DELPHI DATA TYPES

Desika Mawar Safira

165100023

Fakultas Komputer, 448757256

desikamawarsafira.student@umitra.ac.id

Abstract

Every time we make a program it will always be faced with the data type. This data type plays an important role when we are going to process the data. Because we cannot process the data if we do not know the data type of the value or variable that will be processed. Through this paper we will introduce some commonly used data types and examples of their use.

As we know in Delphi the use of strict data types, for example a variable of type string

Kata Kunci : Data type, string.

A. INTRODUCTION

Materi Ke 1 membahas mengenai i cloud system ...dst
tambahkan gambar untuk memperkuat penjelasan (Gunakan Minimal 600 kata)

JAWABAN :

DELPHI DATA TYPES

Storing data in computer programs :

For those new to computer programming, data and code go hand in hand. You cannot write a program of any real value without lines of code, or without data. A Word Processor program has logic that takes what the user types and stores it in data. It also uses data to control how it stores and formats what the user types and clicks.

Data is stored in the memory of the computer when the program runs (it can also be stored in a file, but that is another matter beyond the scope of this tutorial). Each memory 'slot' is identified by a name that the programmer chooses. For example LineTotal might be used to name a memory slot that holds the total number of lines in a Word Processor document.

The program can freely read from and write to this memory slot. This kind of data is called a Variable. It can contain data such as a number or text. Sometimes, we may have data that we do not want to change. For example, the maximum number of lines that the Word Processor can handle. When we give a name to such data, we also give it its permanent value. These are called constants.

Simple Delphi data types

Like many modern languages, Delphi provides a rich variety of ways of storing data. We'll cover the basic, simple types here. Before we do, we'll show how to define a variable to Delphi:

```
var                                //  
This starts a section of  
variables  
    LineTotal : Integer; //  
This defines an Integer  
variable called LineTotal  
    First,Second : String; //  
This defines two variables to  
hold strings of text
```

We'll show later exactly where this var section fits into your program. Notice that the variable definitions are indented - this makes the code easier to read - indicating that they are part of the var block.

Each variable starts with the name you choose, followed by a : and then the variable type. As with all Delphi statements, a ; terminates the line. As you can see, you can define multiple variables in one line if they are of the same type.

It is very important that the name you choose for each variable is unique, otherwise Delphi will not know how to identify which you are referring to. It must also be different from the Delphi language keywords. Don't worry, you don't need to come up with an unusually creative name (such as Me and Ro Jewelry and others in the fashion industry) for each of your variables. Unique in this case simply means the only occurrence. You'll know when you

have got it right when Delphi compiles your code OK (by hitting Ctrl-F9 to compile).

Delphi is not sensitive about the case (lower or upper) of your names. It treats theCAT name the same as TheCat.

Number types

Delphi provides many different data types for storing numbers. Your choice depends on the data you want to handle. Our Word Processor line count is an unsigned Integer, so we might choose Word which can hold values up to 65,535. Financial or mathematical calculations may require numbers with decimal places - floating point numbers.

```
var
  //      Integer      data      types      :
  Int1 : Byte;      //      0 to 255
  Int2 : ShortInt; //      -127 to 127
  Int3 : Word;      //      0 to
65,535
  Int4 : SmallInt; //      -32,768 to
32,767
  Int5 : LongWord; //      0 to
4,294,967,295
  Int6 : Cardinal; //      0 to
4,294,967,295
  Int7 : LongInt; //      -2,147,483,648 to
2,147,483,647
  Int8 : Integer; //      -2,147,483,648 to
2,147,483,647
  Int9 : Int64; //      -9,223,372,036,854,775,808 to
9,223,372,036,854,775,807

  //      Decimal      data      types      :
  Dec1 : Single; // 7 significant digits,
exponent -38 to +38
  Dec2 : Currency; // 50+ significant digits, fixed 4
decimal places
  Dec3 : Double; // 15 significant digits,
exponent -308 to +308
  Dec4 : Extended; // 19 significant digits, exponent
-4932 to +4932
```

Some simple numerical variable usage examples are given below - fuller details on numbers is given in the Numbers tutorial.

Text types

Like many other languages, Delphi allows you to store letters, words, and sentences in single variables. These can be used to display, to hold user details and so on. A letter is stored in a single character variable type, such as Char, and words and sentences stored in string types, such as String.

```
var
  Str1 : Char; //
Holds a single character,
small alphabet
  Str2 : WideChar; //
Holds a single character,
International alphabet
  Str3 : AnsiChar; //
Holds a single character,
small alphabet
  Str4 : ShortString; //
Holds a string of up to 255
Char's
  Str5 : String; //
Holds strings of Char's of any
size desired
  Str6 : AnsiString; //
Holds strings of AnsiChar's
any size desired
  Str7 : WideString; //
Holds strings of WideChar's of
any size desired
```

Some simple text variable usage examples are given below - fuller details on strings and characters is given in the Text tutorial.

Logical data types

These are used in conjunction with programming logic. They are very simple:

```
var
  Log1 : Boolean; // Can
be 'True' or 'False'
```

Boolean variables are a form of enumerated type. This means that they can hold one of a fixed number of values, designated by name. Here, the values can be True or False. See the tutorials on Logic and Looping for further details.

Sets, enumerations and subtypes

Delphi excels in this area. Using sets and enumerations makes your code both easier to use and more reliable. They are used when categories of data are used. For example, you may have an enumeration of playing card suits. You literally enumerate the suit names. Before we can have an enumerated variable, we must define the enumeration values. This is done in a type section.

```
type
    TSuit = (Hearts, Diamonds,
Clubs, Spades); // Defines
the enumeration
var
    suit :
TSuit;
// An enumeration
variable
```

Sets are often confused with enumerations. The difference is tricky to understand. An enumeration variable can have only one of the enumerated values. A set can have none, 1, some, or all of the set values. Here, the set values are not named - they are simply indexed slots in a numeric range. Confused? Well, here is an example to try to help you out. It will introduce a bit of code a bit early, but it is important to understand.

```
type
    TWeek = Set of
1..7; // Set
comprising the days of the week, by
number
var
    week : TWeek;
begin
    week := [1,2,3,4,5]; //
Switch on the first 5 days of the
week
end;
```

See the Set reference, and the Sets and enumerations tutorial for further details. That tutorial introduces a further data type - a subrange type.

Using these simple data types

Variables can be read from and written to. This is called assignment. They can also be used in expressions and programming logic. See the Text tutorial and Programming logic tutorial for more about these topics.

Assigning to and from variables

Variables can be assigned from constant values, such as 23 and 'My Name', and also from other variables. The code below illustrates this assignment, and also introduces a further section of a Delphi program : the const (constants) section. This allows the programmer to give names to constant values. This is useful where the same constant is used throughout a program - a change where the constant is defined can have a global effect on the program.

Note that we use upper case letters to identify constants. This is just a convention, since Delphi is not case

sensitive with names (it is with strings). Note also that we use = to define a constant value.

```
types
  TWeek      =
1..7;        // Set
comprising  the days of the
week,       by number
  TSuit = (Hearts, Diamonds,
Clubs, Spades); // Defines
an          enumeration

const
  FRED      =
'Fred';     // String
constant
  YOUNG_AGE =
23;         // Integer
constant
  TALL      : Single =
196.9;      // Decimal
constant
  NO        =
False;      // Boolean
constant

var
  FirstName, SecondName :
String;     // String variables
  Age          :
Byte;       // Integer variable
  Height       :
Single;     // Decimal variable
  IsTall       :
Boolean;    // Boolean variable
  OtherName    :
String;     // String variable
  Week         :
TWeek;      // A set variable
  Suit         :
TSuit;      // An enumeration
variable

begin // Begin starts a
block of code statements
  FirstName :=
FRED;       // Assign from
predefined  constant
  SecondName :=
'Bloggs';   // Assign from a
literal     constant
```

```
Age      :=
YOUNG_AGE; // Assign from
predefined constant
Age      :=
55;       // Assign from
constant - overrides YOUNG_AGE
Height   := TALL -
5.5;     // Assign from a mix of
constants
IsTall    :=
NO;       // Assign from
predefined constant
OtherName :=
FirstName; // Assign from
another    variable
Week      :=
[1,2,3,4,5]; // Switch on the
first 5 days of the week
Suit      :=
Diamonds;  // Assign to an
enumerated variable
end; // End finishes a
block of code statements
```

```
FirstName is now set
to 'Fred'
SecondName is now set
to 'Bloggs'
Age is now set to 55
Height is now set to 191.4
IsTall is now set to False
OtherName is now set
to 'Fred'
Week is now set
to 1,2,3,4,5
Suit is now set
to Diamonds (Notice no
quotes)
```

Note that the third constant, TALL, is defined as a Single type. This is called a typed constant. It allows you to force Delphi to use a type for the constant that suits your need. Otherwise, it will make the decision itself.

Compound data types

The simple data types are like single elements. Delphi provides compound

data types, comprising collections of simple data types.

These allow programmers to group together variables, and treat this group as a single variable. When we discuss programming logic, you will see how useful this can be.

Arrays

Array collections are accessed by index. An array holds data in indexed 'slots'. Each slot holds one variable of data. You can visualise them as lists. For example:

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

begin
    Suits[1] := 'Hearts';    //
Assigning to array index 1
    Suits[2] := 'Diamonds'; //
Assigning to array index 2
    Suits[3] := 'Clubs';    //
Assigning to array index 3
    Suits[4] := 'Spades';   //
Assigning to array index 4
end;
```

The array defined above has indexes 1 to 4 (1..4). The two dots indicate a range. We have told Delphi that the array elements will be string variables. We could equally have defined integers or decimals.

For more on arrays, see the Arrays tutorial.

Records

Records are like arrays in that they hold collections of data. However, records

can hold a mixture of data types. There are a very powerful and useful feature of Delphi, and one that distinguishes Delphi from many other languages.

Normally, you will define your own record structure. This definition is not itself a variable. It is called a data type (see Types for further on this). It is defined in a type data section. By convention, the record type starts with a T to indicate that it is a type not real data (types are like templates). Let us define a customer record:

```
type
    TCustomer      Record
        firstName  : string[20];
        lastName   : string[20];
        age        :   byte;
end;
```

Note that the strings are suffixed with [20]. This tells Delphi to make a fixed space for them. Since strings can be a variable length, we must tell Delphi so that it can make a record of known size. Records of one type always take up the same memory space.

Let us create a record variable from this record type and assign to it:

```
var
    customer      :
TCustomer;       // Our
customer         variable
begin
    customer.firstName :=
'Fred';          // Assigning to the
customer         record
    customer.lastName  :=
'Bloggs';
    customer.age       := 55;
```

```
customer.firstName is now set  
to 'Fred'  
customer.lastName  is now set  
to 'Bloggs'  
customer.age       is now set  
to 55
```

Notice how we do not use an index to refer to the record elements. Records are very friendly - we use the record element by its name, separated from the record name by a qualifying dot. See the Records tutorial for further on records.

Objects

Objects are collections of both data and logic. They are like programs, but also like data structures. They are the key part of the Object oriented nature of Delphi. See the Object orientation tutorial for more on this advanced topic.

Other data types

The remaining main object types in Delphi are a mixed bunch:

Files

File variables represent computer disk files. You can read from and write to these files using file access routines. This is a complex topic covered in Files.

Pointers

Pointers are also the subject of an advanced topic - see Pointer reference. They allow variables to be indirectly referenced.

Variants

Variants are also an advanced topic - see Variant. They allow the normal Delphi rigid type handling to be avoided. Use with care!

Type definitions

When we discussed Records above, we introduced the concept of types. Delphi has many predefined data types - both simple, such as string, and compound, such as TPoint (which holds X and Y coordinates of a point). See Type for further de

B. CONCLUSION

Kesimpulan dari materi ini adalah(minimal 150 kata)

Conclusion :

Data is a record of facts or facts in the form of numbers, letters / characters, symbols, images, sounds, etc. Data is information needed in various aspects. Often before doing a process data is needed as a reference.

A variable is data whose value can change at any time depending on the situation. Variables are not defined by certain values but only by name.

Various types of data at Delphi :

1. Integer type

Is a data type that states numbers / numbers. Types of integer data types:

Tipe Data	Jangkauan Nilai
Shortint	-128 s/d 127
Smallint	-32768 s/d 32767

Longint	-2147483648 s/d 2147483647	Boolean data	32bit, signed
Int64	-2^{63} s/d $2^{63} - 1$	Tipe Data	Nilai
Byte	0 s/d 255	Boolean	True or False
Word	0 s/d 65535	ByteBool	True or False
LongWord	0 s/d 4294967295	WordBool	True or False
		LongBool	True or False

2. Real type

Is a data type that states the set of numbers / numbers. Real type is also called float type. Types of real data types:

Tipe Data	Jangkauan Nilai	Ukuran
Real48	2.9×10^{-39} s/d 1.7×10^{38}	1 Byte
Single	1.5×10^{-45} s/d 3.4×10^{38}	4 Byte
Double	5.0×10^{-324} s/d 1.7×10^{308}	8 Byte
Extended	3.6×10^{-4951} s/d 1.1×10^{4932}	10 Byte
Comp	$-2^{63} + 1$ s/d $2^{63} - 1$	8 Byte
Currency	-922337203685477.5808 s/d 922337203685477.5807	8 Byte

3. Boolean type

Is a data type that states true or false values (True or False). Various types of

4. Character Type

Is a data type to declare a letter / character. The fundamental character type is AnsiChar and WideChar. AnsiChar is one byte in size and WideChar is more than one byte in size. The generic character type is Char, which is equivalent to AnsiChar. Characters can also be shown in string constants of length 1, for example 'A'. The Chr function produces character values for each AnsiChar integer range. An example of the Chr (65) function produces the letter A.

5. String Type

Is a data type that states a series of several characters. Types of string types:

6. Pointer type

Is a variable that shows a memory address. Pointers are used to indicate the type of data stored at the address they contain. The pointer occupies 4 bytes of memory.

7. Variant Type

A data type that states a value that can change at runtime program. Variants can contain all data types. Variant offers high compatibility but requires large memory.

C. ACKNOWLEDGEMENT

University Of Indonesia
University Of Mitra Indonesia
Telkom University
University Of Melbourne
Saitama University

D. REFERENCE (Based ISO 690)

- [1] A. S. Putra And O. M. Febriani, "Knowledge Management Online Application In Pdam Lampung Province," In *Prosiding International Conference On Information Technology And Business (Icibt)*, 2018, Pp. 181–187.
- [2] A. S. Putra, O. M. Febriani, And B. Bachry, "Implementasi Genetic Fuzzy System Untuk Mengidentifikasi Hasil Curian Kendaraan Bermotor Di Polda Lampung," *J. Sist. Inf. Dan Manaj. Basis Data*, Vol. 1, No. 1, Pp. 21–30, 2018.
- [3] O. M. Febriani And A. S. Putra, "Sistem Informasi Monitoring Inventori Barang Pada Balai Riset Standardisasi Industri Bandar Lampung," *J. Inform.*, Vol. 13, No. 1, Pp. 90–98, 2014.
- [4] Putra, Arie Setya. "2018 Artikel Struktur Data, Audit Dan Jaringan Komputer." (2018).

- [5] Putra, A. S. (2018, July 17). Paperplain Fundamental Create Application With Borland Delphi 7.0 University Of Mitra Indonesia. Retrieved From [Osf.io/Pbrn9](#).

E. REFERENCE (Based APA)

Putra, A. S., Aryanti, D. R., & Hartati, I. (2018, November). Metode SAW (Simple Additive Weighting) sebagai Sistem Pendukung Keputusan Guru Berprestasi (Studi Kasus: SMK Global Surya). In *Prosiding Seminar Nasional Darmajaya* (Vol. 1, No. 1, pp. 85-97).

Sari, D. P., Febriani, O. M., & Putra, A. S. (2018, November). Perancangan Sistem Informasi SDM Berprestasi pada SD Global Surya. In *Prosiding Seminar Nasional Darmajaya* (Vol. 1, No. 1, pp. 289-294).

Putra, A. S. (2018). Paperplain: Execution Fundamental Create Application With Borland Delphi 7.0 University Of Mitra Indonesia.

Putra, A. S., Sukri, H., & Zuhri, K. Sistem Monitoring Realtime Jaringan Irigasi Desa (JIDES) Dengan Konsep Jaringan Sensor Nirkabel. *IJEIS (Indonesian Journal of Electronics and Instrumentation Systems)*, 8(2), 221-232.

Darmawan, A., Yuliawati, D., Marcella, O., & Firmandala, R. (2016). Sistem Absensi dan Pelaporan Berbasis Fingerprint dan SMS Gateway. *EXPLORE*, 7(1).

Febriani, O. M., Wahyuni, T., & Yusuf, S. (2017). DESIGN OF WEBSITE-

BASED INFORMATION SYSTEM
FOR EDOCUMENT
ADMINISTRASI IN THE
COMMUNITY SERVICE UNIT (A
Case Study at Rajabasa District).
*INTERNATIONAL JOURNAL OF
COMPUTERS & TECHNOLOGY*,
16(7), 7010-7020.

Febriani, O. M., & Wahyuni, T. (2017, October). PERANCANGAN SISTEM E-DOCUMENT ADMINISTRASI LOGBOOK PENELITIAN PADA UNIT LAYANAN DI BANDAR LAMPUNG. In *Prosiding Seminar Nasional Darmajaya* (Vol. 1, No. 1, pp. 187-194).

Febriani, O. M., & Permadi, A. B. (2017). Implementasi Sistem Aplikasi Data Bimbingan dan Pelanggaran Siswa pada Sekolah Menengah Atas di Lampung Tengah dengan Metode Analisis dan Desain Sistem Terdistribusi (SSAD). *EXPERT*, 7(1).

Febriani, O. M., & Ambarwati, L. (2015). PERANCANGAN APLIKASI PENGOLAHAN DATA PENJUALAN UKM KELANTING KHAS TELO DESA SIDOHARJO KECAMATAN JATI AGUNG KABUPATEN LAMPUNG SELATAN. *Jurnal Teknologi Informasi dan Bisnis Pengabdian Masyarakat Darmajaya*, 1(1), 77-95.

Febriani, O. M. (2015). Rancang Bangun Aplikasi E-commerce menggunakan Freewebstore pada UKM Kelanting di Desa Sidoharjo Lampung Selatan. *Prosiding Sembistek 2014*, 1(02), 446-458.