|  |  |
| --- | --- |
| Uni21b | МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ,  ФИЛИАЛ В Г.НИЖНЕВАРТОВСКЕ  КАФЕДРА «ИНФОРМАТИКА» |

**СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ**

Контрольные задания и методические указания к выполнению

РГР

для студентов направления   
«Информатика и вычислительная техника»

заочной формы обучения

Нижневартовск

2014

УДК

© Никонова Е.З.

# Одобрено

# редакционно-издательским советом филиала

# (протокол № 2 от 16.10.2014)

# Структуры и алгоритмы обработки данных: Контрольные задания и методические указания к выполнению РГР для студентов направлений «Информатика и вычислительная техника» заочной формы обучения / Е.З. Никонова – Нижневартовск, 2014. – 41 с.

# Задания составлены в соответствии с ФГОС-3 по направлению обучения «Информатика и вычислительная техника» и предназначены для формирования компетенций при изучении данной дисциплины

# Рецензент:

# доцент кафедры естественнонаучных и гуманитарных дисциплин, к.ф.-м.н., О.Р. Нурисламов;

# Утверждено на заседании кафедры

# Протокол №2

# «9» октября 2014 год

1. **ДВУМЕРНЫЕ МАССИВЫ**

**🕮 Теоретические сведения**

Двумерные массивы являются аналогами матриц. Первый индекс элемента двумерного массива определяет номер строки, а второй – номер столбца, на пересечении которых расположен элемент. Строки и столбцы нумеруются либо от единого заранее установленного минимального значения индекса, либо от граничного значения, заданного одновременно с объявлением массива***.***

Описание двухмерного массива производится следующим образом:

***Const***

*N = ранг\_матрицы;*

***Type***

*matr=****array*** *[1..N,1..N]* ***of*** *тип\_элементов\_матрицы;*

***Var***

*Имя\_матрицы : matr;*

Или по упрощенной схеме:

***Var*** *<имя массива>:* ***array*** *[1.. кол-во\_строк, 1..кол-во\_столбцов] of тип\_элементов\_массива;*

Для задания значений двухмерного массива можно воспользоваться процедурой с использованием датчика случайных чисел.

Например:

***Program*** *Array\_Full;*

***Const*** *n=10;*

***Type*** *mas =* ***array*** *[1..n, 1..n]* ***of*** *Integer;*

***Var*** *m : mas;*

***Procedure*** *Enter (****Var*** *tabl: mas);*

***Var*** *i, j: Integer;*

***Begin***

***For*** *i:= 1* ***to*** *n* ***do***

***For*** *j:= 1* ***to*** *n* ***do***

*tabl[i,j]:=Random(10);*

***End;***

***Begin***

*Randomize;*

*Enter (m);*

***End.***

С помощью процедуры Enter осуществляется ввод двумерного массива размерностью 10х10 случайными числами из диапазона от 0 до 9. Задание диапазона значений определяется константой n=10.

**🖳 Примеры решений задач**

Пример программы ввода/вывода двумерного массива с использованием процедур (в одной из процедур используется проце-дура GotoXY, содержащаяся в модуле CRT. Поэтому необходимо при использовании этой процедуры указать имя библиотеки CRT в разделе описания библиотек **Uses**). Дополнительно программа ищет максимальный элемент в массиве.

**Program** Mas\_example;

**Uses** Crt;

**Const** n=10;

**Type** mas = **array** [1..n, 1..n] **of** Integer; // объявляем тип данных для массива

**Var** m : mas; // объявляем переменную типа mas, т.е. массив

k1,k2:integer; // объявляем переменные для кол-ва строк и столбцов

**Procedure** Enter (Var tabl: mas; x,y: integer); // процедура ввода данных в массив, параметры -

**Var** i, j: Integer; // массив и кол-во строк и столбцов

**Begin**

**For** i:= 1 **to** x **do** // цикл для **строк**

**For** j:= 1 **to** y **do** // цикл для **столбцов**

tabl[i,j]:=Random(10); // заполняем текущий элемент массива случайным числом из промежутка

**End**; // от 0 до 9

**Procedure** List(**var** tabl:mas; x,y: integer); // процедура вывода массива на экран, параметры -

**Var** i, j: integer; // массив и кол-во строк и столбцов

**Begin**

**For** i:=1 **to** x do

**For** j:=1 **to** y **do**

**begin**

gotoxy(3\*j, i+3); //вызываем процедуру gotoxy, она перемещает курсор в нужную позицию // текстового экрана, 1-й параметр – по оси ОХ (т.е. для столбцов), // второй – по оси ОУ (т.е. для строк)

write (tabl[i,j]); // выводим на экран очередной элемент

**end**;

writeln; // делаем переход на новую строку

**End**;

**Procedure** mx\_elmnt(**var** tabl:mas; x,y: integer); // процедура поиска максимального эл-та в массиве

**Var** i, j, max: integer;

**Begin**

max:=tabl[1,1]; // делаем по умолчанию максимальным 1-й элемент массива

**For** i:=1 **to** x **do**

**For** j:=1 **to** y **do**

**if** max<tabl[i,j] **then** max:=tabl[i,j]; // ищем максимальный

writeln('Максимальный элемент в массиве: ', max); // выводим его

**end**;

**Begin**

**Randomize**;

write('Введите кол-во строк массива (<10): ');

readln(k1);

write('Введите кол-во столбцов массива (<10): ');

readln(k2);

Enter(m,k1,k2); // вызываем нашу процедуру с нужными параметрами для ввода значений в массив

list(m,k1,k2); // вызываем нашу процедуру с нужными параметрами для вывода массива на экран

mx\_elmnt(m,k1,k2);// вызываем процедуру для нахождения максимального элемента

**End**.

На экране появится примерно следующее:

Введите кол-во строк массива (<10): 3

Введите кол-во столбцов массива (<10): 4

0 6 4 5

3 6 1 5

5 8 7 6

**🖮 Задачи для самостоятельного решения**

1. Даны действительные числа A1, ..., An (можно использовать массив из n элементов, заполняемый случайным образом), действительная квадратная матрица порядка n (n>=6). Получить действительную матрицу размера nх(n+1), вставив в исходную матрицу между пятым и шестым столбцами новый столбец с элементами A1, ..., An.
2. Дана целочисленная матрица размера 6х9. Получить новую матрицу, получающуюся из данной перестановкой столбцов – первого и последнего, второго с предпоследним и т.д.;
3. Даны целые числа A1, ..., A10 (можно использовать массив из 10 элементов, заполняемый случайным образом), целочисленная квадратная матрица порядка n. Заменить нулями в матрице те элементы с четной суммой индексов, для которых имеются равные среди A1, ..., A10.
4. Дана целочисленная матрица размера 6х9. Получить новую матрицу, получающуюся из данной перестановкой строк – первой и последней, второй с предпоследней и т.д.;
5. Дана действительная квадратная матрица порядка n. Преобразовать матрицу по правилу: строку с номером n сделать столбцом с номером n, столбец с номером n сделать строкой с номером n.
6. Дана квадратная матрица A[n, n]. Записать на место отрицательных элементов матрицы нули, а на место положительных – единицы.
7. В данной действительной квадратной матрице порядка n найти наибольший по модулю элемент. Получить квадратную матрицу порядка n-1 путем выбрасывания из исходной матрицы какой-нибудь строки и столбца, на пересечении которых расположен элемент с найденным значением.
8. Дана действительная квадратная матрица порядка n. Найти наибольший элемент среди стоящих на главной и побочной диагоналях и поменять его местами с элементом, стоящим на пересечении этих диагоналей.
9. Вычислить сумму и число положительных элементов матрицы A[N, N], находящихся над главной диагональю.
10. Дана матрица А размером n х m. Определить к — количество особых элементов массива А, считая его элемент особым, если он больше суммы остальных элементов его столбца.
11. Задана квадратная матрица. Поменять местами строку с максимальным элементом на главной диагонали со строкой с за- данным номером m.
12. Дана матрица B[N, M]. Найти в каждой строке матрицы максимальный и минимальный элементы и поменять их местами с первым и последним элементом строки соответственно.
13. Дана целая квадратная матрица n-го порядка. Определить, является ли она магическим квадратом, т.е. такой, в которой суммы элементов во всех строках и столбцах одинаковы.
14. Элемент матрицы назовем седловой точкой, если он является наименьшим в своей строке и одновременно наибольшим в своем столбце или, наоборот, является наибольшим в своей строке и наименьшим в своем столбце. Для заданной целой матрицы размером n х m напечатать индексы всех ее седловых точек.
15. Дана матрица размером n х m. Переставляя ее строки и столбцы, добиться того, чтобы наибольший элемент (или один из них) оказался в верхнем левом углу.
16. Определить, является ли данная целая квадратная матрица n-го порядка симметричной (относительно главной диагонали).
17. Дана действительная матрица размером n х m. В каждой строке выбирается элемент с наименьшим значением, затем среди этих чисел выбирается наибольшее. Указать индексы элемента с найденным значением.
18. Найти наибольший и наименьший элементы прямоугольной матрицы и поменять их местами.
19. Дана прямоугольная матрица. Найти строку с наибольшей и наименьшей суммой элементов. Вывести найденные строки и суммы их элементов.
20. В данной действительной квадратной матрице порядка n найти сумму элементов строки, в которой расположен элемент с наименьшим значением.
21. Пусть дана действительная матрица размером n х m. Требуется преобразовать матрицу следующим образом: поэлементно вычесть последнюю строку из всех строк, кроме последней.
22. Определить наименьший элемент каждой четной строки матрицы А[М, N].
23. Определить номера строк матрицы R[M, N], хотя бы один элемент которых равен c, и элементы этих строк умножить на d.

**2. ТИП ДАННЫХ «МНОЖЕСТВО»**

**🕮 Теоретические сведения**

Множество – набор однотипных элементов базового типа, каким-то образом связанных друг с другом. Базовый тип – это порядковый тип, кроме word, integer, longint.

Число элементов исходного множества не может быть больше 256, а порядковые номера элементов должны находиться в пределах от 0 до 255.

**Type**

<*имя множества*> = **set of** <*тип компонент*>;

**Var**

<*переменная*>:<*имя множества*>;

**Допустимые операции с множествами:**

+ объединение;

– разность;

\* пересечение;

= , <= , >= проверка эквивалентности двух множеств;

< > проверка неэквивалентности двух множеств;

**in** логический оператор проверки присутствия компоненты во множестве.

**🖳 Примеры решений задач**

1. *Ввести строку символов, состоящую из латинских букв, цифр и пробелов. Осуществить проверку правильности введенных символов (т.е. чтобы там были только цифры, латинские буквы и пробелы).*

**Program** stroka;

**Var**

Str: **string;**

L: **byte;**

Test: **boolean;**

**Begin**

**Writeln** (‘Введите строку’);

**Readln** (str);

L:**=Length** (Str);

Test:= L>0**;**

**While** Test **and** (L>0) **do**

*{выход из цикла будет осуществлен, если переменная test станет равна false или L станет равна 0}*

**Begin**

**Test:=Str**[L] **in [**‘0’..’9’,’A’..’Z’,’a’..’z’,’ ‘];

{test = true, если str[L] является одним из перечисленных символов}

{иначе test = false}

**Dec** (L) {аналог L:=L – 1}

**End;**

**If** Test **then**

**WriteLn** (‘Правильная строка’)

**Else**

**WriteLn** (‘Неправильная строка’);

**End.**

1. *Другое решение данной задачи*

**Program** stroka;

**Var**

Str: **string**;

L,i: **byte**;

Test: **boolean**;

**Begin**

**Writeln** ('Введите строку');

**Readln**(str);

L:=**Length** (Str);

**for** i:=1 **to** L **do**

**Begin**

test:=Str[i] **in** ['0'..'9','A'..'Z','a'..'z',' '];

**if** test=**false then break**;

**End**;

**If** Test **then WriteLn** ('Правильная строка')

**Else WriteLn** ('Неправильная строка');

**End**.

1. *Еще одно решение этой задачи*

**Program** *stroka;*

**Var**

*Str:* **string***;*

*L,i,k:* **byte***;*

**Begin**

**Writeln** *('Введите строку');*

**Readln***(str);*

*L:=***Length** *(Str);*

**for** *i:=1* **to** *L* **do**

**Begin**

**if** *Str[i]* **in** *['0'..'9','A'..'Z','a'..'z',' ']* **then** *k:=k+1;*

**End***;*

**If** *k=L* **then**

**WriteLn** *('Правильная строка')*

**Else**

**WriteLn** *('Неправильная строка');*

**End***.*

1. *Заполнить множество A путем ввода n значений:*

**Program** mnozh;

**var** A: **set of** 1..200;

j,x,n: byte;

**begin**

write('Введите кол-во эл-ов в мн-ве: ');

readln(n);

A:=[]; // задаем пустое мн-во

for j := 1 **to** n **do**

**begin**

write('Введите эл-нт мн-ва: ');

readln(x);

A:=A+[x] // добавляем в мн-во введенный элемент

**end**;

write('Вот ваше мн-во: ');

**for** x := 1 **to** 200 **do**

**if** x **in** A **then** write(x:3); // если х в мн-ве, то напечатать его

writeln;

**end**.

**🖮 Задачи для самостоятельного решения**

1. Дано множество, элементами которого являются буквы от а до f и от x до z. Требуется ввести с клавиатуры некую последовательность символов, и выяснить, какие из них входят в заданное множество.
2. Имеется множество, содержащее натуральные числа из некоторого диапазона. Сформировать два множества, первое из которых содержит все простые числа из данного множества, а второе — все составные.
3. Дан текст из строчных латинских букв, за которым следует точка. Определить, каких букв – гласных (a, e, i, o, u) или согласных – больше в этом тексте.
4. Напечатать в возрастающем порядке все цифры, не входящие в десятичную запись натурального числа n.
5. Дан текст из строчных латинских букв, за которым следует точка. Напечатать (5-7):
6. все буквы, входящие в текст не менее двух раз;
7. все буквы, входящие в текст по одному разу;
8. первые вхождения букв в текст, сохраняя их исходный взаимный порядок.
9. Дана последовательность целых чисел. Определить, является ли эта последовательность перестановкой заданного отрезка элементов натурального ряда.
10. Подсчитать количество чётных цифр в исходной символьной строке и распечатать все, кроме пробелов, знаков операций и знаков препинания.
11. Сформировать множество, в которое входят только латинские буквы, встретившиеся во входной строке, и множество знаков препинания из входной строки.
12. Сформировать множество, в которое входят только цифры, встретившиеся во входной строке.
13. Сформировать множество, в которое входят только большие латинские буквы.
14. Дан текст на русском языке. Напечатать в алфавитном порядке все гласные буквы, которые входят в каждое слово.
15. Дан текст на русском языке. Напечатать в алфавитном порядке все звонкие согласные буквы, которые входят более чем в одно слово.
16. Подсчитать количество различных цифр в десятичной записи натурального числа.

**3. ТИП ДАННЫХ «ЗАПИСЬ»**

**🕮 Теоретические сведения**

Тип запись включает ряд компонент, называемых полями, которые могут быть разных типов. При задании типа-записи после зарезервированного слова **record** следует перечислять все поля типа записи с указанием через двоеточие их типов и завершить задание типа словом **end.** Поля отделяются друг от друга точкой с запятой. Количество полей записи может быть любым.

Тип данных запись описывается следующим образом:

**Type**

**Record**

<*имя поля 1*>:<*тип поля 1*>;

<*имя поля 2*>:<*тип поля 2*>;

. . .

<*имя поля n*>:<*тип поля n*>;

**End**;

Если тип нескольких полей совпадает, то имена полей могут быть просто перечислены. После объявления в программе переменной типа «запись» к каждому ее полю можно обратиться, указав сначала идентификатор переменной-записи, а затем через точку – имя поля. Поле записи может иметь практически любой тип. Доступ к вложенным элементам таких структур осуществляется по тем же правила, как и обычно.

Переменная типа «запись» может участвовать только в операциях присваивания. Но поле записи может принимать участие во всех операциях, применимых к типу этого поля.

Присвоение значений переменным типа запись производится одним из двух способов:

1) <*имя переменной*>**.**<*имя поля 1*>:=<*выражение*>;

<*имя переменной*>**.**<*имя поля 2*>:=<*выражение*>;

. . . . . . . .

<*имя переменной*>**.**<*имя поля n*>:=<*выражение*>;

2) для облегчения работы с полями записей вводится оператор присоединения.

**With** <*имя переменной*> **do**

**begin**

<*имя поля 1*>:=<*выражение*>

<*имя поля 2*>:=<*выражение*>

. . . . .

<*имя поля n*>:=<*выражение*>

**end**;

**🖳 Примеры решений задач**

1. *Для каждого студента указаны фамилия и оценки в баллах по пяти дисциплинам. Требуется вычислить средний балл каж-дого студента и максимальный средний балл.*

**Program** BAL;

**Uses** crt;

**Type** zap=**Record //** создаем zap типа Record

Fam: string; // поле с фамилией

B1, B2, B3, B4, B5: 2..5; // поля с оценками (тип множество от 2 до 5)

SB: real; // поле для хранения среднего бала

**End**;

**Var**

Tbl: **array**[1..10] **of** zap; // создаем массив типа zap

I,n: integer;

max: real; // для хранения максимального значения

max\_name: string;

**Begin**

write('Введите кол-во записей: ');

readln(n);

**For** I:=1 **to** n **do**

**Begin**

Write('Введите данные через Enter (фамилия и 5 оценок)');

// считываем данные для каждого поля каждого элемента массива

Readln (Tbl[I].Fam, Tbl[I].B1, Tbl[I].B2, Tbl[I].B3, Tbl[I].B4,Tbl[I].B5);

**end**;

max:=0; // зануляем текущий максимум

**For** I:=1 **to** n **do**

**begin**

Tbl[I].SB:=(Tbl[I].B1+Tbl[I].B2+Tbl[I].B3+Tbl[I].B4+Tbl[I].B5)/5; // считаем ср. балл

Writeln('Средняя оценка у ', Tbl[I].FAM,' равна ',Tbl[I].SB); //выводим ср. балл

**if** Tbl[I].SB>max **then** // если текущий эл-нт массива больше максимума

**begin**

max:=Tbl[I].SB; // записываем текущий максимум

max\_name:=Tbl[I].FAM; //записываем фамилию студента с текущим максимальным ср. баллом

**end**;

**end**;

writeln('Максимальный средний балл у студента с фамилией ',max\_name, ', он равен ',max);

**End**.

**🖮 Задачи для самостоятельного решения**

1. Сформировать переменную типа запись, в которой расположены данные о каждом отдельном ученике в следующем порядке: имя (15 символов), фамилия (15 символов), год обучения (целое число), буква (символ). Требуется перенести эти данные в другую переменную, выводя первую букву имени и фамилию ученика:

И. Петров

П. Иванов

и т. д.

1. Переменная содержит сведения об учениках некоторой школы (см. задачу 1).

а) Собрать сведения об учениках девятых классов школы,

б) Выяснить, на сколько человек в восьмых классах больше, чем в девятых.

Багаж пассажира характеризуется количеством вещей и общим весом вещей. Сформировать переменную Bagaj, содержащую сведения о багаже нескольких пассажиров. Сведения о багаже каждого пассажира представляют собой запись с двумя полями: одно поле целого типа (количество вещей) и одно – действительное (вес в килограммах). Задание (3-7):

1. Найти багаж, средний вес одной вещи в котором отличается не более, чем на 0,3 кг от общего среднего веса одной вещи.
2. Найти число пассажиров, имеющих более двух вещей и число пассажиров, количество вещей которых превосходит среднее число вещей.
3. Выяснить, имеется ли пассажир, багаж которого состоит из одной вещи весом менее 30 кг.
4. Требуется удалить из данной переменной Bagaj сведения о багаже, общий вес вещей в котором меньше, чем 10 кг. Использовать вспомогательную переменную F.
5. Переписать сведения о багаже из переменной Bagaj в переменную Bag. В переменной Bag сведения о багаже каждого пассажира представляются массивом из двух целых чисел - числа вещей и общего веса вещей, выраженного в граммах. Составить также программу обратного преобразования: переписи сведений о багаже из переменной Bag в переменную Bagaj.
6. Сформирована переменная bibl, содержащая сведения о книгах. Сведения о каждой из книг – это фамилия автора, название и год издания.

а) Найти названия книг данного автора, изданных с 1960 года

б) Определить имеется ли книга с названием "Информатика". Если да, то сообщить фамилию автора и год издания. Если таких книг несколько, то сообщить сведения обо всех этих книгах.

1. Дана переменная Т, которая содержит номера телефонов сотрудников учреждения: Указывается фамилия сотрудника, его инициалы и номер телефона. Найти номер телефона сотрудника по его фамилии и инициалам.
2. Сформирована переменная типа запись, содержащая различные даты. Каждая дата - это число, месяц и год. Найти:

а) год с наименьшим номером.

б) все весенние даты.

в) самую позднюю дату.

1. Сформировать переменную Tovar, содержащую сведения об экспортируемых товарах: Указывается наименование товара, страна импортирующая товар, и объем поставляемой партии в штуках. Составить список стран, в которые экспортируется данный товар, и общий объем его экспорта.
2. Сформирована переменная Assortim, содержащая сведения об игрушках: указано название игрушки, ее стоимость в рублях, и возрастные границы. Получить следующие сведения:

а) название игрушек, цена которых не превышает 4 руб., и которые подходят детям 5 лет.

б) цену самого дорогого конструктора.

1. Сформирована переменная Assortim, содержащая сведения об игрушках: указано название игрушки, ее стоимость в рублях, и возрастные границы. Получить следующие сведения:

а) Название наиболее дорогих игрушек (цена которых отличается не более чем на 1 руб. от самой дорогой).

б) название игрушек, которые подходят как детям 4 лет, так и детям 10 лет.

**4. ТИП ДАННЫХ «ФАЙЛ»**

**🕮 Теоретические сведения**

У понятия файл есть две стороны. С одной стороны, файл – это область памяти на внешнем носителе, в котором хранится некоторая информация. Файл в таком понимании называют физическим файлом, т.е. существующим физически на некотором материальном носителе информации. С другой стороны, файл - это одна из структур данных, используемых в программировании. Файл в таком понимании называют логическим файлом, т.е. существующим в нашем логическом представлении при написании программы.

Структура физического файла представляет собой простую последовательность байт памяти носителя информации. Структура логического файла – это способ восприятия файла в программе.

Любой файл имеет следующие характеристики-требования:

* у него есть имя (набор из восьми, допустимых для имени файла, символов плюс расширение, указываемое после точки в имени файла состоящее из трех символов);
* он должен содержать данные одного типа (любой тип Паскаля, кроме типа Файл, то есть не существует типа «Файл файлов»);
* длина создаваемого файла никак не регламентируется при создании файла и ограничивается только емкостью носителя информации.

Работа с файлами в Паскале осуществляется следующим образом: сначала объявляется переменная файлового типа, с указанием свойств переменной (то есть типом содержимого), затем данная файловая переменная связывается («ассигнуется») с именованным дисковым пространством (то есть непосредственно с конкретным файлом, содержащим или, который будет содержать данные того же типа, что и связываемая переменная-файл) или логическим устройством)

Переменная файлового типа может быть объявлена одной из следующих строк:

*<имя> =* ***file of*** *<тип>;*

*<имя> =* ***text;***

*<имя> =* ***file;***

где <имя> – имя переменной-файла;

**file of** – зарезервированные слова (файл, из);

**text** – имя стандартного типа текстовых файлов;

<тип> – имя любого стандартного типа Паскаля, кроме типа файл.

Например файл, содержащий список учеников и их возраст:

**type**

pupil = record

surname : string;

name : string;

age : word

**end;**

**Var**

journal : **file of** pupil;

В зависимости от способа объявления можно выделить три вида файлов:

* типизированные файлы (задаются предложением **file of**);
* текстовые файлы (определяются типом **text**);
* нетипизированные файлы (определяются типом **file).**

Файловые переменные имеют специфическое назначение. Такие операции, как присвоение значения, сравнение и т.д. над переменными типа файл осуществлять нельзя.

***Текстовые файлы*** – это файлы, содержащие символы, разделенные на строки. Причем в конце каждой строки стоит признак конца строки. Текстовые файлы не имеют прямого доступа. При чтении и записи числа преобразуются автоматически. К ним применима процедура Append(<имя переменной текстового файла>). Она открывает текущий файл, с которым связана данная переменная, текущий указатель помещает в конец для добавления новой информации.

***Нетипизированные файлы*** предназначены для низкоуровневой работы с файлами. С их помощью можно обратиться к файлу любого типа и логической структуры. За одно обращение считывается/записывается число байт, приблизительно равное величине буфера ввода/вывода. В качестве буфера может выступать любая переменная. Для записи и чтения используются процедуры BlockRead, BlockWrite.

**Процедуры и функции обработки файловых переменных**

* **Assign** (<*имя файловой переменной*>,’<*путь и имя файла на диске*>’) – связь переменной файлового типа с конкретным внешним файлом.
* **Reset (f)** – процедура открытия существующего файла и подготовка к чтению файла, связанного с файловой переменной f. Указатель текущей позиции файла устанавливается в его начало.
* **Rewrite (f)** – процедура создания нового физического файла и подготовка к записи файла, связанного с файловой переменной f**.** Если такой файл существует, то он удаляется, и на этом месте создается новый пустой файл. Указатель текущей позиции файла устанавливается в его начало.
* **Readln (f) –** пропуск строки файла до начала следующей;
* **Writeln (f) –** запись признака конца строки и переход на следующую;
* **Read (f, x)** – процедура чтения компоненты файла. Данные выводятся из файла.
* **Write (f, x)** – процедура записи значения переменной в файл, который хранится на диске. Указатель перемещается на следую-щий элемент. Если указатель текущей позиции файла находится за последним элементом, т.е. в конце файла, то файл расши-ряется.
* **Eof** – признак конца файла – логическая функция для определения, достигнут ли конец файла.
* **Close (f)** – процедура закрытия файла (**!!! именно в этот момент происходит реальная запись в файл**).

**🖳 Примеры решений задач**

1. *Прочитать из текстового файла А все записанные в него целые числа (их можно ввести в него через Блокнот и сохранить файл), преобразовать их в вещественные и вывести в текстовый файл В по 4 числа в строку.*

**Program** File1;

**Var** F1,F2: text; // объявляем 2 переменные типа текстовых файлов

X: real;

s: string;

I:integer;

**Begin**

Assign (F1,'A.txt'); //связываем переменную F1 с файлом A.txt (находится там же, где и программа)

Reset(F1); // открываем его для чтения (т.е. он уже должен быть создан ранее)

Assign (F2,'B.txt'); //связываем F2 с файлом В.txt (будет находиться там же, где и программа)

Rewrite (F2); // открываем его для записи

writeln('содержимое файла А.txt:');

**Repeat** // цикл

**For** I:=1 **to** 4 **do** // цикл от 1 до 4, т.к. нужно в файле В формировать по 4 числа в строке

**If not** seekeof(F1) **then** // если еще не конец файла, связанного с F1

**Begin** // то делаем

Read (F1,x); //считываем из файла, связанного с F1, 1 компонент и помещаем его в

//переменную х

write(x:-5:0); // выводим на экран, что считали из файла в текущий момент,

//-5 означает, что в поле из 5 позиций с выравниванием по левому краю

Write (F2,x:-5:2) // выводим текущее считанное значение в файл, связанный с F2

**End**;

Writeln(F2); // делаем в файле переход на новую строку (чтобы следующее значение

//выводилось с новой строки)

**Until** seekeof (F1); // будет повторяться, пока файл не закончится

writeln;

Close (F1); // закрываем файл, связанный с F1

Close (F2); // закрываем файл, связанный с F2

Reset(F2); // открываем файл, связанный с F2, для чтения

writeln('содержимое файла B.txt:');

**Repeat** //цикл

Readln(F2,s); // считываем по целой строке из B и записываем в строковую переменную s

writeln(s);

**Until** seekeof (F2); // пока не конец файла, связанного с F2

Close (F2);

**End**.

Пример работы программы:

содержимое файла А.txt:

1 2 3 4 5

содержимое файла B.txt:

* 1. 2.00 3.00 4.00 5.00

1. *Работа с числовыми файлами. Записать в файл числа (кол-во вводит пользователь, сами числа задаются генератором слу-чайных чисел). Записать в другой файл числа, кратные 5 или 4.*

**program** file2;

**var** f1,f2: **file of** integer; //файловые переменные типа integer (в файле будут только целые числа)

i,n,x: integer;

**begin**

assign(f1,'a.dat'); // связываем файловую переменную f1 с файлом a.dat

rewrite(f1); // открываем его на запись или перезапись (если он уже существует)

assign(f2,'b.dat'); // связываем файловую переменную f2 с файлом b.dat

rewrite(f2); // открываем его на запись или перезапись (если он уже существует)

write('Введите кол-во чисел: ');

readln(n);

**for** i:=1 **to** n **do** // будем в цикле записывать данные в файл

**begin**

x:=random(30);

write(f1,x); // текущее значение х записывается в файл

**end**;

close(f1); // закрываем файл, связанный с f1 (это нужно, чтобы все записанные в него данные

// сохранились)

reset(f1); // и открываем его для чтения

writeln('Вот числа из первого файла:');

**for** i:=1 **to** n **do**

**begin**

read(f1,x); // считываем 1 значение из файла, связанного с f1, и записываем его в х

write(x:5); // выводим х на экран

**if** (x **mod** 5 =0) **or** (x **mod** 4 =0) **then** write(f2,x); // если считанный элемент подходит по

//условию, то записываем его в файл, связанный с f2

**end**;

close(f1); // закрываем файл, связанный с f1

close(f2); // закрываем файл, связанный с f2

reset(f2); // открываем файл, связанный с f2, для чтения

writeln;

writeln('Вот числа из второго файла:');

**while** (**not** eof(f2)) **do** // т.к. мы не знаем, сколько было записано элементов в файл,

//связанный с f2, то используем цикл с условием, который будет работать

// пока не будет считан признак конца файла (while (not eof(f2)) do)

**begin**

read(f2,x); // считываем 1 элемент из файла, связанного с f2, и записываем его в х

write(x:5); // выводим х на экран

**end**;

close(f2); // закрываем файл, связанный с f2

**end**.

Пример работы программы:

Введите кол-во чисел: 20

Вот числа из первого файла:

1 20 13 1 12 17 7 8 7 21 10 5 19 9 4 17 13 3 6 11

Вот числа из второго файла:

20 12 8 10 5 4

**🖮 Задачи для самостоятельного решения**

**1. Текстовые файлы.**

1. Дан файл, содержащий текст на русском языке. Составить список всех слов, встречающихся в этом тексте.
2. Дан файл, содержащий текст, набранный заглавными русскими буквами. Провести частотный анализ текста, т. е. указать (в процентах), сколько раз встречается та или иная буква.
3. Дан символьный файл f. В файле не менее двух компонент. Определить являются ли два первых символа файла цифрами. Если да, то установить, является ли число, образованное этими цифрами, чётным.
4. Дан символьный файл f. Записать в файл g компоненты файла f в обратном порядке.
5. Дан файл, содержащий текст, записанный строчными русскими буквами. Получить в другом файле тот же текст, записанный заглавными буквами.
6. Дан файл, содержащий произвольный текст. Выяснить, чего в нем больше: русских букв или цифр.
7. Дан файл, содержащий текст на русском языке. Выяснить, входит ли данное слово в указанный текст, и если да, то сколько раз.
8. Дан файл, содержащий текст на русском языке. Выбрать из него те символы, которые встречаются в нем только один раз, в том порядке, в котором они встречаются в тексте.
9. Даны файл, содержащий текст на русском языке, и некоторые буквы. Найти слово, содержащее наибольшее количество указанных букв.
10. Дан файл, содержащий текст на русском языке, и некоторая буква. Подсчитать, сколько слов начинается с указанной буквы.
11. Дан файл, содержащий текст, включающий русские и английские слова. Подсчитать, каких букв в тексте больше – русских или латинских.
12. Дан текстовый файл. Удалить из него все лишние пробелы, оставив между словами не более одного пробела. Результат поместить в новый файл.
13. Дан файл, содержащий текст на русском языке. Подсчитать количество слов, начинающихся и заканчивающихся на одну и ту же букву.

**2. Числовые файлы**

1. Дан файл f, компоненты которого являются действительными числами. Найти:

а) сумму компонент файла;

б) произведение компонент файла;

1. Дан файл f, компоненты которого являются действительными числами. Найти из значений компонент:

а) наименьшее из значений компонент с чётными номерами;

б) наибольшее из значений компонент с нечётными номерами;

1. Дан файл f, компоненты которого являются целыми числами. Найти:

а) количество чётных чисел среди компонент;

б) сумму нечётных чисел;

1. Дан файл f, компоненты которого являются целыми числами. Получить в файле g все компоненты файла f:

а) являющиеся чётными числами;

б) делящиеся на 3 и не делящиеся на 7;

1. Даны файлы f и g, заполненные случайными числами. Записать в файл h сначала компоненты файла f, затем компоненты файла g с сохранением порядка.
2. Заполнить файл f целыми числами, полученными с помощью генератора случайных чисел. Переписать в файл g те компоненты файла f, которые являются четными.
3. Заполнить файл последовательного доступа f целыми числами, полученными с помощью генератора случайных чисел. Получить в файле g все компоненты файла f, которые делятся на m и не делятся на n.
4. Заполнить файл f целыми числами, полученными с помощью генератора случайных чисел. Из файла f получить файл g, исключив повторные вхождения чисел. Вывести файл g на экран.
5. Записать в файл f N произвольных натуральных чисел. Переписать в другой файл те элементы, которые кратны К. Вывести полученный файл на экран.
6. Заполнить файл f N действительными случайными числами. Найти сумму минимального и максимального элементов этого файла.
7. Записать в файл f N случайных действительных чисел. Найти разность первого и последнего компонентов файла.
8. Заполнить файл f случайными целыми числами. Переписать в файл g те компоненты файла f, которые являются нечетными.
9. Заполнить файл f целыми числами, полученными с помощью генератора случайных чисел. Переписать их в файл g в обратном порядке.

**5. ЛИНЕЙНЫЕ СПИСКИ**

**🕮 Теоретические сведения**

Для работы с динамическими структурами данных используются указатели. Указатели представляют собой специальный тип данных. Они принимают значения, равные адресам размещения в оперативной памяти соответствующих динамических переменных.

***Списком*** называется структура данных, каждый элемент которой посредством указателя связывается со следующим элементом. На самый первый элемент (голову списка) имеется отдельный указатель.

Из определения следует, что каждый элемент списка содержит поле данных (оно может иметь сложную структуру) и поле ссылки на следующий элемент. После ссылки последнего элемента должно содержать пустой указатель (nil).

Число элементов связанного списка может расти или уменьшаться в зависимости от того, сколько данных мы хотим хранить в нем. Чтобы добавить новый элемент в список, необходимо:

1. Получить память для него;

2. Поместить туда информацию;

3. Добавить элемент в конец списка (или начало).

Элемент списка состоит из разнотипных частей (хранимая информация и указатель), и его естественно представить записью. Перед описанием самой записи описывают указатель на нее:

Type *{описание списка из целых чисел}*

PList = ^TList;

TList = record

Inf : Integer;

Next : PList;

end;

**🖳 Примеры решений задач**

1. **Создание списка**

Сформировать список, содержащий целые числа 3, 5, 1, 9.

*Определим запись типа TList с полями, содержащими характеристики данных – значения очередного элемента и адреса следующего за ним элемента*

PList = ^TList;

TList = record

Data : Integer;

Next : PList;

end;

*Чтобы список существовал, надо определить указатель на его начало. Опишем переменные.*

Var

Head, x : PList;

{*Создадим первый элемент*}

New(Head); {*выделяем место в памяти для переменной Head*}

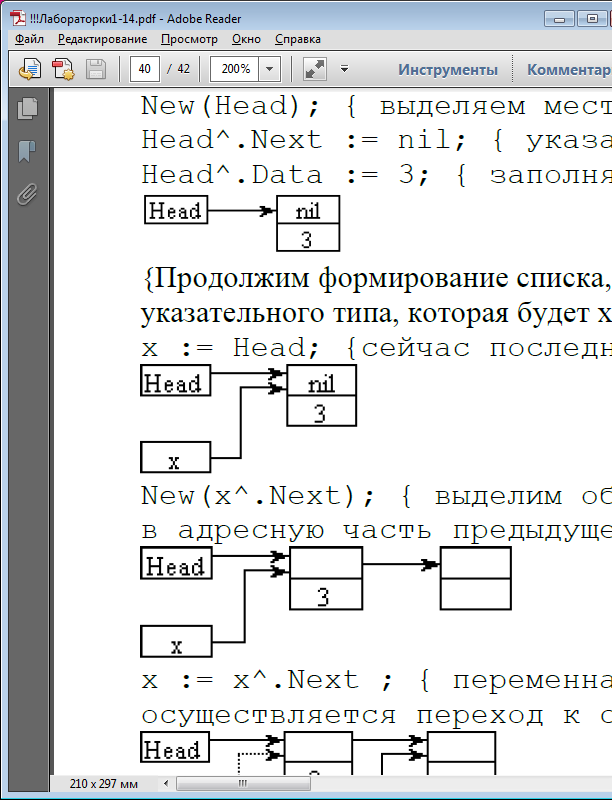
Head^.Next := nil; {*указатель на следующий элемент пуст (такого элемента нет) }*

Head^.Data := 3; *{заполняем информационное поле первого элемента}*

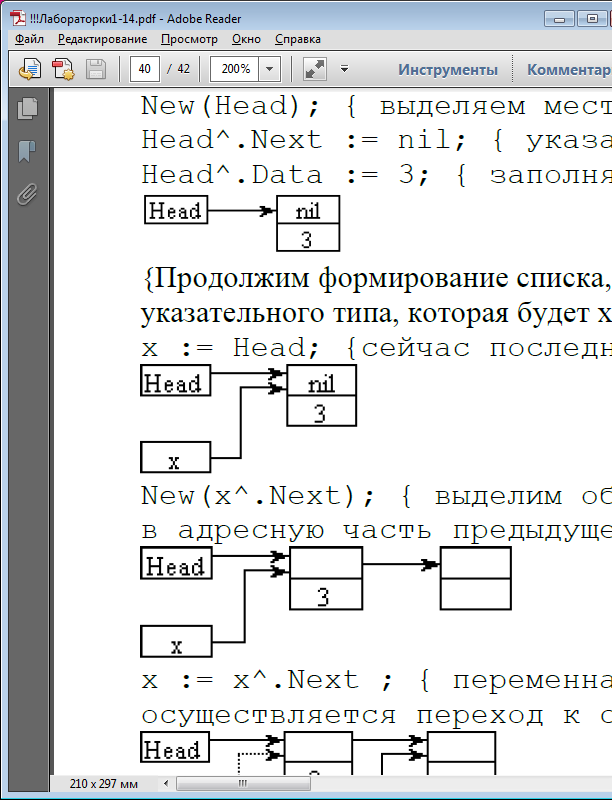


*{Продолжим формирование списка, для этого нужно добавить элемент в конец списка. Введем вспомогательную переменную указательного типа, которая будет хранить адрес последнего элемента списка}*

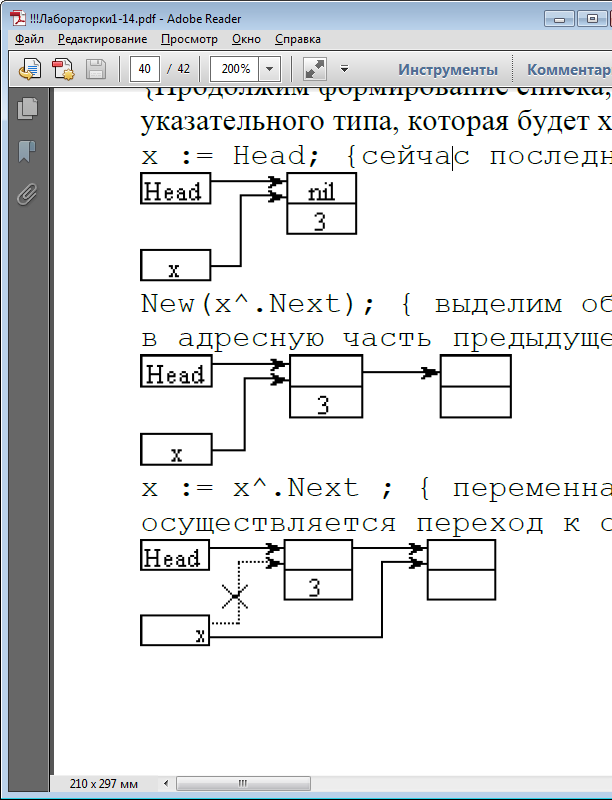
x := Head; {*сейчас последний элемент списка совпадает с его началом*}



New(x^.Next); *{выделим области памяти для следующего (2-го) элемента и поместим его адрес в адресную часть предыдущего (1-го) элемента}*



x := x^.Next ; {*переменная x принимает значение адреса выделенной области. Таким образом осуществляется переход к следующему (2-ому) элементу списка}*



Остальные числа заносятся аналогично:

New(x^.Next); {*выделим области памяти для следующего элемента}*

x := x^.Next ; *{переход к следующему (3-му) элементу списка}*

x^.Data := 1; *{ значение этого элемента}*

x^.Next := nil; *{следующего значения нет}*

New(x^.Next); *{выделим области памяти для следующего элемента}*

x := x^.Next; *{ переход к следующему (4-му) элементу списка}*

x^.Data := 9; *{ значение этого элемента}*

x^.Next := nil; *{следующего значения нет}*

***Замечание.*** Как видно из примера, отличным является только создание первого (Head) элемента – головы списка. Все остальные действия полностью аналогичны и их естественно выполнять в цикле.

Присоединение нового элемента к голове списка производится аналогично:

New(x); *{ввод значения элемента x^.Data := … }*

x^.Next := Head; Head := x; ………………..

В этом случае последний введенный элемент окажется в списке первым, а первый – последним.

**Просмотр списка**

Просмотр элементов списка осуществляется последовательно, начиная с его начала. Указатель List последовательно ссылается на первый, второй и т. д. элементы списка до тех пор, пока весь список не будет пройден. При этом с каждым элементом списка выполняется некоторая операция – например, печать элемента. Начальное значение List – адрес первого элемента списка (Head).

List := Head;

While List <> nil do

begin

WriteLn(List^.Data);

List := List^.Next; *{переход к следующему элементу; аналог для массива i:=i+1}*

end;

**Удаление элемента из списка**

При удалении элемента из списка необходимо различать три случая:

1. Удаление элемента из начала списка.

2. Удаление элемента из середины списка.

3. Удаление из конца списка.

Digit – значение удаляемого элемента.

**Удаление элемента из начала списка**

List := Head; {запомним адрес первого элемента списка}

Head := Head^.Next; {теперь Head указывает на второй элемент списка}

Dispose(List); {освободим память, занятую переменной List^ }

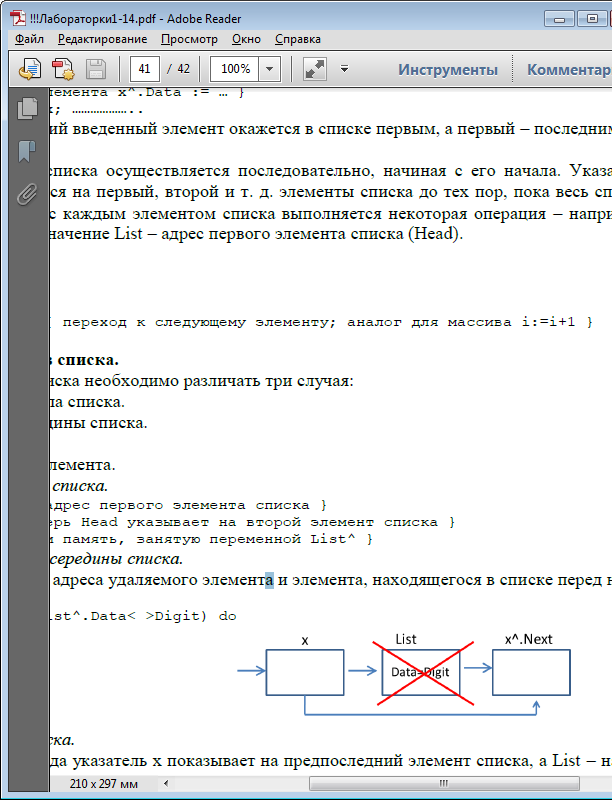
**Удаление элемента из середины списка**

Для этого нужно знать адреса удаляемого элемента и элемента, находящегося в списке перед ним.

List := Head;

While (List< >nil) and (List^.Data< >Digit) do

begin

x := List;

List := List^.Next;

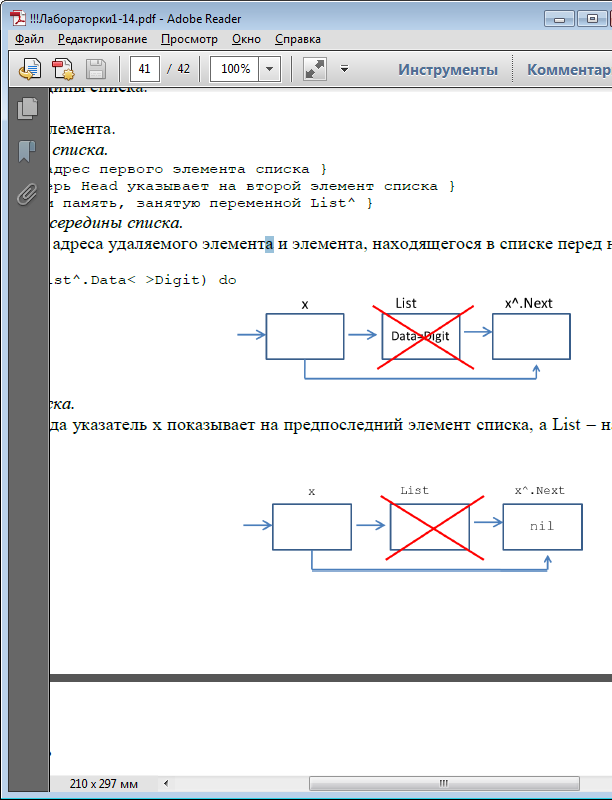
end;

x^.Next := List^.Next;

Dispose(List);

**Удаление из конца списка**

Оно производится, когда указатель х показывает на предпоследний элемент списка, а List – на последний.

List := Head; x := Head;

While List^.Next<>nil do

begin

x := List;

List := List^.Next;

end;

x^.Next := nil;

Dispose(List);

**Контрольные вопросы:**

1. Что такое указатель?

2. Что такое список?

3. Как реализуется описание списка?

4. На что должен указывать последний элемент списка?

5. Как создаются элементы списка?

6. Как удаляются элементы списка?

**🖮 Задачи для самостоятельного решения**

**Часть 1.**

1. Сформировать список строк и а) сохранить его в текстовом файле; б) сохранить его в обратном порядке в текстовом файле.
2. Сформировать список строк из текстового файла.
3. Написать функцию, которая вычисляет среднее арифметическое элементов непустого списка.
4. Написать процедуру присоединения списка L2 к списку L1.
5. Написать функцию, которая создает список L2, являющийся копией списка L1, начинающегося с данного узла (задает пользователь).
6. Написать функцию, которая подсчитывает количество вхождений элемента, который вводит пользователь, в списке.
7. Написать функцию, которая удаляет из списка все вхождения элемента, который вводит пользователь.
8. Сформировать список целых чисел и удалить из него все четные.
9. Сформировать список вещественных чисел и вычислить сумму.
10. Написать функцию, которая проверяет, упорядочены ли элементы списка по алфавиту.
11. Написать функцию, подсчитывающую количество слов в списке, которые начинаются с той же буквы, что и следующее слово.
12. Написать функцию, которая использует исходный список L и создает два новых списка L1 и L2. L1 содержит нечетные узлы, а L2 – четные.
13. Написать функцию, которая использует исходный список L и создает два новых списка L1 и L2. L1 содержит нечетные числа, а L2 – четные.

**Часть 2.**

1. Составить программу, которая вставляет в список *L* новый элемент *F* за каждым вхождением элемента *Е.*
2. Составить программу, которая вставляет в список *L* новый элемент F перед первым вхождением элемента *Е,* если *Е* входит в *L.*
3. Составить программу, которая удаляет из списка *L* все элементы *Е,* если таковые имеются.
4. Составить программу, которая удаляет из списка *L* за каждым вхождением элемента *Е* один эле-мент, если таковой имеется и он отличен от *Е.*
5. Составить программу, которая удаляет из списка *L* все отрицательные элементы.
6. Составить программу, которая проверяет, есть ли в списке *L* хотя бы два одинаковых элемента.
7. Составить программу, которая переносит в конец непустого списка *L* его первый элемент.
8. Составить программу, которая вставляет в список *L* за первым вхождением элемента *Е* все эле-менты списка *L,* если *Е* входит в *L.*
9. Составить программу, которая переворачивает список *L,* т.е. изменяет ссылки в этом списке так, чтобы его элементы оказались расположенными в обратном порядке.
10. Составить программу, которая в списке *L* из каждой группы подряд идущих одинаковых элементов оставляет только один.
11. Составить программу, которая формирует список *L,* включив в него по одному разу элементы, которые входят одновременно в оба списка *L1* и *L2.*
12. Составить программу, которая формирует список *L,* включив в него по одному разу элементы, которые входят в список *L1,* но не входят в список *L2.*
13. Составить программу, которая формирует 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.

**Учебно-методическое и информационное обеспечение дисциплины:**

***а) основная литература***

1. Алгоритмы и структуры данных /Л.Г. Гагарина, В.Д. Колдаев.- М.: ИНФРА-М, 2009.-372 с.
2. Вирт, Н. Алгоритмы и структуры данных. Новая версия для обертона / Н. Вирт.- М.: ДМК-Пресс, 2010.- 339 с.

***б) дополнительная литература***

1. Потопахин, В.В. Искусство алгоритмизации / В.В. Потопахин. - М.: ДМК Пресс, 2011.- 320с. - ISBN 978-5-94074-621-8

***в) программное обеспечение***

1. Пакет MS OFFICE
2. Среды программирования на языках Паскаль, С++

***г) базы данных, информационно-справочные и поисковые системы***

1. Каталог образовательных интернет – ресурсов [Электронный ресурс]: Режим доступа: <http://www.edu.ru/modules.php?name=Web_Links>.