1 курс

ПЛАН – КОНСПЕКТ
проведения занятия по дисциплине «Информатика»

Раздел «Основы алгоритмизации и программирования.»

Тема: «Общие принципы построения базовых алгоритмических структур в среде программирования.»

часть 3

Подготовил: преподаватель
В.Н. Борисов

Рязань 2024

Вопросы занятия:

1. Операторы ввода и вывода в Turbo Pascal.
2. Составной оператор Begin … end.
3. Базовая структура следование.
4. Среда разработки.
5. Управление выводом на экран.
6. Создание программы для расчета по заданной формуле (практическое занятие, теоретическая часть).

Время проведения занятия – 2 часа.

Первый вопрос: Операторы ввода и вывода в Turbo Pascal.

В Паскале ввод осуществляется с помощью процедур **read** и **readln**, а вывод - благодаря **write** и **writeln**. Процедуры, которые имеют окончание **ln**, после своего выполнения переводят указатель на новую строку.

Стандартным устройством ввода является клавиатура, а вывода — монитор. Стандартные — значит "работающие по-умолчанию"; то есть если не указано ничего иного, то программа будет считывать данные с клавиатуры, а выводить их на монитор. Вместе клавиатуру и монитор называют *консолью*. Таким образом, консоль представляет собой стандартное устройство ввода-вывода.

Решение даже самой простой задачи на компьютере не обходится без операций ввода – вывода информации. Ввод данных – это передача информации от внешнего носителя в оперативную память для обработки. Вывод – обратный процесс, когда данные передаются после обработки из оперативной памяти на внешний носитель (экран монитора, принтер, дискету или винчестер и другие устройства). Выполнение этих операций производится путем обращения к стандартным процедурам: Read, Readln, Write, Writeln.

Вывод данных на экран.

Процедура вывода **Write** производит вывод данных.

Общий вид: **Write(<список вывода>);**

В списке вывода могут быть представлены выражения допустимых типов данных (**integer, real, char** и т. д.) и произвольный текст, заключенный в апострофы.

Например: **Write('Привет');**

Write(34.7);
Write(45+55); Write(b, d);

Если в качестве параметра в списке вывода стоит строковая константа в апострофах, то она выводится в том же виде. Если в качестве параметра в списке вывода стоит переменная, то выводится её значение. Если в качестве параметра в списке вывода стоит выражение, то сначала вычисляется его значение, а затем оно выводится на экран.

Процедура **Writeln** аналогична процедуре **Write**. Отличие в том, что после вывода последнего в списке выражения курсор переходит на начало новой строки.

В процедурах вывода **Write** и **Writeln** имеется возможность записи выражения, определяющего ширину поля вывода.

При рассмотрении форматов вывода примем следующие обозначения:

- **I** – целочисленное выражение;
- **R** – выражение вещественного типа;
- **_** – пробел.

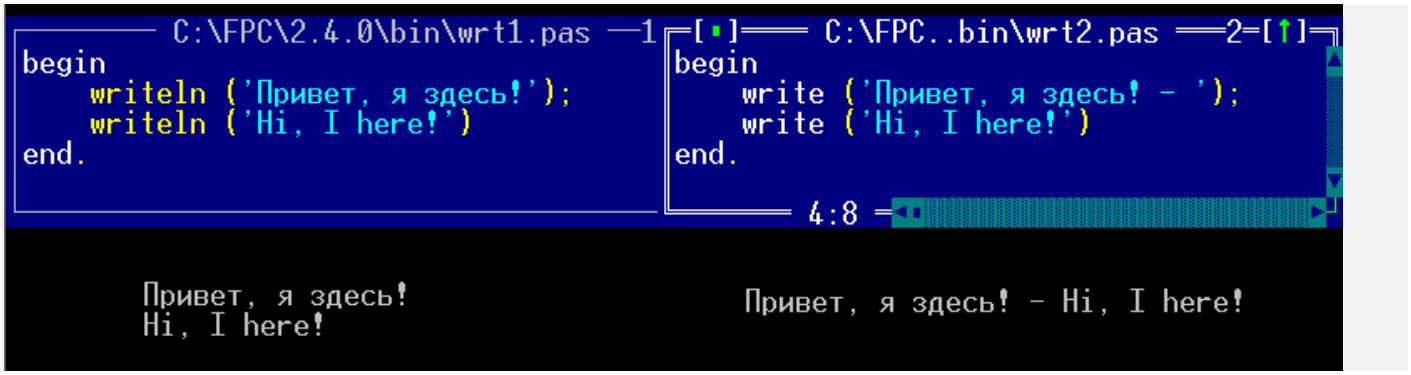
Таблица 2.1.

Значение I	**Выражение**	**Результат**
324	Write (I);	324
34	Write (I,I,I);	343434
324	Write (I:6);	___324
312	Write (I+I:7);	____624
Значение R	Выражение	Результат
123.432	Write (R);	1.2343200000E+02
-1.34E+01	Write (R);	-1.3400000000E+01
304.55	Write (R:15);	3.045500000E+02
Значение R	Выражение	Результат
304.66	Write (R:9:4);	_304.6600
45.322	Write (R:7:2);	__45.32

Оператор **Writeln;** без параметров просто переводит курсор на новую строку, ничего не выводя.

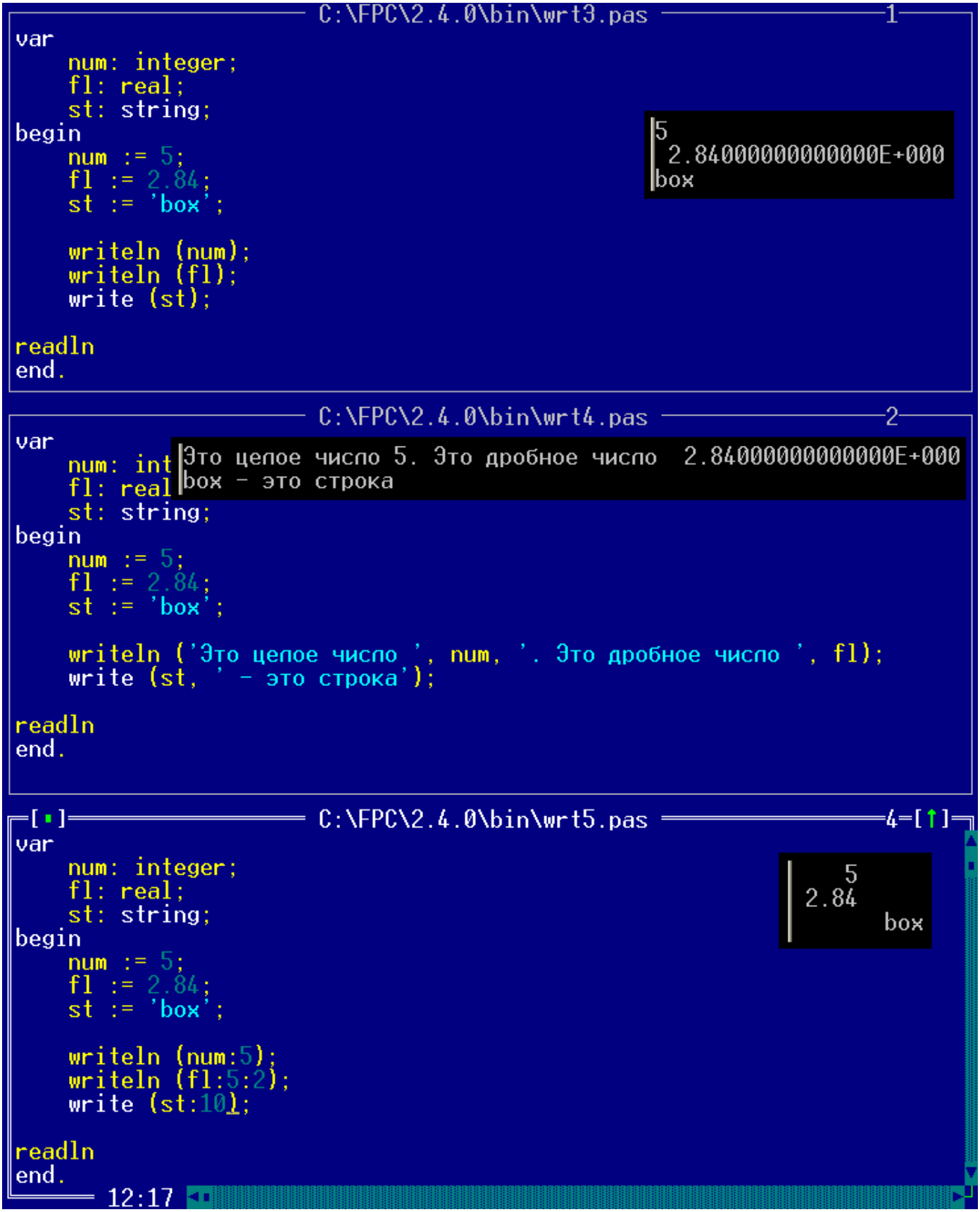
Вывод данных на экран и в файл в языке программирования Pascal осуществляется с помощью процедур write и writeln. Здесь будет рассмотрен вывод только на экран.

Допустим, нам требуется отобразить на экране пару фраз. Если мы хотим, чтобы каждая из них начиналась с новой строки, следует использовать writeln, если нет – то write.



Write нередко используется, когда надо вывести для пользователя сообщение на экран, после чего получить данные, не переводя курсора на новую строку. Например, выводим на экран "Введи число: " и не переводим курсор на новую строку, а ждем ввода.

Еще один пример. В памяти компьютера хранятся данные. Из программы мы обращаемся к ним с помощью переменных **num, fl и st.** Вывести их значения на экран можно по-разному.



Во втором случае мы видим, что процедуры вывода, как **write, так и writeln**, позволяют конструировать выводимую информацию из различных частей (строковых литералов и переменных).

В третьем случае был осуществлен так называемый форматированный вывод. При этом для выводимого значения указывается ширина поля вывода (количество знакомест). Если мы выводим дробное значение, то вторым числом через двоеточие указывается количество знаков после запятой. Если для вещественных чисел не осуществлять форматирование, то они отобразятся так, как определено для данного компьютера. Если указать только число знакомест без фиксирования дробной части, то вывод будет в экспоненциальной форме.

Пример. Использование операторов вывода.

```
program primer;
  var a,b,c,sum:integer;
begin
a:=4; b:=6; c:=55;
Write(a:3); Write(b:3); Write(c:3); Writeln;
Sum:=a+b+c;
Writeln ('A=',a);
Writeln ('B=',b);
Writeln ('C=',c);
Writeln ('Сумма A+B+C равна ', sum);
end.
```

Результат выполнения:

4 6 55

A=4

B=6

C=55

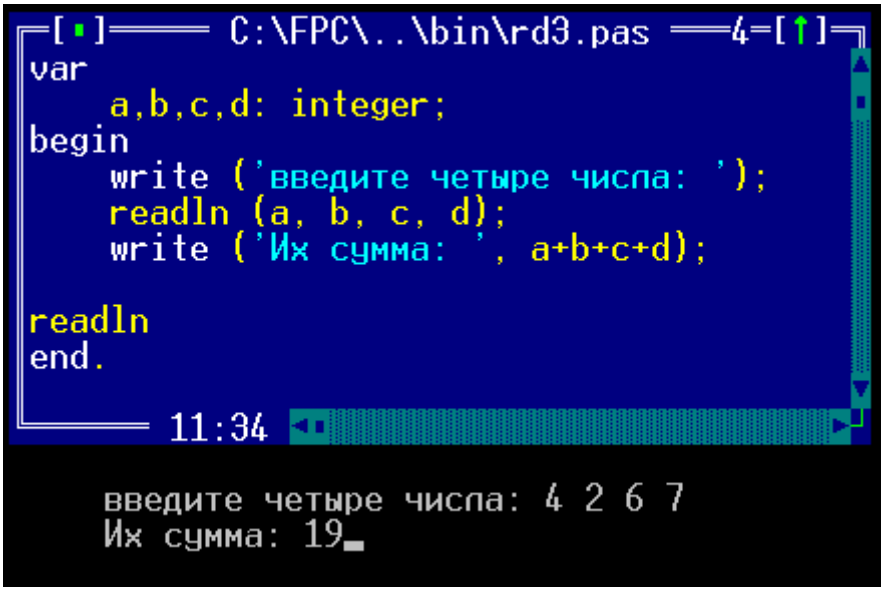
Сумма A+B+C равна 65

Ввод данных с клавиатуры.

Ввод данных в языке программирования Паскаль обеспечивается процедурами **read** и **readln.** Ввод данных осуществляется либо с клавиатуры, либо из файла. Здесь рассматривается только ввод с клавиатуры.

Когда данные вводятся, то они помещаются в ячейки памяти, доступ к которым обеспечивается с помощью механизма переменных. Поэтому, когда в программе на Pascal используется процедура read или readln, то в качестве фактического параметра (аргумента) ей передается имя переменной, которая будет связана с вводимыми данными. Потом эти данные можно будет использовать в программе или просто вывести на экран.

В процедуры ввода можно передавать не один фактический параметр, а множество.



При вводе данных их разделяют пробелом, табуляцией или переходом на новую строку (Enter). Данные символьного типа не разделяются или разделяются переходом на новую строку.

Существуют особенности ввода данных с помощью операторов read и readln. Если используются подряд несколько read, то вводимые данные можно разделять всеми допустимыми способами. При использовании нескольких вызовов readln каждый последующий срабатывает только после нажатия Enter. Программа ниже иллюстрирует это. Комментарии поясняют последовательность возможных действий при вводе данных.

Процедура чтения **Read** обеспечивает ввод данных для последующей их обработки программой. Чаще эту процедуру называют оператором ввода.

Общий вид: **Read** (<список переменных>);

В списке перечисляются имена переменных. Значения этих переменных набираются через пробел на клавиатуре и высвечиваются на экране после запуска программы. После набора данных для одной процедуры Read нажимается клавиша ввода Enter. Значения переменных должны вводиться в строгом соответствии с синтаксисом языка Паскаль. Если соответствие нарушено, то возникают ошибки.

Процедура чтения **Readln** аналогична процедуре **Read,** единственное отличие в том, что после считывания последнего в списке значения курсор переходит на начало новой строки.

Пример 2.2. Использование операторов ввода.

```
program primer;
```

```
var   i, k:integer;  c,d, s: real;

begin

readln (c,d);

read(i,k);

…

end.
```

В данном случае необходимо ввести сначала два действительных числа через пробел. Переменной **c** присваивается значение, равное первому введенному числу, а переменной **d** – значение, равное второму введенному числу. После ввода этих значений курсор переходит на начало новой строки (за это отвечает **ln**, следующий за словом **Read**). Далее требуется ввести еще два целых числа, которые будут присвоены значениям переменных **i** и **k** соответственно.

Оператор ввода работает следующим образом. После запуска программы, когда её выполнение доходит до оператора ввода, программа приостанавливает свою работу и ждёт от вас ввода значений переменных с клавиатуры. При этом на экране мерцает курсор. После ввода требуемых значений программа продолжает работу дальше, выполняя последующие операторы.

Как вы могли заметить, такая организация работы не совсем удобна, т. к. нужно помнить, значения каких переменных нужно ввести. При большом их количестве легко запутаться.

Гораздо удобнее для ввода значений переменных организовать диалог с компьютером. Для этого непосредственно перед каждым оператором ввода переменной нужно поставить оператор вывода **Write** для вывода на экран поясняющего текста:

```
Write ('X='); Readln(X);

Write ('Z='); Readln(Z);

или

Write ('Введите значение A > '); Readln(A);
```

Предпочтительно использовать именно такой ввод переменных с клавиатуры.

Пример 2.3. Организация ввода переменных.

```
program primer;

var   X,Z,Sum: real;
```

begin

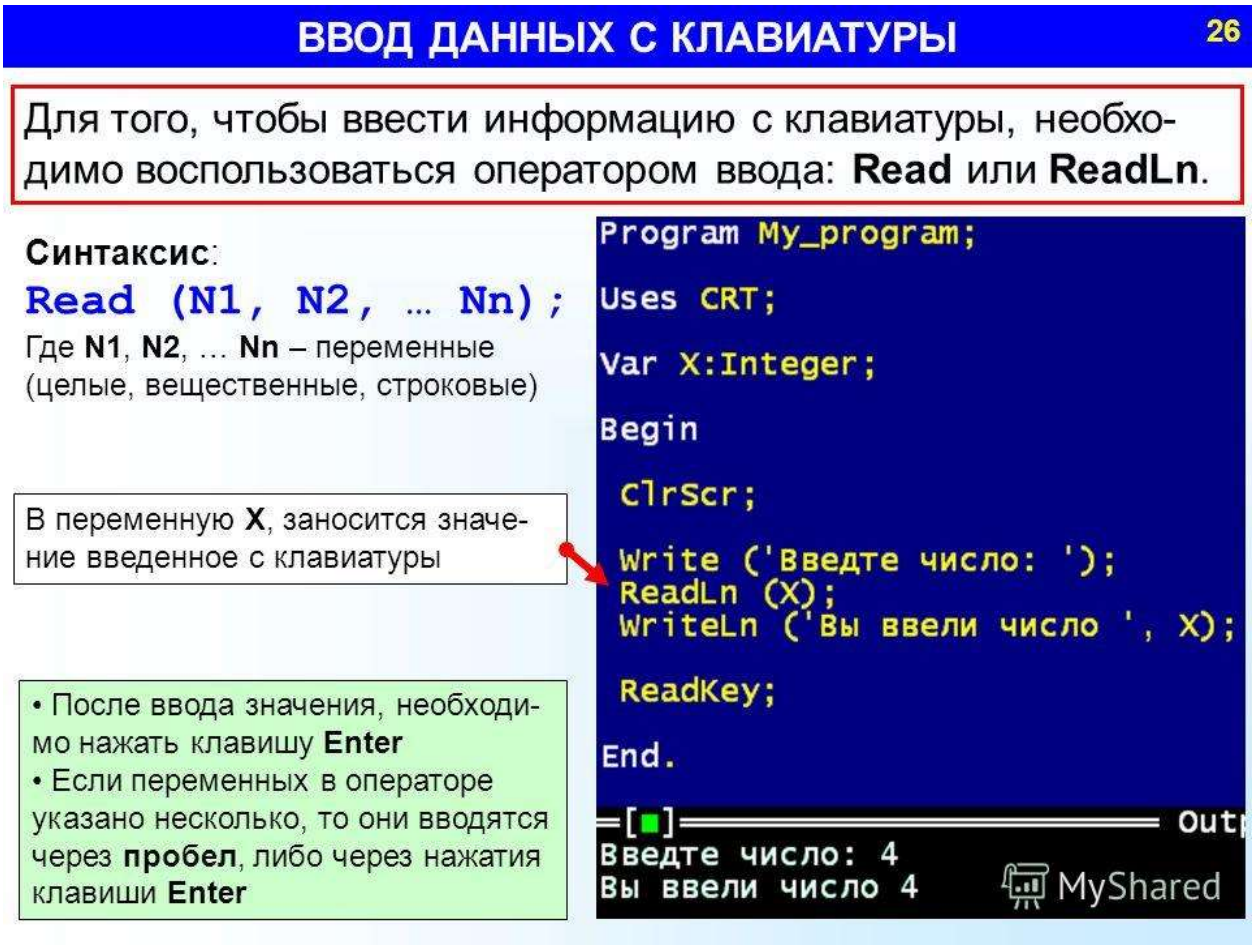
Write (‘X=’); Readln(X);

Write (‘Z=’); Readln(Z);

Sum:=X+Z;

Writeln (‘Сумма равна ’, Sum:10:2);

end.



Второй вопрос: <u>Составной оператор Begin … end.</u>

Оператор в программе - это единое неделимое предложение, выполняющее какое-либо действие. <u>Простой оператор</u> не содержит в себе других операторов. Типичный простой оператор - это оператор присваивания. Другим примером может являться вызов процедуры. Важно, что под любым оператором подразумевается какое-либо <u>действие</u>. В этом смысле составляющие блока описания типов, перменных, констант и т.д. операторами не являются.

Два последовательных оператора должны разделяться точкой с запятой. Этот символ имеет смысл конца оператора, в частности, он разделяет операторы при записи в строку:

a := 11; b:=a*a; Write(a,b);

Если какое-то действие мыслится как единое (например, присваивание значений элементам массива), но реализуется несколькими различными операторами, то эти операторы могут быть представлены как составной оператор. Составной оператор - это последовательность операторов, перед которой стоит слово **begin**, а после слово - **end**. Слова *begin* и *end* сами не являются операторами, поэтому после *begin* и перед *end* точка с запятой не ставится. Чтобы оформить предыдущий пример в один, но составной оператор, необходимо как бы заключить их в операторные скобки *begin...end*:

begin

a := 11;

b := a*a;

Write(a,b)

end;

Составной оператор может содержать любое допустимое число простых операторов. Кроме того, он допускает вложенность, т.е. может содержать внутри себя другие составные операторы (в этом случае нужно лишь, чтобы внутренний составной оператор открывался позже, чем внешний, а закрывался - раньше).

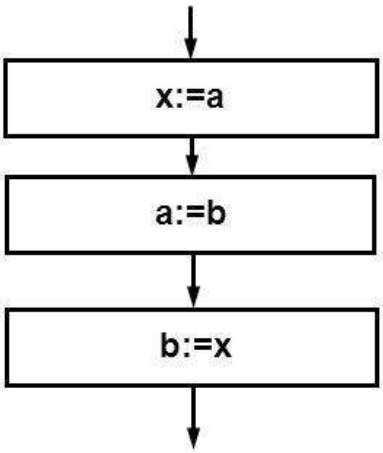
Пустой оператор не содержит никаких действий, просто в программу добавляется дополнительная точка с запятой (например, в последнем примере перед *end*). В основном пустой оператор используется для передачи управления в конец составного оператора.

Третий вопрос: Базовая структура следование.

Следование – действия выполняются строго в том порядке, в котором записаны. Образуется последовательностью действий, следующих одно за другим.

Следование — простейшая алгоритмическая структура. Программа, реализующая следование, называется линейной программой. В линейной программе могут присутствовать только операторы присваивания, ввода, вывода и обращения к процедурам. Заметим, что операторы Read и Write являются обращениями к стандартным процедурам Паскаля.

I. Линейная (следование). Пример:

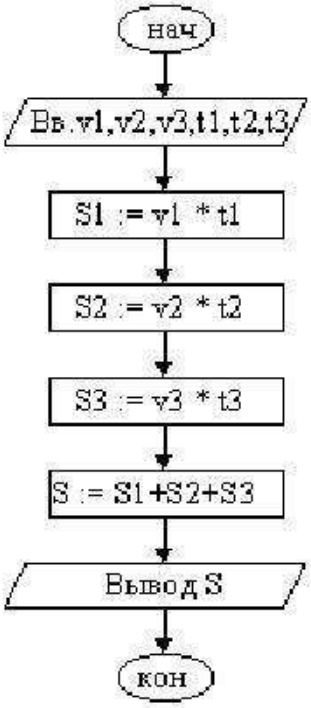


в языке Pascal

```
x:=a;
a:=b;
b:=x
```

Основные алгоритмические структуры

- ***Следование***



Пешеход шел по пересеченной местности. Его скорость движения по равнине v1 км/ч, в гору — v2 км/ч и под гору — v3 км/ч. Время движения соответственно t1, t2 и t3 ч. Какой путь прошел пешеход?

1. Ввести v1, v2, v3, t1, t2,t3.
2. S1 := v1 * t1.
3. S2 := v2 * t2.
4. S3 := v3 * t3.
5. S := S1 + S2 + S3.
6. Вывести значение S.
7. Конец.

Четвертый вопрос: Среда разработки.

Базовыми компонентами система программирования Турбо Паскаль являются компилятор языка Паскаль, средства создания и редактирования исходных текстов программ и средства их отладки (поиска и исправления ошибок). Все эти компоненты объединены в единую **интегрированную среду разработчика**, с которой как раз и работает программист, создавая свои программы.

Запуск программы

Если ваш компьютер настроен для более удобной работы с Турбо Паскалем (в файловом менеджере прописаны ассоциации файлов, настроено пользовательсткое меню), то для начала работы со средой разработчика удобнее всего поступить следующим образом:

1. Запустить Norton Commander (Far Manager);
2. Зайти в каталог, в котором планируется сохранять файлы с исходными текстами программы, а также вспомогательные файлы вашей прграммы;
3. Вызвать горячее меню Norton Commander (нажав клавишу F2);
4. Выбрать строку "Turbo Pascal 7.0";

5. Если окно редактирования не открылось, то открыть его через пункт меню "File" (нажать Alt+F, выбрать New).

Если у вас уже есть некоторый файл с исходным текстом программы (файл с расширением pas), с которым вы хотите продолжить работу, то достаточно навести на него указатель Norton Commander и нажать Enter. В этом случае запустится Turbo Pascal и сразу откроется текст выбранной вами программы.

Окно среды разработчика.

Основной экран интегрированной среды разработчика Turbo Pascal 7.0 выглядит следующим образом:

По функциональному назначению выделяется три области экрана:

- Строка меню
- Рабочая область
- Строка состояния

Строка меню активизируется нажатием клавиши F10. В меню содержатся следующие разделы:

- **File**. Позволяет выполнять все основные действия с файлами (создание, открытие, сохранение ..)
- **Edit**. Позволяет выполнять все основные операции редактирования текста (копирование, вставка, удаление фрагментов, отмена последних изменений ..)
- **Search**. Позволяет осуществлять поиск и замену фрагментов текста.
- **Run**. Позволяет запускать программу, в том числе в пошаговом режиме.
- **Compile.** Позволяет осуществлять компиляцию программы.
- **Debug**. Содержит команды, облегчающие процесс поиска ошибок в программе.
- **Tools**. Содержит некоторые дополнительные средства Турбо Паскаль.

- **Options**. Позволяет установить необходимые для работы параметры компилятора и среды разработчика.
- **Window**. Позволяет выполнять все основные операции с окнами (открывать, закрывать, перемещать, изменять размер).
- **Help**. Позволяет получить имеющуюся в системе справочную информацию.

Все пункты меню доступны через горячие клавиши. Для этого надо нажать клавишу Alt и ту букву, которая выделена красной в названии пункта меню. Меню также позволяет работать с мышью.

В рабочей области имеется возможность открывать различные окна программы - окна редактируемого текста, окна помощи, отладки и настройки. В вышеприведенном примере открыто только одно окно - окно текста программы. В заголовке окна написано имя файла - исходного текста программы.

Строка состояния демонстрирует некоторые доступные и важные в данный момент операции и соответствующие им комбинации клавиш.

Основные команды и горячие клавиши.

Ниже приведены основные команды среды раработчика Турбо Паскаль и соответствующие им горячие клавиши. Более полный перечень горячих клавиш вы можете найти в приложении.

- **Ctrl+F9** - запуск программы
- **Alt+F5** - просмотр пользовательского экрана
- **F2** - сохранение программы
- **F3** - открытие сохраненной программы
- **Alt+F3** - закрытие активного окна
- **Alt+X** - выход из Турбо Паскаль
- **F1** - контекстная помощь
- **Ctrl+F1** - справка об операторе, на котором установлен курсор
- **Alt+Backspace** - отмена последнео изменения
- **Ctrl+Y** - удаление строки
- **Shift+стрелки** - выделение блока текста
- **Ctrl+Insert** - копирование выделенного блока в буфер
- **Shift+Insert** - вставка из буфера

Пятый вопрос: Управление выводом на экран.

Процедура вывода **Write** производит вывод данных.

Общий вид: **Write(<список вывода>);**

ВЫВОД НА ЭКРАН 8

Результат применения операторов:

WRITE

```
Program My_Program;

Begin

 Write('Message 1');
 Write('Message 2');
 Write('Message 3');

End.
```

```
[▪]
Message 1Message 2Message 3_
```

WRITELN

```
Program My_Program;

Begin

 WriteLn('Message 1');
 WriteLn('Message 2');
 WriteLn;
 WriteLn('Message 3');

End.
```

```
[▪]
Message 1
Message 2

Message 3
```

«Пустой» оператор **WRITELN** добавляет пустую строку

Вывод данных

```
write( a );      { вывод значения
                   переменной a}
writeln( a );    { вывод значения
                   переменной a и переход
                   на новую строку}
writeln( 'Привет!' );  { вывод текста }
writeln( 'Ответ: ', c );
 {вывод текста и значения переменной c}
writeln ( a, '+', b, '=', c );
```


Шестой вопрос: Создание программы для расчета по заданной формуле (практическое занятие, теоретическая часть).

в лабораторной работе № 6.

Задачи, разбираемые в данной лабораторной работе, имеют следующий алгоритм выполнения:

1) ввод исходных данных с клавиатуры;
2) вычисление формулы;
3) вывод результатов вычислений на дисплей.

При составлении алгоритмов необходимо руководствоваться ГОСТ 19.701-90, выдержка из которого приведена в приложении 1.

Пример 1.1. Написать алгоритм и программу для вычисления формулы

$$y = \frac{(b+5)^3 \cdot \sin x}{\lg(x+1)+\sqrt{b}} \quad \text{для } b=5{,}5;\ x=0{,}1;\ 0{,}5.$$

Схема алгоритма, реализующего поставленную задачу, изображена на рисунке 2. Здесь отметим, что прежде, чем ввести данные с клавиатуры (элемент № 3 алгоритма), необходимо побудить к этому оператора (элемент № 2). Ниже приведен текст программы.

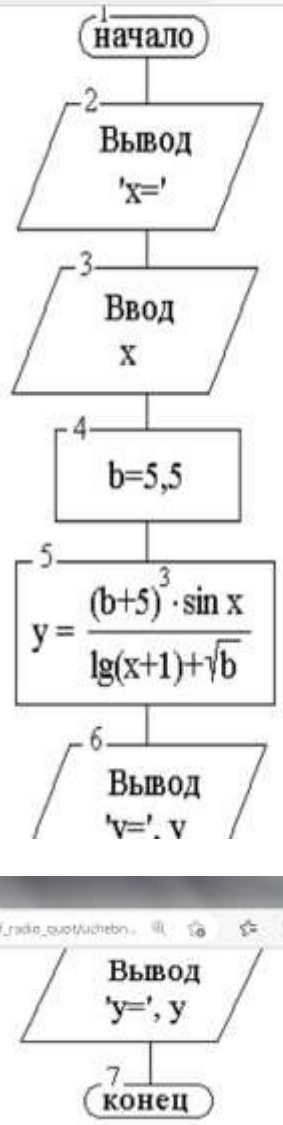


Рисунок 2

```
program lab1;
const B=5.5;
var y, x:Real;
begin
   Write('x=');
   Read(x);
   y:=exp(3*ln(B+5))*sin(x)/(ln(x+1)/ln(10)+sqrt(B));
   WriteLn('y=',y:6:2);
end.
```

Программирование обычно осуществляется в среде Турбо Паскаль, в которую при работе в дисплейном классе можно попасть через меню пользователя, зайдя предварительно в рабочий каталог. Если соответствующая команда в меню пользователя отсутствует, то необходимо включить в меню программу turbo.exe. При работе в среде Windows можно создать на рабочем столе ярлык для этой программы.