СПРАВОЧНЫЙ МАТЕРИАЛ!

Команды редактора

Команды управления движением курсора

-> - (<- -) перемещение курсора на символ вправо (влево);
^ (∨) - перемещение курсора на строку вверх (вниз);
Home (End) - перемещение курсора в начало (конец) текущей строки;
Page Up (Page Down) - перемещение курсора на страницу вверх (вниз);
Примечание. Страница - это число строк текста, составляющих один экран (21 строка).
Ctrl + Home (Ctrl + End) - перемещение курсора в левый верхний угол (левый нижний угол);

Команды вставки и удаления текста

Insert - включение и выключение режима вставки;
Примечание. Если режим вставки включен, то на экране курсор имеет вид мигающей черты. В режиме вставки набираемый символ вводится в позицию, в которой стоит курсор, а все символы (начиная с символа, стоящего в позиции курсора ранее),расположенные правее, сдвигаются вправо. Если режим вставки выключен, то набираемый символ заменит тот символ, который находится в позиции курсора, таким образом можно старый текст заменить на новый.
Delete - удаление символа, стоящего в позиции курсора;
Backspace - удаление символа, стоящего перед курсором;
Ctrl + N - вставка пустой строки над строкой, где находится курсор;
Ctrl + Y - удаление строки, где находится курсор.

Режим помощи

Необходимо познакомить с режимом помощи - Help (F1). Показать, как входить в режим помощи, перемещаться по нему. Подробное знакомство с этим режимом учащиеся проводят самостоятельно.

Запуск программы на выполнение

1. Выполняем **проверку <F9>**
2. Производим компиляцию **<Alt>+<F9>**
3. **Запуск** программы **<Ctrl>+<F9>**
4. Просмотр содержимого окна консоли **<Alt>+<F5>**

Сохранение программы

Для того, чтобы сохранить программу, необходимо:
- выйти в главное меню и выбрать режим File;
- нажать <Enter> и из появившегося окна выбрать режим Save as..., после нажатия клавиши <Enter> появится окно, в котором наберите имя файла. Например, a:\prim1_1.pas; здесь a:\ - это название диска, на котором будем сохранять файл, prim1_1 - имя файла (оно может содержать не более 8 символов), pas - расширение, сообщающее о том, что файл содержит программу, написанную на языке Паскаль.

Примечание. список символов, которые нельзя употреблять в именах файлов:
* = + []\|;:,.<>/?. А также не следует использовать в именах файлов символ пробела и буквы русского алфавита.
После того, как имя файла набрано, нажмите клавишу <Enter>.
Примечание. Следует отметить, что для быстрого сохранения файла можно воспользоваться командами Save или Save all меню File.

Выход из системы программирования Турбо Паскаль

Для того, чтобы закончить работу, необходимо:
- выйти в главное меню и выбрать режим File;
- нажать <Enter> и из появившегося окна выбрать режим Quit <Alt>-<X>, после чего нажать либо <Enter>, либо комбинацию <Alt>-<X>.

Лабораторная работа №1 «Первая программа. Ввод, вывод данных»

Цель работы: Освоение простейшей структуры программы и получение навыков в организации ввода/вывода значений стандартных типов данных.

Часть I. Основная структура программы.

Каждая программа, которую вам предстоит создать при выполнении данной методической разработки, имеет определенную структуру. Определенные служебные слова, которые будут присутствовать в КАЖДОЙ созданной вами программе. Как только вы забудете, или напишите неправильно хотя бы одно из ОБЯЗАТЕЛЬНЫХ служебных слов – программа сразу же сообщит вам об этом!

В первую очередь, каждая программа должна начинаться с определенного заголовка. Для задания заголовка в Pascal используется служебное слово **Program,** затем пишется имя программы:

Program <имя программы>

За ним идет раздел описаний, в котором должны быть описаны все идентификаторы (константы, переменные, типы, процедуры, функции, метки), которые будут использованы в программе.

После раздела описаний следует раздел операторов, который начинается со служебного слова **Begin** и заканчивается служебным словом **End**. В этом разделе задаются действия над объектами программы, введенными в употребление в разделе описаний. Операторы в этом разделе отделяются друг от друга точкой с запятой. После последнего слова **End** ставится точка.

Рассмотрим на примере конкретной программы, запрашивающей имя и возраст.

```
Program lab1_1;

Var     a: Integer;
        b: String;

Begin

Writeln ('Введите имя ');

Readln (b);

Writeln ('Ваш возраст? ');

Readln (a);

Writeln (b,',вам', a,'лет?');

End.
```

Имя этой программы lab1_1 (заметим, что в имени программы не должно быть пробелов, оно должно начинаться с буквы, состоять только из латинских букв, цифр и некоторых символов, не допускается использование символов точки и запятой).

Раздел описания переменных – следующий раздел. Он начинается со служебного слова **Var**, после которого идет последовательность каких-либо переменных, разделенных точкой с запятой.[1]

Тип переменных указывается после перечисления всех переменных, после знака двоеточие. В нашем примере описаны две переменные: ***a*** имеет целый тип (integer), ***b*** имеет строковый тип (String).

Раздел операторов – следующий раздел, начинающийся со служебного слова **Begin**, после которого идут операторы языка. В конце раздела операторов стоит служебное слово **End**, после которого ставится точка.

[1] имена могут включать латинские буквы, цифры и знак подчеркивания; имя состоит из одного слова; имя всегда начинается с буквы; имена не могут совпадать с зарезервированными в языке *служебными словами.*

Сделаем первый вывод. Разрабатывая новую программу вы, в первую очередь, создаете основную структуру программы или, так называемый, «костяк» программы:

Program

Var

Описание переменных

Begin

инструкции

инструкции

End.

И только после этого, начинаете создавать программу, добавляя в нее определенные инструкции (операторы и (или) команды).

Задание 1.

Запишите в рабочем окне основную структуру программы.

Часть II. Команды ввода, вывода данных.

Вернемся опять к программе lab1_1. Какое же действие выполняет эта программа? Рассмотрим подробно содержимое ***раздела операторов***.

```
Program lab1_1;
Var     a: Integer;
        b: String;

Begin

Writeln ('Введите имя ');

Readln (b);

Writeln ('Ваш возраст? ');

Readln (a);

Writeln (b,',вам',a,'лет?');

End.
```

Первый встречающийся оператор - это **Write***ln*('текст'); - записать (вывести) на экран текст, заключенный между апострофами (у нас – ***Введите имя***), *ln* добавляется в конце этого оператора для того, чтобы курсор автоматически переходил на следующую строку.

Следующий оператор - это **Read***ln*(b); - читать данные с клавиатуры. В данном случае необходимо ввести какой-либо текст, для этого мы используем переменную b – строкового типа. Тогда переменной b будет присваиваться значение, которое вы введете с клавиатуры. Например, вы ввели свое имя ***Василий,***

Следующий оператор опять **Write***ln*('текст'); - записать (вывести) на экран текст «Ваш возраст».

Следующий оператор - это **Read***ln*(a); - читать данные с клавиатуры. В данном случае необходимо ввести число, для этого мы используем переменную a – целого типа. Переменной *a* присваивается значение, равное числу которое вы ввели. Например, вы ввели свой возраст 12, тогда *a* = 12.

Следующий оператор - это снова оператор *writeln*('текст', z) - он выведет на экран сначала значение переменной b – «Василий», текст «,вам», а за ним значение переменной *a – 12* , и текст «лет?». После запуска программы вы получите фразу-вопрос: ***Василий. вам 12 лет?***

Задание 2.

К созданной вами ранее основной структуре программы в ***раздел переменных*** и в ***раздел операторов*** допишите соответственно содержимое программы lab1_1.

Часть III. Запуск программы

Для того, чтобы запустить программу, выходим в главное меню (нажатием F10) - первое окно, выбираем режим RUN и дважды нажимаем <Enter>. На экране появляется сообщение:

Введите числа

Курсор мигает в следующей строке, вводим два целых числа через пробел и нажимаем <Enter>, после этого появляется сообщение:

результат –

Нажмите <Enter>.

Протестируйте и сохраните программу, пользуясь справочным материалом. Представьте выполненное задание преподавателю.

Первая контрольная точка.

Часть IV. Простейшие линейные программы.

Разберем подробно несколько задач, прежде чем вы приступите к самостоятельному написанию программ. Для этого еще раз вернемся к некоторым *азам*, которые вы должны очень четко себе представлять, прежде чем создавать первую программу.

Оператор присваивания.

Так или иначе, мы уже познакомились с этим оператором, когда рассматривали задачу lab1_1, когда присваивали переменным a и b различные выражения. В общем виде этот оператор можно описать так:

Идентификатор: = выражение.

Здесь идентификатор – имя переменной, переменная хранится в ячейке памяти с именем – идентификатор. Тип ячейки памяти определен в разделе описаний. Выражение может быть арифметическим, логическим или каким-либо другим, в котором уже известны (определены) все входящие в его состав идентификаторы.

Тип значения выражения и тип идентификатора должны совпадать, иначе error - ошибка.

Задание 3.

Написать программу, реализующую сложение двух целых чисел.

Прежде всего, разделим процесс решения задачи на три блока: собственно программа на языке Pascal, пояснения к этой программе и наконец, поэтапное выполнение данной задачи самой машиной.

Начнем сразу с раздела Var. То, что программа должна иметь свой заголовок, должно быть вам уже понятно без дополнительных объяснений.

На языке Pascal	Пояснение программы	Работа машины
Program lab1_2; **Var** a, b, c**: integer;** **Begin**	**Var** - начало раздела описания переменных, **a, b, c** - идентификаторы. **Integer** означает целый, т.е. в разделе Var идентификаторы (имена) переменных определены как целые.	Встретив такое описание, ЭВМ “выделяет” три ячейки в своей памяти с именами **a**, **b**, **c**, причем такие, чтобы туда могли быть “помещены” <u>только</u> целые числа: a b c [ ] [ ] [ ]
a:=2; **b:=3;**	Переменным a и b присваиваются соответственно значения 2 и 3.	В выделенные ячейки машина записывает соответствующие значения a b c [2] [3] [ ]
c: = a + b;	Переменной ***c*** присваивается значение суммы двух других ячеек – ***a*** и ***b***.	Затем машина встречает оператор присваивания и извлекает из ячеек **a и b** их содержимое. Производит операцию сложения, и результат заносит в ячейку **c**: a b c [2] [3] [5]
Writeln (c); **End.**	На экран выводится содержимое ячейки ***c.***	Машина извлекает из памяти содержимое ячейки c и вывод на экран.

Протестируйте и сохраните программу.

. Вторая контрольная точка.

Комментарии

Комментарий – это произвольная последовательность любых символов, поясняющая текст программы. Комментарий разрешается вставлять в любое место программы, где по смыслу может стоять пробел. Как правило, комментарии пишутся в начале программы как поясняющий текст о предназначении программы, и в тексте программы. В качестве ограничителей комментария используются фигурные скобки « {» и «} »:

<u>Только используя комментарии при написании программ, вы сможете считать себя грамотным пользователем Turbo Pascal!</u>

Задание 4.

Написать программу вычисления **a = c*b**.

Прежде чем создавать программу вспомним, что любая программа:

- начинается с заголовка **Program** – имя программы;
- затем идет раздел описания переменных, где указываются все переменные, которые будут использоваться в программе. Здесь же указывают типы переменных.
- Далее следует раздел операторов, который начинается словом **Begin** и заканчивается словом **END** (End с точкой, точка – конец программы).
- Каждое описание и оператор заканчиваются символом **';'**

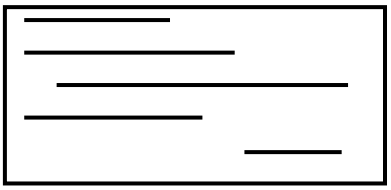
```
Program lab1_3;      {заголовок программы}
Var  a,b,c: integer;   {раздел описания переменных}
BEGIN                  {начало раздела операторов}
        c:=5;
        b:=4;          {занесение в ячейки c и b начальных значений}
        a:=c*b;        {вычисление значения переменной a}
        writeln(a);    {вынесение на экран значения a}
END.                   {конец программы}.
```

Комментарии

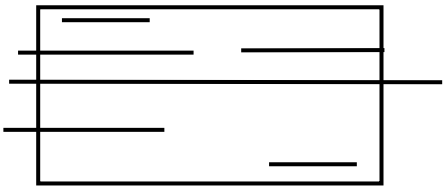
Напоминаем, что тексты, заключённые в скобки { }, являются комментариями, не являются частью программы и могут быть опущены, таким образом, программа для ЭВМ имеет следующий вид:

```
Program lab1_3;
        Var  a,b,c: integer;
BEGIN
        c:=5;
        b:=4;
        a:=c*b;
        writeln(a);
END.
```

Программа записывается в виде строк. Максимальная длина строки 255 символов, но не стоит делать строки длиннее строки экрана – 80 символов, иначе строки будут "прятаться" за пределами окна экрана, хотя строки и можно просмотреть, передвигая "окно":



"Хорошее" расположение строк



"Плохое" расположение строк

рис. 1

Каждую строку можно начинать не с первой позиции, чтобы программа была более читаемой.

Протестируйте и сохраните программу с комментариями к ней.

Третья контрольная точка.

Задания для самостоятельной работы

I. Ответьте на вопросы:

1. Какие окна появляются на экране после загрузки Pascal? Каким образом осуществляется переход между окнами?
2. Укажите основные команды управлением движения курсора.
3. Укажите основные команды вставки и удаления текста
4. Укажите алгоритм сохранения программы и выход из программы Pascal
5. В чем состоит основная структура программы Pascal, поясните каждый раздел
6. Назначение оператора Writeln? В чем разница между Writeln и Write?
7. Назначение оператора Readln? В чем разница между Readln и Read?
8. Укажите алгоритм запуска программы в Pascal на выполнение.
9. Укажите назначение оператора присваивания в Pascal. Приведите пример.
10. Каково основное назначение комментарий в программе Pascal?

II. Задания:

1. Изменить программу для нахождения суммы четырех чисел.
2. Вычислить значение выражения:
   1) $y = 15x^2 + 8x – 9$ (переменная x должна вводиться с клавиатуры);
   2) a = (b + c)*d – k (переменные b,c,d,k должны вводиться с клавиатуры).
3. Выведите на экран свою фамилию, имя и отчество, в следующей строке - дату рождения (все текстовые данные должны вводиться с клавиатуры).
4. Найти значение выражения: **(a + (d - 12)*3) * (c - 5*k)**, где
   1) значения переменных *a, d, c* и *k* вводятся с клавиатуры.
   2) даны в тексте программы
5. Найти периметр:
   1) прямоугольника, ширину и длину вводить с клавиатуры;
   2) треугольника, длины всех сторон вводить с клавиатуры;
   3) произвольного четырехугольника.
6. Вычислить рациональным способом, т.е. за минимальное количество операций [2]:
   1) $y = x^5$ ($y = (x^2)^2 * x$, то есть за 3 операции);
   2) $y = x^6$ ($y = (x^3)^2 = (x^2 * x)^2$, то есть за 3 операции);
   3) $y = x^8$ ($y = ((x^2)^2)^2$, тоже за 3 операции).
7. Написать программу вычисления стоимости покупки, состоящей из нескольких тетрадей и карандашей. Ниже представлен рекомендуемый вид экрана во время работы программы (данные, введенные пользователем, выделены полужирным шрифтом)

ВЫЧИСЛЕНИЕ СТОИМОСТИ ПОКУПКИ.
ВВЕДИТЕ ИСХОДНЫЕ ДАННЫЕ:
ЦЕНА ТЕТРАДИ (РУБ.) – > **2.75**
КОЛИЧЕСТВО ТЕТРАДЕЙ – > **5**
ЦЕНА КАРАНДАША (РУБ.) – > **0.85**
КОЛИЧЕСТВО КАРАНДАШЕЙ – > **2**
СТОИМОСТЬ ПОКУПКИ: **15.45** РУБ.

[2] Использовать для выполнения функцию подсчета квадратного корня x^2 – функция SQR(x)

Лабораторная работа №2 «Условный оператор»

Цель работы: Ознакомиться с разветвляющимися вычислительными процессами на примере условного оператора.

До сих пор мы рассматривали линейные программы, алгоритм которых можно было представить в виде блок-схемы на рис. 2, т.е. четкая последовательность действий, следующая *друг за другом без каких либо отклонений!*

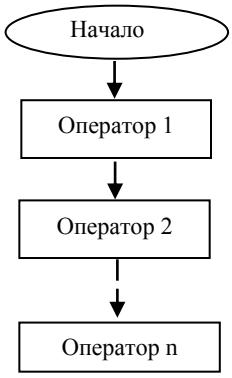


Рис. 2

Рассмотрим ситуацию:

Задание 1. Требуется разделить одно целое число на другое.

Вроде бы все просто, задаем любые целые числа и выполняем операцию деления:

```
Program lab2_1_a;
Var   b,c: integer;
a:  real;    {результат – число a – может не быть числом целым!}
BEGIN
      Write ('Введите число c');
      Read(c);
      Write ('Введите число b');
      Read (b);
      a:=c/b;
      writeln(a);
END.
```

Произведите запуск программы. Введите по запросу любые два целых числа. Программа выполняется, цель вроде бы достигнута.

Но! А если пользователь ввел вместо числа b – ноль? Что тогда? В этом случае, программа выдаст сообщение об ошибке:

Division by zero (Деление на ноль)

Что значит, предшествующая операция пытается выполнить деление на ноль. Как известно из школьного курса математики на ноль делить нельзя!

Произведите запуск программы, введите вместо второго числа – ноль. Просмотрите результат.

Итак, программа несовершенна. Как же быть пользователю?

Каким образом сказать машине, что если пользователь введет число b – ноль, то деление производить не надо?

Вот мы с вами плавно и подошли к новому понятию в программировании – ***«ветвление»***

Вычислительный процесс называется ***разветвляющимся***, если в зависимости от выполнения определенных условий он реализуется по одному из нескольких, заранее предусмотренных (возможных) направлений. Каждое отдельное направление называется ***ветвью вычислений*** (см. рис.3).

Напишем блок-схему для задачи Delenie и поясним ее:

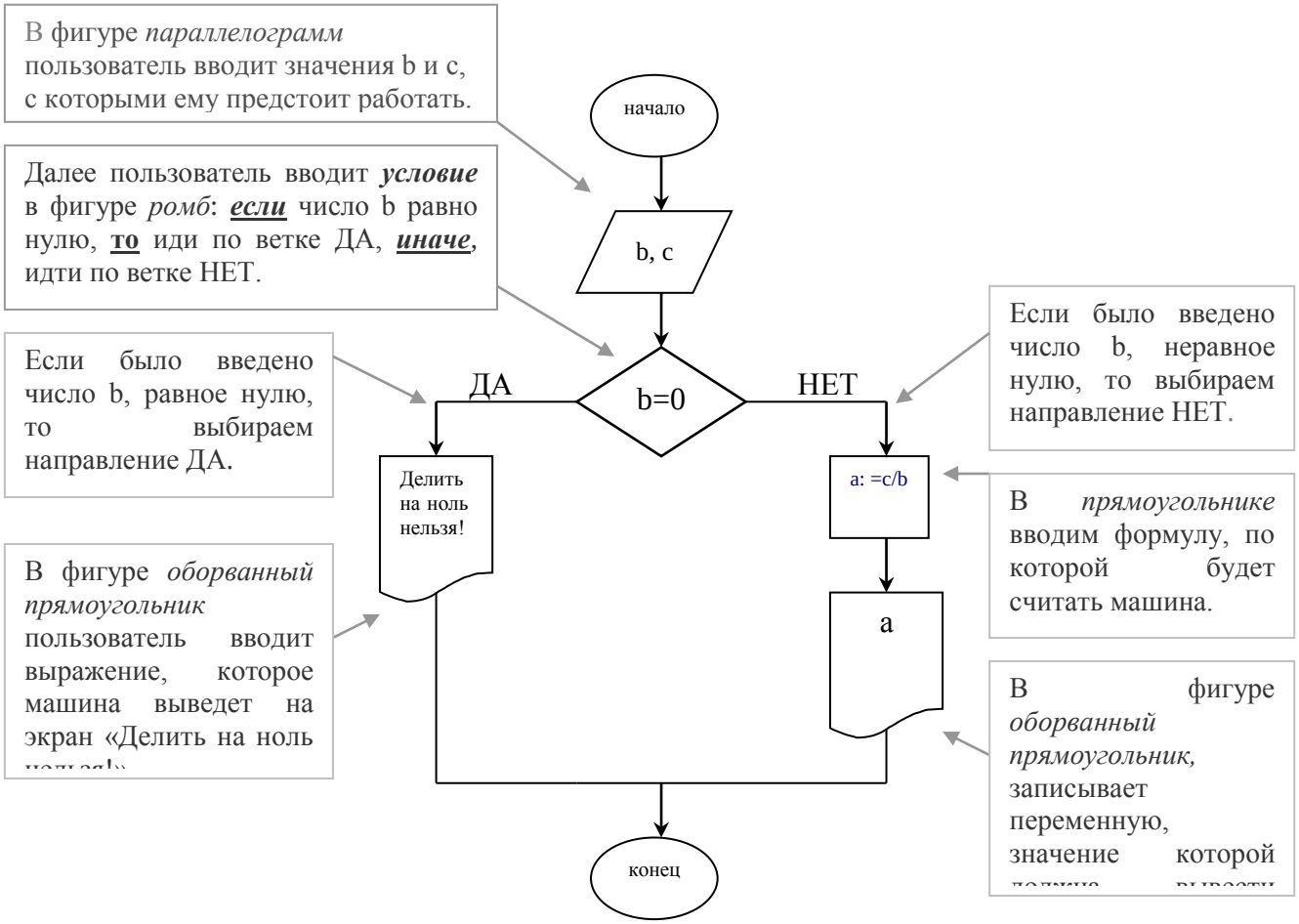


Рис.3

Итак, при составлении программы должны быть учтены все возможные *ветви вычислений.*

Как в предыдущей задаче **Delenie,** если бы мы не задумались о всевозможных случаях выполнения программы (и при вводе нуля!), то имели бы неявную ошибку в программе.

Для программной реализации таких вычислений в языке Паскаль имеются специальные операторы, которые дают возможность перейти из одного места программы в другое.

Если такой переход осуществляется только при выполнении какого-либо условия, то он называется ***условным***, а соответствующий ему оператор – ***оператором условного перехода.***

Оператор условного перехода

Разветвляющийся вычислительный процесс, содержащий две ветви, схематично может быть изображен с помощью структуры выбора (структуры разветвления), которая содержит три элемента: логическое условие, ветвь ДА и ветвь НЕТ.

Общий вид структуры представлен на рис.4.

После вычислений, общих для обеих ветвей, проверяется некоторое условие. Если оно выполняется, то осуществляется переход к первой ветви – ветви ДА, в противном случае – ко второй ветви – ветви НЕТ.

После выполнения вычислений в любой из ветвей осуществляется переход к общему участку 2. – Все это мы уже рассмотрели на примере задачи **Delenie**.

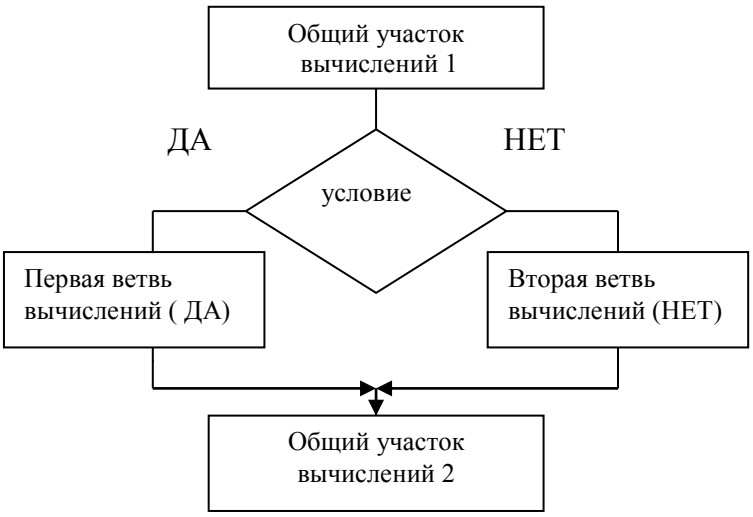


Рис.4

На языке Pascal условное **ветвление описывает оператор If**, который имеет следующий вид:

```
If Условие
   Then
        Begin
        {Эти инструкции выполняются, если Условие истинно}
        end;
   Else
        Begin
        { Эти инструкции выполняются, если Условие ложно}
        end;
```

Где If (если), Then (то), Else (иначе) – ***служебные*** слова.

Напишем программу для задачи lab2_1_a, применив оператор условного перехода If

```
Program lab2_1_b;
        Var  b,c: integer;
             a:  real;
BEGIN
        Write ('Введите число c');
        Read(c);
        Write ('Введите число b');
        Read (b);
        If b=0  Then Writeln('Делить на ноль нельзя!') {если If число b – ноль, то Then
                                                        выводим «Делить на ноль нельзя!»}
                Else  a:=c/b;          {иначе Else, производим операцию деления}
                      writeln(a);      {и выводим результат a }

END.
```

Задание 2.

Напишите программы lab2_1_a и lab2_1_b, применив оператор условного перехода. Впишите также комментарии в программы.

Протестируйте и сохраните оба варианта программы lab2_1_a и lab2_1_b, указав в комментариях разницу между ними.

Первая контрольная точка.

Полный и неполный условный оператор

Рассмотрим еще несколько особенностей оператора условного перехода.

Перед служебным словом **Else** разделитель **точка с запятой** НЕ ставится! Т.к. это сплошная неделимая конструкция IF… Then… Else…. (Если….то…иначе).

Ветвь **Else** также может и вовсе отсутствовать, если в случае невыполнения условия ничего делать не надо. Тогда конструкция оператора If будет иметь следующий вид:

If *Условие*
Then
Begin
{Эти инструкции выполняются, если Условие истинно}
end;

Рассмотрим такой пример.

Задание 3.

Написать программу, которая будет заменять переменную противоположным значением, если она изначально была меньше нуля.

Будем использовать переменную x. Если значение переменной *x* меньше *0*, то поменять его на противоположное. В программе такой условный оператор выглядит следующим образом:

```
Program lab2_2;
        Var  x: integer;
BEGIN
        Writeln ('Введите число x');
        Readln(x);
        If x<0 Then x:= -x; {как видим оператор Else отсутствует!}
        Writeln (x);
END.
```

В случае присутствия в условном операторе второй ветви (else), оператор называется полным. А в случае отсутсвия - неполным.

Задание 4.

Написать программу, проверяющую, принадлежит ли число, введенное с клавиатуры, интервалу (0,5).

Обозначим: *x* - число, вводимое с клавиатуры пользователем (это переменная целого типа). Принадлежность числа *x* интервалу (0,5) определяется следующей системой неравенств: $\begin{cases} x > 0 \\ x < 5 \end{cases}$

Число *X* принадлежит заданному интервалу лишь в том случае, если одновременно выполняются оба условия– число x больше 0 и одновременно меньше 5.

На языке Pascal данное условие позволит осуществить оператор ***AND***:

<Выражение 1> ***AND*** < Выражение 2>

```
Program lab2_3;
Var x : Integer;
Begin
Writeln('Введите число x');
Readln(x);
If (x>0) And (x<5) Then Writeln(x,' принадлежит интервалу(0,5)')
                    Else Writeln(x,' не принадлежит интервалу');
End.
```

Протестируйте и сохраните обе программы.

Вторая контрольная точка.

Задания для самостоятельной работы

I. Ответьте на вопросы:

1. Какой вычислительный процесс называется разветвляющимся?
2. Какой процесс перехода в программе с одного места на другое называется условным?
3. Укажите общий вид структуры разветвления. В чем суть этой структуры?
4. Какой оператор в языке Pascal описывает условное ветвление? Укажите полный синтаксис и поясните его.
5. Требуется ли ставить точку с запятой перед служебным словом Else?
6. В чем разница между полным и неполным условным оператором? Укажите конструкцию неполного оператора.
7. Для чего используется оператор AND?

II. Выберите подходящий по вашему усмотрению уровень сложности заданий и выполните его на компьютере.

Уровень I. (на оценку «3»)

1. Написать программу, вычисляющую значение функции $y = \frac{2a - x}{x}$, где a и x вводятся с клавиатуры.
2. Написать программу, которая будет заменять переменную нулем, если она изначально была меньше какой-либо константы (выберите произвольно).

Уровень II. (на оценку «4»)

1. Написать программу, вычисляющую значение функции $y = \frac{2}{(a - x)^2}$, где a и x вводятся с клавиатуры.
2. Написать программу, проверяющую, принадлежит ли число, введенное с клавиатуры интервалу (4, 8). И если принадлежит, заменить его противоположным числом.

Уровень III. (на оценку «5»)

1. Написать программу, вычисляющую значение функции $y = \frac{2}{(a - x)^3}$, где a и x вводятся с клавиатуры.
2. Вывести на экран большее из двух данных чисел
3. Сравнить возраст брата и сестры

Лабораторная работа №3 «Оператор выбора»

Цель работы: Ознакомиться с разветвляющимися вычислительными процессами на примере оператора выбора.

Мы рассмотрели ветвление двух направлений: по принципу условие верно – идти по одной ветви, условие ложно – по другой (см. рис 5а). В этом случае применяется оператор if.

НО! Как быть, если вариантов ветвления несколько?

На рисунке 5-б приведена схема, когда «ветвей» (или направлений) может быть несколько: если k=l – идти по одной ветви, если k=2 – по другой, k=3 – по третьей и т.д. в этом случае удобнее применять оператор **Case.**

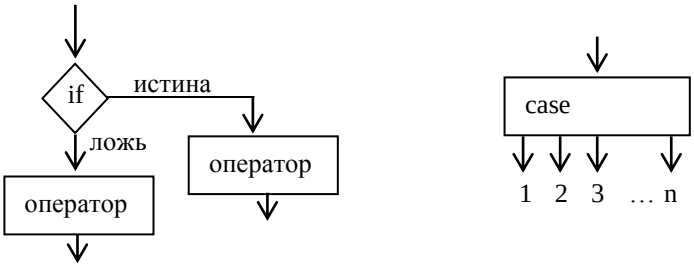


Рис. 5

Проиллюстрируем сказанное на примере программы.

Задание 1.

Напишите программу, выводящую названия дней недели согласно определенному номеру.

Итак, требуется согласно номерам дней недели отображать на экране их названия. Многие начинающие программисты для решения подобной задачи предпочтут воспользоваться последовательностью операторов IF:

```
Program  lab3_1_a;
Var x : Integer;
Begin
Writeln ('Введите x');
Readln(x);
        if x=1  then write (‘понедельник’);
        if x=2  then write (‘вторник’);
        if x=3  then write (‘среда’);
        if x=4  then write (‘четверг’);
        if x=5  then write (‘пятница’);
        if x=6  then write (‘суббота’);
        if x=7  then write (‘воскресенье’);
End.
```

Произведите запуск программы, введите любое из чисел (от одного до семи). Просмотрите результат.

Цель достигнута? Да, при вводе определенного числа, программа выдает определенный день недели.

Тем не менее, программа имеет более оптимальное решение.

Такой подход затрудняет понимание исходного текста программы, поскольку сразу очевидно, что здесь будет выполнен один и только один из семи операторов *IF*. Кроме

того, такая конструкция малоэффективна, поскольку при этом условие придется проверять во всех семи операторах *IF* и это неизбежно увеличит время выполнения программы.

Выйти из положения позволяет ***оператор выбора CASE***. Оператор выбора позволяет выбрать одно из нескольких возможных продолжений программы.

Оператор* CASE *имеет вид

CASE *Переменная-селектор* **of**
*Константа 1***:**
Begin
инструкции
end;
*Константа 2***:**
Begin
инструкции
end;
…….
*Константа n***:**
Begin
инструкции
end;
else
Begin
инструкции
end;
end;

Где ***CASE*** (выбор), ***OF***(из), ***ELSE (***иначе***), END***(конец) – служебные слова;

Выполнение оператора CASE начинается с вычисления переменной-селектора.

Инструкции, помещенные между Begin и End, выполняются в том случае, если значение выражения, стоящего после слова CASE, *совпадает* с константой из соответствующего списка. Если это не так, то выполняются инструкции, следующие после Else, расположенные между Begin и End.

Если Else отсутствует, выполняется оператор программы, следующий за CASE

В качестве примера, рассмотрим два вида селектора:

Селектор целочисленного типа	Селектор интервального типа
case i of 1: writeln(‘введенное число=1’); 2: writeln(‘введенное число=2’); 3: writeln(‘введенное число=3’); *end*;	case i of 1..10: writeln(‘введенное число от 1 до 10’); 11..20: writeln(‘введенное число от 11 до 20’); 21..30: writeln(‘введенное число от 21 до 30’); *end*;

Начнем с целочисленного типа. Селектор интервального типа рассмотрим позднее.

Вернемся к предыдущему примеру вложенного оператора *IF* (отображение названий дней недели). Попробуем реализовать эту задачу с помощью оператора *CASE*.

В качестве типа селектора, как уже было сказано, *будем использовать целочисленный тип.*

Вот как это должно выглядеть:

задача	пояснения
Program lab3_1_b; Var x : Integer;	Начало программы оставим то же. Задействуем в качестве переменной-селектора переменную *X*.
Begin Writeln('Введите x'); Readln(x); **case** x **of**	Если («CASE») при запуске программы, ***переменной*** *X* вы присвоили значение, ***совпадающее*** с одной из констант выбора… (констант выбора в данном случае от 1 до 7)
1:*write* ('понедельник'); 2:*write* ('вторник'); 3:*write* ('среда'); 4:*write* ('четверг'); 5:*write* ('пятница'); 6:*write* ('суббота'); 7:*write* ('воскресенье');	…. то пиши 'понедельник', если вы введете 1; …. то пиши 'вторник', если вы введете 2; …. то пиши 'среда', если вы введете 3; …. то пиши 'четверг', если вы введете 4; …. то пиши 'пятница', если вы введете 5; …. то пиши 'суббота', если вы введете 6; …. то пиши 'воскресенье', если введете 7;
end; **End.**	обратите внимание, один *End* относится к оператору выбора CASE, а другой заканчивает программу!

Выражение, играющее роль переменной-селектора, должно принадлежать порядковому типу данных (т.е. типу, имеющему конечное число значений). К порядковым относятся, например, типы данных *Integer, Boolean* и *Char*. Однако тип *Real* порядковым не является.

Таким образом, при использовании в операторе выбора CASE какой-либо ПЕРЕМЕННОЙ-селектора, важно помнить, что, она НЕ может быть ВЕЩЕСТВЕННОГО ТИПА!

Протестируйте и сохраните оба варианта программы lab3_1_a lab3_1_b, указав в комментариях разницу между ними.

Первая контрольная точка.

Задание 2. Написать программу, которая запрашивает у пользователя номер месяца и выводит соответствующее ему название времени года. В случае, если пользователь укажет недопустимое число, программа должна вывести сообщение «Ошибка!»

В зависимости от вводимого пользователем значения переменной (нуля, единицы, и т.д. до 12 – число месяцев году) программа выдает сообщение, к какому времени года принадлежит данный месяц. Схематическое представление оператора *CASE* – структура отбора, представлена для данного примера на рис.6. Обозначим используемую нами переменную как mesaiz

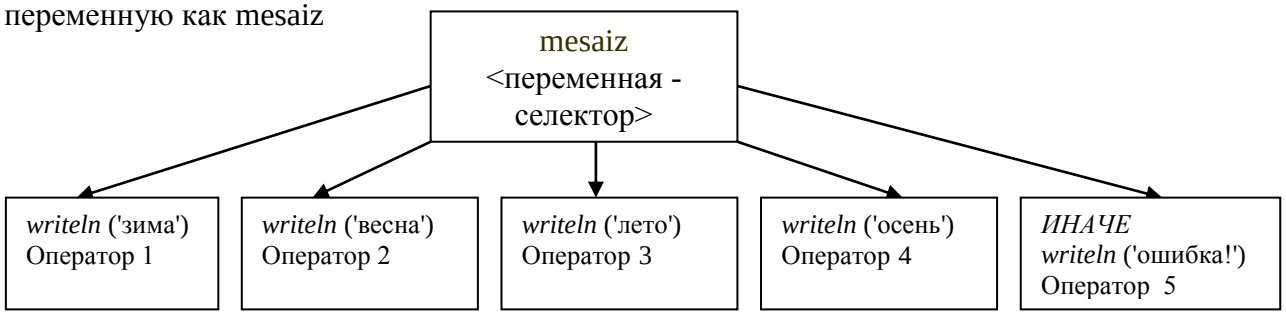


Рис.6

Решение задачи схоже с задачей lab3_1_b, также требуется в зависимости от введенного числа вывести определенное значение.

```
CASE mesaiz OF
    1:  writeln ('зима');
    2:  writeln ('зима');
    12:  writeln ('зима');
    3:  writeln ('весна');
    4:  writeln ('весна');
    5:  writeln ('весна');
    6:  writeln ('лето');
    7:  writeln ('лето');
    8:  writeln ('лето');
    9:  writeln ('осень');
    10:  writeln ('осень');
    11:  writeln ('осень');
End;
```

Вы уже, наверное, представили, насколько этот подход однообразный и утомительный. В данной задаче наиболее удобно применить в качестве типа селектора – *интервальный тип*.

Ведь как мы знаем, к летним месяцам относятся месяцы под номерами 6, 7 и 8, к осенним – месяца под номерами 9,10, 11, а к весенним месяцам – 3,4,5. А к зимним месяцам мы относим месяцы под номерами 1, 2 и 12.

Воспользуемся этим моментом, чтобы не писать по три раза *writeln* ('весна') или *writeln* ('лето') и т.д. ***Сгруппируем*** номера (константы выбора), под которыми программа должна выдавать нам одинаковые значения.

В программе это будет выглядеть следующим образом:

```
Program lab3_2;
Var mesaiz : Integer;
Begin
Writeln ('Введите номер месяца – число от 1 до 12');
Readln (mesaiz);
CASE mesaiz OF

    1, 2, 12:  writeln ('зима');  {Константы выбора могут быть указаны через запятую!}

    3..5:  writeln ('весна');
    6..8:  writeln ('лето');
    9..11:  writeln ('осень');
                                  {А если интервал не нарушается (месяцы 3, 4, 5 подряд
                                  являются летними), то пишутся только начальное и конечное
                                  значения, а между ними ставится знак «..»}

Else writeln ('ошибка! Число должно быть от 1 до 12!');

End;

END.
```

Протестируйте и сохраните программу.

Вторая контрольная точка.

Задание 3.

Написать программу, которая после введенного с клавиатуры числа (в диапазоне от 1 до 99), обозначающего денежную единицу, дописывает слово «рубль» в правильной форме. Например, 12 рублей, 21 рубль и т.д.

Для решения данной задачи нам понадобится функция mod (взятие остатка от деления 2 целых значений)

Остаток := *делимое* **mod** *делитель*;

Операция mod определена только для целочисленных операндов.
К примеру, требуется определить остаток n от деления 2006 на 100:

```
2006 |100
200   2
  6
```

– это и есть остаток от деления 2006 на 100: 2006/100=200,6. на языке паскаль это будет выглядеть следующим образом: n:=2006 mod 100;
Здесь переменная `n` получит значение `6`.

Вернемся к задаче.

В первую очередь, решим вопрос с окончаниями, в каком случае рубл***ей***, в каком – рубл***ь***, а в каком – рубл***я.***

1 – рубл***ь***
2, 3, 4 – рубл***я***
0, 5, 6, 7, 8, 9 … 10 – рубл***ей***
21 – рубл***ь***
22, 23, 24 – рубл***я***
25, 26, 27 … 30 – рубл***ей*** и т.д.

Наблюдаем некоторую закономерность.

1 – рубл***ь***	**2, 3, 4** – рубл***я***	**0, 5, 6, 7, 8, 9** – рубл***ей***
10, 11, 12, 13, 14 …19 – рублей		
21– рубл***ь***	2**2**, 2**3**, 2**4** – рубл***я***	2**0**, 2**5**, 2**6**, 2**7**, 2**8**, 2**9** – рубл***ей***
31– рубл***ь***	3**2**, 3**3**, 3**4** – рубл***я***	3**0**, 3**5**, 3**6**, 3**7**, 3**8**, 3**9** – рубл***ей***
……..		
91– рубл***ь***	9**2**, 9**3**, 9**4** – рубл***я***	9**0** ……9**9** – рубл***ей***

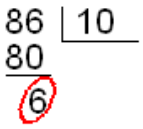
Заметили? Какое бы десятичное число мы не взяли, результат зависит *от последней цифры* в числе. А выявить последнюю цифру в каждом десятичном числе нам поможет функция mod. В качестве делителя будем использовать число 10.

1. Например, найти остаток от деления 91 на 10.

```
91 |10
90  9
 1
```

Остаток равен единице: 91/10=9,**1**. Следовательно, 91 рубл***ь***

2. Найти остаток от деления 86 на 10



Остаток от деления равен шести: 86/10=8,**6**. Следовательно, 86 рубл***ей***

Следовательно, при составлении программы нам нужно ориентироваться только на первую строку нашей таблицы.

Пусть вводимое пользователем число рублей – в программе будет переменной n.

```
Program lab3_3;
Var n : Integer;    {число рублей}
      ost : Integer;  {остаток от деления n на 10 (ищем последнюю цифру)}
Begin
Writeln ('Введите число (от 1 до 99)');
Readln (n);
If (n>=11) and (n<=19)
    Then writeln (‘рублей’)
ost:= n mod 10
CASE ost OF
    0, 5 .. 9: writeln (‘рублей’);
    1:  writeln (‘рубль’);
    2 .. 4: writeln (‘рубля’);
End;
END.
```

{этот случай мы выделяем отдельно – это есть вторая строка нашей таблицы, т.к. 11/10=1,1, остаток =1, но мы не можем сказать 11 рубл***ь***, также 12, 13, 14 и до 19}

Протестируйте и сохраните программу.

Третья контрольная точка.

Задания для самостоятельной работы

I. Ответьте на вопросы:

1. Чем отличается оператор CASE от оператора IF? Объясните на примере схематичной структуры каждого.
2. Запишите синтаксис оператора CASE.
3. Укажите как работает оператор CASE. Поясните основные понятия.
4. Почему в некоторых программах целесообразнее применять оператор CASE, а не IF?
5. Какого типа в операторе выбора CASE может быть использована переменная-селектор? А какого типа не может быть использована?
6. Какие типы селектора выбора вам известны? В чем суть каждого типа?
7. Для чего применяется функция mod?

II. Выберите подходящий по вашему усмотрению уровень сложности заданий и выполните его на компьютере.

Уровень I. (на оценку «3»)

1. Дано целое число в диапазоне 1 – 5. Вывести строку — словесное описание соответствующей оценки (1 — "плохо", 2 — "неудовлетворительно", 3 — "удовлетворительно", 4 — "хорошо", 5 — "отлично").
2. Написать алгоритм, который по номеру дня недели - целому числу от 1 до 7 выдавать в качестве результата количество уроков в классе в соответствующий день

Уровень II. (на оценку «4»)

1. Вовочка, любитель стрелять из рогатки, 7 раз попадал в милицию. Ввести с клавиатуры целое положительное число – № попадания. Определить результат: 4,6,7 – милиционеры вставляли новое стекло, 2,5 – новое стекло вставлял папа Вовочки, 1, 3 – стекло не разбилось
2. Напишите программу, которая по введенному числу из промежутка 0..24, определяет время суток

Уровень III. (на оценку «5»)

1. Определить, является ли введенная буква русского алфавита гласной.
2. Написать алгоритм, классифицирующий треугольники (остроугольные, прямоугольные, тупоугольные), если даны углы.
3. Написать программу, которая после введенного с клавиатуры числа (в диапазоне от 1 до 99), обозначающего денежную единицу, дописывает слово «копейка» в правильной форме. Например, 5 копеек, 41 копейка и т.д.

Лабораторная работа №4 «Оператор безусловного перехода»

Цель работы: Ознакомиться со средством перемещения по программе на примере оператора безусловного перехода.

Каждый дом на улице имеет свой номер, все люди имеют собственные имена, даже ячейки памяти компьютера имеют каждая свой адрес. Все это принято для того, чтобы иметь возможность однозначно указать на определяемый объект. Точно также, для указания на операторы в программах применяются **метки.**

Во всех приведенных ранее программах операторы выполнялись один за другим в том порядке, в котором они были записаны в тексте. Такая алгоритмическая структура называется прямым следованием. Однако в языке Паскаль изначально существует оператор, нарушающий прямолинейное выполнение программы, передающий управление в произвольную ее точку.

Оператор перехода Goto

Если в программе требуется нарушить порядок выполнения операторов без предварительных проверок каких-либо условий, например, необходимо обойти участок программы, а вернуться к нему позже, такой переход называется безусловным.
Оператор перехода имеет вид:

Goto ИмяМетки;

Метка в стандарте языка Паскаль представляет собой целое неотрицательное число. Все используемые в программе метки должны быть перечислены в разделе описания меток, начинающемся служебным словом Label:

label <список_всех_меток_через_запятую>;

Приведем общий вид программы, содержащий конструкцию goto – label и поясним принцип ее работы

Общий вид программы	Пояснения к программе
Program .... **Var** ..... **Label metka1;** .... **Begin** ..... **metka1: оператор;** .... **Goto metka1;** .... **End.**	Начало программы остается без изменений Далее следует служебное слово Label, и собственно описание самой метки, которая будет задействована в программе. Меток может несколько и все они должны быть описаны здесь. Метки должны быть объявлены в разделе описания меток прежде, чем они будут использоваться Одной меткой можно пометить только один любой оператор. Метка от помеченного оператора отделяется двоеточием. Для того чтобы вернуть управление программы к по***меченному*** выше оператору, используем оператор перехода goto. Выполнение программы начнется с места, которое мы зафиксировали словом metka1.

Любая метка может встретиться в тексте программы только один раз.

Использовать оператор безусловного перехода следует крайне осторожно во избежание получения ошибочных результатов или полного "зацикливания" программы.
Приведем пример с использованием данного оператора.

Задание 1.

С клавиатуры вводится положительное двузначное число N. Определить кратно ли оно 12. Предусмотреть проверку вводимых значений.

Определим, что значит число положительное двузначное? ***Если*** число положительное и двузначное, ***то*** оно должно находиться в диапазоне между числами 10 и 99, ***иначе*** число нужно ввести еще раз. Понятно, что в программе, помимо всего прочего, будет работать условный оператор IF. А вот отсылать пользователя к новому вводу числа будет новый, безусловный, оператор GOTO.

Алгоритм выполним в виде блок-схемы (см. рис.7).

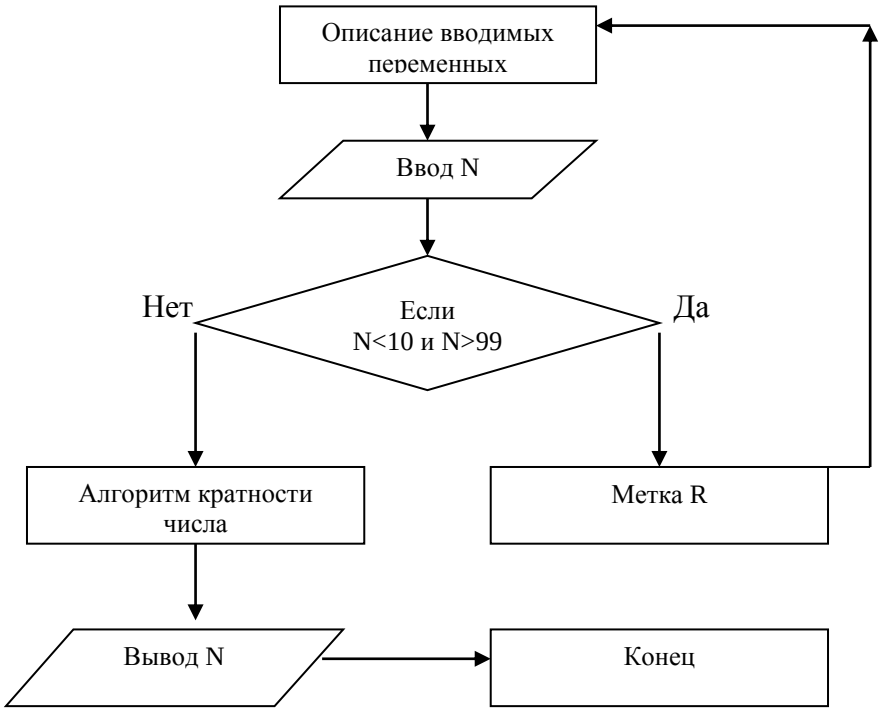


Рис.7

Итак, требуется определить кратность числа 12. Это значит, что при делении вводимого числа на 12, остаток должен быть равен нулю. Работает функция mod.

Программа будет иметь следующий вид:

```
Program lab4_1;
   Var N: Integer;
   Label R;                                  ← описываем метку
   Begin
   R: Writeln ('введите число');             ← помечаем меткой оператор ввода числа
   Readln (N);
   If (N<10) or (N>99) then goto R;          ← если условия не выполнимы, отправляем к метке
   If N mod 12=0 then Writeln ('число кратно 12')
            else Writeln('число не кратно 12');
   Readln;
   End.
```

Протестируйте и сохраните программу.

Первая контрольная точка.

Задание 2.

Составить программу, которая позволит решить квадратное уравнение.

Вспомним основное правило решения квадратного уравнения, используя дискриминант. Общий вид квадратного уравнения $Ax^2+Bx+C=0$. В зависимости от этих коэффициентов уравнение либо не имеет действительных корней, либо имеет два корня (сюда ж отнесем случай, когда уравнение имеет один корень – будем считать, уравнение имеет два одинаковых корня).

При составлении алгоритма и при написании программы будем использовать, как и в предыдущей задаче, операторы условного и безусловного перехода.

Наш алгоритм:

1. Имя программы
2. Описание меток и всех переменных.
3. Ввод коэффициентов уравнения A,B,C
4. Нахождение дискриминанта. D=B*B-4*A*C
5. Условие решения уравнения
   а) если D<0 Уравнение корней не имеет. Переход на End.
   б) если D>0 Уравнение имеет два действительных корня.
   X1= (-B+корень (D))/ (2*A)
   X2= (-B-корень (D))/ (2*A)
6. Вывод результата на экран.
7. Стоп.

Программа	Пояснения к программе
Program lab4_2; Var A, B, C: Integer; D: Integer; X1, X2: Real; Label 100,200; Begin Writeln (‘Введите коэффициенты уравнения'); Readln (A); Readln (B); Readln(C); D: =B*B-4*A*C; If D<0 Then GOTO 100; Begin X1:=(-B+SQRT (D))/(2*A); X2:=(-B-SQRT (D))/(2*A); Writeln('X1=',X1:5:2); Writeln ('X2=', X2:5:2); Readln; GOTO 200; 100: Writeln (Уравнение не	Назовем программу KWUR Опишем переменные: A, B, C – коэффициенты уравнения; D – дискриминант; X1, X2 – корни уравнения (они имеют вещественный тип, т.к. область их значений может быть гораздо большая, чем может позволить нам целочисленный тип); Опишем две метки. После ввода коэффициентов, подсчитаем дискриминант. Если дискриминант <0, т.е. не имеет корней, то отправим программу на завершение по метке 100. Если условие не выполняется, пропускаем его и начинаем считать корни уравнения. Поскольку для переменных X1 и X2 использовался тип Real, выводимое число может иметь большое количество знаков после запятой. Для форматирования данных чисел используется знак **:** (двоеточие), и указывается количество знаков до и после запятой в выводимом числе. Пустой оператор Readln используется для задержки экрана. Далее используем безусловный оператор, отправляя

имеет корней'); End; 200: End.	программу на завершение по метке 200.

Протестируйте и сохраните программу.

Вторая контрольная точка.

Одним из случаев, когда программисту может показаться полезным оператор GOTO, является необходимость прекратить выполнение программы при возникновении той или иной ошибки.

Задание 3.

Написать программу, которая вычисляет некоторую функцию от квадратного корня из заданного числа.

Для вычисления корня квадратного из заданного числа $\sqrt{x}$, в паскале существует функция SQRT(x). Применим ее в данной программе.

На первый взгляд все кажется весьма простым. Вводим число, извлекаем из него корень квадратный с помощью известной функции, и выводим результат на экран.

```
  Program lab4_3_a;
VAR x: Real;
BEGIN
WRITE ('Введите число ');
READ(x);
x:=SQRT(x);   {вычисление функции от x}
WRITE (x);
END.
```

Однако, если введено отрицательное число, то в третьем операторе программы произойдет аварийное прерывание. Стремясь избежать этого, мы применим оператор GO-TO:

```
Program lab4_3_b;
VAR x: Real;
LABEL Finish;
BEGIN
WRITE ('Введите число ');
READ(x);
IF x<0 THEN GOTO Finish;
x:=SQRT(x);     {вычисление функции от x}
Finish: END.
```

Но! Это как раз тот случай, когда можно не использовать GOTO:

```
Program lab4_3_c;
VAR x: Real;
BEGIN
WRITE ('Введите число ');
READ(x);
IF x<0 THEN WRITELN (‘ошибка!')
        ELSE BEGIN
                x:=SQRT(x);   {вычисление функции от x}
               END;
END.
```

Как видите, программа даже стала лучше, т.к. теперь она сообщает о неправильном вводе. Но она все-таки имеет один недостаток - условный оператор усложнил структуру программы[3].

Протестируйте и сохраните все три программы.

Третья контрольная точка.

Задания для самостоятельной работы

I. Ответьте на вопросы:

1. Что представляет собой оператор безусловного перехода?
2. Укажите его назначение и особенности применения.
3. Укажите общий вид программы, содержащей оператор безусловного перехода.
4. Сколько раз можно использовать в тексте программы одну и ту же метку?
5. Почему рекомендуется избегать оператора Goto?

II. Выберите подходящий по вашему усмотрению уровень сложности заданий и выполните его на компьютере.

Уровень I. (на оценку «3»)

1. С клавиатуры вводится положительное двузначное число N. Определить кратно ли оно 15. Предусмотреть проверку вводимых значений
2. Напишите программу, реализующую деление одного целого числа на другое, с применением оператора безусловного перехода.

Уровень II. (на оценку «4»)

1. С клавиатуры вводится положительное трехзначное число N. Определить кратно ли оно 3. Предусмотреть проверку вводимых значений
2. Применив оператор безусловного перехода для заданного X вычислить значение A по формулам:
   $A = X + 1$, если $X < 0$
   $A = 0$, если $X > 10$

Уровень III. (на оценку «5»)

1. Применив оператор безусловного перехода, для заданного X вычислить значение A по формулам:
   $A = X^2 + 1$, если $X < 0$
   $A = 2X$, если $0 =< X < 10$
   $A = X^2 – 1$, если $X >= 10$
2. Составить программу, которая позволит решить квадратное уравнение при любых значениях его коэффициентов (включая возможность равенства одного из них нулю)

[3] В этом и в других подобных случаях очень удобно пользоваться стандартной процедурой Паскаля HALT, которая останавливает выполнение программы, будучи вызвана в любом ее месте