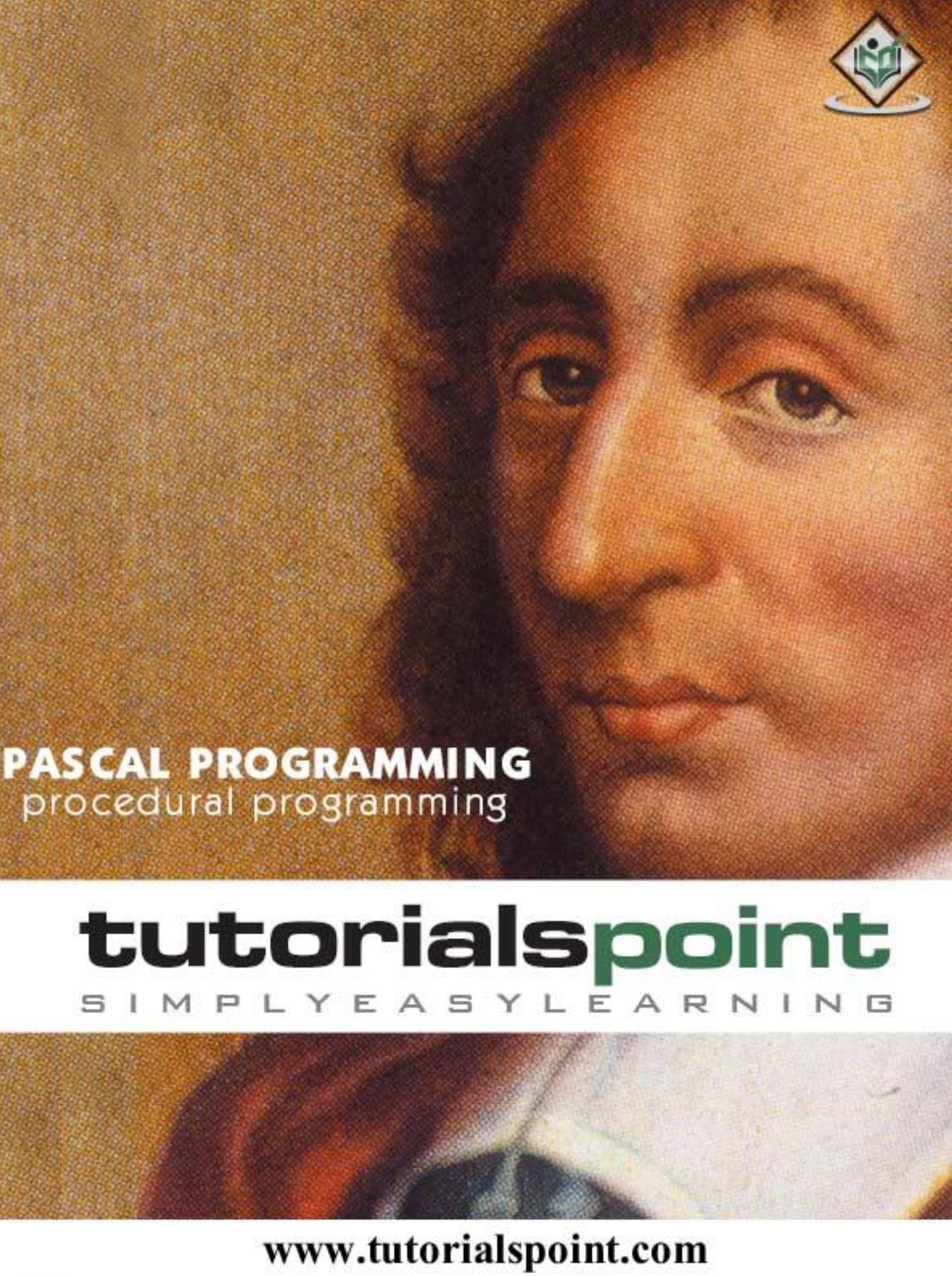


PASCAL PROGRAMMING

procedural programming

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

Pascal is a procedural programming language, designed in 1968 and published in 1970 by Niklaus Wirth and named in honor of the French mathematician and philosopher Blaise Pascal. Pascal runs on a variety of platforms, such as Windows, Mac OS, and various versions of UNIX/Linux.

This tutorial will give you great understanding of Pascal to proceed with Delphi and other related frameworks etc.

Audience

This tutorial is designed for Software Professionals who are willing to learn Pascal Programming Language in simple and easy steps. This tutorial will give you great understanding on Pascal Programming concepts, and after completing this tutorial, you will be at intermediate level of expertise from where you can take yourself to higher level of expertise.

Prerequisites

Before proceeding with this tutorial you should have a basic understanding of software basic concepts like what is source code, compiler, text editor, and execution of programs, etc. If you already have understanding on any other computer programming language, then it will be an added advantage to proceed.

Copyright & Disclaimer

Table of Contents

1. OVERVIEW

Pascal is a general-purpose, high-level language that was originally developed by Niklaus Wirth in the early 1970s. It was developed for teaching programming as a systematic discipline and to develop reliable and efficient programs.

Pascal is Algol-based language and includes many constructs of Algol. Algol 60 is a subset of Pascal. Pascal offers several data types and programming structures. It is easy to understand and maintain the Pascal programs.

Pascal has grown in popularity in the teaching and academics arena for various reasons:

- Easy to learn.
- Structured language.
- It produces transparent, efficient, and reliable programs.
- It can be compiled on a variety of computer platforms.

Features of the Pascal Language

Pascal has the following features:

- Pascal is a strongly typed language.
- It offers extensive error checking.
- It offers several data types like arrays, records, files, and sets.
- It offers a variety of programming structures.
- It supports structured programming through functions and procedures.
- It supports object oriented programming.

Facts about Pascal

- The Pascal language was named for Blaise Pascal, French mathematician and pioneer in computer development.

- Niklaus Wirth completed development of the original Pascal programming language in 1970.
- Pascal is based on the block structured style of the Algol programming language.
- Pascal was developed as a language suitable for teaching programming as a systematic discipline, whose implementations could be both reliable and efficient.
- The ISO 7185 Pascal Standard was originally published in 1983.
- Pascal was the primary high-level language used for development in the Apple Lisa, and in the early years of the Mac.
- In 1986, Apple Computer released the first Object Pascal implementation, and in 1993, the Pascal Standards Committee published an Object-Oriented Extension to Pascal.

Why to use Pascal?

Pascal allows the programmers to define complex structured data types and build dynamic and recursive data structures, such as lists, trees and graphs. Pascal offers features like records, enumerations, subranges, dynamically allocated variables with associated pointers and sets.

Pascal allows nested procedure definitions to any level of depth. This truly provides a great programming environment for learning programming as a systematic discipline based on the fundamental concepts.

Among the most amazing implementations of Pascal are:

- Skype
- Total Commander
- TeX
- Macromedia Captivate
- Apple Lisa
- Various PC Games
- Embedded Systems

2. ENVIRONMENT SET UP

There are several Pascal compilers and interpreters available for general use. Among these are:

- **Turbo Pascal:** provides an IDE and compiler for running Pascal programs on CP/M, CP/M-86, DOS, Windows, and Macintosh.

- **Delphi:** provides compilers for running Object Pascal and generates native code for 32- and 64-bit Windows operating systems, as well as 32-bit Mac OS X and iOS. Embarcadero is planning to build support for the Linux and Android operating system.

- **Free Pascal:** it is a free compiler for running Pascal and Object Pascal programs. Free Pascal compiler is a 32- and 64-bit Turbo Pascal and Delphi compatible Pascal compiler for Linux, Windows, OS/2, FreeBSD, Mac OS X, DOS, and several other platforms.

- **Turbo51:** it is a free Pascal compiler for the 8051 family of microcontrollers, with Turbo Pascal 7 syntax.

- **Oxygene:** it is an Object Pascal compiler for the .NET and Mono platforms.

- **GNU Pascal (GPC):** it is a Pascal compiler composed of a front end to GNU Compiler Collection.

We will be using Free Pascal in these tutorials. You can download Free Pascal for your operating system from the link: Download Free Pascal

Installing Free Pascal on Linux

The Linux distribution of Free Pascal comes in three forms:

- a **tar.gz** version, also available as separate files.
- a **.rpm** (Red Hat Package Manager) version.
- a **.deb** (Debian) version.

Installation code for the .rpm version:

```
rpm -i fpc-X.Y.Z-N.ARCH.rpm
```

Where X.Y.Z is the version number of the .rpm file, and ARCH is one of the supported architectures (i386, x86_64, etc.).

Installation code for the Debian version (like Ubuntu):

```
dpkg -i fpc-XXX.deb
```

Where XXX is the version number of the .deb file. For details read: Free Pascal Installation Guide

Installing Free Pascal on Mac

If you use Mac OS X, the easiest way to use Free Pascal is to download the Xcode development environment from Apple's web site and follow the simple installation instructions. Once you have Xcode setup, you will be able to use the Free Pascal compiler.

Installing Free Pascal on Windows

For Windows, you will download the Windows installer, setup.exe. This is a usual installation program. You need to take the following steps for installation:

- Select a directory.
- Select parts of the package you want to install.
- Optionally choose to associate the .pp or .pas extensions with the Free Pascal IDE.

For details read: Free Pascal Installation Guide

Text Editor

This will be used to type your program. Examples of few editors include Windows Notepad, OS Edit command, Brief, Epsilon, EMACS, and vim or vi.

Name and version of text editor can vary on different operating systems. For example, Notepad will be used on Windows and vim or vi can be used on windows as well as Linux or UNIX.

The files you create with your editor are called source files and contain program source code. The source files for Pascal programs are typically named with the extension **.pas**.

Before starting your programming, make sure you have one text editor in place and you have enough experience to write a computer program, save it in a file, compile it and finally execute it.

3. PROGRAM STRUCTURE

Before we study basic building blocks of the Pascal programming language, let us look a bare minimum Pascal program structure so that we can take it as a reference in upcoming chapters.

Pascal Program Structure

A Pascal program basically consists of the following parts:

- Program name
- Uses command
- Type declarations
- Constant declarations
- Variables declarations
- Functions declarations
- Procedures declarations
- Main program block
- Statements and Expressions within each block
- Comments

Every Pascal program generally have a heading statement, a declaration and an execution part strictly in that order. Following format shows the basic syntax for a Pascal program:

```
program {name of the program}
uses {comma delimited names of libraries you use}
const {global constant declaration block}
var {global variable declaration block}
```

```
function {function declarations, if any}
{ local variables }
begin
...
end;

procedure { procedure declarations, if any}
{ local variables }
begin
...
end;

begin { main program block starts}
...
end. { the end of main program block }
```

Pascal Hello World Example

Following is a simple pascal code that would print the words "Hello, World!":

```
program HelloWorld;
uses crt;

(* Here the main program block starts *)
begin
   writeln('Hello, World!');
   readkey;
end.
```

Let us look various parts of the above program:

- The first line of the program **program HelloWorld;** indicates the name of the program.

- The second line of the program **uses crt;** is a preprocessor command, which tells the compiler to include the crt unit before going to actual compilation.

- The next lines enclosed within begin and end statements are the main program block. Every block in Pascal is enclosed within a **begin** statement and an **end** statement. However, the end statement indicating the end of the main program is followed by a full stop (.) instead of semicolon (;).

- The **begin** statement of the main program block is where the program execution begins.

- The lines within **(*...*)** will be ignored by the compiler and it has been put to add a **comment** in the program.

- The statement **writeln('Hello, World!');** uses the writeln function available in Pascal which causes the message "Hello, World!" to be displayed on the screen.

- The statement **readkey;** allows the display to pause until the user presses a key. It is part of the crt unit. A unit is like a library in Pascal.

- The last statement **end.** ends your program.

Compile and Execute Pascal Program

- Open a text editor and add the above-mentioned code.

- Save the file as *hello.pas*

- Open a command prompt and go to the directory, where you saved the file.

- Type fpc hello.pas at command prompt and press enter to compile your code.

- If there are no errors in your code, the command prompt will take you to the next line and would generate **hello** executable file and **hello.o** object file.

- Now, type **hello** at command prompt to execute your program.

- You will be able to see "Hello World" printed on the screen and program waits till you press any key.

```
$ fpc hello.pas

Free Pascal Compiler version 2.6.0 [2011/12/23] for x86_64

Copyright (c) 1993-2011 by Florian Klaempfl and others
```

```
Target OS: Linux for x86-64
Compiling hello.pas
Linking hello
8 lines compiled, 0.1 sec

$ ./hello
Hello, World!
```

Make sure that free pascal compiler **fpc** is in your path and that you are running it in the directory containing source file hello.pas.

4. BASIC SYNTAX

You have seen a basic structure of pascal program, so it will be easy to understand other basic building blocks of the pascal programming language.

Variable

A variable definition is put in a block beginning with a **var** keyword, followed by definitions of the variables as follows:

```
var
A_Variable, B_Variable ... : Variable_Type;
```

Pascal variables are declared outside the code-body of the function which means they are not declared within the **begin** and **end** pairs, but they are declared after the definition of the procedure/function and before the **begin** keyword. For global variables, they are defined after the program header.

Functions/Procedures

In Pascal, a **procedure** is set of instructions to be executed, with no return value and a **function** is a procedure with a return value. The definition of function/procedures will be as follows:

```
Function Func_Name(params...) : Return_Value;
Procedure Proc_Name(params...);
```

Comments

The multiline comments are enclosed within curly brackets and asterisks as {* ... *}. Pascal allows single-line comment enclosed within curly brackets { ... }.

```
{* This is a multi-line comments
   and it will span multiple lines. *}

{ This is a single line comment in pascal }
```

Case Sensitivity

Pascal is a case non-sensitive language, which means you can write your variables, functions, and procedure in either case. Like variables A_Variable, a_variable and A_VARIABLE have same meaning in Pascal.

Pascal Statements

Pascal programs are made of statements. Each statement specifies a definite job of the program. These jobs could be declaration, assignment, reading data, writing data, taking logical decisions, transferring program flow control, etc.

For example:

```
readln (a, b, c);

s := (a + b + c)/2.0;

area := sqrt(s * (s - a)*(s-b)*(s-c));

writeln(area);
```

Reserved Words in Pascal

The statements in Pascal are designed with some specific Pascal words, which are called the reserved words. For example, the words, program, input, output, var, real, begin, readline, writeline and end are all reserved words. Following is a list of reserved words available in Pascal.

and	array	begin	case	const
div	do	downto	else	end
file	for	function	goto	if
in	label	mod	nil	not
of	or	packed	procedure	program
record	repeat	set	then	to
type	until	var	while	with

Character Set and Identifiers in Pascal

The Pascal character set consists of:

- All upper case letters (A-Z)
- All lower case letters (a-z)
- All digits (0-9)
- Special symbols - + * / := , . ;. () [] = {} ` white space

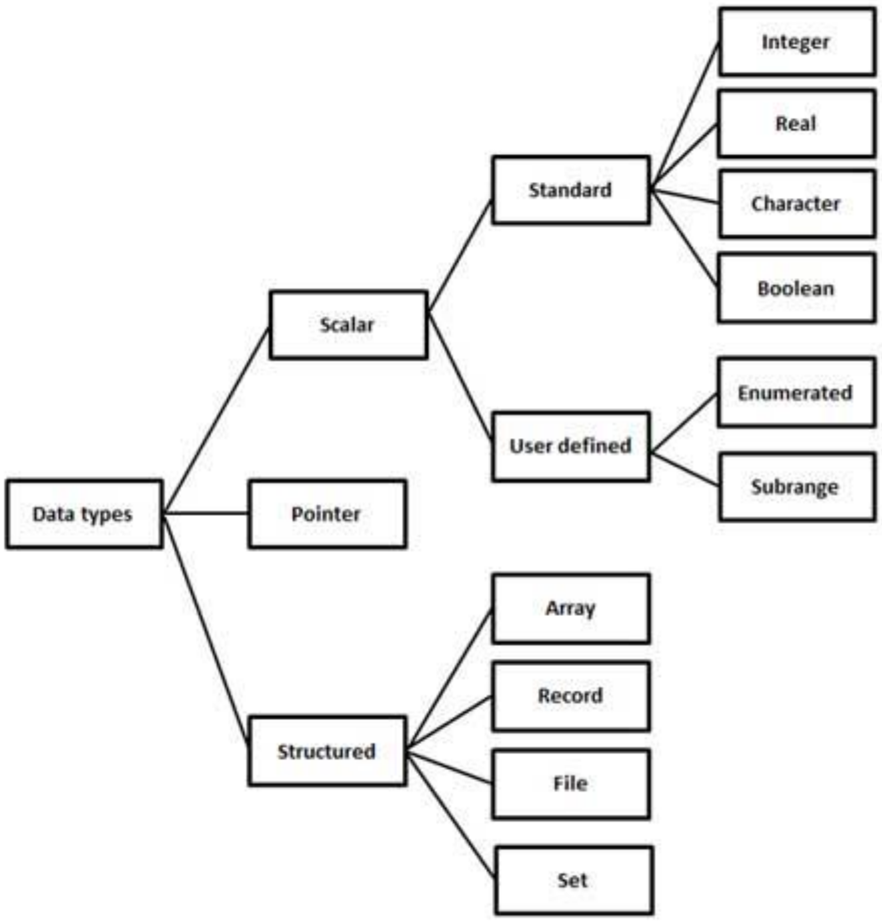
The entities in a Pascal program like variables and constants, types, functions, procedures and records, etc., have a name or identifier. An identifier is a sequence of letters and digits, beginning with a letter. Special symbols and blanks must not be used in an identifier.

5. DATA TYPES

Data types of an entity indicates the meaning, constraints, possible values, operations, functions and mode of storage associated with it. Integer, real, boolean and character types are referred as standard data types. They can be categorized as scalar, pointer and structured data types. Examples of scalar data types are integer, real, boolean, character, subrange and enumerated. Structured data types are made of the scalar types; for example, arrays, records, files and sets. We will discuss the pointer data types later.

Pascal Data Types

Pascal data types can be summarized as below in the following diagram:



Type Declarations

The type declaration is used to declare the data type of an identifier. Syntax of type declaration is:

```
type-identifier-1, type-identfier-2 = type-specifier;
```

For example, the following declaration defines the variables days and age as integer type, yes and true as boolean type, name and city as string type, fees and expenses as real type.

```
type
days, age = integer;
yes, true = boolean;
name, city = string;
fees, expenses = real;
```

Integer Types

Following table gives you details about standard integer types with its storage sizes and value ranges used in Object Pascal:

Type	Minimum	Maximum	Format
Integer	-2147483648	2147483647	signed 32-bit
Cardinal	0	4294967295	unsigned 32-bit
Shortint	-128	127	signed 8-bit
Smallint	-32768	32767	signed 16-bit
Longint	-2147483648	2147483647	signed 32-bit
Int64	-2^63	2^63 - 1	signed 64-bit
Byte	0	255	unsigned 8-bit
Word	0	65535	unsigned 16-bit

Longword	0	4294967295	unsigned 32-bit

Constants

Use of constants makes a program more readable and helps to keep special quantities at one place in the beginning of the program. Pascal allows *numerical, logical, string* and *character* constants. Constants can be declared in the declaration part of the program by specifying the **const** declaration.

Syntax of constant type declaration is as follows:

```
const

Identifier = contant_value;
```

Following are some examples of constant declarations:

```
VELOCITY_LIGHT = 3.0E=10;

PIE = 3.141592;

NAME = 'Stuart Little';

CHOICE = yes;

OPERATOR = '+';
```

All constant declarations must be given before the variable declaration.

Enumerated types

Enumerated data types are user-defined data types. They allow values to be specified in a list. Only *assignment* operators and *relational* operators are permitted on enumerated data type. Enumerated data types can be declared as follows:

```
type

enum-identifier = (item1, item2, item3, ... )
```

Following are some examples of enumerated type declarations:

```
type

SUMMER = (April, May, June, July, September);

COLORS = (Red, Green, Blue, Yellow, Magenta, Cyan, Black, White);

TRANSPORT = (Bus, Train, Airplane, Ship);
```

The order in which the items are listed in the domain of an enumerated type defines the order of the items. For example, in the enumerated type SUMMER, April comes before May, May comes before June, and so on. The domain of enumerated type identifiers cannot consist of numeric or character constants.

Subrange Types

Subrange types allow a variable to assume values that lie within a certain range. For example, if the *age* of voters should lie between 18 to 100 years, a variable named age could be declared as:

```
var
age: 18 ... 100;
```

We will look at variable declaration in detail in the next section. You can also define a subrange type using the type declaration. Syntax for declaring a subrange type is as follows:

```
type
subrange-identifier = lower-limit ... upper-limit;
```

Following are some examples of subrange type declarations:

```
const
P = 18;
Q = 90;
type
Number = 1 ... 100;
Value = P ... Q;
```

Subrange types can be created from a subset of an already defined enumerated type, For example:

```
type
months = (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec);
Summer = Apr ... Aug;
Winter = Oct ... Dec;
```

6. VARIABLE TYPES

A variable is nothing but a name given to a storage area that our programs can manipulate. Each variable in Pascal has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.

The name of a variable can be composed of letters, digits, and the underscore character. It must begin with either a letter or an underscore. Pascal is **not case-sensitive**, so uppercase and lowercase letters mean same here. Based on the basic types explained in previous chapter, there will be following basic variable types:

Basic Variables in Pascal

Type	**Description**
Character	Typically a single octet (one byte). This is an integer type.
Integer	The most natural size of integer for the machine.
Real	A single-precision floating point value.
Boolean	Specifies true or false logical values. This is also an integer type.
Enumerated	Specifies a user-defined list.
Subrange	Represents variables, whose values lie within a range.
String	Stores an array of characters.

Pascal programming language also allows defining various other types of variables, which we will cover in subsequent chapters like Pointer, Array, Records, Sets, and Files, etc. For this chapter, let us study only basic variable types.

Variable Declaration in Pascal

All variables must be declared before we use them in Pascal program. All variable declarations are followed by the *var* keyword. A declaration specifies a list of variables, followed by a colon (:) and the type. Syntax of variable declaration is:

```
var

variable_list : type;
```

Here, type must be a valid Pascal data type including character, integer, real, boolean, or any user-defined data type, etc., and variable_list may consist of one or more identifier names separated by commas. Some valid variable declarations are shown here:

```
var

age, weekdays : integer;

taxrate, net_income: real;

choice, isready: boolean;

initials, grade: char;

name, surname : string;
```

In the previous tutorial, we have discussed that Pascal allows declaring a type. A type can be identified by a name or identifier. This type can be used to define variables of that type. For example,

```
type

days, age = integer;

yes, true = boolean;

name, city = string;

fees, expenses = real;
```

Now, the types so defined can be used in variable declarations:

```
var

weekdays, holidays : days;

choice: yes;

student_name, emp_name : name;

capital: city;
```

```
cost: expenses;
```

Please note the difference between *type* declaration and *var* declaration. Type declaration indicates the category or class of the types such as integer, real, etc., whereas the variable specification indicates the type of values a variable may take. You can compare *type* declaration in Pascal with *typedef* in C. Most importantly, the variable name refers to the memory location where the value of the variable is going to be stored. This is not so with the type declaration.

Variable Initialization in Pascal

Variables are assigned a value with a colon and the equal sign, followed by a constant expression. The general form of assigning a value is:

```
variable_name := value;
```

By default, variables in Pascal are not initialized with zero. They may contain rubbish values. So it is a better practice to initialize variables in a program. Variables can be initialized (assigned an initial value) in their declaration. The initialization is followed by the **var** keyword and the syntax of initialization is as follows:

```
var
variable_name : type = value;
```

Some examples are:

```
age: integer = 15;
taxrate: real = 0.5;
grade: char = 'A';
name: string = 'John Smith';
```

Let us look at an example, which makes use of various types of variables discussed so far:

```
program Greetings;
const
message = ' Welcome to the world of Pascal ';
type
name = string;
var
```

```
firstname, surname: name;
begin
   writeln('Please enter your first name: ');
   readln(firstname);
   writeln('Please enter your surname: ');
   readln(surname);
   writeln;
   writeln(message, ' ', firstname, ' ', surname);
end.
```

When the above code is compiled and executed, it produces the following result:

```
Please enter your first name:
John
Please enter your surname:
Smith
Welcome to the world of Pascal John Smith
```

Enumerated Variables

You have seen how to use simple variable types like integer, real and boolean. Now, let's see variables of enumerated type, which can be defined as:

```
var
var1, var2, ...  : enum-identifier;
```

When you have declared an enumerated type, you can declare variables of that type. For example,

```
type
months = (January, February, March, April, May, June, July, August,
September, October, November, December);
Var
m: months;
...
M := January;
```

The following example illustrates the concept:

```
program exEnumeration;
type
beverage = (coffee, tea, milk, water, coke, limejuice);
var
drink:beverage;
begin
   writeln('Which drink do you want?');
   drink := limejuice;
   writeln('You can drink ', drink);
end.
```

When the above code is compiled and executed, it produces the following result:

```
Which drink do you want?
You can drink limejuice
```

Subrange Variables

Subrange variables are declared as:

```
var
subrange-name : lowerlim ... uperlim;
```

Examples of subrange variables are:

```
var
marks: 1 ... 100;
grade: 'A' ... 'E';
age: 1 ... 25;
```

The following program illustrates the concept:

```
program exSubrange;
var
marks: 1 .. 100;
```

```
grade: 'A' .. 'E';
begin
   writeln( 'Enter your marks(1 - 100): ');
   readln(marks);
   writeln( 'Enter your grade(A - E): ');
   readln(grade);
   writeln('Marks: ' , marks, ' Grade: ', grade);
end.
```

When the above code is compiled and executed, it produces the following result:

```
Enter your marks(1 - 100):
100
Enter your grade(A - E):
A
Marks: 100 Grade: A
```

7. CONSTANTS

A constant is an entity that remains unchanged during program execution. Pascal allows only constants of the following types to be declared:

- Ordinal types
- Set types
- Pointer types (but the only allowed value is Nil).
- Real types
- Char
- String

Declaring Constants

Syntax for declaring constants is as follows:

```
const
identifier = constant_value;
```

The following table provides examples of some valid constant declarations:

Constant Type	Examples
Ordinal(Integer)type constant	valid_age = 21;
Set type constant	Vowels = set of (A,E,I,O,U);
Pointer type constant	P = NIL;
Real type constant	e = 2.7182818; velocity_light = 3.0E+10;
Character type constant	Operator = '+';

String type constant	president = 'Johnny Depp';

The following example illustrates the concept:

```
program const_circle (input,output);
const
PI = 3.141592654;
var
r, d, c : real;   {variable declaration: radius, dia, circumference}
begin
   writeln('Enter the radius of the circle');
   readln(r);
   d := 2 * r;
   c :=  PI * d;
   writeln('The circumference of the circle is ',c:7:2);
end.
```

When the above code is compiled and executed, it produces the following result:

```
Enter the radius of the circle
23
The circumference of the circle is 144.51
```

Observe the formatting in the output statement of the program. The variable c is to be formatted with total number of digits 7 and 2 digits after the decimal sign.