MODULE 4: INTRODUCTION TO C/AL PROGRAMMING

Module Overview

Computer programming languages have many purposes. Many of these purposes are administered by the standard Microsoft Dynamics NAV 2013 objects. Following are some examples:

- Data presentation is handled through the page and report objects.
- Data acquisition is mainly handled through page and XMLport objects.
- Data storage and organization is handled by the table objects together with the built-in Database Management System (DBMS).

To create a coherent application in Microsoft Dynamics NAV, database objects must work together. Client Application Language (C/AL) is the programming language that is used in the Microsoft Dynamics NAV 2013 Development Environment. C/AL code is used to bind all the database objects together to form a unified whole.

Objectives

- Describe the concepts and basic use of C/AL code elements.
- Describe the concepts of data types, simple data types and complex data types.
- Explain the concepts of identifiers, variables, and syntax.
- Explain the syntax of identifiers.
- Explain the scope of variables.
- Explain the initialization of variables.
- Create a simple codeunit to demonstrate how to define variables, assign data types, and investigate several default values that are initialized for several data types.

C/AL Programming

C/AL enables developers to create functions that extend the functionality of Microsoft Dynamics NAV, such as special functions that can you can use anywhere in the database.

In addition, C/AL also enables developers to do the following:

- Design custom functions.
- Connect database objects.
- Read, write, and change data.

The main purpose of the programming language in Microsoft Dynamics NAV 2013 is data manipulation. Through C/AL, developers can create business rules to make sure that the data which is stored in the tables is meaningful and consistent with the way customers do business. Through programming, developers can do the following:

- Add new data or transfer data from one table to another, such as transferring from a journal table to a ledger table.
- Combine data from multiple tables into one report or display it on one page.

Another purpose of C/AL is to control the execution of the application objects. With C/AL, developers can coordinate objects to meet the business needs of their customers.

Code Statements and Triggers

You can find C/AL code in most Microsoft Dynamics NAV objects. The codeunit object is used only for programming. Codeunits consist of programming language statements, also known as C/AL statements or code. The following steps show how to open the C/AL Editor to view triggers in the **DateFilter-Calc** codeunit.

Perform the following steps in Microsoft Dynamics NAV Development Environment:

1. Click **Tools** > **Object Designer**. The **Object Designer** opens.
2. Click **Codeunit** to open the codeunit list.
3. Select codeunit **358**, **DateFilter-Calc**, and then click **Design** to design the **DateFilter-Calc** codeunit.
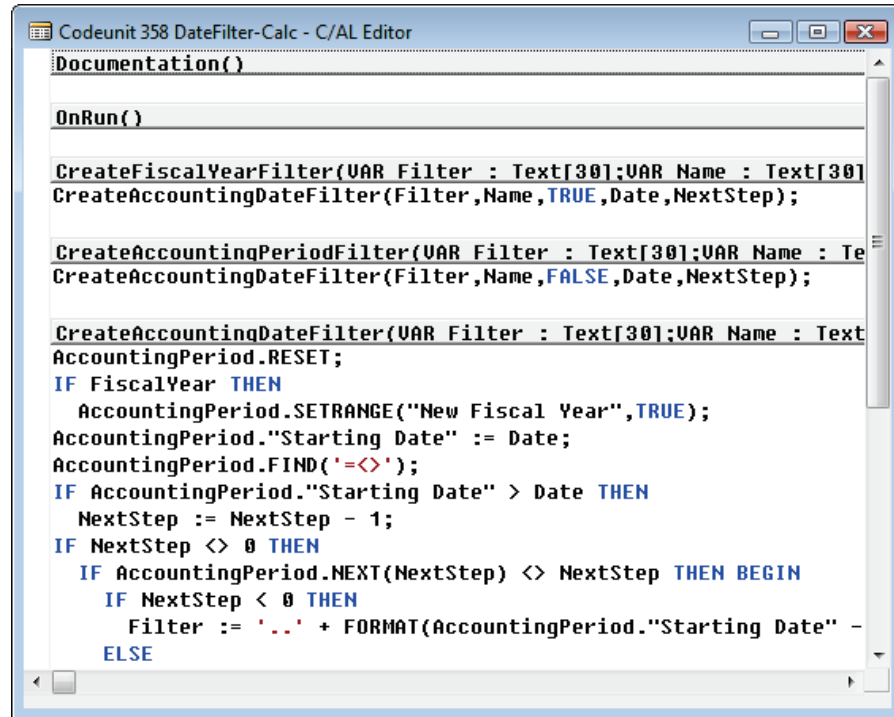
FIGURE 4.1: THE DATEFILTER-CALC CODEUNIT IN C/AL EDITOR

📋 **Note:** *The codeunit designer is the **C/AL Editor**. In any other designer, such as the **Page Designer**, use the following procedure to access the **C/AL Editor**:*

1. *Click **View** > **C/AL Code**.*
2. *Do one of the following:*
   - *Click the **C/AL Code** button on the Toolbar.*
   - *Press **F9**.*

Each shaded bar in the codeunit is known as a *trigger*. The C/AL code under the shaded bar is the trigger code for that trigger. If there is no C/AL code between one trigger and the next trigger, then that trigger is empty. In the **DateFilter-Calc** codeunit, there is no trigger code in the **OnRun** trigger section. Therefore, the **OnRun** trigger is empty.

Following are the three types of triggers:

- **Documentation triggers**. Use documentation triggers to write documentation for a particular object. This is not really a trigger and no code in this trigger runs. Many developers use this space to document their modifications to standard objects. Every object has a documentation trigger.

- **Event triggers**. Event trigger names always starts with the word **On**. The C/AL code in an event trigger runs when the named event occurs. For example, the code in the **OnRun** event trigger executes when a codeunit that contains the trigger runs. In the **DateFilter-Calc** codeunit, because there is no trigger code in the **OnRun** trigger, nothing happens when the codeunit runs. Each object has its own set of predefined event triggers.

- **Function Triggers**. Developers create these triggers when they create a function in an object. The C/AL code in the function trigger executes when the function is called. For example, the **DateFilter-Calc** codeunit has three function triggers: **CreateFiscalYearFilter**, **CreateAccountingPeriodFilter**, and **CreateAccountingDateFilter**.

Accessing C/AL

Reviewing C/AL code in a standard object can help developers become familiar with the code elements in that object. The following steps show how to access C/AL code in the **Customer** table.

1. Design table **18**, **Customer**, from the Object Designer. The **Customer** table opens in the Table Designer.
2. Do any of the following to open the **C/AL Editor**:
   - Click **View** > **C/AL Code**.
   - Click the **C/AL Code** button on the Toolbar
   - Press **F9**.

The **C/AL Editor** opens, and shows the triggers for the **Customer** table and its fields.
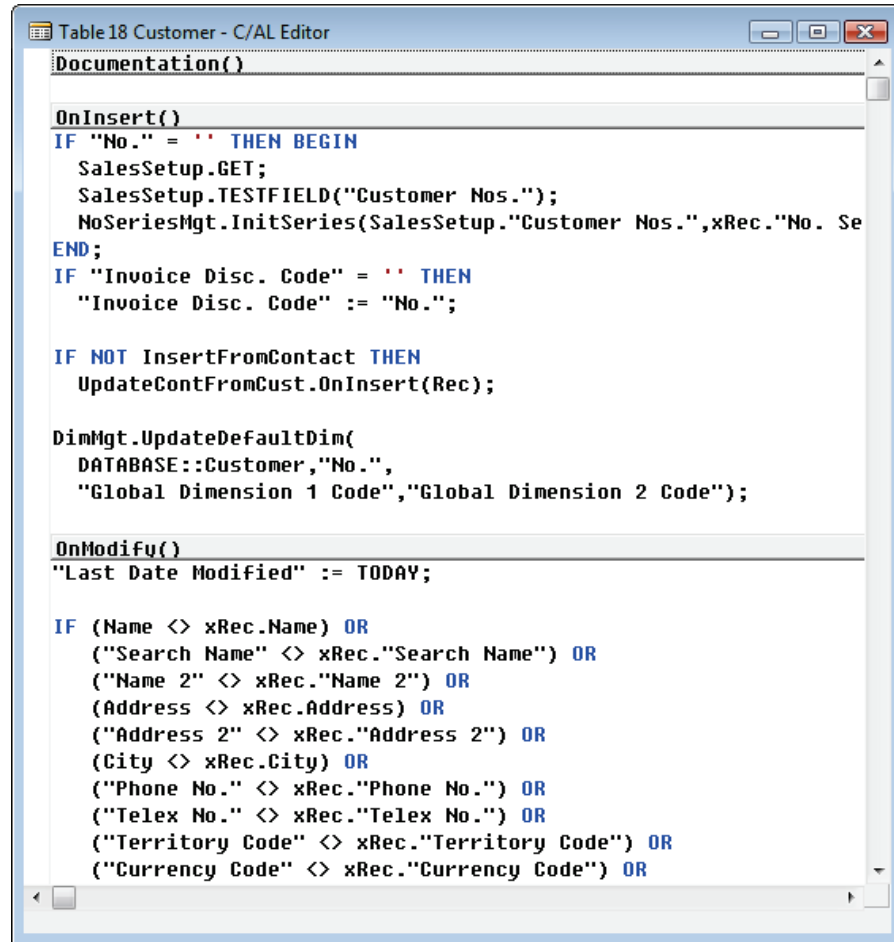
```
Table 18 Customer - C/AL Editor
Documentation()

OnInsert()
IF "No." = '' THEN BEGIN
  SalesSetup.GET;
  SalesSetup.TESTFIELD("Customer Nos.");
  NoSeriesMgt.InitSeries(SalesSetup."Customer Nos.",xRec."No. Se
END;
IF "Invoice Disc. Code" = '' THEN
  "Invoice Disc. Code" := "No.";

IF NOT InsertFromContact THEN
  UpdateContFromCust.OnInsert(Rec);

DimMgt.UpdateDefaultDim(
  DATABASE::Customer,"No.",
  "Global Dimension 1 Code","Global Dimension 2 Code");

OnModify()
"Last Date Modified" := TODAY;

IF (Name <> xRec.Name) OR
   ("Search Name" <> xRec."Search Name") OR
   ("Name 2" <> xRec."Name 2") OR
   (Address <> xRec.Address) OR
   ("Address 2" <> xRec."Address 2") OR
   (City <> xRec.City) OR
   ("Phone No." <> xRec."Phone No.") OR
   ("Telex No." <> xRec."Telex No.") OR
   ("Territory Code" <> xRec."Territory Code") OR
   ("Currency Code" <> xRec."Currency Code") OR
```

FIGURE 4.2: THE CUSTOMER TABLE IN THE C/AL EDITOR

Many triggers in the **Customer** table are empty. Each field has an **OnValidate** and an **OnLookup** trigger. These are event triggers. The code in these event triggers runs when the respective event occurs, such as when the user triggers the event.

Intrinsic Data Types

Data are pieces of information. Data refers to information that is available in Microsoft Dynamics NAV. Data types are the classifications of this information. Classifying data is important because it indicates how the application must handle data when you run code.

Different data types have different values and different meanings for those values. Data types are manipulated differently. Data types can be numeric or text. For example, if developers have two data values, 25 and 37, and add them, they then obtain different results, depending on the values' data type. If the values are numeric, the result is 62. However, if they are text, the result may be 2537.

Constants

Constants are data values that are written directly into programming statements. They are called constants because their values never change while the application is running. You can change constants only by changing the C/AL code.

Simple Data Types

Simple data types are data types that have only one value. They cannot be broken up into other values of different types.

Byte

A byte is a unit of computer data storage space. One character stored in the computer uses one byte of storage. The following are several related terms:

Byte Type	Number of Bytes	Other Value
Kilobyte (KB)	1,024	
Megabyte (MB)	1,048,576	1024 KB
Gigabyte (GB)	1,073,741,824 bytes	1024 MB

Numeric Data Types

Numeric data types are all forms of numbers or amounts. Microsoft Dynamics NAV 2013 uses many automatic methods to convert one type of number to another. This conversion is transparent to the user. You can use numeric data types interchangeably. However, numeric data types have important differences that sometimes cause errors and more subtle problems.

Integer

An integer is a whole number that can range in value from -2,147,483,647 to +2,147,483,647. The default value of an integer is zero. Typical constants of type Integer in C/AL are as follows:

- 12
- 1000 (There are no commas as they are invalid in numeric constants.)
- -100
- 0

BigInteger

BigInteger is used to store large whole numbers. It is a 64-bit integer. Add an "L" to the constant definition to inform C/AL that an integer must be interpreted and treated as a BigInteger. Typical constants of type BigInteger in C/AL are as follows:

- 1L
- 455500000000L

Decimal

A decimal is a whole or fractional number that can range in value from -999,999,999,999,999.99 to +999,999,999,999,999.99. The default value of a decimal is zero. In Microsoft Dynamics NAV 2013, the Decimal data type is mapped to the Microsoft .NET Common Language Runtime (CLR) Decimal data type. Decimal data type precision and limits behave slightly differently than the Binary Coded Decimal (BCD) data type earlier versions of C/AL. Typical constants of type Decimal in C/AL are as follows:

- 12.50
- 52000000000 (there are no commas)
- -2.0
- 0.008
- -127.9533

Option

An option is a special kind of integer that enables developers to define words for each value. For example, if developers create variables named Spectrum, they can set it to an Option data type with the following OptionString: Red,Orange,Yellow,Green,Blue,Indigo,Violet.

The default value of an option is zero, because it is an integer. This represents the first element. In the Spectrum example, this is Red. Therefore, Green is represented by the integer 3. There are no spaces between the elements in the OptionString, as a space becomes part of the element's name.

Char

A char is a single character. For syntax purposes, it is considered as a numeric data type and can have integer values from zero to 255. Use chars together with other numeric data types in expressions. Typical constants of type Char in C/AL are as follows:

- 'b' (with single quotation marks surrounding the character)

- 'C'
- '3'
- '?'

🌟 String Data Types

String data is data that is made up of strings of characters. The data in word processors is string data. In spreadsheets, where most of the data is considered numeric, string data is entered by using a special prefix to distinguish it. In C/AL, the symbol that is used to distinguish string data is the single quotation mark, also known as an apostrophe ('). All string constants are enclosed in single quotation marks.

Text

A text is a string of either specified or unlimited length. If you specify the length, then a text can contain up to 1024 characters.

> 📄 **Note:** *If you do not specify the length of a text variable, then it is of unlimited length. A text variable of unlimited length can contain up to 2GB (gigabytes) of data.*

The actual length of a text is the number of characters it contains. Typical constants of type Text in C/AL are as follows:

- 'Hello'
- '127.50' (This resembles a number but because it is enclosed in quotation marks, it is a text.)
- '' (This is an empty, 0 length text.)
- ' spaces before ... and after '
- 'Here''s how to use an apostrophe' (To insert a single apostrophe in a text constant, insert two apostrophes.)

> 📄 **Note:** *All string data types in Microsoft Dynamics NAV 2013 are Unicode. This means that you can include characters from any character set that is supported under Windows.*

Code

A code is a special kind of text. Code cannot be of unlimited length, and you must specify the length between 1 and 1024 characters. All letters in a code convert automatically to uppercase, and all leading and trailing spaces are removed. In addition, when code is displayed to the user, it is automatically right-justified if all the characters it contains are numbers. Now the text constants in the previous

example look like the following when converted to code:

- 'HELLO'
- '127.50'
- ''
- 'SPACES BEFORE ... AND AFTER'
- 'HERE''S HOW TO USE AN APOSTROPHE'

📓 **Note:** *Similar to text, code also supports Unicode.*

Boolean Data Types

Boolean data, also known as logical data, is the simplest type of data. The constants of type Boolean in C/AL are as follows:

- TRUE
- FALSE

If these values are compared, the FALSE value is less than the TRUE value because it is stored as a zero (0). TRUE is stored as one. However, the integer value is not interchangeable with the constant of TRUE or FALSE. In code, the Boolean variable must be set to TRUE or FALSE, not zero (0) or one.

Date, Time, and DateTime Data Types

Date

Date is a calendar date that can range in value from 1/3/0001 to 12/31/9999. In addition, the value of a date can be either a Normal Date or a Closing Date. The Closing Date represents the last second of the last minute of the last hour of the day. Therefore, it is greater than the Normal Date with the same calendar value. Typical constants of type Date in C/AL are as follows:

- 123197D (December 31, 1997)
- 030595D
- 08171953D (August 17, 1953)
- 0D (The undefined date, less than all other dates)
- 063012D (June 30, 2012)

All of these Date constants are Normal Dates. There are no Closing Date constants in C/AL.

The general syntax is mmddyyD or mmddyyyyD. Two digits may be used for the year. In Microsoft Dynamics NAV 2013, if the year is from 30 to 99, it is considered in the 1900s. If it is from 00 to 29, it is considered in the 2000s.

The Date data type is defined by using a D at the end. If there is no D, then C/AL assumes that it is an integer. An error then occurs because an integer cannot be assigned to a date. In code, do not use slashes to separate the month, day, and year. Slashes imply division and an error occurs because an integer or a decimal cannot be assigned to a date variable.

Time

A Time data type represents the time of day, and not a time interval. It ranges in value from 00:00:00 to 23:59:59.999. Constants of type Time in C/AL are as follows:

- 103000T (10:30am)
- 154530T (3:45:30pm)
- 0T (The undefined time, less than all other times)
- 030005.100T (3:00:05.1am)
- 225930.135T (10:59:30.135pm)

The general syntax is hhmmss[.xxx]T, where the fractions of seconds (.xxx) are optional. Similar to the Date data type, the Time data type must have a T at the end to distinguish it from the numeric data type.

DateTime

A DateTime data type indicates a date and a time of day. The DateTime is stored in the database as Coordinated Universal Time (UTC). UTC is the international time standard (formerly Greenwich Mean Time, or GMT). Zero hours UTC is midnight at zero degrees longitude. The DateTime is always displayed as local time in Microsoft Dynamics NAV. Local time is determined by the time zone regional settings of the computer.

DateTime data must be entered as local time. It then is converted to UTC by using the current settings for the time zone and daylight savings time.

There is only one constant available for this data type is the undefined DateTime.

- 0DT (The undefined DateTime)

Complex Data Types

C/AL also contains several complex data types. Developers use complex data types when they work with records in tables, pictures (bitmaps), or disk files. Because C/AL is object-based, each complex data type can include both member variables and member functions.

Record

The Record data type is a complex data type that consists of several simpler elements called fields. It corresponds to a row in a table. Each field in the record stores values of a certain data type. Access the fields by using the variable name of the record (frequently the same as the name of the corresponding table), a dot (a period), and the field name. A record holds information about a fixed number of properties.

Page

The Page data type stores pages. This is a complex data type and can contain several simpler elements called controls. Controls display information to the user or receive user input.

Codeunit

The Codeunit data type stores codeunits. This is a complex data type that can contain several user-defined functions.

File

The File data type gives developers access to operating system files.

Dialog

The Dialog data type stores dialog windows. Several functions are available for manipulating dialog boxes.

Report

The Report data type stores reports. This is a complex data type that can contain several simpler elements called controls. Controls display information to the user.

DateFormula

The DateFormula data type provides multilanguage capabilities to the CALCDATE function. Variables of this data type contain a date formula that has the same capabilities as the ordinary input string for the CALCDATE function.

GUID

The Globally Unique Identifier (GUID) data type gives a unique identifying number to any database object. Use GUIDs for the global identification of objects, programs, records, and so on.

Each value in a GUID is globally unique. An algorithm, developed by Microsoft, generates this value. This guarantees the GUID's uniqueness.

The GUID is a 16-byte binary data type. It can be logically grouped into the following subgroups: 4byte-2byte-2byte-2byte-6byte. The standard textual representation is {12345678-1234-1234-1234-1234567890AB}.

TableFilter

The TableFilter data type applies a filter to another table. Use this data type only when you set security filters from the Permission table.

RecordRef

The RecordRef data type is a complex data type that identifies a row in a table. Each record consists of fields that form the columns of the table. A record holds information about a fixed number of properties.

A RecordRef object can refer to any table in the database. Use the RecordRef.OPEN function to select the table to be accessed. When you use the RecordRef.OPEN function, you create a new RecordRef object. This object contains references to the open table, filters, the record itself, and all of the fields that it contains.

If one RecordRef variable is assigned to another RecordRef variable, they both refer to the same table instance.

RecordID

The RecordID data type is a complex data type that contains the table number and the primary key of a table. You can store a RecordID in the database, but you cannot set filters on a RecordID.

FieldRef

The FieldRef data type is a complex data type that identifies a field in a table and gives developers access to this field. The FieldRef object can refer to any field in any table in the database.

KeyRef

The KeyRef data type is a complex data type that identifies a key in a table and the fields in this key. This gives developers access to the key and the fields that it contains. The KeyRef object can refer to any key in any table in the database.

InStream and OutStream

The InStream and OutStream data types let developers read from or write to files and BLOBs. In addition, you can use InStream and OutStream to read from and to write to Automation objects and DotNet data types.

Variant

The variant data type can contain the following C/AL data types:

- Record
- File
- Action
- Codeunit
- Automation
- Boolean
- Option
- Integer
- Decimal
- Char
- Text
- Code
- Date
- Time
- Binary
- DateFormula
- TransactionType
- InStream
- OutStream

BigText

The BigText data type is a complex data type that contains large text documents. Data of the BigText data type cannot be displayed in the debugger or in a message window. Use BigText functions to extract part of a BigText and insert a text string that can be displayed.

The maximum length of a BigText variable is 2,147,483,647 characters. This is the equivalent of 2GB.

> 📝 ***Note:*** *There are more complex data types, such as DotNet. This is explained in detail in module "Microsoft .NET Framework Interoperability".*

Identifiers, Variables, and Syntax

An identifier is the name of a programming element. A variable is a location in memory where data of varying values is stored. Think of syntax as grammatical rules for using identifiers and variables.

Identifier

An *identifier* is a name that identifies or labels either a unique object or a unique class of objects. Objects, variables, fields, and functions all have identifiers. Some items in a program do not have identifiers, such as constants, operators, and certain reserved words. Instead, these items are referred to directly. Programming elements that refer to data stored in memory require an identifier to access the data; programming elements that exist in the programming code itself do not refer to data and therefore do not need an identifier.

Variable

A *variable* is a value that may change within the scope of a given set of operations. Additionally, a variable has the following characteristics:

- Reference to an actual location in memory where data is stored.
- Name, also known as the identifier, which a developer uses in the program instead of an actual memory address.
- Data type that describes the type of data that can be stored in the memory address.
- Value that is the actual data currently stored in that memory address.

Syntax

Syntax is a set of grammatical rules that defines the programming language. Programming lines or code that follow these rules are said to follow the correct syntax. The computer does not understand code that does not follow the correct syntax and does not compile or execute the code.

The Syntax of Identifiers

You can construct identifiers in C/AL in the following two ways:

- Follow the Pascal syntax. In this method the first character in the identifier must be either an underscore (_) or a letter (either upper or lowercase). Following this first character, developers can select up to 127 additional characters. Each character must be either a letter (upper or lowercase), an underscore, or a digit (a number from zero through nine).

- Do not follow the Pascal syntax. In this method, you must enclose the identifier in double quotation marks (") when you use it within C/AL. You can use any character except "control" characters (characters whose underlying ASCII code is from zero through 31, or 255,) and the double quotation mark character itself ("). You can also use spaces or any punctuation or similar signs. When you use this method, the number of characters in an identifier can still be 128 and do not include quotation marks.

C/AL does not differentiate between upper and lowercase letters in identifiers. Therefore, if there are two identifiers, such as "Account Number" and "Account number," C/AL sees them as.

Within one object, all identifiers must be unique. An identifier cannot be the same as one of the reserved words (such as **BEGIN** or **END**), or an operator (such as **DIV** or **MOD**). If this occurs, a syntax error results. In addition, two identifiers cannot have the same name in an object. If a reference is ambiguous, that is, if the C/AL compiler cannot tell what exact programming element is being referred to, a compile-time error occurs.

Variable Scope

A *variable scope* is the context within a computer program in which a variable name or other identifier is valid and can be used.

Lesson Objectives

Describe the scope of variables.

Global and Local Variables

All variables have a defined scope. This is a defined set of locations where the variable can be accessed. A variable has a global scope if it can be accessed anywhere in an object. A variable has a local scope if it can only be accessed in a single trigger in an object. Variables cannot be accessed outside the object in which they are defined.

System Defined Variables

Certain variables are automatically defined and maintained by the system. An example is Rec, which is found in table objects. Each object has its own set of system defined variables. Developers can use these variables without creating them or initializing their values.

Variable Initialization

In C/AL, you do not have to explicitly initialize a variable before you can use it. Before any C/AL code runs, all the variables are initialized with their default values. Global variables are initialized when you run an object, and local variables are initialized when the trigger, in which it is defined, runs.

The following are the default values for variables of different simple data types:

- For any numeric variables, the value is zero (0).
- For string variables, the value is an empty string (").
- For Boolean variables, the initial value is FALSE.
- For Date and Time type variables, the initial value is 0D (the undefined date) and 0T (the undefined time). The DateTime variable is initialized to 0DT.
- For Option type variables, the initial value is the first option in the list.

Lab 4.1: Investigate Data Types

Scenario

Isaac is a developer for CRONUS International Ltd. He has just learned how to use variables in C/AL, and now wants to practice declaring and using different types of variables. He also wants to learn how the variable values can be shown on screen, and what the initial (default) values are for various data types.

Exercise 1: Data Types

Exercise Scenario

Isaac starts by creating a codeunit, and then defining variables of different simple data types.

Task 1: Create a New Codeunit

High Level Steps

1. Create a new codeunit.
2. Compile and save the codeunit as 90000, **My Codeunit**.

Detailed Steps

1. Create a new codeunit.

    a. In **Object Designer**, click **Codeunit** to open the codeunit list.

    b. Click **New**. This opens the **C/AL Editor** window.

2. Compile and save the codeunit as 90000, **My Codeunit**.

    a. Click **File** > **Save**.

    b. In the **Save As** dialog window, in the **ID** field, enter "90000".

    c. In the **Name** field, enter "My Codeunit", and then click **OK**.

Task 2: Define Variables

High Level Steps

1. Create the global variables named **LoopNo**, **YesOrNo**, **Amount**, **"When Was It"**, **"What Time"**, **Description**, **"Code Number"**, **Ch** and **Color**, of Integer, Boolean, Decimal, Date, Time, Text[30], Code, Char and Option data types.
2. Set the **OptionString** property for the Color variable, to allow for values of Red, Orange, Yellow, Green, Blue, and Violet.
3. Compile and save the codeunit.

Detailed Steps

1. Create the global variables named **LoopNo**, **YesOrNo**, **Amount**, **"When Was It"**, **"What Time"**, **Description**, **"Code Number"**, **Ch** and **Color**, of Integer, Boolean, Decimal, Date, Time, Text[30], Code, Char and Option data types.

    a. On the **View** menu, click **C/AL Globals**. The **C/AL Globals** window opens, where you can declare variables, text constants and functions.

    b. In the **Variables** tab, enter the following information:

Name	DataType	Length
LoopNo	Integer	
YesOrNo	Boolean	
Amount	Decimal	
When Was It	Date	
What Time	Time	
Description	Text	30
Code Number	Code	10
Ch	Char	
Color	Option	

2. Set the **OptionString** property for the Color variable, to allow for values of Red, Orange, Yellow, Green, Blue, and Violet.

    a. Select the **Color** variable.

    b. Do one of the following to open the **Properties** window:

    ▪ Click **View** > **Properties**.

    ▪ Click the **Properties** button on the Toolbar.

    ▪ Press SHIFT+F4.

    c. In the **OptionString** property, enter the following: "Red,Orange,Yellow,Green,Blue,Violet".

    d. Close the **Properties** window, and then close the **C/AL Globals** window.

3. Compile and save the codeunit.

    a. Click **File** > **Save**, or press CTRL+S.

    b. In the **Save** dialog window, verify that the **Compiled** check box is selected, and then click **OK**.

Exercise 2: Display the Variables

Exercise Scenario

After he defines the variables, Isaac wants to see the default values of the variables.

Task 1: Write the Code to Display the Variable Values

High Level Steps

1. Write code that displays the value of all the variables.
2. Compile, save, and close the codeunit.

Detailed Steps

1. Write code that displays the value of all the variables.

   a. Beginning with the first line under the **OnRun** trigger, enter the following code:

Code Example

```
MESSAGE('The value of %1 is %2','YesOrNo',YesOrNo);

MESSAGE('The value of %1 is %2','Amount',Amount);

MESSAGE('The value of %1 is %2','When Was It',"When Was It");

MESSAGE('The value of %1 is %2','What Time',"What Time");

MESSAGE('The value of %1 is %2','Description',Description);

MESSAGE('The value of %1 is %2','Code Number',"Code Number");

MESSAGE('The value of %1 is %2','Ch',Ch);

MESSAGE('The value of %1 is %2','Color',Color);
```

2. Compile, save, and close the codeunit.

   a. Click **File** > **Save**.
   b. In the **Save** dialog window, verify that the **Compiled** check box is selected, and then click **OK**.
   c. Close the **C/AL Editor** window.

Task 2: Run the Codeunit

High Level Steps

1. Run the codeunit 90000, **My Codeunit**, from **Object Designer**.

Detailed Steps

1. Run the codeunit 90000, **My Codeunit**, from **Object Designer**.
   a. In **Object Designer**, select the codeunit 90000, **My Codeunit**.
   b. Click **Run**.
   c. View the messages that display the variable names and values. Consider how different variable types are formatted on screen.

Module Review

Module Review and Takeaways

C/AL is the programming language that is used in Microsoft Dynamics NAV Development Environment. It is found everywhere throughout the application. Like any other programming language, it has its own data types, identifiers, variables, and syntaxes. Understanding standard objects help developers become familiar with how Microsoft Dynamics NAV is programmed and how to follow these standards.

Test Your Knowledge

Test your knowledge with the following questions.

1. What is the programming language of C/SIDE called?

2. List several uses of programming code.

3. Where can you find programming language statements?

4. Which feature of Object Designer do you use to define or change code in an object?

   ( ) Action Designer

   ( ) Action Editor

   ( ) Code Designer

   ( ) Code Editor

   ( ) C/AL Editor

5. Name the three types of triggers in C/AL.

6. What data type do you use to store an employee's birthday?

7. What data type do you use to store an employee's name?

8. What data type do you use to store an employee's salary?

_____

_____

_____

_____

9. What data type do you use to record whether an employee is tax-exempt?

_____

_____

_____

_____

10. Which complex data type corresponds to a row in a table?

_____

_____

_____

_____

11. You must specify length of variable of the Text data type.

( ) True

( ) False

12. Which complex data type provides access to operating system files?

( ) RecordRef

( ) File

( ) InStream

( ) Variant

( ) System

Test Your Knowledge Solutions

Module Review and Takeaways

1. What is the programming language of C/SIDE called?

   MODEL ANSWER:

   C/AL

2. List several uses of programming code.

   MODEL ANSWER:

   Design custom functions; read, write, and change data; transfer data from one table to another; process data and apply business logic to it.

3. Where can you find programming language statements?

   MODEL ANSWER:

   You can find C/AL statements in triggers of most object types in Microsoft Dynamics NAV 2013.

4. Which feature of Object Designer do you use to define or change code in an object?

   ( ) Action Designer

   ( ) Action Editor

   ( ) Code Designer

   ( ) Code Editor

   (√) C/AL Editor

5. Name the three types of triggers in C/AL.

   MODEL ANSWER:

   Documentation, event, and function trigger.

6. What data type do you use to store an employee's birthday?

   MODEL ANSWER:

   Date

7. What data type do you use to store an employee's name?

MODEL ANSWER:

Text

8. What data type do you use to store an employee's salary?

MODEL ANSWER:

Decimal

9. What data type do you use to record whether an employee is tax-exempt?

MODEL ANSWER:

Boolean

10. Which complex data type corresponds to a row in a table?

MODEL ANSWER:

Record

11. You must specify length of variable of the Text data type.

( ) True

(√) False

12. Which complex data type provides access to operating system files?

( ) RecordRef

(√) File

( ) InStream

( ) Variant

( ) System