CIS 343

Final Paper

Priscila Ontiveros

Winter 2022

Introduction

Delphi is a programming language that evolved from Turbo Pascal and Pascal with objects. Delphi has its own sophisticated integrated development environment, and textual user interface toolkit. Delphi was a project originally developed by Borland Software Corporation in the early 1995s. The main authors for this language were Anders Hejlsberg, Larry Tesler and Niklaus Wirth. Hejlsberg was the original author of Turbo Pascal and the chief architect of Delphi. During the development phase, Borland developer Danny Thorpe suggested the name of Delphi as an ode to the Oracle at Delphi. One of the main goals was to provide a straightforward database connectivity to the programmer. Another key feature is that Delphi provides the key tools for rapid application development, making it easy to create projects with prominent visual features. Additionally, Delphi supports native cross-compilation to many platforms such as Windows, Linux, iOS and Android.

Delphi is a high-level programming language, which supports object oriented programming. One of the main attributes of Delphi is to integrate efficient software development and a visual user interface. Delphi is mostly used for the development of Windows applications; however, it also allows the development of mobile applications. A key feature offered by Delphi is its ability to rapidly compile, which enables programmers to write large code-based applications, which can be run on a small resource machine.

Another advantage of developing with Delphi is that it does not require high maintenance. When developers were looking for a high-quality development tool that can be trusted, Delphi was a great option. Having excellent compilation times, fast GUI software development and a minimum need for maintenance, Delphi was a popular choice for developers during the 2000s.

Data abstractions

One of the features Delphi offers is the ability to import library functions and also create your own libraries. Delphi gives the necessary tools to either create Dynamic Linked Libraries or to use them and make function calls when needed. In order to use DLLs, Delphi requires the libraries to be imported. However, Delphi lets the programmer decide either by declaring an external procedure or function, or by making direct calls to Dynamic Linked Libraries using specific application programming interfaces (API). Specifically to programming with Delphi, there are more advantages to using dynamic loading rather than static loading. For example, a program can be executed even when some of the libraries used are not present. Additionally, it allows for modular development and has less memory consumption than static loading.
Delphi is a strongly typed language, which can distinguish different data types, leading to scenarios where the language will not allow to substitute one type for another. Being a strongly typed language has its advantages, as it lets the compiler treat data intelligently and validate code more thoroughly. Allowing the prevention of errors that are hard to diagnose at runtime.
Delphi data types can be broken down into different categories: predefined, fundamental/general, simple/structured, and parameterized/generic. Predefined data types are recognized automatically by the compiler. The majority of the documented types in Delphi are predefined; other types can be created by user declaration or can be created from libraries. Fundamental data types, the range and format of a fundamental data type is platform-specific. Additionally, most predefined data types are fundamental. Types can be parametrized or generic: a type can be essentially generic if they are the base of a structure or procedure that operates with different types. *Figure 1* shows the taxonomy of Delphi data types.

Delphi also has implicit conversions for primitive numeric types; any value can be implicitly converted into a value of a larger type.

```
myByte          : Byte      := 2;      // (8 bits size)
myWord          : Word      := myByte; // 2 (16 bits size)
```

Control abstractions

Like many modern languages, Delphi provides many ways of storing data, for example to define a variable, a section to declare variables needs to be defined

```
var                                    // Starts a block of variables
   myNum : Integer;                   // defines and integer
   myString1,myString2 : String;      // defines two variables of type string
```

In Delphi, a “:” is the connector to define the type of the variable, and a statement of code is ended with the commonly used semicolon (;). To assign a value to a variable another block of code needs to be created with the statement “begin”

```
begin                               // Starts a block of code statements
   myString1    := 'Hello Mr. Woodring!';      // Assign values
   myNum        := 55;
```

The combination of a “:=” enables the assignment of a value to a variable.

Delphi also provides the ability to group together variables and treat the group as a single variable, also known as collections. Creating an array in Delphi, is similar to many object oriented languages. First, the name and type of the array needs to be declared

```
var
   names : array[1..4] of String;     // A list of names
```

Then a segment of code statements is started, where the array is populated and values are assigned to each element.

```
begin
   names[1] := 'Taylor';  // Assigning to index 1
   names[2] := 'Greg';    // Assigning to index 2
   names[3] := 'Kim';     // Assigning to index 3
   names[4] := 'Emma';    // Assigning to index 4
End;
```

Unlike other programming languages, Delphi indexes its elements starting from element 1, not element 0. Therefore, the array declared about has indexes from 1 to 4 (inclusive).

Another way to store collections of data are Records, which are similar to arrays, but they can store a mixture of data types. This characteristic of storing data, distinguishes Delphi from other programming languages. Differently from creating an array, when creating a record, the data needs to be defined in its own block.

```
type
   student Record
       firstName   : string[15];
       lastName    : string[30];
       age         : byte;
End;
```

After defining the data, a variable can be created from the custom data type.

```
var
  myStudent: student;            // The record variable
```

Now, a segment of the code statement can be started to populate the record. Note that in order to access each attribute of the record, the custom variable needs to be called first, then the attribute of the record is linked by concatenating customer.firstName.

```
begin
  student.firstName := 'Taylor';    // Assigning to the record
  student.lastName  := 'Swift';
  student.age     := 31;
End;
```

Support for Object Oriented Programming

One of the key attributes of Delphi was the ability to use object oriented programming. Object oriented programming, introduced more efficient ways of allowing fields to be accessed externally from the class. A class is the building block of object oriented programming, it defines data and procedures and allows for a class to have many fields and methods. Beyond being a place to define data, a class is a data type, and as usual in Delphi, a new data type needs to be created. The process of creating an instance of a class is instantiation, which generates (instantiates) an object. Additionally, each instance of a class is a separate object. Below a new simple class type will be defined.

```
Type                  // defining the class
 TStudnet = class
 private             // private attributes
   name, lastName: string;
   age:integer;
 protected           // protected attributes
   gpa : real;
 public
 // method to print student info
 procedure printStudnet(pName, pLastName:string; pAge:integer);
End
```

It is also important to note that, conventionally, during instantiation, the name of the class or the object starts with a “T”, this allows for better readability when writing software.

Delphi has three control structures for iteration of code: repeat until, while do and the typical for loop. For example to create a for loop, a variable needs to be declared, a code block needs to be defined and inside the code block, the for loop will iterate the variable.

```
var
   i: integer;
begin
   for i := 1 to 5 do
   begin
       ShowMessage('I is: '+IntToStr(i)) ;
   end;
End;
```

To create a function or procedure in Delphi, the same logic of creating a programming block with variables applies. A new procedure needs to be declared in the code.

```
procedure myProcedure(myParameters);
   localDeclarations;
begin
```

```
   statements
end;
```

To return a value a function must be used instead

```
function myFunction(myParameters): returnType;
   localDeclarations;
begin
  statements
end;
```

Conclusion

One of the advantages of using Delphi, is the writability for developing graphical user interfaces. It is an easy language to code, and mostly self intuitive. For readability, Delphi might be a challenging language for beginner programmers. It is recommended that Delphi is a language to be learned after some exposure to object oriented programming with languages such as C++ or Java. Therefore, Delphi is easy to understand for the intermediate programmer as there is a smaller learning curve. Delphi is a resilient language that has definitely shown why it is still used today, it requires low maintenance and provides a good way to effectively develop programs with a strong graphical user interface.

Sources

http://www.delphibasics.co.uk/Article.asp?Name=Abstract

http://www.delphibasics.co.uk/Article.asp?Name=DataTypes

https://www.folio3.ai/blog/delphi-programming-language/#:~:text=The%20Delphi%20language%20and%20software,iOS%2C%20and%20Android%20operating%20systems.

https://www.thoughtco.com/static-vs-dynamic-1058452

https://rosettacode.org/wiki/Implicit_type_conversion#Delphi

https://docwiki.embarcadero.com/RADStudio/Sydney/en/About_Data_Types_(Delphi)#:~:text=The%20Delphi%20language%20is%20a,substitute%20one%20type%20for%20another.