РОСЖЕЛДОР
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Ростовский государственный университет путей сообщения»
(ФГБОУ ВО РГУПС)
Тихорецкий техникум железнодорожного транспорта – филиал РГУПС
(ТТЖТ – филиал РГУПС)

А.В. Украинский

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

по выполнению практических занятий по дисциплине

Основы алгоритмизации и программирования

для студентов специальности 09.02.01 Компьютерные системы и комплексы

III курс

Тихорецк, 2022

УТВЕРЖДАЮ
Заместитель директора по УР
_____ Н.Ю. Шитикова
«01» сентября 2022 г.

Методические рекомендации по проведению практических занятий по дисциплине **Основы алгоритмизации и программирования** разработан в соответствии с федеральным государственным образовательным стандартом по специальности среднего профессионального образования 09.02.01 Компьютерные системы и комплексы.

Организация-разработчик: Тихорецкий техникум железнодорожного транспорта – филиал Федерального государственного бюджетного образовательного учреждения высшего образования «Ростовский государственный университет путей сообщения» (ТТЖТ – филиал РГУПС)

Разработчик:
Украинский А.В., преподаватель ТТЖТ – филиала РГУПС

Рекомендованы цикловой комиссией № 7 специальностей 09.02.01, 11.02.06, 38.02.01
Протокол заседания № 1 от «01» сентября 2022 г.

Содержание

Практическое занятие №1

Тема: *Решение задач линейной структуры*.

Цель работы: Научиться решать задачи линейной структуры используя линейный алгоритм в виде блок-схемы.

Перечень используемого оборудования:

Монитор, системный блок, устройства ввода-вывода.

Операционная система семейства Windows, интегрированная среда программирования PascalABC.Net.

Ход работы

1. Изучить правила записи линейных алгоритмов в виде блок-схем, используя базовую алгоритмическую конструкцию (БАК) «последовательность».
2. Записать решение задач в виде блок-схем используя БАК «последовательность».
3. Ответить на контрольные вопросы.
4. Сделать выводы.

Краткие сведения из теории

Основные типы алгоритмических конструкций.

Алгоритмы бывают **трех основных видов**, которые являются базовыми при написаний программ.

Первый тип – линейный алгоритм. В нем все действия выполняются в строгом порядке, последовательно, одно за другим. Типичный жизненный пример такого алгоритма – рецепт пирога.

Второй тип – разветвляющийся алгоритм. Здесь те или иные действия выполняются в зависимости от выполнения или невыполнения некоего условия. Пример из жизни – правило перехода улицы по светофору. Если горит красный – стоим, если горит зеленый – идем.

Наконец, третий тип – циклический алгоритм. Он содержит повторяющиеся действия с какой-либо изменяющейся величиной, так называемым параметром. По циклическому алгоритму можно колоть дрова. Берем полено, ставим на попа, колем топором, берем второе полено и т.д., пока поленья не закончатся, и эта работа нам не надоест.

Линейный алгоритм.

Существует большое количество алгоритмов, в которых команды должны быть выполнены последовательно одна за другой. Такие последовательности команд будем называть сериями, а алгоритмы,

состоящие из таких серий, линейными.

Для того чтобы сделать алгоритм более наглядным, чаще всего используют блок-схемы.

Различные элементы алгоритма изображаются с помощью различных геометрических фигур: для обозначения начала и конца алгоритма используются прямоугольники с закругленными углами, а для обозначения последовательности команд – прямоугольники (рис. 1).

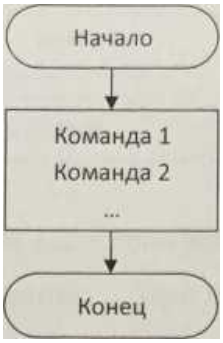


Рисунок 1 – Линейный алгоритм

На блок-схеме хорошо видна структура линейного алгоритма, по которой исполнителю (человеку) удобно отслеживать процесс его выполнения.

Задания:

1. Дана длина ребра куба. Найти площадь грани, полной поверхности и объём этого куба.

2. Допустим, вы получили наследство $ 1 000 000 и хотите красиво пожить. После долгих раздумий вы решаете, что будете жить на $ 800 в месяц. На сколько лет вам хватит наследства?

3. Пушка стреляет под углом 30° к линии горизонта. Масса снаряда 30 кг, начальная скорость 500 м/с. Какова будет дальность полета снаряда? (Формулу вспомните из курса физики.)

4. Документ содержит текст из 32 строк по 60 символов в каждой и точечную черно-белую фотографию 10х15 см. Каждый квадратный сантиметр содержит 300 точек, любая точка описывается 4-мя битами. Каков общий информационный объем документа в Кбайтах?

Пример выполнения задания

Задача_0:

Вычислить длину окружности, если известен её радиус.

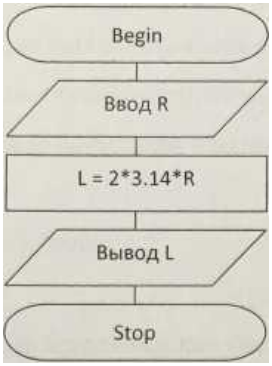


Рисунок 2 – Решение задачи

Результат выполнения программы: При R – 5, L – 31.4.

Содержание отчета

1. Наименование и цель работы;
2. Теоретические сведения;
3. Выполнение заданий;
4. Ответы на контрольные вопросы;
5. Вывод о проделанной работе.

Контрольные вопросы:

1. Перечислите основные алгоритмические конструкции.
2. Дайте определение линейному алгоритму.
3. Дайте определение разветвляющемуся алгоритму.
4. Дайте определение циклическому алгоритму.
5. Для чего используются блок-схемы?
6. Какой геометрической фигурой обозначается начало и конец алгоритма?
7. Каким элементом алгоритма служит прямоугольник?
8. Запишите *Пример выполнения задания* в тетрадь и наберите программу с переменными из *примера*, убедитесь в корректности полученных результатов длина окружности должна получиться 31.4 при радиусе 5.
9. Выполните **Задания** в виде блок-схем в тетради для практических занятий, по аналогии с приведенным *примером*.
10. Наберите все **Задания** на языке программирования Паскаль нового поколения – PascalABC.NET, продемонстрируйте полученные результаты выполнения первой практической работы в тетради и на компьютере преподавателю.

Практическое занятие №2

Тема: *Решение задач разветвляющейся структуры*

Цель: Научиться решать задачи разветвляющейся структуры используя конструкции «ветвление» и «выбор» в виде блок-схемы.

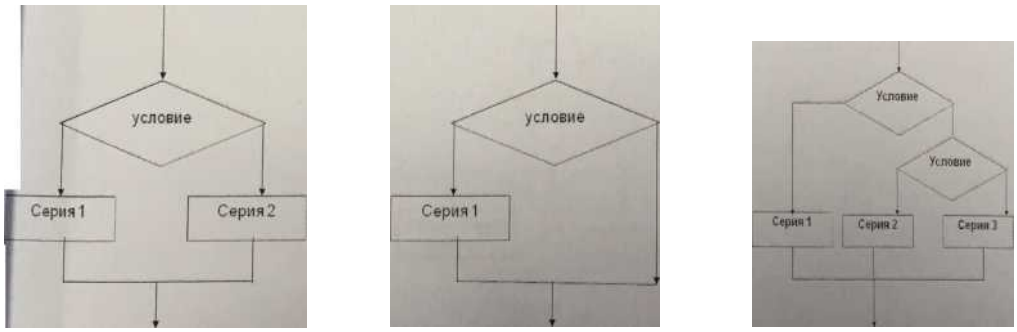
Перечень используемого оборудования: Монитор, системный блок, устройства ввода-вывода. Операционная система семейства Windows, интегрированная среда программирования PascalABC.Net.

Ход работы

1. Изучить правила записи разветвляющихся алгоритмов в виде блок-схем, используя базовую алгоритмическую конструкцию (БАК) «ветвление».
2. Записать решение задач в виде блок-схем используя соответствующие конструкции
3. Ответить на контрольные вопросы.
4. Сделать выводы.

Краткие теоретические сведения

Алгоритмическая конструкция «ветвление». В отличие от линейных алгоритмов, в алгоритмическую структуру «ветвление» входит условие, в зависимости от выполнения или невыполнения которого реализуется та или иная последовательность команд (серия).



а) б) в)

Рисунок 2.1 – Разветвляющиеся алгоритмические конструкции:

а) Полная форма, б) Сокращённая форма, в) Вложенные ветвления (выбор).

Алгоритмическая структура «выбор» применяется для реализации ветвления со многими вариантами серий команд. В структуру выбора входят несколько условий для одного параметра, проверка которых осуществляется строго последовательно. При истинности одного из условий выполняется соответствующая последовательность команд. В алгоритмической структуре «выбор» выполняется одна из нескольких последовательностей команд при истинности соответствующего условия.

Задание

1. Напишите алгоритм, определяющий четность или нечетность введенного с клавиатуры целого числа.
2. Даны действительные числа x y, неравные друг другу. Меньшее из этих двух чисел заменить половиной их суммы, а большее заменить их удвоенным произведением.
3. Вводятся три числа a,в,c. Найти максимальное значение вывести на экран. (a≠в≠c).
4. Составить алгоритм, который по номеру дня недели выводит соответствующее ему название.

Пример выполнения задания

Задача 0: Составить алгоритм вычисления квадратного корня числа.

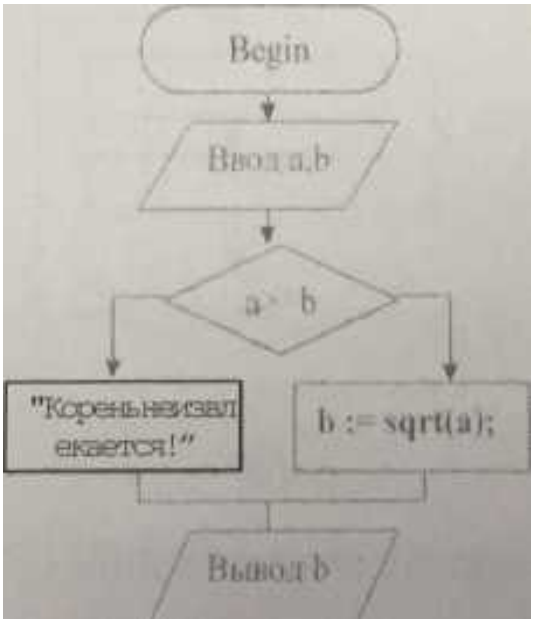


Рисунок 2.2 – Решение задачи

Контрольные вопросы:

1. Дайте определение разветвляющемуся алгоритму.
2. Какие разветвляющиеся алгоритмические конструкции существуют?
3. Что собой представляет конструкция «выбор»

Практическое занятие №3

Тема: *Решение задач циклической структуры*

Цель: Научиться решать задачи циклической структуры, используя конструкции цикла: «с параметром», «с предусловием» и «с постусловием» в виде блок- схемы.

Оборудование и программные средства: инструкционные карты, справочный материал.

Ход работы

1. Изучить правила записи алгоритмов в виде блок-схем, используя циклические алгоритмические конструкции.
2. Записать решение задач в виде блок-схем используя соответствующие конструкции:
3. Ответить на контрольные вопросы.
4. Сделать выводы.

Краткие теоретические сведения

Алгоритмическая конструкция «цикл» содержит последовательность команд, которая выполняется многократно. Такая последовательность команд называется телом цикла.

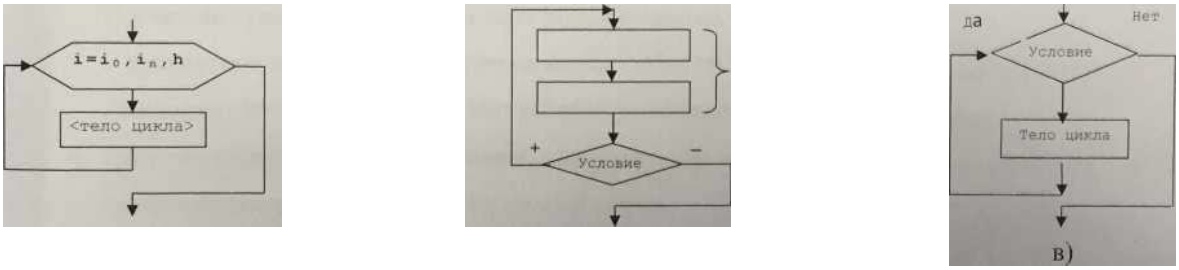


Рисунок 3.1 – Циклические алгоритмические конструкции:
- а) цикл с параметром , б) с предусловием, в) с постусловием.

Циклические алгоритмические структуры бывают двух типов:

а. циклы с параметром (счетчик), в которых тело цикла выполняется определенное количество раз;

б. циклы с условием, в которых тело цикла выполняется, пока условие истинно/ложно:

с предусловием

с постусловием

Цикл с параметром (счетчик).

Когда заранее известно, какое число повторений тела цикла необходимо выполнить, используется циклическая конструкция «цикл с параметром».

Цикл с предусловием

Условие выхода из цикла можно поставить в начале, перед телом цикла. Такой цикл называется циклом с предусловием.

Цикл с постусловием

Условие выхода из цикла можно поставить в конце, после тела цикла. Такой цикл называется циклом с постусловием.

Цикл с постусловием, в отличие от цикла с предусловием, выполняется обязательно как минимум один раз, независимо от того, выполняется условие или нет.

Задание

1. Вводятся числа до тех пор, пока не будет введено число 55. Посчитать их произведение включая последнее.
2. Вводятся числа до тех пор, пока не будет введено число меньше 0. Посчитать сумму всех положительных чисел.
3. С клавиатуры запрашивается любая цифра Z от 2 до 9, а затем компьютер печатает таблицу умножения на эту цифру.
4. Напишите программу, выводящую на экран степени числа x от 2 до 10 включительно в табличной форме.
5. Вычислить У=2x+5 и вывести для каждого значения x gj [3;8], с шагом 0,2.
6. Распечатать в табличном виде (с аргументами) значение функции квадратного корня на интервале [2; 4] с шагом 0,1.
7. Выведите на экран в строку все числа первой сотни, оканчивающиеся на пять.

Пример выполнения задания.

Задача 0: Составить алгоритм вычисления суммы 1+2+3+...+N-1 +N.

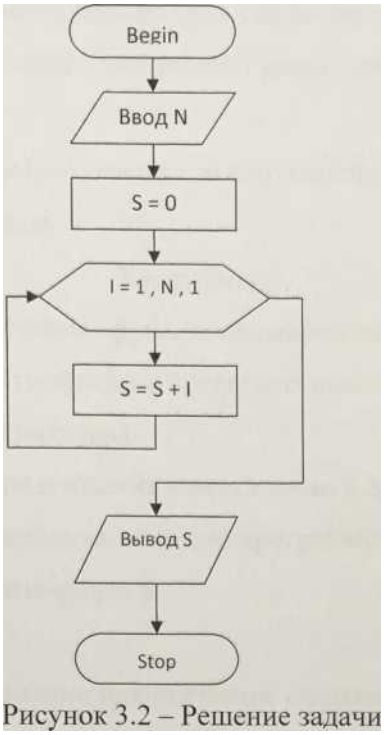


Рисунок 3.2 – Решение задачи

В этом примере в цикле выполняется операция сложения. К полученной до текущего прохода сумме добавляется очередное число (переменная цикла). Тело цикла выполняется N раз. Переменная цикла i меняется от 1 до N с шагом 1. После завершения выполнения цикла (после

того, как тело цикла будет выполнено N раз) выполняется команда вывода результатов, и на этом алгоритм заканчивается.

Контрольные вопросы:

1. Дайте определение циклическому алгоритму.
2. Перечислите 4 составляющих любого цикла.
3. Назовите минимальное число выполнений тела цикла (итераций) в циклах а) с предусловием и б) с постусловием.
4. Что является условием выхода из цикла со счётчиком?

Практическая работа №4,5

Тема: Знакомство с интегрированной средой программирования. Настройка основных параметров среды программирования.

Цел: Научиться работать в интегрированной среде программирования Турбо Паскаль.

Оборудование и программные средства: инструкционные карты, справочный материал, ПК, Турбо Паскаль.

Ход работы

1. Изучить интерфейс, основные функциональные клавиши, порядок работы с командами меню и инструментами среды программирования Турбо Паскаль, используя справочный материал.
2. Отработать навыки использования команд меню в среде программирования ТУРБО ПАСКАЛЬ, выполняя тестовую программу.
3. Ответить на контрольные вопросы.
4. Сделать выводы.

Краткие теоретические сведения

Система программирования ТУРБО ПАСКАЛЬ представляет собой совокупность компилятора языка

Для вызова ТУРБО ПАСКАЛЯ следует запустить файл приложенная TURBO.EXE.

Путь к файлу: D:\Pascal\BP_7.0\BIN\TURBO.EXE

По этой команде в память загружается файл TURBO.EXE, на экране монитора появляется оболочка ТУРБО ПАСКАЛЯ. Верхняя строка оболочки содержит «главное меню» режимов работы ТУРБО ПАСКАЛЯ, нижняя строка – краткую справку о назначении основных функциональных клавиш. Центральная часть экрана принадлежит окну редактора для ввода и редактирования текста программы. Под верхней строкой в центре двойной рамки приводится имя дискового файла (новому файлу присваивается имя NONAMEOO.PAS). Цифра в верхнем правом углу окна редактора обозначает номер окна редактирования. В ТУРБО ПАСКАЛЕ можно одновременно работать с 9 программами, каждая из которых располагается в отдельном окне редактора. Кроме окон редактора используются окна: отладочного режима, вывода результатов работы программы, справочной службы, стека, регистров.

Для перехода в главное меню необходимо нажать клавишу F10, поместить курсор на нужную функцию, нажать ENTER. В появившемся ниспадающем меню выбрать необходимую опцию и нажать ENTER. Например, для сохранения текста программы на диске нужно нажать F10, переместить курсор на FILE, затем SAVE и нажать ENTER, в появившемся диалоговом окне задать имя файла, под которым ваша программа будет храниться на диске. Отметим, что имя файла, под которым текст программы хранится на диске, не эквивалентен имени программы, заданным в первой строке программы, написанной на Паскале.

Функциональные клавиши используются для быстрого управления средой ТУРБО ПАСКАЛЯ.

Назначения функциональных клавиш:

- F1 – вызов справки;
- F2 – записать программу на диск;
- F3 – открыть записанную программу на диске в окне редактора;
- F5 – распахнуть активное окно на весь экран;
- F6 – сделать активным следующее окно;
- F7 – используется в отладочном режиме;
- F8 – используется в отладочном режиме;
- F9 – компилировать программу, но не выполнять ее;
- Ctrl-F9 – компилировать программу, загрузить ее в оперативную память (или записать на диск) и выполнить, после чего вернуться в среду ТУРБО ПАСКАЛЬ;
- Alt-F5 – сменить окно редактора на окно вывода результатов программы;
- Alt-X – выход из ТУРБО ПАСКАЛЯ;

Счет и отладка программы

После подготовки текста программы необходимо откомпилировать программу, при необходимости связать ее с библиотекой стандартных процедур и функций, загрузить ее в оперативную память и передать ей управление. Вся эта последовательность действий реализуется одновременным нажатием клавиш Ctrl-F9.

Если в программе нет ошибок, то все действия выполняются последовательно одно за другим, при этом на экране сообщается о количестве строк откомпилированной программы и объеме доступной оперативной памяти.

Перед передачей управления загруженной программе среда очищает экран и на него выводятся результаты выполнения программы, а после завершения работы программы вновь восстанавливается окно редактора.

Если на каком-нибудь этапе среда обнаружила ошибку, она прекращает дальнейшее действие, восстанавливает окно редактора и помещает курсор на ту строку программы, на которой обнаружена ошибка. При этом в верхней строке редактора появляется диагностическое сообщение о причине ошибки. Необходимо найти причину ошибки и отредактировать текст. В более сложных ситуациях прибегают к пошаговому исполнению программы, например, последовательным нажатием F7. В случае необходимости можно просмотреть значения проверяемых переменных. Для этого поместите курсор в строку, содержащую переменную, и нажмите Ctrl-F4. В появившемся диалоговом окне в верхнем поле будет имя переменной, нажав на ENTER, в среднем поде получим ее значение. В верхнее поле можно с клавиатуры вводить имена переменных или выражение.

Справочная система

В затруднительной ситуации нажмите на клавишу F1 или CTRL-F1 (для объяснения конкретной ситуации) и на экране высветится необходимая справка. Во многих случаях справка содержит пример небольшой программы, которую можно скопировать в окно редактирования, запустить на выполнение и посмотреть результат.

Задание

- создать новый файл,
- ввести текст тестовой программы (без комментариев),
- сохранить файл в корневом каталоге,
- компилировать, при необходимости отладить (исправить ошибки),
- выполнить программу (код и результаты выполнения записать),
- выполнить программу пошагово, при этом просматривая значения переменных на каждом шаге (значения переменных на каждом шаге записать).

Тестовая программа

```
Programpr_l; (Каждая программа имеет своё имя. Имя не может содержать пробелы}
UsesCrt; (Включена библиотека Crt}
Varx, y, s ,p, q, v: real; (Числа вещественные}
Begin{Начало}
    ClrScr; (Очистить экран. Именно для этого нужна библиотека}
    Write('Введите 2 числа'); (Пояснительный текст}
    Readln(x, y); (Ввод двух чисел и переход на следующую строку}
    s x + y; {Сложение}
    p:_ x * y; {Произведение}
    q:— x – y; {Вычитание}
    v:= x / y; {Деление}
    WriteLn('Результат сложения равен', s:8:2);{Форматный вывод}
    WriteLn('Результат произведения равен', p:8:2);{Форматный вывод}
    WriteLn('Результат вычитания равен', q:8:2);{Форматный вывод}
    WriteLn('Результат деления равен', v:8:2);{Форматный вывод}
    RepeatUntilKeyPressed{Дословно – ждать, пока не будет нажата}
End.                    (любая клавиша. И остановиться.}
```

Контрольные вопросы

1. Что такое синтаксис?
2. Что такое семантика?
3. Что представляет собой процесс отладки программы?
4. Основная причина появления ошибок в программе?

Практическая работа №6

Тема: Составление программ линейной структуры Цель: Научиться составлять программы линейной структуры.

Оборудование и программные средства: персональный компьютер, интегрированная среда программирования Pascal., инструкционные карты, справочный материал.

Ход работы

1. Изучить структуру программы на языке Паскаль, операторы присваивания (:=), ввода, вывода и основные математические функции, используя справочный материал.
2. Выполнить задание в виде программ с линейной структурой, используя соответствующие операторы.
3. Ответить на контрольные вопросы.
4. Сделать выводы.

Краткие теоретические сведения

Алгоритмическая структура, в которой команды выполняются последовательно одна задругой, называется следованием. Алгоритм, реализованный с помощью такой структуры, называется линейным.

Структура программы на языке Паскаль

В программе могут быть следующие разделы, каждый из которых, кроме последнего, завершается точкой с запятой:

PROGRAM<заголовок программы>;
LABEL<раздел объявления меток>;
CONST<раздел объявления констант>;
TYPE<раздел объявления типов>;
VAR<раздел объявления переменных>;
PROCEDURE (FUNCTION) <раздел объявления подпрограмм>;
BEGIN
<тело программы>
END.

Программы с линейной структурой составляются из операторов присваивания (:=), ввода, вывода и обращения к подпрограммам.

Операторы ввода

READ(al, a2, ..., an) – каждое вводимое значение получают последовательно переменные al, a2,...,an.

READLN(al, a2,..,an) – каждое вводимое значение получают последовательно переменные al ,a2,...an; осуществляя переход на новую строку при вводе данных.

READLN – переход на новую строку при вводе данных. Такой оператор применяется, когда исполнение программы желательно задержать до нажатия клавиши ENTER.

Операторы вывода

WRlTE(al; a2,..,an) – выводит последовательно значения переменных a1, a2, ..., an без перехода на следующую строку.

WRlTELN(al, a2, .., an) – выводит последовательно значения al, a2, ..., an, осуществляя переход на новую строку.

При использовании этих операторов можно указывать число позиций, в которых нужно вывести значение: WR1TE(J:8,1:12).

Основные математические функции:

ABS(x)	|x|
SIN(X)	SINX
COS(X)	cosx
SQRT(X)	Квадратный корень из X
ArcTAN(X)	ArcTANX
INT(X)	Целая часть X
SQR(X)	Квадрат аргумента
EXP(X)	Показательная функция
LN(N)	Натуральный алгоритм
ROUND(X)	Округляет по правилам арифметики тип longint до целого
RANDOM[(X)]	Выдает случайное число из интервала
FRAC(X)	Дробная часть числа
TRUNC(X)	Округляет число, отбрасывая дробную часть числа тип longint
INC(X,Y)	Увеличивает X на величину Y
1DEC(X,Y)	Уменьшает X на величину Y

Задание

1. Дана длина ребра куба. Написать программу нахождения площади грани, полной поверхности и объём этого куба.
2. Написать программу сложения двух чисел.
3. Составить программу вычисления площади квадрата по его стороне.
4. Написать программу вычисления силы, действующей на тело массой гл, движущегося с ускорением а (fm * а).

Пример выполнения задания

3адача_0вычислить длину окружности, если известен её радиус.

```
Programpr_l; {Каждая программа имеет своё имя. Имя не может содержать пробелы}
UsesCrt; {Включена библиотека Crt}
ConstP:=3.14;
VarL, R: real; (Числа вещественные}
Begin {Начало}
```

```
ClrScr; {Очистить экран}
Write(’Введите радиус’); {Пояснительный текст}
Readln(R); {Ввод и переход на следующую строку}
L: = 2*P*R; {Вычисление длины окружности}
WriteLn(’Результат сложения равен’, L:8:2);{Форматный вывод}
RepeatUntilKeyPressed{ Задержка результата выполнения программы }
    End.{любая клавиша. И остановиться.}
```

Результат выполнения программы; При R-5, L ^ 31,4.

Контрольные вопросы

1. В каком разделе описываются переменные?
2. В чем разница между функциями READ и READLN?
3. Каким знаком препинания обязательно заканчивается каждый оператор программы? |
4. С помощью какой функции можно возвести число в квадрат?

Практическая работа №7

Тема: Составление программ разветвляющейся структуры

Цель: Научиться составлять программы разветвляющейся структуры.

Оборудование и программные средства: персональный компьютер, интегрированная среда программирования Pascal., инструкционные карты, справочный материал.

Ход работы

1. Изучить условные операторы языка Паскаль, используя справочный материал.
2. Выполнить задание в виде программ разветвляющейся структуры, используя соответствующие операторы.
3. Ответить на контрольные вопросы.
4. Сделать выводы.

Краткие теоретические сведения

Алгоритмическая структура, в которой в зависимости от выполнения условия, выполняется либо одна, либо другая последовательность команд, называется ветвлением. Алгоритм, реализованный на языке программирования с помощью такой структуры, называется разветвляющимся.

Формат записи (синтаксис):

if<условие>Then<серия 1>Else<серия 2>;

где <условие> – выражение логического типа; <серия 1> и <серия 2> – любые серии операторов Pascal.

Задание

1. Даны действительные числа а и Ь, неравные друг другу. Меньшее из этих двух чисел заменить половиной их суммы, а большее заменить их удвоенным произведением.
2. Составить программу решения квадратного уравнения.
3. Вводятся три числа а, Ь, с (а^Ь^с). Составить программу поиска максимального значения и вывести максимум на экран.
4. Составить алгоритм, который по номеру дня недели выводит соответствующее ему название.
5. Написать программу, определяющую, пройдёт ли кирпич с заданными размерами (a, b и c) в отверстие со сторонами x и y.

Пример выполнения задания

Задача 0: Составить программу вычисления квадратного корня числа.

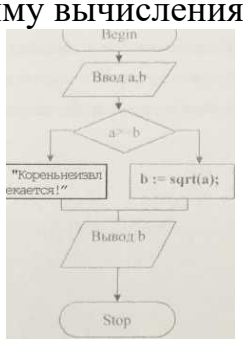


Рисунок 6.1- Алгоритм решения задачи

```
ProgramP_5;
   Uses Crt;
   Var a, b: Integer;
   Begin
   ClrScr;
   WriteLn('Вычисление квадратного корня из числа');
   Write('Введите число');
   Readbn(a);
   If a >= 0 Then Begin
   B:=sqrt(a);
   Writeln('Результат равен', Ь:7:2);
   End;
   Else
   Writeln('Корень из отрицательного не извлекается!');
   ReadKey
End.
```

Результат выполнения программы: При a = 5, b=25. При a =- -22, корень не извлекается.

Контрольные вопросы

1. Какой алгоритм называется разветвляющимся?
2. Каким образом в условном операторе реализуется альтернативная ветвь?
3. В каком случае используется оператор выбора Case ... Of?

Практическая работа №8

Тема: Составление программ циклической структуры

Цель: Научиться составлять программы циклической структуры.

Оборудование и программные средства: персональный компьютер, интегрированная среда программирования Pascal.

Ход работы

1. Изучить операторы циклов языка Паскаль, используя справочный материал.
2. Выполнить задание в виде программ циклической структуры, используя соответствующие операторы.
3. Ответить на контрольные вопросы.
4. Сделать выводы.

Краткие теоретические сведения

Алгоритмическая структура, в которой в многократно выполняется последовательность команд, называется циклом. Такая последовательность команд называется телом цикла. Алгоритм, реализованный на языке программирования с помощью такой структуры, называется циклическим. Циклические алгоритмические структуры бывают двух типов:

в. циклы с параметром (счетчик), в которых тело цикла выполняется определенное количество раз;

г. циклы с условием, в которых тело цикла выполняется, пока условие истинно/ложно:

- с предусловием
- с постусловием

Цикл с параметром (счетчик)

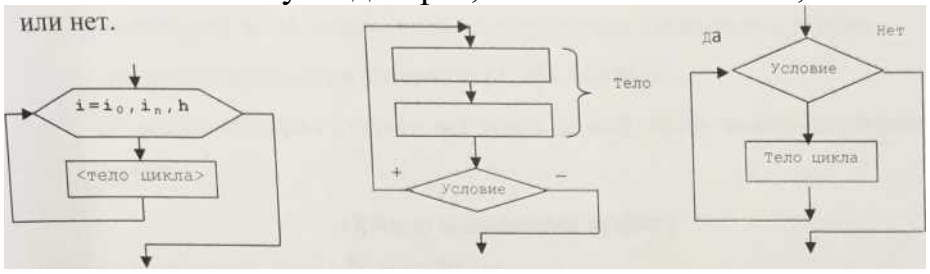
Когда заранее известно, какое число повторений тела цикла необходимо выполнить, используется циклическая конструкция «цикл с параметром».

Цикл с предусловием

Условие выхода из цикла можно поставить в начале, перед телом цикла. Такой цикл называется, циклом с предусловием Цикл с постусловием

Условие выхода из цикла можно поставить в конце, после тела цикла. Такой цикл называется циклом с постусловием.

Цикл с постусловием, в отличие от цикла с предусловием, выполняется обязательно как минимум один раз, независимо от того, выполняется условие или нет.



а) б) в)

Рисунок 7.1 – Циклические алгоритмические конструкции:

а) цикл с параметром , б) с предусловием, в) с постусловием.

Если количество повторений заранее известно, то в Pascal применяется структура For... Next. Оператор For задает цикл, строго ограниченный по количеству проходов.

a) For<параметр>:=<начальное значение>To<конечное значение>To<тело цикла>

Если количество повторений заранее не известно и выход из цикла происходит по достижению некоторого условия, то в Pascal применяется итерационные циклы: с постусловием и с предусловием.

b) Repeat<тело цикла>Until(While) <условие>

c) While<условие>Do<тело цикла>

Задание:

1. Вводятся числа до тех пор пока не будет введено число 55. Посчитать их произведение включая последнее.
2. Вводятся числа до тех пор пока не будет введено число меньше 0. Посчитать сумму всех положительных чисел.
3. С клавиатуры запрашивается любая цифра Z от 2 до 9, а затем компьютер печатает таблицу умножения на эту цифру.
4. Напишите программу, выводящую на экран степени числа x от 2 до 10 включительно в табличной форме.
5. Вычислить У=2x+5 и вывести для каждого значения x е [3;8], с шагом 0,2.
6. Распечатать в табличном виде (с аргументами) значение функций квадратного корня на интервале [2; 4] с шагом 0,1.
7. Выведите на экран в строку все числа первой сотни, оканчивающиеся на пять.

Пример выполнения задания

Задача: Напечатать слово "Слон" 20 раз.

```
ProgrammP_l 1;
UsesCrt;
Vari: Byte; {Цикл организован ради того, чтобы 20 раз выполнить}
Begin {один и тот же оператор: Напечатать слово «Слон»}
ClrScr;
   For I:=l to 20 Do WriteLn(’Слон’);
   ReadKey
End.
```

Контрольные вопросы

1. Какой оператор используется для досрочного выхода из цикла в Pascal?
2. В чем состоит разница в выполнении циклов с постусловием REPEAT...UNTIL и REPEAT...WHILE?
3. В каком случае при решении задачи в Pascal применяется структура For...Next?
4. Какие циклы используются в Pascal, если количество повторений заранее не известно и выход из цикла происходит по достижению некоторого условия?

Практическая работа №9

Тема: Обработка одномерных массивов

Цель: Научиться обрабатывать одномерные массивы.

Оборудование и программные средства: персональный компьютер, интегрированная среда программирования Pascal.

Ход работы

1. Изучить операторы объявления одномерных массивов и способы их заполнения, используя справочный материал.
2. Выполнить задание в виде программ, используя соответствующие операторы.
3. Ответить на контрольные вопросы.
4. Сделать выводы.

Краткие теоретические сведения

Массив – представляет собой заранее известное количество однотипных компонентов, снабженных индексами.

Объявление одномерного массива

TYPE

Имя_типа=ARRAY[размерность] OF тип_эл-ов_массива;

VAR

Имя_массива:имя_типа;

Обращение к элементам одномерного массива:

Имя_массива[индекс] := значение;

Заполнение одномерного массива элементами:

FOR индекс:= 1 TO размерность DO

READLN(имя_массива[индекс]); *с клавиатуры*

или

имя_массива[индекс]:=random(B-A+1)+A; *с помощью генератора случайных чисел от A до B*

или

имя_массива[индекс]:=выражение; *явно значением выражения*

Задание

1. Заполнить массив из N элементов с клавиатуры. Увеличить каждый элемент в 2 раза и вывести изменённый массив на экран в обратном порядке.
2. Заполнить массив из N элементов значениями выражения i*2+l (где i – индекс элемента). Вычислить сумму элементов массива, вывести результат и элементы массива на экран.
3. Заполнить массив из N элементов случайными числами из диапазона [- 10,10]. Вывести массив на экран и посчитать количество положительный и отрицательных элементов.

Пример выполнения задания

Задача: Заполнить случайными числами из диапазона [0,1] вещественный линейный массив из N чисел. Найти максимальное значение и его индекс (первый, если таких значений несколько).

```
PROGRAM MASSIV1;
CONST N=20;
VAR
   X:ARRAY[1..N] OF REAL;
   K, KMAX-.INTEGER;
   MAX :REAL;
BEGIN
FOR K:=l TO N DO X[K]=RANDOM;
MAX:=X[1];
KMAX:=1;
FOR K:=2TO N DO
   IF X[K]>MAX THEN
     BEGIN
        MAX:=X[K];
        KMAX=K;
END;
WRITELN (’максимальное значение: X[\KMAX,’]=’,MAX)
END.
```

Контрольные вопросы

1. Дайте определение одномерному массиву?
2. Что нужно указать для определения массива?
3. Что определяет индекс? Что определяет размерность?
4. Как можно заполнить массив случайными числами из диапазона от 5 до 10?
5. Как обратиться к элементу массива?

Практическая работа №10

Тема: Обработка двумерных массивов

Цель: Научиться обрабатывать двумерные массивы.

Оборудование и программные средства: персональный компьютер, интегрированная среда программирования Pascal.

Ход работы

1. Изучить операторы объявления двумерных массивов и способы их заполнения, используя справочный материал.
2. Выполнить задание в виде программ, используя соответствующие операторы.
3. Ответить на контрольные вопросы.
4. Сделать выводы.

Краткие теоретические сведения

Двумерный массив представляет собой заранее известное количество однотипных компонентов представленных в виде матрицы N*M. Элемент снабжен индексами, определяющими номер строки и номер столбца, на пересечении которых он находится.

Объявление двумерного массива

TYPE ИмяТипа =ARRAY[1..N,1..M] OF ТипЭлементов;

VAR ИмяМассива: ИмяТипа;

Тип элементов массива выбирается в зависимости от поставленной задачи.

Обращение к элементам двумерного массива:

ИмяМассива[i, j]значение;

где i – № строки, j – № столбца.

Ввод элементов двумерного массива:

FOR i:= 1 TO NDO

FOR j:- 1 TO M DO

READLN (имя_массива[i, j]);

Двумерный массив заполняется теми же способами, что и одномерный

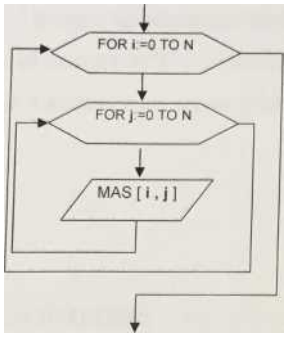


Рисунок 9.1 Алгоритм обработки двумерного массива

Задание

1. Заполнить двумерный массив из M*N элементов целыми числами с клавиатуры. Найти сумму всех элементов и вывести результат на экран.

2. Заполнить случайными числами из диапазона [-10, 10] целочисленный двумерный массив из M*N чисел. Найти минимальное и максимальное значение и его индекс. Вывести первоначальный массив и значение минимума и максимума на экран.

3. Заполнить массив из M* N элементов случайными числами из диапазона [0,10]. Найти сумму элементов каждой строки и сохранить результат во вспомогательном массиве из M элементов. Вывести первоначальный (в виде прямоугольной матрицы) и результирующий массивы на экран.

4. Заполнить массив из M* N элементов случайными числами из диапазона [0,10]. Найти произведение элементов каждого столбца и сохранить результат во вспомогательном массиве из N элементов. Вывести первоначальный (в виде прямоугольной матрицы) и результирующий массивы на экран.

5. Заполнить случайными числами из диапазона [-10,10] целочисленный двумерный массив из M*N чисел. Найти количество отрицательных и положительных .элементов в массиве. Вывести результаты и первоначальный массив в виде прямоугольной матрицы на экран.

Пример выполнения задания

Задача: Заполнить двумерный массив M*N с клавиатуры целыми числами. Увеличить элементы в 2 раза и вывести на экран в виде прямоугольной матрицы.

```
PROGRAM MASSIV;
CONST N=3;
   M=4;
VAR X:ARRAY[1..N,1..M] OF INTEGER;
   K,L:INTEGER;
BEGIN
FOR K:=l TO N DO FORL:=l TOM DO READLN (X[K,L]);
WRITELN;
WRITELN;.
FOR K:=l TOM DO
BEGIN
WRITELN;
FORL:-l TON DO BEGIN
X[K,L]:-X[K,L]*2;
WRITE(’ ’,X[K,L]:4);
END;
END;
END.
```

Контрольные вопросы

1. Дайте определение двумерному массиву?
2. Что нужно указать для определения двумерного массива?
3. Как обратиться к элементу двумерного массива? Что определяют индексы?

Практическая работа №11

Тема: Работа со строковыми переменными

Цель: Научиться работать со строковыми переменными.

Оборудование и программные средства: персональный компьютер, интегрированная среда программирования Pascal.

Ход работы

1. Изучить символьные и строковые типы данных, операторы описания строк и стандартные процедуры и функции для работы со строками, используя справочный материал.
2. Выполнить задание в виде программ, используя соответствующие операторы.
3. Ответить на контрольные вопросы.
4. Сделать выводы.

Краткие теоретические сведения

В Паскале последовательность символов, заключенная в апострофы, называется строкой. Для работы со строками используются 2 типа данных:

– символьный тип данных char
– строковый тип данных string

Описание символьного типа данных char

Диапазон значений: любой символ (код от 0 до 255).

Описание: var a, ch: char;

Обращение к переменным: a:='W'; ch:—'ю';

Описание строкового типа данных string

По сути строка длины К представляет собой массив из К+1 символьных переменных, где нулевой элемент хранит значение длины строки. У типа-строки может быть указан размер (от 1 до 255).

var<имя_стр>: string; !

Обращение к строковым переменным:

strl:='мото'; str2:='цикл';

str3:= str1 + str2; {'мотоцикл'}

Стандартные процедуры и функции для работы со строками

LENGTH (STR) – возвращает длину строки STR;

CONCAT (STR1,STR2) – возвращает объединение строк STR1 и STR2;

COPY (STR,I,J) -возвращает копию подстроки из J символов, начиная с I;

POS (TEXT,STR) – возвращает номер позиции, начиная с которой располагается подстрока TEXT в строке STR;

DELETE (STR,I,J) – удаляет из строки STR J символов, начиная с позиции INSERT (TEXT,STR,I) – вставляет подстроку TEXT в строку STR, начиная с позиции I;

STR (N,STR) – преобразует число N в строковую переменную STR;

VAL (STR,X,COD) – преобразует строку STR в двоичное число X , COD – код неправильного символа;

Задание

Решить задачи на языке Pascal в среде TurboPascal:

1. Написать программу подсчёта количества заданного символа k в строке S. Вывести результат на экран.
2. Написать программу, удаляющую из строки S все пробелы. Вывести преобразованную строку на экран.
3. Написать программу, заменяющую в заданном тексте A одну подстроку b на другую c.

Пример выполнения задания

Задача: Написать программу подсчёта количества символов в заданной строке S без использования стандартной функции length и вывести строчку в обратном порядке. Вывести результат на экран.

```
Program dlina;
Uses crt;
var <
str1:string; I
dlin:integer;||
begin
clrscr;
write('введите строку ');
readln (str1);
dlin:=ord(str1[0]);
for i:=dlin downto 1 do
write(strl [i]);
writeln;
write(‘длина строки’, dlin);
end.
```

Контрольные вопросы

1. Дайте определение типу данных string?
2. Может ли значение функции length(str) быть равным 300?
3. Какое слово называется пустым?

Практическая работа №12

Тема: Работа с данными типа множество

Цель: Научиться работать с данными типа множество.

Оборудование и программные средства: персональный компьютер, интегрированная среда программирования Pascal.

Ход работы

1. Изучить тип данных множество, операторы описания множественного типа и операции над множествами, используя справочный материал.
2. Выполнить задание в виде программ, используя соответствующие операторы.
3. Ответить на контрольные вопросы.
4. Сделать выводы.

Краткие теоретические сведения

Множеством называется совокупность однотипных элементов, рассматриваемых как единое целое.

В Паскале множество может содержать от 0 до 25$ элементов. В отличие от элементов массива элементы множества не пронумерованы и не упорядочены. Выполнение действий возможно только над множеством в целом, с отдельным элементом множества – нельзя.

Конкретные значения множества задаются с помощью конструктора множества, представляющего собой список элементов, заключённый в квадратные скобки. Порядок записи элементов множества в конструкторе не имеет значения.

Описание переменных множественного типа данных:

var<идентификатор>;^ Setof<Тип данных>;

Пример:

var A,D: Set of Integer;

B: Set of‘A’..’Z’;

C: SetofBoolean;

Обращение к переменным:

<идентификатор>:=<множественное выражение>;

Пример:

A:=[50,100,150,200];

B:=[‘b’,’m’,’n’,’k’];

C:=[True, False];

Операции над множествами

В Паскале реализованы основные операции теории множеств: объединение, пересечение и разность множеств

Объединением двух множеств A и B называется множество, состоящее из всех элементов принадлежащих хотя бы одному из множеств A и B (+).

[1,2,3,4]+[3,4,5,6]-> [1,2,3,4,5,6];

Пересечением двух множеств A и B называется множество, состоящее из всех элементов принадлежащих одновременно множеству A и множеству B (-).

[1,2,3,4]*[3,4,5,6]-> [3,4];

Разностью двух множеств A и B называется множество, состоящее из элементов множества A, не принадлежащих множеству B (-).

-[3,4,5,6]-> [1,2];

[3,4,5,6]-[1,2,3,4]->[5,6];

Множества можно сравнить между собой, т.е. для них определены операции отношения. Результатом отношения является логическая величина true или false.

= – совпадение множеств,

<> – несовпадение множеств,

<=, >= – принадлежность одного множества другому.

Операция вхождения (IN) позволяет проверить принадлежит ли элемент данному множеству. Если принадлежит, то True, иначе – False.

Пример:

A:=[50,100,150,200];

50 INA – True,

70INA-False.

Задание

1. Даны 2 символьные строки S1 и S2, содержащие только строчные латинские буквы. Построить строку S3, в которую войдут только общие символы S1 и S2. Вывести её на экран.
2. Составить программу, по которой из последовательности натуральных чисел от 2 до N (l<N<-255) будут выбраны все простые числа. Можно использовать алгоритм «Решето Эратосфена».
3. Составить программу подсчёта количества различных значащих цифр в десятичной записи натурального числа.

Пример выполнения задания

Задача: Дана символьная строка S. Посчитать в ней количество знаков препинания и вывести на экран.

```
program pi;
usescrt;
var
s:string;
k:byte;
begin
clrscr;
write('введите строку ');
readln (s);
for i:=l to length(s) do
if s[i] in ['.',',',';',':','?','!', '-', '*'] then k:=k+l;
writeln('число знаков препинания равно: ',k);
```

```
writeln;
end.
```

Контрольные вопросы

1. Дайте определение типу данных множество?
2. Как синтаксически описываются переменные множественного типа данных?
3. Какие операции над множествами вы знаете?

Практическая работа №13

Тема: Реализация алгоритмов сортировки

Цель: Изучить различные алгоритмы сортировки, научиться использовать их при сортировке массивов.

Оборудование и программные средства: персональный компьютер, интегрированная среда программирования Pascal.

Ход работы

1. Изучить различные алгоритмы сортировки, используя справочный материал. Научиться использовать их при сортировке массивов.
2. Выполнить задание в виде программ, используя соответствующие операторы.
3. Ответить на контрольные вопросы.
4. Сделать выводы.

Краткие теоретические сведения

Методы сортировки

Под сортировкой понимают упорядочение элементов массива или файла по тем или иным признакам. Например, сортировка файла, содержащего фамилии, в алфавитном порядке, сортировка массива чисел по возрастанию или убыванию.

Чтобы сортировать данные, необходимо иметь однозначное трактование критерия сравнения данных. Иными словами, для каждой пары данных не должно возникать сомнения, какое из них будет предыдущим, а какое последующим.

Сортировка методом выборки

В исходном (сортируемом) массиве находим минимальное число. Переписываем его на первое место во втором массиве, а в первом – ’’стираем" (чтобы оно нам потом еще раз не попалось как минимальное). Затем вновь ищем минимальное в первом массиве и переписываем его во второй массив на (уже) второе место. В первом – "стираем". Так продолжаем до тех пор, пока все элементы из первого массива не будут переписаны во второй. Очевидно, что в конце работы программы элементы в выходном (втором) массиве будут расположены в порядке возрастания.

Этот алгоритм не применяется на практике так как, во-первых, съедает памяти вдвое больше, чем занимают сортируемые данные, во-вторых, он очень медлителен.

Метод перестановок (пузырьковый метод)

Просматриваем все элементы от первого до предпоследнего. Если очередной элемент больше последующего, то что-то неправильное есть в их расположении (ведь сортируем по возрастанию, а впереди – меньший). Не знаем, что там потом будет, но эти два давайте поменяем местами.

Условие перестановки соседних данных: текущий элемент меньше (больше) предыдущего, если сортировка идет по возрастанию (убыванию)»

В цикле: если очередной элемент a[i] меньше или равен a[i + 1], то их взаимное расположение отвечает требованиям сортировки (сортируем ведь

по возрастанию!), и менять их местами не нужно. В противном случае – меняем местами элементы a[i] и a[i + 1]. При этом увеличилось число перестановок.

При каждом проходе будем подсчитывать количество перестановок. Счетчик перестановок обнуляем перед каждым проходом.

Если перестановки были, проход нужно повторить. Если же нет, то сортировка завершена.

К достоинствам метода следует отнести простоту, малый расход памяти.

Недостатки метода: медлительность, избыточность.

Сортировка методами возврата

Практически все применяемые в настоящее время современные методы сортировок можно с той или иной степенью приближения назвать методами возврата.

Просматриваем все элементы от второго до последнего. Если очередной элемент больше предыдущего, значит, пока он стоит на своем месте. Если нет, то он должен стоять где-то левее, ’'вынимаем" этот элемент из массива (запоминаем его значение), чтобы "понести" его к началу, предыдущий продвигаем на его место (освободившееся)

Оцениваем ситуацию: здесь ли он должен стоять, или еще левее? Если левее, то продвигаем следующий. Так в конце концов находим место для нашего “вынутого” и там его размещаем (ну, а не найдем, значит, он первый). • Потом вспоминаем, где остановились и продолжаем проход с прерванной точки.

В цикле: найдя элемент, стоящий не на своем месте, "вынимаем" его из массива (переменная к) и осуществляем возврат к началу в поисках места для этого элемента. Продвинув предыдущий на единицу вперед, оцениваем место для элемента. Очевидно, оно найдено, если элемент в этой точке оказался больше, чем предыдущий. Если это не так, то поиск продолжаем с прерванной/ точки, если еще не достигли конца массива.

По сравнению с методами выборки и перестановок, алгоритмы возврата дают существенный выигрыш во времени. Расход памяти также минимален. И. всё же такой алгоритм избыточен.

Попробуем иначе.

Итак, "вынули" элемент, который должен стоять где-то левее (т.е. между первым и текущим, i-ым). Делим этот отрезок пополам. Это будет номер к:= (i+1) div 2) ("начало" + "конец" разделить на 2, причем, целая часть этого выражения; не может же быть номер элемента дробным). Оцениваем новое место для элемента.

Если оно подходит (элемент больше, чем a[k] и не больше чем a[k + 1]), то тогда поступаем так: все элементы между k + 1 и i – 1 сдвигаем на шаг вперед (вспоминайте: цикл сдвига должен иметь обратное направление – от i – 1 до k + 1 с шагом -1), элемент записываем на место k + 1 и продолжаем сортировку с точки i + 1 (со следующей).

Если же выбранное делением отрезка пополам место оказалось для элемента неподходящим, то будем искать дальше. Для этого нужно выяснить, где расположено нужное место: слева или справа от только что выбранного (от точки k). Очевидно, это определяется условиями: если a[i] > a[k + 1], то справа; в этом случае новое место k = (k + i) div 2, если же a[i] < a[k], то слева, и тогда новое k := (1 + к) Div 2.

Здесь полезно вспомнить метод половинного деления.

Алгоритм поиска номера элемента методом половинного деления обладает одним существенным недостатком: для первого элемента нет предыдущего, а для последнего нет последующего.

Чтобы не обрабатывать исключительные ситуации на практике в программах добавляют на время сортировки к массиву еще два элемента (фиктивных) и располагают их в начале и конце сортируемого массива. При сортировке по возрастанию выбирают такие элементы, которые заведомо меньше минимального (начальный) и больше максимального (конечный). После сортировки эти элементы удаляют из массива.

Сортировка рекурсией.

Этот метод был разработан в 1962 г. английским программистом К.Хоаром. Суть его в следующем:

1. Выбираем центральный элемент исходного массива X и записываем его в переменную – посредник A. Потом элементы исходного массива просматриваются в левой части слева – направо, а в правой части справа – налево. В левой части находим элемент x[i], который больше или равен A .и запоминаем его позицию. В правой части ищем такой элемент и его позицию, который меньше или равен A. Найденные элементы меняем местами и продолжаем встречный поиск с теми же самыми условиями. На этом первый этап закончен, причём массив окажется разделённым таким образом, чпэ элементы со значениями меньше или равно A будут в левой части массива, все остальные – в правой.

2. А теперь делим уже левые и правые части пополам... до тех пор, пока длина сортируемых частей не станет длиной в один элемент!

Естественно, что поскольку происходит возврат к одной и той же процедуре, то и оформить программу логично с помощью рекурсии.

```
Program P_51;
Uses Crt;
Varx: Array [1..20] Of Integer;
  i: Word;
  Procedure Sort(l, r: Word);
  Var a, tmp: Integer;
i ,j: Word;
  Begin
  a := x[(l + r) div 2];
  i:= l;
  j:= r;
    While i< = j Do
```

```
    Begin
    While x[i] < a do i:= i + 1;
    While x[i] > a do i:= i – 1;
    If i< j Then
    Begin
      tmp:= x[i];
      x[i]:= x[j];
      x[j]:=tmp;
      i:=i+l;
      j:=j-1
      End
    End;
  If 1 < j Then Sort(l, j);
  If 1 < r Then Sort(i, r)
  End;
Begin
ClrScr;
WriteLn('Введите элементы массива: ');
For i:=l to 20 Do Read(x[i]);
ReadLn;
Sort(l, 20);
WriteLn('Отсортированный массив: ');
For i:=T To 20 Do Write(x[i]:8);
WriteLn;
ReadKey
End.
```

Сортировка	Время выполнения
Методом выборки	Несколько дней
Методом перестановок	Несколько часов
Методом возврата с алгоритмом поиска номера элемента простым перебором	10 мин.
Методом рекурсии	6 мин.

Таблица 13.1 – Анализ времени работы различных методов сортировки

Сравнительный анализ времени работы различных методов сортировки приведен в таблице. Здесь приведены опытные данные, полученные при внешней сортировке файла прямого доступа, содержащего 25 тысяч записей по 128 байт. Программы выполнялись на компьютере с процессором AMDAthlonXP-2000 и RAM 256 Mb.

Задание

1. Задан одномерный массив. Сортировать его в таком порядке, чтобы в начале были расположены отрицательные элементы, затем нули, а в

конце – положительные (без нарушения их положения друг относительно друга в исходном массиве).

2. Сортировать одномерный массив в параболическом порядке (по краям должны быть расположены максимальные значения, которые должны убывать по мере приближения к середине).

3. Сортировать главную диагональ квадратной матрицы в порядке возрастания.

4. Задан двумерный массив. Разместить его элементы в одномерном массиве в порядке возрастания модулей.

5. Двумерный массив сортировать таким образом, чтобы элементы каждой строки и каждого столбца были расположены в порядке возрастания.

Пример выполнения задания

Задача: Сортировать строки матрицы пузырьковым методом.

```
…
For i:=1 TonDo
Repeat {Сортируем все строки от 1 до n}
l:= 0; {пузырьковым методом (1- число перестановок)}
For j:= 2 To m Do {цикл сортировки i-й строки}
If a[i.j] <= a[i,j-1] Then
Begin
d:= a[i,j]; {Сортируем по возрастанию. Чтобы сортировать}
a[i,j]:= a[i,j-l]; {по убыванию, нужно изменить знак}
a[i, j-1]:= d;{< = на => – в операторе If.}
l:=l+1
End;
Until l=0;
…
```

Контрольные вопросы

1. Запишите все алгоритмы сортировки.

Практическая работа №14

Тема: Работа с файлами последовательного доступа

Цель: научиться работать с файлами последовательного доступа.

Оборудование и программные средства: персональный компьютер, интегрированная среда программирования Pascal.

Ход работы

1. Изучить файловый тип данных, классификацию файлов, синтаксис описания файлового типа, процедуры и функции для работы с файлами, используя справочный материал.
2. Выполнить задание в виде программ, используя соответствующие операторы.
3. Ответить на контрольные вопросы.
4. Сделать выводы.

Краткие теоретические сведения

Файловый тип представляет собой последовательность однотипных компонент, расположенных на внешнем носителе.

Классификация файлов:

- По логической структуре (типу):
  а. текстовые (TEXT);
  б. типизированные (FILE OF<тип>);
  в. файлы без типа (FILE).
- По способу доступа:
  а. файлы последовательного доступа
  б. файлы произвольного доступа

Записи в файлах последовательного доступа доступны в порядке очереди.

Та запись, которая в настоящее время текущая, называется указателем файла. Писать в последовательные файлы можно только в конец, добавлять записи в "хвост" файла. Длина записи может быть различной и внутри одного файла могут быть записи разной длины.

Описание файлового типа: |

TYPE<имя типа> = FILEOF<тип-данных>;

Объявление файловых переменных:

VAR<имя_переменной>:<имя_типа>;

Пример:

TYPE NUM = FILE OF INTEGER;

VARF1:NUM;

Перед началом работы, файловая переменная должна быть связанна с конкретным внешним файлом. Затем файл должен быть открыт для чтения и/или записи. После этого можно осуществлять организацию ввода-вывода. После окончания работы файл должен быть закрыт.

Процедуры и функции для работы с файлами:

ASSIGN(<ФайлПерем>,’<ПолноеИмяФайла>’); {Процедура ASSIGN связывает файловую переменную с конкретным внешним файлом}

RESET (<ФайлПерем>); {RESET – открывает существующий физический файл, который связан с файловой переменной}

WRITE(<ФайлПерем>,<элемент>); {WRITE – записывает данные в файл, который связан с файловой переменной}

CLOSE (<ФайлПерем>); {Процедура CLOSE – закрывает файл, который связан с файловой переменной}

Задание

В файле F записано N целых чисел в диапазоне от -20 до +20. Написать программу, которая считывает числа из файла F и положительные числа записывает в файл G, а отрицательные – в файл H. Содержимое файлов G и H вывести на экран.

Пример выполнения задания

Открыть файл f86.txt и сохранить в нем N целых чисел в пределах от 65 до 90:

а) считайте информацию из данного файла;

б) найдите сумму элементов, находящихся в файле f86.txt

```
program fil86;
uses crt;
var f:file of integer;
n,m,i,S: integer;
a: string;
begin
clrscr;
a:= 'd:\uroki\f86.txt';
assign(f,a);
rewrite(f);
randomize;
write('n=');
readln(n);
for i:= 1 to n do begin
m:=trunc(random(25))+65;
write(f,m);
write(m,' ');
end;
close(f);
Write (n);
Writeln;
assign(f,a);
reset(f);
while not eof (f) do begin
read(f,n);
write(n,' ');
s:=s+n
end;
close(f);
writeln('s=' ,s);
readln;
end.
```

Контрольные вопросы

1. Что такое файл?
2. Какие типы файлов вы знаете?
3. Опишите процедуры для чтения из файла и записи в файл

Практическая работа № 15

Тема: Работа с файлами произвольного доступа

Цель: научиться работать с файлами произвольного доступа.

Оборудование и программные средства: персональный компьютер, интегрированная среда программирования Pascal.

Ход работы

1. Изучить файловый тип данных, классификацию файлов, синтаксис описания файлового типа, процедуры и функции для работы с файлами, используя справочный материал.
2. Выполнить задание в виде программ, используя соответствующие операторы.
3. Ответить на контрольные вопросы.
4. Сделать выводы.

Краткие теоретические сведения

Файловый тип представляет собой последовательность однотипных компонент, расположенных на внешнем носителе.

Классификация файлов:

- По логической структуре (типу):
  а. текстовые (TEXT);
  б. типизированные (FILE OF «ran»);
  в. файлы без типа (FILE).
- По способу доступа:
  а. файлы последовательного доступа
  б. файлы произвольного доступа

В файлах произвольного доступа можно прочитать (записать) любую запись, указав ее номер (порядковый номер от начала файла). Здесь все записи одинаковой величины, поэтому не трудно вычислить, где располагается запись номер N. Система производит эти вычисления автоматически при обращении к записи в файле произвольного доступа. Символы – разделители записей в файлах прямого доступа отсутствуют.

Текстовые файлы – это файлы, содержащие совокупность символов, разделённых на строки. В конце каждой строки стоит символ конца строки. Процедуры и функции для работы с файлами:

APPEND(<ФайлПерем>); Процедура APPEND открывает текстовый файл, который связан с файловой переменной.

READLN(<ФайлПерем>,<список_элементов>); Процедура READLN читает (считывает) строку из файла, который связан с файловой переменной.

WRITELN(<ФайлПерем>,<список_элементов>); Процедура WRITELN записывает строку в файл, который связан с файловой переменной.

EOLN(<ФайлПерем>); Процедура EOLN предназначена для определения конца строки файла

SEEKOF (<ФайлПерем>); Функция SEEKOF предназначена для перемещения в конец файла

SEEKOLN(<ФайлПерем>); Функция SEEKOLN предназначена для перемещения в конец строки файла

Типизированные файлы – это файлы, содержащие совокупность символов, разделённых на строки. В конце каждой строки стоит символ конца строки. Процедуры и функции для работы с файлами:

Seek (<файл_перем>, ‘<номер_элемента’); Процедура Seek позволяет явно изменить значение текущего указателя файла (номер элемента).

FileSize(<файл_перемен>); Процедура FileSize возвращает общее число элементов файла.

FilePos (<файл_перемен>); Процедура FilePos возвращает номер элемента файла, на который установлен текущий указатель.

Задание

Даны текстовый файл F и строка S. Посчитать количество строк в файле F, совпадающих со строкой S и результат записать в файл G.

Контрольные вопросы

1. Каковы особенности файлов произвольного доступа?
2. Что собой представляют типизированные файлы?
3. Что собой представляют текстовые файлы?

Практическая работа №16

Тема: Разработка программ со структурированными типами данных

Цель: Научиться работать со структурированным типом данных RECORD. Оборудование и программные средства: персональный компьютер, интегрированная среда программирования Pascal.

Ход работы

1. Изучить тип данных RECOKO, операторы описания типа RECORD и способы доступа к полям записи, используя справочный материал.
2. Выполнить задание в виде программ, используя соответствующие операторы.
3. Ответить на контрольные вопросы.
4. Сделать выводы.

Краткие теоретические сведения

Запись включает в себя несколько полей, тип которых может отличаться друг от друга.

Синтаксис описания типа RECORD:

TYPE
<имя_типа> = RECORD
<поле 1>:<тип_данных1 >;
<поле2>:<тип_данных2>;
…
<noneN> :<тип_данных>;
END;
VAR
<имя_записи>: <имя_типа>;

Доступ к полям записи осуществляется по имени записи и имени поля (через точку) либо с помощью оператора WITH:

<имя_записи>.<имя_поля>:=<значение>;
WITH <имя записи>DO<оператор>;

Задание

1. Создать базу данных группы студентов, используя массив элементов типа RECORD (поля: Фамилия, Имя, год рождения, средний бал). Заполнить базу данными и вывести на экран в виде таблицы.
2. Создать базу данных товаров, используя массив элементов типа RECORD (поля: Код, Наименование, производитель, цена). Заполнить базу данными и вывести на экран в виде таблицы.

Пример выполнения задания

Задача 1: Создать базу данных группы студентов, используя массив элементов типа RECORD (поля: Фамилия, Имя, год рождения, средний бал). Заполнить базу данными и вывести на экран в виде таблицы.

…
type
student=record

```
name: string;
data: integer;
iq: real;
end;
var
group: array[10] of student;
f: file of student;
```

Контрольные вопросы

1. Каково назначение типа данных RECORD?
2. Перечислите способы доступа к полям записи?

ЗАКЛЮЧЕНИЕ

Методическое пособие для проведения практических занятий по дисциплине Основы алгоритмизации и программирования специальности 09.02.01 Компьютерные системы и комплексы предназначено для самостоятельного выполнения практических работ в шестом и седьмом семестрах семестре третьего и четвёртого курсов. Пособие содержит необходимый теоретический материал задания, примеры выполнения, правила составления отчета, а также контрольные вопросы, которые необходимо отразить в отчете. Для правильного и окончательного проведения практических занятий необходимо руководствоваться также теоретическим материалом из курса лекций по предмету. Добросовестное выполнение выше описанных рекомендаций гарантирует хорошее усвоение предмета обучающимися.

Список используемой литературы

Основные источники:

1. Трофимов, В. В. Основы алгоритмизации и программирования : учебник для среднего профессионального образования / В. В. Трофимов, Т. А. Павловская ; под редакцией В. В. Трофимова. — Москва : Издательство Юрайт, 2022. — 137 с. — (Профессиональное образование). — ISBN 978-5-534-07321-8. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/493261.

2. Черпаков, И. В. Основы программирования : учебник и практикум для среднего профессионального образования / И. В. Черпаков. — Москва : Издательство Юрайт, 2022. — 219 с. — (Профессиональное образование). — ISBN 978-5-9916-9984-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/491068.

3. Огнева, М. В. Программирование на языке С++: практический курс : учебное пособие для среднего профессионального образования / М. В. Огнева, Е. В. Кудрина. — Москва : Издательство Юрайт, 2022. — 335 с. — (Профессиональное образование). — ISBN 978-5-534-05780-5. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/493047.

4. Мейер Б. Инструменты, алгоритмы и структуры данных [Электронный ресурс] / Б. Мейер. – 2-е изд. – Электрон. текстовые данные. – М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2021. – 542 c. – 2227-8397. – Режим доступа: http://www.iprbookshop.ru/73680.html

5. Борисенко В.В. Основы программирования [Электронный ресурс] / В.В. Борисенко. – Электрон. текстовые данные. – М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2020. – 323 c. – 978-5-9556-00039-0. – Режим доступа: http://www.iprbookshop.ru/52206.html

Интернет-ресурсы

www.ttgt.org (Сайт Тихорецкого Техникума Железнодорожного Транспорта)

www.studentlibrary.ru (Электронная библиотека)

www.urait.ru (Электронная библиотека)

www.fcior.edu.ru (Федеральный центр информационно-образовательных ресурсов – ФЦИОР).

www.school-collection.edu.ru (Единая коллекция цифровых образовательных ресурсов).

www.intuit.ru/studies/courses (Открытые интернет-курсы «Интуит» по курсу «Информатика»).

www.lms.iite.unesco.org (Открытые электронные курсы «ИИТО ЮНЕСКО» по информационным технологиям).

www.ru.iite.unesco.org/publications (Открытая электронная библиотека «ИИТО ЮНЕСКО» по ИКТ в образовании).