МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Государственное образовательное учреждение
высшего профессионального образования
«Санкт-Петербургский государственный
университет аэрокосмического приборостроения»

Кафедра Вычислительных систем и сетей (№44)

Программирование на языках высокого уровня

Учебное пособие
для студентов заочной формы обучения

Санкт-Петербург
2003

Составители: Л.Н.Бариков, Н.Н.Бровин, Л.В.Плющева
Рецензенты: кафедра Компьютерных систем автоматизации Санкт-Петербургского государственного университета аэрокосмического приборостроения;
канд. техн. наук В.А.Галанина.

Содержатся необходимые материалы к выполнению курсовой, контрольных, практических и лабораторных работ, предусмотренных учебным планом по дисциплине “Программирование на языках высокого уровня”.

Предназначено для студентов, обучающихся специальности 220100 "Вычислительные машины, комплексы, системы и сети" на заочной форме обучения.

Подготовлены к публикации кафедрой Вычислительных систем и сетей по рекомендации методической комиссии факультета Вычислительных систем и программирования Санкт-Петербургского государственного университета аэрокосмического приборостроения.

Подписано к печати Формат 60x84 1/16
Объем 6,6 п.л. Уч.-изд.л. 6,6 Тираж 150 экз.
Зак. №

Ротапринт ЛИАП 190000, Ленинград, ул.Б.Морская, 67

Содержание

Введение

Учебное пособие содержит информационный материал, необходимый студентам заочной формы обучения специальности 220100 “Вычислительные машины, комплексы, системы и сети” для выполнения контрольных и лабораторных работ, а также курсового проекта по дисциплине “Программирование на языках высокого уровня”. Приводятся необходимые теоретические материалы, методические указания к выполнению работ и варианты индивидуальных заданий.

В пособие включен перечень тем, составляющих содержание теоретического курса по этой дисциплине с указанием объема в часах установочных лекций, читаемых во время экзаменационной сессии. Приведен список вопросов, вошедших в экзаменационные билеты.

Кроме того, в пособии содержится подробный перечень основной и дополнительной литературы по этой дисциплине, а также перечень действующих Государственных стандартов, которые требуется соблюдать при выполнении обязательных работ в течение семестра и экзаменационной сессии.

В приложении приведены формы титульных листов контрольных и лабораторных работ, а также пояснительной записки к курсовому проекту, которые должны быть выполнены на стандартных листах бумаги формата А4.

1 Распределение фонда учебного времени по семестрам и видам занятий, формы контроля

Номер семестра	Число недель	Число учебных часов							Форма контроля по семестрам			
		Всего	Аудиторные часы					Самос. работа				
			Лек.	Лаб.	Прак.	ИРП	Всего		Экз.	Зач.	КР	Кр
2	18	96	14	9	-	13	36	60		*		*
3	18	67	16	6	12	11	45	22	*			*
4	18	87		9	-	9	18	69			*	

2 Цели и задачи дисциплины

Дисциплина служит для формирования у студентов представлений о современном состоянии алгоритмизации и программирования, о методах разработки алгоритмов и программ, о современном программном обеспечении, операционных системах и средствах для разработки программ различного уровня сложности. Задачей дисциплины является развитие практических навыков разработки алгоритмов и программ с использованием любых языков программирования и сред для разработки программ.

В результате изучения дисциплины студенты должны:

- *знать*

основные принципы разработки, написания и отладки программ

разной степени сложности на языках программирования с использованием современных инструментальных средств;

-уметь

для сформулированной задачи разработать алгоритм, написать программу на языке (Turbo Pascal, Object Pascal, C++ или другом языке) высокого уровня, отладить программу и получить ее решение в заданной инструментальной среде;

- иметь представление

о современном состоянии средств разработки программ, тенденциях развития средств и систем для проектирования программ.

3 Содержание дисциплины

Теоретический курс включает следующие темы:

Тема 1. Этапы решения задач на ЭВМ

Постановка задачи и спецификация программы. Формализация задачи. Алгоритмизация. Программирование. Тестирование и отладка. Документирование. Сопровождение программы. Алгоритмы. Свойства и способы записи алгоритма: естественные языки, схемы, структурограммы, псевдоязыки, языки программирования. Основные правила разработки алгоритмов. Базовые алгоритмические структуры: следование, развилка, повторение. Способы их изображения. Типы алгоритмов. Пошаговая детализация как метод проектирования алгоритмов.

Тема 2. Современные языки программирования

Понятие языка программирования. Этапы развития языков программирования. Современные тенденции в области языков программирования. Сравнение развития языков в представлении данных и способах реализации алгоритмов. Сравнительная характеристика языков программирования высокого уровня. Синтаксис и семантика. Способы описания синтаксиса: лингвистические формулы и синтаксические диаграммы. Структура языка программирования. Базовые элементы языка: алфавит, лексемы, выражения. Предложения языка: описания и операторы. Программа на языке высокого уровня: состав и структура. Критерии качества программы. Жизненный цикл программы.

Тема 3. Средства реализации основных типов алгоритмов

Описательные предложения языка программирования высокого уровня. Описание используемых библиотек, модулей, меток, констант, типов, переменных. Области действия описаний. Исполнительные предложения языка высокого уровня. Представление основных управляющих структур программирования. Средства реализации линейных алгоритмов: операторы присваивания, обращения к процедуре, составной, пустой. Средства реализации разветвляющихся алгоритмов: условный оператор, операторы выбора, перехода. Средства реализации циклических алгоритмов: операторы цикла с предусловием, с постусловием, с параметром. Реализация арифметических, итерационных и вложенных циклов. Реализация рекуррентных вычислений. Реализация алгоритмов сортировки структур данных и поиска в этих структурах.

Тема 4. Концепция данных

Данное как совокупность значения и типа. Характеристики, определяемые типом данного: множество значений, множество операций, структурная организация и внутреннее представление. Три классификации типов данных. Стандартные типы данных и нестандартные типы данных. Простые и структурированные типы данных. Порядковые и непорядковые типы данных. Дерево типов. Простые типы данных: целые, вещественные, символьный, логический, перечисляемые и ограниченные типы. Встроенные языковые средства для работы с данными простых и порядковых типов. Преобразование типов данных. Структурированные типы данных: одномерные и многомерные массивы, строки постоянной и переменной длины, множества, записи с постоянной и вариантной частью. Встроенные языковые средства для работы со строками и множествами. Статическое распределение памяти.

Файлы. Структурная организация. Виды файлов. Файлы прямого и последовательного доступа к элементам. Файл как основное понятие баз данных и знаний. Файлы типизированные, не типизированные, текстовые. Стандартные файлы ввода и вывода. Встроенные языковые средства для работы с файлами разных видов.

Динамические структуры данных. Указатели и ссылки. Встроенные языковые средства для работы с динамической памятью. Динамические массивы. Списки. Виды списков: односвязные и двусвязные списки, линейные и циклические списки. Линейные списки: основные виды и способы реализации. Линейный список как абстрактный тип данных. Деревья. Виды деревьев и способы их реализации. Другие виды динамических структур: стек, очередь, дек. Правила использования

памяти при работе с динамическими структурами данных.

Тема 5. Способы конструирования программ

Обзор современных методов программирования. Структурное, процедурное, модульное и объектно-ориентированное программирование. Технологии нисходящего и восходящего проектирования программ.

Сущность структурного программирования: разбиение на подзадачи, нисходящее проектирование, стандартные структуры управления. Достоинства и недостатки. Виды стандартных управляющих структур. Базовые управляющие структуры: следование, развилка, цикл с предусловием. Дополнительные управляющие структуры: обход, выбор варианта, цикл с постусловием, цикл с параметром. Реализация стандартных управляющих структур на современных языках программирования. Примеры использования управляющих структур. Правила проектирования и оформления структурных программ.

Концепции процедурного программирования. Процедуры и функции. Основные понятия. Принципы использования процедур и функций в программах. Параметры процедур и функций. Виды параметров: параметры-значения, параметры-переменные, параметры-константы. Вызов процедур и функций на исполнение. Формальные и фактические параметры. Механизм передачи параметров. Процедурные типы. Параметры процедурного типа. Примеры использования. Области действия описаний процедур и функций. Внутренние и внешние блоки. Локальность и глобальность. Организация интерфейса диалоговых программ.

Понятие рекурсии. Рекурсивные определения и алгоритмы. Программирование рекурсивных алгоритмов: рекурсивные процедуры и функции. Механизм рекурсивных вызовов. Бинарное дерево как рекурсивная структура данных. Рекурсивные процедуры обхода дерева: инфиксная форма, префиксная форма, постфиксная форма. Особенности использования рекурсии при построении дерева.

Концепции модульного программирования. Модули: назначение, структура, трансляция, тестирование. Особенности использования модулей. Модульные программы. Стандартные модули в системах программирования: назначение и правила использования. Организация взаимодействия программных модулей. Построение многомодульных программ средствами языка программирования высокого уровня. Запуск внешних программ. Командная строка. Многопрограммные комплексы.

Тема 6. Инструментальные средства разработки программ.

Современные интегрированные среды проектирования программ. Состав и назначение элементов интегрированной среды программирования: текстовый редактор, транслятор, редактор связей, компоновщик, загрузчик, отладчик, инструктор, библиотекарь, профайлер. Схема обработки программы на языке программирования. Трансляция, виды трансляторов. Основные этапы трансляции. Набор, редактирование, отладка и выполнение программ в интегрированной среде программирования. Интерфейс пользователя среды.

Распределение времени по разделам программы и видам занятий

Номер и наименование раздела программы	Число учебных часов				
	Лекции	Лабораторные работы	Практические занятия	Индивидуальная работа преподавателя	Самостоятельная работа
1.Этапы решения задач на ЭВМ	4	-	-	4	20
2.Современные языки программирования	4	-	-	6	30
3.Средства реализации основных типов алгоритмов	6	9	6	6	20
4.Концепция данных	4	6	6	6	20
5.Способы конструирования программ	6	9	-	5	31
6.Инструменталь-ные средства разработки программ	6	-	-	6	30
Итого	30	24	12	33	151

4 Методические указания к выполнению лабораторных работ

Лабораторные занятия проводятся с целью приобретения практических навыков алгоритмизации, программирования, тестирования и отладки программ на компьютере с использованием современных технологий и инструментальных средств.

Перечень лабораторных работ

1. Лабораторная работа №1. Разветвления.
2. Лабораторная работа №2. Выбор варианта.
3. Лабораторная работа №3. Циклы.
4. Лабораторная работа №4. Массивы.
5. Лабораторная работа №5. Подпрограммы.
6. Лабораторная работа №6. Текстовые файлы.
7. Лабораторная работа №7. Файлы прямого доступа.
8. Лабораторная работа №8. Линейные списки.

Выполнение каждой лабораторной работы включает разработку алгоритма, написание программы, тестирование и отладку программы на компьютере в одной из компьютерных лабораторий университета, демонстрацию результатов преподавателю, составление отчета о лабораторной работе. Содержание отчета должно полностью соответствовать заданию на эту лабораторную работу. Форма титульного листа отчета приведена в приложении.

Лабораторная работа №1. Разветвления

Объем в часах: аудиторных занятий - 3, самостоятельных - 4.

Цель лабораторной работы:

изучение концепций и освоение технологии структурного программирования, приобретение навыков программирования на языке Турбо Паскаль при решении логических задач.

Задание на программирование

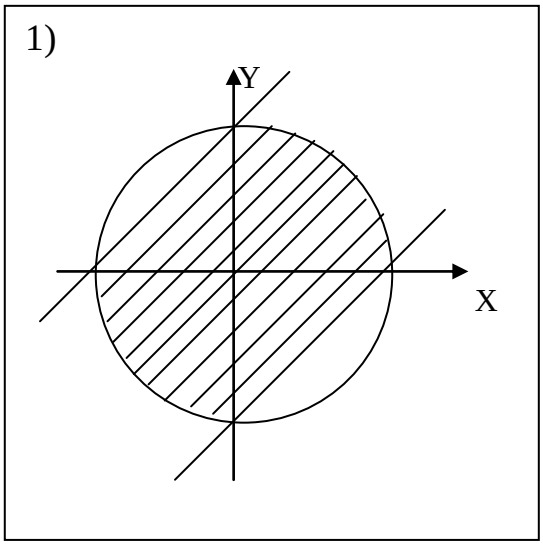
Используя технологию структурного программирования, разработать разветвляющуюся программу для решения индивидуальной задачи определения местонахождения на плоскости точки с заданными координатами.

Порядок выполнения работы

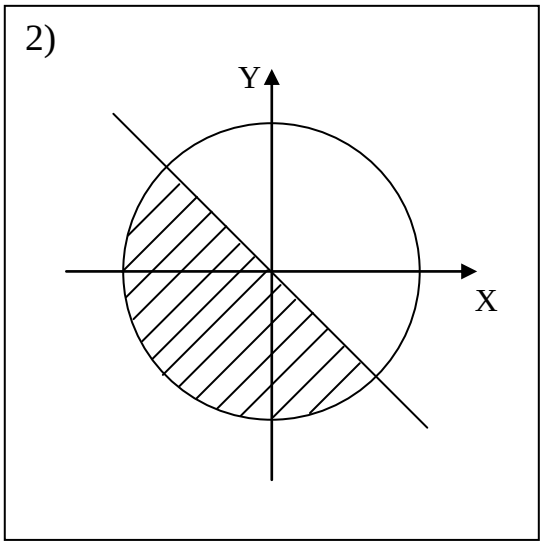
1. Получить у преподавателя индивидуальное задание и выполнить постановку задачи: сформулировать условие, определить входные и выходные данные.
2. Разработать математическую модель: привести уравнения линий, ограничивающих выделенные штриховкой области, описать условия попадания точки в каждую область (количество областей от 3 до 6).
3. Построить схему алгоритма решения задачи.
4. Составить программу на языке Турбо Паскаль.
5. Входные данные вещественного типа ***Real*** вводить с клавиатуры по запросу. Выходные данные (сообщения) выводить на экран в развернутой форме.
6. Проверить работу программы на полном наборе тестов. Продемонстрировать преподавателю возможные варианты выполнения, в том числе с ошибочными данными.
7. Оформить отчет о лабораторной работе в составе: *постановка задачи, математическая модель, схема алгоритма решения, текст программы, контрольные примеры.*

Варианты индивидуальных заданий

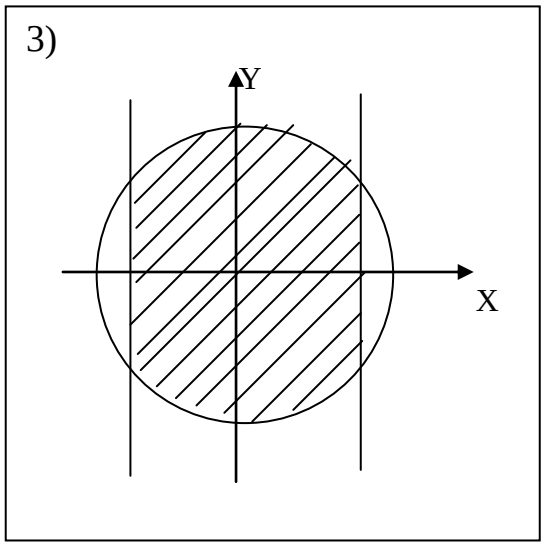
1)



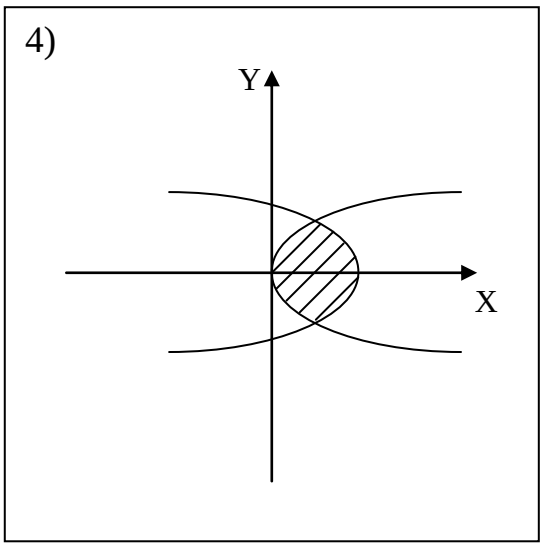
2)



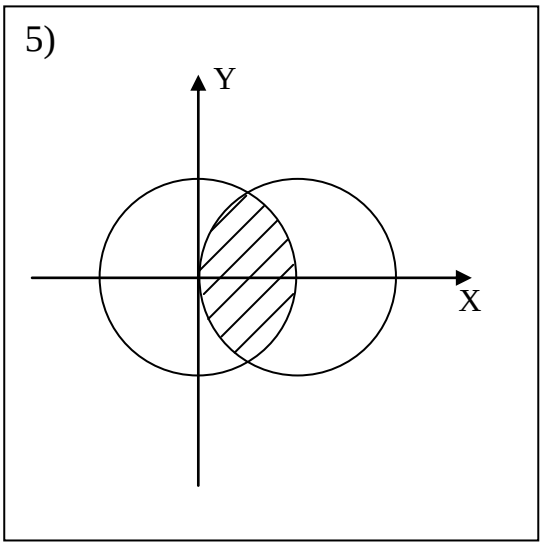
3)



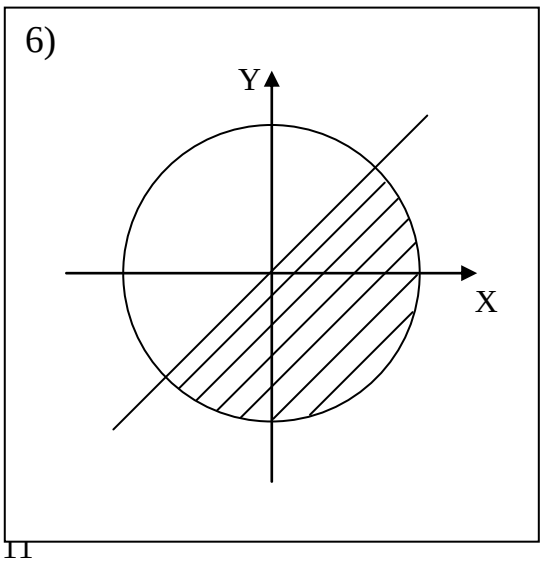
4)



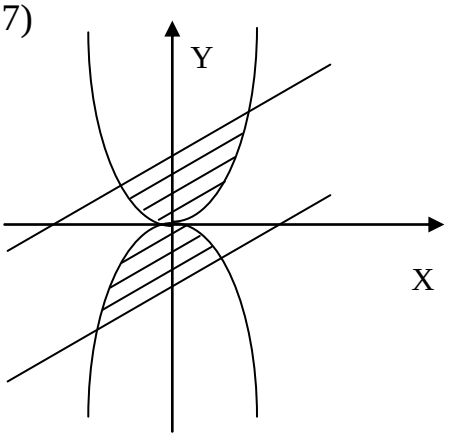
5)



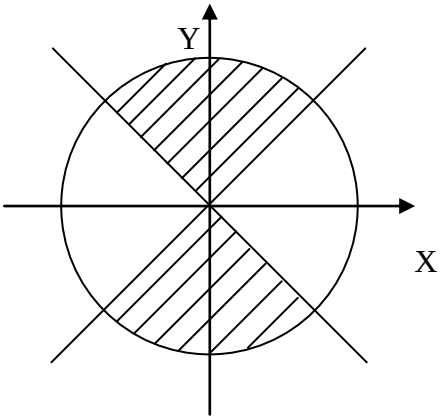
6)



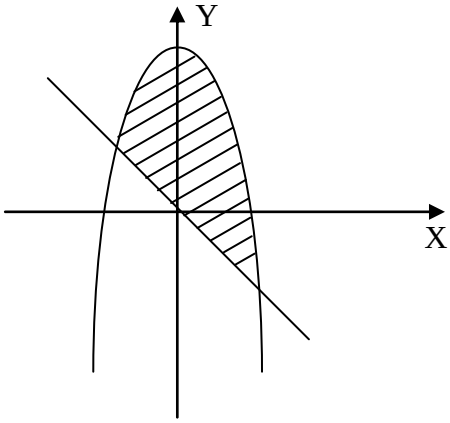
7)



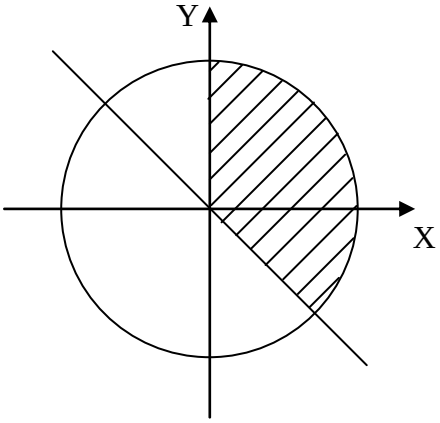
8)



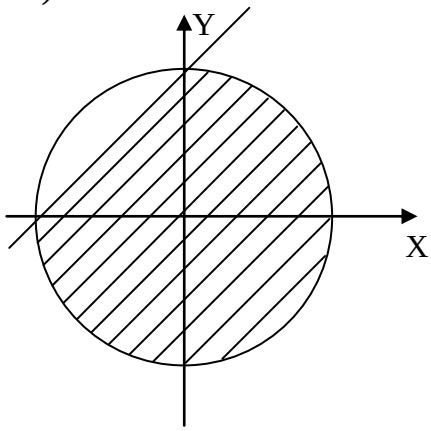
9)



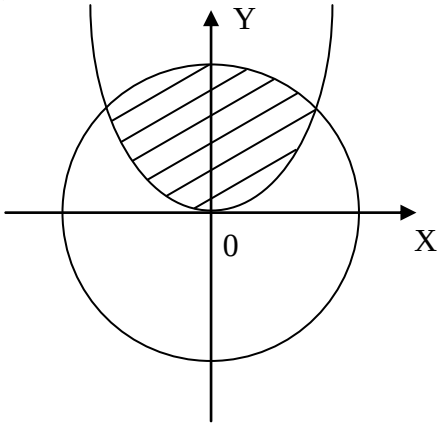
10)



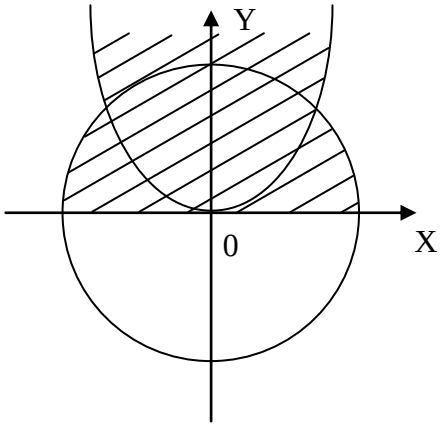
11)



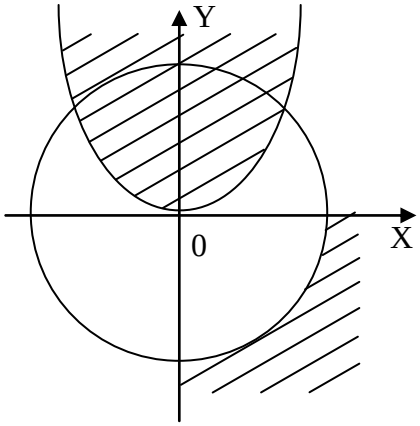
12)



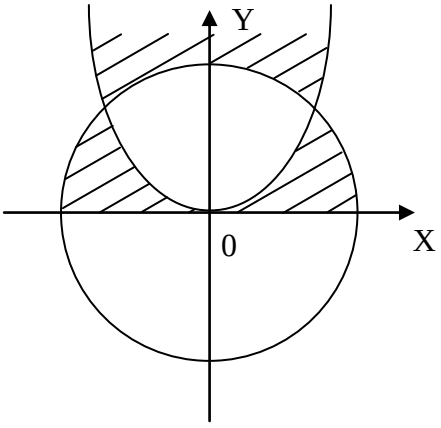
13)



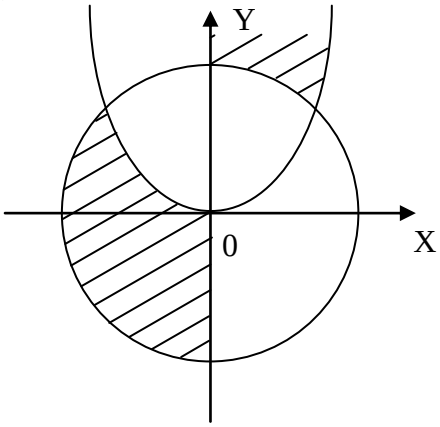
14)



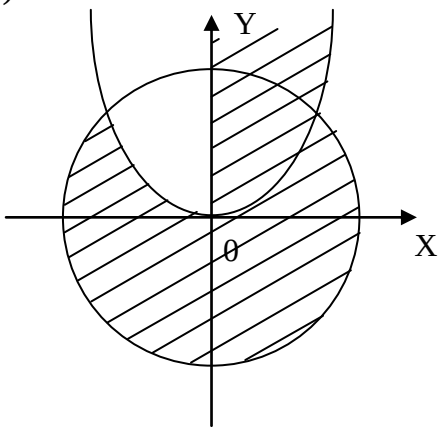
15)



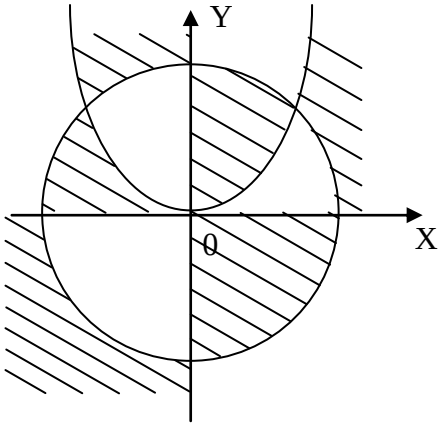
16)



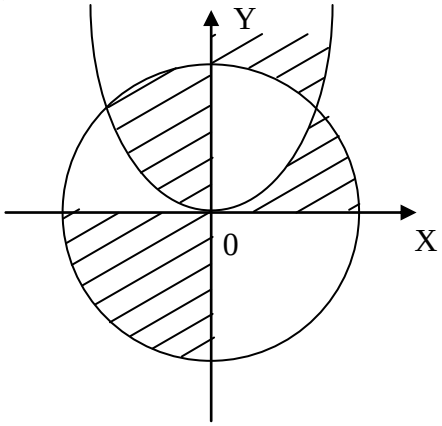
17)



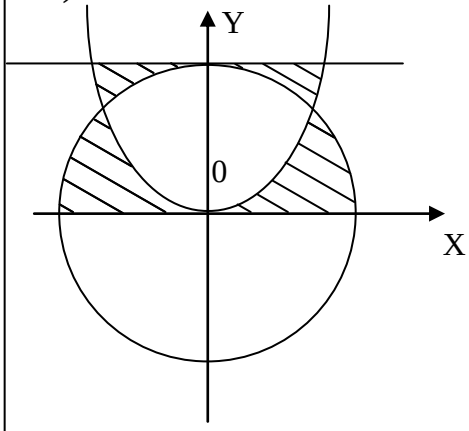
18)



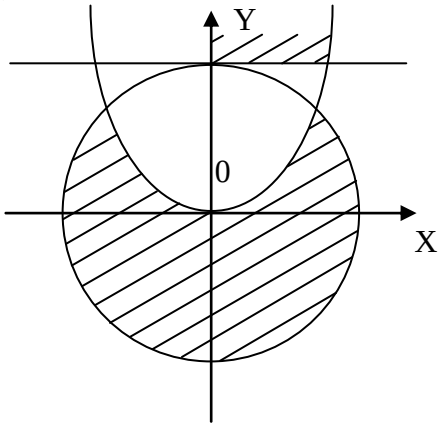
19)



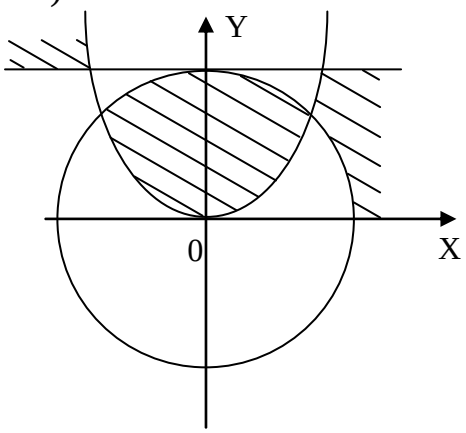
20)



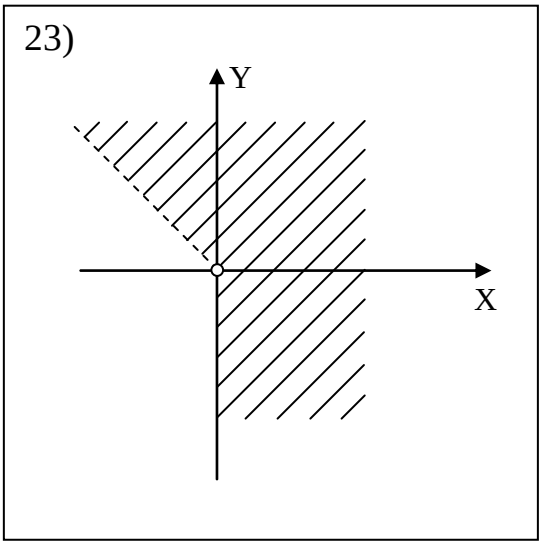
21)



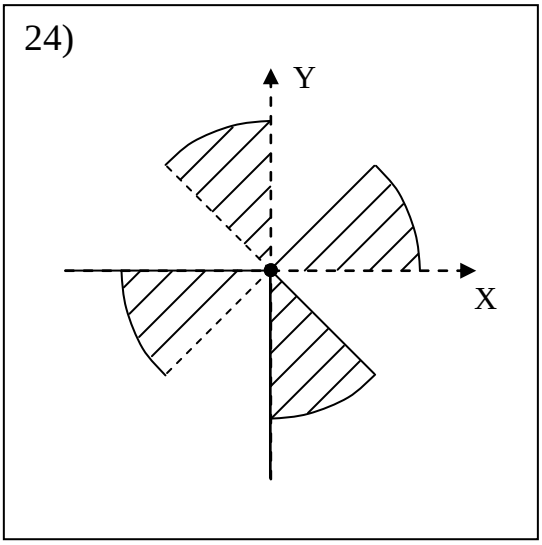
22)



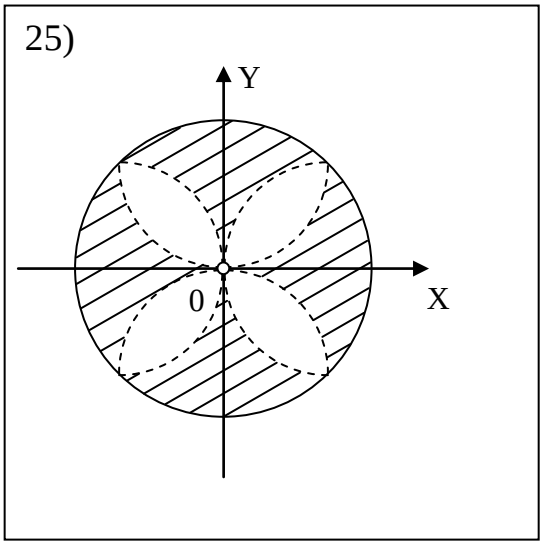
23)



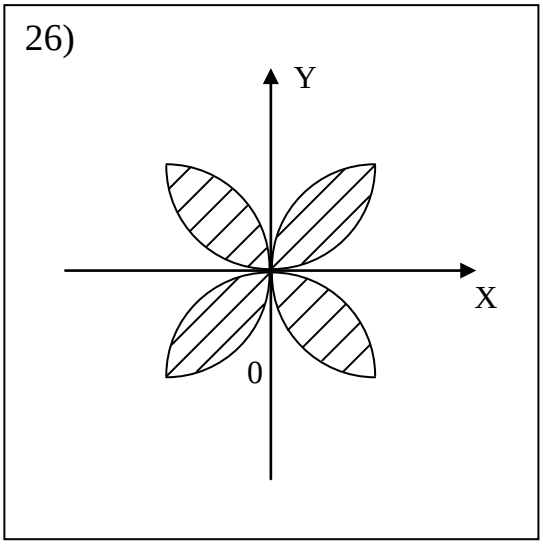
24)



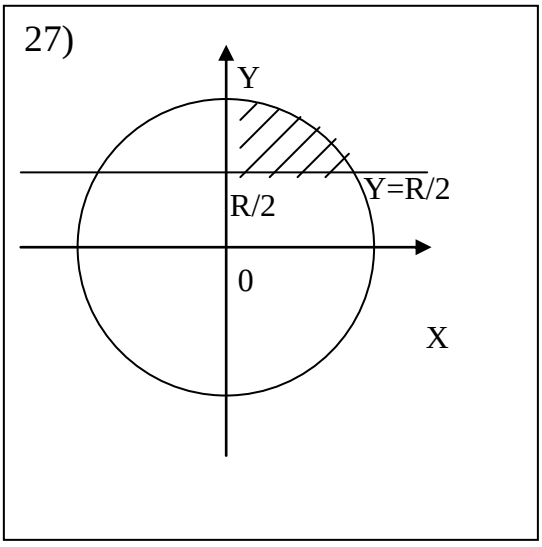
25)



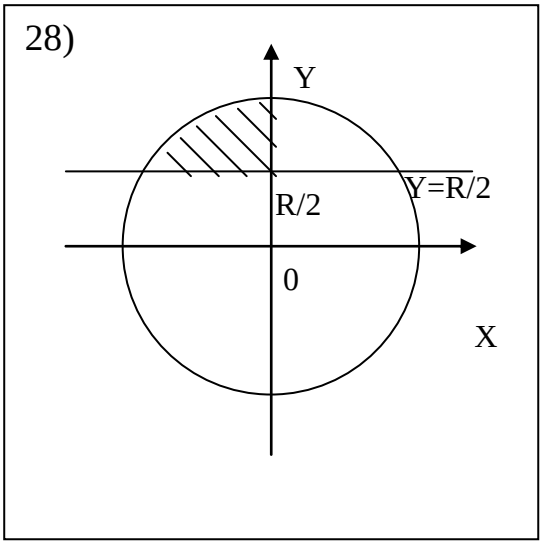
26)



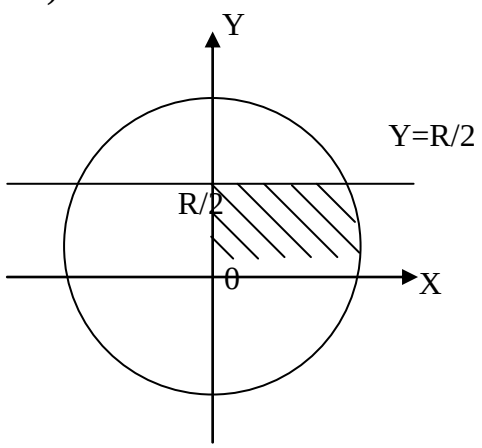
27)



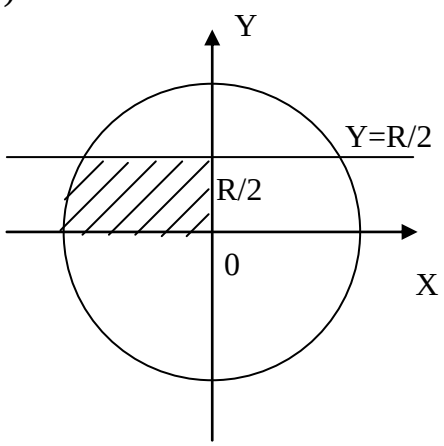
28)



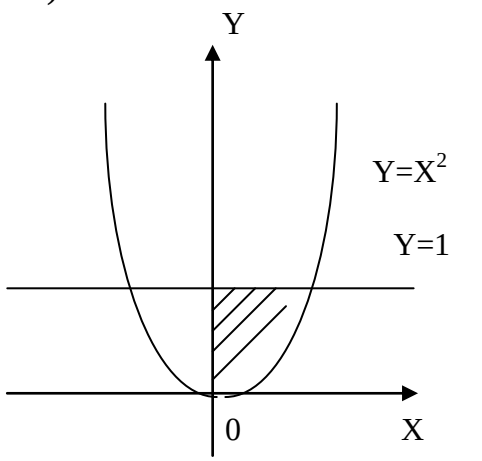
29)



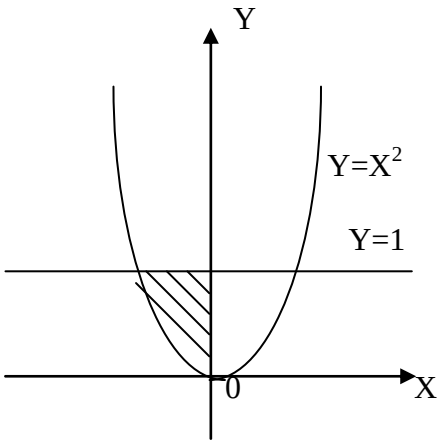
30)



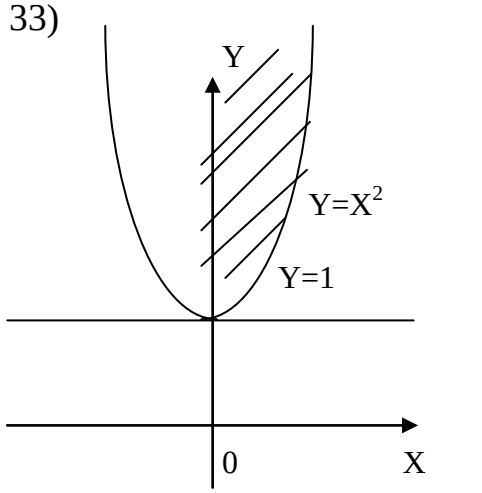
31)



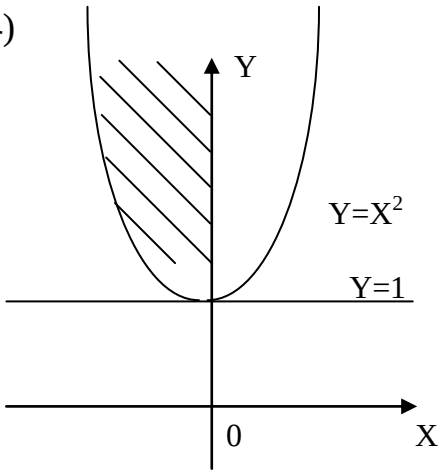
32)



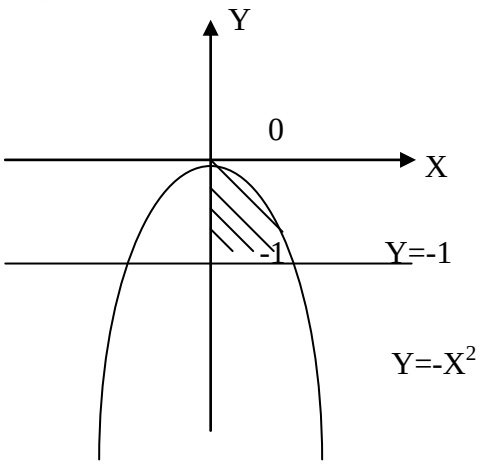
33)



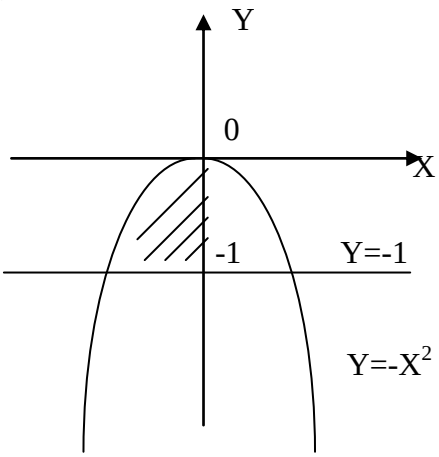
34)



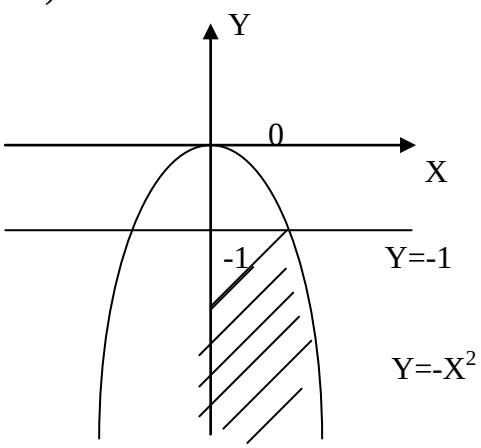
35)



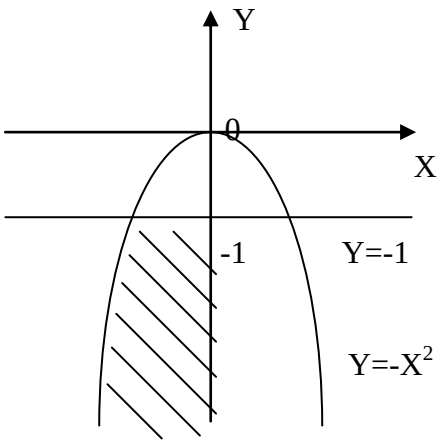
36)



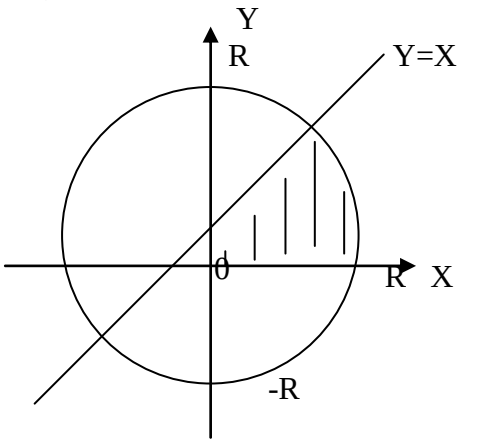
37)



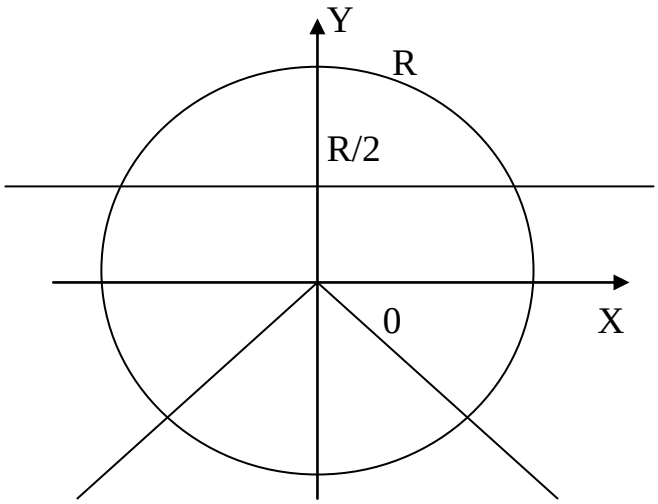
38)

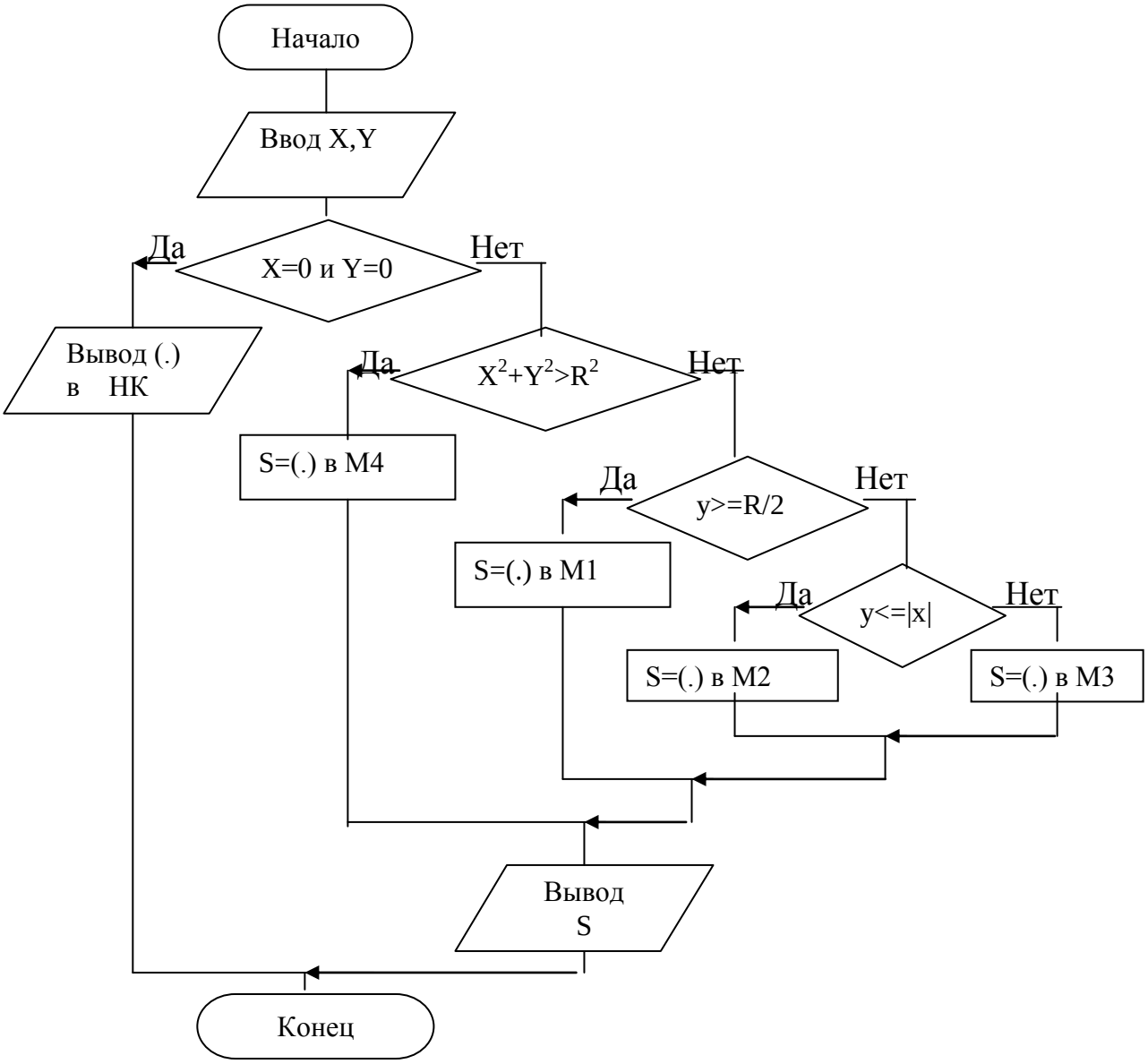


39)



Пример схемы алгоритма и текста программы определения местоположения точки для варианта задания вида:





```
Program Tochka;
{Определение местоположения точки на плоскости
  Входные данные: x, y – координаты точки
  Выходные данные: s - сообщение}
Var x, y: Real;
      s: String;
Begin
{Ввод и контроль}
  Write('Введите координаты точки :');
  ReadLn(x, y);
{Анализ координат}
  If (x = 0) And (y = 0)
  Then WriteLn ('Точка в начале координат');
  Else Begin
         If x * x + y * y > r * r {Вне круга?}
         Then s:= ' вне круга'
         Else {В круге}
           If y >= r / 2    {Выше горизонтальной линии?}
           Then s:= ' в круге выше горизонтальной линии'
           Else {Ниже горизонтальной линии}
             If y <= - Abs (x)
             Then s:= ' ниже ломаной'
             Else s:= ' между горизонт. линией и ломаной';
{Вывод сообщения}
         WriteLn ('Положение точки:', s);
       End;
End.
```

Лабораторная работа №2. Выбор варианта

Объем в часах: аудиторных занятий - 3, самостоятельных - 4.

Цель лабораторной работы:

изучение концепций и освоение технологии структурного программирования, приобретение навыков программирования на языке Турбо Паскаль многовариантных вычислений.

Задание на программирование

Используя технологию структурного программирования, разработать разветвляющуюся программу для решения индивидуальной задачи выбора варианта вычисления по ключу.

Порядок выполнения работы

1. Получить у преподавателя индивидуальное задание и выполнить постановку задачи: сформулировать условие, определить входные и выходные данные.
2. Разработать математическую модель:
   - составить список различных вариантов получения выходных данных задачи,
   - выявить ключ выбора - данное целого типа, значения которого могут служить ключами различных вариантов выполнения действий,
   - с помощью формул описать варианты получения выходных данных задачи в зависимости от значения ключа выбора варианта.
3. Построить схему алгоритма решения задачи.
4. Составить программу на языке Турбо Паскаль.
5. Входные данные вводить с клавиатуры по запросу. Выходные данные выводить на экран в развернутой форме с пояснениями.
6. Проверить работу программы на полном наборе тестов. Продемонстрировать преподавателю возможные варианты выполнения, в том числе с ошибочными данными.
7. Оформить отчет о лабораторной работе в составе: *постановка задачи, математическая модель, схема алгоритма решения, текст программы, контрольные примеры.*

Варианты индивидуальных заданий

1
Определить название месяца года, следующего за заданным месяцем.

2
Определить название k-го месяца после заданного месяца года.

3
Определить название столицы по заданному названию страны.

4
Определить название десятичной цифры по заданному ее значению.

5
Определить написание заданной десятичной цифры римскими цифрами.

6
Определить двоичный код заданной десятичной цифры.

7
Определить сезон года (зима, весна, лето, осень), на который приходится заданный месяц.

8
Определить название континента (Азия, Америка, Африка, Европа) по заданному названию страны.

9
Определить название цвета радуги, следующего за заданным цветом.

10
Определить название интервала (секунда, терция, кварта, квинта, секста, септима), образованного двумя заданными нотами (до, ре, ми, фа, соль, ля, си).

11
Определить величину в метрах некоторой длины, заданной в одной из указанных единиц измерения (километр, метр, дециметр, сантиметр, миллиметр).

12
Для целого числа k от 1 до 99 вывести фразу “Мне k лет”, учитывая при этом, что при некоторых значениях k слово “лет” надо заменить словом “год” или “года”.

13
Для натурального числа k вывести фразу “Мы нашли k грибов в лесу”, согласовав слово “гриб” с числом k.

14
Для целого числа d от 1 до 999, обозначающего денежную единицу, дописать слово “рубль” в правильной форме.

15
Для целого числа d от 1 до 99, обозначающего денежную единицу, дописать слово “копейка” в правильной форме.

16
Вычислить стоимость междугородного телефонного разговора заданной продолжительности. Цена одной минуты определяется по указанному коду города.

17
Вывести указанное слово из группы однотипно склоняемых слов (степь, боль, тетрадь, дверь) в заданном падеже (им., род., дат., вин., твор., предл.).

18
Корабль сначала шел по заданному курсу (север, восток, юг, запад). Затем его курс был изменен согласно заданному приказу (вперед, вправо, назад, влево). Определить новый курс корабля.

19
Определить количество дней в указанном месяце заданного года.

20
Определить, образует ли заданная тройка чисел y (год), m (месяц), d (день) правильную дату.

21
По заданной дате d (день), m (месяц), y(год) определить дату d1, m1, y1 следующего дня.

22
Определить порядковый номер того дня високосного года, который имеет заданную дату d (день), m (месяц).

23
Определить d (день), m (месяц) – дату k-го по счету дня високосного года.

24

Считая, что год не високосный и его 1 января приходится на день недели wd1, определить wd – день недели, на который приходится день с датой d (день), m (месяц).

25

Считая, что год не високосный и его 1 января приходится на день недели wd1, определить количество понедельников в году, приходящихся на 13-е числа.

Пример программы с оператором CASE

```
Program PloFig;
{Вычисление площадей геометрических фигур.
 Входные данные:  t - тип фигуры,
            a,l,h,r - параметры фигур.
 Выходные данные: s - площадь фигуры.      }
Var t:Byte;
    a,l,h,r,s:Real;
Begin {Ввод и контроль}
      WriteLn('Задайте тип фигуры:');
      Write('1-квадрат,2-прямоугольник,3-круг? ->');
      ReadLn(t);
      If (t<1)or(t>3)
      Then Begin WriteLn('Ошибочный тип фигуры!');
                 Write('Нажмите Enter ->');
                 ReadLn; Halt;
           End;
      {Вычисление площади}
      Case t Of
        1: Begin Write('Сторона квадрата? ->');
                 ReadLn(a);
                 s:=a*a;
           End;
        2: Begin Write('Стороны прямоугольника? ->');
                 ReadLn(l,h);
                 s:=l*h;
           End;
        3: Begin Write('Радиус круга?->');
                 ReadLn(r);
                 s:=Pi*r*r;
           End;
      End;
      WriteLn('Площадь фигуры: ',s:10:5);
End.
```

Лабораторная работа №3. Циклы

Объем в часах: аудиторных занятий - 3, самостоятельных - 4.

Цель лабораторной работы:
изучение концепций и освоение технологии структурного программирования, приобретение навыков программирования на языке Турбо Паскаль циклических вычислений.

Задание на программирование

Используя технологию структурного программирования, разработать программу решения двух индивидуальных задач, содержащую 3 вида циклических управляющих структур: Цикл - Пока (с предусловием), Цикл - До (с постусловием), Цикл - Для (с параметром).

Порядок выполнения работы

1. Получить у преподавателя индивидуальное задание и выполнить постановку задач: сформулировать условия, определить входные и выходные данные.
2. Разработать математическую модель.
3. Построить схему алгоритма. Обосновать выбор циклических управляющих структур.
4. Составить программу на языке Турбо Паскаль.
5. В программе использовать управляющую структуру ***For*** только для реализации цикла с известным числом повторений. Если число повторений в цикле неизвестно, то привести два различных варианта его реализации: управляющей структурой ***While*** и управляющей структурой ***Repeat.***
6. Входные данные вводить с клавиатуры по запросу. Выходные данные выводить на экран в развернутой форме с пояснениями.
7. Проверить работу программы на полном наборе тестов.
8. Оформить отчет о лабораторной работе в составе: *постановка задачи, математическая модель, схема алгоритма, текст программы, контрольные примеры.*

Варианты индивидуальных заданий

1

1. По введенным с клавиатуры значениям X, m вычислить S:

$$S = \sum_{i=1,3,5,\ldots}^{2\cdot m+1} i \cdot X^{-2}$$

2. Вычислить предел последовательности $\{Y_n\}$ при $n \to \infty$, где Y_n вычисляется по формуле $Y_n = 0.25\sin(Y_{n-1}) + \sin(Y_{n-2}); n = 2,3,4\ldots$

Значения Y_0, Y_1 вводятся с клавиатуры. Вычисления прекратить при выполнении условия $|Y_n - Y_{n-1}| < \varepsilon$.

2

1. По введенным с клавиатуры значениям X, m вычислить P:

$$P = \prod_{i=1}^{m} (m + \frac{X}{m - i + 1})$$

2. Вычислить предел последовательности $\{Y_n\}$ при $n \to \infty$, где Y_n вычисляется по формуле $Y_n = 0.2 + 0.1 \sin(Y_{n-1})$; n=1,2,3...

Значение Y_0 вводится с клавиатуры. Вычисления прекратить при выполнении условия $|Y_n - Y_{n-1}| < \varepsilon$.

3

1. По введенным с клавиатуры значениям A, B, n, m и X вычислить S:

$$S = A + \sum_{i=m}^{n} (X + \frac{B}{i})^2$$

2. Вычислить предел последовательности $\{Y_n\}$ при $n \to \infty$,где Y_n вычисляется по формуле

$$Y_n = 0.1 \operatorname{tg}(Y_{n-1}) + 0.3 \operatorname{tg}(Y_{n-3}); \quad n = 3,4,5...$$

Значения Y_0, Y_1, Y_2 вводятся с клавиатуры. Вычисления прекращаются при выполнении условия $|Y_n - Y_{n-1}| < \varepsilon$.

4

1. По введенным с клавиатуры значениям A, B, n и X вычислить S:

$$S = A + B \sum_{i=2,4,6,\ldots}^{n} \frac{X - A \cdot B \cdot i}{X + A \cdot B \cdot i}$$

2. Вычислить предел последовательности $\{Y_n\}$ при $n\to\infty$,где $Y_0=0$, а Y_n вычисляется по формуле $Y_n = \frac{1}{1+Y_{n-1}}; n = 1,2,3...$

Значение Y_0 вводится с клавиатуры. Вычисления прекращаются при выполнении условия $|Y_n - Y_{n-1}|<\varepsilon$.

5

1. По введенным с клавиатуры значениям A, B, n, m и X вычислить S:

$$S = A + B\sum_{i=m}^{n}(-1)^i \frac{A + X\cdot i}{B + X\cdot i}$$

2. Вычислить предел последовательности $\{Y_n\}$ при $n\to\infty$, где Y_n вычисляется по формуле $Y_n = 0.352\cdot Y_{n-1} + \cos(\frac{\pi}{2} + Y_{n-2}); n = 2,3,4...$

Значения Y_0, Y_1 вводятся с клавиатуры. Вычисления прекратить при выполнении условия $|Y_n - Y_{n-1}|<\varepsilon$.

6

1. Вычислить сумму S значений функции Y=f(x):

$$S = \sum_i \frac{x^2 - 3x + 2}{\sqrt{2x^2 - 1}}; \text{ при } x = 1.5 + 0.1\cdot i; i = \overline{1,40}$$

2. Вычислить предел последовательности $\{Y_n\}$ при $n\to\infty$, где Y_n вычисляется по формуле

$$Y_n = \frac{1}{\sqrt{12 + Y_{n-1}^2 + Y_{n-2}^2}}; n = 2,3,4...$$

Значения Y_0, Y_1 вводятся с клавиатуры. Вычисления прекратить при выполнении условия $|Y_n - Y_{n-1}|<\varepsilon$.

7

1. Вычислить сумму S значений функции Y=f(x):

$$S = \sum_i \lg(\frac{x^2 + 1}{(i-1)!}); \text{ при } x = -1 + 0.2\cdot i; i = \overline{1,10}$$

2. Вычислить предел последовательности $\{Y_n\}$ при $n\to\infty$, где Y_n вычисляется по формуле

$$Y_n = \frac{1}{\sqrt{10 + Y_{n-2}^2 + Y_{n-3}^2}}; n = 3,4,5...$$

Значения Y_0, Y_1, Y_2 вводятся с клавиатуры. Вычисления прекратить при выполнении условия $|Y_n - Y_{n-1}|<\varepsilon$.

8

1. По введенному с клавиатуры значению X вычислить S:

$$S = \frac{(X-2)\cdot(X-4)\cdot(X-8)\cdot\ldots\cdot(X-128)}{(X-1)\cdot(X-3)\cdot(X-7)\cdot\ldots\cdot(X-127)}$$

2. Вычислить предел последовательности $\{Y_n\}$ при $n\to\infty$, где Y_n вычисляется по формуле

$$Y_n = \frac{1}{\sqrt{1+\sin^2 Y_{n-1}+\sin^2 Y_{n-2}}}; n = 2,3,4...$$

Значения Y_0, Y_1 вводятся с клавиатуры. Вычисления прекратить при выполнении условия $|Y_n - Y_{n-1}|<\varepsilon$.

9

1. Для заданного с клавиатуры значения N найти $(2\cdot N)!!$

2. Вычислить предел последовательности $\{Y_n\}$ при $n\to\infty$, где Y_n вычисляется по формуле

$$Y_n = \frac{Y_{n-2}+0.5\cdot Y_{n-3}}{Y_{n-2}^2+2Y_{n-3}^4+1.5}; n = 3,4,5...$$

Значения Y_0, Y_1, Y_2 вводятся с клавиатуры. Вычисления прекратить при выполнении условия $|Y_n - Y_{n-1}|<\varepsilon$.

10

1. Для заданного с клавиатуры значения N найти $(2\cdot N+1)!!$

2. Последовательность функций $Y_n = Y_n(x)$, где $0\le X\le 1$ определяется следующим образом:

$$Y_1 = \frac{X}{2}; Y_n = \frac{1}{2}(X + Y_{n-1}^2); n = 2,3,4...$$

При заданном X найти предел последовательности, принимая за таковой значение Y_n, удовлетворяющее условию $|Y_n - Y_{n-1}|<\varepsilon$.

11

1. Найти сумму всех целых чисел, кратных 5, из отрезка [A,B].

2. Последовательность функций $Y_n = Y_n(x)$, где $X>0$ определяется следующим образом:

$$Y_1 = X; Y_n = Y_{n-1}(2 - X\cdot Y_{n-1}); n = 2,3,4...$$

При заданном X найти предел последовательности, принимая за таковой значение Y_n, удовлетворяющее условию $|Y_n - Y_{n-1}|<\varepsilon$.

12

1. Найти сумму всех целых чисел, кратных 7, из отрезка [A,B].
2. Найти предел произведения $P = \prod_{n=1}^{\infty}(1+\frac{1}{Y_n})$ для последовательности $\{Y_n\}$, пользуясь рекуррентной формулой

$$Y_1 = 1; Y_n = n(Y_{n-1}+1); n = 2,3,4,...$$

Вычисления закончить при выполнении условия $\frac{1}{Y_n} < \varepsilon$.

13

1. Найти сумму всех целых чисел, дающих при делении на 5 в остатке 3, из отрезка [A,B].
2. Вычислить $\sqrt[K]{A}$ - корень k-ой степени из положительного числа A, пользуясь последовательным приближением

$$X_0 = A; X_n = \frac{k-1}{k}X_{n-1} + \frac{A}{k \cdot X_{n-1}^{k-1}}; n = 1,2,3,...$$

За корень принять такое X_n, при котором $|X_n - X_{n-1}| < \varepsilon$

14

1. Найти сумму всех целых чисел, дающих при делении на 7 в остатке 4, из отрезка [A,B].
2. Для приближенного решения уравнения Кеплера X-q*sin(X)=m, 0<q<1 полагают $X_0 = m, X_1 = m + q*\sin(X_0),..., X_n = m + q*\sin(x_{n-1}),...$

При заданном m найти решение уравнения Кеплера, принимая за него такое X_n, при котором $|X_n - X_{n-1}| < \varepsilon$.

15

1. Пользуясь рекуррентной формулой для заданного с клавиатуры m, вычислить $S_m = \sum_{i=1}^{m}\sqrt{Y_i}$ если известны Y_0, Y_1, Y_2, а Y_i вычисляется по формуле

$$Y_i = \lg\left|Y_{i-2}^2 + Y_{i-3} + 1\right|; i = 3,4,5,...$$

2. Вычислить предел последовательности $\{Y_n\}$ при $n \to \infty$, где Y_n вычисляется по формуле: $Y_n = \frac{n}{\sqrt{n^2 + 1 + \sqrt{2*n^2 - 1}}}$

Вычисления прекращаются при выполнении условия $|Y_n - Y_{n-1}| < \varepsilon$.

16

1. Пользуясь рекуррентной формулой для заданного с клавиатуры m, вычислить Y_m, если известны Y_0, Y_1,Y_2, а Y_i вычисляется по формуле

$$Y_i = tg^2(Y_{i-3}) + Y_{i-2}; i = 3,4,5,\ldots,\text{m}$$

2. Найти предел последовательности $\lim\limits_{n\to\infty} \dfrac{5*n}{3*\sqrt{(n^2+1)}+2*\sqrt{(n^2-1)}}$ с точностью ε.

17

1. Пользуясь рекуррентной формулой для заданного с клавиатуры m, вычислить $S_m = \sum\limits_{i=1}^{m} \ln(|Y_i| + 0.5)$, если известны Y_0, Y_1, Y_2, а Y_i вычисляется по формуле

$$Y_i = Y_{i-1} + Y_{i-2}^2 - 2*Y_{i-3}; i = 3,4,5,\ldots,\text{m}$$

2. Найти предел последовательности $\lim\limits_{n\to\infty} \dfrac{n^3+5}{2*n^3+n^2+1}$ с точностью ε.

18

1. Пользуясь рекуррентной формулой для заданного с клавиатуры m, вычислить Y_m, если известны Y_0, Y_1, а Y_i вычисляется по формуле

$$Y_i = \frac{2*Y_{i-1} + Y_{i-2}}{3}; i = 2,3,4,\ldots,\text{m}$$

2. Найти сумму бесконечного ряда $\sum\limits_{n=1}^{\infty} \dfrac{1}{n^2*(\sin(n)+1.1)}$ с точностью ε.

19

1. Пользуясь рекуррентной формулой для заданного с клавиатуры m, вычислить Y_m, если известны Y_0 ,Y_1,Y_2, а Y_i вычисляется по формуле

$$Y_i = \sin^2(Y_{i-1}) + \cos^2(Y_{i-3}); \text{ i=3,4,5, …,m}$$

2. Найти сумму бесконечного ряда $\sum\limits_{n=1}^{\infty} \dfrac{1}{n*(n+A)}$ с точностью ε.

20

1. Пользуясь рекуррентной формулой для заданного с клавиатуры m, вычислить $S_m = \sum\limits_{i=1}^{m} Y_i$, если известны Y_0, Y_1, Y_2, а Y_i вычисляется по формуле

$$Y_i = \sin(Y_{i-1}) + \cos(Y_{i-3}); \text{ i=3,4,5,…,m}$$

2. Найти сумму бесконечного ряда $\sum\limits_{n=1}^{\infty} \dfrac{1}{(5*n-1)*(5*n+1)}$ с точностью ε.

21

1. Пользуясь рекуррентной формулой для заданного с клавиатуры m, вычислить $S_m = \sum_{i=1}^{m} Y_i^2$ при известных Y_0 ,Y_1, если Y_i вычисляется по формуле

$$Y_i = \sqrt{\left|\sin(Y_{i-1}) + \cos(Y_{i-2})\right|};\ \text{i=2,3,4,…,m}$$

2. Найти сумму бесконечного ряда $\sum_{n=1}^{\infty}(-1)^n \frac{n}{2*n^2-1}$ с точностью ε.

22

1. Члены последовательностей $\{X_i\}$ и $\{Y_i\}$ вычисляются по двум рекуррентным формулам. Вычислить X_{20} ,Y_{20}, если

$$X_{i+1} = \sqrt{\frac{X_i*(Y_i+5)^{-1}}{2}}; X_0 = 3.5;$$
$$Y_{i+1} = \sqrt{X_i+1.6}; Y_0 = 2.2$$

2. Найти сумму бесконечного ряда $\sum_{n=1}^{\infty} \frac{\cos(\frac{\pi}{n} + 30^0)}{n}$ с точностью ε.

Пример программы с оператором FOR

```
Program Fibon;
{Определение числа Фибоначчи с заданным номером.
 Входное данное:   n - номер искомого числа.
 Выходное данное: fn - искомое число Фибоначчи.}
Var n,i:Byte; { 0..255 }   { i-текущий номер числа }
    fn,fi,fi1,fi2:Longint;
    {fi-текущее число, fi1-предыдущее число,
     fi2-предшествующее предыдущему число.}
Begin {$R+}
      Write('Номер числа Фибоначчи ? ');  ReadLn(n);
      If n>46
      Then Begin WriteLn('Недопустимый номер числа!');
                 Write('Нажмите Enter->');
                 ReadLn; Halt;
           End;
      If n<2 Then fn:=n
      Else Begin fi2:=0; fi1:=1;
                 For i:=2 To n
                 Do Begin fi:=fi1+fi2;
                          fi2:=fi1;
                          fi1:=fi;
                    End;
                 fn:=fi;
           End;
      WriteLn('Число Фибоначчи F',n,' = ',fn);
End.
```

Пример программы с оператором WHILE

```
Program ProSum;
{Вычисление произведения и суммы по условию}
{Входное данное: c - граничное значение.
 Выходные данные: p - произведение, s - сумма.}
Var c,p,s,a,b:Real;
    i,j:Byte; {i-номер сомножителя, j-номер слагаемого
               a- сомножитель, b- слагаемое}
Begin WriteLn('Задайте граничное значение 0<c<1:');
     ReadLn(c);
     If (c<=0)or(c>=1)
     Then Begin WriteLn('Недопустимое граничное значение!');
                Write('Нажмите Enter->');
                ReadLn; Halt;
          End;
{Вычисление произведения}
      p:=1; i:=0;
      While p>=c
      Do Begin i:=i+1; a:=1-i/(i+1); p:=p*a;
         End;
      WriteLn('Первое произведение <',c:6:3,' = ',p:7:5);
{Вычисление суммы}
      s:=0; j:=0;
      While s<=c
      Do Begin j:=j+1; b:=1/j/(j+1); s:=s+b;
         End;
      WriteLn('Первая сумма >',c:6:3,' = ',s:7:5);
End.
```

Пример программы с операторами REPEAT и WHILE

```
Program KvadrKoren;
{Вычисление приближенного значения квадратного корня с
заданной точностью.
Входные данные:  x - аргумент, eps - точность вычисления.
Выходное данное: y - квадратный корень}
Var x,y,yp,eps:Real;
    {y - текущее значение корня,
     yp - предыдущее значение корня}
Begin Write('Задайте аргумент >0 : ');
      ReadLn(x);
      If x<0 Then Begin WriteLn('Ошибка: аргумент <0!');
                        Write('Нажмите Enter->');
                        ReadLn; Halt;
                  End;
      Write('Задайте точность 0<eps<<1 : ');
      ReadLn(eps);
      If(eps<=0)Or(eps>0.1)
      Then Begin WriteLn('Ошибка: точность <=0 или >0.1!');
                 Write('Нажмите Enter->');
                 ReadLn; Halt;
           End;
{Реализация управляющей структурой Repeat - Until}
      yp:=(x+1)/2;
      Repeat y:=0.5*(yp+x/yp);
             yp:=y
      Until Abs(y*y-x)<eps;
      WriteLn('Repeat: Корень из ',x:6:3,'=',y:5:3);

{Реализация управляющей структурой While - Do}
      yp:=(x+1)/2; y:=0;
      While Abs(y*y-x)>eps
      Do Begin y:=0.5*(yp+x/yp);
               Yp:=y;
         End;
      WriteLn('While: Корень из ',x:6:3,'=',y:5:3);
End.
```

Лабораторная работа №4. Массивы

Объем в часах: аудиторных занятий - 3, самостоятельных - 4.

Цель лабораторной работы:

изучение структурной организации массивов и способов доступа к их элементам;

совершенствование навыков структурного программирования на языке Турбо Паскаль при решении задач обработки массивов.

Задание на программирование

Используя технологию структурного программирования, разработать программу обработки числовых массивов в соответствии с индивидуальным заданием.

Порядок выполнения работы

1. Получить у преподавателя индивидуальное задание: задачу обработки числовых массивов.
2. Выполнить постановку задачи: сформулировать условие, определить входные и выходные данные, их ограничения.
3. Разработать математическую модель: описать с помощью формул и рисунков структуру массивов и процесс их преобразования.
4. Построить схему алгоритма.
5. Составить программу на Турбо Паскале.
6. Входные данные вводить с клавиатуры по запросу. Выходные данные выводить на экран с пояснениями**.**
7. Отладить программу, проверить ее работу на полном наборе тестов. Продемонстрировать преподавателю несколько вариантов выполнения, в том числе, с ошибочными данными.
8. Оформить отчет о лабораторной работе в составе: *постановка задачи, математическая модель, схема алгоритма, текст программы, контрольные примеры.*

Варианты индивидуальных заданий

1

Дан массив $b_1,b_2,\ldots,b_{2n}$. Написать программу построения массивов $x_1,x_2,\ldots,x_n$ и $y_1,y_2,\ldots,y_n$, элементы которых равны соответственно значениям: $b_1,b_3,\ldots,b_{2n-1}$ и $b_2,b_4,\ldots,b_{2n}$.

2

Дан целочисленный массив $a_1,a_2,\ldots,a_m$. Из абсолютных величин его элементов выбрать наибольшую. Далее построить массив, i-й элемент которого равен нулю, если $|a_i|$ не совпадает с выбранным значением, и равен 1 в противном случае.

3

Написать программу построения массива с элементами: $a_1, a_1 + a_2, a_1 + a_2 + a_3, \ldots, a_1 + a_2 + a_3 + \ldots + a_n$ по данному массиву $a_1,a_2,\ldots,a_n$.

4

В вещественном массиве $x_1,x_2,\ldots,x_n$ заменить нулем все отрицательные элементы, предшествующие его максимальному элементу.

5

Даны массивы $a_1,a_2,\ldots,a_n$ и $b_1,b_2,\ldots,b_n$. Получить массив C, элементы которого: $a_1, b_1, a_2, b_2, \ldots, a_n, b_n$.

6

Дан вещественный массив $x_1,x_2,\ldots,x_m$. Все его элементы, следующие за наибольшим элементом, заменить на b.

7

Даны вещественные массивы $x_1,x_2,\ldots,x_n$ и $y_1,y_2,\ldots,y_n$. Преобразовать их по правилу: большее из x_i и y_i принять в качестве нового значения x_i, а меньшее – в качестве нового значения y_i.

8

Дан целочисленный массив $a_1,a_2,\ldots,a_n$. Если в массиве нет ни одной компоненты с заданным значением K, то первую по порядку компоненту этого массива, не меньшую всех остальных компонент, заменить на это значение K. .

9

Написать программу, осуществляющую циклический сдвиг компонент массива $x_1,x_2,\ldots,x_n$ (n>=2) на одну позицию влево, то есть получающую
массив $x_2,x_3,\ldots,x_n,x_1$.

10

Дан вещественный массив $a_1,a_2,\ldots,a_n$. Если в этом массиве есть хотя бы один элемент, меньший, чем P, то все отрицательные элементы массива заменить их квадратами, в противном случае массив a умножить на b.

11

Написать программу вычисления обратной величины $к$ произведению тех элементов массива $b_1,b_2,\ldots,b_n$, для которых выполнимо: $2^i<b_i<i!$. Если таких элементов нет, то ответом должно служить сообщение.

12

Преобразовать массив $a_1,a_2,\ldots,a_n$ так, чтобы его элементы расположились в обратном порядке: $a_n,a_{n-1},\ldots,a_1$.

13

Написать программу выбора среди элементов массива $a_1,a_2,\ldots,a_n$ наибольшего среди остающихся после выбрасывания наибольшего и всех ему равных. Предполагается, что не все элементы равны между собой.

14

Из массива $a_1,a_2,\ldots,a_{3n}$ получить массив $b_1,b_2,\ldots,b_n$, очередная компонента которого равна среднему арифметическому тройки очередных компонент массива a.

15

Дан целочисленный массив $b_1,b_2,\ldots,b_n$. Если элементы этого массива не образуют убывающей последовательности, то заменить его отрицательные элементы единицами.

16

Дан целочисленный массив $a_1,a_2,\ldots,a_n$, среди элементов которого могут быть равные. Из каждой группы равных между собой элементов нужно оставить только один, выбросив все остальные. Освободившийся хвост массива заполнить нулями.

17

Дан вещественный массив $a_1,a_2,\ldots,a_n$. Если в этом массиве есть хотя бы один элемент, принадлежащий отрезку $[x,y]$, то все элементы, не принадлежащие этому отрезку, заменить на K.

18

Дан массив $a_1,a_2,\ldots,a_n$. Переставить его элементы так, чтобы в начале массива расположились все его неотрицательные элементы, а в конце – отрицательные.

19

Написать программу выполнения следующего задания: из всех непрерывных участков массива $a_1,a_2,\ldots,a_n$, состоящих из нулей, выбрать наибольший по длине. Вывести индексы его начала и конца.

20

Написать программу сжатия массива $x_1,x_2,\ldots,x_n$ отбрасыванием нулевых элементов. Освобождающийся хвост заполнять нулями.

21

Написать программу нахождения в целочисленном массиве $a_1>a_2>\ldots>a_n$ совокупности элементов $a_i,a_{i+1},\ldots,a_{i+k}$ с суммой, равной M.

22

Дан массив $b_1,b_2,\ldots,b_{2m}$. Написать программу построения массива с элементами, соответственно равными: $b_{2m},b_1,b_{2m-1},b_2,\ \ldots,b_{m+1},b_m$.

23

Дан массив $x_1,x_2,\ldots,x_n$. Все элементы массива, предшествующие наибольшему из его отрицательных элементов, заменить их квадратами.

24

Дан массив $a_1,a_2,\ldots,a_{2n}$. Написать программу построения массива с элементами, соответственно равными: $a_1,a_{n+1},a_2,a_{n+2},\ldots,a_n,a_{2n}$.

25

Дан вещественный массив $c_1,c_2,\ldots,c_n$. Если элементы этого массива не образуют неубывающей последовательности, то все его отрицательные элементы заменить их квадратами.

26

Написать программу циклического сдвига вправо на K позиций одномерного массива из n элементов.

27

Написать программу слияния двух упорядоченных одномерных массивов $x_1,x_2,\ldots,x_n$ и $y_1,y_2,\ldots,y_m$ в третий массив $z_1,z_2,\ldots,z_{n+m}$, упорядоченный по возрастанию. Примечание. Исходные массивы упорядочены по возрастанию.

28

Написать программу циклического сдвига влево на K позиций одномерного массива из n элементов.

29
Найти сумму и произведение всех элементов массива $b_1,b_2,\ldots,b_m$.

30
Найти сумму положительных и число отрицательных элементов массива $a_1,a_2,\ldots,a_n$.

31
Написать программу вычисления суммы тех элементов целочисленного массива $a_1,a_2,\ldots,a_n$, которые являются удвоенными нечетными числами.

32
Даны 2 целочисленных массива $a_1,a_2,\ldots,a_n$ и $b_1,b_2,\ldots,b_n$. Написать программу вычисления величины $a_1b_1 + a_2b_2 + \ldots + a_nb_n$.

33
Написать программу вычисления величины $\sqrt{a_1^2 + a_2^2 + \ldots + a_n^2}$, если $a_1,a_2,\ldots,a_n$ – массив действительных чисел.

34
Написать программу, выясняющую, является ли массив $a_1, a_2,\ldots, a_m$ упорядоченным по убыванию.

35
Даны упорядоченный по возрастанию массив $y_1<y_2<\ldots<y_m$ и величина x. Известно, что $y_1< x <=y_m$. Написать программу определения целого числа k, удовлетворяющего условию:
$y_{k-1}< x <y_k$.

36
Написать программу вычисления числа положительных и суммы отрицательных элементов массива $x_1, x_2,\ldots, x_m$.

37
Дан вещественный массив $x_1, x_2,\ldots, x_n$. Определить сумму и количество компонент этого массива, принадлежащих отрезку $[a,b]$.

38
Дан вещественный вектор $x_1,x_2,\ldots,x_m$. Найти число компонент, предшествующих первой по порядку отрицательной компоненте, значения которых принадлежат отрезку $[c,d]$.

Пример программы на обработку одномерного массива

```
Program MinMaxMas;
{Последовательный поиск и перестановка
 минимального и максимального элементов в массиве
 Входные данные:  k - количество элементов в массиве,
                  M - массив из целых чисел.
 Выходное данное: M - преобразованный массив.}
Const R=10;       {Размер массива}
Type Tind=1..R;  {Тип индекса элемента массива}
     Tmas=Array [Tind] Of Integer; {Тип массив}
Var k,i,nmin,nmax:Tind;   {1..R}
    M:Tmas;    {Исходный и преобразованный массив}
    min,max:Integer;     {Текущий минимакс}
Begin
     {$R+}  {Установка режима контроля индекса элемента}
     Write('Задайте количество элементов не более ',R,': ');
     ReadLn(k);
{Ввод массива}
     Write('Вводите ',k,' целых чисел одной строкой:');
     For i:=1 To k
     Do Read(M[i]);      {Ввод элемента массива}
{Поиск минимума и максимума в массиве}
     min:=M[1]; nmin:=1; {Начальные установки минимума}
     max:=M[1]; nmax:=1; {Начальные установки максимума}
     For i:=2 To k       {Перебор элементов массива}
     Do If M[i]<min      {Сравнение элемента с минимумом}
        Then Begin min:=M[i];      {Текущий минимум}
                   nmin:=i         {Номер минимума}
             End
        Else If M[i]>max {Сравнение элемента с максимумом}
             Then Begin max:=M[i]; {Текущий максимум}
                        nmax:=i    {Номер максимума}
                  End;
{Перестановка минимума и максимума}
    M[nmin]:=max; M[nmax]:=min;
{Вывод массива}
    WriteLn('Массив после перестановки:');
    For i:=1 To k
    Do Write (M[i],' '); {Вывод элемента массива}
    WriteLn;
End.
```

Лабораторная работа №5. Подпрограммы

Объем в часах: аудиторных занятий - 3, самостоятельных - 4.

Цель лабораторной работы:

изучение концепций, освоение технологии, приобретение навыков процедурного программирования на языке Турбо Паскаль;

приобретение навыков процедурной реализации дружественного оконного интерфейса пользователя, применение подпрограмм разных видов при решении задач обработки матриц.

Задание на программирование

Используя технологию процедурного программирования, разработать процедурную реализацию интерфейса, обеспечивающего заданное расположение и назначение окон на экране при выполнении программы преобразования матрицы с использованием подпрограмм разных видов в соответствии с индивидуальным заданием.

Порядок выполнения работы

1. Получить у преподавателя индивидуальное задание:

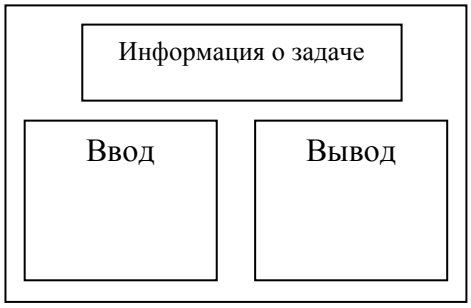
а) схему расположения и назначения окон на экране; б) задачу обработки матрицы.

2. Построить дерево подзадач и на его основе структурную диаграмму программы для решения индивидуальной задачи.
3. Составить алгоритмы подпрограмм: создания окна, ввода матрицы, вывода матрицы, поиска элементов в матрице, перестановки элементов матрицы.
4. Составить описания на языке Турбо Паскаль подпрограмм разных видов: процедур и функций. Обосновать вид и список параметров каждой подпрограммы.
5. Составить текст программы с подпрограммами. Раздел операторов программы должен содержать только вызовы подпрограмм.
6. Входные данные вводить с клавиатуры по запросу в окне ввода данных. Выходные данные выводить на экран в окнах вывода результатов.
7. Отладить программу, проверить ее работу на полном наборе тестов.
8. Продемонстрировать преподавателю несколько вариантов выполнения, в том числе с ошибочными данными. Обеспечить одновременный показ в окнах на экране входной и выходной матриц в одном и том же формате.
9. Оформить отчет о лабораторной работе в составе: *постановка задачи, математическая модель, структура программы, текст программы, контрольные примеры*

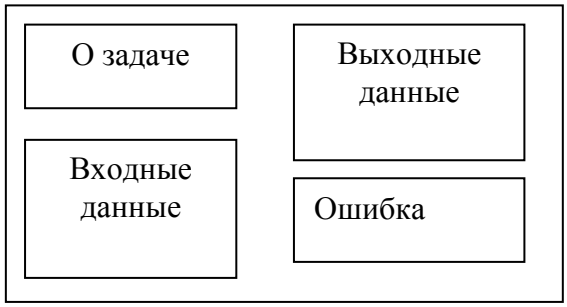
Варианты индивидуальных заданий

Расположение окон

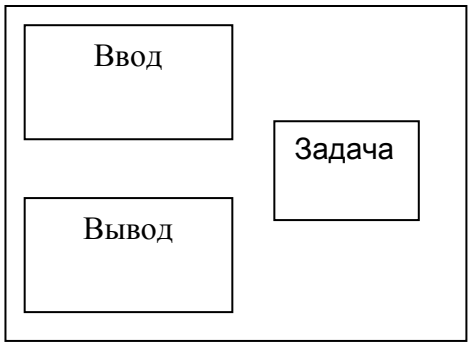
1 Расположение окон на экране:



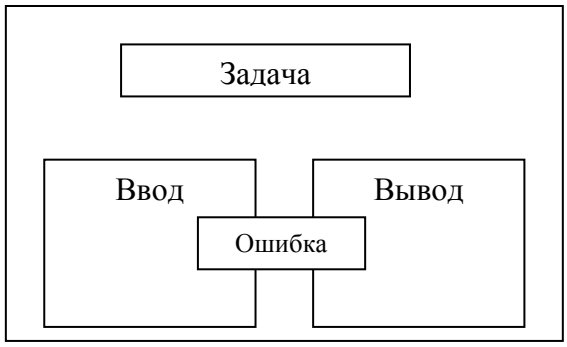
2 Расположение окон на экране:



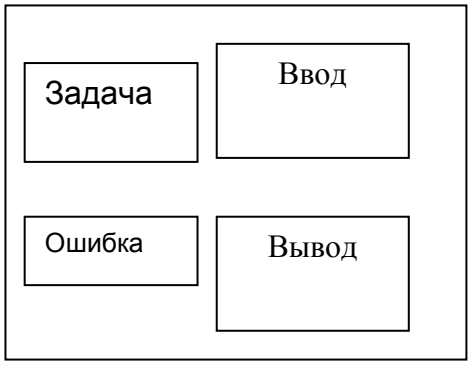
3 Расположение окон на экране:



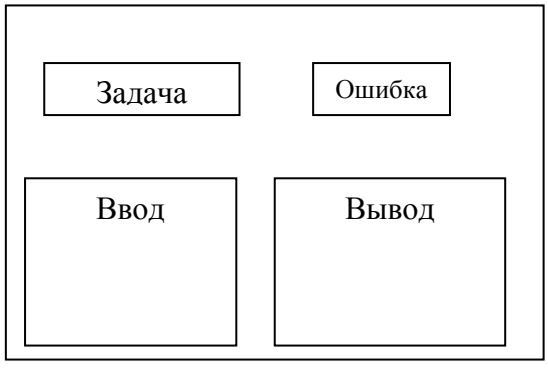
4 Расположение окон на экране:



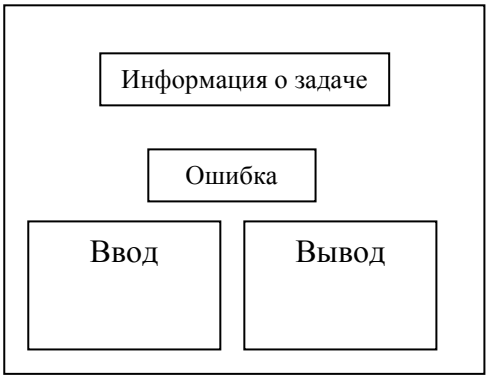
5 Расположение окон на экране:



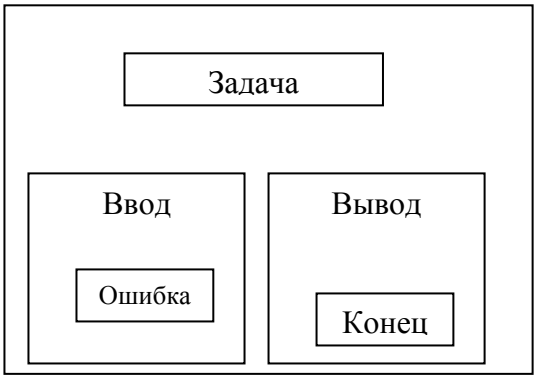
6 Расположение окон на экране:



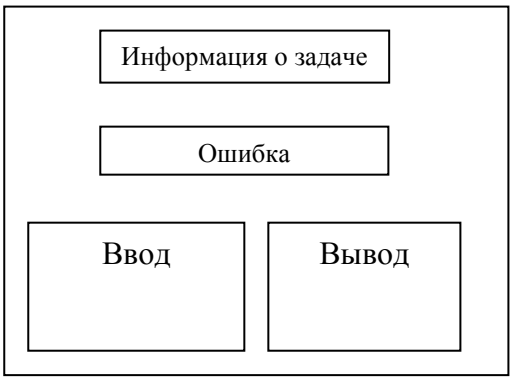
7 Расположение окон на экране:



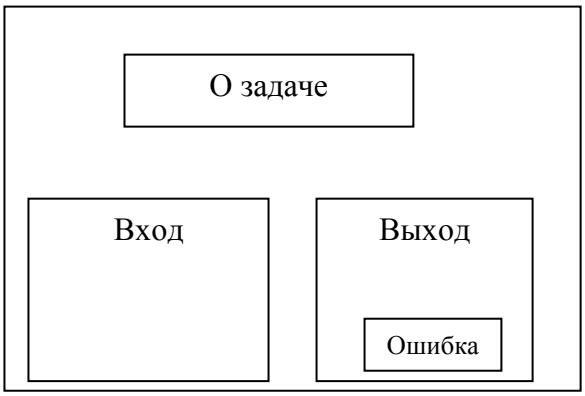
8 Расположение окон на экране:



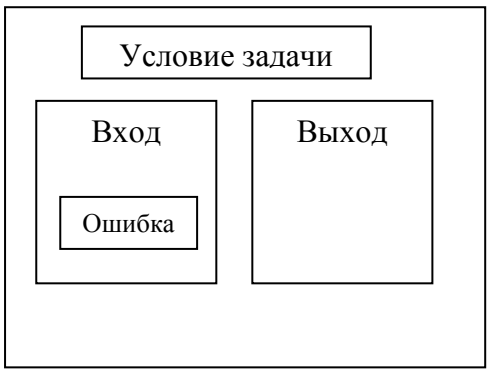
9 Расположение окон на экране:



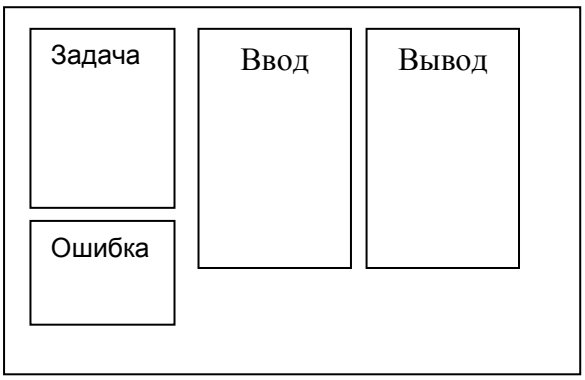
10 Расположение окон на экране:



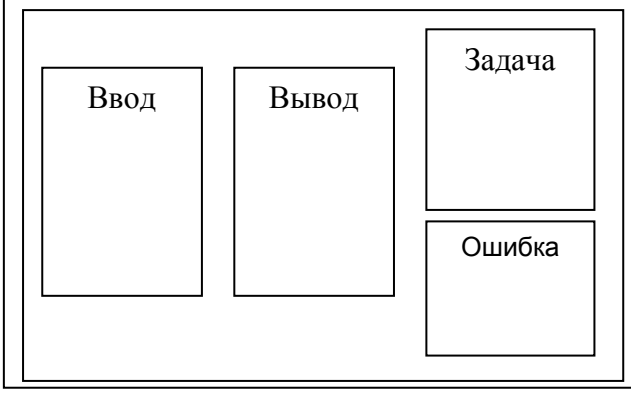
11 Расположение окон на экране:



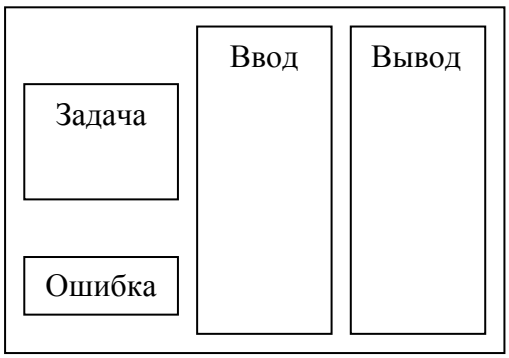
12 Расположение окон на экране:



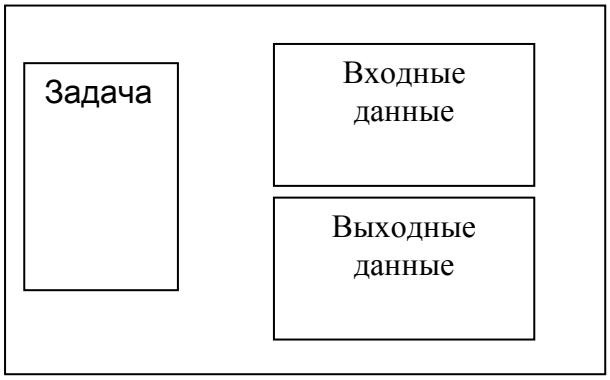
13 Расположение окон на экране:



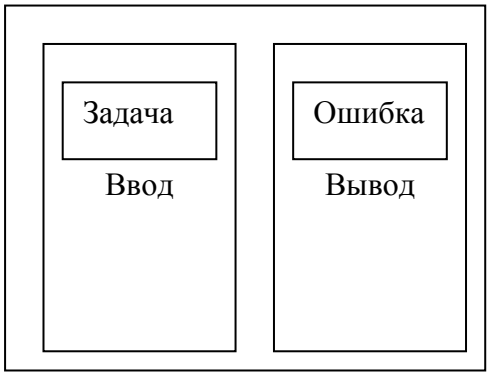
14 Расположение окон на экране:



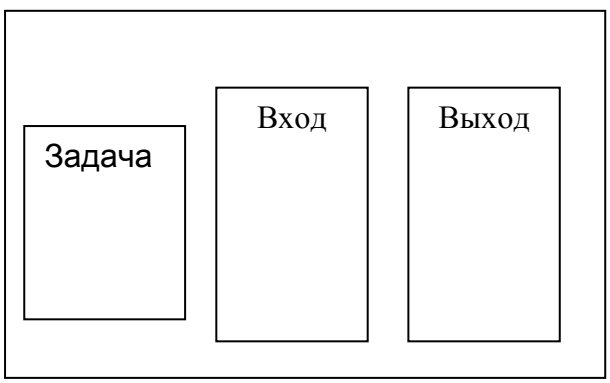
15 Расположение окон на экране:



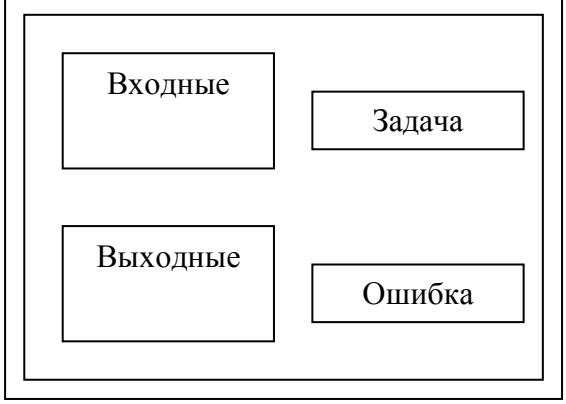
16 Расположение окон на экране:



17 Расположение окон на экране:



18 Расположение окон на экране:



Матрицы

1

В заданной матрице из целых чисел поменять местами первую строку и строку, содержащую наибольший по абсолютной величине элемент матрицы.

2

В заданной матрице из вещественных чисел поменять местами последний столбец и столбец, содержащий наименьший элемент матрицы.

3

В заданной матрице из неотрицательных вещественных чисел поменять местами первый столбец со столбцом, содержащим наибольший элемент матрицы.

4

В заданной матрице из целых отрицательных чисел поменять местами последнюю строку со строкой, содержащей наибольший элемент матрицы.

5

В заданной матрице из вещественных чисел поменять местами последний столбец со столбцом, содержащим наибольший по абсолютной величине элемент матрицы.

6

В заданной матрице из целых чисел поменять местами последнюю строку со строкой, содержащей наименьший положительный элемент матрицы.

7

В заданной матрице из положительных вещественных чисел поменять местами два столбца: столбец, содержащий максимальный элемент матрицы, и столбец, содержащий минимальный элемент матрицы.

8

В заданной матрице из неотрицательных целых чисел поменять местами первую строку и строку, содержащую максимальный элемент матрицы.

9

В заданной матрице из вещественных чисел поменять местами последнюю строку со строкой, содержащей минимальный положительный элемент матрицы.

10

В заданной матрице из целых чисел поменять местами последний столбец и столбец, содержащий минимальный по абсолютной величине элемент матрицы.

11

В заданной матрице из отрицательных вещественных чисел поменять местами последнюю строку со строкой, содержащей максимальный элемент матрицы.

12
В заданной матрице из целых чисел поменять местами минимальный элемент главной диагонали и максимальный элемент побочной диагонали.

13
В заданной матрице из положительных вещественных чисел поменять местами два столбца: столбец, содержащий максимальный элемент матрицы, и столбец, содержащий минимальный элемент матрицы.

14
В заданной матрице из целых чисел поменять местами первую строку и строку, содержащую наибольший отрицательный элемент матрицы.

15
В заданной матрице из вещественных чисел поменять местами последний столбец и столбец, содержащий наименьший положительный элемент матрицы.

16
В заданной матрице из целых чисел поменять местами два столбца: столбец, содержащий максимальный отрицательный элемент матрицы, и столбец, содержащий минимальный положительный элемент матрицы.

17
В заданной матрице из целых положительных чисел поменять местами две строки: строку, содержащую максимальный элемент матрицы, и строку, содержащую минимальный элемент матрицы.

18
В заданной матрице из символов поменять местами максимальный элемент главной диагонали и минимальный элемент побочной диагонали.

19
В заданной матрице из вещественных чисел поменять местами последний столбец и столбец, содержащий минимальный по абсолютной величине элемент матрицы.

20
В заданной матрице из отрицательных целых чисел поменять местами первую строку со строкой, содержащей максимальный элемент матрицы.

Пример программы с подпрограммами

```
Program MatrInt;
{Программа поиска и перестановки в матрице двух строк:
 строки с минимальным и строки с максимальным элементом}
Uses Crt;          {Подключение модуля}

Const R=10;        {Размер строки (столбца) матрицы}
Type Tind=1..R;    {Тип индекса элемента матрицы}
     Tvect=Array[Tind] Of Integer; {Тип вектор целых чисел}
     Tmatr=Array[Tind] Of Tvect;   {Тип матрица целых чисел}
{$R+}
Procedure InMatr(kStr,kStb:Tind;Var M:Tmatr);
{Процедура ввода матрицы.
 Входные данные:   kStr - количество строк матрицы,
                   kStb - количество столбцов матрицы.
 Выходное данное:     M - матрица.}
Var i,j:Tind;       {Локальные переменные}
Begin  WriteLn('Вводите матрицу по строкам:');
       For i:=1 To kStr
       Do Begin
            For j:=1 To kStb
            Do Read(M[i,j]);
            ReadLn;
          End;
End;{InMatr}

Function NomMin(kStr,kStb:Tind; Const M:Tmatr):Tind;
{Функция определения номера строки с минимальным элементом.
 Входные данные:   kStr - количество строк матрицы,
                   kStb - количество столбцов матрицы,
                      M - матрица.
 Выходное данное: NomMin - номер минимального элемента.}
Var i,j,nmin:Tind; min:Integer;   {Локальные переменные}
Begin
     min:=M[1,1]; nmin:=1;
     For i:=1 To kStr
     Do For j:=1 To kStb
        Do If M[i,j]<min
           Then Begin min:=M[i,j];
                      nmin:=i;
                End;
     NomMin:=nmin;
End; {NomMin}
```

```
Function NomMax(kStr,kStb:Tind; Const M:Tmatr):Tind;
{Функция определения номера строки с максимальным элементом.
 Входные данные:    kStr - количество строк матрицы,
                    kStb - количество столбцов матрицы,
                       M - матрица.
 Выходное данное: NomMax - номер максимального элемента.}
Var i,j,nmax:Tind; max:Integer;    {Локальные переменные}
Begin
     max:=M[1,1]; nmax:=1;
     For i:=1 To kStr
     Do For j:=1 To kStb
        Do If M[i,j]>max
           Then Begin max:=M[i,j];
                      nmax:=i;
                End;
     NomMax:=nmax;
End; {NomMax}

Procedure ObmenStr(kStr,kStb:Tind; Var M:Tmatr);
{Процедура перестановки строк с минимальным и максимальным
элементом.
 Входные данные:   kStr - количество строк матрицы,
                   kStb - количество столбцов матрицы,
                      M - матрица целых чисел.
 Выходное данное:     M - преобразованная матрица.}
Var strM:Tvect; nmin,nmax:Tind;     {Локальные переменные}
Begin
     nmin:=NomMin(kStr,kStb,M);
     nmax:=NomMax(kStr,kStb,M);
     If nmin<>nmax
     Then Begin strM:=M[nmin];
                M[nmin]:=M[nmax];
                M[nmax]:=strM;
          End;
End; {ObmenStr}

Procedure Okno(x1,y1,x2,y2,cf,ct:Byte);
{Процедура формирования окна}
Begin
      Window(x1,y1,x2,y2); {Установка параметров окна}
      TextBackGround(cf);  {Установка цвета фона}
      TextColor(ct);       {Установка цвета текста}
      ClrScr;              {Очистка окна}
End;{Okno}
```

```
Procedure OutMatr(kStr,kStb:Tind; Const M:Tmatr);
{Процедура вывода матрицы.
 Входные данные:   kStr - количество строк  матрицы,
                   kStb - количество столбцов  матрицы,
                      M - матрица.}
Var i,j:Tind;      {Локальные переменные}
Begin
      For i:=1 To  kStr
      Do Begin For j:=1 To kStb
               Do Write(M[i,j]:3);
               WriteLn;
         End;
End;{OutMatr}

Var n,m,nStb:Tind; Matr:Tmatr;
Begin
      Okno(1,1,80,25,0,15);     {На черном фоне белый текст}
      Write('Размеры матрицы? '); ReadLn(n,m);
      Okno(1,6,38,20,2,15);     {На зеленом фоне белый
текст}
      InMatr(n,m,Matr);         {Ввод матрицы}
      WriteLn('Исходная матрица');
      OutMatr(n,m,Matr);        {Вывод матрицы}
      ObmenStr(n,m,Matr);       {Перестановка строк}
      Okno(40,6,80,20,3,15);    {На голубом фоне белый
текст}
      WriteLn('Матрица после перестановки строк');
      OutMatr(n,m,Matr);        {Вывод матрицы}
      ReadLn;
End.
```

Лабораторная работа №6 Текстовые файлы

Объем в часах: аудиторных занятий - 3, самостоятельных - 4.

Цель лабораторной работы:

изучение структурной организации, способов доступа к элементам и других особенностей текстовых файлов;

изучение стандартных средств Турбо Паскаля для работы с множествами, строками и текстовыми файлами;

приобретение навыков работы с текстовыми файлами;

совершенствование навыков процедурного программирования при решении задач редактирования текстовых файлов.

Задание на программирование

Используя технологию процедурного программирования разработать программу обработки текстовых файлов с числом строк не менее 5 в соответствии с индивидуальным заданием. При обработке строк необходимо использовать множества. Массивы не использовать!

Порядок выполнения лабораторной работы

1. Получить у преподавателя индивидуальное задание.
2. Построить дерево подзадач и на его основе структурную диаграмму программы для решения индивидуальной задачи.
3. Описать и использовать подпрограмму обработки строки, подпрограммы проверки существования, создания, просмотра и редактирования текстового файла. Обосновать вид и список параметров каждой подпрограммы. Использовать оконный интерфейс предыдущей лабораторной работы.
4. Составить текст программы с подпрограммами. Раздел операторов программы должен содержать только вызовы подпрограмм.
5. При демонстрации выполнения программы обеспечить одновременный показ в окнах на экране исходного и отредактированного файлов.
6. Оформить отчет о лабораторной работе в составе: *постановка задачи, математическая модель, структура программы, текст программы, контрольные примеры*

Варианты индивидуальных заданий

1

Дана строка. Словом текста является последовательность букв русского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова, в которых гласные буквы алфавита образуют симметричную последовательность букв (палиндром). Все остальные слова удалить. Малые и большие буквы алфавита считать эквивалентными.

2

Дана строка. Словом текста является последовательность цифр; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова, в которых все четные цифры образуют неубывающую последовательность чисел. Все остальные слова удалить. Одну цифра не считать неубывающей последовательностью.

3

Дана строка. Словом текста является последовательность цифр и букв латинского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова, в которых цифры и буквы латинского алфавита чередуются. Все остальные слова удалить.

4

Дана строка. Словом текста считается любая последовательность цифр и букв латинского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова текста, в которых есть хотя бы одна цифра. Все остальные слова удалить.

5

Дана строка. Словом текста считается любая последовательность букв русского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова текста, которые содержат только большие буквы алфавита. Все остальные слова удалить.

6

Дана строка. Словом текста считается любая последовательность цифр; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова текста, которые образованы неубывающей последовательностью символов. Все остальные слова удалить.

7
Дана строка. Словом текста считается любая последовательность букв русского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова, символы которых образуют симметричную последовательность букв (палиндром). Все остальные слова удалить. Большие и малые буквы алфавита считать эквивалентными.

8
Дана строка. Словом текста считается любая последовательность букв русского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Удалить из строки те слова, которые содержат двойные согласные буквы.

9
Дана строка. Словом текста считается любая последовательность цифр; между соседними словами - не менее одного пробела, за последним словом – точка. Поменять местами в строке первое и последнее слово.

10
Дана строка. Словом текста считается любая последовательность букв русского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова текста, которые содержат одинаковое количество гласных и согласных букв алфавита. Все остальные слова удалить.

11
Дана строка. Словом текста считается любая последовательность букв русского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова текста, количество гласных букв в которых превышает количество согласных. Все остальные слова удалить.

12
Дана строка. Словом текста считается любая последовательность букв латинского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова, которые начинаются с прописной буквы. Все остальные слова удалить.

13
Дана строка. Словом текста считается любая последовательность букв русского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова последовательности, которые отличны от последнего слова и являются симметричными. Все остальные слова удалить.

14
Дана строка. Словом текста считается любая последовательность букв латинского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова последовательности, которые отличны от первого слова и удовлетворяют следующему свойству: первая буква слова входит в него еще один раз. Все остальные слова удалить.

15
Дана строка. Словом текста считается любая последовательность букв латинского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова последовательности, которые отличны от последнего слова и удовлетворяют следующему свойству: слово совпадает с начальным отрезком латинского алфавита (a, ab, abc, abcd,…). Все остальные слова удалить.

16
Дана строка. Словом текста считается любая последовательность букв латинского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова последовательности, которые отличны от первого слова и удовлетворяют следующему свойству: слово совпадает с конечным отрезком латинского алфавита (z, yz, xyz,…). Все остальные слова удалить.

17
Дана строка. Словом текста считается любая последовательность букв русского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова последовательности, которые отличны от последнего слова и удовлетворяют следующему свойству: в слове нет повторяющихся букв. Все остальные слова удалить.

18
Дана строка. Словом текста считается любая последовательность букв латинского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова последовательности, которые отличны от первого слова и удовлетворяют следующему свойству: каждая буква входит в слово не менее двух раз. Все остальные слова удалить.

19
Дана строка. Словом текста считается любая последовательность букв русского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова последовательности, которые отличны от последнего слова и удовлетворяют следующему свойству: в слове гласные буквы чередуются с согласными. Все остальные слова удалить.

20
Дана строка. Словом текста считается любая последовательность букв латинского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова последовательности, которые отличны от первого слова, предварительно преобразовав каждое из них по следующему правилу: перенести первую букву в конец слова. Все остальные слова удалить.

21
Дана строка. Словом текста считается любая последовательность букв русского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова последовательности, которые отличны от последнего слова, предварительно преобразовав каждое из них по следующему правилу: перенести последнюю букву в начало слова. Все остальные слова удалить.

22
Дана строка. Словом текста считается любая последовательность букв латинского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова последовательности, которые отличны от первого слова, предварительно преобразовав каждое из них по следующему правилу: удалить из слова первую букву. Все остальные слова удалить.

23
Дана строка. Словом текста считается любая последовательность букв русского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова последовательности, которые отличны от последнего слова, предварительно преобразовав каждое из них по следующему правилу: удалить из слова последнюю букву. Все остальные слова удалить.

24
Дана строка. Словом текста считается любая последовательность букв латинского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова последовательности, которые отличны от первого слова, предварительно преобразовав каждое из них по следующему правилу: удалить из слова все последующие вхождения первой буквы. Все остальные слова удалить.

25
Дана строка. Словом текста считается любая последовательность букв русского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова последовательности, которые отличны от последнего слова, предварительно преобразовав каждое из них по следующему правилу: удалить из слова все предыдущие вхождения последней буквы. Все остальные слова удалить.

26
Дана строка. Словом текста считается любая последовательность букв латинского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова последовательности, которые отличны от первого слова, предварительно преобразовав каждое из них по следующему правилу: оставить в слове только первые вхождения каждой буквы. Все остальные слова удалить.

27
Дана строка. Словом текста считается любая последовательность букв русского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова последовательности, которые отличны от последнего слова, предварительно преобразовав каждое из них по следующему правилу: если слово нечетной длины, то удалить его среднюю букву. Все остальные слова удалить.

28
Дана строка. Словом текста считается любая последовательность букв латинского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Разместить в строке последовательность ее слов в обратном порядке.

29
Дана строка. Словом текста считается любая последовательность букв русского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова, перед которыми в последовательности находятся только меньшие (по алфавиту) слова, а за ними только большие. Все остальные слова удалить.

30
Дана строка. Словом текста считается любая последовательность букв латинского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Сохранить в строке последовательность слов, удалив из нее повторные вхождения слов.

31

Дана строка. Словом текста считается любая последовательность букв русского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова, которые встречаются в последовательности по одному разу. Все остальные слова удалить.

32

Дана строка. Словом текста считается любая последовательность букв латинского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Разместить слова строки в алфавитном порядке.

Пример программы на обработку текстовых файлов

```
Program FileText;
{Лексикографическая сортировка строк текстового файла}
Uses Crt; {Подключение стандартного модуля Crt}

Procedure Exist(Var nameFT:String);
{Проверка существования файла с таким именем}
Var ch:Char;
    FT:Text;
Begin
  Assign(FT,nameFT);
  {$I-}   {Отключение контроля ошибок ввода-вывода}
  Reset(FT);
  {$I+}   {Включение контроля ошибок ввода-вывода}
  If IOResult=0
  Then Begin
         WriteLn('Такой файл уже существует!');
         Write('Хотите его уничтожить?  Y/N ->');
         ReadLn(ch);
         If ch In ['n','N','т','Т']
         Then Repeat
                WriteLn('Введите другое имя:');
                ReadLn(nameFT);
                Assign(FT,nameFT);
                {$I-}
                Reset(FT);
                {$I+}
                If IOResult=0
                Then Begin
                       WriteLn('Такой файл уже существует!');
                       Write('Хотите его уничтожить? Y/N->');
                       ReadLn(ch);
                     End;
              Until (IOResult<>0)Or(ch In['y','Y','н','Н']);
       End;
End;

Procedure SozdFT(Const nameFT:String);
{Создание текстового файла}
Var
  FT:Text;
   i:Byte;
  st:String;
Begin
```

```
  Assign(FT,nameFT);
  ReWrite(FT); {Открытие файла для записи}
  Write('Начинаем ввод. ');
  WriteLn('Признак окончания ввода-пустая строка.');
  i:=0;
  WriteLn('Введите ',i+1,'-ую строку файла:');
  ReadLn(st);           {Ввод строки с клавиатуры}
  While st<>''          {Если строка не пустая}
  Do Begin
       WriteLn(FT,st);  {Запись строки в файл}
       Inc(i);
       WriteLn('Введите ',i+1,'-ую строку файла:');
       ReadLn(st);
     End;
  WriteLn('Введено ',i,' строк');
  Close(FT);            {Закрытие файла}
End;



Procedure ProsmFT(Const nameFT:String);
{Процедура просмотра текстового файла}
Var st:String;
    FT:Text;
Begin
  Assign(FT,nameFT);
  Reset(FT);          {Открытие файла для чтения}
  If Eof(FT) Then Begin
                    Writeln('Файл пуст !');

                    WriteLn('Нажмите Enter ->');
                    ReadLn; Halt;
                  End;
  Writeln(' содержимое файла:'); Writeln;
  While Not Eof(FT)         {Пока не конец файла:}
  Do Begin
       Readln(FT,st);   {чтение строки из файла}
       Writeln(st);     {вывод строки на экран}
     End;
  Writeln;
  Close(FT);                {Закрытие файла}
End; {ProsmFT}

Procedure RedaktFT(Const nameF1,nameF2:String);
{Процедура редактирования текстового файла.
```

```
Входные данные:  F1 - исходный текстовый файл.
Выходные данные: F2 - отредактированный текстовый файл.}

  Procedure UdalLP(Var st:String);
  {Процедура удаления лишних пробелов.
  Входное данное: st-строка из слов, разделенных пробелами.
  Выходное данное: st - отредактированная строка.}
  Var L,i:Byte;
  Begin
    L:=Length(st); {Номер последнего символа строки}
    i:=1;          {Текущий номер символа строки}
    While i<=L     {Пока текущий номер в пределах строки}
    Do Begin
         If st[i]=' '  {Если текущий символ - пробел}
         Then Begin
                If (i=1) Or (i=L)   {Удаление пробела}
                Then Delete(st,i,1)
                Else If (i<L) And (st[i+1]=' ')
                     Then Delete(st,i+1,1)
                     Else i:=i+1; {Текущий номер символа}
                L:=Length(st);    {Новая длина строки}
              End
         Else i:=i+1; {Новый текущий номер символа}
       End;
  End; {UdalLP}

  Procedure LexSort(Var st:String);
  {Процедура сортировки слов строки по алфавиту.
  Входное данное: st-строка из слов, разделенных пробелами.
  Выходное данное: st - упорядоченная строка из слов.}

      Function Slovo(pn:Byte; st:String):String;
      {Функция определения очередного слова строки.
    Входные данные: pn - начальная позиция слова в строке,
                    st - строка из слов.
    Выходное данное: Slovo - очередное слово строки.}
       Var L,p:Byte;
       Begin
         L:=Length(st); p:=pn;
         {Цикл поиска очередного пробела}
         While (p<=L)And(st[p]<>' ') {Пока не конец слова}
         Do p:=p+1;         {изменение позиции в строке}
         Slovo:=Copy(st,pn,p-pn);         {Выделение слова}
       End;{Slovo}
```

```
   Var L,p1,p2:Byte;
      {L - длина строки,
       p1,p2 - начальные позиции соседних слов в строке}
       sl1,sl2:String;{Соседние слова в строке}
       flag:Boolean;  {Флаг перестановки слов}
   Begin
     L:=Length(st); {Определение длины строки}
     Repeat
       flag:=FALSE; {Нет перестановки}
       p1:=1;  sl1:=Slovo(p1,st); {Первое слово}
       While p1+Length(sl1)<L {Пока sl1 не последнее}
         Do Begin
              p2:=p1+Length(sl1)+1;{Позиция 2-го слова}
              sl2:=Slovo(p2,st);   {Второе слово}
              If sl2<sl1
              Then Begin           {Обмен соседних слов}
                     Delete(st,p1,p2+Length(sl2)-p1);
                     Insert(sl2+' '+sl1,st,p1);
                     p1:=p1+Length(sl2)+1;
                     flag:=TRUE;   {Есть перестановка}
                   End
              Else Begin {Переход к очередной паре слов}
                     sl1:=sl2; {Новое первое слово}
                     p1:=p2;   {Новая позиция слова}
                   End;
            End;
     Until Not flag; {До отсутствия перестановок}
   End; {LexSort}

Var F1,F2:Text;
       st:String;
Begin
      Assign(F1,nameF1); Assign(F2,nameF2);
      Reset(F1); Rewrite(F2);{Открытие файлов}
      While Not Eof(F1)   {Пока не конец входного файла}
      Do Begin
           ReadLn(F1,st); {Прочитать строку,}
           UdalLP(st);    {Удалить лишние пробелы,}
           LexSort(st);   {Сортировать слова по алфавиту,}
           WriteLn(F2,st);{Записать строку в другой файл.}
         End;
      Close(F1); Close(F2);      {Закрытие файлов}
End; {RedaktFT}
```

```
{Основная программа}
Var nameF1,nameF2:String; {Имена текстовых файлов}
Begin
  ClrScr;              {Очистка экрана}
  Write('Введите имя исходного файла: ');
  ReadLn(nameF1);
  Exist(nameF1);
  SozdFT(nameF1);      {Создание исходного файла}
  Write('После создания ');      ProsmFT(nameF1);
  Write('Введите имя результирующего файла: ');
  ReadLn(nameF2);
  Exist(nameF2);
  RedaktFT(nameF1,nameF2); {Редактирование файла}
  Write('После редактирования '); ProsmFT(nameF2);
  ReadLn;
End. {FileText}
```

Лабораторная работа №7. Файлы прямого доступа

Объем в часах: аудиторных занятий - 3, самостоятельных - 4.

Цель лабораторной работы:

изучение структурной организации, способов доступа к элементам и других особенностей типизированных файлов;

изучение стандартных подпрограмм библиотеки Турбо Паскаля для работы с типизированными файлами;

приобретение навыков работы с типизированными файлами;

совершенствование навыков процедурного программирования при решении задач обработки типизированных файлов.

Задание на программирование

Используя технологию процедурного программирования разработать программу обработки типизированных файлов с числом записей не менее 5, структуру которых определяет индивидуальное задание. Вспомогательные файлы и массивы не использовать!

Порядок выполнения лабораторной работы

1. Получить у преподавателя индивидуальное задание.
2. Построить дерево подзадач и на его основе структурную диаграмму программы для решения индивидуальной задачи.
3. Сформулировать условие поиска данных в файле и организовать поиск по условию с сохранением найденных записей в новом файле.
4. Описать и использовать подпрограммы: проверки существования, создания, просмотра, сортировки файла, поиска данных в файле.
5. Предусмотреть в программе возможность выбора варианта выполнения программы с помощью меню в окне диалога с пользователем.
6. При демонстрации выполнения программы обеспечить одновременный показ в окнах на экране исходного и результирующего файла.
7. Оформить отчет о лабораторной работе в составе: *постановка задачи, математическая модель, структура программы, текст программы, контрольные примеры.*

Варианты индивидуальных заданий

1 Самолеты

Наименование типа	Фамилия конструктора	Год выпуска	Количество кресел	Грузоподъемность в Т

2 Расчет движения

Наименование воздушной линии	Тип самолета	Количество рейсов	Налет в тыс.км	Пассажирооборот в чел. км

3 Перевозки

Тип самолета	Номер борта	Количество рейсов	Налет в часах	Налет в тыс.км

4 Расписание

Номер Рейса	Наименование рейса	Тип самолета	Стоимость билета	Протяженность линии

5 Сооружения аэропорта

Наименование	Площадь	Этажность	Год сооружения	Стоимость в млн. руб.

6 Ремонт аэродромных сооружений

Наименование	Шифр	Вид ремонта	Стоимость ремонта	Наименование подрядчика

7 Кассы авиабилетов

Номер кассы	ФИО кассира	Количество проданных билетов	Суммарная выручка	Дата продажи

8 Характеристики ПК

Тип процессора	Тактовая частота	Емкость ОП в Мбайт	Емкость ЖМД в Мбайт	Тип монитора

9 Города

Наименование	Количество жителей	Площадь в кв.км	Год основания	Количество школ

10 Мосты

Наименование	Высота	Ширина	Количество опор	Протяженность

11 Программные продукты

Наименование	Фирма	Стоимость	Объем	Количество

12 Музеи

Наименование	Назначение	Адрес	Время работы	Стоимость билета

13 Экскурсии

Наименование	Страна	Стоимость	Продолжительность	Транспорт

14 Квартиры

Адрес	Площадь в кв.м	Сторона света	Стоимость 1 кв.м	Этаж	Колич. комнат

15 Кинотеатры

Наименование	Стоимость билета	Время сеансов	Адрес мест	Количество мест

16 Магазин

Наименование товара	Фирма изготовитель	Сорт	Цена	Размер партии

17 Театр

Наименование спектакля	Дата	Время	Место	Цена билета

18 Железная дорога

Пункт назначения	Поезд	Вагон	Место	Стоимость проезда

19 Библиотека

Название книги	Автор	Издание	Год издания	Количество экземпляров

20 Автоинспекция

Марка машины	Цвет	Номер	Год выпуска	Владелец

21 Метрополитен

Номер линии	Название линии	Число станций	Время стоянки	Время разворота

Пример программы на обработку файлов прямого доступа

```
Program FileZap;
{Формирование файла студентов}
Const MAX_DL=15; {Максимальная длина фамилии}
      KOL_OC=3;  {Количество оценок}
Type Tzap=Record
                fam:String[MAX_DL];          {Фамилия}
                OC:Array[1..KOL_OC] Of 0..5; {Оценки}
             End;
     Tfile=File Of Tzap;

Function FileExist(Const nameF:String):Boolean;
{Функция проверки существования файла.
 Входное данное:  nameF - физическое имя файла.
 Выходное данное: FileExist - логическое значение.}
Var F:File; {Любой вид файла}
Begin Assign(F,nameF);
      {$I-} {Отключение контроля чтения-записи}
      Reset(F); {Открытие файла для чтения}
      {$I+}  {Включение контроля чтения-записи}
      If IOResult=0  {Файл существует ?}
      Then Begin FileExist:=TRUE; Close(F); {Закрытие файла}
           End
      Else FileExist:=FALSE;
End; {FileExist}

Procedure SozdFZ(Const nameF:String);
{Процедура создания файла абитуриентов.
 Входное данное: nameF - физическое имя файла.}
Var z:Tzap; j:Byte; otv:Char; FZ:Tfile;
Begin If FileExist(nameF)  {Файл существует ?}
      Then Begin WriteLn('Файл ',nameF,' существует!');
                 Exit; {Выход из процедуры}
           End;
      Assign(FZ,nameF);
      Rewrite(FZ);     {Открытие файла для записи}
      WriteLn('Вводите данные:');
      Repeat Write('Фамилия ? '); ReadLn(z.fam);
             Write('Оценки ? ');  For j:=1 To KOL_OC
                                  Do Read(z.OC[j]); ReadLn;
             Write(FZ,z); {Запись элемента в файл}
             Write('Продолжить ввод ? '); ReadLn(otv);
      Until Not ( otv In ['Д','д','Y','y']);
```

```
      Close(FZ);      {Закрытие файла}
End; { SozdFZ }

Procedure ProsmFZ(Const nameF:String);
{Процедура просмотра файла из записей.
 Входное данное: nameF - физическое имя файла.}
Var z:Tzap; j:Byte; FZ:Tfile;
Begin If Not FileExist(nameF) {Файл не существует ?}
      Then Begin WriteLn('Файл ',nameF,' не существует!');
                 Exit;
           End;
      Assign(FZ,nameF);
      Reset(FZ); {Открытие файла для чтения}
      If Eof(FZ) {Конец файла ?}
      Then Begin WriteLn('Файл ',nameF,' пуст!');
                 Exit;
           End;
      WriteLn('Содержимое файла : ',nameF);
      While Not(Eof(FZ))   {Пока не конец файла}
      Do Begin Read(FZ,z); {Чтение элемента из файла}
               Write(z.fam,' ':MAX_DL-Length(z.fam)+1);
               For j:=1 To KOL_OC
               Do Write(z.OC[j]:3); WriteLn
         End;
      Close(FZ); {Закрытие файла}
End; {ProsmFZ}

Procedure SortFZ(Const nameF:String);
{Процедура сортировки файла абитуриентов по убыванию
 суммарного балла методом обмена с флагом перестановки.
 Входное данное: nameF - физическое имя файла.}

  Function SumB(z:Tzap):Byte;
{Функция определения суммарного балла.
 Входное данное:     z - текущая запись.
 Выходное данное: SumB - суммарный балл.}
  Var s,j:Byte;
  Begin s:=0;
        For j:=1 To KOL_OC
        Do s:=s+z.OC[j];
        SumB:=s
  End; {SumB}
```

```
Var z1,z2:Tzap; i:Word; flag:Boolean; FZ:Tfile;
    nkz:Word; {Номер конечной записи}
Begin
  If Not FileExist(nameF) {Файл не существует?}
  Then Begin WriteLn('Файл ',nameF,' не существует !');
             Exit;
       End;
  Assign(FZ,nameF);
  Reset(FZ);       {Открытие файла для чтения}
  nkz:=FileSize(FZ)-1; {Начальная длина файла}
  flag:=TRUE;          {Начальная установка флага}
  While (nkz>0) and flag {Пока файл не упорядочен}
  Do Begin
       flag:=FALSE;   {Нет перестановки}
       For i:=0 To nkz-1 {Перебор записей файла}
       Do Begin
            Seek(FZ,i);
            Read(FZ,z1,z2);{Чтение двух элементов файла}
            If SumB(z1)<SumB(z2)
            Then Begin {Обмен соседних элементов файла}
                   Seek(FZ,FilePos(FZ)-2);
                   Write(FZ,z2,z1);{Запись элементов в файл}
                   flag:=TRUE;     {Есть перестановка}
                 End;
          End; { For }
       nkz:=nkz-1;
     End; {While}
  Close(FZ)  {Закрытие файла}
End; {SortFZ}

Procedure UsechFZ(Const nameF:String;kz:Word);
{Процедура усечения файла.
 Входные данные: nameF - физическое имя файла,
                   kz - количество записей в файле.}
Var   FZ:Tfile;
Begin
  If Not FileExist(nameF)     {Файл не существует?}
  Then WriteLn('  UsechFZ: Файла ',nameF,' нет!')
  Else Begin
         Assign(FZ,nameF);
         Reset(FZ);         {Открытие файла для чтения}
         If kz<FileSize(FZ) {Усечение возможно?}
         Then Begin Seek(FZ,kz); {Установка маркера}
                    Truncate(FZ);   {Усечение файла}
                    Close(FZ);      {Закрытие файла}
```

```
              End;
        End;
End; {UsechFZ}

{Основная программа}
Var nameFZ      {Имя файла}
    kStud:Word; {Количество студентов}
    r:0..4;     {Режим работы программы}  {$R+}
Begin
  WriteLn{‘Введите имя файла:’}
  ReadLn(nameFZ);
  Repeat
    WriteLn('Задайте режим работы программы !');
    WriteLn('1 - создание, 2 - сортировка,');
    Write('3 - просмотр, 4 - усечение, 0 - выход : ');
    ReadLn(r);
    Case r Of
         1: SozdFZ(nameFZ);    {Создание файла}
         2: SortFZ(nameFZ);    {Сортировка файла}
         3: ProsmFZ(nameFZ);   {Просмотр файла}
         4: Begin Write('Количество студентов ? ');
                  ReadLn(kStud);
                  UsechFZ(nameFZ,kStud);  {Усечение файла}
            End;
    End; {Case}
    WriteLn;
  Until r=0;
End.
```

Лабораторная работа №8. Линейные списки

Объем в часах: аудиторных занятий - 3, самостоятельных - 4.

Цель лабораторной работы:

изучение способов создания и принципов использования односвязных линейных списков;

изучение стандартных подпрограмм библиотеки Турбо Паскаля для работы с динамической памятью.

Задание на программирование

Используя технологию процедурного программирования разработать программу обработки односвязных линейных списков с числом элементов не менее 5 в соответствии с индивидуальным заданием.

Порядок выполнения лабораторной работы

1. Получить у преподавателя индивидуальное задание.
2. Построить дерево подзадач и на его основе структурную диаграмму программы для решения индивидуальной задачи.
3. Описать и использовать подпрограммы инициализации, создания, просмотра, обработки списка.
4. При демонстрации выполнения программы обеспечить одновременный показ в окнах на экране исходных и результирующих списков.
5. Оформить отчет о лабораторной работе в составе: *постановка задачи, математическая модель, структура программы, текст программы, контрольные примеры.*

Варианты индивидуальных заданий

1
По списку L построить два новых списка L1 и L2: первый из положительных элементов, а второй из остальных элементов списка.

2
Вставить в список L новый элемент E1 за каждым вхождением заданного элемента E, если E входит в L.

3
Вставить в список L новый элемент E1 перед каждым вхождением элемента E, если E входит в L.

4
Вставить в непустой список L перед его последним элементом пару новых элементов E1 и E2.

5
Вставить в непустой список L, элементы которого упорядочены по не убыванию, новый элемент E так, чтобы сохранить упорядоченность списка.

6
Удвоить каждое вхождение элемента E в списке L.

7
Удалить из списка L все вхождения элемента E.

8
Удалить из списка L все отрицательные элементы.

9
Удалить из списка L за каждым вхождением элемента E один элемент, если он есть и отличен от E.

10
Оставить в списке L только первые вхождения одинаковых элементов.

11
В списке L из каждой группы подряд идущих равных элементов оставить только один.

12
Перевернуть список L, то есть изменить ссылки в этом списке так, чтобы его элементы оказались расположенными в обратном порядке.

13
Определить, входит ли список L1 в список L2.

14
Проверить, есть ли в списке L хотя бы два одинаковых элемента.

15
Проверить на равенство два списка L1 и L2.

16
Построить список L1 – копию списка L.

17
Добавить в конец списка L1 все элементы списка L2.

18
Вставить в список L за первым вхождением элемента E все элементы списка L1, если E входит в L.

19
Сформировать список L, включив в него по одному разу элементы, которые входят хотя бы в один из списков L1 и L2.

20
Сформировать список L, включив в него по одному разу элементы, которые входят одновременно в оба списка L1 и L2.

21
Сформировать список L, включив в него по одному разу элементы, которые входят в список L1, но не входят в список L2.

22
Сформировать список L, включив в него по одному разу элементы, которые входят в один из списков L1 и L2, но в то же время не входят в другой из них.

23
Объединить два упорядоченных списка L1 и L2 в один упорядоченный список, построив новый список L.

24
Объединить два упорядоченных списка L1 и L2 в один упорядоченный список L1, меняя соответствующим образом ссылки в L1 и L2.

25
Найти среднее арифметическое элементов непустого списка.

26
Поменять местами первый и последний элемент списка.

27
Проверить, упорядочены ли элементы списка по алфавиту.

28
Найти сумму последнего и предпоследнего элементов списка.

29
Вставить в начало списка новый элемент.

30
Вставить в конец списка новый элемент.

31
Вставить новый элемент после первого элемента непустого списка.

32
Удалить из непустого списка первый элемент.

33
Удалить из списка второй элемент, если такой есть.

34
Удалить из непустого списка последний элемент.

35
Удалить из списка первый отрицательный элемент, если такой есть.

36
Заменить в списке L все вхождения элемента E1 на E2.

37
Перенести в конец списка его первый элемент.

38
Перенести в начало списка его последний элемент.

39
Определить, входит ли элемент E в список L.

40
Подсчитать число вхождений элемента E в список L.

41
Найти максимальный элемент непустого списка.

42
Удалить из списка L первое вхождение элемента E, если такое есть.

43
Подсчитать количество слов списка, которые начинаются и оканчиваются одной и той же литерой.

44
Подсчитать количество слов списка, которые начинаются с той же литеры, что и следующее слово.

45
Подсчитать количество слов списка, которые совпадают с последним словом.

Пример программы обработки линейного списка

```
Program Spisok;
{Программа для работы с линейным списком}
Type Tinf=Char;    {Тип информации}
     TP=^Tel;      {Тип ссылка на элемент}
     Tel=Record    {Тип элемента списка}
            inf:Tinf;{Поле информации}
            pSled:TP;{Поле ссылки на следующий элемент}
         End;

Procedure Init(Var head:TP);
{Процедура инициализации списка}
Begin
  head:=NIL;
End;{Init}

Procedure InSpN(infEl:Tinf; Var head:TP);
{Процедура включения элемента в начало списка}
Var pn:TP; {Ссылка на новый элемент}
Begin
      New(pn); {Резервирование памяти для нового элемента}
      pn^.inf:=infEl; {Заполнение поля информации}
      pn^.pSled:=head;{Заполнение поля ссылки}
      head:=pn;       {Новая ссылка на головной элемент}
End;{InSpN}

Procedure SozdSpN(Var head:TP);
{Процедура создания списка из символов:
 обратное включение элементов}
Var s:Char;
Begin
      Init(head); {Инициализация пустого списка}
      WriteLn('Вводите символы! Признак конца ввода * ');
      Repeat
        Read(s);
        InSpN(s,head) {Включение элемента в начало списка}
      Until s='*';
      ReadLn;
End;{SozdSpN}
```

```
Procedure InitZ(infEl:Tinf; Var head:TP);
{Процедура создания заглавного элемента списка}
Begin
      New(head); {Резервирование памяти для нового элемента}
      head^.inf:=infEl; {Заполнение поля информации}
      head^.pSled:=NIL; {Заполнение поля ссылки}
End;{InitZ}

Procedure InSpP(infEl:Tinf; Var pred:TP);
{Процедура включения элемента после указанного}
Var pn:TP; {Ссылка на новый элемент}
Begin
  If pred<>NIL  {Предыдущий элемент существует?}
  Then Begin
         New(pn);
         pn^.inf:=infEl; {Заполнение поля информации}
         pn^.pSled:=pred^.pSled; {Заполнение поля ссылки}
         pred^.pSled:=pn; {Обновление поля ссылки}
         pred:=pn;
       End;
End;{InSpP}

Procedure SozdSpP(Var head:TP);
{Процедура создания списка с заглавным элементом:
 прямое включение элементов}
Var s:Char; pred:TP;
Begin
      InitZ(' ',head);
      pred:=head; {Установка ссылки на головной элемент}
      WriteLn('Вводите символы! Признак конца ввода * ');
      Repeat
        Read(s);
        If s<>'*'
        Then InSpP(s,pred){Включение элемента в список}
      Until s='*';
      ReadLn
End;{SozdSpP}

Procedure ProsmSp(head:TP);
{Процедура просмотра списка}
Var pt:TP; {Ссылка на текущий элемент}
Begin
  If head=NIL
  Then Begin
         WriteLn('Список пуст!');
```

```
          Write('Нажмите Enter ->');
          ReadLn; Exit;
        End;
  WriteLn('элементы списка:');
  pt:=head; {Установка ссылки на головной элемент}
  While pt<>NIL
  Do Begin
       Write(pt^.inf);
       pt:=pt^.pSled;{Установка ссылки на следующий элемент}
     End;
  WriteLn;
End; {ProsmSp}

Procedure OutSpN(Var head:TP);
{Процедура исключения из списка начального элемента}
Var pu:TP; {Ссылка на удаляемый элемент}
Begin
  If head<>NIL
  Then Begin
         pu:=head; {Установка ссылки на головной элемент}
         head:=head^.pSled;
         Dispose(pu);
       End;
End; {OutSpN}

{Основная программа}
Var head:TP;
Begin
       SozdSpN(head); {Прямое включение элементов}
       Write('После создания '); ProsmSp(head);
       OutSpN(head);  {Исключение начального элемента}
       Write('После удаления '); ProsmSp(head);
       SozdSpP(head); {Обратное включение элементов}
       Write('После создания '); ProsmSp(head);
       ReadLn;
End.
```

5 Методические указания к выполнению контрольных работ

Контрольные работы проводятся для закрепления и проверки текущих знаний студентов, не контролируемых другими способами.

Контрольные работы выполняются в соответствии с индивидуальным заданием и оформляются на стандартных листах бумаги формата А4. Форма титульного листа контрольной работы приведена в приложении.

Тема контрольной работы №1: Строки

Порядок выполнения работы

1. В соответствии с индивидуальным заданием на обработку строки выполнить постановку задачи: сформулировать условие, определить входные и выходные данные, их ограничения.
2. Разработать математическую модель: описать с помощью формул и рисунков структуру строки и процесс ее преобразования.
3. Построить дерево подзадач для выполнения индивидуального задания и на его основе структурную диаграмму программы.
4. Разработать алгоритм обработки строки.
5. Описать подпрограмму обработки строки.
6. Составить программу с подпрограммами на языке Турбо Паскаль.
7. Программа должна предусматривать ввод входных данных с клавиатуры по запросу. Выходные данные должны выводиться на экран с пояснениями**.**
8. Оформить отчет о контрольной работе в составе: *постановка задачи, математическая модель, структура программы, текст программы, контрольные примеры***.**

Задание на контрольную работу №1

Индивидуальное задание на контрольную работу выбирается из списка заданий на лабораторную работу №6 в соответствии с двумя последними цифрами номера зачетной книжки студента по формуле

$$V = D \bmod 32 + 1$$

где V – номер варианта контрольной работы;

D – две последние цифры зачетной книжки студента;

mod – операция получения остатка от деления.

Например, если номер зачетной книжки 03467,

то $V = 67 \bmod 32 + 1 = 3 + 1 = 4$, и значит, задание имеет вид:

4

Дана строка. Словом текста считается любая последовательность цифр и букв латинского алфавита; между соседними словами - не менее одного пробела, за последним словом – точка. Найти и сохранить в строке те слова текста, в которых есть хотя бы одна цифра. Все остальные слова удалить.

Пример программы на обработку строк

```
Program LexSortStr;
{Лексикографическая сортировка строки}

Procedure LexSort(Var st:String);
{Процедура сортировки слов по алфавиту.
 Входное данное:  st - строка текста из слов.
 Выходное данное: st - упорядоченная строка из слов.}

  Function Slovo(pn:Byte; st:String):String;
{Функция определения очередного слова строки.
 Входные данные: pn - начальная позиция слова в строке,
                 st - строка из слов.
 Выходное данное: Slovo - очередное слово строки.}
  Var L,p:Byte;
  Begin L:=Length(st); p:=pn;
        {Цикл поиска очередного пробела}
        While (p<=L)And(st[p]<>' ') {Пока не конец слова}
        Do p:=p+1;        {изменение позиции в строке}
        Slovo:=Copy(st,pn,p-pn)     {Выделение слова}
  End;{Slovo}

Var L,p1,p2:Byte;
  {L - длина строки,
  p1,p2 - начальные позиции соседних слов в строке}
  sl1,sl2:String;{Соседние слова в строке}
  fp:Boolean;    {Флаг перестановки слов}
Begin
  L:=Length(st); {Определение длины строки}
  Repeat
    fp:=FALSE; {Нет перестановки}
    p1:=1;  sl1:=Slovo(p1,st); {Первое слово}
    While p1+Length(sl1)<L {Пока sl1 не последнее
                            слово строки}
    Do Begin
         p2:=p1+Length(sl1)+1;{Позиция 2-го слова}
```

```
          sl2:=Slovo(p2,st);   {Второе слово}
          If sl2<sl1
          Then Begin           {Обмен соседних слов}
                 Delete(st,p1,p2+Length(sl2)-p1);
                 Insert(sl2+' '+sl1,st,p1);
                 p1:=p1+Length(sl2)+1;
                 fp:=TRUE  {Есть перестановка}
               End
          Else Begin {Переход к очередной паре слов}
                 sl1:=sl2;  {Новое первое слово}
                 p1:=p2;    {Новая позиция слова}
               End;
       End;
  Until Not fp {До отсутствия перестановок соседних слов}
End; {LexSort}

Var st:String;
Begin
      Write('Введите строку из слов: ');
      ReadLn(st);
      LexSort(st);
      WriteLn('Упорядоченная строка: ',st);
      WriteLn; ReadLn;
End.
```

Тема контрольной работы №2: Массивы записей.

Порядок выполнения работы

1. В соответствии с индивидуальным заданием на обработку массива записей выполнить постановку задачи: сформулировать условие, определить входные и выходные данные, их ограничения.
2. Разработать математическую модель: описать с помощью формул и рисунков структуру массива записей и процесс его преобразования.
3. Упорядочить записи массива по одному из полей. Вспомогательные массивы не использовать!
4. Сформулировать условие поиска данных в массиве и организовать поиск по условию с сохранением найденных записей в новом массиве.
5. Описать и использовать подпрограммы: создания, просмотра, сортировки массива, поиска данных в массиве.

6. Предусмотреть в программе возможность выбора варианта выполнения программы с помощью меню в окне диалога с пользователем.

7. Программа должна предусматривать ввод входных данных с клавиатуры по запросу. Выходные данные должны выводиться на экран с пояснениями**.**

8. Отладить программу, проверить ее работу на полном наборе тестов.

9. Оформить отчет о контрольной работе в составе: *постановка задачи, математическая модель, структура программы, текст программы, контрольные примеры.*

Задание на контрольную работу №2

Индивидуальное задание на контрольную работу выбирается из списка заданий на лабораторную работу №7 в соответствии с двумя последними цифрами номера зачетной книжки студента по формуле

V = D mod 21 + 1

где V – номер варианта контрольной работы;

D – две последние цифры зачетной книжки студента;

mod – операция получения остатка от деления.

Например, если номер зачетной книжки 03469,

то V = 69 mod 21 + 1 = 6 + 1 =7, и значит, что оно имеет вид:

Вариант индивидуального задания №7

7 Кассы авиабилетов

Номер кассы	ФИО кассира	Количество проданных билетов	Суммарная выручка	Дата продажи

Пример программы обработки массива записей

```
Program MasZap;
{Создание и сортировка картотеки стран}
Const KMAX=10; {Максимальное количество стран}
Type  Tzap=Record
              name:String[15];   {Название}
              area:Real;         {Площадь}
              popul:Real;        {Население}
            End;
     Tmas=Array[1..KMAX] of Tzap;
{$R+}
```

```
Procedure SozdMZ(Var kz:Byte; Var MZ:Tmas);
{Процедура создания массива записей.
 Выходные данные: kz - количество записей,
                  MZ - массив записей.}
Var  i:Byte;
Begin  Write('Задайте число стран не более ', KMAX,': ');
       ReadLn(kz);
       For i:=1 To kz Do
       Begin Write('Название страны ? ');
             ReadLn(MZ[i].name);
             Write('Площадь и население ? ');
             ReadLn(MZ[i].area, MZ[i].popul);
       End;
   End; { SozdMZ }

Procedure SortMZ(kz:Byte;Var MZ:Tmas);
{Процедура сортировки массива записей методом обмена.
 Входные данные:  kz - количество записей,
                  MZ - массив записей.
 Выходные данные: MZ – отсортированный массив записей.}
Var i:Byte; z:Tzap; flag:Boolean;
Begin
  If kz=0
  Then Begin WriteLn('Массив пуст!');
             WriteLn('Нажмите Enter ->');
             ReadLn; Exit; {Выход из процедуры}
       End;
  Repeat
    flag:=FALSE;
    For i:=1 To kz-1
    Do If MZ[i].popul/MZ[i].area<MZ[i+1].popul/MZ[i+1].area
       Then Begin z:=MZ[i];
                  MZ[i]:=MZ[i+1];
                  MZ[i+1]:=z;
                  flag:=TRUE;
            End;
  Until Not flag;
End; {SortMZ}

Procedure OutMZ(comStr:String;kz:Byte;Const MZ:Tmas);
{Процедура вывода массива записей.
 Входные данные: comStr - комментирующая строка,
                 kz - количество записей,
                 MZ - массив записей}
Var i:Byte;
```

```
Begin
  If kz=0
  Then Begin
         WriteLn('Массив пуст!');
         Write('Нажмите Enter ->');
         ReadLn; Exit; { Выход из процедуры }
       End;
  WriteLn(ComStr:40); WriteLn;
  WriteLn('Название страны','Площадь':20,'Население':20);
  WriteLn;
  For i:=1 To kz
  Do With MZ[i]
     Do WriteLn(name:16,area:20:7,popul:20:7);
  WriteLn;
End; {OutMZ}

Var {Описание переменных программы}
      ks:Byte;  {Количество стран}
      MS:Tmas;  {Массив стран}
Begin
     SozdMZ(KS,MS);
     OutMZ('Исходный массив',ks,MS);
     SortMZ(KS,MS);
     OutMZ('Результат сортировки',ks,MS);
     Write('После просмотра нажмите Enter ->');
     ReadLn;
End.
```

6 Методические указания к выполнению практических работ

Практические занятия проводятся с целью приобретения практических навыков алгоритмизации, программирования.

Выполнение каждой практической работы включает разработку алгоритма и написание программы, демонстрацию результатов преподавателю, составление отчета о практической работе. Содержание отчета должно полностью соответствовать заданию на эту практическую работу. Форма титульного листа отчета приведена в приложении.

Практическое занятие №1. Рекурсия.

Варианты индивидуальных заданий

1

Написать рекурсивную подпрограмму вычисления целой степени вещественного ненулевого числа (степень может быть и отрицательной).

2

Написать рекурсивную подпрограмму для определения разбиения целых чисел. Разбиениями целого числа называют способы его представления в виде суммы целых чисел. Например, разбиениями числа 4 являются: 4, 3+1, 2+2, 2+1+1, 1+1+1+1.

3

Написать рекурсивную подпрограмму для вычисления биномиального коэффициента C_n^m по следующей формуле:

$$C(n,m) = \begin{cases} 1, & \text{если } m=0,\ n>0 \text{ или } m=n>=0; \\ 0, & \text{если } m>n>=0; \\ C(n-1,m-1) + C(n-1,m) & \text{в остальных случаях.} \end{cases}$$

4

Написать рекурсивную подпрограмму, которая находит с точностью **ε** корень уравнения ***f(x)=0*** на отрезке ***[a,b]*** методом деления отрезка пополам.

5

Задан массив из вещественных чисел. Описать функцию ***min(x)*** для определения минимального элемента массива ***x***, введя вспомогательную рекурсивную функцию ***min1(k)***, находящую минимум среди последних элементов массива ***x***, начиная с ***k***.

6

Задана строка в виде массива символов. Описать рекурсивную логическую функцию, проверяющую, является ли симметричной часть строки ***s***, начинающаяся ***i***-м и кончающаяся ***j***-м ее элементом.

7

Дана последовательность ненулевых целых чисел, за которой следует ***0***. Напечатать сначала все отрицательные числа этой последовательности, а затем все положительные (в любом порядке).

8

Заданный вещественный массив из ***n*** различных элементов упорядочить по возрастанию следующим методом быстрой сортировки: выбрать какой-нибудь элемент массива (например, средний) и переставить элементы массива так, чтобы слева от выбранного элемента оказались только меньшие элементы, а справа – только большие (тем самым выбранный элемент окажется на своем окончательном месте), после чего рекурсивно применить этот же метод к левой и правой частям массива.

9

“Ханойские башни”. Имеются три колышка ***A,B,C*** и ***n*** дисков разного размера, пронумерованных от *1* до ***n*** в порядке возрастания их размеров. Сначала все диски надеты на колышек ***A*** меньший на больший. Требуется перенести все диски с колышка ***A*** на колышек ***C***, соблюдая при этом следующие условия: диски можно переносить только по одному, больший диск нельзя ставить на меньший. Написать рекурсивную подпрограмму, которая печатает последовательность действий, решающую данную задачу для ***n*** дисков, где ***n*** – заданное натуральное число.

10

Имеется ***n*** населенных пунктов, пронумерованных от *1* до ***n***. Некоторые пары пунктов соединены дорогами. Определить, можно ли попасть по этим дорогам из *1*-го пункта в ***n***-й. Информация о дорогах задается в виде последовательности пар чисел ***i*** и ***j*** (***i***<***j***), указывающих, что ***i***-й и ***j***-й пункты соединены дорогой.

11

Дано ***n*** различных натуральных чисел. Напечатать все перестановки этих чисел.

12

Задача о восьми ферзях: на шахматной доске расставить 8 ферзей так, чтобы они не “били” друг друга. Напечатать все 92 такие расстановки.

13

Даны натуральные числа n и m, найти $НОД(n,m)$. Использовать программу, включающую рекурсивную подпрограмму вычисления $НОД$, основанную на соотношении $НОД(n,m)=НОД(m,r)$, где r – остаток от деления n на m.

14

Числа Фибоначчи $u_0,u_1,u_2,\ldots$ определяются следующим образом:
$u_0=0,\ u_1=1,\ u_n=u_{n-1}+u_{n-2}\quad (n=2,3,\ \ldots)$. Написать программу вычисления u_n для заданного неотрицательного целого n, включающую рекурсивную подпрограмму.

15

Даны натуральные числа a,c,m. Получить $f(m)$, где

$$f(n)=\begin{cases} n, & \text{если } 0<=n<=9; \\ g(n)f(n-1-g(n))+n & \text{в противном случае.} \end{cases}$$

$g(n)$ = остаток от деления $a(n+c)$ на 10.

Использовать программу, включающую рекурсивную подпрограмму вычисления $f(n)$.

16

Даны неотрицательные целые числа n и m. Вычислить функцию $A(n,m)$ вида:

$$A(n,m)=\begin{cases} m+1, & \text{если } n=0; \\ A(n-1,1), & \text{если } n<>0,\ m=0; \\ A(n-1,A(n,m-1)), & \text{если } n>0,\ m>0. \end{cases}$$

Использовать программу, включающую рекурсивную подпрограмму.

17

Треугольником Паскаля называется числовой треугольник в котором по краям стоят 1, а каждое число внутри равно сумме двух стоящих над ним в ближайшей строке сверху. Дано натуральное n. Получить первые n строк треугольника Паскаля. Использовать рекурсивную подпрограмму.

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
```

18

Даны натуральные числа $m,n_1,\ldots,n_m$ ($m>=2$). Вычислить $НОД(n_1,\ldots,n_m)$, воспользовавшись для этого соотношением

$$НОД(n_1,\ldots,n_к)=НОД(НОД(n_1,\ldots,n_{к-1}),n_k)$$
$$(\ к=3,\ldots,n\ )$$

и алгоритмом Евклида.

19

Пусть $t_0=1,\ t_k=t_0t_{k-1}+t_1t_{k-2}+\ldots+t_{k-2}t_1+t_{k-1}t_0,\ \ k=1,2,\ldots$. Получить t_n.

20

Вдоль доски расположено ***2n+1*** лунок, в которых лежат ***n*** черных и ***n*** белых шаров (***n*** черных слева, ***n*** белых справа, посередине пустая лунка). Передвинуть черные шары на место белых, а белые на место черных. Шар можно передвинуть либо в соседнюю с ним пустую лунку, либо в пустую лунку, находящуюся непосредственно за ближайшим шаром. Написать рекурсивную подпрограмму, которая печатает последовательность действий, решающую данную задачу.

21

Вдоль доски расположено ***2n*** лунок, в которых расставлено ***n*** черных и ***n-1*** белых шаров (***1,...,n*** лунки с черными шарами, ***n+1*** лунка пустая, ***n+2,...,2n*** лунки с белыми шарами). Поменять местами черные и белые шары. Шар можно передвинуть либо в соседнюю с ним пустую лунку, либо в пустую лунку, находящуюся непосредственно за ближайшим шаром. Черные шары можно передвигать только вправо, а белые только влево. Написать рекурсивную подпрограмму, которая печатает последовательность действий, решающую данную задачу.

22

В квадрате размером 4 на 4 клетки расставить 16 букв (четыре ***a***, четыре ***b***, четыре ***c,*** четыре ***d***) так, чтобы в каждом горизонтальном и в каждом вертикальном ряду любая буква встречалась только один раз. Напечатать все возможные решения.

23

В каждой из 9 клеток квадрата размером 3 на 3 клетки поставить одно из чисел 1, 2, 3 так, чтобы сумма чисел, стоящих в каждом вертикальном ряду, в каждом горизонтальном ряду, а также по любой диагонали равнялась 6. Решение напечатать.

24

В квадрате размером 3 на 3 клетки расставить числа 1,2,3,4,5,6,7,8,9 так, чтобы суммы чисел, стоящих в каждом вертикальном ряду, в каждом горизонтальном ряду, а также на любой диагонали были равны.

25

На квадратном поле размером 4 на 4 случайным образом расставлены 15 фишек с номерами от 1 до 15. Имеется одна свободная позиция. Расставить фишки по возрастанию их номеров. Передвигать фишки можно только на соседнюю свободную позицию. Написать рекурсивную подпрограмму, которая печатает последовательность действий, решающую данную задачу.

Пример программы с рекурсией

```
Program FactRecFunc;
{Рекурсивное вычисление факториала}

Function FactR(k:Byte):Longint;
{Рекурсивная функция вычисления факториала F=K!
 Входное данное:  k - аргумент.
 Выходное данное: FactR - факториал.}
Begin
     If k=0 Then FactR:=1
     Else FactR:=k*FactR(k-1);
End; { FactR }

Const P=12; {Предельное значение аргумента факториала}
Var   n:0..P; f:Longint;
Begin {$R+}
      Write('Введите 0<=n<=',P,': ');
      ReadLn(n);
      f:=FactR(n);
      WriteLn(n,'! = ',f);
End.
```

Практическое занятие №2. Сортировка.

Создать двумерный массив целых чисел, содержащий 2*n строк и 2*n столбцов. В этом массиве отсортировать указанную часть по указанному в варианте задания правилу, остальную часть массива заполнить нулями.

Варианты индивидуальных заданий

Методы сортировки

1
Сортировка по возрастанию методом выбора минимума.

2
Сортировка по возрастанию методом выбора максимума.

3
Сортировка по убыванию методом выбора минимума.
4
Сортировка по убыванию методом выбора максимума.

5
Сортировка по возрастанию и по убыванию методом выбора минимума.

6
Сортировка по возрастанию и по убыванию методом выбора максимума.

7
Сортировка по возрастанию методом обмена без флага перестановки.

8
Сортировка по убыванию методом обмена без флага перестановки.

9
Сортировка по возрастанию и по убыванию методом обмена без флага перестановки.

10
Сортировка по возрастанию методом обмена с флагом перестановки.

11
Сортировка по убыванию методом обмена с флагом перестановки.

12
Сортировка по возрастанию и по убыванию методом обмена с флагом перестановки.

13
Сортировка по возрастанию методом вставки.

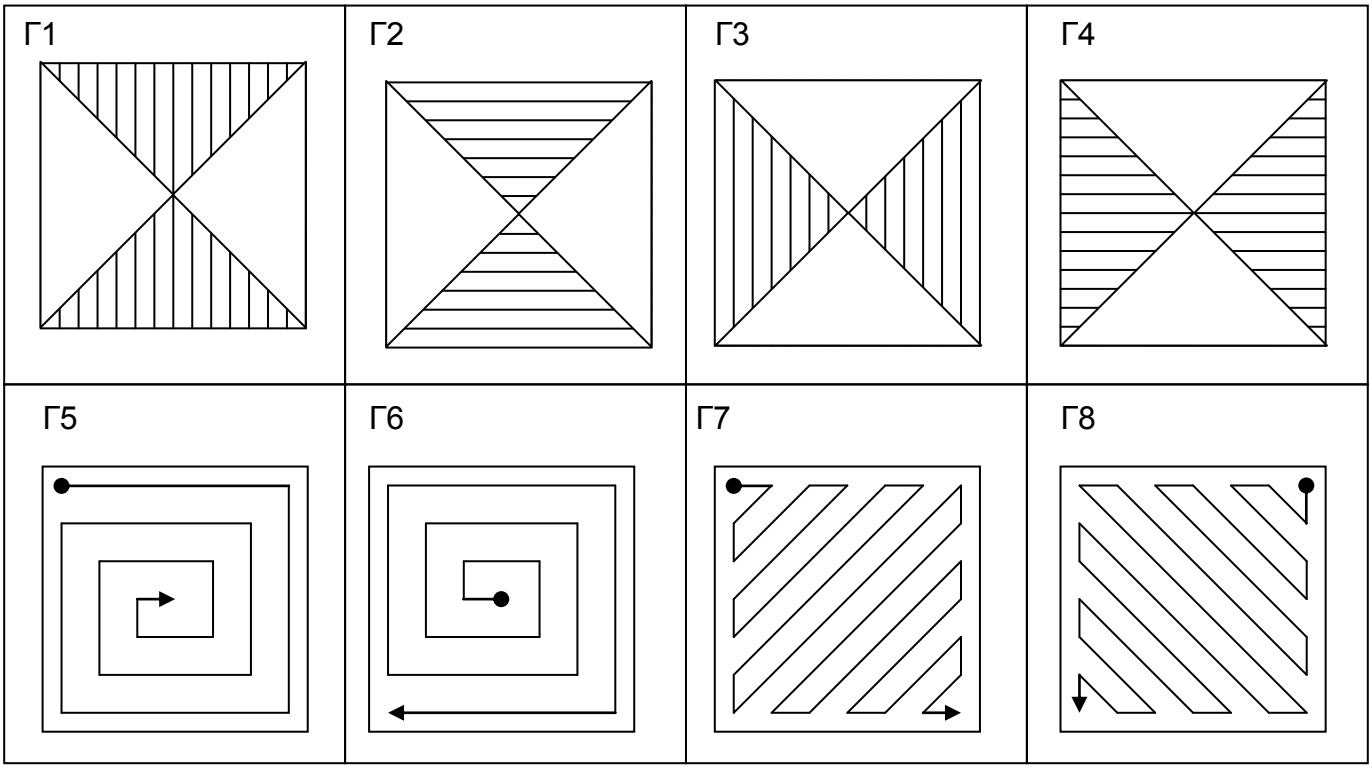
14
Сортировка по убыванию методом вставки.
15
Сортировка по возрастанию и по убыванию методом вставки.

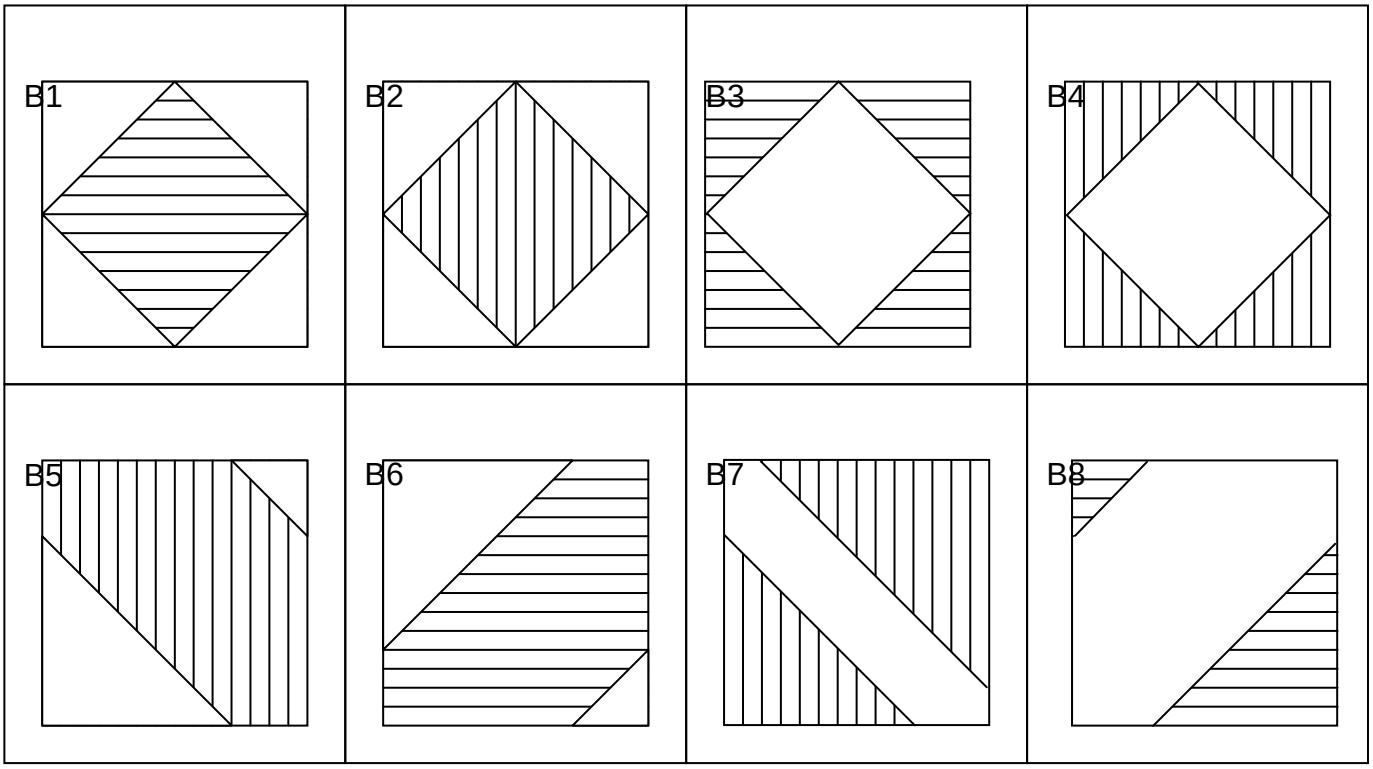
16
Быстрая сортировка по возрастанию.

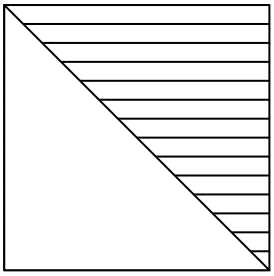
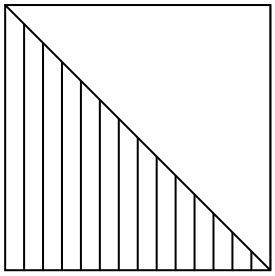
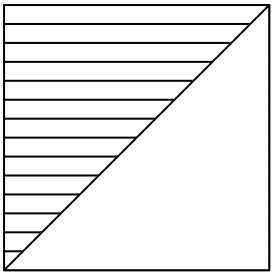
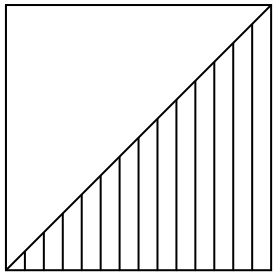
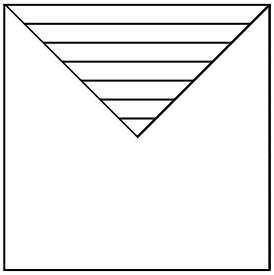
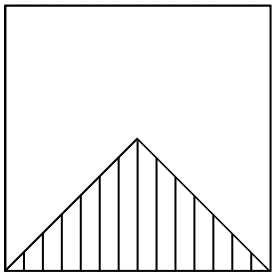
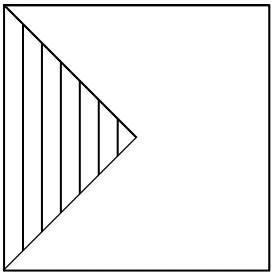
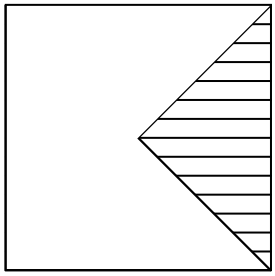
17
Быстрая сортировка по убыванию.

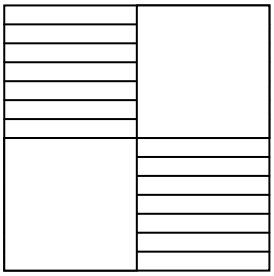
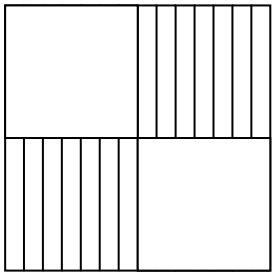
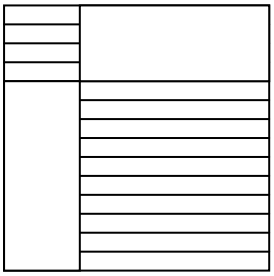
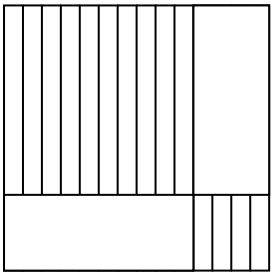
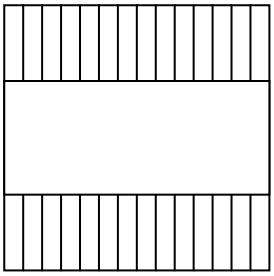
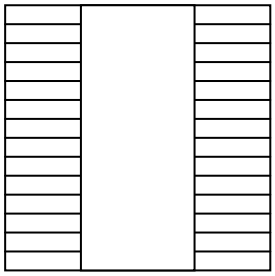
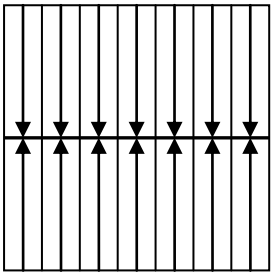
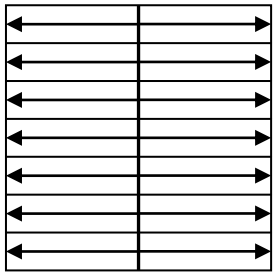
18
Быстрая сортировка по возрастанию и по убыванию.

Сортируемые фрагменты матриц





Б1	Б2	Б3	Б4
			
Б5	Б6	Б7	Б8
			

А1	А2	А3	А4
			
А5	А6	А7	А8
			

Примеры программ сортировки массива

Пример №1

```
Program SortMas;
{Методы сортировки числовых массивов}
Const R=10;    {Длина числового массива}
Type Tind=1..R;
     Tmas=Array[Tind] Of Integer;

Procedure SortVibor(n:Tind; Var M:Tmas);
{Процедура сортировки массива по возрастанию
 методом выбора элементов}
Var i,k,imin:Tind; t:Integer;
Begin For i:=1 To n-1
      Do Begin imin:=i; {Поиск очередного минимума}
               For k:=i+1 To n
               Do If M[k]<M[imin]
                  Then imin:=k;
               {Перестановка элементов}
               t:=M[i]; M[i]:=M[imin]; M[imin]:=t;
         End;
End; {SortVibor}

Procedure SortObmen(n:Tind; Var M:Tmas);
{Процедура сортировки массива по возрастанию
 методом обмена элементов}
Var i,k:Tind; t:Integer;
Begin For k:=n DownTo 2
      Do For i:=1 To k-1   {Цикл сравнений и обменов}
         Do If M[i]>M[i+1] {соседних элементов}
            Then Begin     {Перестановка элементов}
                         t:=M[i]; M[i]:=M[i+1]; M[i+1]:=t;
                 End;
End; {SortObmen}

Procedure SortObmenF(n:Tind; Var M:Tmas);
{Процедура сортировки массива по возрастанию
 методом обмена с флагом}
Var i,k:Tind; flag:Boolean; t:Integer;
Begin
   k:=n;{Начальное количество не отсортированных элементов}
   Repeat flag:=FALSE;  {Нет перестановок}
          For i:=1 To k-1
          Do If M[i]>M[i+1]
```

```
              Then Begin    {Перестановка элементов}
                     t:=M[i]; M[i]:=M[i+1]; M[i+1]:=t;
                     flag:=TRUE; {Есть перестановка}
                   End;
           k:=k-1;
   Until (Not flag) Or (k=1);
End; {SortObmenF}

Procedure SortVstav(n:Tind; Var M:Tmas);
{Процедура сортировки массива по возрастанию
 методом вставки элементов}
Var i,j,k:Tind; t:Integer;

Begin For i:=2 To n
      Do Begin t:=M[i]; {Выделение текущего элемента}
               j:=1;
               {Поиск места вставки в отсортированной части}
               While (j<i) And (M[j]<=M[i])
                  Do j:=j+1;
               For k:=i-1 DownTo j
               Do M[k+1]:=M[k]; {Сдвиг элементов}
               M[j]:=t;   {Вставка текущего элемента}
         End;
End; {SortVstav}

Procedure OutMas(n:Tind; name:String; Var M:Tmas);
{Процедура вывода массива с именем}
Var i:Tind;
Begin Write('Числовой массив ',name,': ');
      For i:=1 To n
      Do Write(M[i],' '); WriteLn;
End; { OutMas }

Const MasA:Tmas=(10,9,8,7,6,5,4,3,2,1);
      MasB:Tmas=(19,17,15,13,11,9,7,5,3,1);
      MasC:Tmas=(20,18,16,14,12,10,8,6,4,2);
      MasD:Tmas=(1,10,2,9,3,8,4,7,5,6);
Begin SortVibor(R,MasA);  OutMas(R,'MasA',MasA);
      SortObmen(R,MasB);  OutMas(R,'MasB',MasB);
      SortObmenF(R,MasC); OutMas(R,'MasC',MasC);
      SortVstav(R,MasD);  OutMas(R,'MasD',MasD);
      ReadLn;
End.
```

Пример №2

```
Program SortStrMatr;
{Сортировка строк матрицы по главному столбцу}
Uses Crt;     {Подключение модуля}
Const R=10;
Type Tind=1..R;
     Tvect=Array[Tind] of Integer;
     Tmatr=Array[Tind] of Tvect;
{$R+}
Procedure InMatr(kstr,kstb:Tind;Var M:Tmatr);
{Процедура ввода матрицы}
Var i,j:Tind;
Begin  Writeln('Вводите матрицу по строкам:');
       For i:=1 To kstr
       Do Begin For j:=1 To kstb
                Do Read(M[i,j]);
                ReadLn;
          End;
End;{InMatr}
Function NStbMin(kstr,kstb:Tind; Const M:Tmatr):Tind;
{Функция определения номера столбца с минимальным элементом}
Var i,j:Tind; min:Integer;
Begin min:=M[1,1]; NStbMin:=1;
      For i:=1 To kstr
      Do For j:=1 To kstb
         Do If M[i,j]<min
            Then Begin min:=M[i,j]; NStbMin:=j;
                 End;
End;{NStbMin}

Procedure SortMatr(kstr,nstb:Tind; Var M:Tmatr);
{Процедура сортировки строк матрицы методом выбора}
Var i,k,imax:Tind; StrM:Tvect;
Begin For i:=1 To kstr-1
      Do Begin imax:=i;
                For k:=i+1 To kstr
                Do  If M[k,nstb]>M[imax,nstb]
                    Then imax:=k;
                StrM:=M[i]; M[i]:=M[imax]; M[imax]:=StrM;
          End;
End;{SortMatr}
```

```
Procedure Okno(x1,y1,x2,y2,cf,ct:Byte);
{Процедура формирования окна на экране}
Begin Window(x1,y1,x2,y2); {Установка параметров окна}
      TextBackGround(cf);  {Установка цвета фона}
      TextColor(ct);       {Установка цвета текста}
      ClrScr;              {Очистка окна}
End;{Okno}

Procedure OutMatr(kstr,kstb:Tind; Const M:Tmatr);
{Процедура вывода матрицы}
Var i,j:Tind;
Begin For i:=1 To  kstr
      Do Begin For j:=1 To kstb
               Do Write(M[i,j]:4);
               WriteLn;
         End;
End;{OutMatr}

Var N,M,NStb:Tind; Matr:Tmatr;
Begin Okno(1,1,80,25,0,15);   {На черном фоне белый текст}
      Write('Размеры матрицы? '); ReadLn(N,M);
      Okno(1,6,38,20,2,15);   {На зеленом фоне белый текст}
      InMatr(N,M,Matr);       {Ввод матрицы}
      NStb:=NStbMin(N,M,Matr);{Поиск главного столбца}
      SortMatr(N,NStb,Matr);  {Сортировка строк}
      Okno(40,6,80,20,3,15);  {На голубом фоне белый текст}
      WriteLn('Отсортированная матрица');
      OutMatr(N,M,Matr);       ReadLn;
End.
```

7 Методические указания к выполнению курсовой работы

Целью курсовой работы является закрепление теоретических и практических знаний, полученных студентами во время лекционных, лабораторных и самостоятельных занятий.

За время курсового проектирования студенты должны выполнить все этапы решения задачи с помощью ЭВМ от постановки задачи до выпуска документации на разработанное программное средство. Курсовой проект заканчивается оформлением пояснительной записки и устной защитой проекта с показом работы программного средства на ЭВМ.

Все задания предполагают создание и обработку дисковых файлов. Поэтому программа, с помощью которой решается поставленная задача, должна содержать, независимо от варианта задания, подпрограммы:

- проверки существования файла;
- создания файлов с исходной информацией;
- просмотра исходных файлов (вывода их на экран);
- просмотра файлов с результатами.

В состав пояснительной записки входят:

- титульный лист (его форма приведена в приложении);
- содержание пояснительной записки (форма приведена в приложении);
- введение;
- четыре программных документа, оформленных по Стандартам ЕСПД (требования к оформлению документов приведены в [1], а содержание – в приложении):
  - Спецификация
  - Текст программы
  - Описание программы
  - Описание применения
- заключение;
- список использованных источников.

Варианты заданий

Индивидуальное задание студента выбирается из предлагаемого списка в соответствии с последней цифрой номера зачетной книжки студента, при этом 0 соответствует варианту №10

1
Заполнение накладной

Заготовка накладной (исходный файл прямого доступа) содержит наименование товара и стоимость единицы товара. При заполнении накладной указывается количество отпущенного товара и рассчитывается его стоимость. Заполненная накладная должна быть записана в новый файл. После заполнения накладной рассчитывается сумма, на которую отпущены товары.

2
Расчет отчислений в пенсионный фонд

Создается исходный файл с фамилиями работников предприятия и начисленной заработной платой для каждого из них.

На его основе сформировать новый файл с указанием для каждого работника

- фамилии,
- величину отчислений в пенсионный фонд (1% от начисленной суммы),
- остатка после вычета в пенсионный фонд.

После заполнения результирующего файла вычислить сумму отчислений в пенсионный фонд.

3
Расчет подоходного налога на начисленную зарплату

Создается исходный файл с фамилиями работников предприятия и начисленной заработной платой для каждого из них.

На его основе сформировать файл с указанием для каждого работника

- фамилии,
- величины подоходного налога, рассчитываемого по формуле:

(начисленная сумма - минимальная зарплата)*13%

Если начисленная сумма меньше минимальной зарплаты, то налог не взимается,

- остатка после вычета налога.

После заполнения результирующего файла вычислить сумму подоходного налога.

4

Составление ведомости переоценки основных средств

Создается исходный файл с наименованием основных средств и их стоимости. Задается процент переоценки основных средств.

На его основе сформировать новый файл с указанием для каждого основного средства его названия и новой стоимости. После заполнения результирующего файла вычислить новую общую стоимость всех основных средств.

5

Составление ведомости индексации вкладов

Создается исходный файл с указанием фамилий вкладчиков и размеров их вкладов. Задается процент индексации.

На его основе сформировать новый файл с указанием для каждого вкладчика фамилии и нового размера вклада. После заполнения результирующего файла вычислить сумму, на которую увеличились вклады.

6

Сортировка базы данных котировки акций по возрастанию их стоимости

Создается исходный файл с указанием наименования акций и их котировок.

На его основе сформировать новый файл, отсортированный по возрастанию стоимости акций.

7

Сортировка базы данных котировки акций по убыванию их стоимости

Создается исходный файл с указанием наименования акций и их котировок.

На его основе сформировать новый файл, отсортированный по убыванию стоимости акций.

8

Выборка из базы данных котировки акций

Создается исходный файл с указанием наименования акций и их котировками.

На его основе сформировать новый файл, содержащий информацию об акциях, стоимость которых находится в задаваемых пределах.

9

Перерасчет пенсий с учетом индексации

Создается исходный файл с указанием фамилий пенсионеров и размеров их пенсий. Задается процент индексации пенсий.

На его основе сформировать новый файл с указанием для каждого пенсионера фамилии и нового размера пенсии. После заполнения результирующего файла вычислить сумму, на которую увеличились пенсии.

10

Выборка задолжников по уплате налогов

Вводится исходный файл с указанием для каждого налогоплательщика фамилии, величины причитающегося налога и величины налога, уплаченного налогоплательщиком.

На его основе сформировать новый файл с указанием для каждого должника по налогу фамилии и долга (долг рассчитывается как разность между величиной налога и уже уплаченной налогоплательщиком суммы налога). После заполнения результирующего файла вычислить общую сумму недоимок.

8 Экзаменационные вопросы

1.Языки программирования. Основные понятия. Способы описания синтаксиса. Примеры описания синтаксиса элементов языка.

2.Язык программирования Турбо Паскаль. История создания. Основные характеристики. Концепция данных. Интегрированная среда программирования

3.Простые типы данных Турбо Паскаля. Внутреннее представление, множество значений, множество операций.

4.Структура языка Турбо Паскаль. Основные символы и слова Турбо Паскаля. Виды слов, их назначение и синтаксис. Примеры.

5.Выражения Турбо Паскаля. Виды выражений. Состав и правила вычисления. Примеры выражений.

6.Описательные предложения Турбо Паскаля. Виды описаний, их назначение и синтаксис. Примеры описаний.

7.Оператор присваивания. Синтаксис и семантика. Совместимость и преобразование типов. Примеры.

8.Оператор обращения к процедуре. Синтаксис и семантика. Примеры использования.

9.Операторы Турбо Паскаля для программирования разветвлений. Синтаксис и семантика. Примеры использования.

10.Операторы Турбо Паскаля для программирования циклических вычислений. Синтаксис и семантика. Примеры использования.

11.Тип массив. Описание и внутреннее представление. Операции над массивами. Доступ к элементам массивов. Способы создания массивов. Примеры создания одномерных и многомерных массивов с индексами разных типов.

12.Процедурное программирование на Турбо Паскале. Основные понятия и возможности. Структурная диаграмма и спецификация программы. Спецификации подпрограмм. Пример разработки программы с подпрограммами поиска и перестановки экстремальных элементов в числовом массиве.

13. Процедуры и функции. Спецификации. Описание и обращение. Сходства и различия. Примеры использования.

14.Процедуры и функции. Способы передачи данных. Виды параметров подпрограмм, схемы передачи и применение. Примеры.

15.Механизм передачи параметров подпрограмм. Пример передачи параметров-значений, параметров- переменных, параметров-констант.

16.Пример использования подпрограмм с параметрами: программа сортировки прямоугольной числовой матрицы.

17.Пример использования подпрограмм с параметрами: программа сортировки треугольной числовой матрицы.

18.Классификация методов сортировки. Процедуры сортировки числовых массивов: метод выбора, метод обмена и метод вставки. Сравнение методов.

19.Процедурные типы. Описание, внутреннее представление, множество значений, множество операций. Пример использования подпрограмм с параметрами процедурного типа: программа решения нелинейных уравнений.

20.Области действия описаний имен. Пример многоблочной программы и ее трассировка. Побочные эффекты и борьба с ними.

21.Рекурсивные процедуры и функции. Механизм рекурсивных вызовов. Виды рекурсивных подпрограмм и их примеры.

22.Тип строка. Два способа реализации строк. Описание и внутреннее представление. Операции над строками. Примеры.

23.Программа редактирования строки: удаление начальных и конечных пробелов, удаление и вставка пробелов между словами, замена повторяющихся фрагментов и другие процедуры.

24.Программа лексикографической сортировки строки текста.

25.Тип множество. Описание и внутреннее представление. Операции над множествами. Способы создания и вывода множеств. Примеры.

26.Пример использования множества чисел: программа вывода последовательности простых чисел.

27.Пример использования множества символов: программа анализа и редактирования текста. Определение количества элементов, младшего и старшего элементов множества.

28.Тип запись. Запись с постоянной частью. Описание и внутреннее представление. Операции над записями. Оператор присоединения. Синтаксис и семантика. Примеры использования.

29.Тип запись. Запись с вариантной частью. Описание и внутреннее представление. Операции над записями. Примеры.

30.Пример использования записей с вариантной частью: программа сортировки массива характеристик геометрических фигур.

31.Тип файл. Описание, структурная организация. Виды файлов. Способы доступа к элементам. Основные правила работы с файлами. Стандартные процедуры и функции для работы с файлами любого вида.

32.Типизированные файлы. Структурная организация. Доступ к элементам. Стандартные процедуры и функции для работы с типизированными файлами.

33.Программа поиска и перестановки экстремальных элементов в файле из целых чисел.

34.Программа сортировки и усечения типизированного файла списка абитуриентов с оценками вступительных экзаменов. Проверка существования файла.

35.Текстовые файлы. Структурная организация. Доступ к элементам. Стандартные процедуры и функции. Стандартные текстовые файлы. Форматы вывода. Примеры использования форматов.

36.Программа создания и редактирования текстового файла. Проверка существования файла.

37.Статическая и динамическая память. Особенности использования. Размещение программ и данных в оперативной памяти ПЭВМ.

38.Ссылочные типы. Описание, внутреннее представление, множество значений, множество операций. Правила использования стандартных процедур для работы с динамической памятью.

39.Работа с большими матрицами в оперативной памяти ПЭВМ. Пример программы.

40.Организация и использование динамических массивов. Пример программы.

41.Списки. Основные понятия. Виды списков. Описания типов. Принципы работы со списками на Турбо Паскале.

42.Процедуры создания односвязных линейных списков. Прямое и обратное включение элементов.

43.Процедуры включения элемента в односвязный линейный список: в начало, в конец, по ссылке (после указанного, перед указанным).

44.Процедуры исключения элемента из односвязного линейного списка: из начала, из конца, по ссылке.

45.Функции поиска элемента в неупорядоченном и в упорядоченном односвязном линейном списке.

46.Односвязные линейные списки. Пример программы.

47.Односвязные циклические списки. Пример программы.

48.Двусвязные линейные списки. Пример программы.

49.Двусвязные циклические списки. Пример программы.

50.Динамические структуры данных. Виды и реализация на Турбо Паскале.

51.Стек и работа с ним. Процедуры инициализации, включения и исключения элемента. Проверка пустоты.

52.Очередь и работа с ней. Процедуры инициализации, включения и исключения элемента. Проверка пустоты.

53.Дек и работа с ним. Процедуры инициализации, включения и исключения элемента. Проверка пустоты.

54.Модульное программирование на Турбо Паскале. Основные понятия и возможности. Структура и компиляция модулей. Установка связи с модулями.

55.Модуль для работы со стеком. Пример использования его для проверки правильности расстановки скобок в тексте программы.

56.Стандатные модули библиотеки Турбо Паскаля: основные характеристики. Модуль System: назначение и состав.

57.Стандартный модуль Crt. Назначение и состав. Пример использования.

Примечание. Каждый билет содержит два вопроса и задачу.

9 Учебно-методические материалы по дисциплине

9.1 Литература

а) основная литература

1. Минакова Н.И. и др. Методы программирования. Учебное пособие. М.: Вузовская книга. 1999 г. - 280с.
2. Марченко А.И., Марченко Л.М. Программирование в среде Turbo Pascal 7.0. Киев: ВЕК, М.: ДЕСС, 2000 г. – 496с.
3. Фаронов В.В. Турбо Паскаль 7.0. Начальный курс. Учебное пособие. М.: Нолидж, 2000 г. - 616с.

б) дополнительная литература

1. Бондарев В.М., Рублинецкий В.И., Качко Е.Г. Основы программирования. Харьков: Фолио, 1997 г. – 368с.
2. Культин Н.Б. Turbo Pascal в задачах и примерах. СПб.: Изд. БХВ, 2000 г. - 256с.
3. Программирование на языке Паскаль. Задачник / Под ред. Усковой О.Ф. СПб.: Питер, 2002 г. – 336с.

9.2 Действующие стандарты

1. ГОСТ 19.701-90 ЕСПД Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.
2. ГОСТ 19.101-77 ЕСПД Виды программ и программных документов.
3. ГОСТ 19.106-78 ЕСПД Требования к программным документам, выполненным печатным образом.
4. ГОСТ 19.202-78 ЕСПД Спецификация. Требования к содержанию и оформлению.
5. ГОСТ 19.401-78 ЕСПД Текст программы. Требования к содержанию и оформлению.
6. ГОСТ 19.402-78 ЕСПД Описание программы. Требования к содержанию и оформлению.
7. ГОСТ 19.502-78 ЕСПД Описание применения. Требования к содержанию и оформлению.
8. ГОСТ 7.32-2001 Система стандартов по информации, библиотечному и издательскому делу. Отчет по научно-исследовательской работе. Структура и правила оформления

ПРИЛОЖЕНИЕ. Формы титульных листов

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Государственное образовательное учреждение
высшего профессионального образования
«Санкт-Петербургский государственный
университет аэрокосмического приборостроения»

Кафедра 44

Рейтинг за работу

Преподаватель ФИО

ОТЧЕТ

по лабораторной работе по дисциплине ПЯВУ

Циклы

ПЯВУ 44.2201.03 ЛР

Работу выполнил
студент гр. ФИО

Санкт-Петербург
2003 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Государственное образовательное учреждение
высшего профессионального образования
«Санкт-Петербургский государственный
университет аэрокосмического приборостроения»

Кафедра 44

Курсовой проект
защищен с оценкой

Руководитель ФИО

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по дисциплине
“Программирование на языках высокого уровня”

Заполнение накладной

ПЯВУ 44.2201.01 КП

Работу выполнил
студент гр. ФИО

Санкт-Петербург
2003 г.

Содержание

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Государственное образовательное учреждение
высшего профессионального образования
«Санкт-Петербургский государственный
университет аэрокосмического приборостроения»

Кафедра 44

Рейтинг за работу

Преподаватель ФИО

Контрольная работа

по дисциплине “Программирование на языках высокого уровня”

Строки

ПЯВУ 44.2201.01 кр

Работу выполнил
студент гр. ФИО

Санкт-Петербург
2003 г.