**ПРОГРАММИРОВАНИЕ НА ПАСКАЛЕ**

**1. Структура программы на Паскале**

**2. Система программирования Турбо Паскаль**

**1. Структура программы на Паскале.**

По определению стандартного Паскаля программа состоит из **заголовка**программы и **тела**программы (блока), за которым следует **точка** *–* признак конца программы. В свою очередь, блок содержит разделы **описаний**и раздел **операторов***.*

Program <имя программы>;

Label <раздел меток>;

Const <раздел констант>;

Туре <раздел типов>;

Var <раздел переменных>;

Procedure (Function) <раздел подпрограмм>

Begin

<раздел операторов>

End.

Раздел операторов имеется в любой программе и является основным. Предшествующие разделы носят характер описаний и не все обязательно присутствуют в каждой программе.

В Турбо Паскале, в отличие от стандарта, возможно следующее:

* отсутствие заголовка программы;
* разделы Const, Type, Var, Label могут следовать друг за другом в любом порядке и встречаться в разделе описаний сколько угодно раз.

**Примеры программ.** Уже было сказано, что Паскаль разрабатывался Н. Виртом как учебный язык. Основной принцип, заложенный в нем, — это поддержка структурной методики программирования. Этот же принцип лежит в основе псевдокода, который используется для обучения школьников составлению алгоритмов и называется **Алгоритмическим языком** (АЯ). Расхождение между АЯ и Паскалем заключается в следующем: АЯ — русскоязычный, Паскаль — англоязычный; синтаксис Паскаля определен строго и однозначно в отличие от сравнительно свободного синтаксиса АЯ.

Запись программы на Паскале похожа на английский перевод алгоритма, записанного на Алгоритмическом языке.

Пример 1. Алгоритм деления простых дробей, записанный на АЯ, с соответствующей программой на Паскале.

Здесь использовано следующее равенство:



**алг** деление дробей; **Program** Division;

**цел** a,b,c,d,m,n; **Var** a,b,c,d,m,n: Integer;

**нач** ввод(a,b,с,d); **Begin** ReadLn(a,b,с,d);

m:=a\*d; m:=a\*d;

n:=b\*c; n:=b\*c;

вывод(m,n) WriteLn(m,n)

**кон**. **End**.

Заголовок программы начинается со слова **Program** (программа), за которым следует произвольное имя, придуманное программистом (division *–* деление). Раздел описания переменных начинается со слова **Var** (variables— переменные), за которым следует список переменных. Тип указывается после двоеточия словом **integer** — целый. Начало и конец раздела операторов программы отмечаются словами **Begin** (начало) и **End** (конец). В конце программы **обязательно ставится точка***.*

Ввод исходных данных с клавиатуры производится с помощью процедуры **ReadLn** (read line *—* читать строку). На клавиатуре набирают четыре числа, отделяемые друг от друга пробелами, которые отражаются строкой на экране дисплея. После набора чисел нажимают на клавишу ввода.

Операторы присваивания в Паскале записываются так же, как в АЯ **:=**чтобы отличать от знака строгого равенства **=**. Знак умножения — \* (звездочка).

Вывод результатов на экран дисплея производится с помощью процедуры **WriteLn** (write line— писать в строку). В рассмотренном примере два целых числа выведутся в строчку, курсор на экране перейдет в начало следующей свободной строки и работа программы завершится.

Необходимо строгое соблюдение правил правописания (синтаксиса) программы. В частности, в Паскале однозначно определено назначение знаков пунктуации. **Точка с запятой** (;) ставится в конце заголовка программы, в конце раздела описания переменных, после каждого оператора. Перед словом End точку с запятой можно не ставить. **Запятая** (,) является разделителем элементов во всевозможных списках: списке переменных в разделе описания, списке вводимых и выводимых величин.

Строгий синтаксис в языке программирования необходим прежде всего для транслятора. Транслятор — это программа, которая исполняется формально. Если, допустим, разделителем в списке переменных должна быть запятая, то любой другой знак будет восприниматься как ошибка. Если точка с запятой является разделителем операторов, то транслятор в качестве оператора воспринимает всю часть текста программы от одной точки с запятой до другой. Если вы забыли поставить этот знак между какими-то двумя операторами, то транслятор будет принимать их за один, что неизбежно приведет к ошибке.

**Основное назначение синтаксических правил — придать однозначный смысл языковым конструкциям***.* Если какая-то конструкция может трактоваться двусмысленно, значит, в ней обязательно содержится ошибка.

Пример 2. «Оттранслируем» алгоритм вычисления факториала натурального числа (N!) на Паскале.

****

**алг** ФАКТОРИАЛ; **Program** FACTORIAL;

**цел** N,I,F; **Var** N,I,F: Integer;

**нач** ввод(N); **Begin** ReadLn(N);

F:=1; F:=1;

I:=1; I:=1;

**пока I<=N While I<=N Do**

**нц F:=F\*I; Begin** F:=F\*I;

I:=I+1 I:=I+1

**кц**; **End;**

вывод(F) WriteLn(F)

**кон. End.**

В этом примере используется оператор **цикла с предусловием**(цикл-пока):

**While** <условие выполнения> **Do** <тело цикла>

(While — пока, Do — делать). Если тело цикла содержит последовательность операторов, то говорят, что оно образует **составной****оператор***,* в начале и в конце которого надо писать Begin и End. Служебные слова Begin и End называют **операторными скобками***,* которые объединяют несколько операторов в один составной. Если же тело цикла — один оператор (не составной), то операторных скобок не требуется. Тогда транслятор считает, что тело Цикла заканчивается на ближайшем знаке «;».

Пример 3. Рассмотрим еще один пример программы — решение квадратного уравнения.

**алг** КОРНИ; **Program** ROOTS;

**вещ** a,b,с,d,x1,х1; **Var** a,b,c,d,x1,x2: Real;

**нач Begin** {ввод данных с

**повторять** контролем}

вывод(«введите a,b,c; **Repeat**

a0»); WriteLn('введите а,

ввод(а,b,с); b,с;а< >0');

**до** а0; ReadLn(a,b,с)

d:=b^2-4\*a\*c; **Until a****0;**

**если** d0 {вычисление дискриминанта}

**то** d:=b\*b-4\*a\*c;

x1:=(-b+sqrt(d))/(2\*а); If d>=0

х2:=(-b-sqrt(d))/(2\*а); **Then Begin** {есть корни}

вывод(x1,x2) x1:=(-b+sqrt(d))/(2\*a);

**иначе** вывод(«нет вещ. х2:=(-b-sqrt(d))/(2\*а);

корней»); WriteLn('x1=',x1,

**кв** 'x2=',x2);

**кон**. **End**

**Else** WriteLn('нет вещ. корней')

 **End.**

В этой программе по сравнению с предыдущими появилось много новых элементов.

 Имя вещественного типа в Паскале — **real**.

**Цикл с постусловием**(цикл-до) программируется оператором

**Repeat <тело цикла> Until <условие окончания>**

(здесь Repeat — повторять, Until — до). Тело цикла может быть как одиночным, так и составным оператором, однако употребления Begin и End не требуется, поскольку сами слова Repeat и Until выполняют роль операторных скобок.

Знак **не равно**в Паскале пишется так:< >, знак **больше или равно***:* **>=**.

В формулах вычисления корней используется стандартная функция квадратного корня **sqrt(x)**.

Порядок выполнения операций в выражении определяется скобками и старшинством операций. Старшинство операций такое же, как и в алгебре. Операции одинакового старшинства выполняются в порядке их записи (слева направо).

Ветвление в Паскале программируется с помощью **условного оператора**;который имеет следующую форму:

**If <условие> Then <оператор 1> Else <оператор 2>**

(здесь if — если, Then — то, Else — иначе). Операторы 1 и 2 могут быть как простыми, так и составными. Составной оператор следует заключать в операторные скобки Begin и End.

Возможно использование неполной формы условного оператора:

**if <условие> then <оператор>**

Характерной чертой данной программы является использование в тексте комментариев.

**Комментарий**— это любая последовательность символов, заключенных в фигурные скобки {...}. Можно употреблять также следующие ограничители комментариев (\*...\*). Комментарий не определяет никаких действий программы и является лишь пояснительным текстом. Он может присутствовать в любом месте программы, где можно поставить пробел. Наличие подробных комментариев является обязательным условием структурной методики программирование. Комментарий придает тексту программы большую ясность. Хорошо откомментированные программы называют **самодокументированными**. Во многих подобных программах объем комментариев превышает объем вычислительных операторов.

Удачное использование комментариев — признак хорошего стиля программирования.

Чтобы выполнить программу на ЭВМ, ее нужно **ввести в память***,* **оттранслировать**и **исполнить***.* Для того чтобы проделать всю эту работу, на компьютере должны быть специальные средства программного обеспечения – система программирования.

**2. Система программирования Турбо Паскаль**

Название Турбо Паскаль обычно воспринимается в двух смыслах: как диалект языка Паскаль, представляющий собой расширение стандартного Паскаля; как система программирования Турбо Паскаль, являющаяся совокупностью системных программ, предназначенных для создания, отладки и выполнения Паскаль-программ.

Систему программирования **Турбо** Паскаль предоставляет пользователю необходимые средства работы с Паскаль-программами.

Так как система программирования **Турбо** Паскаль опирается в своей работе на возможности операционной системы (MS DOS), то каждая конкретная система программирования **Турбо** Паскаль может работать с определенной операционной системой, ее конкретными версиями.

Турбо Паскаль (как язык программирования и как операционная оболочка) значительно изменился за историю своего существования. Первый вариант Турбо Паскаля фирма Borland выпустила в середине 1980-х гг. К сегодняшнему дню этой фирмой созданы шесть модификаций системы, известных как версии 3.0, 4.0, 5.0, 5.5, 6.0, 7.0. Каждая из них представляет собой усовершенствование предыдущей версии. Все они создавались для семейства машин IBM PC и совершенствовались вместе с компьютерами.

Версия 3.0 ориентирована на ПК малой мощности (IBM PC/XT). Разрабатываемые на ней программы имеют ограничение на длину (не более 64 Кбайт); в этой версии нет средств раздельной компиляции взаимосвязанных программ; операционная среда весьма несовершенна.

Большие изменения были внесены в версию 4.0. Появились современная диалоговая среда, средства раздельной компиляции программных модулей, мощная графическая библиотека.

Версия 5.0 отличается в основном дальнейшими усовершенствованиями среды, к которой добавлен встроенный отладчик. В версию 5.5 были впервые включены средства поддержки объектно-ориентированного программирования — современной технологии создания программ.

Главные отличия версии 6.0: новая среда, ориентированная на работу с устройством ввода — мышью и использующая много оконный режим работы; объектно-ориентированная библиотека Turbo-Vision, а также возможность включать в текст программы команды Ассемблера.

Версия 7.0 не содержит каких-то принципиальных новшеств по сравнению с 6.0. Введены некоторые расширения языка программирования, а также дополнительные сервисные возможности системной оболочки.

Программа на Турбо Паскале проходит три этапа обработки: 1) создание текста программы; 2) компиляция; 3) исполнение откомпилированной программы.

В соответствии с этими функциями Турбо-система включает в себя три главные компоненты: 1) редактор текстов; 2) компилятор; 3) исполнительную систему.

С помощью встроенного в систему текстового редактора можно формировать в памяти любые тексты, не только программы на Паскале. В частности, это могут быть исходные данные решаемой задачи в текстовой форме. Текст программы, созданный редактором, можно сохранить на диске в виде файла с именем следующего формата

<имя файла>.pas

где pas — это стандартное расширение имени файла, созданного системным редактором. Имя файла задается пользователем.

Обращение к текстовому редактору происходит по команде **Edit** системного меню**.**

Компилятор переводит программу с языка Паскаль на язык машинных команд. При этом проверяется соответствие программы правилам языка программирования (синтаксический и семантический контроль). При обнаружении ошибки компьютер выдает сообщение о ней пользователю и прекращает работу. Программа, полученная в результате компиляции, может быть сохранена на диске в файле с именем

<имя файла>.ехе

Работа компилятора инициируется системной командой **Compile Ctrl+F9.**

Исполнение откомпилированной программы производится по команде **Run F9** системного меню**.** При этом исполнение программы остается под контролем системы программирования Турбо-Паскаль. В частности, система программирования Турбо-Паскаль помогает обнаружить ошибку в программе, если при исполнении произошел сбой. Пользователю сообщается причина сбоя и указывается место, где он случился в Паскаль-программе. Происходит автоматический возврат в режим редактирования.

В старших версиях Турбо Паскаля имеется система отладки **(Debug).** С ее помощью можно просмотреть на экране значение любой переменной, найти значение любого выражения, установить новое значение переменной. Можно прервать выполнение программы в указанных местах, которые называются контрольными точками. Система отладки существенно облегчает программисту поиск ошибок.

**3. Элементы языка Турбо Паскаль**

**Алфавит.** Алфавит языка состоит из множества символов, включающих в себя **буквы**, **цифры** и **специальные символы***.*

1. *Латинские буквы:* от *А* до *Z* (прописные) и от *а* до *z* (строчные).
2. *Цифры:0*, 1,2, 3, 4, 5, 6, 7, 8, 9.
3. *Шестнадцатеричные цифры:* 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, А, В, С, D, E, F.
4. *Специальные символы:* + — \*/ = < > [] . , () : ; {} ^ @ $ #.

Следующие комбинации специальных символов являются едиными символами (их нельзя разделять пробелами):

:= знак присваивания;

<= меньше или равно;

>= больше или равно;

(\*…\*) ограничители комментариев (используются наряду с { }, // );

 <> не равно;

(..) эквивалент [ ].

1. *Пробелы* — символ пробела (ASCII-32) и все управляющие символы кода ASCII (от 0 до 31).
2. К спецсимволам относятся *служебные слова*, смысл которых определен однозначно. Служебные слова не могут быть использованы для других целей. С точки зрения языка это единые символы. Вот список служебных слов Турбо Паскаля:

absolute end inline procedure type

and external interface program unit

array file interrupt record until

begin for label repeat uses

case forward mod set var

const function nil shl while

div goto not shr with

do if of string xor

downto implementation or then

else in packed to

Последние версии языка содержат еще ряд служебных слов, относящихся к работе с объектами и встроенным ассемблером.

**Идентификаторы.** Идентификатором называется символическое имя определенного программного объекта. Такими объектами являются имена констант, переменных, типов данных, процедур и функций, программ. С помощью синтаксической диаграммы идентификатор можно определить, как показано на рис. 1.



Рис. 1. Определение идентификатора с помощью синтаксической диаграммы.

Расшифровать это можно так: идентификатор — это любая последовательность букв и цифр, начинающаяся с буквы. В Турбо Паскале к буквам приравнивается также знак подчеркивания. Строчные и прописные буквы в идентификаторах и служебных словах не различаются. Например: max, мах, Мах и mАх — одно и то же имя.

Длина идентификатора может быть произвольной, но значащими являются только первые 63 символа.

**Комментарии.** Следующие конструкции представляют собой комментарии и поэтому игнорируются компилятором:

{любой текст, не содержащий символ «}» }

(\* любой текст, не содержащий символы «\*)»\*)

Буквы русского алфавита употребляются только в комментариях, в литерных и текстовых константах.

Строка, начинающаяся с символов {$ или (\*$, является **директивой компилятора***.* За этими символами следует мнемоника команды компилятора.

1. Напишитепрограмму на Паскале которая: а) выбирает наибольшее значение из трех; б) определяет существования треугольника с данными длинами сторон;

**2.** Опишите синтаксис операторов: цикл с предусловием, цикл с постусловием, условный оператор.

**Дополнительные упражнения**

«Оттранслируйте» с Алгоритмического языка на Паскаль следующие алгоритмы:

а) алгоритм Евклида;

б) алгоритм выбора наибольшего значения из трех;

в) алгоритм определения существования треугольника с дан
ными длинами сторон;

г) алгоритм умножения двух целых чисел с использованием
только операций сложения и вычитания;

д) алгоритм вычисления частного и остатка от целочисленно
го деления.

**3.4. Типы данных**

Концепция типов данных является одной из центральных в любом языке программирования. *С типом величины связаны три ее свойства: форма внутреннего представления, множество принимаемых значений и множество допустимых операций.* Турбо Паскаль характеризуется большим разнообразием типов данных, отраженном на рис. 9.

В стандартном Паскале отсутствует строковый тип. Кроме того, в Турбо Паскале целые и вещественные — это группы типов. В старших версиях Турбо Паскаля существует процедурный тип и тип объект.

Каждый тип имеет свой идентификатор.

В табл. 3.1 представлена информация о простых типах данных, определенных в Турбо Паскале. Для вещественных типов в скобках указано количество сохраняемых значащих цифр мантиссы в Десятичном представлении числа.