# 

# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ](#_Toc201488321) 4

ОСНОВНАЯ ЧАСТЬ

[1.1 Строковый тип данных в языке Pascal](#_Toc201488323) 6

[1.2 Тип string и стандартный тип char](#_Toc201488324) 7

[1.3 Функция Copy](#_Toc201488325) 9

[1.4 Функция Concat](#_Toc201488326) 9

[1.5 Функция Length](#_Toc201488327) 9

[1.6 Процедура Delete](#_Toc201488328) 10

1. 7 Str(x, S)………………………………………………………………………10

1.8 Val(S, x, kod)………………………………………………………………....10

[2.1 Постановка задачи](#_Toc201488330) 11

2.2 Блок- схема алгоритма решения задач………………..……………………12

2.3 Особенности паскаля……………………………………………………..14

[2.4 Программа](#_Toc201488332) 14

2.5 Описание работы программы………………………………………………17

[2.6 Пояснение работы операторов используемых в прграмме](#_Toc201488333) 20

[2.7 Примеры работы прграммы](#_Toc201488334) 23

[ЗАКЛЮЧЕНИЕ](#_Toc201488336) 24

[СПИСОК ЛИТЕРАТУРЫ 2](#_Toc201488337)5

# Приложение……………………………………………………………………26

Инструкция пользователя……………………………………………………….26

Программа на контрольном примере…………………………………………..28

# ВВЕДЕНИЕ

Программа - это детальное и законченное описание алгоритма средствами языка программирования. Исполнителем программы является компьютер. Для выполнения компьютером программа должна быть представлена в машинном коде - последовательности чисел, понимаемых процессором. Написать программу в машинных кодах вручную достаточно сложно. Поэтому сегодня практически все программы создаются с помощью языков программирования, которые по синтаксису и семантике приближены к естественному человеческому языку. Это снижает трудоемкость программирования. Однако, текст программы, записанный с помощью языка программирования, должен быть преобразован в машинный код. Эта операция выполняется автоматически с помощью специальной служебной программы, называемой транслятором.

Трансляторы делятся на два типа: интерпретаторы и компиляторы.

Интерпретатор переводит в машинный код и выполняет очередной оператор (команду) программы. Если команда повторяется, то интерпретатор рассматривает ее как встреченную впервые.

Компилятор переводит в машинный код исходный текст программы целиком. Поэтому достоинство компиляторов - быстродействие и автономность получаемых программ. Достоинство интерпретаторов - их компактность, возможность остановить в любой момент выполнение программы, выполнить различные преобразования данных и продолжить работу программы.

Примерами служебных программ - интерпретаторов являются GW Basic, Лого, школьный алгоритмический язык, многие языки программирования баз данных. Компиляторами являются Turbo Pascal, С++, Delphi.

В общем случае для создания программ нужно иметь следующие компоненты:

* текстовый редактор - для набора исходного текста программы;
* компилятор - для перевода текста программы в машинный код;
* редактор связей - для сборки нескольких откомпилированных модулей в одну программу;
* библиотеки функций - для подключения стандартных функций к программе.

Современные системы программирования включают в себя все указанные компоненты и называются интегрированными системами.

Исходный текст программы можно получить без записи его вручную в текстовом редакторе. Существуют системы визуального программирования - RAD-среды (Rapid Application Development), которые, не исключая возможности записи программы вручную, позволяют создавать текст программы автоматически, путем манипуляций со стандартными элементами управления, включенными в RAD-среду. Поэтому для RAD-среды понятие «программирование» часто заменяют понятием «проектирование».

По способу разработки программ можно выделить два подхода:

* процедурное программирование - это программирование, при котором выполнение команд программы определяется их последовательностью, командами перехода, цикла или обращениями к процедурам;
* объектно-ориентированное программирование - программирование, при котором формируются программные объекты, имеющие набор свойств, обладающие набором методов и способные реагировать на события, возникающие как во внешней среде, так и в самом объекте (нажатие мыши, срабатывание таймера, превышение числовой границы и т.д.).

Таким образом, выполнение той или иной части программы зависит от событий в программной системе.

Объектно-ориентированное программирование (ООП) не исключает, а охватывает технологию процедурного программирования.

Из универсальных языков программирования наиболее популярны следующие: Basic; Pascal; C++; Java.

Для языка Basic существует много версий, реализованных и как интерпретаторы и как компиляторы. В России Basic традиционно используется в курсе информатики средней школы. Среда визуального программирования Microsoft Visual Basic используется как программная поддержка приложений MS Office.

Язык Pascal является компилируемым и широко используется как среда для обучения программированию в ВУЗах. RAD-средой, наследующей его основные свойства, является среда Borland Delphi.

Для языка C++ RAD-средой является Borland C++ Builder. Этот компилируемый язык часто используется для разработки программных приложений, в которых необходимо обеспечить быстродействие и экономичность программы.

## 1.1 Строковый тип данных в языке Pascal

Познакомимся с типом данных, который относится к числу структурированных. Это строковый тип данных (строка). Строка - это последовательность символов. Каждый символ занимает 1 байт памяти (код ASCII). Количество символов в строке называется ее длиной. Длина строки может находиться в диапазоне от 0 до 255. Строковые величины могут быть константами и переменными. Особенностью строки в Turbo Pascal является то, что с ней можно работать как с массивом символов, с одной стороны, и как с единым объектом, - с другой. За счет этого обработка строк достаточно гибка и удобна. Строковая константа есть последовательность символов, заключенная в апострофы. Например: «это строковая константа», «272». Строковая переменная описывается в разделе описания переменных следующим образом:

Var <идентификатор> : string[<максимальная длина строки>];

Например:

Var Name : string [20].

Параметр длины может и не указываться в описании. В таком случае подразумевается, что он равен максимальной величине - 255. Например: Var slovo : string.

Строковая переменная занимает в памяти на 1 байт больше, чем указанная в описании длина. Дело в том, что один (нулевой) байт содержит значение текущей длины строки. Если строковой переменной не присвоено никакого значения, то ее текущая длина равна нулю. По мере заполнения строки символами ее текущая длина возрастает, но она не должна превышать максимальной по описанию величины.

Символы внутри строки индексируются (нумеруются) от единицы. Каждый отдельный символ идентифицируется именем строки с индексом, заключенным в квадратные скобки. Например: N[5], S[i], slovo[k+l]. Индекс может быть положительной константой, переменной, выражением целого типа. Значение индекса не должно выходить за границы описания.[9]

## 

## 1.2 Тип string и стандартный тип char

Тип string и стандартный тип char совместимы. Строки и символы могут употребляться в одних и тех же выражениях.

Строковые выражения строятся из строковых констант, переменных, функций и знаков операций. Над строковыми данными допустимы операции сцепления и операции отношения.

Операция сцепления (конкатенации) (+) применяется для соединения нескольких строк в одну результирующую строку. Сцеплять можно как строковые константы, так и переменные.

Пример. Объединение двух строк.

var

Str, Strl, Str2:string[80] ;

Begin

. . .

Str1 := 'Turbo ';

Str2 := 'Pascal';

Str := Str1 + Str2; {в переменной Str - 'Turbo Pascal'}

. . .

end.

Сравнивать можно строки разной длины. Сравнение осуществляется слева направо в соответствии с ASCII-кодами соответствующих символов. Считается, что отсутствующие символы в более короткой строке имеют код меньше кода любого действительного символа. Например, 'XS' больше, чем 'Х'.

Пример. Проверить, является ли введенная совокупность символов именем месяца на русском языке.

program EXAMPLE11;

const

Instance: array[l..l2] of string[10]= ('ЯНВАРЬ', 'ФЕВРАЛЬ', 'МАРТ ', 'АПРЕЛЬ' ' МАЙ', 'ИЮНЬ', ' ИЮЛЬ', 'АВГУСТ ', 'СЕНТЯБРЬ', 'ОКТЯБРЬ ', ' НОЯБРЬ', 'ДЕКАБРЬ');

Month: Boolean = False;

var

Str: string[10];

i: Integer;

begin

Writeln('Введите заглавные символы: ');

ReadLn(Str);

for i := 1 to 12 do

if Str = Instance[i] then Month:=True;

if Month then

WriteLn('Введено имя месяца')

else

WriteLn('Введено не имя месяца')

end .

Фактически строка N символов представляет собой массив из N+1 символа:

string[N] = array[0..N] of Char.

Нулевой символ предназначен для указания используемого количества символов строки и может изменяться от символа с кодом 0 до символа с кодом N. С ним можно работать как и с остальными символами строки (записывать и читать его значение и т. д.), но не забывая о его основном предназначении. [9]

## 

## 1.3 Функция Copy

Функция Copy(S, Pozition, N) выделяет из строки S подстроку длиной N символов, начиная с позиции Pozition. Здесь N и Pozition – целочисленные выражения.

Пример:

|  |  |  |
| --- | --- | --- |
| Значение S | Выражение | Результат |
| ‘Мама мыла раму’ | Copy(S, 6, 4) | ‘мыла’ |
| ‘Маша ела кашу’ | Copy(S, 1, 8) | ‘Маша ела’ |

[10]

## 

## 1.4 Функция Concat

Функция Concat(S1, S2, …, SN) выполняет сцепление (конкатенацию) строк S1, S2, …, SN в одну строку.

Пример:

|  |  |
| --- | --- |
| Выражение | Результат |
| Concat('Маша ', 'ела ', 'кашу') | 'Маша ела кашу' |

[9]

## 

## 1.5 Функция Length

Функция Length(S) - определяет текущую длину строки S. Результат - значение целого типа.

Пример:

|  |  |  |
| --- | --- | --- |
| Значение S | Выражение | Результат |
| 'test-5' | Length(S) | 6 |
| '(A+B)\*C' | Length(S) | 7 |

[9]

## 

## 1.6 Процедура Delete

Процедура Delete(S, Poz, N) - удаление N символов из строки S, начиная с позиции Poz.

Пример:

|  |  |  |
| --- | --- | --- |
| Исходное значение S | Оператор | Конечное значение S |
| 'abcdefg' | Delete(S, 3, 2) | 'abefg' |
| 'abcdefg' | Delete(S, 2, 6) | 'a' |

[10]

**Процедуры преобразования типов в Паскале.**

**1.7 Str(x, S)**

Str(x, S) преобразует число x в строковый формат. Здесь x – любое числовое выражение, S – строковая переменная. В процедуре есть возможность задавать формат числа x. Например, str(x: 8: 3, S), где 8 – общее число знаков в числе x, а 3 – число знаков после запятой.Оператор процедуры [10]

**1.8 Val(S, x, kod)**

Val(S, x, kod) преобразует строку символов S в число x. Здесь S – строковое выражение, x – числовая переменная (именно туда будет помещен результат), kod – целочисленная переменная (типа integer), которая равна номеру позиции в строке S, начиная с которой произошла ошибка преобразования, если преобразование прошло без ошибок, то переменная kod равна 0.[10]

**2.1 Постановка задачи**

Написать программу – телеграф, который принимает от пользователя сообщения и выводит его на экран в виде последовательности точек и тире. Вывод точек и тире можно сортировать звуковым сигналом соответствующей длительности

Азбука Морзе для букв русского алфавита приведена ниже:

|  |  |  |  |
| --- | --- | --- | --- |
| А .- | Б -… | В .- - | Г - - . |
| Д - .. | Е | Ж …- | З - - .. |
| И .. | Й . - - - | К - . - | Л . - .. |
| М - - | Н - . | О - - - | П . - - . |
| Р . - . | С … | Т - | У .. - |
| Ф .. - . | Х …. | Ц - . - . | Ч - . - - . |
| Ш - - - - | Щ - - . - | Ъ - .. - | Ы - - - . |
| Ь - .. - | Э .. - . | Ю .. - - | Я . - . - |

**2.2 Блок- схема алгоритма решения задач.**

начало

сlrser

mes

Soud[1000]

repeat

J=1

Sim=morse[…]

Case j of

Delay[L1]

Delay[L2]

Nosound

Delay[L5]

J=j+1

Delay[L3]

Конец

**2.3 Особенности паскаля.**

Язык Паскаль относится к процедурно-ориентированным языкам высокого уровня. Разработан американским ученым Никласом Виртом в 1971г. в качестве языка для обучения программированию. Базой при разработке явился язык Алгол.

Достоинства языка Паскаль:

1) относительная простота (т.к. разрабатывался с целью обучения программированию);

2) идеология языка Паскаль близка к современным методикам и технологиям программирования, в частности, к структурному программированию и нисходящему проектированию (метод пошаговой детализации) программ. Паскаль может использоваться для записи программы на различных уровнях ее детализации, не прибегая к помощи схем алгоритмов;

3) гибкие возможности в отношении используемых структур данных;

4) высокая эффективность программ;

5) наличие средств повышения надежности программ, включающих контроль правильности использования данных различных типов и программных элементов на этапах трансляции, редактирования и выполнения.

В связи с этим язык Паскаль в настоящее время находит самое широкое распространение для решения большого круга разнообразных практических и научных задач.[6]

**2.4 Программа**

{Телеграф-передача сообщений при помощи азбуки Морзе}

Program a;

uses crt;

const

morse : array [128..159] of string[4]=('.- ','-...','.---','--. ',

'-.. ','. ','...-','--..',

'.. ','.---','-.- ','.-..',

'-- ','-. ','--- ','.--.',

'.-. ','... ','- ','..- ',

'..-.','....','-.-.','---.',

'----','--.-','-..-','-.--',

'-..-','..-.','..--','.-.-'

);

Tone=100;

L1=50;

L2=100;

L3=50;

L4=100;

L5=150;

Var

mes : string;

sim : string[4];

znak : string[1];

I, j : integer;

begin

clrscr;

writeln (‘\*\*\* телеграф \*\*\*’);

writeln (‘Введите сообщение, которое надо передать ‘);

writeln (‘(используйте только большие русские буквы )‘);

write (‘ => ‘);

readln (mes);

for i=1 to Length (mes) do

begin

if (mes [i] > = ‘A’) and (mes[i] <= ’Я’) then

begin

sim := morse [ord (mes[i] :=1];

repeat

if (sim[j]= ‘-’ ) or (sim[j]= ‘.’ ) then

begin

write (sim [j]);

sound (1000);

case sim[j] of

‘.’: Delay (L1);

‘-’: Delay (L2);

end;

Nosound;

Delay (L3);

end;

j:=j+1;

until ((sim[j]=’ ‘ ) or (j>4));

Delay (L4);

End;

else

if mes [i]= ‘ ‘ then

begin

write (‘ ‘);

Delay (L5);

end;

end;

writeln

writeln (‘ Сообщение передано! ‘);

writeln (‘ Для завершения работы с программой нажмите ‘, ‘<Enter>’);

readln;

end.

**2.5 Описание работы программы.**

Program a;

Uses crt;

Const

Morse: array[128..159] of [4]= (

‘.-‘ , ‘-…’ , ‘.---‘ , ‘--. ‘ , {А, Б, В, Г}

‘-.. ‘ , ‘. ‘,’…-‘ , ‘--..’ , {Д, Е, Ж, З}

{И, Й, К, Л,}

{М, Н, О, П}

{Р, С, Т, У}

{Ф, Х, Ц, Ч}

{Ш, Щ, Ъ, Ы}

{Ь, Э, Ю, Я}

); {Параметры передачи}

TONE=100; {частота сигнала (Гц)}

L1=50; {длительность (мс) «точки»}

L2=100; {длительность (мс) «тире»}

L3=50; {пауза (мс) между точками и тире одной буквы}

L4=100; {пауза (мс) между буквами}

L5=150; {пауза (мс) между словами}

Var

mes : string; {сообщение}

sim : string[4]; {символ в кодировке Морзе – последовательность точек и тире}

znak : string[1]; {“ передаваемый“ знак – тире или точка}

I, j : integer; {номер символа и знака}

begin

clrscr; { Очистка экрана}

writeln (‘\*\*\* телеграф \*\*\*’);

writeln (‘Введите сообщение, которое надо передать ‘);

writeln (‘(используйте только большие русские буквы )‘);

write (‘ => ‘);

readln (mes);{Ввести сообщение, которое нужно передать}

for i=1 to Length (mes) do

begin

if (mes [i] > = ‘A’) and (mes[i] <= ’Я’) then { выставляем диапазон, А и Я даны в константах, т..е. определение идет автоматически}

begin

{определим код очередной буквы (функция ord) сообщения и получим из таблицы кодировки соответствующий элемент массива – последовательность точек и тире}

sim := morse [ord (mes[i

j :=1;

repeat

if (sim[j]= ‘-’ ) or (sim[j]= ‘.’ ) then { В Сим передалось значение состоящее из точек и тире, выбираем единичную позицию благодаря переменной J, и определяем ее. Она может быть тире, точка либо пробел. Задаем ограничения, т.е. узнаем, будет ли присутствовать тире или точка, ведь если они есть, значит есть и буква. Если пробел \_ -буквы или символа первоначального сообщения тоже нету.}

begin

write (sim [j]);{Если есть точка или тире – выведется на печать}

sound (1000);

case sim[j] of{ Оператор выбора условия. Если СИМ равно точке – задержка 5, если тире – задержеа 100}

‘.’: Delay (L1);

‘-’: Delay (L2);

end;

Nosound; {Функция, определяющая окончание звука.}

Delay (L3);

end;

j:=j+1;{Буква А например в азбуке морзе имеет вид « .-.-« Задавая выше Jравное 1 мы определяли первую позицию для числа А . В нашем случае первая позиция это точка. Затем каждый раз мы прибавляем единицу. Во втором случае это тире. В третьем точка. В четвертом тире. Дальше не пойдет, тк. А состоит из 4 символов }

until ((sim[j]=’ ‘ ) or (j>4));{Этот оператор будет выполняться так долго пока условия((sim[j]=’ ‘ ) or (j>4) будут ложными. Когда они станут истинными, то оператор сразу прекращает работу. }

Delay (L4); {пауза между буквами}

end

else

if mes [i]= ‘ ‘ then {пробел между символами}

begin

write (‘ ‘); {пробел между словами сообщения}

Delay (L5);

end;

end;

writeln

writeln (‘ Сообщение передано! ‘);

writeln (‘ Для завершения работы с программой нажмите ‘, ‘<Enter>’);

readln;

end.

**2.6 Пояснение работы операторов которые применяются в этой программе в этой программе.**

В данной программе были использованы различные функции, операторы, процедуры. Такие как:

**Составной оператор Begin и End**

Составной оператор представляет собой последовательность некоторых операторов, выполняющихся в том порядке, в каком они представлены в тексте программы.

**Program**<Имя программы>;  
<Раздел описаний>  
**Begin**  
<Тело программы>  
**End.**

Зарезервированные слова *Begin* и *End* являются так называемыми операторными скобками, в которые заключены операторы, входящие в составной оператор. Когда в зависимости от определённого условия, требуется обеспечить последовательное выполнение некоторого набора операторов, без составного оператора не обойтись.[5]

**Условный оператор If и Case**

Если в программе, в зависимости от некоторого условия, требуется выбрать тот или иной вариант действий, можно воспользоваться одним из условных операторов. В Turbo Pascal предусмотрены два условных оператора- *If* и *Case*.

Работа *Case* такова. Сначала вычисляется значение селектора, следующего за словом *Case*. Затем выполняется оператор с константой выбора, равной значению селектора. Если ни одна из констант не равна текущему значению селектора, то исполняется оператор, стоящий после слова *Еlse*. Часть *Еlse* <оператор> в тексте программы можно опустить. Тогда, если среди констант селектора нужное значение отсутствует, выполнение оператора *Case* ни к чему не приведет.

**Case <**выбор селектора**> of**

<метка 1>: <оператор 1>;

<метка n>:<оператор n>;

**else** <оператор>;

**end;**

Если необходимо выбрать одно из возможных действий в зависимости от некоторого условия, применяют условный оператор *if.. then..else*. Во многих программах, в зависимости от некоторого условия, часто требуется выполнить не одно, а последовательность действий. Однако оператор *If* в Turbo Pascal обеспечивает выполнение единственного оператора, присутствующего после зарезервированного слова *Then* или *Else*.

**If** <условие> **then** <оператор 1> **else <**оператор 2**>; [2]**

**Оператор Repeat until.**

Оператор цикла, *Repeat,* известный как оператор цикла с постусловием, имеет вид:

**Repeat**

<оператор #1>;  
<оператор #2>;  
<оператор #3>;  
. . .

**Until** <условие>;

Здесь не требуется использование составного оператора, потому, что сами слова Repeat и Until являются операторными скобками.

Этот вид цикла отличается от других в основном тем, что проверка условия повторения тела цикла находится не перед ним, а после. Поэтому цикл "До" называют циклом "с постусловием", а "Пока" - "с предусловием".

Повторное выполнение тела цикла происходит не тогда, когда условие справедливо, а как раз тогда, когда оно ложно. Поэтому цикл и получил свое название (выполнять тело цикла до выполнения соответствующего условия).

Когда условие цикла изначально истинно, тело цикла все равно будет выполнено хотя бы один раз. Именно это отличие "до" от "пока" привело к тому, что в программировании они не подменяют друг друга, а используются для решения задач, к которым они более подходят.[3]

**Оператор For to do.**

В данном случае параметром будет являться целочисленная переменная, которая будет изменяться на единицу при каждой итерации цикла. Таким образом, задав начальное и конечное значения для такой переменной, можно точно установить количество выполнений тела цикла.

Форматов у этого вида цикла предусмотрено два:  
 **For** <И.П.>:=<Н.З.> **To** <К.З.> **Do** <оператор>;  
 **For** <И.П.>:=<Н.З.> **Downto** <К.З.> **Do** <оператор>;

Здесь И.П. - имя переменной-параметра, Н.З. - его начальное значение, К.З. - соответственно конечное значение параметра. В качестве начального и конечного значений.

Выполняется этот цикл по следующему алгоритму:

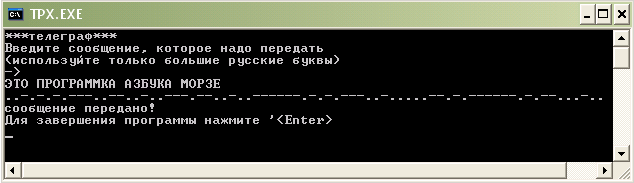
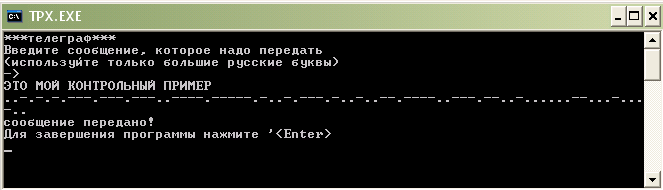
1. переменной-параметру присваивается начальное значение;

2. выполняется тело цикла;

3. переменная-параметр автоматически увеличивается на 1 (в первом случае формата);

4. если параметр превышает конечное значение, то происходит выход из цикла, иначе - переход к пункту 2. [3]

**2.7 Примеры работы программы.**



Принцип работы: вводим текст на русском языке, а получаем тот же текст нописаный азбукой Морзе.

**Заключение**

Программа написана на языке Turbo Pascal. Этот язык представляет мощное инструментальное средство написания прикладных программ, кото­рое можно использовать в реальной работе. Turbo Pascal является средой разработки Программного Обеспечения фирмы Borland. ПО в среде Turbo Pascal реализуется на процедурно- ориентированном языке Паскаль. Расши­рение языка Turbo Pascal предоставляет все возможности объектно-ориенти­рованного программирования: наибольшую структурность, абстрактность, модульность, встроенные непосредственно в язык.

В Delphi упор ставится на написание ПО под Windows, много возможностей и легко запутаться. Программы, разработанные на TURBO Pascal, компактны, и занимают мало места на жестком диске. Современные технологии программирования, такие как Delphi, в буквальном смысле слова ”стоят на плечах” TP.

Неизвестно что и когда может пригодиться в жизни. Азбука Морзе широко используется на международном уровне.

Ознакомившись с программой, пользователь, возможно, заинтересуется данной системой передачи сигналов, и в дальнейшем захочет знать и применять код Морзе.

# 

# СПИСОК ЛИТЕРАТУРЫ

1. Turbo Pascal для студентов и школьников - СПб.: БЧВ- Петербург, 2004. - 352с.: ил. (Г.Г. Рапаков, С.Ю. Ржеуцкая)

2. Turbo Pascal для школьников. Версия 7.0. - М.: Финансы и статистика, 1996. - 446с. (Попов В.Б.)

3. О.А. Меженный: Turbo Pascal. Учитесь программировать. Санкт- Петербург, 2001 год.

4. Информатика, 7- 11 класс. Киев, 2004. Гаевский А.Ю.

5. Turbo Pascal: Учись программировать. 2002. – 448с.: ил.(Меженный О.А.)

6. Грызлов С.В., Грызлова Ю.А. ”Турбо Паскаль 7.0 “ 1999г. Москва 400стр.

7. Информатика и вычислительная техника, ред. В.Н. Ларионова, «ВШ», Москва,1992

8.Фаронов В.В. Турбо Паскаль 7.0. Начальный курс. Учебное пособие.-М:"Нолидж",1997.-616 с, ил.

9. <http://www.5byte.ru/tp7pub/0017.php>

10. http://www.debryansk.ru/~lpsch/uchitel/konspekt/pasc...

**Приложение**

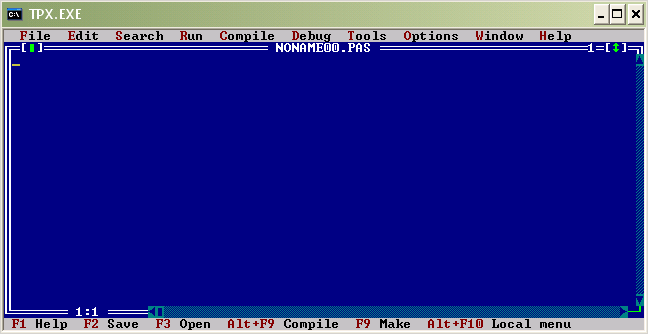
**Инструкция пользователя**

Работа в среде программирования Turbo Pascal начинается с загрузки её в память компьютера. Система Турбо Паскаль довольно значительна по объему. Она поставляется на нескольких дистрибутивных дискетах и устанавливается на жесткий диск. При развертывании системы на жестком диске обычно создается отдельный каталог с именем *ТР* (или *PAS, TURBOPAS, PASCAL* и т.п.), в который помещаются все файлы с дистрибутивных дискет. Для вызова Турбо Паскаля необходимо отыскать в древовидной структуре каталогов ПК этот каталог и в нем файл с именем *TURBO.EXE*. Этот файл содержит готовую к работе диалоговую систему программирования Турбо Паскаль. В него входят минимально необходимые части Турбо Паскаля (текстовый редактор, компилятор, компоновщик, загрузчик). Для нормальной работы в диалоговой среде понадобятся также основная библиотека, располагающаяся в файле *TURBO.TPL*, и справочная служба (файл *TURBO.HLP*). В принципе, этих файлов достаточно для написания, компиляции и исполнения большинства примеров.

Пусть перечисленные файлы располагаются в каталоге *ТР* на диске *D*. Тогда для вызова Турбо Паскаля следует дать команду

D:\TP\TURBO

Для запуска Turbo Pascal необходимо перейти на диск в каталог с системой Turbo Pascal и дважды щёлкнуть по файлу Turbo.exe. После загрузки на экране появится окно редактирования Turbo Pascal.



В верхней части окна редактирования Turbo Pascal размещается меню, состоящее из опций: File (управление файлами), Edit (редактирование текста программы), Search (поиск и замена текста), Run (выполнение программы в различных режимах), Compile (компиляция, создание exe-файла), Debug (отладка программы), Tools (отладочные команды сообщения), Options (настройка среды программирования), Window (управление окнами), Help (доступ к встроенной справочной системе). Команды меню можно выбирать с помощью клавиатуры (F10 и клавиши управления курсором) либо мышью (щелчок по нужной опции меню).

Центральную часть окна занимает рабочая область, предназначенная для работы с текстом программ. В нижней части окна расположена строка статуса, в которую вводится информация о выполняемой в данный момент операции. В строке статуса отображаются также функциональные клавиши и комбинации клавиш, допустимые в данный момент в активном окне (F1-Help, F2-Save и т.д. ). Над строкой статуса в разрыве рамки окна указываются координаты курсора: номера редактируемой строки и позиции в строке, например, 6:11. Помимо этого имеются ещё вертикальные и горизонтальные полосы прокрутки, кнопка закрытия окна и др.

Не рекомендуется работать с системой, назначив в качестве каталога по умолчанию (текущего каталога) тот, в котором хранятся перечисленные выше файлы (этот каталог будем называть *системным).* Во-первых, в таком случае можно ошибочно стереть какой—либо из файлов системы программирования и тем самым нарушить ее работоспособность, а во-вторых, этот каталог очень скоро заполнится другими файлами, прямо не относящимися к Турбо Паскалю. Существует и еще одна причина, по которой нежелательно работать в системном каталоге. Дело в том, что Турбо Паскаль имеет свойство «запоминать» свою настройку в двух файлах с именами *TURBO.TP* и *TURBO.PCK*. При вызове система начинает поиск этих файлов в текущем каталоге. Если этот каталог — индивидуальный, система всякий раз будет настраиваться так, как этого хочешь. Если эти файлы не обнаружены в каталоге (а при первом обращении к Турбо Паскалю так оно и будет), система продолжит поиск в системном каталоге, а не найдя их там, настроится стандартным образом. Впоследствии можно сохранить настроечные файлы в своем каталоге и тем самым избавить себя от необходимости перенастройки системы всякий раз при обращении к ней.

. В Турбо Паскале можно работать одновременно с несколькими программами (или частями одной крупной программы), каждая из которых может располагаться в отдельном окне редактора. Среда позволяет использовать до девяти окон редактора одновременно.

Кроме окна (окон) редактора, в Турбо Паскале используются также окна: отладочного режима, вывода результатов работы программы, справочной службы, стека, регистров. По желанию они могут вызываться на экран поочередно или присутствовать на нем одновременно.

Для выхода из Турбо Паскаля следует нажать клавишу *Alt* и, *не отпуская ее*, — клавишу с латинской буквой *X*, после чего можно отпустить обе клавиши.

**Программа на контрольном примере.**

Для начала заходим в программу Turbo Pascal, затем набираем заданную программу.

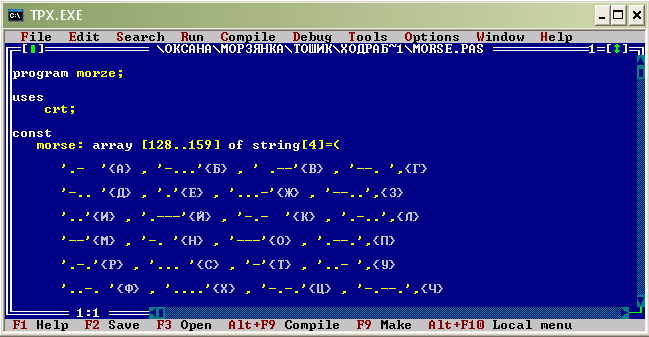


Рис.1

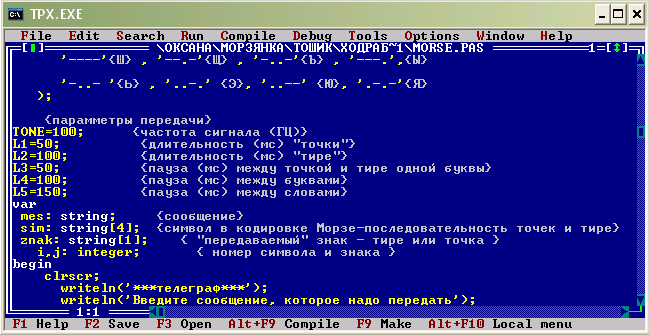


Рис.2

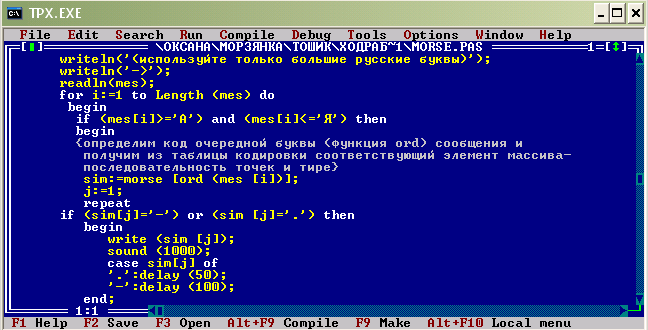


Рис.3

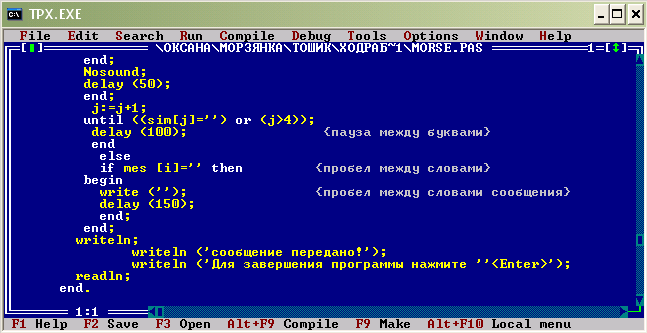


Рис.4

После того, как мы набрали программу, мы проверяем на наличие ошибок, нажав при этом комбинацию клавиш Alt + F9.

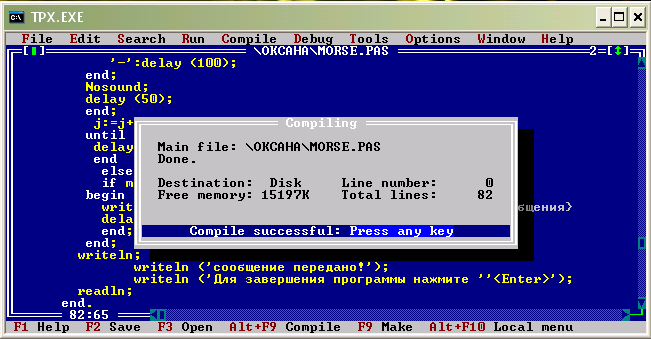


Рис.5

После проверки на ошибки мы переходим к следующему этапу, нажав комбинацию Ctrl + F9, после чего появляется окно, которое просит ввести сообщение.

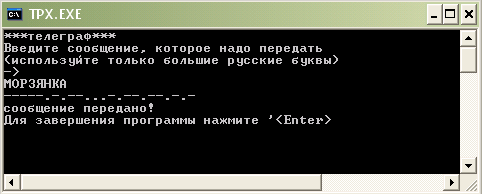


Рис.6

Для возвращения к программе мы нажимаем Enter. Чтобы просмотреть результат набранного сообщения используется комбинация Alt + F5. Для полного выхода из программы нажимаем Alt+ X.