

Chapter
2

**PROGRAMMING
BASICS**

*Get on the
Fast Track!*



TM

**SYS-ED/
COMPUTER
EDUCATION
TECHNIQUES, INC.**

Objectives

You will learn:

- C Programs and units.
- C Features of the Object Pascal language.
- C How to add comments.
- C Parts of programs and units.



1 General Syntax of a Program

Pascal is a structured programming language. This feature requires that specific program parts appear in a certain order.

General Program Syntax:

```
Program <name>;  
  
Uses <list of units>;  
  
Const <list of constants>;  
  
Type <list of data user-defined types>;  
  
Var <list of variables>;  
  
<list of functions and/or procedures>  
  
Begin  
    <statements>  
End.
```

Example:

```
Program Sum;  
  
Uses WinCrt;  
  
Var i, j : Integer;  
  
Begin  
    I    := 5;  
    j    := I + I;  
    Writeln(I, ' doubled = ', j);  
End.
```

1.1 Parts of a Pascal Program

Part	Explanation
Program	Every Pascal program starts with the keyword program followed by the program's name.
Uses	The Uses keyword lists the names of the library units that are employed by the program. These library units provide important support and operations to the main program.
Const	The Const keyword lists the names of constants that are defined in the main program. Constants are names that have fixed values.
Type	The Type keyword lists the name of the user-defined types that are defined in the main program. Pascal empowers you to define your own data types that represents specific information.
Var	The Var keyword lists the names of variables that are defined in the main program. Variables store information that can be updated.
Begin...End	<p>The main program may also define its own set of procedures and functions, which are called in the main begin-end block. These procedures and functions (which are sometimes collectively called routines) are semi-independent program components. These routines have access to the constants, types, and variables defined in the main program. In addition, the routines can access what the units have to offer.</p> <p>The Begin and End keywords define the block of statements that start executing first. There should be at least one statement in this block. Interestingly, the Pascal compiler does not object to a Begin-End block that has no statements!</p>

2 Comments

Comments can be placed inside pairs of curly braces or sets of (* and *) characters. The rule is that matching comment-enclosing characters must be used.

The following comment is legal:

```
{ This line will be ignored by the compiler }
```

By contrast, the following comment is illegal:

```
{ This line will cause an error condition * }
```

Single-line comments can be created by putting two slashes (//) in front of the comment text. The compiler then ignores everything until the end of the line.

```
sCity:='New York'; //Initialize sCity
```

3 Units

Pascal uses units to create reusable library modules. These modules empower developers to create code that can be used in different programs without having to retype the code.

A unit is a supplier of constants, data types, variables, and routines. Units empower software developers to create large programs that can be easily maintained and updated.

3.1 Unit Syntax

A Pascal unit has a unit heading followed by three parts:

C the interface.

The interface part (also called interface section) tells the compiler which parts of the unit are accessible to the unit's clients (that is, to other units and programs).

C the implementation.

The implementation part (also called the implementation section) contains local declarations and the detailed implementation of procedures and functions.

C the initialization.

The initialization part contains statements that allow the unit's exported variables to be initialized.

The unit heading specifies the unit's name, which appears in the uses clauses of other units and programs.

Syntax:

Unit name;

Interface

Uses <list of units>;

Const <list of exported constants>;

Type <list of exported data user-defined types>;

Var <list of exported variables >;

<list of declarations of exported functions and/or procedures>

Implementation

Uses <list of units>;

Const <list of local constants>;

Type <list of local data user-defined types>;

Var <list of local variables>;

<implementation of functions and/or procedures>

[Begin

<initializing statements>]

End.

Example:

```
Unit Math;
```

```
Interface
```

```
function Cube(X : Real) : Real;  
function MulInverse (X : Real) : Real;  
function SecondDegree (A, B, C, X : Real) : Real;  
function ThirdDegree (A, B, C, D, X : Real ) : Real;
```

```
Implementation
```

```
function Cube(X : Real) : Real;  
Begin  
    Cube :=X * X * X;  
End;  
  
function MulInverse(X : Real) : Real;  
Begin  
    MulInverse := 1 / X;  
End;  
  
function SecondDegree (A, B, C, X : Real) : Real;  
Begin  
    SecondDegree := (A * X + B) * X + C;  
End;  
  
function ThirdDegree (A, B, C, D, X : Real) : Real;  
Begin  
    ThirdDegree := ((A * X + B) * X + C) * X + D;  
End;  
End.
```

If the unit has no initializing code, there is no need to write the `Begin` keyword. Instead, only the final `End` keyword is required.

3.2 Programs versus Units

Every Pascal application has one and only one main program part or module. The main program may use the exported parts of one or more units. These units may use the exported parts of yet another set of units.

An application can be made up of a hierarchy of low-level (that is independent) units, medium-level units, high-level units, and the program part.

3.3 Interface Section

This section looks in more detail at the interface section of a library unit. The interface section of a unit specifies what items in the unit are accessible to other units and programs. These items are exportable.

3.4 uses Clause

The uses clause tells the compiler that the unit employs the exportable parts of other units. The uses clause lists the supporting units in a comma-delimited list.

3.5 const Clause

The const clause lists the names of identifiers that have fixed values associated with them. These values can be integers, floating-point types, strings, and other predefined or user-defined data types.

Syntax for Declaring a Constant:

```
Const  
    ConstantName = value;
```

Examples:

```
Const  
    SCHOOL = 'Sys-Ed' ;  
    LANGUAGE = 'Object Pascal';  
    InchPerFoot = 12;  
    FeetPerYard = 3;  
    InchPerYard = FeetPerYard * InchPerFoot;  
    PI = 3.1416;
```

3.6 type Clause

The type clause empowers the declaration of data types by programmers. These new data types are either records or classes. A record is a collection of fields that represent an item.

Syntax for Declaring a Record:

```
Type  
    recordName = Record  
        field1 : typeField1;  
        field2 : typeField2;  
        <other fields>  
end;
```

The declaration of a user-defined record starts by defining the name of the record type. The fields that make up the record definition have predefined data types and/or previously declared record types.

Although the names of record types must be unique in a program, the record types can have fields with similar names.

Examples:

```
type
  TDate = record
    Day, Month, Year : Integer;
  end;
  TDuration = record
    Start, Finish, : TDate
  end;
```