скрыто. Клавиша F5 показывают/скрывают окно вывода. Для скрытия окна вывода используется также клавиша Esc.

Окно вывода обязательно открывается при любом выводе в него.

Для очистки окна вывода следует нажать комбинацию клавиш Ctrl-Del или кнопку  .

Окно ввода открывается при выполнении процедур read и readln в ходе работы программы.

Ввод данных в окно ввода сопровождается эхо-выводом в окно вывода. После нажатия клавиши Enter данные из окна ввода попадают в соответствующие переменные, окно ввода закрывается, и программа продолжает работать дальше.

Запуск и остановка программы

Для запуска программы в текущем окне редактора следует нажать клавишу F9 или кнопку  панели инструментов.

Программа вначале компилируется во внутреннее представление, после чего, если не найдены ошибки, программа начинает выполняться. При выполнении программы кнопка запуска программы становится неактивной, кнопка останова программы, наоборот, активной и в строке статуса отображается информация “Программа выполняется”.

Выполнение программы можно в любой момент прервать нажатием комбинации клавиш Ctrl-F2 или кнопки  При этом в окне вывода появится сообщение: • Программа прервана пользователем.

Практические задания

Содержание отчета по результатам выполнения практической работы

Отчет должен содержать:

1. Название работы.
2. Цель работы.
3. Результат выполнения задания № 1 (записать программу на компьютер и в тетрадь).
4. Результат выполнения задания № 2 (записать программу на компьютер и в тетрадь)
5. Результат выполнения задания № 3 (записать программу на компьютер и в тетрадь)
6. Результат выполнения задания № 4 (записать программу на компьютер и в тетрадь)
7. Результаты выполнения задания № 5 (записать программу на компьютер и в тетрадь)
8. Результаты выполнения задания № 6 (ответить письменно на вопросы)
9. Вывод по работе (результат выполнения № 7).

Задание 1. Наберите текст программы, описывающей решение примера C=A+B. Выполните компиляцию программы и тестирование.

1. На рабочем столе запустите ярлык Pascal ABC.
2. В окне текстового редактора программы Pascal ABC наберите текст программы:

```
Program Z1;
Var A, B, C: integer;
Begin
Writeln ('Задача 1');
Writeln ('Введите число A');
Readln (A);
Writeln ('Введите число B');
C:=A+B;
Writeln ('c=' , c );
Readln ;
End .
```

3. Проверьте программу на ошибки (компиляция) - нажать *Программа/Компилировать* (или Ctrl+F9).

4. Протестируйте программу:

1) запустить программу, выбрав в меню пункт *Программа/Выполнить (или F9).*
2) в ответ на приглашение *«Введите число А»* набрать *23*,
3) в ответ на приглашение *«Введите число В»* набрать *17*,
4) если программа работает правильно, то в результате будет выведено – С=40

5. Самостоятельно протестируйте программу с другими числовыми данными:
A=2; B= -12;
A = -10; B= -17;
A = 0; B= - 20;
Запишите результаты в тетрадь.

Задание 2. Измените исходную программу, чтобы она находила **частное** двух чисел.
Задание 3. Измените исходную программу, чтобы она находила сумму четырех чисел.

Задание 4. Наберите текст программы, описывающей решение примера $D = \frac{6A + 2B}{4A}$

Выполните компиляцию программы и тестирование.

1. В окне текстового редактора программы Pascal наберите текст программы:

```
Program Z2;
Var A, B, D : real;
Begin
Writeln ('Задача 2');
Writeln ('Введите A');
Readln (A);
Writeln ('Введите B');
Readln ( B );
D:= (6*A+2*B)/ (4*A);
 Writeln ('D= ', D );
Readln ;
End.
```

2. Проверить программу на ошибки - нажать *Программа/Компилировать* (или Ctrl+F9).

3. Протестировать программу:

1) запустить программу, выбрав в меню пункт *Программа/Выполнить (или F9)*
2) в ответ на приглашение *«Введите число A»* набрать *2,*
3) в ответ на приглашение *«Введите число B»* набрать *2,*
4) если программа работает правильно, то в результате будет выведено – *E=2.0000000000E+0000.*

4. Введите ограничение для вещественного числа, для этого внесите изменения в текст программы Writeln ('D = ', D:4:2);

5. Самостоятельно протестируйте программу с другими числовыми данными:

A=2; B=1,5;
A = 3; B= -7;
A = 0; B= - 2;

Запишите результаты в тетрадь.

Задание 5. Создайте и протестируйте программу вычисления переменной **D=(a*b)+2**

Контрольные вопросы:

1. Как открыть новое окно программы?
2. Как сохранить новый текст программы?
3. Как «запустить» программу?
4. Назовите оператор вывода в Pascal.
5. Назовите оператор ввода в Pascal.
6. Какие типы данных применимы в этой практической работе.

Сделайте вывод о проделанной практической работе.

Практическая работа №2. Составление программ линейной структуры.
Цель:
– освоить методы разработки алгоритмов и программирования задач линейной структуры.

Краткие теоретические сведения

Алгоритм линейной структуры – это алгоритм, в котором все действия выполняются последовательно друг за другом и только одинраз. Блок-схема алгоритма представляет собой последовательностьблоков, которые располагаются сверху вниз в порядке их выполнения. Все промежуточные или исходные данные влияния на направление процесса выполнения не оказывают.

Пример. Вычислить высоты треугольника со сторонами a, b, c, используя формулы:

$$h = \frac{2}{a}\sqrt{p(p-a)(p-b)(p-c)}$$

$$h = \frac{2}{b}\sqrt{p(p-a)(p-b)(p-c)}$$

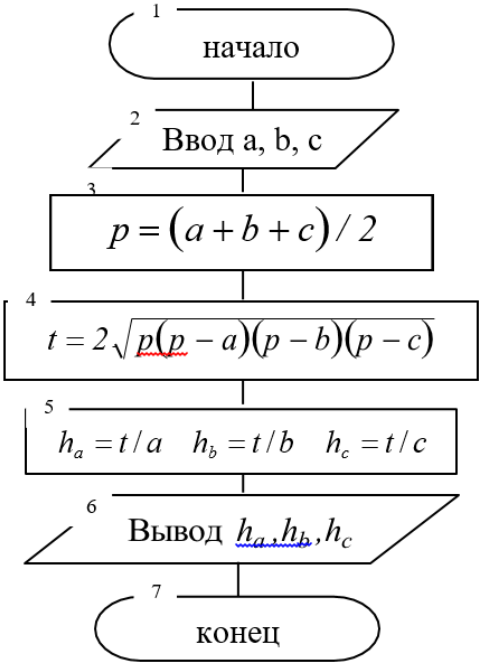
$$h = \frac{2}{c}\sqrt{p(p-a)(p-b)(p-c)}$$

При решении данной задачи для исключения повторений следует вычислять высоты не по приведенным выше формулам непосредственно, а используя промежуточную переменную:

$$t = 2\sqrt{p(p-a)(p-b)(p-c)}$$

тогда $h_a = t/a,\ h_b = t/b,\ h_c = t/c$

Схема алгоритма решения задачи имеет следующий вид:



Для записи программы линейной структуры необходимы операторы присваивания, ввода исходных данных и вывода результатов вычислений.

Программа на языке Pascal состоит из заголовка, раздела описаний и раздела операторов:

Program имя;

раздел описаний begin

раздел операторов

end.

Заголовок программы начинается со служебного слова Program, после которого указывается имя программы. Раздел описаний предназначен для объявления в программе всех данных, которые встречаются, а также их характеристик.

Существует определенный порядок в разделе описаний:

- раздел меток Label;
- раздел констант Const;
- раздел типов Type;
- раздел переменных Var;
- раздел процедур и функций (Function и Procedure).

Раздел операторов заключается в операторные скобки вида: Begin (начать) и End (закончить), при этом после End ставится точка. Точка – это признак окончания программы. В разделе операторов записывается последовательность выполняемых операторов. Операторы отделяются друг от друга символом ";"

Оператор присваивания – это основной оператор любого языка программирования. Он предназначен для замены текущего значения переменной новым значением.

Форма оператора присваивания имеет вид:

Переменная : = выражение;

При выполнении этого оператора значение выраженияvычисляется и присваивается переменной.

Например,

a := b+c;

w := sin(sqr(t))/(s + ln(v));s := „строка“;

Имя переменной и результат выражения должны принадлежать к одному типу.

Для ввода данных используются операторы:

read (элемент 1, элемент 2, ...);

readln (элемент 1, элемент 2, ...);

При выполнении оператора read (элемент1, элемент2, ...) происходят следующие действия: программа приостанавливает свою работу и ждет, пока на клавиатуре будут набраны данные и нажата клавиша Enter. После нажатия клавиши Enter, введенные значения присваиваются переменным, имена которых указаны в операторе read. Числовые значения должны быть набраны в одной строке иразделены пробелами.

Оператор readln (элемент 1, элемент 2, ...) осуществляет ввод данных, а затем обеспечивает переход к началу новой строки.

Для вывода данных используются операторы:

write (элемент 1, элемент 2, ...);

writeln (элемент 1, элемент 2, ...);

где элемент – это переменная или строка символов, заключеннаяв апострофы.

Например, оператор

write („ Значение B= „, B);

выводит на экран дисплея текст

Значение B=

а затем числовое значение переменной B.

Для вывода целых и действительных чисел можно указыватьформаты в операторе WRITE. Формат указывается через двоеточие после переменной. Для действительных чисел формат состоит из двух величин. Первая величина указывает на общее количество позиций для вывода числового значения переменной: знак числа, количество цифр в целой части, точку и количество цифр в дробной части, второе – количество позиций для вывода дробной части. Например, оператор WRITE (Y:5:2) осуществит вывод значения Y на экран и при этом для отображения значения Y будет отведено пять позиций, из них две – на дробную часть.

Для вывода целых чисел количество позиций для дробной части не указывается. Например, если необходимо вывести значение целого

числа N=125, то оператор вывода будет иметь вид: WRITE ('N=', N:3),т.е. для вывода числового значения N отведено 3 позиции.

Допускается использование оператора вывода writeln (элемент1, элемент2 ...), который сначала выводит значения переменных, а затемосуществляет переход на новую строку.

Оператор writeln обеспечивает пропуск одной строки и переход к началу новой строки.

Текст программы решения задачи, схема которой приведена в примере имеет вид:

```
Program rabota;
Var a,b,c,p,t,ha,hb,hc:real;Begin
Writeln(„vvod a,b,c“);
Readln(a,b,c);
p:=(a+b+c)/2;
t:=2*sqrt(p*(p-a)*(p-b)*(p-c));ha:=t/a;
hb:=t/b;hc:=t/c;
writeln („ha=“,ha:6:2,“ hb=“,hb:6:2, „ hc=“, hc:6:2);end.
```

Исходными данными для решения задачи являются значения длин сторон треугольника: a, b, c. Для ввода этих значений используется оператор Readln. В программе

используется переменнаяр для вычисления полупериметра и вспомогательная переменная t дляисключения повторений.

Вычисленные значения высот ha, hb, hc выводятся с соответствующими именами переменных. Когда выводятся элементы списка вывода, пробелы между ними не устанавливаются автоматически, поэтому необходимо непосредственно их указать в списке вывода. Для улучшения внешнего вида данных, которые выводятся на экран дисплея, используется форматированный вывод.

Практические задания

Задание 1.

1. Изучить лекционный материал и методические рекомендациидля выполнения задания.
2. Разработать алгоритм для расчета заданных переменных.
3. Составить программу на алгоритмическом языке Pascal.
4. Выполнить отладку и тестирование программы накомпьютере.
5. Подготовить ответы на контрольные вопросы.

Задание 2. Составить блок-схему и программу вычисления значений функции при заданных значениях аргумента.

1) $P = \frac{x \sin x}{(2x - y)^2} - \frac{\sqrt{\lvert 2x - y \rvert}}{3y \cos y}$	2) $P = \frac{\sqrt{a^2 + b^2 + c^2}}{2a + b \sin c} - \frac{\ln(a + 2bc)}{a^2 + b^2 + c^2}$
3) $Z = \frac{x^3 + y^2}{y \sin x} - \frac{\ln(y + 5e^{y+x})}{\sqrt[4]{\lvert x^3 + y^2 \rvert}}$	4) $P = \frac{e^{2a} + 2{,}5 Sin3b}{\sqrt{a^2 + b^2}} - 2tg(a^2 + b^2)$
5) $Y = \sqrt{a^2 + x^2} + \frac{tg(a + x) - \ln(a + x)}{\sin x + \cos^2 a}$	6) $Z = \frac{Sin^2(x + y)}{(x + y)^2} - \frac{\sqrt{\lvert 3x - 4y \rvert}}{\cos x + e^{5x}}$
7) $Y = \cos x - \frac{\ln x + \sin^2(5x + 2)}{5{,}3x + \frac{7}{\sqrt{2x^2 + 20}}}$	8) $P = \frac{\ln a + \ln b}{x + \frac{ax + b}{\sqrt{x^2 + 2b^4}}} - tg(ax + b)$
9) $Y = \frac{\sqrt{\lvert 5x - 45 \rvert}}{tgx^2 + tg^2 x} + \frac{\sqrt{2x^2 + 6}}{\ln\lvert 5x - 45 \rvert}$	10) $Y = \cos^3 a + \frac{2ab - \sqrt{a^2 + b^2}}{e^{2ab} + \ln a}$
11) $Z = \frac{\lg(x + y) - \lvert x - 5y \rvert}{\sin 5{,}3x + \frac{2x}{\sqrt{\lvert x - 5y \rvert}}} - \frac{\cos^3 y}{x + y}$	12) $P = \frac{x \sin^y x}{\cos x + e^x} - \sqrt{\lvert \sin^y x + \cos^2 x \rvert}$
13) $Z = \frac{y \sin x - \cos x^2 y^3}{\sqrt{\lvert x^y - \cos x + 1 \rvert}} + \ln x$	14) $Z = \frac{\cos^2(x + y) - 2x}{\sin^3(x + y) + 3y} - \sqrt{x^4 + y^2}$

15) $P = 2tg(a+3b) + \frac{e^{2a} + 2{,}5Sin3b}{\sqrt{\lvert a+3b \rvert - 5}}$	16) $Z = \frac{\sqrt{x^2+y^4}}{\cos^2 x + e^{x+y}} + \frac{\sin^2(x+y)}{2x+y}$
17) $P = \frac{y \sin^2 x}{y^{x-2} + e^{xy}} - \frac{tgx - 3y}{x - 2}$	18) $Y = \frac{7 \cos a}{\sqrt{\lvert 5a+2b \rvert}} - \frac{\sin^2(5a+2b)}{e^{\cos a}}$
19) $Y = \frac{\sin^2(5x+2)}{5{,}3x+9} + \frac{7x - e^{2x}}{\sqrt{\lvert 5x+2 \rvert}}$	20) $P = \frac{\sin a + \cos b}{x + tg(ax+b)} - \frac{a^3 + \sin b}{\sqrt{x^2+7b^2}}$
21) $Y = \frac{tgx^2 - \sqrt{2x^2}}{ln\lvert 5x - 45 \rvert} - \frac{\sqrt{\lvert 5x - 45 \rvert}}{x + tg^2 x}$	22) $Y = \frac{\sqrt{a^2+5b^2}}{e^{2ab} + \ln a} + \cos^3\left(a^2 + 5b^2\right)$
23) $Z = \frac{\cos y}{lg\, x - \lvert y - x \rvert} + \frac{x + y - \sqrt{\lvert xy \rvert}}{5x + \frac{2xy}{\sqrt{\lvert y - x \rvert}}}$	24) $Z = \frac{x \sin x - \cos x}{x^y + \ln x} + \sqrt{\lvert x^y - x \sin x \rvert}$
25) $Z = \frac{\sin^2\left(x^4 + y^2\right) - 2}{\sin^3(xy) + e^{2y-x}} - \sqrt{x^4 + y^2}$	26) $Y = \frac{2 \sin a}{\sqrt{\lvert a+2b \rvert}} - \frac{\cos^2(a+2b)}{e^{\sin a}}$
27) $P = \frac{\sin(ax+b)}{x + tg(ax+b)} - \frac{a^3 + \sin b}{\sqrt{x^2+3b^2}}$	28) $P = \frac{x + \cos x}{(3y - x)^2} - \frac{\sqrt{\lvert 3y - x \rvert}}{3y \cos x}$
29) $P = \frac{\sqrt{a^2+b^2}}{a+2bc} - \frac{ln(a+2bc)}{a^2+b^2+c^2}$	30) $P = \frac{\sin^2 x}{y^{x-2} + e^{xy}} - \frac{x - 2}{\cos y^2}$

Контрольные вопросы:

1. Какими свойствами обладает алгоритм?
2. Какие способы используются для описания алгоритма?
3. Какие геометрические фигуры могут входить в блок-схемулинейного алгоритма?
4. Каковы типовые структуры алгоритма?
5. Какова структура программы?
6. Как обозначается оператор присваивания?
7. Какой оператор позволяет осуществить ввод значенияпеременной во время выполнении программы?
8. Какой оператор позволяет вывести на экран текстовоесообщение?
9. Каким образом каждый оператор в программе отделяетсядруг от друга?

Практическая работа №3. Составление программ циклической структуры. Обработка одномерных массивов.

Цель:

– закрепление знаний о программах циклической структуры, составление программы и работа с ней.

Краткие теоретические сведения

Алгоритм циклической структуры - это алгоритм, в котором происходит многократное повторение одного и того же участка программы. Такие повторяемые участки вычислительного процесса называются циклами. Программа циклической структуры содержит один или несколько циклов. Различают детерминированные циклы с заранее известным числом повторений и итерационные циклы, в которых число повторений заранее неизвестно. Изменяющаяся в цикле переменная называется параметром цикла.

Для организации цикла необходимо выполнить следующие действия:

1) задать перед циклом начальное значение параметра цикла;

2) изменять параметр перед каждым новым повторением цикла;

3) проверять условие повторения цикла;

4) управлять циклом, т.е. переходить к его началу, если он не закончен, или выходить из него по окончании.

В языке Паскаль существует 3 вида циклов: 1) цикл с параметром или цикл типа for, 2) цикл с предусловием или цикл типа while, 3) цикл с постусловием или цикл типа repeat ... until. В цикле типа for число повторений известно заранее, в циклах типа while и repeat ... until число повторений цикла заранее неизвестно, производится проверка условия повторения цикла: в цикле типа while - перед циклом, в цикле типа repeat ... until - после его окончания.

В циклах типов for и while повторяющяся часть (тело цикла) состоит из одного оператора, если требуется выполнить в цикле несколько операторов, они заключаются в операторные скобки begin ... end, образуя составной оператор. В цикле типа repeat ... until тело цикла помещается между зарезервированными словами языка (лексемами) repeat и until, операторные скобки не требуются, в названии цикла его тело условно обозначается тремя точками.

С помощью цикла типа for удобно находить суммы, произведения, искать максимальные и минимальные значения и т.п. При нахождении суммы некоторой переменной, например S присваивается значение 0, затем в цикле к этой переменной прибавляется соответствующий член заданной последовательности. При нахождении произведения переменной присваивается значение 1, затем в цикле эта переменная умножается на общий член последовательности.

Пример цикла типа for. Вычисление n чисел Фибоначчи:

F1=1; F2=1;…; Fn=Fn-1+Fn-2 ,

например F3=F2+F1=1 + 1 = 2; F4 = 2 + 1 = 3 и т.д.

```
program fib; {Нахождение чисел Фибоначчи}
var x, y, z, i, n : integer;
begin
writeln (‘Введите n’); read (n);
x:=1; y:=0;
for i:=1 to n do
begin
z:=x ; x:=x+y ; y:=z ;
write (' ', x ) ;
end;
end.
```

Пример цикла типа while. Составление таблицы функции y = a3/(a2+x2) для x, принадлежащих отрезку [-1; 1] с шагом 0.1. Так как параметр цикла типа for должен быть целочисленным, удобнее использовать цикл while, в котором значение x можно изменять при каждом шаге на x = 0.1

```
program cycl_while;
uses crt; {вызов модуля Crt для управления режимом экрана}
var
a, x, y : real ; i : integer;
begin
clrscr ; {процедура очистки экрана из модуля Crt}
writeln ('Введите a ' ); readln (a);
x:= -1.; writeln (‘ x ‘, ‘ y ’);
while do x<1.05 begin
y:= sqr(a)*a/ (a*a+x*x);
writeln (x:6:2, y:8:4);
x:=x+0.1
end
end .
```

Условие x<1.05 соответствует каждому значению x плюс половина шага .

Пример цикла типа repeat ... until. Определить число n, при котором сумма квадратов натурального ряда чисел от 1 до n не превысит величину K, введенную с клавиатуры. Т.е.

```
S ≤ K, где S= = 12 + 2 2 + . . . + n 2
program sum_sq; {Сумма квадратов натурального ряда }
uses crt;
var k, s, n : integer;
begin
clrscr ; writeln( 'Введите K' ); readln (k);
s:=0; n:=1;
repeat
s :=s+n*n; n := n+1;
until s > k;
writeln ('N= ', n : 3, ' s= ' , s : 5 );
end.
```

Цикл повторяется до тех пор, пока условие записанное после ключевого слова until, будет ложным (не выполняется). Как только это условие выполнится, происходит выход из цикла. После окончания цикла производится печать результата (оператор writeln).

Отметим, что цикл с предусловием (типа while) может не выполниться ни разу, цикл с постусловием repeat ... until выполнится по крайней мере 1 раз.

Когда число повторений цикла неизвестно заранее, применяются циклы с предусловием или с постусловием. Когда число повторений цикла известно заранее, как правило, применяется цикл типа for. Но любой цикл типа for можно заменить циклом с предусловием или постусловием.

Практические задания

Задания

1. Разобрать и проанализировать приведенные выше программы с циклами типа for, while и repeat ... until

2. Составить алгоритм задачи: Вычислить сумму ряда, указанного в варианте задания для любого значения N, введенного с клавиатуры.

3. Составить 3 варианта программ циклической структуры типа for, while и repeat, откомпилировать их, ввести исходные данные, сравнить полученные результаты.

Варианты заданий

1. $\sum_{i=1}^{N} \frac{1+i}{2+i^2}$. 2. $\sum_{i=1}^{N} \frac{6}{4+i^3}$.

3. $\sum_{i=1}^{N} \frac{3i^2}{24+i}$. 4. $\sum_{i=1}^{N} \frac{8+3i^2}{12+i^2}$.

5. $\sum_{i=1}^{N} \frac{8}{16+i^2}$. 6. $\sum_{i=1}^{N} \frac{3+i}{20+i}$.

7. $\sum_{i=1}^{N} \frac{12-i^2}{6+i}$. 8. $\sum_{i=1}^{N} \frac{3-i}{2+i^2}$.

Контрольные вопросы:

1. Какой алгоритм является алгоритмом циклической структуры?
2. Типы циклов в языке Паскаль.
3. Цикл с параметром в языке Паскаль.
4. Циклы с предусловием и постусловием в языке Паскаль.
5. Какой из операторов цикла целесообразнее использовать для вашей задачи?
6. Какой цикл выполнится по крайней мере один

Практическая работа №4. Обработка двумерных массивов.

Цель:

– приобретение практических навыков в составлении программ с массивами.

Краткие теоретические сведения

Массивы - структурированный тип данных с элементами одного типа. Количество элементов определяет размер массива. Например, массив составляют заработные платы сотрудников подразделения предприятия, здесь число элементов равно числу сотрудников; массив образуют набор чисел, их количество равно числу элементов массива. Номер элемента массива называется его индексом. Массив может иметь не один, а большее число индексов. Число индексов называется размерностью массива, например, массив с двумя индексами называется двумерным массивом. Таким двумерным массивом является, в частности, матрица системы n линейных алгебраических уравнений с n неизвестными. В то же время столбец свободных членов этой системы является одномерным массивом.

Массив должен быть описан либо в разделе переменных VAR, либо в разделе типов TYPE следующим образом:

TYPE <имя_типа> = ARRAY [ti,t2....,tn] OF <тип_элементов_массива>;

VAR <имя_массива> : <имя_типа> ;

или

VAR<имя_массива>: ARRAY [ti,t2,...,tn]OF<тип_элементов_массива>;

где tl,...,tn - тип индекса (перечислимый или интервальный);

Например:

1) VAR a: ARRAY [1..5] OF real;

Описан массив a действительных чисел, который состоит из 5 элементов.

2) TYPE t = ARRAY[1..3,1..4] OF integer;

VAR b : t;

Описана матрица b из 3 строк и 4 столбцов, элементы которой являются целыми числами.

3) TYPE

tl = ARRAY [1..4] OF integer;

t = ARRAY [1..5] OF t1;

VAR d : t;

к : t1;

В начале описан тип одной строки t1, затем тип всей матрицы t через тип строки t1. В разделе переменных указано ,что d-двумерный массив размером (5,4), а k - одномерный массив k(4).

2.1. Пример обработки одномерного массива

Дан одномерный массив MAS(12) из вещественных чисел. Найти наибольший элемент массива и его индекс.

```
program pr4_1 ;
const n = 12; (*константа n определяет размер массива в описании*)
type
m = array [1..n] of real ; (* m – тип массива mas *)
var mas : m ;
i , num : integer; max : real;
begin
for i:=l to n do (*ввод элементов массива mas по 1 в строке*)
begin
writeln(‘введите элемент массива’, i) ; read (mas[i]);
end;
```

```
num :=1; max := mas [1];
for i: =2 to n do
if mas [ i ] > max then begin
max := mas [i ]; num := i ;
end;
writeln; (* вывод массива в строку*)
for i:=l to n do write (mas [ i ]:5:1 ); writeln;
writeln (‘максимальный элемент = ‘, max:4:1,‘ его индекс=’ ,num );
end.
```

Переменная max сравнивается с элементами массива, и если элемент массива больше max, то переменной max присваивается значение элемента массива, а переменной num - индекс этого элемента. По окончании цикла переменная max будет иметь значение, равное максимальному элементу массива, а переменная num - значение индекса этого элемента.

Как правило, при обработке многомерных массивов используются вложенные циклы, т.е. цикл по столбцам располагается внутри цикла по строкам.

2.1. Пример обработки многомерного массива

Дана матрица A(3,4), и вектор B (4), состоящие из целых чисел . Умножить матрицу A на вектор B .

```
program pr4-2 ;
const m=3; n=4;
var
a : array [ l .. m, 1 .. n ] of integer; (* описание матрицы *)
b : array [ 1 .. n ] of integer; (* описание вектора *)
c : array [ 1 .. m ] of integer; ( * описание C *)
i, j: integer;
begin
for i:=l to m do (* ввод матрицы *)
begin
writeln (‘введите элементы ‘, i , ‘-той строки’);
for j:=1 to n do read (a [i, j] ); writeln;
end;
writeln (‘введите элементы вектора’);
for j:=1 to n do (* ввод вектора *)
read (b[ j]); writeln;
for i:=l to m do

begin
c [ i ]:=0; for j:=l to n do c[i] := c[ i ]+ a[i , j]* b[j];
end;
for i:=l to m do (*форматный вывод матрицы *)

begin
for j:=1 to n do write (a [i, j]: 4 ); writeln;
end;
for j:=l to n do write (b [ j ] :4); (* вывод массива B *)
writeln ;
for i:=l to m do write (c [ i ] :4); (* вывод массива C *)
readln;
end.
```

В программе элементы матрицы вводятся по строкам по одному с подтверждением

клавишей Enter. А выводятся в общепринятом виде: каждая строка матрицы с новой строки экрана (цикл i по строкам внешний, а цикл j – внутренний).

Практические задания

Задание 1.

1. Набрать и откомпилировать приведенные выше программы, исправить выявленные ошибки. Ввести элементы массива, убедиться в правильности выполнения программ.

2. Составить и выполнить программы с применением массивов согласно вариантам заданий .

Задание 2.

Варианты задания 1. Ввести массив А из 10 элементов

1. Найти наибольший элемент и переставить его с первым элементом. Преобразованный массив вывести.
2. Найти наименьший элемент и переставить его с последним элементом. Преобразованный массив вывести.
3. Найти произведение положительных элементов и вывести его на экран.
4. Найти произведение отрицательных элементов и вывести его на экран.
5. Найти сумму положительных элементов и вывести ее на экран.
6. Найти сумму отрицательных элементов и вывести ее на экран.
7. Найти сумму элементов, больших 3 и меньших 8 и вывести ее на экран.
8. Найти сумму элементов, меньших по модулю 5 и вывести ее на экран.

Задание 3.

Варианты задания 2

1. Даны матрица А размером m*n и вектор В размером m. Записать на главную диагональ элементы вектора, а в вектор - элементы главной диагонали.
2. Выбрать максимальный элемент матрицы С (размер m*n), элементы четных строк разделить на максимальный элемент, а к элементам нечетных прибавить максимальный элемент .
3. Найти минимальный элемент матрицы С (размер m*n), и поменять его местами с первым элементом.
4. Дана матрица Е размером m*n. Вычислить суммы элементов каждого столбца. Определить наибольшее значение этих сумм и номер соответствующего столбца.
5. В матрице К размером m*n найти в каждом столбце произведение отрицательных элементов и количество нулевых элементов в матрице.
6. Даны две матрицы А и В одинаковой размерности m*n. Получить матрицу С = max (a i j, b i j), и матрицу D = min (a i j, b i j).
7. Дана матрица Р размером m*n. Найти сумму минимальных элементов каждого столбца матрицы.
8. Даны две матрицы: А размером m*k и В размером k*n. Получить матрицу С=А*В.

Контрольные вопросы:

1. Что такое массив? индекс элемента массива?
2. В каких пределах изменяются индексы элементов массива?
3. Как ввести и вывести элементы вектора в строку и в столбец?
4. Способы описания массивов.
5. Вывод матрицы по строкам и по столбцам

Практическая работа №5. Работа со строками.

Цель:

– изучение символьных типов данных CHAR и STRING и операций над ними.

Краткие теоретические сведения

Наряду с числовой информацией в Паскале используется алфавитно-цифровая или символьная информация, которая включает в себя заглавные, строчные буквы, цифры от 0 до 9 и вспомогательные символы. Для описания символьных переменных используется тип данных CHAR или STRING.

2.1. Тип данных CHAR

Каждая переменная символьного типа может принимать значение только одного символа. Все символы упорядочены в соответствии с принятым в ЭВМ коде (например ASCII). При этом порядковый номер символов называется кодом (например, код латинского символа 'A ' равен 65; символа '3' равен 51).

Для символьных данных не определены никакие арифметические операции, но они могут сравниваться по своим кодам, участвовать в чтении, печати, операторах присваивания. Существуют две стандартные функции преобразования :

1) ORD (C) принимает значение кода символа C;

2) значение функции CHR(I) является символ с кодом Например:ORD('A ')=65

CHR(ORD(C))=C ; CHR (65) = A ;

Строка - это последовательность символов. Строку можно представить как массив, элементы которого имеют тип CHAR. Например:

BUK: array[1..17] of char;

Массив BUK-массив символов, который содержит 17 символов. Если символов меньше, то строка дополняется пробелами справа. В противном случае возникает ошибка несоответствия типов. Так как массивы символов являются обычными массивами с элементами типа CHAR, они обладают всеми свойствами массивов.

Пример: Из набора 10 любых символов напечатать только заглавные английские буквы и их коды.

```
program lr2;
type sl =array [1..10] of char;
 var   s: sl; {описание массива символов}
          i: integer;
begin
          writeln ('введите 10  символов'):
 for i:=l to 10 do  readln (s[i]);          {ввод массива}
 for i:=l to 10 do
            if (s[i]>=’A’) and (s[i]<= ‘Z’) then
          writeln (‘Символ :’, s[i], ‘ его код =’, ord (s[i]);
 readln;
end.
```

2.2. Тип данных STRING

В Турбо Паскале предусмотрен тип данных STRING. Переменная типа STRING может принимать значения переменной длины. Максимально возможная длина переменной 255 символов.

Например:

```
 str: STRING[200];
              ow: STRING[10];
```

В скобках указывается максимальная длина для данной переменной. Для ввода значений типа STRING необходимо использовать READLN, а не READ. За один раз может быть введена только одна строка. Две строки можно сравнивать, используя операции отношения (сначала сравниваются самые левые символы, если они равны, то сравниваются следующие). Для работы с переменными типа STRING используют следующие стандартные

процедуры и функции:

1) Функция LENGTH

C:=LENGTH(str); Переменной C будет присвоено целое значение, показывающее количество символов в строковой переменной str .

2) Функция CONCAT - сцепление строк в порядке их перечисления.

str:=CONCAT(st1,st2,...,stN);str-переменная типа STRING, состоящая из строк st1,...,stN.

3) Функция POS

P:=POS (st1, st2); P-целое число, показывающее номер позиции, с которой начинается строка st1 в строке st2.

4) Функция COPY

S1:=COPY(str, I, J); Sl-символьная подстрока, выделенная из строки str с позиции I, длиной J символов.

5) Процедура DELETE(Str, I, J);

Из строки str удаляется J символов, начиная с I позиции.

6) Процедура INSERT(Str1, Str2, I);

Строка Str1 вставляется с I позиции в строку Str2.

7) Процедура STR (V, S1);

Числовое значение переменной V преобразуется в строку символов и записывается в строку S1.

7) Процедура VAL (S1, V, C);

Строковое выражение S1 преобразуется в величину целочисленного или вещественного типа и записывается в переменной V . Если при этом ошибок не обнаруживается, то C будет равно 0 . В противном случае значение C будет равно номеру позиции первого ошибочного символа и V будет неопределено. Строка S1 не должна содержать незначащих пробелов, переменная V может быть целой или вещественной, а переменная C - только целой .

Пример: Подсчитать количество слов во введенной с клавиатуры строке.

```
program lr2;
var
  s: string[30];
     kol, i, n: integer;
begin
  writeln ('введите строку'); readln (s);
  kol:=0;          {счетчик количества слов}
  n:= length(s);          {определяем длину введенного текста}
  s:= concat('  ',s); {добавляем пробел к первому слову}
  for i:=1 to n do
  if (copy (s,i,1)='  ') and (copy (s,i+1,1)<>' ')
                then  kol := kol+1;{подсчет количества слов}
    writeln (s,'  количество слов= ',  kol);
    readln
   end.
```

Практические задания

Задание

1. Набрать и откомпилировать приведенные выше программы, исправить выявленные ошибки. Ввести разные исходные данные, убедиться в правильности выполнения программ.

2. Составить программы с использованием символьных данных согласно вариантам заданий, откомпилировать их, проверить полученные результаты.

Задание 2.

Варианты заданий. Текст вводится с клавиатуры в символьную переменную. Исходный текст и результаты распечатать.

1. Вывести на печать список слов, имеющих приставку (несколько букв), задаваемую с терминала.
2. Раздвинуть заданный текст, вставив введенную с клавиатуры последовательность символов после 1-го символа каждого слова.
3. В заданном тексте слова разделены запятыми. Напечатать список слов, начинающихся с символа, введенного с клавиатуры.
4. Определить наличие слов в заданном тексте, содержащих сочетание символов, задаваемое с экрана.
5. Из заданной последовательности слов удалить слова, содержащие числа.
6. Каждое слово текста преобразовать таким образом, чтобы оно читалось слева направо.
7. Подсчитать количество слов, разделенных запятыми, содержащих k гласных букв (k-задается с экрана).
8. Из введенного текста сначала распечатать слова, заканчивающиеся на согласную букву, а потом на гласную букву.

Контрольные вопросы:

1. Описание переменных символьного типа.
2. Функции преобразования переменных символьного типа.
3. Отличие типов данных STRING и CHAR.
4. Приведите конкретные примеры использования каждой функции и процедуры.

Практическая работа №6. Работа с данными типа множество.

Цель:

– познакомить с понятием "множество" в языке программирования Pascal; выработать навыки работы со структурой данных множество.

Практические задания

Перед выполнением работы необходимо ознакомится с правилами описания и использования переменных типа множество, типизированных констант типа множество, переменных, заданных перечислением, изучить допустимые операции над переменными этих типов.

Понятие множества в языке ПАСКАЛЬ основывается на математическом представлении о множествах: это ограниченная совокупность различных элементов. Для построения конкретного множественного типа используется перечисляемый или интервальный тип данных. Тип элементов, составляющих множество, называется базовым типом.

Множественный тип описывается с помощью служебных слов Set of, например:

```
type  M= Set of B;
```

Здесь M - множественный тип, B - базовый тип.

Пример описания переменной множественного типа:

```
type
  M= Set of 'A'..'D';
var
  MS : M;
```

Принадлежность переменных к множественному типу может быть определена прямо в разделе описания переменных:

```
var
  C : Set of 0..7;
```

Константы множественного типа записываются в виде заключенной в квадратные скобки последовательности элементов или интервалов базового типа, разделенных запятыми, например:

['A', 'C'] [0, 2, 7] [3, 7, 11..14].

Константа вида [] означает пустое подмножество.

Множество включает в себя набор элементов базового типа, все подмножества данного множества, а также пустое подмножество. Если базовый тип, на котором строится множество, имеет K элементов, то число подмножеств, входящих в это множество, равно 2 в степени K. Пусть имеется переменная P интервального типа:

```
var
  P : 1..3;
```

Эта переменная может принимать три различных значения - либо 1, либо 2, либо 3. Переменная T множественного типа

```
var
  T : Set of 1..3;
```

может принимать восемь различных значений:

```
[ ]     [1,2]
[1]     [1,3]
[2]     [2,3]
[3]     [1,2,3]
```

Порядок перечисления элементов базового типа в константах безразличен.

Значение переменной множественного типа может быть задано конструкцией вида [T], где T - переменная базового типа.

К переменным и константам множественного типа применимы операции присваивания (:=), объединения(+), пересечения(*) и вычитания(-):

['A','B'] + ['A','D'] даст ['A','B','D']
['A'] * ['A','B','C'] даст ['A']
['A','B','C'] - ['A','B'] даст ['C'].

Результат выполнения этих операций есть величина множественного типа.

К множественным величинам применимы операции: тождественность (=), нетождественность (<>), содержится в (<=), содержит (>=). Результат выполнения этих операций имеет логический тип, например:

['A','B'] = ['A','C'] даст FALSE
['A','B'] <> ['A','C'] даст TRUE
['B'] <= ['B','C'] даст TRUE
['C','D'] >= ['A'] даст FALSE.

Кроме этих операций для работы с величинами множественного типа в языке ПАСКАЛЬ используется операция in проверяющая принадлежность элемента базового типа, стоящего слева от знака операции, множеству, стоящему справа от знака операции. Результат выполнения этой операции - булевский. Операция проверки принадлежности элемента множеству часто используется вместо операций отношения, например:

A in ['A', 'B'] даст TRUE,
2 in [1, 3, 6] даст FALSE.

При использовании в программах данных множественного типа выполнение операций происходит над битовыми строками данных. Каждому значению множественного типа в памяти ЭВМ соответствует один двоичный разряд. Например, множество

['A','B','C','D']

представлено в памяти ЭВМ битовой строкой

1 1 1 1.

Подмножества этого множества представлены строками:

['A','B','D'] 1 1 0 1
['B','C'] 0 1 1 0
['D'] 0 0 0 1

Величины множественного типа не могут быть элементами списка ввода - вывода.

В каждой конкретной реализации транслятора с языка ПАСКАЛЬ количество элементов базового типа, на котором строится множество, ограничено. В TURBO PASCAL количество базовых элементов не должно превышать 256.

Инициализация величин множественного типа производится с помощью типизированных констант:

```
const  seLit: Set of 'A'..'D'= [];
```

Проиллюстрируем применение данных множественного типа на примере.

Пример. Составить программу, которая вырабатывает и выводит на экран дисплея наборы случайных чисел для игры в "Спортлото 5 из 36".

Для заполнения каждой карточки спортлото необходимо получить набор из пяти псевдослучайных чисел. К этим числам предъявляются два требования:

числа должны находиться в диапазоне 1..36;

числа не должны повторяться.

```
Program Lotto;
 var
    nb, k: Set of 1..36;
    kol, l, i, n: Integer;
 begin
   Randomize;
   WriteLn('ВВЕДИ kol');
   ReadLn(kol);
```

```
      nb:=[1..36];
      for i:=1 to kol do
        begin
         k:=[];
         for l:=1 to 5 do
           begin
             repeat
               n:=Random(36)
             until (n in nb) and not (n in k);
             k:=k+[n];
             Write(n:4)
           end;
         WriteLn
        end
       end.
```

Пример
Пример1: Дан текст. Определить каких букв больше - гласных или согласных.

Этапы решения задачи:
1. Составим блок схему программы
Опишем подробнее блок "Подсчитываем количество гласных и согласных букв"
Рассмотрим блок "Печатаем соответствующее сообщение"
Запишем блок-схему целиком

2. Переведем алгоритм на язык Паскаль

```
program example1;
const
  glasn=['а','е','и','о','у','ы','э','ю','я'];
   soglas=['б','в','г','д','ж','з','й','л','м',
        'н','р','к','п','с','т','ф','х','ц','ч','ш','щ'];
 var
   st: string;
     g,s,i:integer;
begin
  write('Введите строку> '); readln(st);
    g:=0; s:=0;
      for i:= 1 to length(st) do
        if st[i] in glasn then inc(g) else if st[i] in soglas then inc(s);
        if g> s then writeln('Гласных больше')
    else if g< s then writeln('Согласных больше')
  else writeln('Согласных и гласных букв поровну');
 readln;
end.
```

Практические задания:

Примечание: Гласные буквы - а,е,и,о,у,ы,э,ю,я (ё обычно не входит в литерный тип); согласные - все остальные буквы, кроме ь, ъ; звонкие согласные - б,в,г,д,ж,з,й,л,м,н,р; глухие согласные - к,п,с,т,ф,х,ц,ч,ш,щ.

Задание 1.
Дан текст из строчных латинских букв, за которым следует точка. Напечатать:
- первые вхождения букв в текст, сохраняя их взаимный исходный порядок;

- все буквы, входящие в текст не менее двух раз;
- все буквы, входящие в текст по одному разу.

Задание 2.

Дана непустая последовательность слов из строчных русских букв; между соседними словами - запятая, за последним словом - точка. Напечатать в алфавитном порядке:

все гласные буквы, которые входят в каждое слово; все согласные буквы, которые не входят ни в одно слово;

все звонкие согласные буквы, которые входят хотя бы в одно слово; все глухие согласные буквы, которые не входят хотя бы в одно слово;

все согласные буквы, которые входят только в одно слово; все глухие согласные буквы, которые не входят только в одно слово;

все звонкие согласные буквы, которые входят более чем в одно слово; все гласные буквы, которые не входят более чем в одно слово;

все звонкие согласные буквы, которые входят в каждое нечетное слово и не входят ни в одно четное слово; все глухие согласные буквы, которые входят в каждое нечетное слово и не входят хотя бы в одно четное слово.

Задание 3.

Имеются три множества символьного типа, которые заданы своими конструкторами:

Y1=['A','B','D','R','H']

Y2=['R','A','H','D']

Y3=['A','R'].

Сформировать новое множество .

Предусмотреть формирование исходных множеств с клавиатуры.

Задание 4. Подсчитать общее количество цифр и знаков '+', '-', и '*', входящих в строку s.

Контрольные вопросы:

1. Что такое множество, как оно описывается в языке Pascal?
2. Как определить новый тип данных с использованием перечисления?
3. Как описываются типизированные константы типа множество?
4. Как осуществляется ввод-вывод значений переменных типа множество?
5. Какие типы данных используются в качестве базовых при объявлении типа множество?
6. Какие операции определены над множествами?
7. Какие операции допустимы над переменными, заданными перечислением?
8. Чем похожи и чем отличаются множества и массивы?
9. Какое значение у выражений: а) x in [x]; б) [ ] <= [x,y,z]; в) [x]<>[x,x,x]

Практическая работа №7. Файлы последовательного доступа.
Цель:
– изучение файловых типов данных, приобретение практических навыков создания и обработки файлов.

Краткие теоретические сведения

Работа с файлами

Информация, вводимая с клавиатуры или обрабатываемая с помощью программных средств Бейсика, размещается в оперативной памяти компьютера.

Алгоритм, набранный в Бейсике, может быть сохранен на диске в виде файла.

Файл — это поименованная область на магнитном или лазерном диске. В файлах могут содержаться тексты, графические и видеоизображения, звуки и музыка, таблицы и базы, данные программы, данные для этих программ.

Требования к имени файла

- имя не должно быть больше чем 8 символов;
- имя может состоять из букв латинского алфавита, цифр и символов, например, _, -, (,), $ и некоторых других.
- в имени файла **запрещены** символы <Пробел>, *, точка, запятая, кавычки, двоеточие.

Впрочем, злоупотреблять специальными символами не стоит — букв и цифр вполне хватает.

Расширение файла

Файл имеет расширение.

Расширение имени файла (англ. *filename extension*, часто говорят просто расширение файла или расширение) — последовательность символов, добавляемых к имени файла и предназначенных для идентификации типа (формата) файла. Расширение имеет длину не более трех символов, указывается через точку после имени.

Расширение файла	Описание формата файла
*.aif, *.aifc, *.aiff	Файлы аудиоданных
*.asm	Исходный текст программы на Ассемблере.
*.avi	Основной формат видеоизображений
*.bas	Текст программы на языке алгоритмическом языке Basic и различных его вариантах (GWBasic, TurboBasic, QuickBasic)
*.bmp	Формат графических файлов (растровая графика).
*.com	Исполняемый файл в двоичном коде
*.cpp	Текст программы на языке C++
*.doc	Файл с документами или продукт работы текстового процессора Microsoft Word for Windows
*.dot	Шаблон документа текстового процессора MS Word
*.exe	Это всегда исполняемый бинарный файл
*.gif	(Graphics Interchange Format). Растровый графический формат фирмы CompuServe
*.htm, *.html	Специальный файл текстового типа, написанный на Hyper Text Markup Language
*.mdb	Файл баз данных Microsoft Access

*.mov -	Формат хранения видео и аудио
*.pas -	Текст программы на языке Pascal
*.ppt	Файл с презентацией Microsoft PowerPoint
*.sys -	Системные файлы ядра DOS IO.sys и MSDOS.sys.
*.txt	Текстовой файл, созданный в блокноте
*.xls	Файл работы табличного процессора Microsoft Excel
*.zip -	Файл архива сжатого архиваторами

В файлах вы можете хранить как исходные данные для обработки, так и результаты работы программы.

Для работы в Бейсике необходимы файлы, хранящие однородные по типу или структуре сведения, о каких-либо объектах. Набор данных о каком-либо одном объекте называется **записью.**

Файл может быть пустым, т. е. содержать 0 байт информации, но имя файла и символ конца файла будут присутствовать. (**Байт** - единица измерения количества информации, объема памяти и емкости запоминающего устройства. По умолчанию байт считается равным 8 битам).

Записи могут содержать данные разных типов, но должны быть обязательно одинаковы по структуре, например:

"Запорожец", "4067 ЛДЕ", "1972", "100$"

"ГАЗ-34", "6666 ЛАА", 1989, "3500$"

В соответствии со способом доступа к файлам они делятся на два вида.

1. Файл с последовательным доступом;
2. Файл с прямым доступом.

Файлы последовательного доступа наиболее просты как в организации, так и в работе с ними. Записи обрабатываются последовательно одна за другой.

Информация в таких файлах хранится в виде текста в кодах ASCII. Такие файлы легко просмотреть на экране, используя любой простейший редактор, или в самом Бейсике.

Простота — хорошо, а последовательность в данном случае — плохо. Если информация об интересующем объекте упорядочена в файле по алфавиту, то придется перебирать практически весь файл, чтобы добраться до нужной записи. Отсюда, при большом информационном объеме файла обработка его замедляется.

Файлы прямого доступа хранят информацию в специальном формате, в котором каждая запись занимает строго фиксированную одинаковую с остальными длину. Такие файлы занимают на диске больше места, чем файлы последовательного доступа, но скорость работы с ними значительно выше.

Операции над файлами

Независимо от того, какие действия происходят с информацией, хранящейся в файле, производятся следующие обязательные операции:

1. открытие файла;
2. чтение и запись обрабатываемых данных;

В Паскале определены текстовые файлы, типизированные и нетипизированные. Файл, не содержащий ни одного элемента, называется пустым. Создается файл путем добавления новых записей в конец первоначально пустого файла. Длина файла, т.е. количество элементов, не задается при определении файла.

Все файлы должны быть описаны в программе либо в разделе переменных VAR, либо в разделе типов TYPE. Под чтением файла понимают ввод данных из внешнего файла,

находящегося на диске, в оперативную память машины. Запись в файл - вывод результатов работы программы из оперативной памяти на диск в файл.

Работа с файлами выполняется следующими процедурами:

Assign – устанавливает связь между именем файла в программе (файловой переменной) и физическим именем файла, принятым в ОС.

Reset - открывает существующий файл для чтения.

Rewrite – создает и открывает новый файл для записи на внешнем устройстве (если файл ранее существовал, вся предыдущая информация из него стирается).

Close - закрывает открытый файл.

Для определения конца файла используется стандартная встроенная функция EOF (файловая переменная), которая принимает значение True, если достигнут конец файла, и значение False в противном случае.

2.1. Текстовые файлы

Текстовые файлы – файлы на диске, состоящие из символов ASCII. Для разделения строк используются символы «конец строки». Текстовые файлы являются файлами с последовательным доступом. В любой момент времени доступна только одна запись файла. Другие записи становятся доступными лишь в результате последовательного продвижения по файлу. Текстовые файлы внутренне разделены на строки, длины которых различны. Для разделения строк используется специальный маркер конца строки. Объявляются текстовые файлы переменной типа text. Обрабатывать их можно только последовательно и с помощью процедур и функций:

Readln (f , st)- чтение строки st из файла f и переход на начало следующей ;

Writeln (f, st)- запись строки st в файл f и маркера конца строки ;

Append (f) - процедура, открывающая файл f для добавления строк в конец файла;

Eoln (st)- логическая функция, результат выполнения которой равен TRUE, если достигнут маркер конца строки st.

Пример 1. Создать текстовый файл, в который записать 3 предложения. Прочитать этот файл, вывести его содержимое на экран. Определить длину каждого предложения.

```
Program File_text;
  var
 f1 : text;
 st : string;
 n: byte;
   begin
assign  (f1, 'file1.txt'); {связать с файлом file1.txt  файловую переменную f1  }
        rewrite  (f1); { создать новый файл  с именем file1.txt }
writeln ( f1, 'Очень полезно изучать'); { записать предложения в файл}
writeln ( f1, ' всем  студентам ');
writeln (f1, ' язык  Pascal ');
close (f1);    { закрыть файл для записи }
        reset (f1); { открыть файл для чтения }
        while not eof (f1) do { пока не конец файла  f1}
          begin
            readln (f1, st);    {читаем строку из файла f1 }
            writeln(st);        { выводим на экран }
            n:= length (st);    {определяем длину строки }
            writeln ('  длина =',n);
          end;
         close (f1); { закрыть файл  для чтения}
          end .
```

Практические задания

Задание

1. Разобрать и проанализировать приведенные программы.
2. Используя подпрограммы, создать внешний файл из 7 записей, прочитать созданный файл и, применяя режим прямого доступа, выполнить задания по своему варианту.

Контрольные вопросы:

1. Чем отличается файл от массива?
2. Особенности организации текстовых файлов.
3. Что понимается под чтением, и что под записью в файл?
4. Типы доступа к файлам.
5. Назначение процедуры SEEK.
6. Привести пример корректировки К-той записи.
7. Как определить размер файла?

Практическая работа №8. Типизированные файлы.

Цель:

– изучение файловых типов данных, приобретение практических навыков создания и обработки файлов.

Краткие теоретические сведения

Типизированные файлы

Типизированные файлы – это файлы, состоящие из нумерованной последовательности объектов (записей) любого типа. С такими файлами можно работать в режиме прямого доступа, при котором выполняется непосредственное обращение к любой записи файла. Каждая запись файла имеет свой номер, начиная с 0 и т.д.

Процедуры и функции обработки файлов:

1) Write и Read- записывают и читают информацию из указанного файла и перемещают указатель файла к следующей записи.
2) Seek (файловая переменная, номер записи); процедура перемещения указателя на запись файла с заданным номером .
3) Truncate (файловая переменная); процедура, усекающая файл по текущей позиции указателя файла, т.е. все записи, находящиеся после указателя файла, удаляются.
4) Функция Filesize (файловая переменная); имеет тип Integer и определяет размер файла, т.е. число записей.
5) Функция Filepos (файловая переменная); имеет тип Integer и возвращает текущую позицию указателя файла.

Для добавления записей в конец файла используются процедуры:

```
Readln (a );
Seek (f, filesize (f));
        Write (f, a);
```

При этом указатель устанавливается за конец файла, т.к. нумерация записей начинается с нуля. После чего с помощью Write можно добавлять записи. Открывать файл можно только процедурой Reset (f).

Для того, чтобы в режиме произвольного доступа считать, а затем изменить значение записи, следует выполнить два вызова процедуры Seek.Один вызов перед операцией Read, а другой - перед операцией Write (т.к. Read после чтения записи переместит указатель к следующей записи).

Пример: Создать файл из списка 10 студентов с их оценками (номер, Ф.И.О. и три оценки). Вывести его содержимое на экран, изменить фамилию студента с номером, введенным с клавиатуры, заново прочитать файл.

```
Program  lab6;
Type
  wed  =  record      {Тип wed включает 3 поля: n, fio, bal}
     n : byte ;       fio : string[15] ;
    bal : array [1..3] of byte;   {Поле bal – массив из 3 оценок }
         end;
Var    spisok : wed ;       {Запись spicok  типа  wed}
          sp : file  of wed;  {Файл записей типа wed}
procedure vvod;     { процедура создания файла}
 var    i,j:byte;
 begin
{ оператор assing  находится в основной прграмме }
rewrite ( sp);            {открытие файла для записи}
  with spisok do
      For   i:=1  to  10  do      begin
          n:=i;
           writeln (' Введите фамилию -  ',  i );    readln (fio);
```

```
          writeln (' Введите 3 оценки  ', fio );    For  j:= 1 to 3 do  readln ( bal [j] );
          write (sp , spisok);     { запись в файл информации о студенте}
           end;
  close (sp);      { закрытие файла для записи }
  end;
procedure    print;  { процедура  чтения   и печати всего файла }
 var    j : byte;
 begin
    reset ( sp);              {открытие файла для чтения}
    writeln (‘               Список студентов:  ‘);
    while not eof (sp) do
       with spisok do
          begin
            Read (sp, spisok);   {чтение данных из файла}
           write (n,'  ',fio);      {вывод записи на экран}
            For  j:= 1 to 3 do  write (' ', bal [j] );
           writeln ;
          end;
     readln;
    close (sp) ;
   end;
procedure       work;
 var    num: integer;
 begin
    reset ( sp);              {открытие файла для чтения}
    writeln ('номер= ');    readln (num);
    seek (sp, num-1); {поиск записи с указанным номером (нумерация записей с 0)}
    read (sp,spisok);{чтение и перемещение указателя к след. записи}
    write ('fio='); writeln (spisok.fio);
    seek (sp,filepos(sp)-1); {возвращение к изменяемой записи }
    writeln (‘ Введите новую фамилию’  ); readln (spisok.fio);
    write (sp, spisok);  {запись в файл измененной записи}
    close (sp);
  end;
begin      {начало основной программы}
     assign (sp,'Vedom.DAT'); {связать файловую перем-ю  sp с файлом Vedom.dat}
      vvod;      print; {процедуры создания и чтения файла}
      work;   print;   {корректировка и чтение измененного файла}
     readln
 end.
```

Практические задания

Задание

1. Запись имеет вид: фамилия, пол, год рождения и рост. Вывести данные о самом высоком спортсмене.

2. Запись имеет вид: название вуза, число студентов, количество факультетов. Добавить в конец файла информацию о трех новых вузах.

3. Запись имеет вид: название издания, газеты или журнала, стоимость одного экземпляра, количество экземпляров в год . Вывести на печать информацию о самом дешевом издании.

4. Запись имеет вид: фамилия студента, номер зачетной книжки, 4 оценки за экзамен. Выводить информацию о всех двоечниках и корректировать ее.

5. Запись имеет вид: фамилия спортсмена, его номер, количество набранных очков. Поменять местами в файле записи о первых двух спортсменах. *

6. Запись имеет вид: фамилия, номер телефона, дата рождения. Внести в начало списка информацию о четырех новых знакомых.

Контрольные вопросы:

1. Как можно описать файлы?

2. Какие типы файлов существуют в языке Pascal?

3. Как организовать прямой доступ к типизированным файлам?

4. Особенности работы с типизированными файлами?

5. Особенности работы с текстовыми файлами?

6. Особенности работы с нетипизированными файлами?

7. Основные стандартные процедуры и функции для работы с типизированными файлами ?

8. Основные стандартные процедуры и функции для работы с нетипизированными файлами?

9. Основные стандартные процедуры и функции для работы с текстовыми файлами ?

10. Общий алгоритм создания файла.

11. Общий алгоритм обработки файла.

12. Что называют физическим файлом?

13. Что называют логическим файлом?

14. Как связывается логический файл с физическим файлом?

15. Для чего и как открываются файлы?

16. Как закрыть файл?

17. Что называют текущим указателем?

18. Опишите операции ввода-вывода для файлов.

19. Опишите перемещение в среде типизированных файлов.

20. Назовите основные отличия текстовых файлов от типизированных файлов.

21. Назовите основные процедуры и функции, предназначенные для работы с текстовыми файлами.

Практическая работа №9. Применение рекурсивных функций.
Цель:

– научиться решать задачи с использованием рекурсивных функций и процедур

Краткие теоретические сведения

Если поставить два зеркала напротив друг друга и между ними поместить предмет, то получится бесконечное множество изображении, причем каждое из них содержит свое собственное. Любое из этих изображений можно рассматривать как рекурсивный объект, который частично состоит или определяется с помощью самого себя. Рекурсивные объекты обладают несколькими свойствами: — простотой построения;

— несхожестью конечного результата с начальными данными;

— внутренним самоподобием.

В математике встречаются рекурсивные определения, позволяющие описать объекты через самих себя. К таким определениям относится, например, определение натурального числа:

1) единица есть натуральное число;

2) число, следующее за натуральным (т. е. больше его на единицу), есть натуральное число.

Определение, которое задает некоторый объект в терминах более простого случая этого же объекта, называется рекурсивным определением.

Мощность рекурсивного определения заключается в том, что оно позволяет с помощью конечного высказывания определить бесконечное множество объектов. Как и цикл, рекурсивное определение содержит повторения, но каждый раз при этом используются новые данные, т. е. повторения не являются явными.

Рекурсия — это способ описания функций или процессов через самих себя.

Процесс может быть описан некоторым алгоритмом, называемым в данном случае рекурсивным. В нем выделяется два этапа выполнения:

1) «погружение» алгоритма в себя, т. е. применение определения «в обратную сторону», пока не будет найдено начальное определение, не являющееся рекурсивным;

2) последовательное построение от начального определения до определения с введенным в алгоритм значением.

Примеры рекурсивных алгоритмов, часто оформляемых в виде процедур и функций.

1. Наиболее распространенным рекурсивным определением является определение факториала (нерекурсивное вычисление факториала приведено в примере Р9): (*a*) 1! = 1, *(b) n* > 1, *n: = n*(n* - 1)!

На основе этого определения можно записать программу вычисления факториала, использующую рекурсивную функцию.

```
program P25;
var n, y: integer;
function F (x: integer): integer; {описание рекурсивной функции}
begin
        if x=1
        then F: = 1 {вызов для начального определения}
        else F: = x * F (x - 1) {вызов для предыдущего определения}
end; {конец описания функции}
begin {начало главной программы}
        readln (n);
        Y: = F (n); {вызов функции в главной программе}
        write (n, '! = ', Y)
```

end. *{конец главной программы}*

Выполним программу Р25 для *n* = 4. Рекурсивная функция будет работать следующим образом (при вызове функции значение n присваивается переменной *x).* Сначала осуществляется «погружение», работает оператор ветви **else** условного оператора:

1-й шаг: *x* = 4, *x*-1 = 3, выполняется промежуточное вычисление 4! = 4 * 3!

2-й шаг: *x* = 3, *x*-1 = 2, выполняется промежуточное вычисление 3! = 3 * 2!

3-й шаг: *x* = 2, *x*-1 = 1, выполняется промежуточное вычисление 2! = 2 * 1!

4-й шаг (последний): 1! = 1 по начальному определению, работает оператор F: = 1 ветви **then** условного оператора.

Следующий этап выполнения рекурсивного алгоритма — построение «прямого» определения, от начального до получения результата с исходными для алгоритма данными (числом 4). При этом осуществляется подстановка предыдущих вычислений (более поздних шагов) в более ранние:

5-й шаг: 2! = 2 * 1 = 2

6-й шаг: 3! = 3 * 2 = 6

7-й шаг: 4! = 4 * 6 = 24 — получен результат, он возвращается в главную программу и присваивается переменной Y.

2**. Вычисление степени с натуральным показателем** можно определить рекурсивно:

(a) $x^0 = 1$

(b) $k > 0$: $x^k = x * x^{k-1}$

Этому определению соответствует рекурсивная функция **power (k, x).** Программа имеет вид:

```
program P26;
var y: real; n: Integer;
function power (k: integer; x: real): real; {описание рекурсивной функции} begin
   if k=0
   then power: = 1 {начальное определение}
   else power: = x * power (k - 1, x) {рекурсивное определение}
end; {конец описания функции}
begin {начало главной программы}
   write (‘ основание степени x = ‘);
   readln (y);
   write (‘показатель степени k = ‘);
   readln (n);
   write (‘x в степени k’, power(n, y)) {вызов функции и печать результата} end. {конец
главной программы}
```

3**. Вычисление чисел Фибоначчи**. Итальянский математик Фибоначчи придумал последовательность натуральных чисел: 1, 1, 2, 3, 5, 8, 13, ... Первые два члена последовательности равны единице, а каждый, начиная с третьего, равен сумме двух предыдущих. Для чисел Фибоначчи верно соотношение:

$F_k = F_k + F_{k-2}$

Рекурсивная функция получения значения n-го числа Фибоначчи имеет вид:

```
function Fib (n: integer): integer;
begin
   if k<3
   then Fib: = 1
   else Fib: = Fib (n - 1) + Fib (n - 2)
end;
```

Для чисел Фибоначчи используется следующее рекурсивное определение:

(a)*n*= 1, *n*=2: fib(*n*) = 1

(b) n > 2: fib *(n)* = fib (*n* - 2) + fib *(n - 1)*

Для того чтобы определить fib (6), применяя данное рекурсивное определение, надо вычислить:

fib (6) = fib(4) + fib(5) = fib(2) + fib(3) + fib (5) = 1 + fib(3) + fib(5) =
= 1 + fib(l) + fib(2) + fib(5) = 1 + 1 + 1 + fib(5) = 3 + fib(3) + fib(4) =
= 3 + fib(l) + fib (2) + fib(4) = 3 + 1 + 1 + fib(4) = 5 + fib(2) + fib(3) =
= 5 + 1 + fib(l) + fib(2) = = 6+1+1=8

Количество действий в данных вычислениях с использованием рекурсивного определения чисел Фибоначчи резко возрастает, потому что это определение ссылается само на себя дважды. При вычислении факториала количество действий при выполнении программы с рекурсивной функцией и примера Е9 одинаково.

4. **Рекурсивные алгоритмы** могут быть оформлены и **в виде процедур**. Примером такой процедуры является решение задачи о Ханойских башнях.

Эта задача связана с легендой о том, что в одном из восточных храмов находится бронзовая плита с тремя алмазными стержнями. На один из них при сотворении мира нанизали 64 диска из чистого золота так, как показано на рисунке 7. Жрецы должны переносить диски с одного стержня на другой, следуя следующим законам:

1) диски можно перемещать только по одному;

2) нельзя класть больший диск на меньший.

Согласно легенде, когда все диски будут перенесены с одного стержня на другой, наступит конец света.

Решение этой задачи реализовано в виде рекурсивного алгоритма, который представляет собой инструкцию по перемещению дисков. Сформулируем задачу, присвоив имена стержням *(A, B, C)* и номера дискам (от 1 до *n*). Надо перенести диски со стержня *A* на стержень С, используя *B* как вспомогательный и следуя приведенным выше правилам переноса дисков.

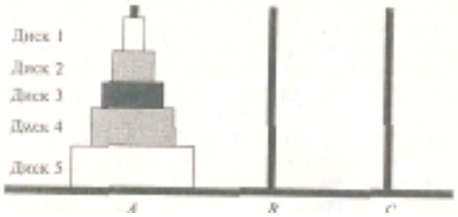


Рис. 7. *Ханойские башни*

Алгоритм на естественном языке имеет вид:

1) если *n* = 0, остановиться;

2) переместить верхние *n* - 1 дисков со стержня *A* на стержень *B,* используя стержень С как вспомогательный;

3) переместить оставшийся диск со стержня *A* на стержень С;

4) переместить *n* - 1 дисков со стержня *B* на стержень С, используя стержень *A* как вспомогательный.

В процедуре появляется новый тип данных — **char,** значение этого типа — один символ, заключенный в апострофы.

```
program P27;
var k: integer;
procedure Hanoy (n: integer; One, Two, Three: char);
begin
    if n > 0 then
    begin
          Hanoy (n - 1, One, Three, Two);
          write (‘ переместить диск’, n, ‘со стержня ‘, One, ‘на стержень ‘, Two);
          Hanoy (n - 1, Two, One, Three)
  end;
end;
begin
    write (‘введите количество дисков’);
    readln (k):
    Hanoy (k,’A’,’B’,’C’)
end.
```

Результат работы программы для *n* = 3 — это инструкция из 7 пунктов *(n* = 4 — инструкция из 15 пунктов):

переместить диск 1 со стержня *A* на стержень С

переместить диск 2 со стержня *A* на стержень *B*
переместить диск 1 со стержня C на стержень *B*
переместить диск 3 со стержня *A* на стержень C
переместить диск 1 со стержня *B* на стержень *A*
переместить диск 2 со стержня *B* на стержень C
переместить диск 1 со стержня *A* на стержень C

Практические задания

Задание.

№	Условие задачи
1	Написать функцию, которая находит цифровой корень целого числа.
2	Найти сумму цифр заданного натурального числа.
3	Найти количество цифр в заданном натуральном числе.
4	Составить программу вычисления суммы четных факториалов. (n-четное, n≤10)
5	Описать рекурсивную логическую функцию Simm(S,I,J), проверяющую, является ли симметричной часть строки S, начинающаяся i-м и заканчивающаяся j-м ее элементом.
6	Составить программу вычисления суммы нечетных факториалов. (n-четное, n≤10)
7	Составить программу сортировки массива целых чисел.
8	Составить программу вычисления НОД двух натуральных чисел.
9	Составить программу нахождения числа, которое образуется из данного натурального числа при записи его цифр в обратном порядке.(173⇒371)
10	Составить программу перевода данного натурального числа в p-ичную систему счисления(2≤p≤9)
11	Дан прямоугольник со сторонами A и B, где A,B- натуральные числа. Начнем отсекать от него квадраты. Сколько квадратов можно отсечь, если каждый раз отсекается самый большой квадрат.
12	Поиска значений в упорядоченном списке.
13	Найти сумму 1/1+1/2+1/3+1/4+…+1/n,основываясь на рекурсии.(сумма k слагаемых равна сумме (k-1) слагаемых плюс k-е слагаемое).
14	Напишите главную программу для вычисления *n*-го числа Фибоначчи. Почему использовать рекурсивный алгоритм вычисления *n*-го числа Фибоначчи невыгодно?
15	Определите рекурсивно умножение как сложение и деление как вычитание и оформите алгоритмы в виде рекурсивных функций с вызовом из главных программ.
16	Составить программу сортировки массива целых чисел.
17	Составить программу вычисления НОД двух натуральных чисел.
18	Составить программу нахождения числа, которое образуется из данного натурального числа при записи его цифр в обратном порядке.(173⇒371)
19	Составить программу перевода данного натурального числа в p-ичную систему счисления(2≤p≤9)
20	Дан прямоугольник со сторонами A и B, где A,B- натуральные числа. Начнем отсекать от

	него квадраты. Сколько квадратов можно отсечь, если каждый раз отсекается самый большой квадрат.
21	Поиска значений в упорядоченном списке.
22	Найти сумму 1/1+1/2+1/3+1/4+…+1/n,основываясь на рекурсии.(сумма k слагаемых равна сумме (k-1) слагаемых плюс k-е слагаемое).

Контрольные вопросы:

1. Что такое рекурсивный объект и каковы его свойства?
2. Приведите примеры рекурсивного определения в математике.
3. Что такое рекурсия?
4. Как выполняется рекурсивный алгоритм?
5. Поясните выполнения рекурсивной функции вычисления степени с натуральным показателем.

Практическая работа №10. Программирование модуля.

Цель:

– усвоить знание основ модульного программирования; освоить способы создания и применения модулей.

Краткие теоретические сведения

Модульное программирование основано на понятии **модуля** – **программы** или функционально завершенного **фрагмента** программы.

Модуль характеризуют:

1) один вход и один выход. На входе программный модуль получает определенный набор исходных данных, выполняет их обработку и возвращает один набор выходных данных;

2) функциональная завершенность. Модуль выполняет набор определенных операций для реализации каждой отдельной функции, достаточных для завершения начатой обработки данных;

3) логическая независимость. Результат работы данного фрагмента программы не зависит от работы других модулей;

4) слабые информационные связи с другими программными модулями. Обмен информацией между отдельными модулями должен быть минимален;

5) размер и сложность программного элемента в разумных рамках.

С помощью модулей решаются различные профессиональные задачи обработки данных разного типа.

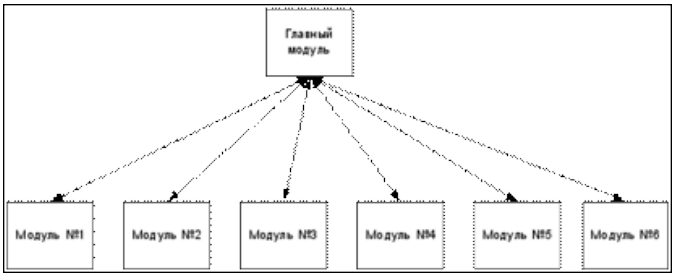


Рис. 1. Схема подключения модулей

Предлагается использовать следующие **характеристики программного модуля** для оценки его приемлемости: **размер** модуля, **прочность** модуля, **сцепление** с другими модулями и **рутинность** модуля.

Размер модуля измеряется числом содержащихся в нем операторов. Модуль не должен быть слишком маленьким или слишком большим.

Прочность модуля – это мера его внутренних связей. Чем выше прочность модуля, тем больше связей скрыто от внешней по отношению к нему части программы и, следовательно, тем проще сама программа.

Сцепление модуля – это мера его зависимости по способу передачи данных от других модулей. Чем слабее сцепление модуля с другими модулями, тем сильнее его независимость от других модулей.

Рутинность модуля — это его независимость от предыстории обращений к нему. Модуль будем называть рутинным, если результат обращения к нему зависит только от значений его параметров и не зависит от результатов предыдущих обращений к нему.

С помощью модулей решаются различные профессиональные задачи обработки данных разного типа.

Практические задания

Задача 1. Разработать и реализовать программу ввода данных, вычисления функций при помощи модулей, вывода результатов.

Программа работы

1. Составить список функций и соответствующих им модулей для реализации в программе.
2. Разработать интерфейс программы ввода, обработки и вывода данных.
3. Написать программный код для каждого модуля.
4. Реализовать интерфейс и программный код в среде визуальной разработки программ.
5. Провести тестирование и отладку программы.
6. Нарисовать интерфейс программы со спецификацией и записать программный код с комментариями в отчете по работе.
7. Записать несколько вариантов тестирования программы.
8. Провести тестирование исполняемого файла.

Задача 2. Разработать и реализовать программу вычисления значения сложного выражения при помощи шагов:

1. Ввода данных, вычисления функций при помощи первого модуля, вывода результатов;
2. Чтения выходных данных первого модуля, вычисления функций при помощи второго модуля, вывода результатов;
3. Чтения выходных данных второго модуля, вычисления функций при помощи третьего модуля, вывода результатов.

Программа работы

1. Составить список функций и соответствующих им модулей для реализации в программе.
2. Разработать интерфейс программы ввода, обработки и вывода данных.
3. Написать программный код для каждого модуля.
4. Реализовать интерфейс и программный код в среде визуальной разработки программ.
5. Провести тестирование и отладку программы.
6. Нарисовать интерфейс программы со спецификацией и записать программный код с комментариями в отчете по работе.
7. Записать несколько вариантов тестирования программы.
8. Провести тестирование исполняемого файла.

Контрольные вопросы:

1. Что называют модулем в контексте модульного программирования?
2. Сколько входов и выходов имеет один модуль?
3. Существуют ли типы данных, которые невозможно обработать при помощи модулей?
4. Зависит ли результат работы модуля от работы других модулей?
5. Чем должны быть ограничены размер и сложность модуля?
6. Что используют для оценки приемлемости программного модуля?
7. Чем определяется размер модуля?
8. Какую характеристику называют прочностью модуля?
9. Как зависит независимость модуля от его сцепления?
10. Какой модуль называют рутинным?

Практическая работа №11. Использование указателей для организации связанных списков.

Цель:

– изучение порядка действий при вычислении выражений;

– приобретение навыков в записи выражений на языке Паскаль ииспользовании стандартных функций.

Практические задания

1. Найти значение функции Y(X) при заданном X. Используя стандартные функции, вычислить Y11=[Y], где [ ] означает целую часть числа; Y22=[Y0.5].
2. Записать выражение, зависящее от координат точки X1 и Y1принимающее значение TRUE, если точка принадлежит заштрихованной области, и FALSE, если не принадлежит. Для заданной точки вычислить это выражение и результат выдать на печать.

1. Функции, отсутствующие в списке стандартных функций языка Паскаль, следует выразить через имеющиеся.
2. Печать значения выражения в данной точке организовать,используя запись выражения в операторе WRITELN.
3. Пример программы, которая вычисляет Y(X)=3-x+1SIN(X) при заданном X, печатает Y11,Y22 и проверяет принадлежность точки скоординатами (X1,Y1) заштрихованной области. (рис. 1) Исходные данные: X=-1.5, X1=0.5,Y1=1.2.

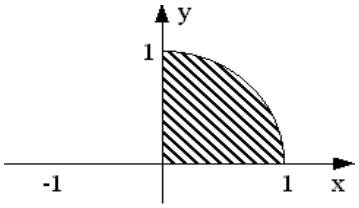


Рис. 1.

Варианты задания

1 а) $Y=2^{-x} + |x|$ при X=4.741;
б) координаты используемой точки: (0.5;0.5).Область изображена на рис. 2.

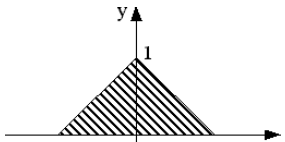


Рис. 2.

2 а) $Y=e^{x} –\sin(x)$ при X=2.312;
б) координаты используемой точки: (1.5;0.5).Область изображена на рис. 3

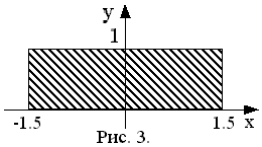


Рис. 3.

3 а) Y= |x-1|+sin(x) при X=12.7409;
б) координаты используемой точки: (0.2;0.9).Область изображена на рис. 4.

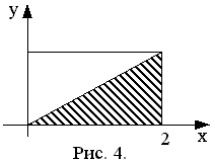


Рис. 4.

4 а) Y=xcos(x)+sin(3x) при X=30.872;
б) координаты используемой точки: (0.75;-0.3).Область изображена на рис. 5

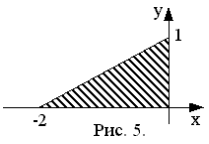


Рис. 5.

5 а) Y=tg(x)+|x| при X=-2.6312;

б) координаты используемой точки: (0.2;0.45).Область изображена на рис. 6.

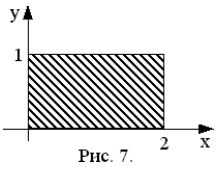


Рис. 7.

6 а) $Y=1+1/x+1/x^2$ при X=-0.387;
б) координаты используемой точки: (0.5;-2.5).Область изображена на рис. 7.

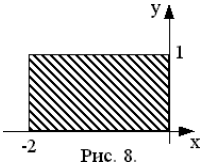


Рис. 8.

7 а) Y=ch|x+1| при X=4.352;
б) координаты используемой точки: (0.0;0.0).Область изображена на рис. 8.

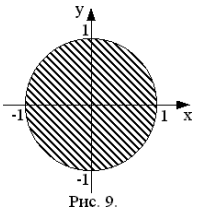


Рис. 9.

8 а) $Y=\arcsin(x)+x^2$ при X=0.112;
б) координаты используемой точки: (1.0;1.5).Область изображена на рис. 9.

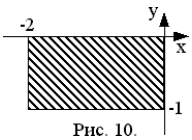


Рис. 10.

9 а) Y=sin(x)arctg(x) при X=-0.7129;
б) координаты используемой точки: (-0.5;0.9).Область изображена на рис. 10.

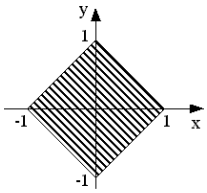


Рис. 11.

10 а) Y=5arctg(x) при X=-4.4172
б) координаты используемой точки: (1.5;0.0).Область изображена на рис. 11.

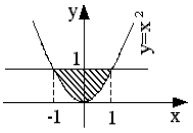


Рис. 12.

11 а) Y=ch|x+2| при X=3.123;
б) координаты используемой точки: (0.3;0.2).Область изображена на рис. 12.

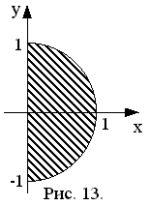


Рис. 13.

12 а) Y=arccos|x+1| при X=0.345;
б) координаты используемой точки: (-0.5;0.3).Область изображена на рис. 13.

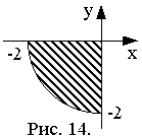


Рис. 14.

13 а) $Y=2^{-x}+2+\ln|x|$ при X=-1.521;
б) координаты используемой точки: (-0.5;-0.5).Область изображена на рис. 14.

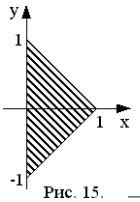


Рис. 15.

14 а) $Y=\ln(x+1)+x^3$ при X=2.571;
б) координаты используемой точки: (1.5;1.5).Область изображена на рис. 15.

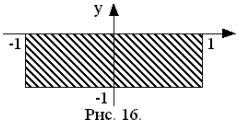


Рис. 16.

15 а) $Y=x^2\cos(x)+\sin^2 x$ при X=31.871

б) координаты используемой точки: (0.2;-0.5).Область изображена на рис. 16.

16 а) Y=6arcctg(x) при X=-2.237;
б) координаты используемой точки: (1.5;0.5).Область изображена на рис. 17.

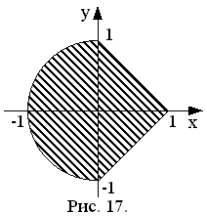


Рис. 17.

17 а) Y=ch|x|+sh|x| при X=-4.289;
б) координаты используемой точки: (2.5;-0.5).Область изображена на рис. 18.

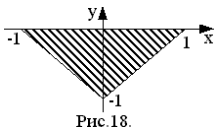


Рис.18.

18 а) Y=$\cos^3 x+\cos^2 x+\cos(x)$ при X=1.324;
б) координаты используемой точки: (0.1;0.2).Область изображена на рис. 19.

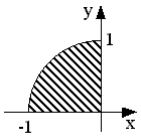


рис. 19.

19 а) Y=e^{-x} x при X=2.125;
б) координаты используемой точки: (0.6;0.2).Область изображена на рис. 20.

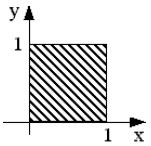


Рис.20.

20 а) Y=1+$\ln^2$ x при X=0.4371;
б) координаты используемой точки: (-0.6;0.5).Область изображена на рис. 21.

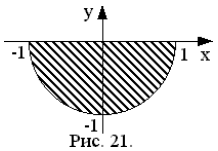


Рис. 21.

21 а) Y=1+e^x +x^3 при X=0.581;
б) координаты используемой точки: (0.4;-0.2).Область изображена на рис. 22.

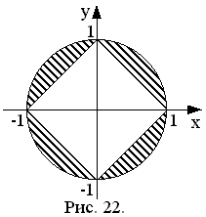


Рис. 22.

22 а) Y=1/(1+sinx) при X=1.012;
б) координаты используемой точки: (-0.7;0.2).Область изображена на рис. 23.

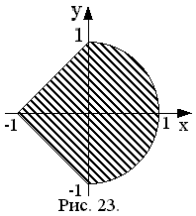


Рис. 23.

23 а) Y=x+lnx при X=3.531;
б) координаты используемой точки: (1.5;0.2).Область изображена на рис. 24.

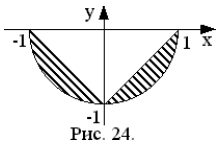


Рис. 24.

24 а) Y=e^{-x} ch(x) при X=3.521;
б) координаты используемой точки: (2.0;-0.5).Область изображена на рис. 25.

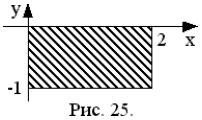


Рис. 25.

25 а) Y=cos(x)e^{-x} при X=1.578;
б) координаты используемой точки: (3.5;-1.5).Область изображена на рис. 26.

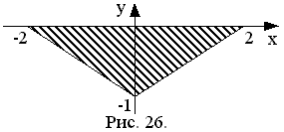


Рис. 26.

26 а) $Y=\sin(x^2+1)$ при $X=1.212$;

б) координаты используемой точки: (-2.5;-1.5).Область изображена на рис. 27.

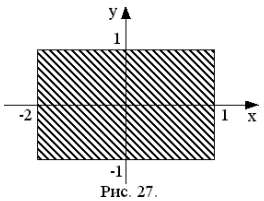


Рис. 27.

27 а) $Y=x^2-\cos(x)$ при $X=0.2314$;

б) координаты используемой точки: (0.2;-0.5).Область изображена на рис. 28.

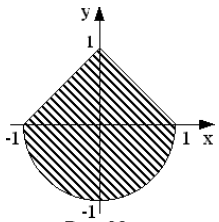


Рис. 28.

28 а) $Y=x(1-\cos(x))$ при $X=0.6327$;

б) координаты используемой точки: (0.1;0.9).Область изображена на рис. 29.

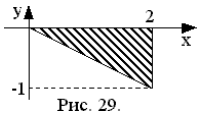


Рис. 29.

29 а) $Y=\sin(2x+1)\ln|x|$ при $X=0.435$;

б) координаты используемой точки: (0.3;-0.3).Область изображена на рис. 30.

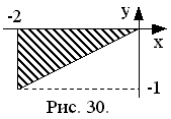


Рис. 30.

30 а) $Y=0.5+sh^2(x)$ при $X=2.381$;

б) координаты используемой точки: (-0.2;-2.0).Область изображена на рис. 31.

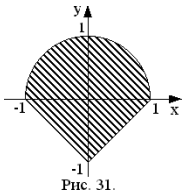


Рис. 31.

Контрольные вопросы:

1. Что представляют собой связанные списки и какие виды связанных списков вы знаете?
2. К каким структурам данным относятся связанные списки?
3. С помощью какой структуры данных можно наиболее эффективно решить задачу сортировки и почему?
4. Что представляет собой элемент динамической структуры данных?
5. Какие операции можно выполнять над списками?

Практическая работа №12. Создание проекта с использованием компонентов для работы с текстом.

Цель:

- ознакомление с оператором варианта и перечисляемыми типами;
- получение навыков в организации ввода/вывода значенийпеременных перечислимых типов данных.

Практические задания

Задание 1.

1. Задание может быть выполнено двумя способами:

а) номер квартала, дата и день недели задаются непосредственно в программе в виде констант;

б) исходная дата и день надели вводится при помощи оператора варианта.

2. Учесть переход на новую неделю, новые месяц, год, квартал, високосный или невисокосный год.

3. Пример программы, которая по заданной дате (число, месяц, год,квартал) определяет дату следующего дня.

Задание2.

Вариант 1

4. Написать программу, которая считывает из текстового файла три предложения и выводит их в обратном порядке.

5. Вариант 2

6. Написать программу, которая считывает текст из файла и выводит на экран только предложения, содержащие заданное с клавиатуры слово.

7. Вариант 3

8. Написать программу, которая считывает текст из файла и выводит на экран только строки, содержащие двузначные числа.

9. Вариант 4

10. Написать программу, которая считывает английский текст из файла и выводит на экран слова, начинающиеся с гласных букв.

11. Вариант 5

12. Написать программу, которая считывает текст из файла и выводит его на экран, меняя местами каждые два соседних слова.

13. Вариант 6

14. Написать программу, которая считывает текст P13 файла и выводит на экран только предложения, не содержащие запятых.

15. Вариант 7

16. Написать программу, которая считывает текст из файла и определяет, сколько в нем слов, состоящих не более чем из четырех букв.

17. Вариант 8

18. Написать программу, которая считывает текст из файла и выводит на экран только цитаты, то есть предложения, заключенные в кавычки.

19. Вариант 9

20. Написать программу, которая считывает текст из файла и выводит на экран только предложения, состоящие из заданного количества слов.

21. Вариант 10

22. Написать программу, которая считывает английский текст из файла и выводит на экран слова текста, начинающиеся с гласных букв и оканчивающиеся гласными буквами.

23. Вариант 11

24. Написать программу, которая считывает текст из файла и выводит на экран только строки, не содержащие двузначные числа.

25. Вариант 12

26. Написать программу, которая считывает текст из файла и выводит на экран только предложения, начинающиеся с тире, перед которым могут следовать только пробельные символы.

27. Вариант 13

28. Написать программу, которая считывает английский текст из файла и выводит его на экран, заменив каждую первую букву слов, начинающихся с гласной буквы, на прописную.

29. Вариант 14

30. Написать программу, которая считывает текст из файла и выводит его на экран, заменив цифры от 0 до 9 на слова «ноль», «один», ..., «девять», начиная каждое предложение с новой строки.

31. Вариант 15

32. Написать программу, которая считывает текст из файла, находит самое длинное слово и определяет, сколько раз оно встретилось в тексте.

33. Вариант 16

34. Написать программу, которая считывает текст из файла и выводит на экран сначала вопросительные, а затем восклицательные предложения.

35. Вариант 17

36. Написать программу, которая считывает текст из файла и выводит его на экран, после каждого предложения добавляя, сколько раз встретилось в нем заданное с клавиатуры слово.

37. Вариант 18

38. Написать программу, которая считывает текст из файла и выводит на экран все его предложения в обратном порядке.

39. Вариант 19

40. Написать программу, которая считывает текст из файла и выводит на экран сначала предложения, начинающиеся с однобуквенных слов, а затем все остальные.

41. Вариант 20

42. Написать программу, которая считывает текст из файла и выводит на экран предложения, содержащие максимальное количество знаков пунктуации.

Задание 3.

Варианты задания

Номер варианта	Номер квартала	Дата и день недели, год
1	1	27 февраля пятница 1991
2	2	12 апреля воскресенье 1810
3	3	26 августа среда 1992
4	4	30 ноября понедельник 1980
5	1	23 марта понедельник 1882
6	2	9 мая суббота 1993
7	3	1 сентября вторник 1992

8	4	9 декабря среда 1998
9	1	12 января понедельник 2000
10	2	1 мая пятница 1930
11	3	30 августа воскресенье 2000
12	4	7 октября среда 1996
13	1	23 января пятница 1989
14	2	11 мая понедельник 1984
15	3	22 сентября вторник 1982
16	4	16 ноября понедельник 1962
17	1	29 марта воскресенье 1956
18	2	1 июня понедельник 1934
19	3	13 августа четверг 1975
20	4	21 декабря понедельник 1974
21	1	31 января понедельник 1962
22	2	21 апреля воскресенье 1953
23	3	31 августа вторник 1988
24	4	31 декабря понедельник 1991
25	1	28 февраля среда 1980
26	2	30 июня пятница 1965
27	3	1 сентября воскресенье 1997
28	4	1 ноября понедельник 1800
29	1	29 февраля воскресенье 1981
30	2	31 мая воскресенье 1992

Контрольные вопросы:

1. Что такое файл?
2. В каком включаемом файле объявляются потоки *cin* и *coutl*
3. Для чего используется функция fail?
4. Что такое функция EOF и для чего она используется?
5. Каким образом можно упростить ввод имен файлов?

Практическая работа №13. Создание процедур на основе событий.
Цель:

– получение навыков в использовании оператора цикла спараметром;
– знакомство с методами оптимизации программ.

Практические задания

Составить программу вычисления значений функции F(X) на отрезке[A,B] в точках Xi=A+iH, где H=(B-A)/M, M – заданное целое число.

1. Для задания значений и соответствующих значений функции следует использовать простые переменные.
2. Значение шага H должно вычисляться один раз.
3. При изменении значения аргумента X использовать оператор присваивания X:=X+H, а не оператор с использованием операции умножения X:=A+I*H, что существенно сокращает время выполнения программы.

Варианты задания

Номер варианта	Функция	Параметры		M
		A	B	
1	x-sin(x)	0	π/2	10
2	sin(x)	π /4	π/2	15
3	cos(x)	π /3	2π/3	20
4	tg(x)	0	π/4	10
5	ctg(x)	π/4	π/2	15
6	arcsin(x)	0	1	20
7	arcos(x)	0.5	1	10
8	arctg(x)	2	7	15
9	sin(x)-cos(x)	0	π/2	20
10	xsin(x)	0	3π	10
11	sin(1/x)	π/8	2/π	15
12	cos(1/x)	π/4	4/π	20
13	$\sin(x^2)$	π/6	2π/3	10
14	$\cos(x^2)$	π/3	3π/2	15
15	sin(x)+cos(x)	0	π/4	20
16	cos(x)+ctg(x)	π/4	π/2	10
17	tg(x/2)	0	2π/3	15
18	tg(x/2)+cos(x)	π/2	π	20
19	ctg(/3)+sin(x)	π/4	π/2	10
20	sin(x/4)/2	π/2	π	15

Контрольные вопросы:

1. В каких случаях используется таймер? Что определяет свойство Interval?
2. Как остановить работу таймера?
3. Как определить код нажатой клавиши?
4. На какие события реагирует компонент Shape?

Практическая работа №14. Разработка функциональной схемы работы приложения.
Цель:

– получение практических навыков в работе с массивами;
– знакомство с алгоритмами упорядочения.

Практические задания

1. Исходные данные должны включать и положительные числа, и отрицательные, и нули.
2. Пример программы. Дан одномерный массив найти в нём максимальный элемент.

Варианты задания

1. В массиве из 20 целых чисел найти наименьший элемент ипоменять местами первым элементом.

2. В массиве из 10 целых чисел найти наименьший элемент ипоменять его местами с последним элементом.

3. В массиве из 15 вещественных чисел найти наибольшийэлемент и поменять его местами с последним элементом,

4. В массиве из 25 вещественных чисел найти наименьшийэлемент и поменять его местами с первым элементом.

5. Упорядочить по не убыванию массив, содержащий 20целых чисел.

6. Упорядочить по не возрастанию массив, содержащий 25вещественных чисел.

7. Упорядочить по не убыванию массив, содержащий 15 вещественных чисел.

8. Упорядочить по не возрастанию массив, содержащий 25вещественных чисел.

9. Вычислить сумму и количество элементов массива X(100)для 0<=X<=1

10. Вычислить среднее арифметическое значение элементаположительного массива A(80).

11. Переписать элементы массива X(70) в массив Y и подсчитать их количество, (-1 <=X<=1).

12. Определить максимальный элемент массива B(50) и егопорядковый номер.

13. Вычислить минимальный элемент массива C(40) и егопорядковый номер.

14. Найти максимальный и минимальный элементы массиваD(80) и поменять их местами,

15. Вычислить среднее геометрическое элементов массива Y(20).

16. Расположить в массиве R сначала, положительные, а затем отрицательные элементы массива Z (30).

17. В массиве N(50) определить сумму и количество элементов массива, кратным трём.

18. Вычислить сумму и количество элементов массива X(N).

19. Найти среднее геометрическое элементов массива A, гдеa>0, N<=40.

20. Переписать в массив Y подряд положительные элементымассива X(N), где X>0, N<=40.

21. Определить максимальный элемент массива B(k) и егопорядковый номер, где X<0, K<=40.

22. Определить минимальный элемент массива C(k) и егопорядковый номер, где -1 <=X<=1, K<=20.

23. В массиве F(45) подчитать количество положительных иотрицательных элементов.

24. В массиве Y(30) найти максимальный элемент и заменить его на 0.

25. В массиве A(20) найти последний отрицательный элемент и вывести его номер.

26. В массиве C(15) найти последний чётный элемент и по- ставить его на место первого элемента.

27. В массиве B(25) на место всех положительных элементов записать нули, а все отрицательные элементы сложить. На печать вывести новый массив и сумму отрицательных элементов.

28. В массиве A(10) найти минимальный и максимальный элементы, соответственно поменять с первым и последним элементами массива.

29. Из массива X(20) выбрать все чётные элементы и переписать их в массив A.

30. Из массива Y(l5) выбрать все нечётные элементы и переписать их в тот же массив.

Контрольные вопросы:

1. Какие окна присутствуют по умолчании на экране в момент начала работы над новым проектом в Delphi и каковы их функции?
2. Что такое Properties и Events в окне инспектора объектов?
3. В чем разница между свойствами Caption и Name?
4. Что означают значок «+» перед названием свойства в окне инспектора объектов и кнопка с многоточием в строке свойства?
5. Какие файлы создает Delphi при работе с проектом? Каково их назначение? Где они сохраняются?

Практическая работа №15. Разработка оконного приложения с несколькими формами.
Цель:

- получение навыков в написании программ с использованием процедур.

Практические задания

1. Если в качестве исходной информации в процедуру передаётся массив, то его следует передавать по ссылке для экономии памяти, так как в этом случае при вызове процедуры не образуется локальный массив.
2. Несмотря на то что обрабатываемые массивы имеют разную длину, они описываются в программе как массив одного и того же типа, так как при обращении к процедуре типы соответствующих формальных и фактических параметров должны совпадать.
3. Приведённая ниже программа.

Постановка

Переписать положительные элементы массивов X(100) и Y(100) вмассив Z подряд. Запись положительных элементов в массив осуществить в процедуре.

Варианты задания

1. Заданы 4 вектора X(3), Y(3), Z(4), P(4). Логическойпеременной A присвоить значение TRUE, если скалярное произведение векторов X и Y больше скалярного произведения Z и P, иначе A присвоить FALSE. Вычисление скалярного произведения оформить в виде процедуры.

2. Заданы две матрицы A(3,3) и B(3,3). Проверить, является ли произведение этих матриц перестановочным, т.е. проверить равенство AB=BA. Если ответ положительный, напечатать “AB=BA”, иначе напечатать "Произведение не перестановочно". Вычисление произведения двух матриц оформить в виде процедуры.

3. Заданы два вектора X(4) и Y(5). Логической переменной A присвоить значение TRUE, если длина вектора X больше длины вектора Y и присвоить FALSE в противном случае. Вычисление длинывектора оформить в виде процедуры.

4. Заданы две матрицы A(3,3) и B(4,4). Выяснить и напечатать, сколько из них являются симметричными (0.1 и 2). Матрица называется симметричной, если транспонированная матрица равна исходной. Транспонирование матриц оформить в виде процедуры.

5. Заданы два массива A(4) и B(6). Переменной C присвоить значение 1, если максимальный элемент массива A больше максимального элемента массива B; 0, если максимальные элементы массивов равны; и -1, если максимальный элемент массива A меньше максимального элемента массива B. Поиск максимального элемента оформить в виде процедуры.

6. Четыре точки заданы своими координатами X(2), Y(2), Z(2), P(2). Выяснить и напечатать какие из них находятся намаксимальном расстоянии друг от друга. Вывести значение этого расстояния. Вычисление расстояния между двумя точками оформить ввиде процедуры.

7. Четыре точки заданы своими координатами X(3), Y(3), Z(3), P(3). Выяснить и напечатать какие из них находятся намаксимальном расстоянии друг от друга. Вывести значение этого расстояния. Вычисление расстояния между двумя точками оформить ввиде процедуры.

8. Заданы две матрицы A(3,3) и B(4,4). Построить таблицу функций $y=cx^2+p$, при x меняющемся от 0 до 1 с шагом 0.1, где c - след матрицы A, p - след матрицы B. Следом матрицы называется сумма элементов главной диагонали. Вычисление следа матрицы оформить в виде процедур.

9. Заданы три массива X(4), Y(3), Z(5). Упорядочить по возрастанию три числа

a, b, c, где a - минимальный элемент массива X,b - минимальный элемент массива Y, c минимальный элемент массива

Z. Поиск минимального элемента массива оформить в виде процедуры.

10. Заданы 2 вектора X(3), У(3) и матрица A(3,3). Найти C и B, где C - произведение вектора X на матрицу A, вектор B - произведениевектора Y на матрицу A. Вычисление произведения вектора на матрицу оформить в виде процедуры.

11. Заданы три матрицы A(3,3), B(2,2), C(3,3). Найти минимальное из 3-х чисел x, y, z, где x - след матрицы A, y - след матрицы B, z - след матрицы C. Следом матрицы называется сумма элементов главной диагонали. Вычисление следа оформить в виде процедуры.

12. Заданы стороны двух треугольников: ABC (стороны a, в, c) и PLF (стороны p, l, f). Переменной S присвоить значение -1, если площадь треугольника ABC меньше или равна площади треугольника PLF; и значение 1, если площадь треугольника ABC больше площади PLF. Площадь треугольника MNK со сторонами m, n, k вычисляется поформуле Герона w=r*(r-m)*(r-n)*(r-k), где г- полупериметр треугольника MNK. Вычисление площади треугольника оформить в виде процедуры.

13. Построить таблицу функций z=sh(x+y), где x меняется от 0 до 1 с шагом 0.2, y меняется от 2 до 3 с шагом 0.1. Гиперболический синус вычисляется по формуле: $sh(r)=(e^{r}-e^{-r})/2$. Вычисление гиперболического синуса оформить в виде процедуры.

14. Вычислить суммы положительных элементов массивов X(60), Y(60), Z(70). Вычисление суммы оформить в виде процедуры.

15. Вычислить среднее арифметическое положительных элементов для массивов A(100), B(100), C(100). Среднее арифметическое оформить в виде процедуры.

16. Подсчитать количество элементов матриц X(10,15) и Y(20,12), удовлетворяющих условиям $0\le x_{ij}\le 1$ и $0\le y_{ij}\le 1$. Количество элементов подсчитывать с помощью процедуры.

17. Подсчитать число нулевых элементов для матриц A(20,20),B(30,30). Число нулевых элементов подсчитывать в процедуре.

18. Вычислить суммы элементов нижних треугольных матриц для A(15,15) и B(20,20). Суммы вычислять в процедуре.

19. Найти наибольшие элементы и их порядковые номера массивов X(80), Y(70). Нахождение наибольших элементов оформитьв виде процедуры.

20. Переписать положительные элементы массива X(100) и Y(80) в массив Z подряд. Запись в массив Z осуществлять в процедуре.

21. Найти наименьшие элементы и номера строк и столбцов, в которых они расположены, для матриц A(10,15) и B(15,12), используя процедуру.

22. Вывести на печать элементы целочисленных матриц H(5,8)и M(10,б) кратным трём (с помощью процедуры).

23. Вычислить и запомнить количество отрицательных элементов каждого столбца для матриц A(10,10), B(15,15). Используяпроцедуру.

24. Найти среднее значения и стандартные отклонения дляэлементов массивов X(100), У(100), с помощью процедуры.

25. Вычислить суммы и количества элементов, находящихся винтервале от A до B для матриц X(10,8) и У(10,2), используя процедуру.

26. Преобра

зовать массивы X(50) и У(60), расположив в них подряд только положительные элементы. Вместо остальных элементов записать нули,используя процедуру.

27. Даны две матрицы A(12,12) и B(10,10) используяпроцедуру, организовать ввод этих матриц и подсчитать в каждой сумму положительных элементов.

28. В массивах K(10) и H(15) упорядочить элементы по возрастанию, используя

процедуру.

29. В матрицах B(10,10) и P(15,15) на место отрицательных элементов записать 1, уровне выполнить с помощью процедуры.

30. В матрицах X(40,40) и Y(100,100), подсчитать сумму положительных элементов стоящих над главной диагональю. Суммы подсчитать с помощью процедуры.

Контрольные вопросы:

1. В каких случаях используется таймер? Что определяет свойство Interval?
2. Как остановить работу таймера?
3. Как определить код нажатой клавиши?
4. На какие события реагирует компонент Shape?

Практическая работа №16. Тестирование, отладка приложения.
Цель:

- получение навыков в задании переменных множественного типаи выполнении простейших операций над ними;
- знакомство с задачами, в которых целесообразно использоватьпеременные множественных типов.

Практические задания

1. Программа должна правильно работать для произвольного набора символов. Для выполнения задания следует ознакомиться LAT. Входная строка символов может быть длиннее строки экрана терминала, при этом программа работает не с функцией EOLN, а с признаком конца строки, который задается программистом.
2. Примерная программа:

Дана непустая последовательность символов. Требуется построить и напечатать множество, элементами которых является встречающаяся в последовательности цифры от ‘5’ до ‘9’.

Варианты задания

Дана непустая последовательность символов. Требуется построить и вывести на печать множество, элементами которого является встречающиеся в последовательности:

1. цифры от ‘0’ до ‘9’;
2. буквы от ‘A’ до ‘F’ и от ‘X’ до ‘Z’;
3. буквы от ‘G’ до ‘N’ и цифры от ‘0’ до ‘9’;
4. знаки препинания;
5. буквы от ‘A’ до ‘Z’ и цифры от ‘0’ до ‘9’;
6. буквы от 'T' до 'X' и знаки препинания;
7. цифры от '5' до '9' и знаки арифметических операций;
8. знаки арифметических операций и знаки препинания;
9. цифры и знаки арифметических операций;
10. знаки препинания и буквы от ‘E’ до ‘N’;
11. знаки операций отношения;
12. цифры от ‘3’ до '9', буквы от ‘A’ до ‘F’ и знаки препинания;
13. знаки арифметических операций и операций отношения;
14. буквы от ‘F’ до ‘M’ и знаки арифметических операций;
15. знаки препинания и операций отношения;
16. скобки всех видов и цифры от '0'до ‘9’;
17. цифры от ‘1’ до ‘7’ и знакипрепинания;
18. цифры, операции отношения и буквы от ‘T’ до ‘X’;
19. согласные буквы английского алфавита;
20. гласные буквы английского алфавита;
21. все символы не являющиеся буквами английского алфавита;
22. знаки препинания и гласные буквы английского алфавита;
23. спецсимволы(@ $ % # ^ & * ~)
24. цифры, знаки арифметических операций и спецсимволы;
25. операции отношения и спецсимволы;
26. гласные английского алфавита скобки всех видов,

27. буквы английского алфавита, знаки операций;

28. спецсимволы, знаки препинания, знаки отношения;

29. скобки, знаки препинания, знаки арифметическихопераций.

Контрольные вопросы:

1. Что такое отладка программного обеспечения?
2. В каком режиме работает отладчик GDB?
3. Какие параметры командной строки использует GDB?
4. Каким образом должны быть подготовлены программы, чтобы их можно было исследовать с помощью GDB?
5. Как влияет уровень детализации отладочной информации на размер исполняемого файла?
6. Можно ли выделить отладочную информацию в отдельный файл? Если да, то каким образом?
7. Что такое «точка останова»? Сколько точек останова можно задать в процессе отладки программы? Можно ли задать условие срабатывания точки останова?
8. Какую информацию можно получить в процессе отладки программы?

Практическая работа №17. Классы ООП: виды, назначение, свойства, методы, события. Объявления класса.

Цель:

– ознакомление с возможностями организации файлов на внешних носителях в используемой ЭВМ.

– получение практических навыков работы с внешними файлами.

Практические задания

Подготовить данные об абитуриентах, поступающих в техникум. Информацию о каждом абитуриенте оформить в виде записи, содержащей следующие поля:

1. Фамилия, имя, отчество.
2. Год рождения.
3. Год окончания школы.
4. Оценки в аттестате.
5. Признак - нуждается ли в общежитии.
6. Оценки вступительных экзаменов.

Разработать программу, которая создавала файл на внешнем носителе, удаляла из внешнего файла все записи, удовлетворяющие условию, заданному в варианте, распечатать информацию, оставшуюся в файле. Добавить N записей в начало (конец) внешнего файла и распечатать записи полученного файла согласно варианту.

1. При подготовке исходных данных необходимо учесть, что выходная информация программы обработки внешнего файла должна составлять не менее одной четверти от входной.

2. Запись данных во внешний файл должна иметь структуру,аналогичную структуре созданного файла.

3. При оставлении программы обработки данных, хранящихся во внешнем файле, необходимо ознакомиться с предложенной программойFA1 (удаление) и FA2 (добавление).

<u>Постановка</u>

Разработать программу записи подготовленных данных во внешнийфайл и программу обработки созданного файла.

1. Удалить из внешнего файла записи об студентах, нуждающихсяв общежитии и получивших хотя бы одну тройку.

2. Используя внешний файл, содержащий исходные данные добавить N записей и распечатать список абитуриентов сдавших экзамены с двумя оценками 4 и 5.

Варианты задания

Из внешнего файла, содержащего исходные данные, удалить все данные, соответствующие следующим заданиям, а затем добавить несколькозаписей:

1. Анкетные данные абитуриентов- получивших одни пятёрки.
2. Анкетные данные абитуриентов, получивших 4 и 5.
3. Анкетные данные абитуриентов, получивших одну оценку 3 за всё время экзаменов.
4. Анкетные данные абитуриентов, получивших за последний

экзамен оценку 2,

5. Анкетные данные абитуриентов, получивших за первый экзамен оценку 5.

6. Анкетные данные абитуриентов, получивших за все экзамены одну оценку 4.

7. Список абитуриентов, фамилии которых начинаются с буквы А, и их оценки за экзамены.

8. Список абитуриентов, фамилии которых начинаются с Б, и их даты рождения.

9. Оценки в последний экзамен абитуриентов, фамилии которых начинаются с буквы В и Г.

10. Фамилии и даты рождения абитуриентов не получивших ни одной оценки 3 за все экзамены

11. Упорядочить список абитуриентов но среднему баллу и распечатать его.

12. Упорядочить списокабитуриентов по среднему баллу аттестатов и распечатать его.

13. Вычислить средний балл группы и распечатать список абитуриентов, имеющих средний балл выше среднего балла группы.

14. Вычислить средник балл группы и распечатать списокабитуриентов, имеющих средний балл ниже среднего балла группы.

15. Вычислить средний балл группы и распечатать список абитуриентов, имеющих средний балл равный среднему баллу группы.

16. Вычислить средний балл: группы за последний экзамен и распечатать список абитуриентов, имеющих средний балл, равный среднему баллу группы.

17. Упорядочить списокабитуриентовпо году рождения распечатать его.

18. Распечатать списокабитуриента, упорядоченный по алфавиту и среднему баллу.

19. Распечатать список абитуриентов, упорядоченный по году окончания школы и среднем баллу.

20. Распечатать список абитуриентов, получивших одни пятерки, упорядоченный по году рождения.

21. Распечатать список абитуриентов, упорядоченный по дате рождения и нуждающихся в общежитии.

22. Распечатать списокабитуриентов, нуждающихся в общежитии и получивших на экзаменах четвёрки.

23. Список абитуриентов отличников и одного возраста.

24. Список абитуриентов одного года рождения. Год ввести склавиатуры.

Контрольные вопросы:

1. Что такое наследование?
2. Для каких целей применяют наследование?
3. Какие члены класса наследуются?
4. Какие члены класса не наследуются?
5. Каков порядок вызова конструкторов при наследовании?

Практическая работа №18. Создание наследованного класса.
Цель:

- научиться использовать операторы циклов при составлении программ на языке Паскаль; составлять блок-схему циклической структуры.

Краткие теоретические сведения

Операторы цикла используются для многократного повторения аналогичных вычислений.

Для организации цикла в Паскале имеются три различных оператора.

1. Регулярный оператор For

For <параметр цикла>:=<начальное значение> to <конечное значение> do S;

S- простой или составной оператор.

Инструкция for используется для организации циклов с фиксированным, определяемым во время разработки программы, числом повторений; количество повторений цикла определяется начальным переменной-счетчика; переменная-счетчик должна быть целого типа (integer).

При каждом прохождении цикла < параметр цикла >, начиная с <начального значения>, увеличивается на единицу. Цикл выполняется, пока <параметр цикла> не станет больше <конечного значения>.

Другой вариант записи оператора For:

For <параметр цикла >:=< начальное значение> downto <конечное значение> do S;

В этом случае при каждом прохождении цикла <параметр цикла> уменьшается на единицу от <начального значения> до <конечного значения>.

2 Оператор цикла While с проверкой предусловия:

While <условие> do S; {Пока выполняется условие, делать}

Цикл выполняется, пока условие истинно (true).

3 Оператор цикла Repeat с проверкой постусловия:

Repeat S until <условие>; {Выполнять до тех пор, пока не будет выполнено условие}

Цикл выполняется, пока условие ложно (false).

Рассмотрим примеры.

Постановка задачи. Найти сумму 5 целых чисел от 1 до 5. Написать программы для определения суммы с помощью трех рассмотренных операторов цикла.

Структограмма и программа приведены в таблице 1

Таблица 1. Операторы цикла

Цикл **For**	**While**	**Repeat**	
3.1. Описание S,i; S:=0; For i:=1 to 5 do S:=S+i; Вывод S	3.2 Описание S,i; S:=0; i:=1; While i<=5 do S:=S+i; I:=i+1; Вывод S	3.3 Описание S,i; S:=0; i:=1; S=S+i; I:=i+1; Until i>=6; Вывод S	**Структограммы**

4.1:
Program P2;
Var i,S:integer;
Begin
S:=0;
For i:=1 to 5 do
S:=S+i;
Writeln('S=',S:5);
End.
	4.2:
	Program P2;
	Var i,S:byte;
	Begin
	S:=0; i:=1;
	While i<=5 do
	Begin
	S:=S+i;
	I=i+1;
	End;
	Writeln('S=',S);
	End.
		4.3:
		Program P3;
		Var i,S:integer;
		Begin
		S:=0; i:=1;
		Repeat
		S:=S+i;
		I=i+1;
		Until i>=6;
		Writeln(S);
		End.
			Текст программы

Практические задания

Задание 1. Составьте программы, используя регулярный оператор цикла.

Написать программу, которая выводит таблицу квадратов первых десяти чисел. Ниже представлен рекомендуемый вид экрана во время работы программы.

Таблица квадратов

Число	Квадрат
1	1
2	4
3	9
4	16
5	25
6	36
7	49

Задание 2. Составьте программу, используя оператор Repeat

Написать программу, вычисляющую произведение положительных чисел, которые вводятся с клавиатуры, используя оператор Repeat. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом).

Вычисление произведения положительных чисел.

Вводите после стрелки числа. Для завершения ввода введите ноль.

*-> **45** -> **23** -> **15***
Введено чисел: 3
Произведение чисел =

Контрольные вопросы:

1. В каких случаях в программе необходимо использовать итерационный цикл, а в каких регулярный цикл?
2. Назовите отличия итерационных циклов и цикла с параметром.
3. Какова структура оператора цикла с параметром? Как выполняется цикл с парметром?
4. Какого типа должны быть пареметр цикла, его начальное и конечное значения в цикле с параметром в языке Pascal?
5. Могут ли параметр цикла, его начальное и конечное значения в цикле с параметром в языке Pascal быть разных типов? Обоснуйте ответ.
6. Чем отличается цикл «До» от цикла «Пока»?
7. Сколько раз повторится итерационный цикл?
8. Какова структура цикла с постропроверкой условия?
9. Какова структура цикла с предпроверкой условия?