Содержание

Введение

1. Основные символы

2. Этапы разработки программы

3. Алгоритмы и алгоритмизация

4. Структура программ на Паскале

5. Программирование основных конструкций языка

5.1 Паскаль

6. Задание

Список использованной литературы

## Введение

Турбо Паскаль появился на рынке программных продуктов в 1884 году и, несомненно, совершил революцию в программировании. До этого предпочтение отдавалось Бейсику - простому, дешевому и легко усвояемому. Паскаль же был аппаратно зависимым, дорогим и сложным в обращении. С появлением Турбо Паскаля положение в корне изменилось. Замечателен в Турбо Паскале не язык, который является лишь языком Паскаля, а программное окружение, в котором теперь можно использовать Паскаль и которое обеспечивает пользователю комфорт и высокую скорость работы. С успехом Турбо Паскаля на рынке программных продуктов связано также и развитие многочисленных пакетов, облегчающих применение Паскаля для самых разных целей.

Концепция Паскаля была разработана Н. Виртом примерно в 1970 году и Паскаль быстро получил широкое распространение благодаря легкости его изучения, наглядности составленных на нем текстов программ.

В середине 70-х годов была сделана попытка разработать международный стандарт на Паскаль. В результате в 1982 году появился стандарт ИСО 7185. Здесь следует различать этап 0 и заключительный этап 1. В Германии стандарт был опубликован в 1983 году в качестве стандарта ДИН 66256 на немецком языке.

После просуществовавшей сравнительно недолго и не получившей широкого распространения версии 1 в середине 1984 года появляется версия 2, распространение которой пошло стремительными темпами. К осени 1985 года появляется версия 3.0, отличающаяся от версии 2 следующими особенностями: компилятор и редактор стали работать существенно быстрее, появилась возможность передачи параметров в программу с помощи команды RUN, стал возможным вызов MS-DOS из программы, стала более удобной работа с файлами.

С начала 1988 года начинает распространяться версия 4.0. Здесь Турбо Паскаль представлен в совершенно новой форме. Не только становится еще более быстрым компилятор, но и появляется совершенно новое программное окружение. Существенно то, что компилятор стал встроенным, так что появилась возможность разбиения программ на части, компилируемые по отдельности. Поскольку каждый такой модуль имеет собственный сегмент кодов, программы могут занимать и более 64 Кбайт. Осенью 1988 года появилась версия 5.0 с еще более развитым программным окружением.

Фирма Borland / Inprise завершила линию продуктов Турбо Паскаль и перешла к выпуску системы визуальной разработки для Windows - Delphi. Несмотря на это, Турбо Паскаль сохраняет свое значение отличного языка для первого знакомства с миром "серьезного" программирования. Это связано как с его четкой логической структурой, так и с теми возможностями, которые позволяют использовать Турбо Паскаль для решения разнообразных задач. Среди них вычисления и обработка данных, компьютерная графика, работа со звуком, системно программирование. Турбо Паскаль позволяет применять приемы объектно-ориентированного программирования, которое стало одной из ведущих современных технологий программирования.

## 1. Основные символы

Основные символы языка - буквы, цифры и специальные символы - составляют его алфавит. Турбо паскаль включает следующий набор основных символов:

26 латинских строчных и 26 латинских прописных букв:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

a b c d e f g h i j k l m n o p q r s t u v w x y z

2) \_ подчеркивание

3) 10 цифр:

0 1 2 3 4 5 6 7 8 9

4) знаки операций:

+ - \* / = <> < > <= >=: = @

5) ограничители:

., ' () [] (.) { } (\* \*). .:;

6) спецификаторы:

^ # $

7) служебные (зарезервированные) слова:

|  |  |  |  |
| --- | --- | --- | --- |
| ABSOLUTE | EXPORTS | LIBRARY | SET |
| ASSEMBLER | EXTERNAL | MOD | SHL |
| AND | FAR | NAME | SHR |
| ARRAY | FILE | NIL | STRING |
| ASM | FOR | NEAR | THEN |
| ASSEMBLER | FORWARD | OF | UNIT |
| BEGIN | NOT | FUNCTION | CASE |
| CONST | GOTO | IF | OR |
| CONSTRUCTOR | IMPLEMENTATION | USES | UNTIL |
| PACKED | VAR | IN | INDEX |
| DESTRUCTOR | PRIVATE | DIV | PROCEDURE |
| END | INTERRUPT | TYPE | VIRTUAL |
| ELSE | LABEL | EXPORT | RESIDENT |
| INTERFACE | TO | OBJECT | REPEAT |
| XOR | RECORD | DOWNTO | PROGRAM |
| PUBLIC | INLINE | DO | WHILE |
| INHERITED | WITH |  |  |

Кроме перечисленных, в набор основных символов входит пробел. Пробелы нельзя использовать внутри сдвоенных символов и зарезервированных слов.

## 2. Этапы разработки программы

В процессе создания любой программы, будь то небольшая учебная программа, предназначенная для демонстрации учителю информатики, или серьезный проект, над которым работают десятки программистов, можно выделить несколько этапов. Затраты труда и времени на их выполнение различны, различаются эти затраты и для различных программ. Некоторые из этапов могут быть опущены или пройдены "незаметно", однако анализ процесса разработки приводит к выводу о том, что почти всегда, явно или не явно, приходится проходить следующие этапы разработки программы:

постановка задачи;

анализ, формальное описание задачи, выбор модели;

выбор и разработка алгоритма решения задачи;

проектирование общей структуры программы;

кодирование;

отладка и верификация программы;

получение результата;

публикация или передача заказчику результата работы;

сопровождение программы.

Описание каждого из перечисленных этапов:

**Постановка задачи** выполняется заказчиком, в качестве которого может выступать внешняя организация, организация в которой работает программист, начальник программиста, преподаватель, сам программист. На этом этапе задача, которую необходимо решить посредством составления программы для компьютера, формулируется на естественном языке.

**Анализ** задачи включает определение входных и выходных данных, выявление возможных ограничений на их значения и обычно завершается формализованным описанием задачи, которое часто предполагает её математическую формулировку.

**Выбор и разработка алгоритма и численного метода решения задачи** имеют важнейшее значение для успешной работы над программой. Тщательно проработанный алгоритм решения задачи - необходимое условие эффективной работы по составлению программы.

**Проектирование общей структуры программы.** На этом этапе происходит "архитектурная" проработка проекта. Определяются те части алгоритма, которые целесообразно оформить в виде подпрограмм, модулей. Определяется и способ хранения информации - в виде набора простых переменных, массивов или других структур.

**Кодирование** - это запись алгоритма на языке программирования. Если алгоритм решения задачи, структура программы и структура данных тщательно продуманы и аккуратно записаны, затраты времени на кодирование уменьшаются, а вероятность ошибок на этом этапе снижаются.

**Отладка и верификация программы** представляют собой очень важную часть процесса разработки программы. Отладка заключается в устранении ошибок программирования, ошибок перевода алгоритма на язык программирования. Верификация - доказательство того, что программа работает "правильно", дает правильный результат.

**Получение результата** - сравнить результаты наблюдений и результаты компьютера. Результат может отличаться от требуемого. В этом случае, возможно, придется изменить саму модель, сделав ее более реалистичной.

**Публикация или передача заказчику результата работы -** момент рождения качественной программы. В научных исследования значение имеют результаты моделирования, которые публикуются в научных журналах. В нашем случае это сама программа, которая передается заказчику для дальнейшей эксплуатации или выкладывания на ftp - сервер для свободного распространения.

**Сопровождение программы -** предполагают консультации заказчику по работе программы, устранение замеченных в процессе ее эксплуатации недостатков, обучение пользователей работе с программой.

## 3. Алгоритмы и алгоритмизация

**Алгоритмизация** - составление пошагового описание процесса решения задачи.

**Алгоритм -** конечный набор правил, расположенных в логическом порядке, позволяющий исполнителю решать любую конкретную задачу из некоторого класса однотипных задач.

Алгоритм должен удовлетворять определенным требованиям. Принято выделять следующие семь:

Наличие ввода исходных данных.

Наличие вывода исходных данных.

Однозначность.

Общность - алгоритм предназначен для решения некоторого класса задач.

Корректность - алгоритм должен давать правильное решение задачи.

Конечность - решение задачи должно быть получено за конечное число шагов.

Эффективность - для решения задачи должны использоваться ограниченные ресурсы компьютера (процессорное время, объем оперативной памяти).

Для разработки структуры программы удобнее пользоваться записью алгоритма в виде блок-схемы. Для изображения основных алгоритмических структур и блоков на блок-схемах используют специальные графические символы:

## 4. Структура программ на Паскале

Начало / конец алгоритма

Передача управления

Ввод / вывод данных

Начало (заголовок) цикла

Блок вычислений

Конец цикла

Ветвление

нет

да

Программа на языке Паскаль состоит из заголовка, разделов описаний и раздела операторов.

Program имя программы;

Uses подключаемые модули;

Блок описания:

Меток

Констант

Типов

Переменных

Процедур

Функций

Begin

Тело программы.

Исполняемая часть.

End.

Строка программы может иметь максимальную длину 127 символов. Если строка будет длиннее, ее часть, выходящая за пределы 127 символов, будет компилятором игнорироваться.

Комментарии заключаются в фигурные скобки {……}.

## 5. Программирование основных конструкций языка

## 5.1 Паскаль

**Линейная программа**

Простейшей алгоритмической структурой является линейная последовательность операций, которые выполняются по очереди и именно в том порядке, в котором они записаны. Линейные алгоритмы и линейные программы обычно предназначены для решения относительно простых задач, в которых не предусмотрен выбор из нескольких возможных альтернатив или циклических повторений каких-либо операций.

**Ветвления**

В Паскале имеются две реализации одной из основных алгоритмических структур - ветвления. Это условный оператор и оператор выбора. Условный оператор имеет вид:

If a>b then Writeln (‘значение а больше’);

Полная версия условного оператора имеет вид:

If a > b then writeln (‘значение а больше’)

else writeln (‘значение b больше’);

**Циклы**

Одной из важнейших алгоритмических структур является цикл. Цикл представляет собой последовательность операторов, которая выполняется неоднократно. В программах, связанных с обработкой данных или вычислениями, часто приходиться выполнять циклические повторяющиеся действия.

В языке программирования Паскаль имеется три разновидности цикла - цикл со счетчиком, цикл с предусловием и цикл с постусловием.

Операторы цикла:

Оператор While

Оператор Repeat

Оператор For

**Оператор While:**

При использовании оператора while условие повторения цикла запрашивается в начале цикла:

**while** выражение **do** оператор

Выражение после while должно быть логическим выражением. Оператор после do выполняется, если логическое выражение имеет значение true. Если после do, т.е. в цикле, должны быть выполнены несколько операторов, их следует поместить в операторные скобки begin и end. Если логическое выражение в начале имеет значение false, цикл вообще не выполняется.

**Оператор Repeat:**

**repeat** оператор **until** выражение

Выражение после repeat должно быть логическим выражением. Выражение после repeat выполняются до тех пор, пока логическое выражение после until не примет значение true. Итак, построенный с помощью оператора repeat цикл выполняется по крайней мере один раз.

**Оператор For:**

**for** переменная: **=** выражение **to** выражение **do** оператор

Переменная цикла и оба выражения должны иметь одинаковый тип, который должен быть порядковым. Первое выражение является начальным значением, второе - конечным. При выполнении to начальное значение увеличивается каждый раз, пока не достигнет конечного значения.

**Простые типы данных**

Все переменные, используемые в программе, должны быть перечислены в разделе описания переменных.

var a, b: integer;

c: real;

Паскаль - это гибкий язык, в котором имеется большое число различных типов. Все имеющиеся в Паскале типы принято делить на группы. Типы, принадлежащие одной группе, имеют сходство. Прежде всего, выделяют простые и структурные типы. Простые типы, в свою очередь, подразделяют на порядковые и вещественные типы.

В таблице приведено описание простых типов:

|  |  |  |
| --- | --- | --- |
| Название | Идентификатор | Множество допустимых значений |
| Порядковый | | |
| Короткий целый | Shortint | -128. .127 |
| Байтовый | Byte | 0. .255 |
| Слово | Word | 0. .65535 |
| Целый | Integer | -32768. .32767 |
| Длинный целый | Longint | -2147483. .2147483647 |
| Символьный | Char | Символы из расширенного набора символов кода ASCII |
| Булев | Boolean | True, False |
| Вещественный | | |
| Вещественный | Real | -1.7x1038. - 2.9x10-39 |
| С одинарной точностью | Single | 2.9x10-39. .1.7x1038 |
| С двойной точностью | Double | -3.4x1038. - 1.5x10-45 |
| Сложный | Comp | -263+1. .263-1 |

Допустимые значения порядковых типов представляют собой множество, состоящее из конечного числа элементов. В этом множестве есть первый и последний элементы. Кроме того, каждый элемент порядкового типа имеет предшествующий ему и следующий за ним элементы.

В Паскале имеется пять вещественных типов. Вещественные типы различаются диапазоном и точностью связанных с ними значений. Действия над типами с одинарной, двойной и повышенной точностью, а также над сложным типом могут выполняться только при наличии математического сопроцессора.

**Процедуры и функции**

Подпрограмма - это последовательность операторов, которые определены и записаны только в одном месте программы, однако их можно вызвать для выполнения из одной или нескольких точек программы. Каждая подпрограмма определяется уникальным именем. В языке Паскаль существуют два типа подпрограмм - процедуры и функции.

Процедура и функция - это именованная последовательность описаний и операторов. При использовании процедур или функций Паскаль - программа должна содержать текст процедуры или функции и обращение к процедуре или функции. Тексты процедур и функций помещаются в раздел описаний процедур и функций.

Процедура может содержать такие - же разделы описаний, что и Паскаль - программа, а именно: разделы описания модулей, меток, констант, типов, переменных, процедур и функций.

## 6. Задание

Составить программу вывода на экран всех натуральных чисел, не превосходящих N и не делящихся на каждую из своих цифр. Описать соответствующую функцию, получающую из основной программы в качестве параметра натуральное число и возвращает TRUE, если оно удовлетворяет указанному условию.

**Блок - схема программы**

Начало

Ввод n

I=1 to n

viborka(i)

Вывод i

Конец

**Блок - схема функции { viborka }**

f:=true

k:=n

Начало

Конец

m < > 0

n mod m = 0

f:=false

k < > 0

m:=k mod 10

viborka:=f

k:=k div 10

да

нет

**Текст программы**

program nat\_chisla;

uses crt;

function viborka (var n: integer): boolean;

var f: boolean;

k: integer;

m: integer;

begin

f: =true;

k: =n;

while k<>0 do begin

m: =k mod 10;

if m <> 0 then

if n mod m = 0 then begin

f: =false;

break;

end;

k: =k div 10;

end;

viborka: =f;

end;

var i,n: integer;

begin

clrscr;

writeln ('Введите число');

readln (n);

for i: =1 to n do

if viborka (i) then writeln (i);

readln;

end.

## Список использованной литературы

1. Turbo Pascal: практикум. - СПб.: Питер, 2002. - 256 с.: ил.
2. Turbo Pascal / 2-е изд., перераб., - Вологда: МП "МИК", 1991. - 342с.