Введение

Pascal — это императивный язык программирования, разработанный Никлаусом Виртом в 1970 в качестве языка обучения структурному программированию. Название языку дано в честь выдающегося французского математика, физика, литератора и философа Блеза Паскаля. Первоначально язык компилировался в байт-код, подобно языку Java.

Особенностями языка являются строгая типизация и наличие средств структурного (процедурного) программирования. Паскаль был одним из первых таких языков. По мнению Н. Вирта, язык должен способствовать дисциплинированию программирования, поэтому, наряду со строгой типизацией, в Паскале сведены к минимуму возможные синтаксические неоднозначности, а сам синтаксис интуитивно понятен даже при первом знакомстве с языком.

Тем не менее, первоначально язык обладал множеством недостатков: невозможность передачи функциям массивов переменной длины, отсутствие нормальных средств работы с динамической памятью, ограниченная библиотека ввода-вывода, отсутствие средств для подключения функций написанных на других языках, отсутствие средств раздельной компиляции и т. п.

Структура программы.

В программе,написанной на языке Pascal, могут быть следующие разделы :

\* заголовок программы; program имя;

\* список используемых модулей; uses cписок исп.модулей;

\* раздел объявления переменных; var

\* раздел объявления процедур и функций; procedure,function

Begin

\* тело программы . End.

Заголовок программы состоит из зарезервированного слова program и имени программы (со списком параметров ,заключенных в круглые скобки). Завершается заголовок точкой с запятой.

Порядок размещения разделов произвольный. Разделы могут отсутствовать.

Тело программы начинается словом begin , а заканчивается словом end с точкой, которая является признаком конца программы.

const имя:тип=значение;

Обычные константы могут быть целого,вещественного,символьного,логического типа и типа string, типизированные константы- любого типа,кроме типа файл.

Раздел переменных:

В этом разделе должны быть описаны все переменные,встречающиеся в программе:

var список1:тип\_1;

список2:тип\_2;

список\_1,список\_2-перечень переменных через запятую. Тип переменной можно задать двумя способами :

а) указать имя типа из раздела type,

б) описать сам тип.

Раздел процедур и функций содержит описание процедур и функций,вызываемых в теле программы.

Подробнее см. раздел Процедуры и функции.

Тело программы содержит операторы языка Pascal.

|  |  |  |  |
| --- | --- | --- | --- |
| Integer | -32768..32767 | знаковый | 2 |

Вещественные типы данныж.

|  |  |  |  |
| --- | --- | --- | --- |
| Тип | Диапазон | Число знечащих цифр | Размер в байтах |
| real | 2.9\*10^-39..  1.7\*10^38 | 11-12 | 6 |

Тип comp фактически является типом целых чисел увеличенного диапазона, однако порядковым не считается.

Типы single,double,extended,comp можно использовать в программах только при наличии арифметического сопроцессора или включенного эмулятора сопроцессора.

Нельзя присваивать переменной значение из описания другого типа.

В языке Паскаль нельзя непосредственно вводить и выводить на внешние устройства значения перечисляемых типов.

В следующем фрагменте с внешнего носителя вводится порядковый номер объекта из списка значений перечисляемого типа . Оператор case присваивает соответствующее значение переменной clr.

ВЫРАЖЕНИЯ

Выражения состоят из операндов , связанных знаками операций , и круглых скобок.

Операнды выражений:

Операндами являются константы,переменные,стандартные функции ( и написанные сам-но).

Строковые или символьные константы:

Cтрока символов- это последовательность любого, в том числе и равного нулю , количества символов , расположенных на одной строке и заключенных в апострофы ( ‘ ).

Максимальный размер строки- не более 126 символов.

Строка , состоящая из одного символа , называется символьной константой.

Если между апострофами нет ни одного символа, то такая строка называется нулевой строкой.

Переменными называются параметры программы, значения которых могут изменяться в процессе ее выполнения.

Все используемые в программе переменные должны быть определены с указанием их типов. Переменные определяются :

1. в разделе объявления переменных программы
2. в разделе объявления переменных подпрограммы
3. в разделе объявления переменных модуля
4. в заголовке подпрограммы

Раздел объявления переменных начинается зарезервированным словом var , за которым следуют объявления конкретных переменных,состоящие из имени переменной ( имен переменных, перечисленных через запятую,если они одного типа ),двоеточия и типа переменной ( переменных). Каждое объявление завершается точкой с запятой:

var

x,y,z:real;

i,j,s:integer;

В разделе объявления переменных наряду с предопределенными типами можно использовать типы,объявленные перед этим в разделе объявления типов , а также новые, вводимые только для конкретных переменных.

Переменные.объявленные не в программе, называются глобальными, т.к. они доступны во всех частях программы,в том числе и в подпрограммах, и размещаются в сегменте данных размером 65520 байтов.

Переменные, объявленные в подпрограмме ,называются локальными,т.к. доступны только этой подпрограмме, и размещаются в стековом сегменте. По умолчанию стековый сегмент имеет размер 16384 байта, но его можно изменять . Глобальные переменные постоянно находятся в памяти,локальные размещаются в стеке лишь на время выполнения програмы.

Cтандартные функции

1. арифметические функции

Знаки операций

Все операции в Pascale можно разбить на следующие группы:

1. арифметические операции
2. логические операции
3. операции с битами информации
4. операции со строками
5. операции отношения
6. адресная операция @

|  |  |  |
| --- | --- | --- |
| Функция | Назначение | Тип результата |
| abs(x) | Абсолютное значение аргумента | Совпадает с типом х |
| arctan(x) | Арктангенс аргумента | Вещественный |
| cos(x) | Косинус аргумента | Вещественный |
| exp(x) | х  е | Вещественный |
| Frac(x) | Дробная часть числа | Вещественный |
| Int(x) | Целая часть числа | Вещественный |
| Ln(x) | Натуральный логарифм | Вещественный |
| Pi | Значение величины  pi=3.1415926535897932385 |  |
| Sin(x) | Синус аргумента | Вещественный |
| Sqr(x) | Квадрат аргумента | Совпадает с типом х |
| Sqrt(x) | Квадратный корень аргумента | Вещественный |

Если в операциях используются несколько данных,то их типы должны быть либо идентичными, либо совместимыми.

Арифметические операции

|  |  |  |  |
| --- | --- | --- | --- |
| Знак | Операция | Типы операндов | Тип результата |
| + | Сложение | Целые  Хотя бы один вещественный | Целый  Вещественный |
| - | Вычитание | Целые  Хотя бы один вещественный | Целый  Вещественный |
| \* | Умножение | Целые  Хотя бы один вещественный | Целый  Вещественный |
| / | Деление | Целые или вещественные | Вещественный |
| div | Деление целых чисел | Целые | Целый |
| mod | Остаток от деления целых чисел | Целые | Целый |

Знаки операций + ,- и \* используются также и с другими типами операндов,но тогда они имеют другой смысл. При использовании знака операции,являющегося служебным словом, он должен быть отделен от операндов хотя бы одним разделителем: a mod b.

В Pascale имеется два вида вычисления логических выражений : полное и краткое. Полное вычисление означает,что вычисляется каждый операнд,даже если уже известен результат всего выражения.

Укороченное вычисление производится до тех пор,пока не станет известен результат всего выражения, после чего вычисление последующих операндов прекращается. Выбор выда вычисления осуществляется с помощью ключа компилятора $B.

Операции отношения.

Операции отношения предназначены для сравнения двух величин ( величины должны быть сравнимых типов). Результат сравнения имеет логический тип.

|  |  |
| --- | --- |
| Операция | Назначение |
| = | равно |
| <> | не равно |
| < | меньше |
| <= | меньше или равно |
| > | больше |
| >= | больше или равно |

Круглые скобки.

Круглые скобки используются для заключения в них части выражения,значения которой необходимо выполнить в первую очередь. В выражении может быть любое количество круглых скобок, причем количество открывающих круглых скобок должно быть равно количеству закрывающих. Части выражений, заключенные в круглые скобки, должны быть либо не пересекающимися, либо вложенными друг в друга.

Порядок вычисления выражений .

|  |  |  |
| --- | --- | --- |
| Группа | Тип действий | Операции или элементы |
| 1 | Вычисления в круглых скобках | ( ) |
| 2 | Вычисления значений функции | Функции |
| 3 | Унарные операции | @,not,унарный +,унарный - |
| 4 | Операции типа умножения | \*,/,div,mod,and,shl,shr |
| 5 | Операции типа сложения | +,-,or,xor |
| 6 | Операции отношения | =.<>,<,>,<=,>=,in |

Простые операторы

К простым операторам относятся те операторы, которые не содержат в себе других операторов:

1. оператор присваивания;
2. обращения к процедуре;
3. оператор безусловного перехода goto ;
4. пустой оператор.

Оператор присваивания

С помощью этого оператора переменной или функции присваивается значение выражения.

Для этого используется знак присваивания := , слева от которого записывается имя переменной или функции, которой присваивается значение ,а справа- выражение, значение которого вычисляется перед присваиванием.

Допустимо присваивание значений переменным и функциям любого типа, кроме типа файл. Тип выражения и тип переменной должны быть совместимы для присваивания.

X:=y;

z:=a+b;

r:=( i>0) and (i<100);

Оператор безусловного перехода goto . Использование меток.

Оператор goto позволяет изменить стандартный последовательный порядок выполнения операторов и перейти к выполнению программы, начиная с заданного оператора. Оператор, на который происходит переход, должен быть помечен меткой. Эта метка должна быть указана и в операторе goto.

Метки, используемые в паскале должны быть двух типов:

1. целым числом в пределах от 0 до 9999;
2. обычным идентификатором.

Структурированными операторами являются такие операторы, которые состоят из других операторов:

1. составной
2. условный оператор if
3. условный оператор case
4. оператор цикла repeat
5. оператор цикла while
6. оператор цикла for
7. оператор над записями with

Cоставной оператор

Составной оператор представляет из себя совокупность последовательно выполняемых операторов, заключенных в операторные скобки begin и end;

begin

оператор1;

оператор2;

.................

операторn;

end;

Он может потребоваться в тех случаях, когда в соответствии с правилами построения конструкций языка можно использовать один оператор, а нужно выполнить несколько дейсвий. В такой составной оператор входит ряд операторов, выполняющих требуемые действия.

В дальнейшем, везде, где будет указываться, что можно использовать один оператор, им может быть и составной оператор.

Условный оператор.

Условный оператор позволяет проверить некоторое условие и в зависимости от результатов проверки выполнить то или иное действие.Таким образом,условный оператор-это средство ветвле-ния вычислительного процесса.

Структура условного оператора имеет следующий вид:

IF<условие>THEN<оператор1>ELSE<оператор2>,

где IF, THEN, ELSE - зарезервированные слова (англ.если,то,иначе);

<условие>-произвольное выражение логического типа;

<оператор1>,<оператор2>-любые операторы Паскаля.

Условный оператор работает по следующему алгоритму.Вначале

выполняется условное выражение.Если результат есть TRUE,то вы-

полняется <оператор1>,а<оператор2>пропускается; если результат есть FALSE,то,наоборот,<оператор1> пропускается,а выполняется <оператор2>.

Часть оператора ELSE <оператор2>может быть опущена.Тогда при значении условного выражения ТRUE выполняется <оператор1>, в противном случае этот оператор пропускается.

Поскольку каждый из операторов <оператор1>и<оператор2> может быть оператором любого типа,в том числе и условным,а в то же время не каждый из “вложенных” условных операторов может иметь часть ELSE<оператор2>,то возникает неоднозначность трак-товки условий. Эта неоднозначность в Паскале решается следующим образом: любая встретившаяся часть ELSE соответствует ближайшей к ней “сверху” части THEN условного оператора. Например:

Пример программы, анализирующей принадлежит ли точка с координатами x,y прямоугольнику с координатами вершин x1,x2,y1,y2.

**СТРОКИ**

**ОПИСАНИЕ ТИПА**

Для обработки последовательностей символов в ТУРБО-ПАСКАЛЕ предусмотрены строчные типы. Строчные типы - это структурные типы. Количество символов в строке (длина строки) может динамически изменяться от 0 до 255. Для определения данных строкового типа используется идентификатор string, за которым следует заключенное в квадратные скобки значение максимально допустимой длины строки данного типа. Если размер строки не указан, то он считается равным 255.

Переменную строкового типа можно определить через описание типа в разделе описания типов или непосредственно в разделе описания переменных. Строковые данные могут использоваться также в качестве констант. Недопустимо применение строковых переменных в качестве селектора в операторе case. Строковая переменная занимает определённую максимальную длину байт в памяти плюс один байт, содержащий её текущую длину. Отдельные символы в строке индексируются от 1 до длины строки.

Допускается смешение в одном выражении операндов строкового и литерного типа. Если при этом литерной переменной присваивается значение строкового типа, длина строки должна быть равна единице, иначе возникает ошибка выполнения. К отдельным символам строки, можно обратиться по номеру (индексу) данного символа в строке. Индекс определяется выражением целочисленного типа, которое записывается в квадратных скобках сразу за идентификатором строковой переменной или константы.

**ПРОЦЕДУРЫ**

Delete(St,Pos,N) - удаление N символов строки St, начиная с

позиции Pos. Если значение Pos > 255, возникает ошибка.

Значение St Выражение Результат

'река Волга' Delete(Str, 1, 5); 'Волга'

Insert(Str1,Str2,Pos) - вставка строки Str1 в строку Str2,

начиная с позиции Pos.

П р и м е р:

Var S1, S2, S3 : string[11];

...

S1 := ' ЕС '; S2 := 'ЭВМ1841 '; S3 := Insert(S1,S2,4);

Р е з у л ь т а т:

'ЭВМ ЕС 1841'.

Str(IBR,St) - преобразование числового значения величины IBR и помешение результата в строку St. После IBR может записываться формат, аналогичный формату вывода.

Значение Выражение Результат

4.8E+03 Str(IBR:10,St); ' 4800'

Val(St,IBR,Cod) - преобразует значение St в величину целочисленного или вещественного типа и помещает результат в IBR. Значение St не

должно содержать незначащих пробелов в начале и в конце. Cod - целочисленная переменная.

Если во время операции преобразования ошибки не обнаружено, значение Cod равно нулю, если ошибка обнаружена, Cod будет содержать номер позиции первого ошибочного символа, а значение IBR не определено.

Записи.

Запись есть структура данных,построенная из компонент,имеющих в общем случае разные типы данных.

Например, чтобы описать в программе время суток (10.45 утра или 7.15 вечера) , можно использовать структуру данных следующего формата:

|  |  |  |
| --- | --- | --- |
| часы | минуты | до полудня |

в которой 10.45 утра может храниться как

|  |  |  |
| --- | --- | --- |
| 10 | 45 | true |

а 7.15 вечера :

|  |  |  |
| --- | --- | --- |
| 7 | 15 | false |

Секция записи определяет одно или несколько имен полей и тип соответствующей компоненты. Если в секции перечисляется несколько имен полей, то все они однотипны.

Тип компонента в секции записи может быть любым типом данных и быть именем предопределенного типа, именем ранее определенного типа.

К каждому компоненту записи можно обратиться, используя имя переменной типа записи и имя поля, разделенные точкой: now.hour, later.befornoon;

Рассмотрим программу bestseller ( лучший товар), которая вводит с клавиатуры данные по сбыту за несколько лет, распечатывает их и находит продукцию с максимумом сбыта.

Входные данные : Каждая вводимая строка содержит информацию о продаже в следующей форме:

год продажи: целое в диапазоне 1900..2099;

код продукции: четырехзначное целое ;

общий объем сбыта: вещественное число, обозначающее сумму в денежных единицах.Ввод заканчивается,когда введена строка с отрицатедьным значением года.

Массивы записей.

Ввести список студентов из 5 человек, включающий в себя фамилию (10 символов), год рождения ( четырехзначное целое), оценки ( массив из 5 целых чисел).

Выдать данный список, добавив к нему средний балл. Выдать список отличников, упорядоченных по году рождения.

ПРОЦЕДУРЫ и ФУНКЦИИ

В практике программирования, типичной является такая ситуация . Когда в разных местах программы приходится выполнять по сути дела один и тот же частный алгоритм , который имеет достаточно самостоятельное значение , т.е. предназначен для решения некоторой подзадачи , выделенной из основной решаемой задачи . Например нахождение наибольшего общего делителя двух натуральных чисел , упорядочение компонентов вектора и т.д. Если этот частный алгоритм достаточно сложен и представляется достаточно большим фрагментом текста , то было бы явно нерационально выписывать его каждый раз заново в том месте программы , где этот алгоритм должен использоваться.

Для обеспечения большей компактности программы и повышения её наглядности язык Паскаль позволяет выделить любой частный алгоритм из программы и записать его только один раз, представив этот частичный алгоритм в качестве самостоятелной программной единицы , называемой процедурой.

ПОДПРОГРАММА-ПРОЦЕДУРА

Описание процедур. Процедуры описываются в специальном разделе основной программы вслед за разделом переменных. Любая процедура является логически законченной программной единицей и поэтому содержит внутри себя теже разделы , что и основная программа :

1. раздел описаний (меток , констант , типов , переменных процедур и функций);
2. раздел операторов, представляющий собой составной оператор BEGIN-END.

Процедура состоит , аналогично программе , из заголовка и тела процедуры.

Заголовок процедуры имеет вид:

PROCEDURE имя (список формальных параметров);

где PROCEDURE-служебное слово; имя-имя процедуры , определяемое в соответствии с общими правилами образования идентификаторов; список формальных параметров-перечень имен для обозначения исходных данных и результатов работы процедуры с указанием их типов.

Допускается описание процедуры не содержащей формальных параметров:

PROCEDURE имя;

Всякая подпрограмма-процедура может содержать свои внутренние или вложенные процедуры.Степень вложенности определяется возможностями машины.

Заканчивается блок процедуры точкой с запятой.

В заголовке процедуры с именем STEP1 перечислены парамет ры N,X, определяющие исходные данные программы и параметр Y, обозначающий значение искомой степени-результат выполнения процедуры.Указан также тип всех формальных параметров.Подроб-ней о параметрах будет расказано ниже. Обращение к подпрограмме-процедуре

Описание процедуры , расположенное в разделе описаний ,само по себе никаких действий не вызывает.Чтобы исполнить процедуру необходимо в нужном месте программы поместить обра-щение ней.Обращение производится с помощью оператора вызова процедуры или оператора процедуры , имеющего вид:

имя (список формальных параметров);

где имя-имя вызываемой процедуры;

список формальных параметров-перечень конкретных зна чений (выражений) и имен , передаваемых из основной программы в процеду-

ру на место формальных параметров. При вызове процедуры формальные параметры заменяются фактичес кими в порядке их следования.Число и тип формальных и факти-ческих параметров должны попарно совпадать.В данном примере оператор процедуры осуществляет её вызов.В теле процедуры произойдёт замена формальных переменных на фактические. Затем выполняется совокупность действий, предусмотренных операторами процедуры , и в основную программу будет возвращен результат Z.

ФУНКЦИИ

Заголовок содеpжит заpезеpвиpованное слово

FUNCTION,идентификатоp(имя) функции,заключенный в кpуглые скобки,необязатель-ный список фомальных паpаметpов и тип возвpащаемого функцией значения. FUNCTION <имя> {(фоpмальные паpаметpы)}:<тип pезультата>;

Имя функции-уникальный идентификатоp в пpеделах блока.Возвpащаемый pезультат может иметь любой скаляpный тип,тип string и тип “указатель”. Тело функции пpедставляет собой локальный блок,по стpуктуpе аналогичный пpогpамме:

function<имя>{(фоpмальные паpаметpы)}:<тип pезультата>;

<pазделы описаний>

begin

<pаздел опеpатоpов>

end;

В pазделе опеpатоpов должен находиться по кpайней меpе один опеpатоp, пpисваивающий идентификатоpу функции значение. Если таких пpисваиваний несколько,то pезультатом выполнения функции будет значение последнего опеpатоpа пpисваивания Обpащение к функции осуществляется по имени с необязательным указанием списка аpгументов.Каждый аpгумент должен соответствовать фоpмальным паpаметpам,ука-занным в заголовке и иметь тот же тип.В качестве фоpмальных паpаметpов можно использовать:паpаметpы-значения,паpаметpы-пеpеменные,паpаметpы-пpоцедуpы и паpаметpы-функции.

Процедурные типы данных.

В Turbo Pascale процедуры и функции можно рассматривать как некоторые параметры и можно использовать переменные, принимающие значение процедуры или функциию С этой целью вводятся процедурные типы, которые указывают , какой вид подпрограммы можно использовать в качестве параметра и с какими параметрами должны быть эти одпрограммы.

Описание переменных процедурного типа:

type имя\_типа1=procedure( формальные\_параметры);

имя\_типа2=function(формальные параметры): тип;

var имя1:имя\_типа1;

имя2:имя\_типа2;

Все процедуры и функции , имена которых присваиваются процедурным переменным, необходимо транслировать в режиме { $F+}. Вложенные функции и процедуры, т.е. , описанные внутри других процедур , с процедурными переменными использовать нельзя.Это не должны быть также стандартные роцедуры и функции.

ОПЕРАТОРЫ ВВОДА - ВЫВОДА

Для ввода данных в языке ПАСКАЛЬ предусмотрены стандартные процедуры READ и

READLN. Разлличают три вида операторов ввода:

а) READ(a1,a2,...,aN)-каждое вводимое значение поллучают послледовательно переменные a1,a2,...,aN

б) READLN(A1,A2,...,An)-переменным A1,A2,...,An присвиваются последовательно вводимые значения, после чего следующий оператор ввода вводит данные с новой строки.

в) READLN-переход на новую строку при вводе.

В языке ПАСКАЛЬ допускается вводить целые,действительные и символьные данные. Логические данные вводить не разрешается.

ВВОД ЧИСЛОВЫХ ДАННЫХ

Числовые данные, целые и действительные, должны разделятся пробелом или клавишей ENTER. Пробелы и ENTER перед числом игнорируются.

При выполнении программы происходит останов программы привстрече

READ(A,B,C), на клавиатуре необходимо набрать три действительных числа, после чего с новой строки два целых числа соответственно оператору

READ(K,T),например:

0.5 6.23 -7.1 [enter]

3 48 [enter]

ВВОД СИМВОЛЬНЫХ ДАННЫХ

Ввод символьных данных имеет свои особенности. Поскольку пробел относится к символьным данным то символьные данныевводятся сплошной строкой. Одной переменной можно присвоить значение только одного символа.

Пример ввода:

var a,b,c:char;

...........

read(a,b,c);

Если при вводе ввести данные -- SNR [enter] -- получим:A='S',B='N',C='R'

При вводе данныхв виде: -- S N R [enter] -- получим:A='S',B=' ',C='N'

ОПЕРАТОР ВВОДА ДЛЯ ЧТЕНИЯ ФАЙЛА

Оператор ввода для чтения файла обладает всеми свойствамии обычного оператора READ. Вкачестве параметров могут быть переменные; каждая переменная поучает значение одного эемента считанного из внешнего файа.

Процедура READ(F1,X) применяется для считывания (по одному) данных из внешнего файла в переменную X.

ОПЕРАТОР ВЫВОДА.

Опеатор вывода данных из памяти ЭВМ на экран дисплея имеет три формы:

1) WRITE(b1,b2,...bn)-выводит последовательно значения b1,b2,...,bn;

2) WRITELN(b1,b2,...,bn)-выводит последовательно значения b1,b2,...,bn и осуществляет переход на новую строку;

3) WRITELN-осуществляет переход на новую строку при выводе данных.

В качестве параметров b1,b2,...,bn могут быть целые,действительные,символьные и логические переменные, а так же символьные константы. Допускается вывод данных с форматами и без них. Форма представления выводимых данных зависит от типа данных: значения целых переменных выводятся как целые числа ,значения действительных переменных-в виде мантиссы и десятичного порядка, значения символьных переменных- в виде отдельных символов,значения логических переменных- в виде TRUE или FALSE.

Для выводимых значений переменных отводятся следующие поля: 13-для целого числа,13-для действительного, 1-для символа,6-для логических данных.Если целое число занимает позиций меньше, чем 13, то соответствующее поле перед числом заполняется пробелами.Если отсутствует формат для действительного числа, оно выводится в форме с плавающей точкой. Например число Z=10.9 при выводе в виде WRITE (`Z=`,Z) на экране будет представлено

Z= 1.090000E+01

Символьная константа(строка)выводится явно в том виде ,как она записана но без апострофов. Для наглядности выводимых значений прредусмотрены форматы. Они указываются за переменной в операторе WRITE через двоеточие. Для действительных переменных формат имеет вид:

a:m:n

где а-переменная, m-общее поле выводимого числа(включая знак числа, целую часть числа, точку и дробную часть); n-поле дробной части.

В качестве m и n могут быть как целые переменные,выражения,так и целые

константы. Например:

WRITE (SK:M:N,FGH:7:2);

Если формат указан больше,чем необходимо позиций для изображения числа, то перед целой частью числа представлены избыточные пробелы,а после дробной части-нули.

Для других типов данных формат имеет вид: a:m

Например:

WRITE(K:5,C:4);

Для вывода нескольких пробелов символ пробела указывается с форматом ,

Например: ` `:7

Для пpопуска пустых позиций пеpед выводимыми данными или между ними можно использовать фоpмат для символа пpобела,напpимеp

WRITE(` `:5,A:3,` `:7,B:4)

**МОДУЛИ**

Наличие модулей в Turbo Pascal позволяет программировать и отлаживать программу по частям , создавать библиотеки подпрограмм и данных, воспользоваться возможностями стандартных модулей , неограниченно увеличивать кодовую часть программы.

Модуль состоит из следующих частей:

а) заголовок модуля

б) интерфейс модуля

в) исполнительная часть модуля

г) секция инициализации

Все разделы модуля ,за исключением секции инициализации,являются обязательными . Обязательна также указанная последовательность рапзделов.

Заголовок модуля

Заголовок модуля состоит из зарезервированного слова unit и идентификатора.

Unit имя\_модуля;

Модуль должен быть помещен в файл, имя которого совпадает с именем модуля, а его расширение должно быть .pas .

Интерфейс модуля.

Через интерфейс осуществляется взаимодействие основной программы с модулем ( модуля с модулем) . В интерфейсе указываются константы, типы, переменные, процедуры и функции , которые могут быть использованы основной программой ( модулем) при вызове этого модуля. Интерфейс начинается словом interface. Далее после слова uses указываются имена модулей, которые используются данным модулем ( необязательная часть).После этого могут быть : раздел объявления констант, раздел объявления типов, раздел объявления переменных, раздел объявления процедур и функций. Объявление процедуры может содержать директиву inline. В разделе объявления процедур и функций указываются лишь заголовки программ ( за исключением тех, которые используют директиву inline) . Сами подпрограммы приводятся в исполнительной части. В интерфейсах различных модулей недопустимо циклическое обращение друг к другу.

Исполнительная часть модуля.

Исполнительная часть включает в себя все подпрограммы модуля. Она может также включать локальные метки, константы, типы и переменные.

Исполнительная часть начинается словом implementation . Затем после слова uses указываются имена модулей, которые используются подпрограммами данной исполнительной части ( этот раздел необязателен). Если какой-то модуль уже указан в интерфейсе модуля, то в исполнительной части его повторять не следует. Далее может следовать раздел локальных описаний . При описании подпрограмм допустимо использовать сокращенные заголовки.

Секция инициализации.

В некоторых случаях перед обращением к модулю следует провести его инициализацию ( установить связь с теми или иными файлами, инициализировать какой-то переменные). Эта секция начинается со слова begin , после которого идут выполняемые операторы, а затем слово end. ( c точкой).

Использование модуля в основной программе.

Чтобы использовать подпрограммы, константы, типы и переменные , описанные в интерфейсе модуля, в основной программе следует записать слово uses, после которого указать имена модулей . После этого в основной программе можно использовать идентификаторы, указанные в интерфейсах перечисленных модулей.Таким образом можно создавать собственные модули, расширяя библиотеку стандартных программ и функций.

Модуль Crt

Mодуль Crt содержит константы, переменные и подпрограммы,предназначенные для работы с консолью.

В отличие от стандартного ввода-вывода,когда он осуществляется через операционную систему, подпрограммы модуля Crt работают с BIOS , и даже непосредственно с видеопамятью.

При работе с экраном через модуль Crt весь экран разбивается на отдельные строки , а каждая строка - на отдельные позиции, в каждую из которых можно поместить один символ, (в том числе и пробел ). Т.о. весь экран разбивается на отдельные неделимые прямоугольные элементы. Для каждого элемента можно задать цвет фона и цвет символа. Кроме того , символ можно сделать мерцающим.

Модуль Crt позволяет работать не только со

всем экраном, но и выделять в нем прямоугольные окна . Любое окно задается своим левым верхним углом и правым нижним углом. Эти углы, как и любое положение на экране, задаются двумя координатами X и Y.

Приведенные ниже процедуры и функции расширяют возможности использования текстового режима экрана, находятся в стандартном модуле CRT.TPU и будут доступны в программе после объявления USES CRT.

1)Функция без параметров KeyPressed возвращает true, если на клавиатуре была нажата любая символьная клавиша, и false в противном случае. Не задерживает выполнение программы.

2)Функция без параметров ReadKey возвращает символ нажатой клавиши без эхо повтора на экране. Приостанавливает выполнений программы до нажатия на любую клавишу.

3) Функции без параметров WhereX, WhereY возвращают горизонтальную или вертикальную координату текущей позиции курсора относительно текущего окна.

4) Процедура без параметров CIrEol удаляет все символы оттекущей позиции курсора до конца строки без перемещения курсора.

5) Процедура без параметров DelLine удаляет строку, на которой

находится курсор, и перемещает все строки ниже этой строки на

строку вверх. Нижняя строка очищается.6) Процедура без параметров ClrScr очищает экран и помещает курсор в верхний левый угол.

7) Процедура Delay приостанавливает работу программы на

указанное число миллисекунд (t). DELAY(t);

8) Процедура GotoX,Y перемещает курсор в точку с заданными

координатами X,Y текстового экрана или окна, если оно задано.

GOTOXY(X,Y);

Коды цветов

|  |  |  |
| --- | --- | --- |
| Black | 0 | Черный |
| Blue | 1 | Синий |
| Green | 2 | Зеленый |
| Cyan | 3 | Голубой |
| Red | 4 | Красный |

|  |  |  |
| --- | --- | --- |
| Magenta | 5 | Фиолетовый |
| Brown | 6 | Коричневый |
| LightGray | 7 | Светло-серый |
| DarkGray | 8 | Темно-серый |
| LightBlue | 9 | Светло-синий |
| LightGreen | 10 | Светло-зеленый |
| LightCyan | 11 | Светло-голубой |
| LightRed | 12 | Розовый |
| LightMagenta | 13 | Светло-фиолетовый |
| Yellow | 14 | Желтый |
| White | 15 | Белый |
| Blink | 128 | Мерцание символов |

Цвета с кодами от 0 до 7 включительно можно использовать как для символов, так и для фона. Остальные цвета и код мерцания можно использовать только для символов.

Pascal — это императивный язык программирования, разработанный Никлаусом Виртом в 1970 в качестве языка обучения структурному программированию. Название языку дано в честь выдающегося французского математика, физика, литератора и философа Блеза Паскаля. Первоначально язык компилировался в байт-код, подобно языку Java.

Особенностями языка являются строгая типизация и наличие средств структурного (процедурного) программирования. Паскаль был одним из первых таких языков. По мнению Н. Вирта, язык должен способствовать дисциплинированию программирования, поэтому, наряду со строгой типизацией, в Паскале сведены к минимуму возможные синтаксические неоднозначности, а сам синтаксис интуитивно понятен даже при первом знакомстве с языком.

Переменные модуля Crt.

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| CheckBreak | Boolean | Разрешение анализа Ctrl+Break |
| CheckEof | Boolean | Разрешение помещения признака конца файла |
| CytcrSnow | Boolean | Разрешение подавления “снега” |
| DirectVideo | Boolean | Разрешение прямой адресации видеопамяти |
| LastMode | Word | Текущий текстовый режим |
| TextAttr | Byte | Текущие атрибуты текста |
| WindMin | Word | Координаты левого верхнего угла |
| WindMax | Word | Координаты правого нижнего угла окна |

Переменная CheckBreak управляет работой программы. Если эта переменная равна true ,то комбинация клавиш Ctrl+Break вызывает прекращение работы программы. Переменная CheckEof управляет размещением признака конца файла. Если она равна true, то комбинация клавиш Ctrl+Z помещает в файл признак конца файла. Прерменная СheckSnow ( если она равна true) разрешает подавление “снега” на экране для Видеоадаптеров CGA. Переменная DirectVideo ( если она равна true) позволяет непосредственную запись информации процедурами Write и Writeln в видеопамять. В переменной LastMode помещается значение текущего текстового режима, установленного процедурой TextMode .Переменная TextAttr содержит установленные атрибуты ( цвет фона и символов ) текста. В переменных WindMin и WindMax Находятся координаты окна.

Процедуры и функции модуля Crt.

|  |  |
| --- | --- |
| Имя процедуры (функции) | Назначение |
| procedure AssignCrt(var F:Text); | Связывает текстовый файл с устройством CRT. F- файловая переменная типа text, связывающаяся с устройством CRT. |
| procedure ClrEof; | Удаляет все символы от курсора (включительно) до конца строки,заполняя этот участок строки цветом фона. |

|  |  |
| --- | --- |
| procedure ClrScr; | Очищает текущее окно,заполняя его цветом фона и помещает курсор в его левый ерхний угол с координатами(1,1). |
| procedure Delay(Ms:word); | Задает задержку выполнения программы в Ms миллисекундах. |
| procedure DelLine; | Удаляет строку , в которой находится курсор. |
| procedure GotoXY(X,Y:Byte); | Перемещает курсор к элементу экрана с заданными координатами.  X,Y-координаты элемента экрана (отсчитываются от левого верхнего угла текущего окна) |
| procedure HighVideo; | Устанавливает высокую яркость символов (заменяет цвета 0-7 на 8-15) |
| procedure InsLine; | Вставляет пустую строку на экране в месте расположения курсора и заполняет ее цветом фона. |
| procedure LowVideo; | Устанавливает малую яркость символов ( заменяет цвета 8-15 на 0-7) |
| procedure NormVideo; | Устанавливает первоначальную яркость символов,выводимых далее на экран. |
| procedure NoSound; | Выключает источник звука. |
| procedure Sound(Hz:word); | Запускает источник звука с частотой Hz герц. |
| procedure TextBackGround(color:byte); | Задает цвет фона. Color-задаваемый цвет фона. |
| procedure TextColor(Color)Byte; | Задает цвет символов.Color-задаваемый цвет символов. |
| procedure TextMode( Mode:word); | Устанавливает текстовый режим,увеличивает текущее окно до целого экрана,устанавливает переменным DirectVideo и CheckSnow значение true.Помещает значение текущего режима в переменную LastMode,запускает процедуру NormVideo . Mode- требуемый текстовый режим. |
| procedure Window (X1,Y1,X2,Y2:Byte); | Задает размеры окна на экране и помещает курсор в левый верхний угол окна с координатами (1,1)  X1,Y1- координаты левого верхнего угла экрана  X2,Y2- координаты правого нижнего угла экрана.  Если хотя бы одна из координат недопустима, то процедура не выполняется. |
| Function KeyPressed:boolean; | Анализирует нажатие клавиши клавиатуры ( за исключением вспомогательных клавиш- Shift,Alt,NumLock и т.п.). Результат - true ,если клавиша на клавиатуре нажата и false - в противном случае. |
| function Readkey:char; | Считывает символ с клавиатуры и освобождает буфер клавиатуры от считанного символа. |
| function WhereX:Byte; | Возвращает текущую координату X курсора |
| function WhereY:byte: | Возвращает текущую координату Y курсора |

Пример программы, выполняющей вывод на экран дисплея меню следующего вида:

Выберите режим

Ввод и корректировка данных

Расчет свойств

Выбор состава

Графка

program menu;

uses Crt;label 1,2;

type

mas=array [1..4] of string[33];

const Stor :mas=(' Ввод и корректировка данных ', );

' Р а с ч е т с в о й с т в ',

' В ы б о р с о с т а в а ',

(' Г р а ф и к а ');

Выбор конкретного режима из этого меню осуществляется с помощью клавиш

При нажатии этих клавиш генерируется звук, продолжительностью 0.5 сек.

При этом строка,на которую падает выбор, выделяется цветом и фоном.

Выбор фиксируется с помощью клавиши ENTER

После этого выдается сообщение о выбранном режиме.

В программе используются возвращаемые коды клавиатуры:

|  |  |
| --- | --- |
| клавиша | код |
|  | 0 75 |
|  | 0 77 |
|  | 0 72 |
|  | 0 80 |
| enter | 13 |

Коды цветов

|  |  |  |
| --- | --- | --- |
| Имя | Значение | Назначение |
| Black | 0 | Черный |
| Blue | 1 | Синий |
| Green | 2 | Зеленый |
| Cyan | 3 | Голубой |
| Red | 4 | Красный |
| Magenta | 5 | Фиолетовый |
| Brown | 6 | Коричневый |
| LightGray | 7 | Светло-серый |
| DarkGray | 8 | Темно-серый |
| LightBlue | 9 | Светло-синий |
| LightGreen | 10 | Светло-зеленый |
| LightCyan | 11 | Светло-голубой |
| LightRed | 12 | Розовый |
| LightMagenta | 13 | Светло-фиолетовый |
| Yellow | 14 | Желтый |
| White | 15 | Белый |

|  |  |
| --- | --- |
| Комментарий до конца строки комментарии, могущие быть вложенными | (\*...\*) или {...} |
| Регулярное выражение идентификатора переменной | [A-Za-z\_][A-Za-z0-9\_]\* |
| Регулярное выражение идентификатора функции | [A-Za-z\_][A-Za-z0-9\_]\* |
| Присваивание значения переменной | := |
| Объявление переменной | variable: type |
| Группировка выражений | ( ... ) |
| Блок | begin ... end |
| Тождественное равенство | = |

|  |  |
| --- | --- |
| Тождественное неравенство | <> |
| Сравнение | < > <= >= |
| Определение функции | function f(p1: type1, p2: type2, ...): returntype |
| Вызов функции | f(a, b, ...) |
| Вызов функции без параметров | f |
| Последовательность | ; |
| Если - то | if condition then |
| Если - то - иначе | if condition then ... else ... |
| Бесконечный цикл | while true do ... |