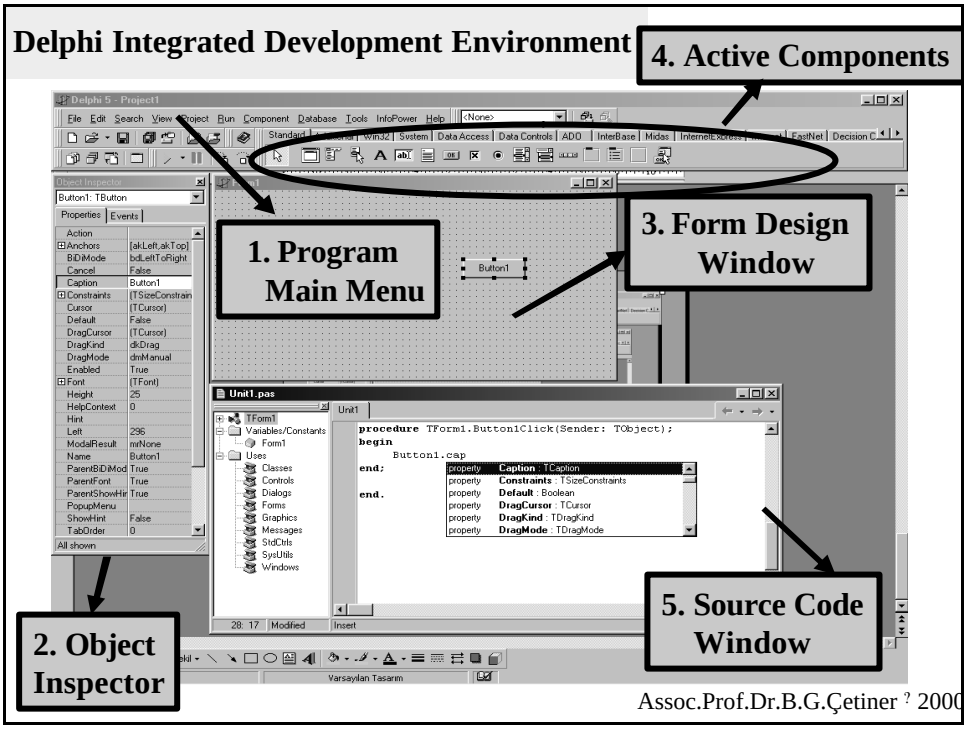
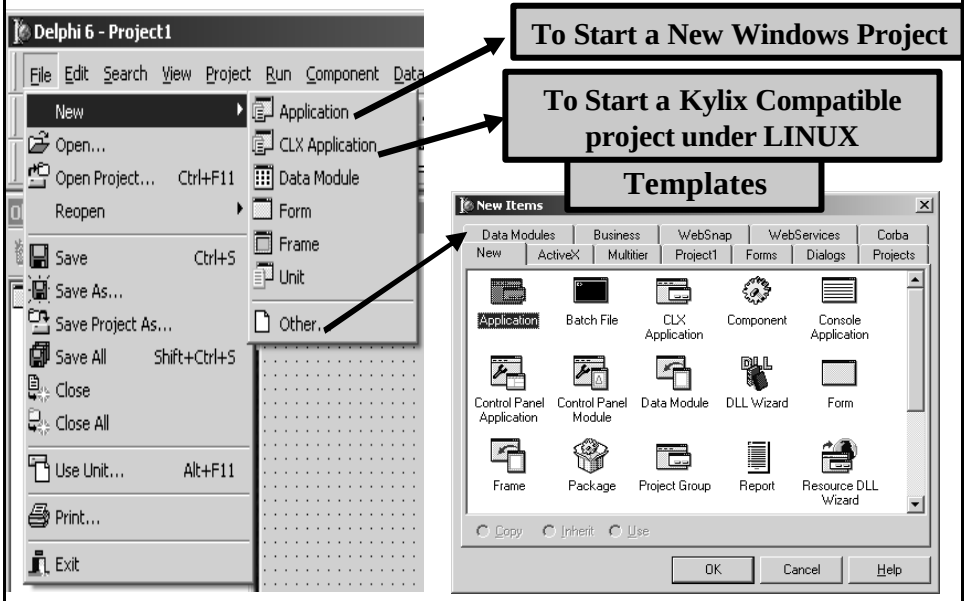
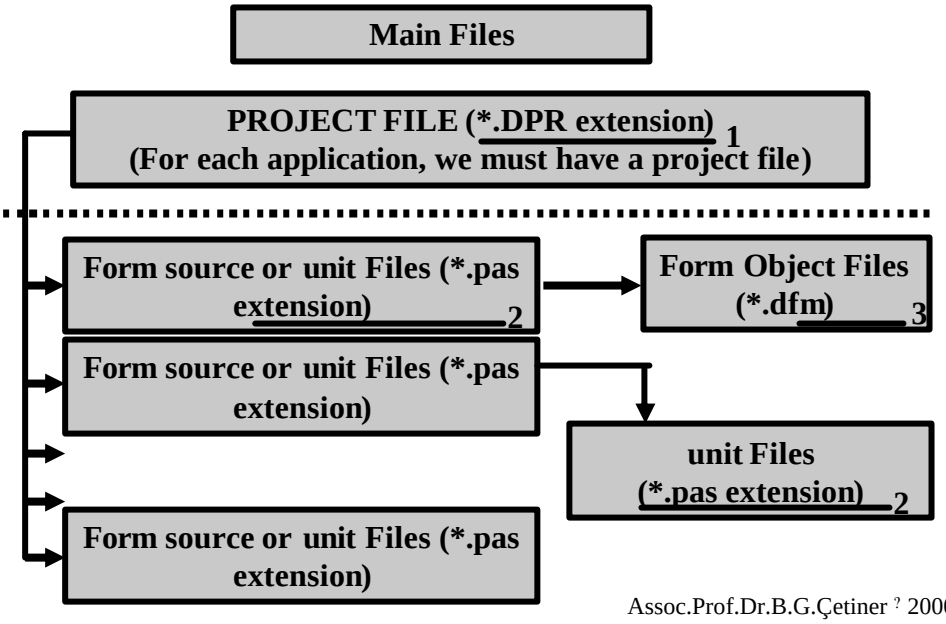
Delphi Integrated Development Environment



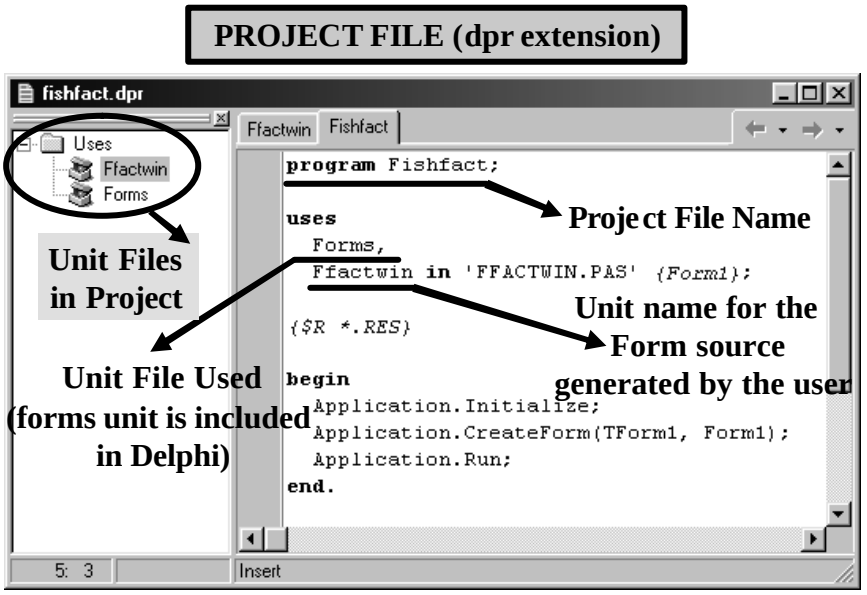
Delphi Integrated Development Environment



Main Files

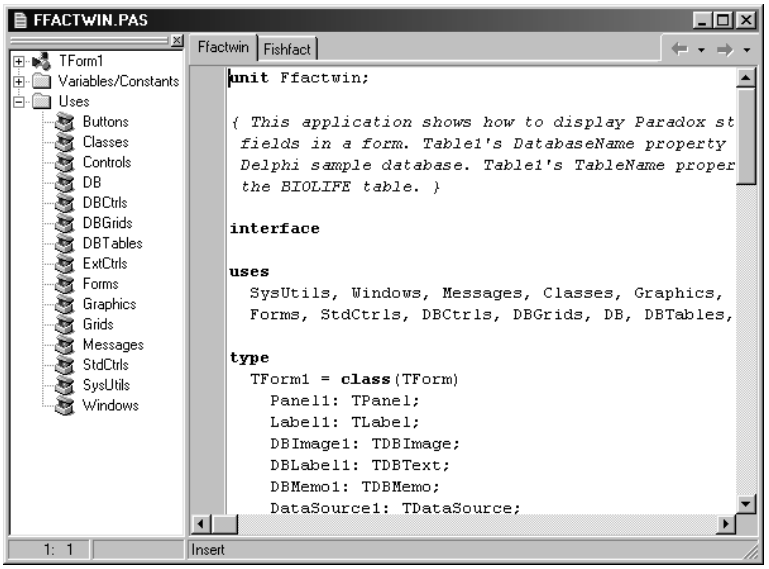


Main Structure of a Delphi Application



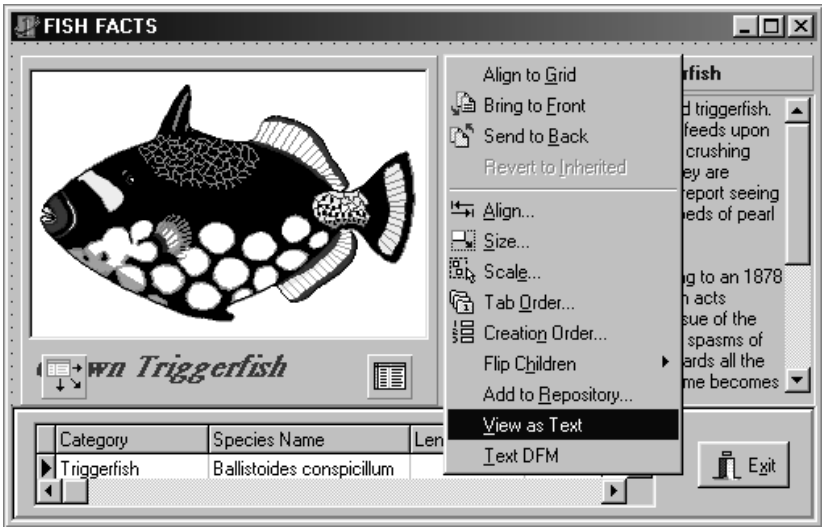
Main Structure of a Delphi Application

FORM SOURCE FILE (pas extension)



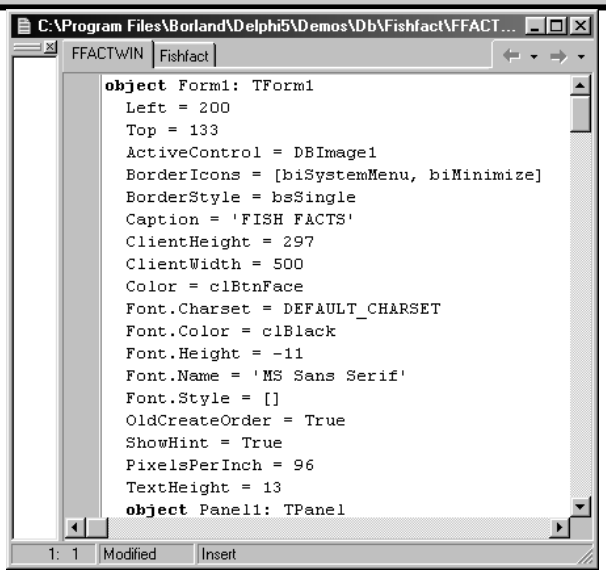
Main Structure of a Delphi Application

APPEARANCE OF FORM OBJECTS

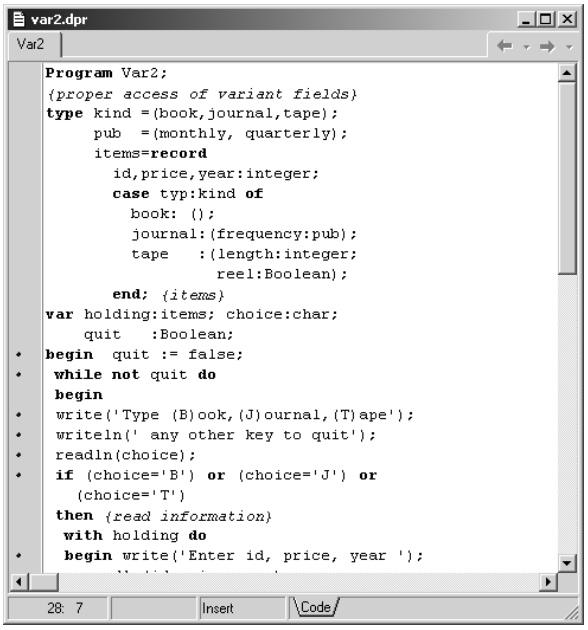


Main Structure of a Delphi Application

FORM OBJECTS (DESIGN) FILE (dfm extension)



Main Structure of a Delphi Application



Example: var2.dpr

This example is one single project file and does not use any form.

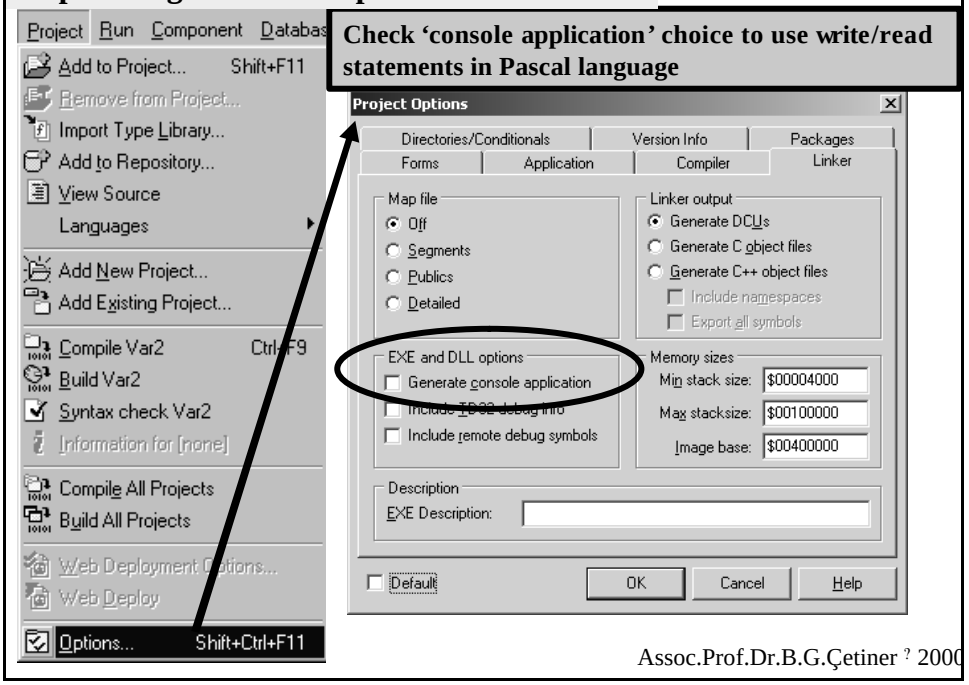
(Therefore, There is no *Uses* statement)

Delphi Integrated Development Environment

In this part, Delphi Integrated Development Environment is described.

Delphi Integrated Development Environment

Check 'console application' choice to use write/read statements in Pascal language



Delphi Integrated Development Environment

First Sample

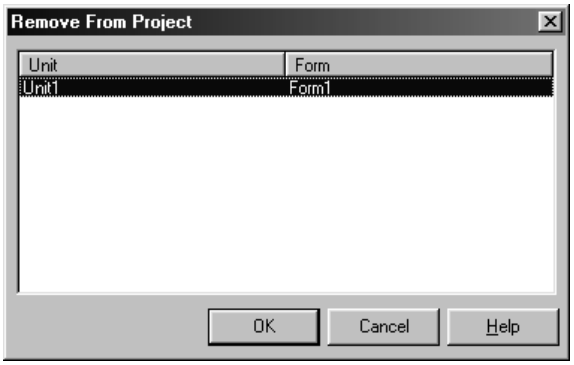
1.

Choose New Application

Delphi Integrated Development Environment

First Sample

2.



A Default **form named as Form1 (source name is Unit1) is constructed.**

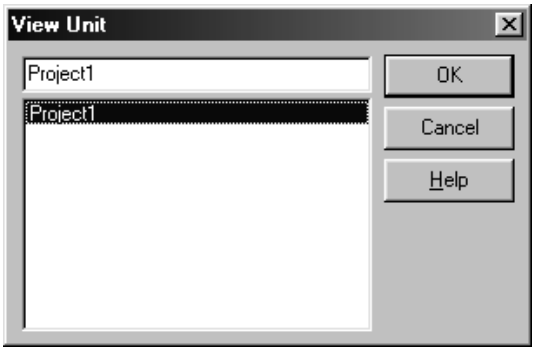
Remove this form from the project.

Delphi Integrated Development Environment

First Sample

3. 

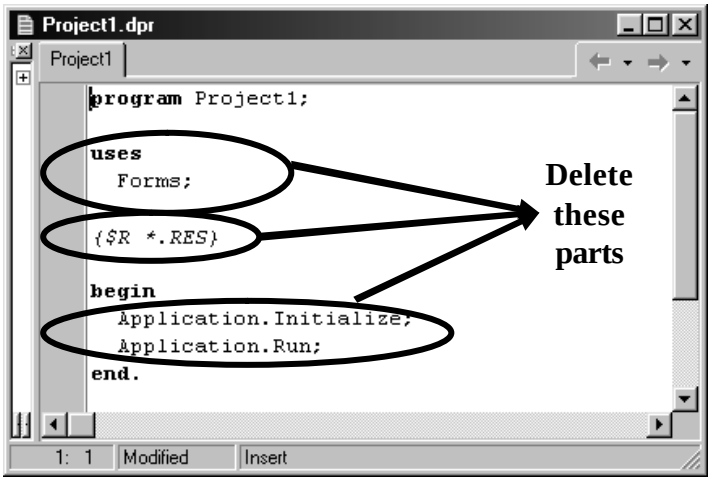
For viewing the project code press the view unit button and choose Project1 file from the dialog box as below.



Delphi Integrated Development Environment

First Sample

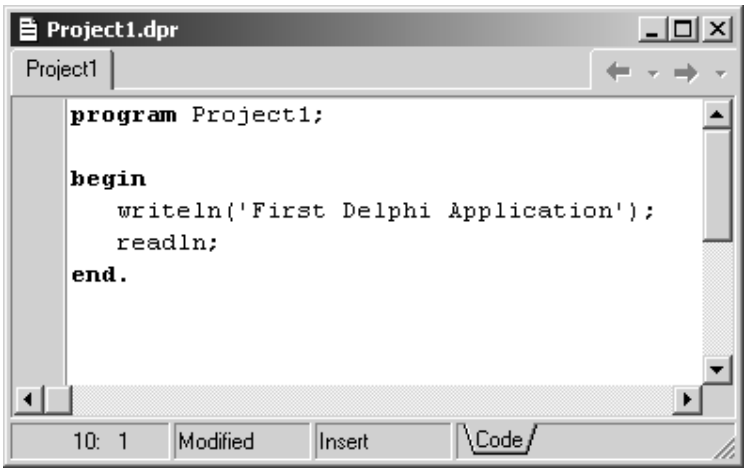
3.



Delphi Integrated Development Environment

First Sample

4.

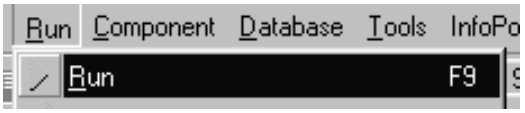


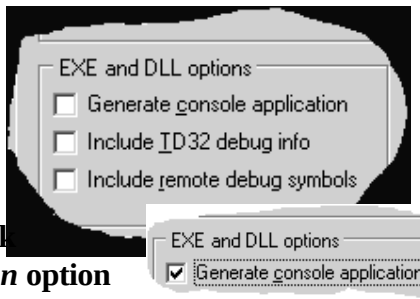
Delphi Integrated Development Environment

First Sample

5. Compile and Run the program

or





If runtime error occurs then
Go to Project/Options and check
The *Generate console application* option

Delphi Integrated Development Environment

First Sample **Screen Output**

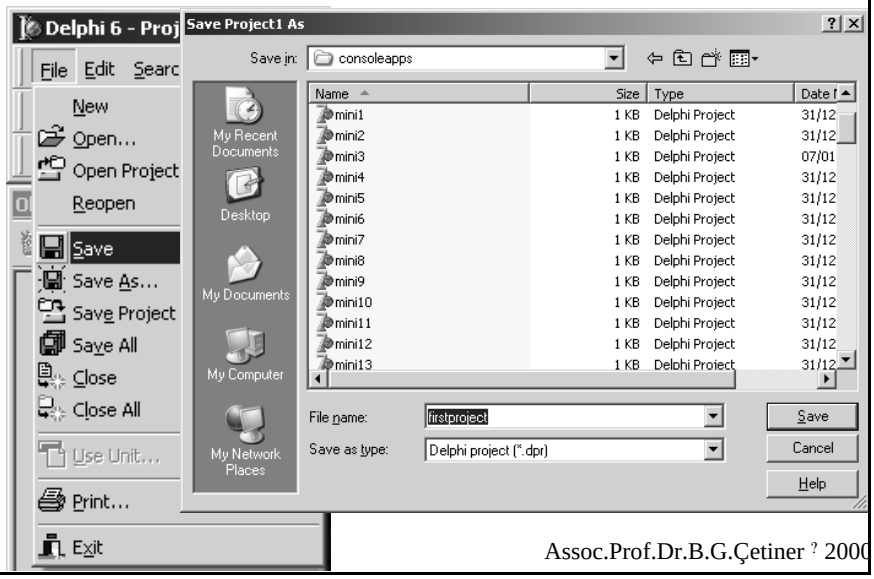
6.

Console Application is run in a DOS screen console

Delphi Integrated Development Environment

First Sample **For saving the Project**

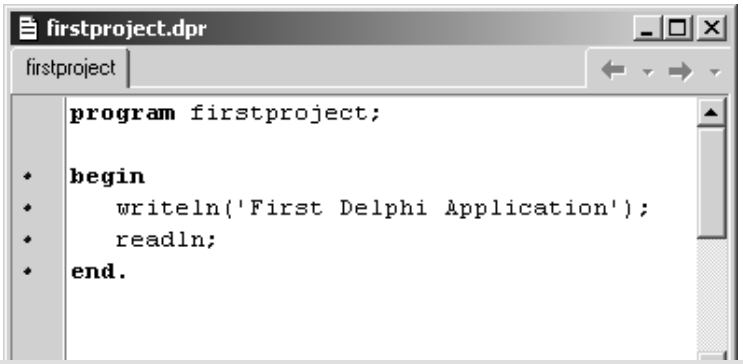
6.



Delphi Integrated Development Environment

First Sample

Project was saved as 'firstproject'.



Writeln and readln statements are not normally used in visual applications. They are basic input-output statements under standard pascal.

Delphi Integrated Development Environment

An application including a simple form

Choose new application and run it directly.

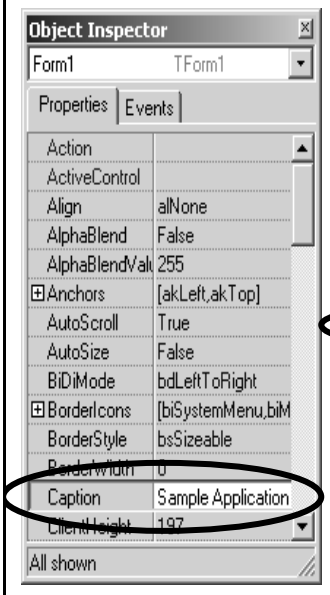
Delphi Integrated Development Environment

An application including a simple form

New application is a windows program with a single form.

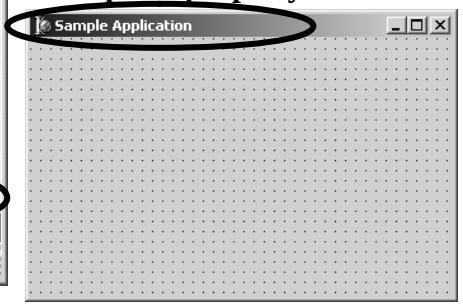
Delphi Integrated Development Environment

Object Inspector Window



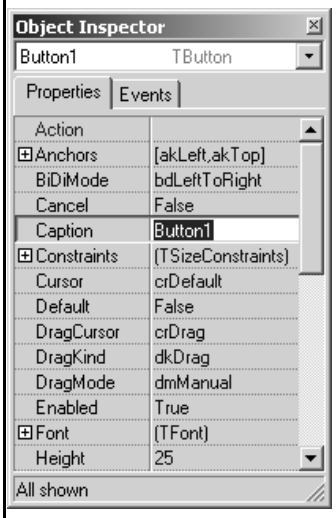
You can change the properties of selected objects in *Properties* page of Object Inspector Window.

Example: You select Form object and then, you can change properties of Form1 object. Enter the new value 'Sample Application' for caption property.

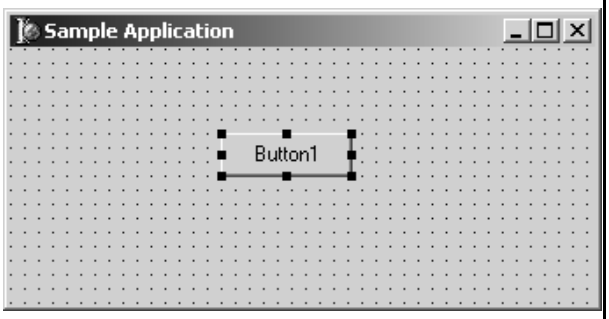


Delphi Integrated Development Environment

Object Inspector Window



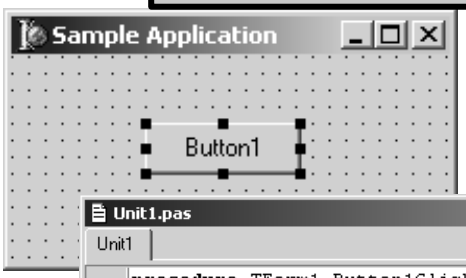
Now Choose a TButton object from Components Pallette and place on the form. Now under properties Page, you can change the properties of TButton object.



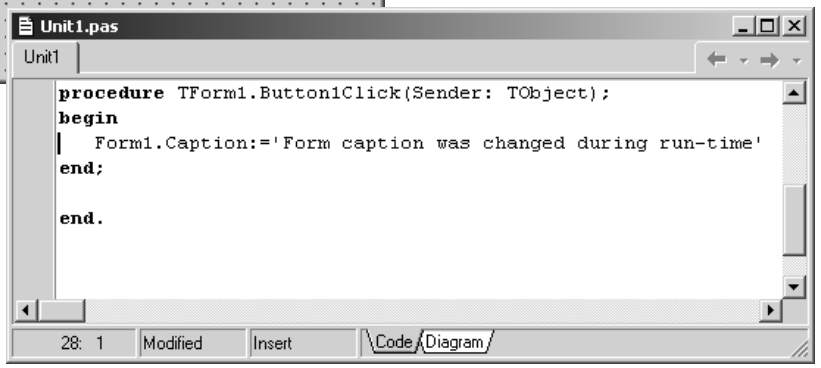
These changes are performed in *Design Time.*

Delphi Integrated Development Environment

Changing object properties during run-time

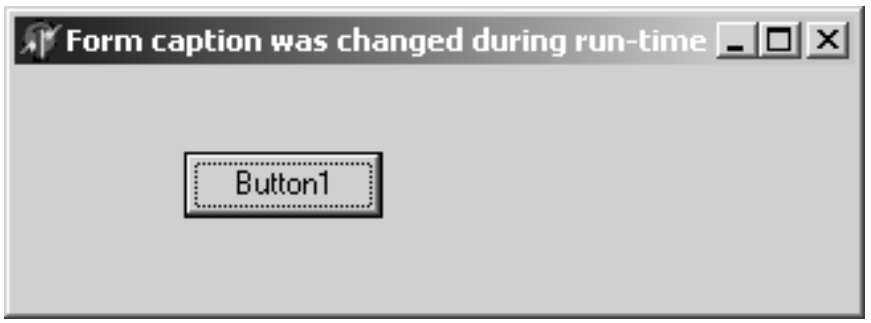


Double-click on Button1. **Then add the line below in source code.**



Delphi Integrated Development Environment

Changing object properties during run-time

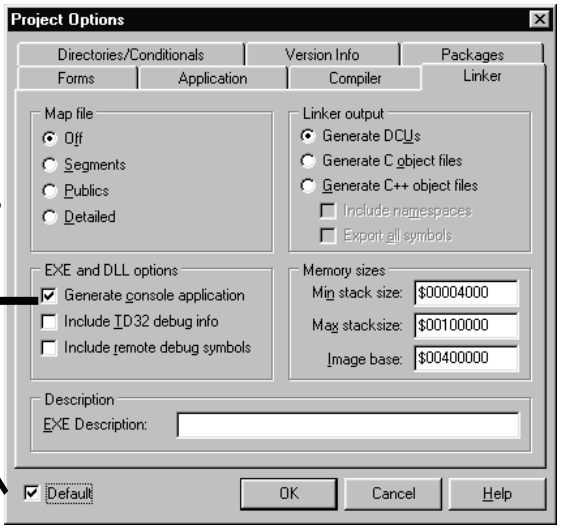


Press *Button1* button to change the Form caption when program is running.

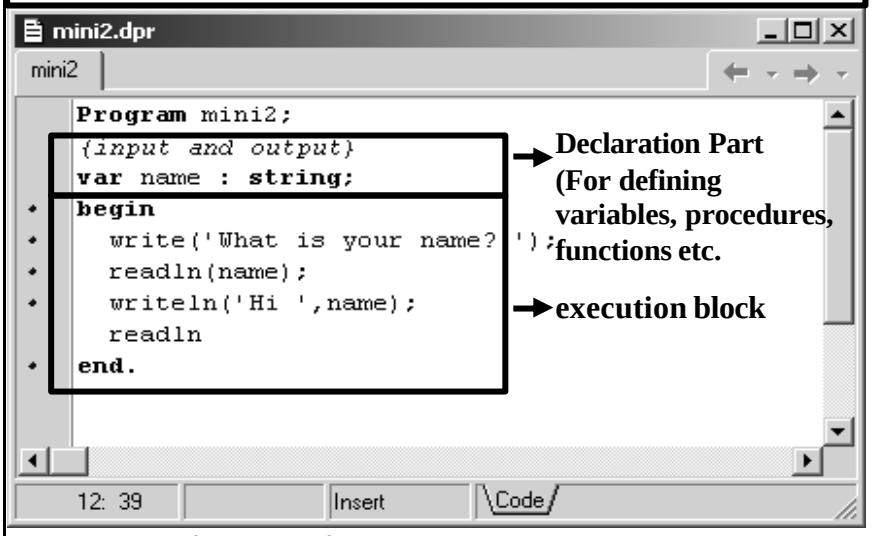
Basics of Language

Examples regarding Basics of Language are in CONSOLEAPPS folder.

For *Basics of Language* examples, choose these two checkboxes.



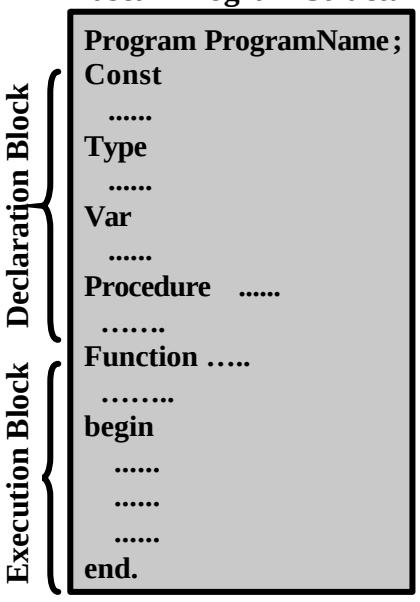
Sample program ***mini2.dpr*** *in consoleapps folder*



Note: Samples for Basics of Language are given as *console* applications under *consoleapps* directory

Pascal Program Structure

Declaration Block:
Consists of *Variable, procedure* and *function* definitions.

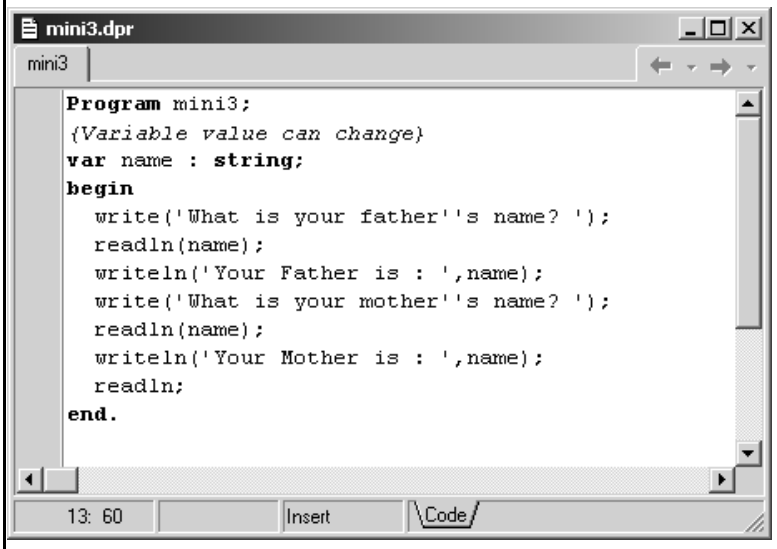


String Data Type

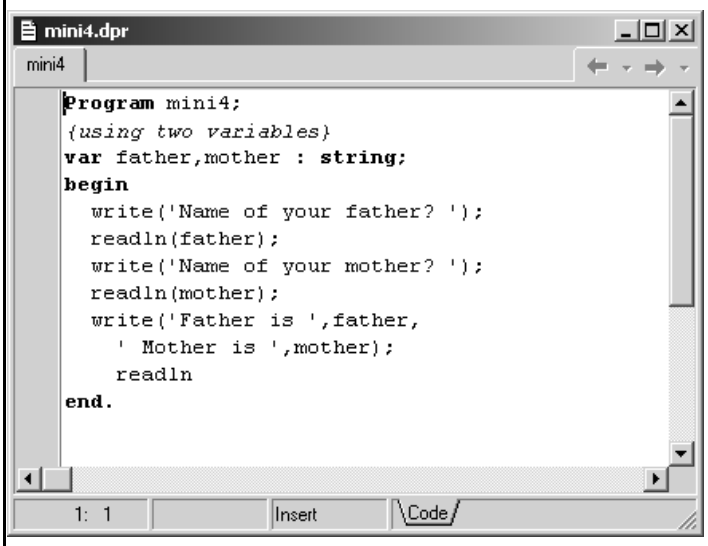
Type	Maximumum Length	Required Memory
ShortString	255 characters	2-256 bytes
AnsiString	~2^31 characters	4 bayt-2GB
WideString	~2^30 characters	4 bayt-2GB

If you define a variable as *String*, it is treated as *ansistring*.

Sample program mini3.dpr in consoleapps folder



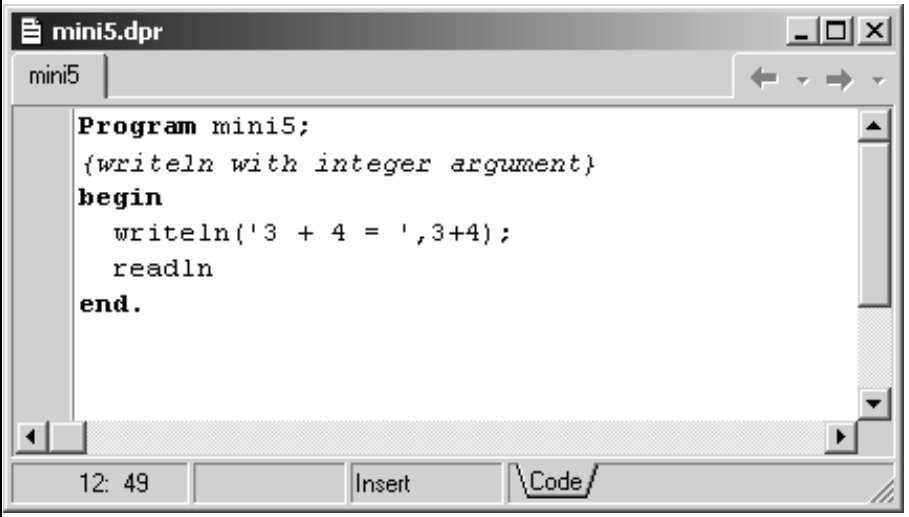
*Sample program **mini4.dpr** in consoleapps folder*



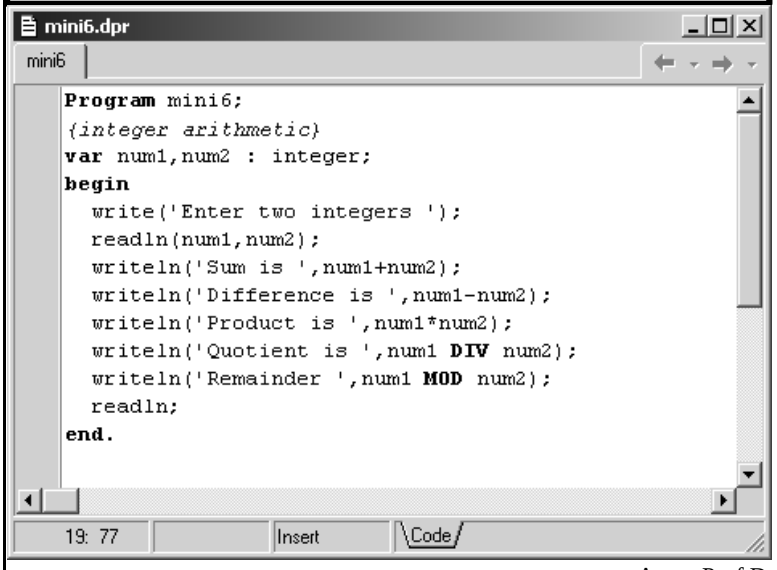
Integer Data Types

Type	Range	Memory Req.
Integer	-2147483648..2147483647	4
Cardinal	0..4294967295	4
Shortint	-128..127	1
Smallint	-32768..32767	2
Longint	-2147483648..2147483647	4
Int64	-2^63..2^63-1	8
Byte	0..255	1
Word	0..65535	2
Longword	0..4294967295	4

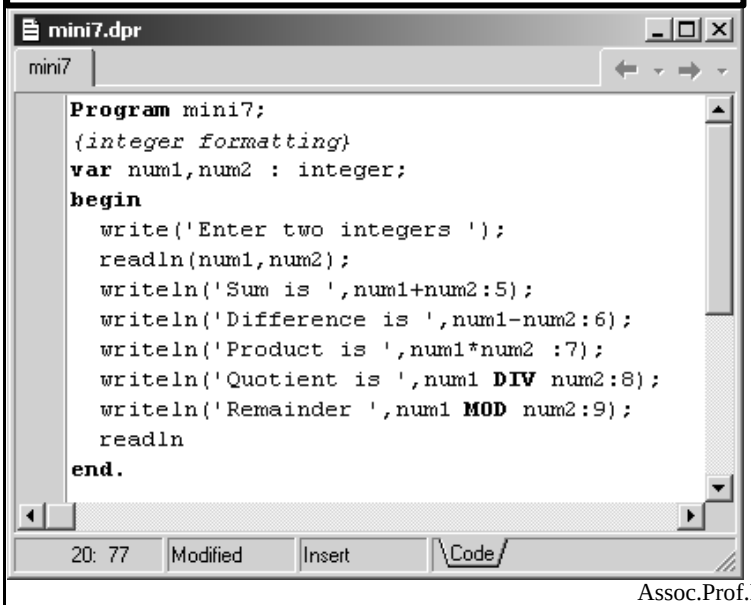
*Sample program **mini5.dpr** in consoleapps folder*



*Sample program **mini6.dpr** in consoleapps folder*



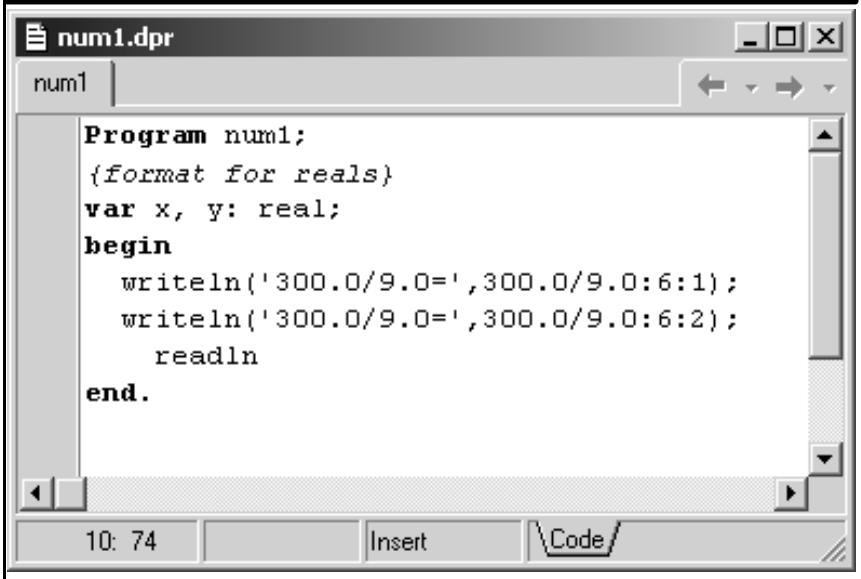
*Sample program **mini7.dpr** in consoleapps folder*



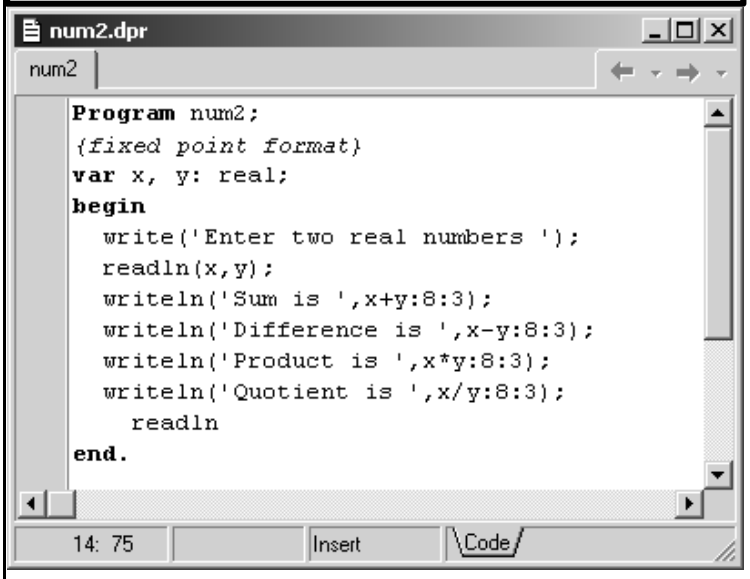
Real Data Types

Type	Range	Number of Stored Digits	Memory Req. (bytes)
Real48	2.9x10^-39..1.7x10^38	11-12	6
Single	1.5x10^-45..3.4x10^38	7-8	4
Double	5.0x10^-324..1.7x10^308	15-16	8
Extended	3.6x10^-4951..1.1x10^4932	19-20	10
Comp	-2^63+1..2^63-1	19-20	8
Currency	-922337203685477.5808.. 922337203685477.5807	19-20	8
Real	5.0x10^-324..17x10^308	15-16	8

*Sample program **num1.dpr** in consoleapps folder*



*Sample program **num2.dpr** in consoleapps folder*



*Sample program **num3.dpr** in consoleapps folder*

num3.dpr

num3

```
Program num3;
{floating point format}
var x,y : real;
begin
  write('Enter two real numbers ');
  readln(x,y);
  writeln('Sum is ',x+y:8);
  writeln('Difference is ',x-y:8);
  writeln('Product is ',x*y:8);
  writeln('Quotient is ',x/y:8);
  readln
end.
```

14: 65 | Modified | Insert | Code

*Sample program **num4.dpr** in consoleapps folder*

num4.dpr

num4

```
Program num4;
{run-time format specification}
var field, precision : integer;
    int : integer; rel : real;
begin
  write('Enter field and precision ');
  readln(field,precision);
  write('Enter a number '); readln(int);
  write('Enter a real '); readln(rel);
  writeln(int:field);
  writeln(int:field+5);
  writeln(rel:field:precision);
  writeln(rel:field+5:precision+2);
  readln
end.
```

17: 50 | Insert | Code

Sample program num6.dpr in consoleapps folder

num6.dpr

num6

```
Program num6;
{assignment statement}
var num1, num2, sum, difference :
  integer;
begin
  write('Enter two integers ');
  readln(num1,num2);
  sum := num1 + num2;
  difference := num1 - num2;
  writeln('Sum is ',sum);
  writeln('Difference is ',difference);
  readln
end.
```

15: 79 | Insert | Code

Sample program num7.dpr in consoleapps folder

num7.dpr

num7

```
Program num7;
{ROUND and TRUNC}
var x,y : real; int_sum : integer;
begin
  write('Enter two real numbers ');
  readln(x,y);
  writeln('Truncated ',trunc(x),' ',trunc(y));
  writeln('Rounded ',round(x),  ' ',round(y));
  int_sum := round(x+y);
  writeln('Rounded sum ',int_sum);
  int_sum := round(x) + round(y);
  writeln('Sum of rounded ',int_sum);
  int_sum := trunc(x+y);
  writeln('Truncated sum ',int_sum);
  int_sum := trunc(x) + trunc(y);
  writeln('Sum of truncated ',int_sum);
    readln
end.
```

20: 78 | Insert | Code

Sample program num8.dpr in consoleapps folder

num8.dpr

num8

```
Program num8;
{real assignment and arithmetic}
var area,width,height,length : real;
begin
  write('Height of TV screen? ');
  read(height);
  write('Width of TV screen? ');
  read(width);
  area := height * width;
  writeln('Area of screen ',area:6:2);
  write('Enter area of pool ');
  readln(area);
  write('Enter its length ');
  readln(length);
  width := area / length;
  writeln('Width of pool ',width:6:2);
    readln
end.
```

20: 78 Insert Code

Sample program num9.dpr in consoleapps folder

num9.dpr

num9

```
Program num9;
{built-in functions}
var x, s, c, l, e : real;
begin
  write('Enter a real number ');
  readln(x);
  s := sin(x);
  c := cos(x);
  e := exp(x);
  l := ln(e);
  writeln('sin      ',s);
  writeln('cos      ',c);
  writeln('exp      ',e);
  writeln('ln(exp)  ',l);
    readln
end.
```

18: 76 Insert Code

Sample program mini8.dpr in consoleapps folder

mini8.dpr

```
Program mini8;
{integer and string variables}
var f_name,m_name : string;
    f_age,m_age : integer;
begin
  write('Father''s name? ');
  readln(f_name);
  write('Mother''s name? ');
  readln(m_name);
  write('Father''s and', ' mother''s age? ');
  readln(f_age,m_age);
  writeln('Father ',f_name,' is ',  f_age,' years old');
  writeln('Mother ',m_name,' is ',  m_age,' years old');
  readln
end.
```

22: 76 Insert Code

Sample program mini9.dpr in consoleapps folder

mini9.dpr

```
Program mini9;
{text files}
var f : text; x : integer;
  begin
  writeln('Writing to file1');
  assign(f,'file1'); {step 1}
  rewrite(f); {step 2}
  {step 3 - writing}
  writeln(f,3+4,' ',17-25,' ',8,' ');
  writeln(f,-100);
  close(f); {step 4}
  {now use f for reading - step 1 done}
  writeln('Reading from file1');
  {step 2 - set file for reading}
  reset(f);
  {step 3 - reading}
  read(f,x); writeln(x);
  read(f,x); writeln(x);
  close(f); {step 4}
  readln
end.
```

20: 72 Insert Code

mini10.dpr

mini10

```
Program mini10;
{two files}
var f1,f2 : text;
    fname : string;
    x : integer;
begin
  write('We are going to read from ',
   'file1 and write into another file');
  write('Enter name of second file ');
  readln(fname);
  assign(f1,'file1');
  assign(f2,fname);
  {note use of string variable above}
  reset(f1);
  rewrite(f2);
  read(f1,x); write(f2,x,' ');
  write('Enter a number '); readln(x);
  writeln(f2,x);
  read(f1,x); write(f2,x,' ');
  write('Enter a number '); readln(x);
  writeln(f2,x);
  close(f1); close(f2);
  readln
end.
```

25: 81 Insert Code

Sample program mini10.dpr in consoleapps folder

Sample program mini12.dpr in consoleapps folder

mini12.dpr

mini12

```
Program mini12;
{declared string and integer constants}
const c1 = 32; c2 = 5; c3 = 9;
  prompt = 'Enter Fahrenheit ';
  answer = ' degrees Celsius';
var fahr, cels : integer;
begin
  write(prompt); readln(fahr);
  cels := ((fahr-c1) * c2) div c3;
  write(cels,answer);
  readln;
end.
```

15: 74 Insert Code

IF-THEN (Control Statements)

```
if { condition1 } then
  { statement1 }
  else
   {statement2};
```

```
if ... { condition1 } then
begin
  if ... {condition2 } then
    ... { statement1 }
  else
    ... { statement2 }
end;
```


IF-THEN (Control Statements)

Examples

```
if x>12 then y:=x*2;
```

If there are more than one statement,

```
if Salary>10000 then
begin
    Tax:=Oran1*SalaryBeforeTax;
    Net:=SalaryBeforeTax-Tax;
end;
```

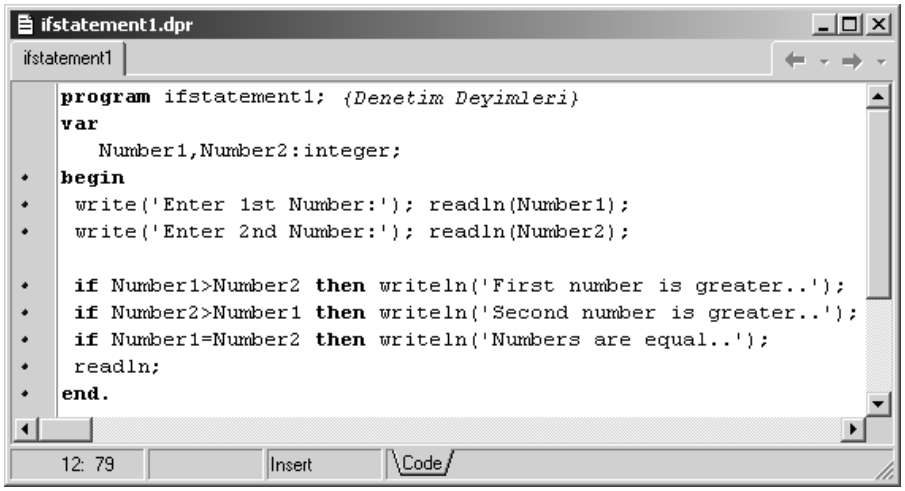
statements are put between begin-end words as a block

```
if Delta<0 then
     message:='imaginary numbers'
else
begin
   x1:=(-b+Sqrt(Delta))/2*a;
   x2:=(-b-Sqrt(Delta))/2*a;
end;
```

Basics of Language

IF-THEN (Control Statements)

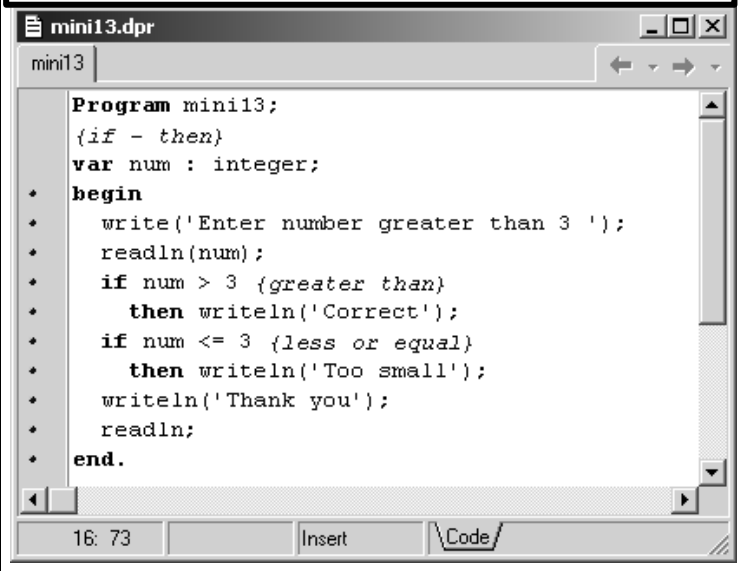
Sample program ifstatement1.dpr in consoleapps folder



Basics of Language

IF-THEN (Control Statements)

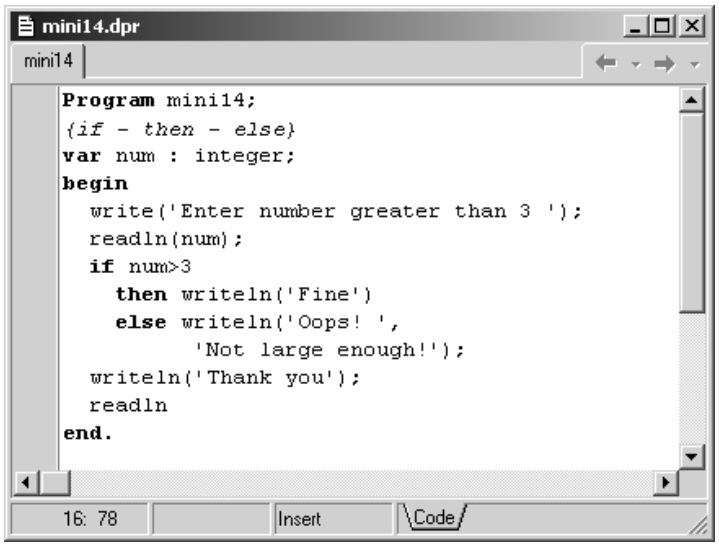
Sample program mini13.dpr in consoleapps folder



Basics of Language

IF-THEN (Control Statements)

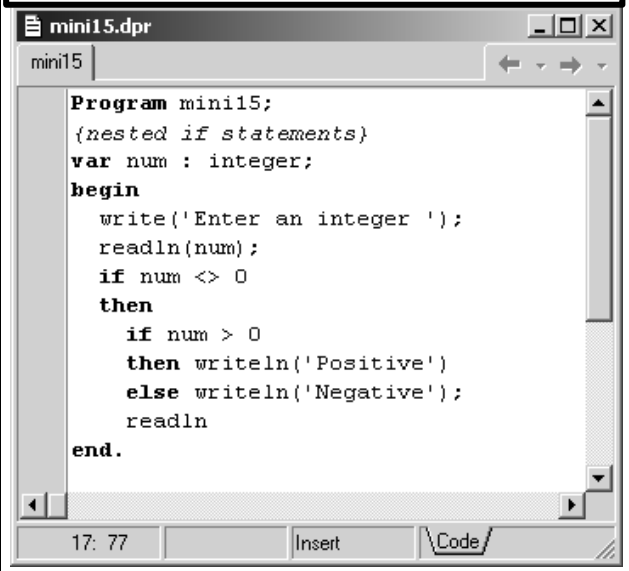
Sample program mini14.dpr in consoleapps folder



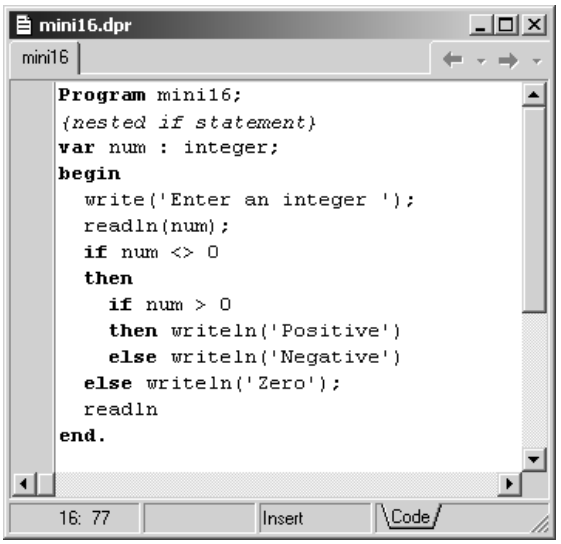
Basics of Language

IF-THEN (Control Statements)

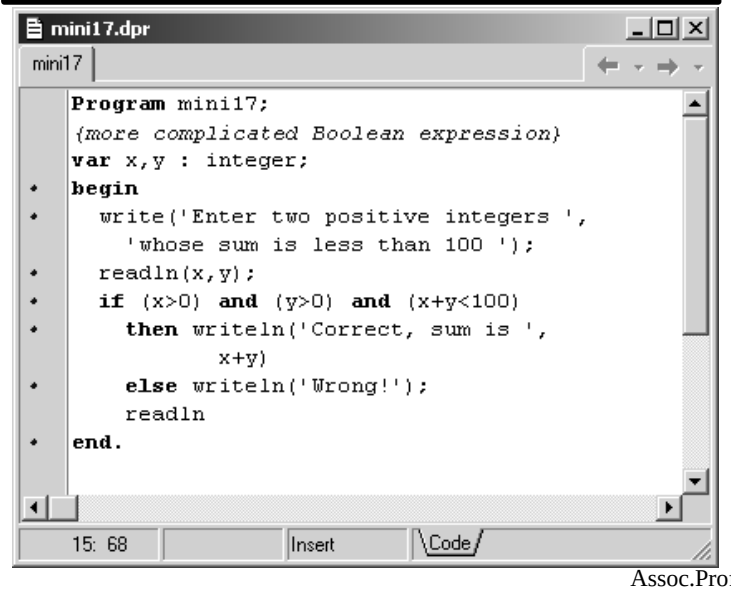
Sample program mini15.dpr in consoleapps folder



Sample program mini16.dpr in consoleapps folder



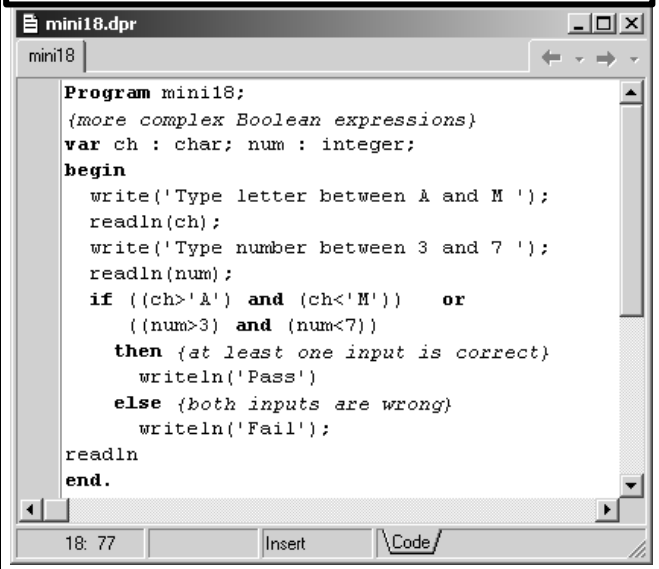
Sample program mini17.dpr in consoleapps folder



Basics of Language

IF-THEN (Control Statements)

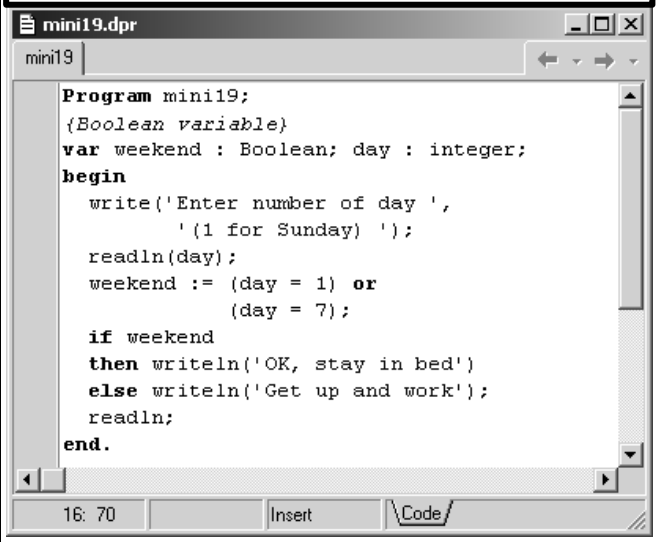
Sample program mini18.dpr in consoleapps folder



Basics of Language

IF-THEN (Control Statements)

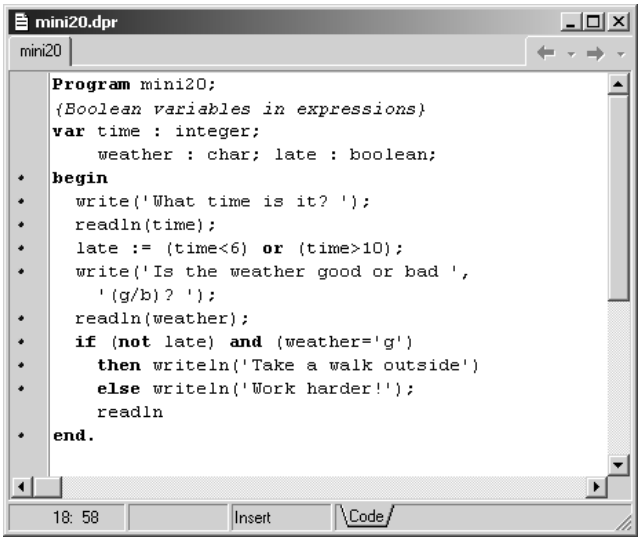
Sample program mini19.dpr in consoleapps folder



Basics of Language

IF-THEN (Control Statements)

Sample program mini20.dpr in consoleapps folder



Basics of Language

CASE (Control Statements)

```
case Option of
  List1: statement1;
  ...
  List_n: statement_n;
else
  statement;
end
```

If there are more than one statements, these statements are put between begin-end Words.

EXAMPLE

```
case I of
  1..5: Caption := 'Small';
  6..9: Caption := 'Big';
  0, 10..99: Caption := 'Invalid Value';
else
  Caption := '';
end;
```

Both are same

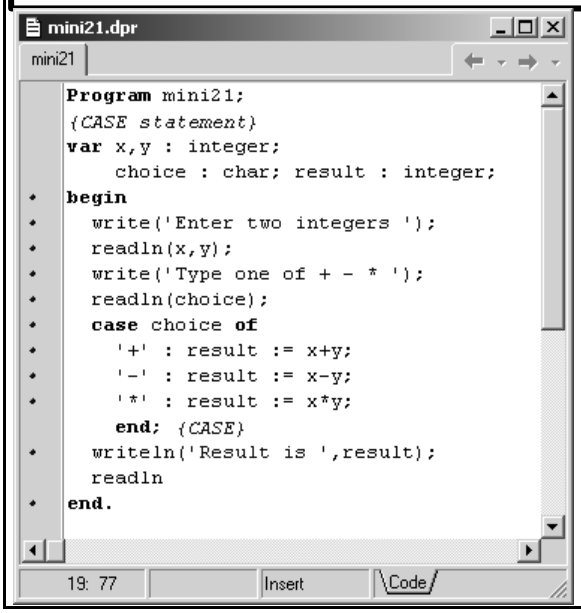
```
if I in [1..5] then
  Caption := 'Small'
  else if I in [6..10] then
    Caption := 'Big'
    else if (I = 0) or (I in [10..99]) then
      Caption := 'Invalid Value'
      else
        Caption := '';
```


Examples

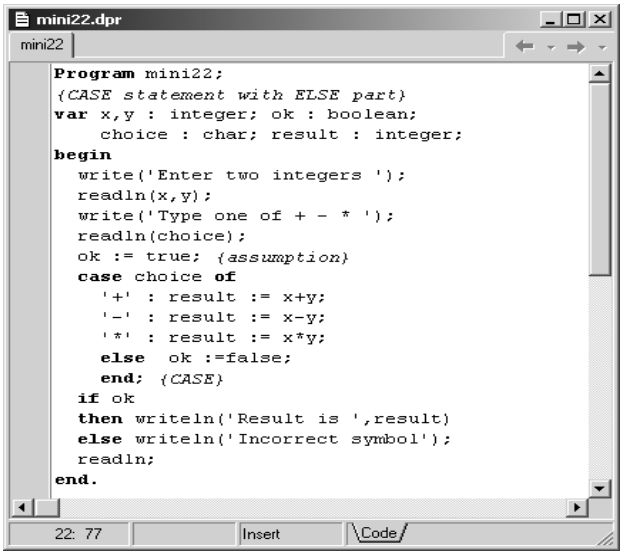
```
case Colour of
  Red: begin
            X := 1; y:=x*2;
          end;
  Green: X := 2;
  Blue: X := 3;
  Yellow, Orange, Black: X := 0;
end;
```

```
case Selection of
  OK: Form1.Close;
  Calculate: CalculateTotal(UnitCost, Quantity);
else
  Beep;
end;
```

Sample program mini21.dpr in consoleapps folder

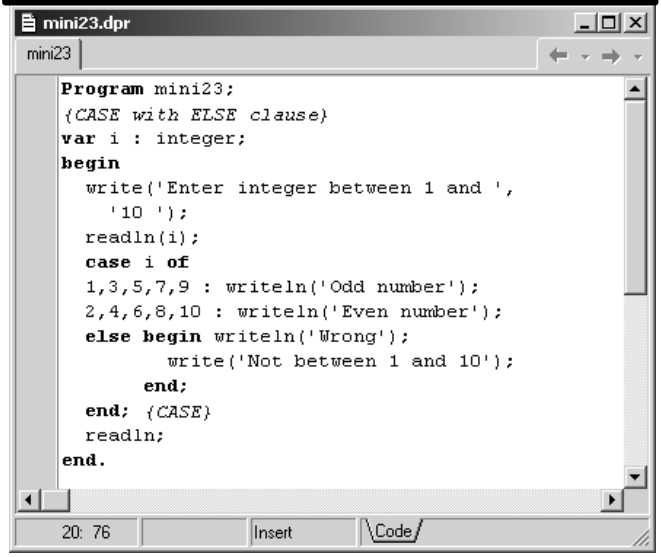


Sample program mini22.dpr in consoleapps folder



CASE (Control Statements)

Sample program mini23.dpr in consoleapps folder



Repeat-Until Loops (Loops)

```
repeat

    statement1;
    ...;
    statement_n;

until conditionStatement
```

Statements between repeat-until are repeated until the condition is met.

Repeat-Until Loops (Loops)

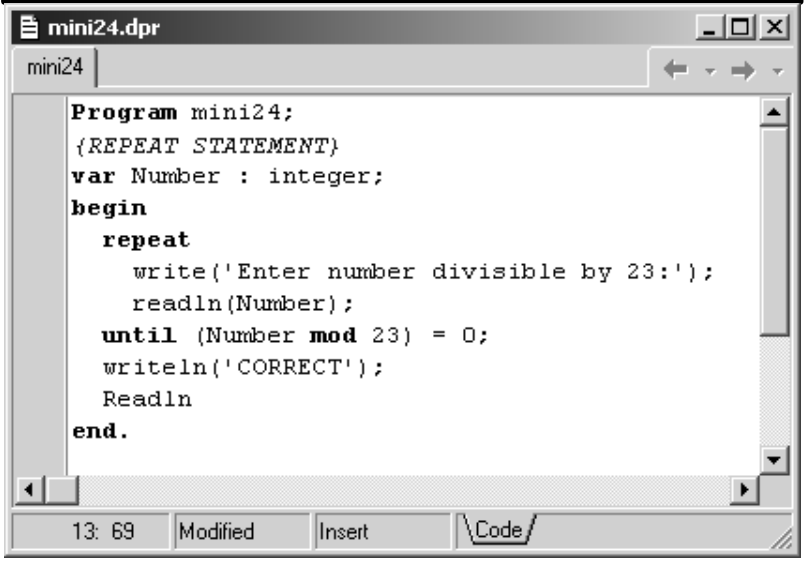
EXAMPLE

```
repeat

  Write('Enter a value between 0..9 : ');
  Readln(I);

until (I >= 0) and (I <= 9);
```

Two statements between Repeat-until are repeated until user enters a value outside the range of 0..9

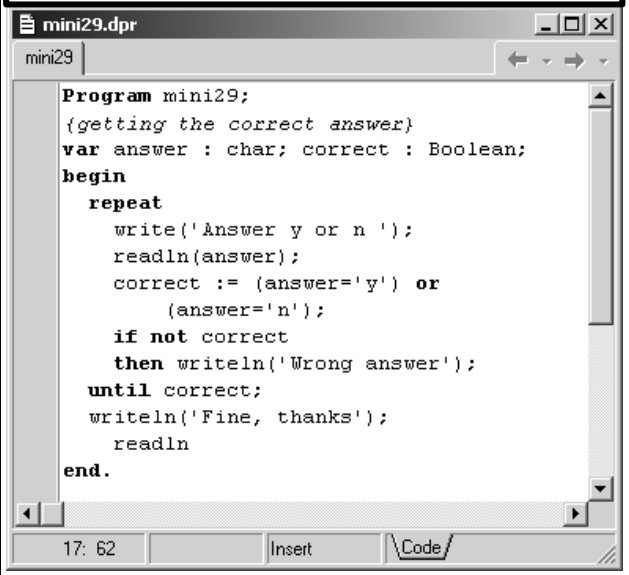
Repeat-Until Loops (Loops)

Sample program mini24.dpr in consoleapps folder



Repeat-Until Loops (Loops)

Sample program mini29.dpr in consoleapps folder



WHILE Loops (Loops)

```
while condition do statement;
```

```
while condition do
begin
   statement1;
   .....
   statement_n;
end;
```

Statements are repeated as long as condition is valid.
Condition is checked at the beginning.

WHILE Loops (Loops)

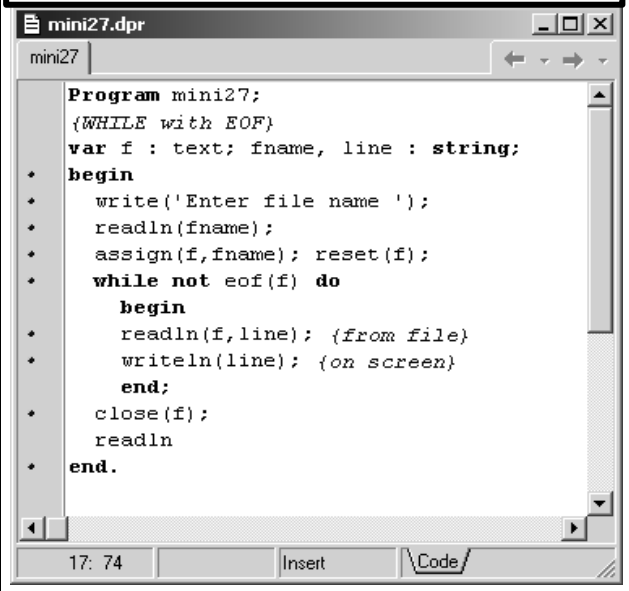
Examples

```
while Data[I] <> X do I := I + 1;
```

```
while I > 0 do
begin
  if Odd(I) then Z := Z * X;
  I := I div 2;
  X := Sqr(X);
end;
```


WHILE Loops (Loops)

Sample program mini27.dpr in consoleapps folder



Basics of Language

WHILE Loops (Loops)

Sample program mini28.dpr in consoleapps folder

mini28.dpr

mini28

```
Program mini28;
{IORESULT - checking file operations}
var f : text; fname, line : string;
begin
  write('Enter file name ');
  readln(fname);
  assign(f,fname); reset(f);
  if ioresult <> 0
  then writeln('Cannot open file')
  else {OK, read file}
    begin {same as mini24}
    while not eof(f) do
      begin
      readln(f,line); {from file}
      writeln(line); {on screen}
      end;
    close(f);
    end; {successful file access}
      readln
end.
```

22: 76 Insert Code

Basics of Language

WHILE Loops (Loops)

Sample program mini30.dpr in consoleapps folder

mini30.dpr

mini30

```
Program mini30;
{counting with WHILE}
var count, number, sum : integer;
begin
  count := 0; sum := 0;
  while count < 5 do
    begin
    write('Enter an integer ');
    readln(number);
    sum := sum + number;
    count := count +1;
    end;
  writeln('Sum is ',sum);
  readln
end.
```

17: 76 Insert Code

FOR Loops (Loops)

```
FOR variable := FirstValue TO LastValue DO statement;

or

FOR variable := FirstValue DOWNTO LastValue DO statement;
```

Statement (or statements) are repeated starting from the first value until the last value.

FOR Loops (Loops)

Examples

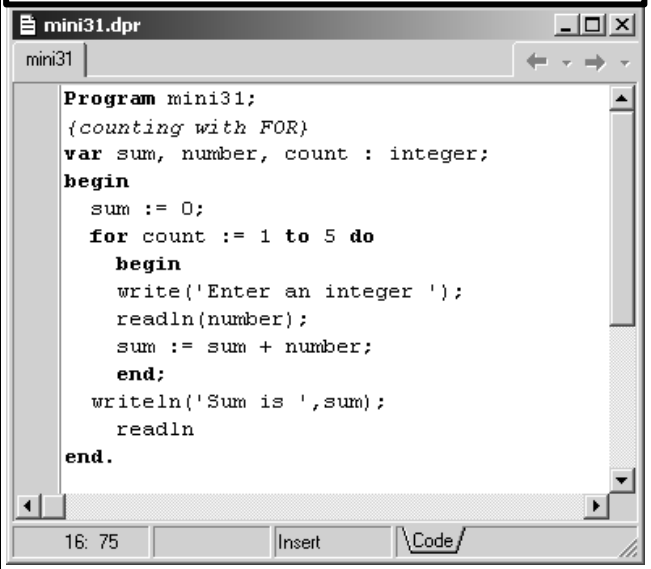
```
for I := 2 to 63 do
  if Data[I] > Max then Max := Data[I];
```

```
for I := ListBox1.Items.Count - 1 downto 0 do
    ListBox1.Items[I] := UpperCase(ListBox1.Items[I]);
```

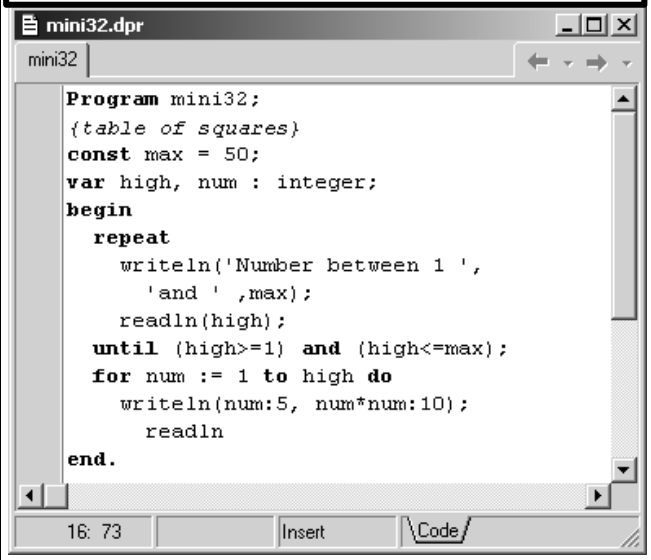
```
for I := 1 to 10 do
  for J := 1 to 10 do
  begin
    X := 0;
    for K := 1 to 10 do
      X := X + Mat1[I, K] * Mat2[K, J];
    Mat[I, J] := X;
  end;
```

```
for C := Red to Blue do Check(C);
```

*Sample program **mini31.dpr** in consoleapps folder*



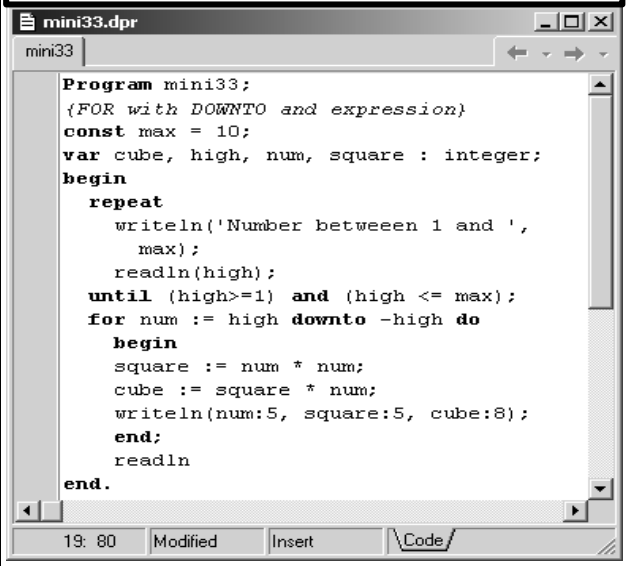
*Sample program **mini32.dpr** in consoleapps folder*



Basics of Language

FOR Loops (Loops)

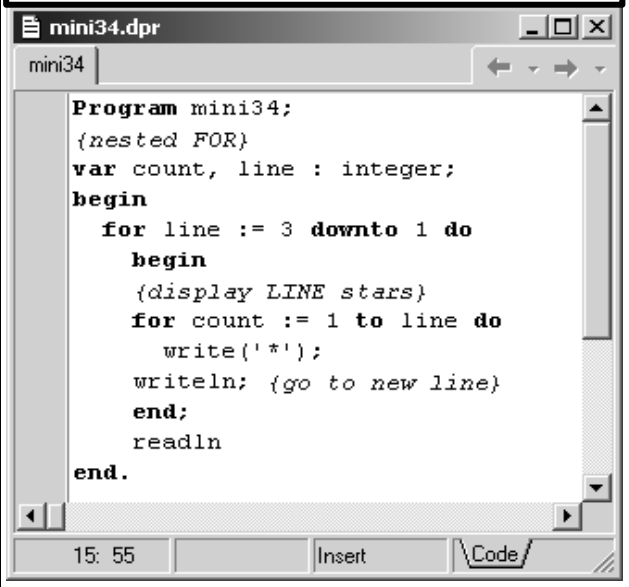
Sample program mini33.dpr in consoleapps folder



Basics of Language

FOR Loops (Loops)

Sample program mini34.dpr in consoleapps folder



Functions for Character and String Manipulations

```
function Chr(X: Byte): Char;
function UpCase(Ch: Char): Char;
function Copy(S; Index, Count: Integer): string;
procedure Delete(var S: string; Index, Count:Integer);
procedure Insert(Source: string; var S: string; Index: Integer);
function Concat(s1 [, s2,..., sn]: string): string;
function Length(S): Integer;
function Pos(Substr: string; S: string): Integer;
procedure Str(X [: Width [: Decimals ]]; var S);
procedure Val(S; var V; var Code: Integer);
```


Functions for Character and String Manipulations

```
function SizeOf(X): Integer;
function IsDelimiter(const Delimiters, S: string; Index: Integer): Boolean;
function LowerCase(const S: string): string; (veya AnsiLowerCase)
function UpperCase(const S: string): string; (veya AnsiUpperCase)
function TrimLeft(const S: string): string;
function TrimRight(const S: string): string;
function Trim(const S: string): string;
```

Basics of Language

Functions for Character and String Manipulations

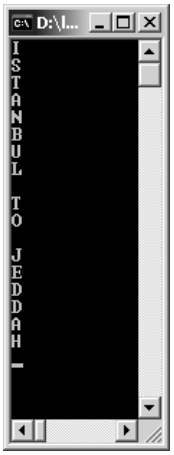
Sample program stringfunct.dpr in consoleapps folder

stringfunct.dpr

```
program stringfunct;
{$APPTYPE CONSOLE}
uses
  sysutils;

Var
    Line:string;
    i:integer;  ch:char;
begin
    Line:='Istanbul to Jeddah';
    for i:=1 to length(Line) do
    begin
       ch:=Line[i];
       ch:=upcase(ch);
       writeln(ch);
    end;
    readln
end.
```

19: 57 Insert Code

RESULT:



Basics of Language

Functions for Character and String Manipulations

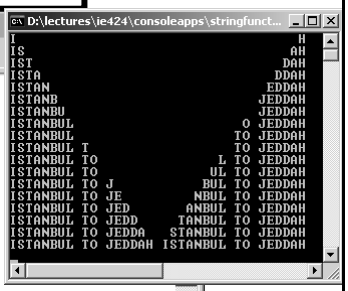
Sample program stringfunct2.dpr in consoleapps folder

stringfunct2.dpr

```
program stringfunct2;
uses
  sysutils;
Var
    Line,TempLine1,Blanks,TempLine2:string;
    i:integer;
begin
    Line:='Istanbul to Jeddah';
    for i:=1 to length(Line) do
    begin
     TempLine1:=UpperCase(copy(Line,1,i));
     TempLine2:=UpperCase(copy(Line,length(Line)-i+1,i));
     Blanks:=StringOfChar(' ',length(Line)*2-i+1-Length(TempLine1));
     writeln(Templine1,Blanks,TempLine2);
    end;
    readln
end.
```

17: 73 Insert Code

RESULT:



Functions for Character and String Manipulations

Question: Write a program which searches each line in a text file for a specific word or character given by the user and deletes the whole line if the specific word is matched the word in the searched line.

Example; The Word being searched: a

The file being searched

Question: Write a program which searches each line in a text file for a specific word or character given by the user and deletes the whole line if the specific word is matched the word in the searched line.

The resulting file

deletes the whole line if the specific word is line.

All lines containing "a" have been deleted in the file.

Functions for Character and String Manipulations

Question: Given a word by the user, write a computer program Which takes this word as input and counts the numbers, alphabetic and other chars in the input word and writes the frequencies for each.

Example; Word: 'Anka23[]andF2\\'

Frequency of alphabetic characters =8
Frequency of Numbers =3
Other Characters =4

Basics of Language

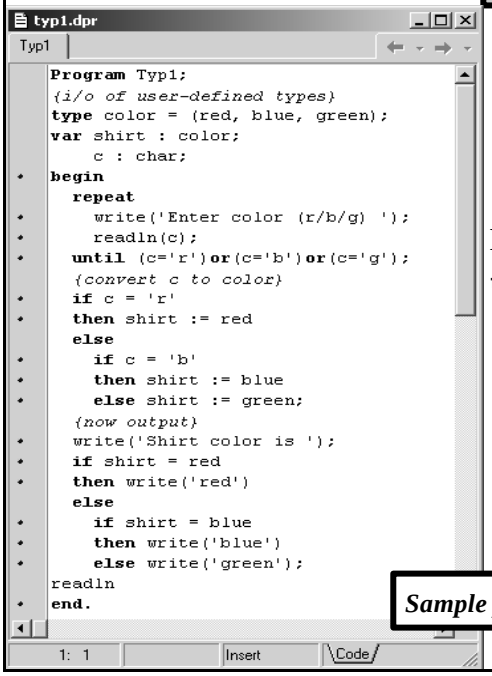
New Data Types can be constructed via the use of TYPE and Record

Basics of Language

The simplest usage

```
type TValue = Real;

var
  X: Real;
  Y: TValue;
```

User Defined Data Types




```
type recordTypeName = record

  fieldList1: type1;
  ...
  fieldListn: typen;
end;
```

Example

```
type

 TDateRec = record
  Year: Integer;
  Month: (Jan, Feb, Mar, Apr, May, Jun
        Jul, Aug, Sep, Oct, Nov, Dec);
  Day: 1..31;
 end;
```

Define as Variable

```
var Record1, Record2: TDateRec;
```

Sample program typ1.dpr in consoleapps folder

User Defined Data Types

Example

```
type

 TDateRec = record
  Year: Integer;
  Month: (Jan, Feb, Mar, Apr, May, Jun,
        Jul, Aug, Sep, Oct, Nov, Dec);
  Day: 1..31;
 end;
```

Define as Variable

```
var Record1, Record2: TDateRec;
```

Usage

```
Record1.Year := 1904;
Record1.Month := Jun;
Record1.Day := 16;
```

OR

```
with Record1 do
begin
  Year := 1904;
  Month := Jun;
  Day := 16;
end;
```

Example

```
type

  TEmployee = record
  FirstName, LastName : string[40];
  BirthDate: TDate;
  case Salaried: Boolean of
    True: (AnnualSalary: Currency);
    False: (HourlyWage: Currency);
end;

Var
    Employee:TEmployee;
```


Example

```
  type

    TShapeList = (Rectangle, Triangle, Circle, Ellipse, Other);
    TFigure = record
      case TShapeList of
        Rectangle: (Height, Width: Real);
        Triangle: (Side1, Side2, Angle: Real);
        Circle: (Radius: Real);
        Ellipse, Other: ();
    end;


  Var
      Figure:TFigure;

  USAGE
  Figure.Rectangle.width:=30.0;
```

Basics of Language

User Defined Data Types

typ13.dpr

```
program typ13;
uses
  sysutils,controls;
type
  TPerson = record
  FirstName, LastName: string[40];
  BirthDate: TDate;
  case Citizen: Boolean of
    True: (Birthplace: string[40]);
    False: (Country: string[20];
            EntryPort: string[20];
            EntryDate, ExitDate: TDate);
  end;
Var
     Person:TPerson;
begin
  With Person do
  begin
   FirstName:='Gultekin';
   LastName:='Cetiner';
   Citizen:=false;
   EntryPort:='London';
   EntryDate:=StrToDate('6/6/1990');
  end;
  with person do
       if not Citizen then
           writeln(FirstName+' '+LastName+
                   ' made entry from '+EntryPort+
                   ' on '+DateToStr(EntryDate));
readln
end.
```

25: 1 Insert Code

D:\lectures\ie424\consoleapps\typ13.exe

```
Gultekin Cetiner made entry from London on 06/06/1990
```

Note: TDate is defined in controls and StrToDate/DateToStr is defined in sysutils lib.

Sample program typ13.dpr in consoleapps folder

Basics of Language

User Defined Data Types

typ2.dpr

```
Program Typ2;
{FOR, ORD, SUCC on user-defined types}
type day = (mond,tues,wedn,thurs,frid,
            sat,sund);
var predec,succes,today : day;
    ordinal : integer;
begin
  for today := mond to sund do
    begin
    ordinal := ord(today);
    if today = mond
    then writeln('PRED UNDEFINED ',
      'for first value - MOND')
    else predec := pred(today);
    if today = sund
    then writeln('SUCC UNDEFINED ',
      'for last value - SUND')
    else succes := succ(today);
    end;
readln
end.
```

24: 81 Insert Code

Sample program typ2.dpr in consoleapps folder

Basics of Language

Array (Array Variables)

typ4.dpr

Typ4

```
Program Typ4;
{i/o, one-dimensional array}
Type
    TMyArray=array [1..4] of integer;
var number :TMyArray;
    index : integer;
begin
  write('Enter 4 integers ');
  for index := 1 to 4 do
    readln(number[index]);
  writeln('Input in reverse');
  for index := 4 downto 1 do
    writeln(number[index]);
readln
end.
```

16: 59 Insert Code

Sample program typ4.dpr in consoleapps folder

```
Var
  number:array [1..4] of integer;
```


Basics of Language

Array (Array Variables)

typ6.dpr

Typ6

```
Program Typ6;
{two arrays, averages}
var mark : array [1..3,1..4] of integer;
    average : array [1..3] of integer;
    student, sum, test : integer;
begin
  {input data one student at a time}
  writeln('4 marks for each student');
  for student :=1 to 3 do
    begin
    writeln('Student number ',student);
    for test := 1 to 4 do
      readln(mark[student,test]);
    end;
    {calculate averages}
    for student := 1 to 3 do
      begin
      sum := 0;
      for test := 1 to 4 do
        sum := sum + mark[student,test];
      average[student] := sum div 4;
      end;
    {now display averages}
    writeln('Average marks');
    for student := 1 to 3 do
      writeln('Student ',student,
      ' has average ',average[student]);
readln
end.
```

30: 48 Insert Code

Sample program typ6.dpr in consoleapps folder

Basics of Language

Record Types

rec1.dpr

Rec1

```
Program Rec1;
{elementary use of records}
type tape = record
       name : string;
       duration : integer;
       end;
var tape1, tape2 : tape;
begin
  write('Name of tape 1? ');
  readln(tape1.name);
  write('Length (minutes)? ');
  readln(tape1.duration);
  tape2 := tape1;
  writeln('Tape 2: ',tape2.name,' ',
    tape2.duration);
  tape2.duration := tape1.duration + 20;
  tape2.name := 'sonata of drums';
  writeln(tape2.name,tape2.duration:5);
readln
end.
```

22: 71 Insert Code

Sample program rec1.dpr in consoleapps folder

Basics of Language

Record Types

rec2.dpr

Rec2

```
Program Rec2;
{array of records}
type item = record
       cost, count : integer;
       end;
var inv : array[1..3] of item;
    i, total : integer;
begin
  {read data}
  for i := 1 to 3 do
    begin writeln('Item ', i);
    write('Enter cost and count ');
    readln(inv[i].cost, inv[i].count);
    end;
    {calculate and display}
    total := 0;
    for i := 1 to 3 do
      total := total + inv[i].cost *
         inv[i].count;
    writeln('Total value is ',total);
readln
end.
```

23: 75 Insert Code

Sample program rec2.dpr in consoleapps folder

Basics of Language

Record Types

Sample program ***rec3.dpr*** *in consoleapps folder*

rec3.dpr

Rec3

```
Program Rec3;
{array of records, WITH}
type item = record
       cost, count : integer;
       end;
var inv : array[1..3] of item;
    i, total : integer;
begin
  {read data}
  for i := 1 to 3 do
    begin writeln('Item ', i);
    write('Enter cost and count ');
    with inv[i] do
      readln(cost, count);
    end;
    {calculate and display}
    total := 0;
    for i := 1 to 3 do
      with inv[i] do
        total := total + cost * count;
    writeln('Total value is ',total);
readln
end.
```

24: 69 Insert Code

Basics of Language

SUBROUTINES

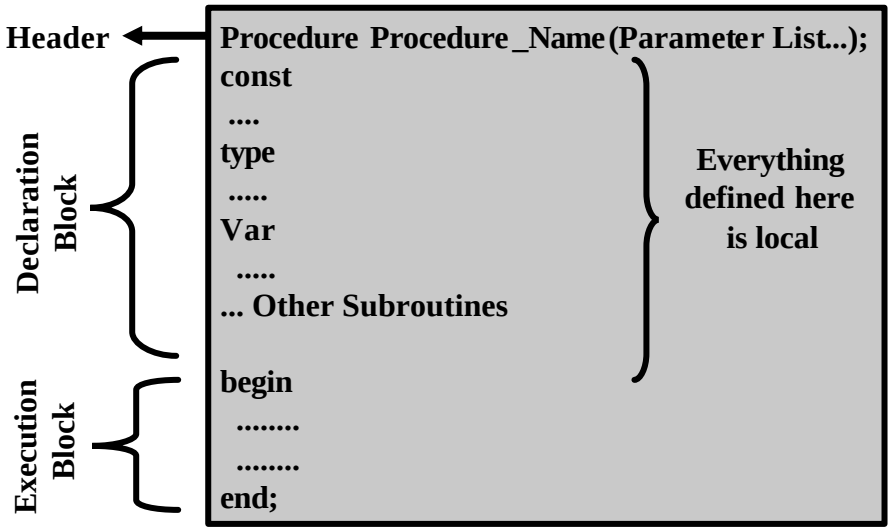
1. Procedure
2. Function

When you will define a *procedure* or *function*;

Decide

a. Name
b. Number of its parameters, their names and types
c. If it is a function then returning data type.

Syntax for a procedure is given below;



Examples

Call By Value

Call By Reference

Local

```
Procedure GetSquare(Input:real; Var Output:real);
Var
 Temporary:real;
begin
    Temporary:=Input*Input;
    Output:= Temporary;
end;
```

Call By Value: Used to enter/pass value into the subroutine
Call By Reference: Used to take a value out of subroutine.
Write VAR in front of variable.

Examples

```
Procedure GetSquare(Input:real; Var Output:real);
Var
 Temporary:real;
begin
    Temporary:=Input*Input;
    Output:= Temporary;
end;
```

Calling inside Program;

```
i:real;
.......
........
GetSquare(3,i);
....
```

You have to use a variable to get the value out

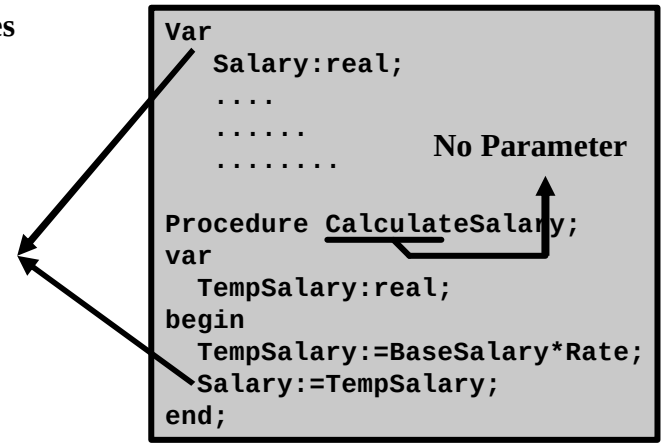
Examples

```
procedure ConvertNumberToString(N:Integer; var S:string);
var
  V: Integer;
begin
  V := Abs(N);
  S := '';
  repeat
    S := Chr(V mod 10 + Ord('0')) + S;
    V := V div 10;
  until V = 0;
  if N < 0 then S := '-' + S;
end;
```

To call inside Program;
ConvertNumberToString(123,MyString);

Examples

Global Variable



To call from within the Program;
CalculateSalary;

Basics of Language

Procedure

proc1.dpr

Proc1

```
Program Proc1;
{declaration and call of procedures}
var count, i : integer;
Procedure Pluses;
begin
  writeln('++++++++++++++++++++++++');
  writeln;
end; {pluses}
Procedure Minuses;
begin
  writeln('------------------');
  writeln;
end; {minuses}
begin {main program}
  write('How many times? ');
  readln(count);
  pluses; minuses;
  for i := 1 to count do
    writeln('Pascal is GREAT');
  minuses; pluses; pluses;
readln
end.
```

21: 72 | Insert | Code

Sample program proc1.dpr in consoleapps folder

Basics of Language

Procedure

proc2.dpr

Proc2

```
Program Proc2;
{procedure with parameters}
Procedure Symbols(chara : char; count : integer);
var i : integer;
begin
  for i := 1 to count do
    write(chara);
    writeln;
end; {symbols}
begin {main body}
  symbols('+',23); symbols('-',19);
  writeln('Pascal is GREAT');
  symbols('-',19);
  symbols('+',23); symbols('+',23);
readln
end.
```

17: 63 | Modified | Insert | Code

Sample program proc2.dpr in consoleapps folder

Basics of Language

Procedure

proc3.dpr

Proc3

```
Program Proc3;
{scope of variables}
var i, j : integer;
Procedure Stars(start,number:integer);
var i : integer;
begin
  writeln(number,' stars in field ',j+start-1);
  write(' ':start-1);
  for i := 1 to number do write('*');
  writeln;
end;
begin {main program}
  write('How many stars? '); readln(j);
  write('Total field width? '); readln(i);
  if i > j then
    begin
      stars(i-j+1,j);
      write(j,' stars, field width ',i);
    end
  else write('Impossible');
readln
end.
```

22: 79 Insert \Code/

Sample program proc3.dpr in consoleapps folder

Basics of Language

Procedure

proc4.dpr

Proc4

```
Program Proc4;
{nested procedures and scope}
var a, b : integer; {global}
Procedure x;
var product : integer;
  Procedure y;
  {multiplication}
      var j : integer; {local to y}
  begin
      product := 0;
      for j := 1 to b do product := product + a;
  end; {procedure y}
begin {body of x}
  writeln('Calculation of product');
  y;
  writeln('Product is ',product);
  write('Enter two integers '); readln(a, b);
end; {procedure x}
Procedure y;
begin
  if b<> 0 then writeln('Quotient is ',a div b)
     else writeln('No division by 0');
end; {procedure y}
begin {main program}
  write('Enter two integers '); readln(a, b);
  x; {product}  y; {divide}
readln
end.
```

29: 66 Insert \Code/

Sample program proc4.dpr in consoleapps folder

Procedure

proc5.dpr

Proc5

```
Program Proc5;
{recursive procedure}
Procedure Symb(chara : char;
          count : integer);
begin
  if count > 0
  then
    begin
      symb(chara,count-1);
      write(chara);
    end;
end; {symb}
begin {main program}
  symb('+',3); symb('-',3);
    symb('+',3); writeln;
  symb('-',3); symb('+',3);
    symb('-',3); writeln;
readln
end.
```

20: 73 Insert Code

Sample program proc5.dpr in consoleapps folder

Procedure

proc6.dpr

Proc6

```
Program Proc6;
{limitations of value parameters}
var x, y : integer;
Procedure Exchange(a, b : integer);
var buffer : integer;
begin
  buffer := a;
  a := b;
  b := buffer;
end; {exchange}
begin {main program}
  write('Enter two integers ');
  readln(x, y);
  exchange(x, y);
  writeln('Exchanged ',x,' ',y);
readln
end.
```

17: 69 Insert Code

Sample program proc6.dpr in consoleapps folder

Basics of Language

Procedure

proc7.dpr

Proc7

```
Program Proc7;
{calculating power with a PROCEDURE}
var b, e, result : integer;
Procedure Power(base,exponent:integer;
                var result : integer);
var count, product : integer;
begin
  product := 1;
  for count := 1 to exponent do
    product := product * base;
  result := product;
end; {power}
begin {main program}
  write('Enter base and exponent ');
  readln(b, e);
  if e > 0 then
    begin power(b, e, result);
    writeln('Result is ',result);
    end
  else writeln('No negative exponent');
readln
end.
```

22: 77 Insert Code

Sample program proc7.dpr in consoleapps folder

Basics of Language

Function

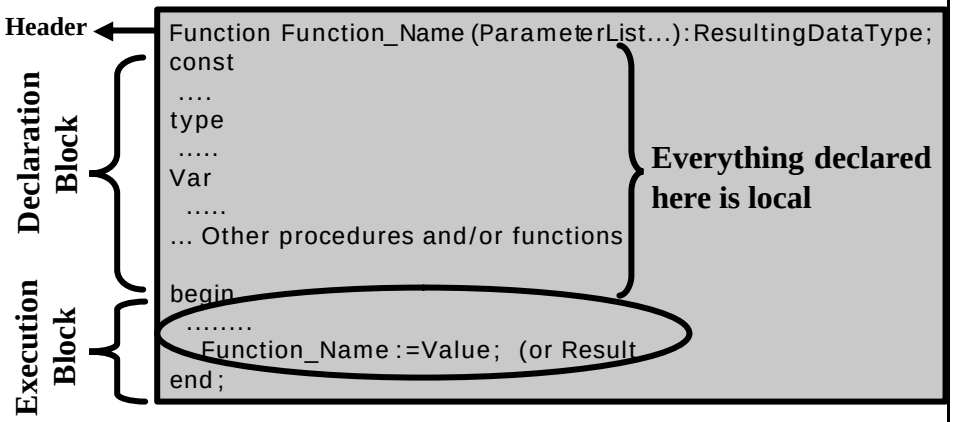
The difference between procedure and function is that the function itself returns a value whereas a procedure returns no value (like void function in C language)

e.g. y = sin(x) → **Function**

e.g. GetSin(x,SinResult); → **Procedure**

y := SinResult;

Syntax for a *function* is given below;



Examples

```
Function Divide (Number1,Number2: real):Real;
begin
   if Number2<>0 then Divide :=Number1/Number2;
end;
```

Usage

```
d:=Divide (2,3);
```

As seen, you can directly assign the function to a suitable value

proc8.dpr

Proc8

```
Program Proc8;
{calculating power as a function}
var b, e : integer;
Function Power(base,exponent:integer)
                : integer;
var count, product : integer;
begin
  {calculate value}
  product := 1;
  for count := 1 to exponent do
    product := product * base;
  {assign to function identifier}
  power := product;
end; {power}
begin {main program}
  write('Enter base and exponent ');
  readln(b, e);
  if e > 0
    then writeln('Result is ',
           power(b, e))
    else
      writeln('No negative exponents');
readln
end.
```

22: 78 Insert Code

Sample program proc8.dpr in consoleapps folder

proc10.dpr

proc10

```
Program Proc10;
{Boolean function}
var n : integer;
Function Divisible(x:integer):Boolean;
begin
  divisible := (x mod 3 = 0) or
    (x mod 4 = 0) or (x mod 5 = 0) or
    (x mod 7 = 0);
end; {divisible}
begin {main}
  n := 3;
  while (n <= 100) do
    begin
    if divisible(n) then writeln(n);
    n := n + 1;
    end;
readln
end.
```

18: 90 Insert Code

Sample program proc10.dpr in consoleapps folder

Function

Sample program ***proc11.dpr*** *in consoleapps folder*

proc11.dpr

proc11

```
Program Proc11;
{Boolean function in conditions}
var n : integer;
Function Divisible(x:integer):Boolean;
begin
  divisible := (x mod 3 = 0) or
    (x mod 4 = 0) or (x mod 5 = 0) or
    (x mod 7 = 0);
end; {divisible}
begin {main}
  n := 3;
  while divisible(n) do
    begin write(n:4); n := n + 1;
    end;
readln
end.
```

16: 95 Insert Code

Function

Sample program ***proc12.dpr*** *in consoleapps folder*

proc12.dpr

Proc12

```
Program Proc12;
{example of char function}
var c : char;
Function To_caps(chara:char):char;
begin
  if (chara >= 'a') and (chara <= 'z')
  then {convert to code, shift, convert}
    to_caps := chr(ord(chara) +
               ord('A') - ord('a'))
    else to_caps := chara;
end; {to_caps}
begin {main}
  repeat
    write('Type any character,',
          ' $ to stop ');
    readln(c);
    writeln(to_caps(c));
  until c = '$';
readln
end.
```

20: 76 Insert Code

Sample program proc14.dpr in consoleapps folder