Start programming

using

Object Pascal

Written by: Motaz Abdel Azeem
Edited by: Pat Anderson, Jason Hackney

code.sd

18.June.2011

Introduction

This book is written for programmers who want to learn the Object Pascal Language. It is also suitable as a first programming book for new students and non-programmers. It illustrates programming techniques in general in addition to the Object Pascal Language.

The Object Pascal Language

The first appearance of the Pascal Language supporting Object Oriented programming was in 1983 by Apple computer company. After that Borland supported Object Oriented programming for their famous Turbo Pascal line.

Object Pascal is a general purpose hybrid (structured and object oriented programming) language. It can be used for a vast range of applications, like learning, game development, business applications, Internet applications, communication applications, tools development, and OS kernels.

Delphi

After the success of Turbo Pascal, Borland decided to port it to Windows and introduced component driven technology to it. Soon Delphi became the best RAD (Rapid Application Development) tool at that time.

The first version of Delphi was released in 1995 with a rich set of components and packages that supported Windows and Database applications development.

Free Pascal

After Borland dropped support for the Turbo Pascal line, the Free Pascal team started an open source project to write a compatible compiler for Turbo Pascal from scratch, and then make it compatible with Delphi. This time the Free Pascal compiler was targeting additional platforms and operating systems like Windows, Linux, Mac, ARM, and WinCE.

Version 1.0 of the Free Pascal compiler was released in July 2000.

Lazarus

Free Pascal is a compiler, and it lacks an Integrated Development Environment (IDE) similar to the

Delphi IDE for Windows. The Lazarus project was started to provide an IDE for Free Pascal. It provides a source code editor, debugger, and contains a lot of frameworks, packages, and component libraries similar to the Delphi IDE.

Version 1.0 of Lazarus has not been released yet, but there are a lot of applications developed with beta versions of Lazarus. A lot of volunteers write packages and components for Lazarus, and the community is growing.

Object Pascal features

Object Pascal is a very easy and readable language for beginners, its compilers are very fast, and the applications it produces are reliable, fast and can be compared with C, and C++. You can write robust and large applications with its IDEs (Lazarus and Delphi) without complexity.

Author: Motaz Abdel Azeem

I am graduated from Sudan University of Science and Technology in 1999. and I started learning Pascal as a second language after BASIC. Since then, I've been using it continuously, and I found it a very easy and powerful tool, specially after I studied C, and C++. Then I moved to Delphi. Since then I have been using Delphi and Lazarus for all my applications.

I live in Khartoum. My current job is Software Developer.

First Editor

Pat Anderson graduated from Western Washington State College in 1968 and Rutgers Law School in 1975. He works as the City Attorney of Snoqualmie, Washington. Pat began programming on a Radio Shack TRS-80 Model III in 1982 with the built-in BASIC interpreter, but soon discovered Turbo Pascal. He has owned all versions of Turbo Pascal from 4.0 to 7.0, and every version of Delphi from 1.0 to 4.0. Pat took a hiatus from programming from 1998 until 2009, when he came upon Free Pascal / Lazarus, which reignited his passion for programming.

Second Editor

Jason Hackney is a graduate of Western Michigan University's College of Aviation. He works full-time as a professional pilot for a power company based in southeast Michigan. Jason has been a casual programmer since his first exposure to the Commodore 64 around 1984. Briefly introduced to Turbo Pascal in 1990, he recently rekindled latent programming interest after discovering Linux, Lazarus, and Free Pascal.

License:

The License for this book is **Creative Commons**.

Environment for book examples

We will use Lazarus and Free Pascal for all the examples in this book. You can get the Lazarus IDE, including the Free Pascal compiler, from this site: http://lazarus.freepascal.org.

If you are using Linux, then you can get Lazarus from the software repository. In Ubuntu you can use the command:

```
sudo apt-get install lazarus
```

In Fedora you can use the command:

```
yum install lazarus
```

Lazarus is a free and open source application. And it is available on many platforms. Applications written in Lazarus can be re-compiled on another platform to produce executables for that platform. For example if you write an application using Lazarus in Windows, and you want to produce a Linux executable for that application, you only need to copy your source code to Lazarus under Linux, then compile it.

Lazarus produces applications that are native to each operating system, and it does not require any additional libraries or virtual machines. For that reason, it is easy to deploy and fast in execution.

Using Text mode

All examples in the first chapters of this book will be console applications (text mode applications/ command line applications), because they are easy to understand and standard. Graphical user interface applications will be introduced in later chapters.

Contents

Introduction.....2
The Object Pascal Language.....2
Delphi.....2
Free Pascal.....2
Lazarus.....2
Object Pascal features.....3
Author: Motaz Abdel Azeem.....3
First Editor.....3
Second Editor.....3
License:.....4
Environment for book examples.....4
Using Text mode 4

Chapter One

Language Basics

Our First Application.....10
Other examples.....12
Variables.....14
Sub types.....19
Conditional Branching.....20
The If condition.....20
Air-Conditioner program:.....20
Weight program.....22
Case .. of statement.....25
Restaurant program.....25
Restaurant program using If condition.....26
Students' Grades program.....27
Keyboard program.....27
Loops.....29
For loop.....29

Multiplication Table using for loop.....30
Factorial program.....31
Repeat Until loop.....32
Restaurant program using Repeat loop.....32
While loop.....34
Factorial program using while loop.....34
Strings.....36
Copy function.....39
Insert procedure.....40
Delete procedure.....41
Trim function.....41
StringReplace function.....42
Arrays.....44
Records.....47
Files.....49
Text files.....50
Reading text file program.....50
Creating and writing into text file.....52
Appending to a text file.....55
Add to text file program.....55
Random access files.....56
Typed files.....56
Marks program.....56
Reading student marks.....57
Appending student marks program.....58
Create and append student marks program.....59
Cars database program.....60
File copying.....62
Copy files using file of byte.....62
Untyped files.....64
Copy files using untyped files program.....64
Display file contents program.....66
Date and Time.....68

Date/time comparison....70
News recorder program....71
Constants....73
Fuel Consumption program....73
Ordinal types....75
Sets....77
Exception handling....79
Try except statement....79
Try finally....80
Raise an exception....81

Chapter Two

Structured Programming

Introduction....84
Procedures....84
Parameters....85
Restaurant program using procedures....86
Functions....87
Restaurant program using functions....88
Local Variables....89
News database application....90
Functions as input parameters....93
Procedure and function output parameters....94
Calling by reference....95
Units....97
Units in Lazarus and Free Pascal....99
Units written by the programmer....99
Hejri Calendar....100
Procedure and function Overloading....103
Default value parameters....104
Sorting....105
Bubble sort algorithm....105

Sorting students' marks.....107
Selection Sort algorithm.....109
Shell sort algorithm.....110
String sorting.....112
Sorting students name program.....112
Sort algorithms comparison.....113

Chapter Three

The Graphical User Interface

Introduction.....118
Our First GUI application.....118
Second GUI application.....123
ListBox application.....125
Text Editor Application.....126
News Application.....128
Application with a Second form.....129

Chapter Four

Object Oriented

Programming

Introduction.....131
First example: Date and Time.....131
News application in Object Oriented Pascal.....136
Queue Application.....142
Object Oriented File.....147
Copy files using TFileStream.....147
Inheritance.....148

Chapter One

Language Basics

Our First Application

After installing and running Lazarus, we can start a new program from the main menu:

Project/New Project/Program

We will get this code in the Source Editor window:

```
program Project1;

{$mode objfpc}{$H+}

uses
  {$IFDEF UNIX}{$IFDEF UseCThreads}
  cthreads,
  {$ENDIF}{$ENDIF}
  Classes
  { you can add units after this };

{$IFDEF WINDOWS}{$R project1.rc}{$ENDIF}

begin
end.
```

We can save this program by clicking File/Save from the main menu, and then we can name it, for example, first.lpi

Then we can write these lines between the *begin* and *end* statements:

```
  Writeln('This is Free Pascal and Lazarus');
  Writeln('Press enter key to close');
  Readln;
```

The complete source code will be:

```
program first;

{$mode objfpc}{$H+}

uses
  {$IFDEF UNIX}{$IFDEF UseCThreads}
  cthreads,
  {$ENDIF}{$ENDIF}
  Classes
  { you can add units after this };
```

```
{$IFDEF WINDOWS}{$R first.rc}{$ENDIF}

begin
  Writeln('This is Free Pascal and Lazarus');
  Writeln('Press enter key to close');
  Readln;
end.
```

The *Writeln* statement displays the text on the screen (Console window). *Readln* halts execution to let the user read the displayed text until he/she presses enter to close the application and return to the Lazarus IDE.

Then press *F9* to run the application or click the button:

After running the first program, we will get this output text:

```
This is Free Pascal and Lazarus
Press enter key to close
```

If we are using Linux, we will find a new file in a program directory called (first), and in Windows we will get a file named first.exe. Both files can be executed directly by double clicking with the mouse. The executable file can be copied to other computers to run without the need of the Lazarus IDE.

Notes

If the console application window does not appear, we can disable the debugger from the Lazarus menu:

Environment/Options/Debugger
In Debugger type and path select (None)

Other examples

In the previous program change this line:

```
  Writeln('This is Free Pascal and Lazarus');
```

to this one:

```
  Writeln('This is a number: ', 15);
```

Then press *F9* to run the application.

You will get this result:

```
This is a number: 15
```

Change the previous line as shown below, and run the application each time:

Code:

```
  Writeln('This is a number: ', 3 + 2);
```

Output:

```
This is a number: 5
```

Code:

```
  Writeln('5 * 2 = ', 5 * 2);
```

Output:

```
5 * 2 = 10
```

Code:

```
  Writeln('This is real number: ', 7.2);
```

Output:

```
This is real number: 7.2000000000000E+0000
```

Code:

```
  Writeln('One, Two, Three : ', 1, 2, 3);
```

Output:

```
One, Two, Three : 123
```

Code:

```
  Writeln(10, ' * ',  3 , ' = ', 10 * 3);
```

Output:

```
10 * 3 = 30
```

We can write different values in the *Writeln* statement each time and see the result. This will help us understand it clearly.

Variables

Variables are data containers. For example, when we say that X = 5, that means X is a variable, and it contains the value 5.

Object Pascal is a strongly typed language, which means we should declare a variable's type before putting values into it. If we declare X as an integer, that means we should put only integer numbers into X during its life time in the application.

Examples of declaring and using variables:

```
program FirstVar;

{$mode objfpc}{$H+}

uses
  {$IFDEF UNIX}{$IFDEF UseCThreads}
  cthreads,
  {$ENDIF}{$ENDIF}
  Classes
  { you can add units after this };

var
  x: Integer;
begin
  x:= 5;
  Writeln(x * 2);
  Writeln('Press enter key to close');
  Readln;
end.
```

We will get 10 in the application's output.

Note that we used the reserved word *Var*, which means the following lines will be variable declarations:

```
  x: Integer;
```

This means two things:

1. The variable's name is X; and
2. the type of this variable is Integer, which can hold only integer numbers without a fraction. It could also hold negative values as well as positive values.

And the statement:

```
  x:= 5;
```

means put the value 5 in the variable X.

In the next example we have added the variable Y:

```
var
  x, y: Integer;
begin
  x:= 5;
  y:= 10;
  Writeln(x * y);
  Writeln('Press enter key to close');
  Readln;
end.
```

The output of the previous application is:

```
50
Press enter key to close
```

50 is the result of the formula (x * y).

In the next example we introduce a new data type called ***character***:

```
var
  c: Char;
begin
  c:= 'M';
  Writeln('My first letter is: ', c);
  Writeln('Press enter key to close');
  Readln;
end.
```

This type can hold only one letter, or a number as an alphanumeric character, not as value.

In the next example we introduce the *real* number type, which can have a fractional part:

```
var
  x: Single;
begin
  x:= 1.8;
  Writeln('My Car engine capacity is ', x, ' liters');
  Writeln('Press enter key to close');
```

```
  Readln;
end.
```

To write more interactive and flexible applications, we need to accept input from the user. For example, we could ask the user to enter a number, and then get this number from the user input using the ***Readln*** statement / procedure:

```
var
  x: Integer;
begin
  Write('Please input any number:');
  Readln(x);
  Writeln('You have entered: ', x);
  Writeln('Press enter key to close');
  Readln;
end.
```

In this example, assigning a value to *X* is done through the keyboard instead of assigning it a constant value in the application.

In the example below we show a multiplication table for a number entered by the user:

```
program MultTable;

{$mode objfpc}{$H+}

uses
  {$IFDEF UNIX}{$IFDEF UseCThreads}
  cthreads,
  {$ENDIF}{$ENDIF}
  Classes
  { you can add units after this };

var
  x: Integer;
begin
  Write('Please input any number:');
  Readln(x);
  Writeln(x, ' * 1 = ', x * 1);
  Writeln(x, ' * 2 = ', x * 2);
  Writeln(x, ' * 3 = ', x * 3);
  Writeln(x, ' * 4 = ', x * 4);
  Writeln(x, ' * 5 = ', x * 5);
  Writeln(x, ' * 6 = ', x * 6);
  Writeln(x, ' * 7 = ', x * 7);
  Writeln(x, ' * 8 = ', x * 8);
```

```
  Writeln(x, ' * 9 = ', x * 9);
  Writeln(x, ' * 10 = ', x * 10);
  Writeln(x, ' * 11 = ', x * 11);
  Writeln(x, ' * 12 = ', x * 12);
  Writeln('Press enter key to close');
  Readln;
end.
```

Note that in the previous example all the text between single quotation marks (') is displayed in the console window as is, for example:

```
  ' * 1 = '
```

Variables and expressions that are written without single quotation marks are evaluated and written as values.

See the difference between the two statements below:

```
  Writeln('5 * 3');
  Writeln(5 * 3);
```

The result of first statement is:

```
5 * 3
```

Result of the second statement is evaluated then displayed:

```
15
```

In the next example, we will do mathematical operations on two numbers (x, y), and we will put the result in a third variable (Res):

```
var
  x, y: Integer;
  Res: Single;
begin
  Write('Input a number: ');
  Readln(x);
  Write('Input another number: ');
  Readln(y);
  Res:= x / y;
  Writeln(x, ' / ', y, ' = ', Res);
  Writeln('Press enter key to close');
  Readln;
end.
```

Since the operation is division, it might result in a number with a fraction, so for that reason we have