Visual Basic

An Introduction to Programming

Michael Lappenbusch
IT-SPECIALIST APPLICATION DEVELOPMENT

Table of contents

Introduction to programming with Visual Basic

1.1 What is Visual Basic?

Visual Basic (VB) is an object-oriented programming language and development environment developed by Microsoft. It was first released in 1991 and has since become one of the most widely used programming languages for developing Windows applications.

Visual Basic allows developers to create graphical user interfaces (GUIs) for their applications by dragging controls such as buttons, text boxes, and list boxes onto a form and then linking them to code. This approach is known as visual programming and it makes it easier for developers to create user-friendly applications without requiring in-depth coding knowledge.

Visual Basic supports the development of applications on Microsoft's .NET Framework platform and can be used for both desktop and web applications. It also offers a wide range of libraries and classes that make it easy for developers to create applications with advanced features such as database access and network communication.

Over the years, Visual Basic has undergone several versions that added new features and improvements. The current version is Visual Basic.NET (VB.NET) and was released in 2002. It differs from previous versions of Visual Basic by using the .NET Framework, supporting object-oriented concepts, and using modular, event-driven programming.

1.2 The Visual Basic development environment

The Visual Basic development environment, also known as Visual Studio, is an integrated development environment (IDE) from Microsoft that makes it easy for developers to build, debug, and deploy applications in Visual Basic. It offers a variety of tools and features that enable developers to work more efficiently and productively.

Some of the most important parts of the Visual Basic development environment are:

Solution Explorer: A tool that developers can use to manage and organize their projects and files.

Form Editor: A tool that allows developers to create graphical user interfaces for their applications by dragging controls such as buttons, text boxes, and list boxes onto a form and then linking them to code.

Code Editor: A tool that developers can use to write, edit, and debug their code. It offers features like syntax highlighting, code completion, and auto-indenting.

Debugger: A tool that developers can use to run their code and find and fix errors. It provides features such as the ability to pause code execution, view the values of variables, and track execution steps.

Server Explorer: A tool that allows developers to access databases, web services, and other server resources to connect their applications to data.

Toolbox: A collection of controls and components available to developers when building their applications.

Solution Explorer: An overview of all projects and files that are part of a solution.

Visual Studio comes in several editions, including Community, Professional, and Enterprise, which contain different features and tools.

1.3 Getting started with Visual Basic

Getting started with Visual Basic can be intimidating for a beginner, but with a little guidance and practice, you can quickly gain a basic understanding of the language and development environment.

One of the first steps in learning Visual Basic is installing the development environment, Visual Studio. It can be downloaded and installed from Microsoft website. After the installation is complete, you can start Visual Studio and create a new project.

An easy way to get started with Visual Basic is to create a simple application with a button and a text box. To do this, you can create a new form by clicking on the "Add Project" -> "Windows Form" button. Then you can open the toolbox and drag a button and a text box onto the form.

After the controls are placed, it's time to start writing the code. To perform an action when the button is pressed, one can open the code editor and select the button's "Click" event. Then one can write the code to be executed when the button is pressed, e.g. to fill the text field with a certain text.

This is a very simple example, but it shows how easy it is to create an application using Visual Basic. With practice and further exploration of the language and development environment, one can quickly learn advanced techniques and concepts and build more complex applications.

Basics of Programming

2.1 Variables and Data Types

Variables are an important part of any programming language, and Visual Basic is no exception. A variable is a container in which to store a value. The value of a variable can change during the execution of a program.

In Visual Basic there are different data types that can be used for different kinds of values. Some of the most common data types are:

Integer (Integer): An integer value with no decimal places, e.g. -5, 0, 10.

Single (Single): A floating point value with a precision of 7 decimal places, eg 3.14, -0.01.

Double (Double): A floating point value with a precision of 15 decimal places, eg 3.14159265358979323846.

String: A string of characters, such as "Hello World!".

Boolean (Boolean): A logical value, either True or False.

A variable in Visual Basic is declared by specifying the data type and the name of the variable, e.g.:

```
Dim myInteger As Integer

Dim myString As String
```

After a variable has been declared, one can assign a value by using the variable name and value in an assignment operation, e.g.:

```
myInteger = 5

myString = "Hello world!"
```

It is also possible to declare variables and assign a value at the same time, e.g.:

```
Dim myInteger As Integer = 5

Dim myString As String = "Hello world!"
```

It is important to note that the data type of the variable is determined at the time of declaration and cannot change, and that one should use the variable types according to the assigned value to avoid mistakes.

2.2 Branches and Loops

Branching and looping are important concepts in programming that allow a program to adapt its behavior to specific conditions or to repeatedly execute specific instructions. Visual Basic supports both branching and looping through the use of statements such as If...Then...Else and For...Next, While...End While, and Do...Loop.

Branches allow the behavior of a program to be adapted to specific conditions. In Visual Basic, this is accomplished using the If...Then...Else statement. An example of using If...Then...Else is:

```
If x > 0 Then

'statements to be executed if x is greater than 0

else

'Statements to be executed if x is less than or equal to 0

End If
```

It is also possible to check multiple conditions using the ElseIf statement.

Loops allow certain statements to be executed repeatedly. There are several types of loops in Visual Basic, including For...Next, While...End While, and Do...Loop. An example of using For...Next is:

```
For i = 1 to 10

'Statements that are executed repeatedly

Next
```

The loop runs until the value of i is greater than 10. It is also possible to run a loop with a while condition, which will be executed until a certain condition is met, e.g.:

```
While x < 10

'Instructions to be executed

x = x + 1

End While
```

It is important that the conditions and counters are set and changed correctly to avoid infinite loops or errors.

2.3 Functions and Procedures

In Visual Basic, functions and procedures are both types of blocks of code that perform specific tasks. The difference is that functions return a value while procedures do not return a value.

2.3 Features

Functions are blocks of code that perform specific tasks and return a value. They can be referred to as subprograms that can be called within a larger program. Functions allow code to be organized and reused. A function can be called with arguments that are used as input and can return a value that can be used as output.

Example:

```
Function AddNumbers(ByVal number1 As Integer, ByVal number2 As Integer) As Integer

Return number1 + number2

End function
```

This example defines the AddNumbers function that takes two integer arguments and returns their sum.

2.3 Procedures

Procedures are blocks of code that perform specific tasks but don't return a value. They can also be called subprograms and are used to organize and reuse code. Procedures can be called with arguments taken as input and cannot return a value.

Example:

```
Sub ReturnAMessage(ByVal message As String)

MsgBox(message)

end sub
```

This example defines the procedure "OutputAMessage" that takes a string argument and displays a message box with the passed message.

Advanced Techniques in Visual Basic

3.1 Controls and Forms

There are various controls in Visual Basic that you can place on a form to provide specific functionality. Some examples of controls are text boxes, buttons, list boxes, and check boxes. Each control has its own properties, methods, and events that you can customize and use to control the control's behavior.

Forms are windows that you can use to create user interfaces. You can place controls on a form to provide functionality and transfer data to or from the user. Forms also have their own properties, methods, and events that you can use to control the form's behavior.

Example:

You can place a text box (TextBox control) on a form and configure it to accept only numeric input by setting the AcceptsReturn property to False and the AcceptsTab property to False. You can also add a button (button control) that performs a specific action when the user clicks it by attaching a handler to the control's Click event.

In this example, a form was used to create a user interface that contains a text box and a button. The text field has been configured to only accept numeric input and the button has been programmed to perform a specific action when the user clicks on it.

3.2 Database Access with Visual Basic

Visual Basic offers several ways to access and manage databases. One way is to use ADO (ActiveX Data Objects), a library of components that make it possible to access different types of databases such as MS Access, SQL Server and Oracle. Another option is to use LINQ (Language Integrated Query), a technology from Microsoft that allows database queries to be written in native language.

To access a database using ADO, you must first connect to the database. This can be done using the ADO Connection object. Once connected, you can use the ADO Command object to send queries to the database and use the ADO Recordset object to receive the result. You can also use the ADO DataAdapter object to transfer data between a database and a data container such as a DataSet or DataTable.

Example:

```
Dim cnn As New ADODB.Connection

cnn.Open("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\db.mdb")

Dim cmd As New ADODB.Command

cmd.ActiveConnection = cnn

cmd.CommandText = "SELECT * FROM Customers"

Dim rs As New ADODB.Recordset

rs.Open(cmd)
```

This example connects to an Access database and sends a query to the Customers table. The result is stored in a recordset.

LINQ makes it possible to write queries in native language and offers an intuitively understandable syntax. It uses Language-Integrated Query (LINQ) technology to query data from various sources, including databases, XML, and objects. LINQ supports both querying data and updating data in a database.

Example:

```
Dim db As New DataContext("C:\db.mdb")

Dim query = From c In db.Customers Select c.Name, c.Address
```

This example connects to an Access database and sends a query to the Customers table. The results are the customers' names and addresses.

There are other ways to access databases with Visual Basic, such as using ODBC (Open Database Connectivity) or OLEDB (Object Linking and Embedding, Database), which are also available.

An important thing to keep in mind when working with Visual Basic and databases is to use proper security measures to prevent unauthorized access to data. This includes using secure credentials, using encryption technologies, and limiting permissions for users accessing the database.

Overall, Visual Basic allows data to be stored, queried, manipulated, and manipulated through various ways of accessing databases to create a variety of applications that use databases.

3.3 Creation of Custom Controls

In Visual Basic, you can also create custom controls tailored specifically to your needs. Custom controls allow you to define your own functionality and look and make them reusable. There are two types of custom controls you can create: custom controls that are based on existing controls and custom controls that are built from scratch.

An example of a custom control based on an existing control is an extended TextBox that includes additional functionality such as checking input for numeric value or displaying error messages. To create this custom control, you can create a new class that inherits from the TextBox class and add or override the necessary methods and properties.

An example of a custom control built from scratch is a standalone thermometer control that displays the current temperature. To create this custom control, you must create a new class and add the necessary methods and properties to define the control's behavior and appearance.

It's important to note that creating custom controls is usually a bit more complex than using existing controls and requires some understanding of programming and the technology being used. However, it is a powerful way to customize and extend applications and allows you to save time and resources by creating reusable blocks of code.

Application development with Visual Basic

4.1 Creation of a simple program

Creating a simple program in Visual Basic requires using the Visual Studio development environment, which provides the tools and functionality you need to write, test, and deploy applications.

To create a simple program, you can follow the steps below:

Open Visual Studio and select "New Project" from the File menu.

In the New Project dialog box, select a project type, such as Windows Forms Application for a graphical user interface application. Give the project a name and choose a location.

After the project is created, the Form1 form will open automatically. You can rename it by clicking and editing the "Form1" name in the project explorer.

Add controls such as text boxes, buttons, list boxes, etc. to the form by dragging them from the toolbox onto the form.

Write the code to respond to specific events by double-clicking a control in the form designer and writing the code in the code editor. For example, you can use a button's click event to perform a specific action when the user clicks the button.

Test the program by clicking Start on the Debug menu. You can also set breakpoints in the code to monitor the program flow.

Once the program is tested and bug-free, you can deploy it by compiling and creating a setup file or by publishing it as a portable application.

Note that this is just a simple example and depending on the needs and complexity of the project, it may be necessary to add more steps like using databases, threading, web service integration, etc.

4.2 Creation of a database application

A database application in Visual Basic allows you to store, modify, query, and manipulate data to create a variety of applications that use databases.

To create a database application, you can follow the steps below:

Create a new project in Visual Studio by selecting "Windows Forms Application" or another desired project type.

Create a new database or use an existing one. You can use an Access database or a SQL database (like SQL Server or MySQL).

Connect to the database via ADO (ActiveX Data Objects) or LINQ (Language Integrated Query). You can configure the connection string and connection methods according to the type of database used.

Create tables, indexes, queries, and other database objects required by your application by executing the appropriate SQL statements.

Add controls such as text boxes, buttons, list boxes, etc. to the form by dragging them from the toolbox onto the form.

Write the code to respond to specific events by double-clicking a control in the form designer and writing the code in the code editor. For example, you can use a button's click event to read or write data from the database.

Test the program by clicking Start on the Debug menu. You can also set breakpoints in the code to monitor the program flow.

Once the program is tested and bug-free, you can deploy it by compiling and creating a setup file or by publishing it as a portable application.

It is important to note that security plays a crucial role when building a database application. It is important to implement appropriate security measures to prevent unauthorized access to data, such as using strong credentials, encryption technologies, and limiting permissions for users accessing the database.

Also when developing database applications, it is important to consider the performance of the application and make optimizations to ensure good performance as the number of users or the amount of data in the database increases.

Overall, creating a database application in Visual Basic allows you to store, query, manipulate, and manipulate data and allows you to create a variety of applications that use databases. However, it is important to consider the security and performance of the application and take appropriate measures.

4.3 Creation of a web application with Visual Basic

Creating a web application with Visual Basic requires some preparation and knowledge in different areas. The following are the steps to create a simple web application using Visual Basic:

Select the project type: In Visual Studio, select File -> New -> Project, then select "ASP.NET Web Application" from the list of available project types. Give the project a name and choose a location on your computer.

Select the project template: Select "Empty" or "Web Forms" as the project template. "Empty" allows you to create a custom structure for your application, while "Web Forms" offers a pre-built structure with standard elements such as text boxes and buttons.

Create an HTML page: Create an HTML page for your application by selecting "File" -> "New" -> "HTML Page" in Visual Studio. Here you can define the structure and design of your application using HTML markup, CSS and JavaScript.

Add Visual Basic code: Add Visual Basic code to create your application's functionality. For example, this can be processing forms, querying databases, or processing user actions. You can add Visual Basic code in the code-behind file of your HTML page by clicking the appropriate Code button in Visual Studio.

Test and Deploy: Test your application by launching it in a web browser and verify that it works as expected. If everything is fine, you can deploy your application to a web server for users around the world to use.

It is important to note that this is only a general overview of the steps to create a web application using Visual Basic and there may be additional steps and considerations depending on the application requirements. Some important aspects to consider when developing a web application using Visual Basic are:

Security: Ensure your application is protected against attacks such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).

Database access: Use ADO.NET or another database access tool to access databases and store and retrieve data.

Input Validation and Sanitization: Validate and sanitize all user input to ensure only expected and acceptable data is processed.

Error Handling: Implement error and exception handling mechanisms to ensure your application is stable and robust.

Responsive Design: Make sure your application looks and works well on different devices and screen sizes.

There are many resources and tutorials available online that can help you learn more about building web applications with Visual Basic and help you complete your projects.

Error handling and debugging

5.1 Error Handling with Try-Catch-Finally

Error handling is an important aspect of software development as it helps ensure an application is stable and robust. A common method of error handling in Visual Basic is to use Try-Catch-Finally blocks.

Try block: The try block contains the code where an error can occur. If an error occurs in the try block, execution of the code is handed over to the catch block.

Catch block: The catch block contains the code that is executed when an error occurs in the try block. Here the developer can store specific instructions on how to deal with the error that has occurred, eg outputting an error message or writing the error to a log file. There can be multiple catch blocks that respond to different error types.

Finally Block: The Finally block contains the code that is always executed, regardless of whether the Try block encountered an error or not. Here the developer can, for example, release resources or carry out other clean-up work.

Example:

```
Try
'Code that may fail
Catch ex As Exception
'Code to run when an error occurs
Console.WriteLine("Error: " & ex.Message)
Finally
'Code that is always running
Console.WriteLine("Finally block executed")
end try
```

It's important that the developer understands the type of error that can occur and provides appropriate catch blocks to respond to it. It's also a good idea to only run the statements necessary to handle the error in the catch blocks, and not add extra code that could potentially cause more errors.

It is also important to note that Try-Catch-Finally blocks should not be used for handling unexpected errors due to application architecture issues or unexpected changes in requirements. Instead, these issues should be identified and fixed during the development phase.

There are other methods of error handling in Visual Basic, such as using On Error Resume Next or letting the Application object handle unhandled exceptions. However, it is usually best to use the Try-Catch-Finally method because it provides an explicit and structured way of error handling and allows the developer to react specifically to errors.

5.2 Debugging Tools in Visual Basic

Debugging tools are important for developing software as they enable the developer to find and fix bugs in their code quickly and effectively. Visual Studio, the development environment for Visual Basic, provides a variety of debugging tools that allow developers to step through their code, view variable values, and identify errors.

Breakpoints: A breakpoint allows the developer to pause code execution at a specific point and examine the state of the application. The developer can set breakpoints by simply clicking on the left side of the line of code where he wants the breakpoint to be set.

Debug Window: The Debug window allows the developer to view variable values and other information as the application runs. The developer can open the debug window by clicking "Windows" -> "Debug" in the "Debug" menu.

Debug Controls: Debug controls allow the developer to step through the code. There are controls such as "Step Over", "Step In" and "Resume" that allow the developer to control the execution of the code and pause at specific points to examine the state of the application.

Debug Assistants: Visual Studio also provides a set of debug assistants like "Call Stack", "Watch" and "Immediate" which allow the developer to dig deeper into the code and get specific information to identify errors.

Debug Output: The Debug Output is a window that can be used to print messages that are generated during the application's runtime. This can be very useful to see what is happening in the application and what errors it has encountered.

It's important to note that debugging tools are only part of the development process. Thorough planning and a clean architecture can help reduce the number of bugs encountered during development. Regular code testing and the use of code reviews can also help identify and fix errors early on.

It's also important to note that each developer may have their own preferences when using debugging tools. It is therefore advisable to familiarize yourself with the different tools and to use the ones that are best suited to the individual way of working and the needs of the project.

Some of the tools like breakpoints and debug controls are intended for use during the development phase, while others like debug output and debug wizards can also be used in the production environment to identify and solve problems.

It's also important to mention that Visual Studio also offers support for remote debugging and the ability to use debugging tools with other languages and frameworks.

Extensions and Addons

6.1 Extensions for the development environment

Extensions are add-ons that add functionality to a development environment and make it easier for developers to do their jobs. Visual Studio offers a wide range of extensions developed by the Microsoft community and third parties that allow developers to increase their productivity and simplify their work.

IntelliSense Extensions: IntelliSense is a Visual Studio feature that makes it easier for developers to write code by suggesting code snippets, methods, properties, and more. There are extensions that increase the functionality of IntelliSense by adding additional suggestions and features.

Code Formatting Extensions: These extensions allow the developer to automatically format their code to improve readability. You can set formatting rules for line length, indentation, spaces, and more.

Refactoring Extensions: Refactoring refers to restructuring code to make it more readable, maintainable, and easier to understand and maintain. There are extensions that support various refactoring methods such as renaming, method extraction, and code moving.

Debugging and Profiler Extensions: These extensions extend the debugging functionality of Visual Studio and allow the developer to dive deeper into the code to identify and fix errors. They provide additional features such as the ability to monitor memory usage, measure code performance, and visualize code sections executing.

Project and File Management Extensions: These extensions make it easier to manage projects and files within Visual Studio. They can provide features like browsing projects, managing file versions, syncing files to a remote repository, and more.

Integration Extensions: These extensions allow Visual Studio to integrate with other tools and services such as Git, JIRA, Trello, and Slack. They facilitate project collaboration and management, and allow the developer to access these tools directly from the development environment.

It's important to note that extensions can affect the performance of Visual Studio and that it's important to carefully choose which extensions to install to ensure they support the needs of the project and the way the developer works. It is also recommended to regularly review the installed extensions and remove those that are no longer used in order to improve Visual Studio performance.

6.2 Add-ons for Visual Basic

Add-ons are extensions that increase the functionality of Visual Basic and make it easier for developers to do their jobs. There are a variety of add-ons developed by the Microsoft community and third parties that allow developers to increase their productivity and simplify their work.

UI Controls: There are many third-party add-ons that allow developers to add custom UI controls to their applications, such as charts, tables, menus, and more. These controls can increase the functionality and appearance of the application without the developer having to write a lot of code themselves.

Data Binding: There are add-ons that allow the developer to bind data to their applications from various sources such as databases, web services and local files. These add-ons can reduce development time and increase application flexibility.

Report Generators: There are add-ons that allow the developer to integrate reports into their applications that allow them to view and analyze data from various sources. They can also provide the ability to export reports in various formats such as PDF, Excel, and HTML.

Mailing and SMS Components: There are add-ons that allow the developer to send email and SMS notifications from their application. This can be very useful to notify users about specific events in the application or to send automated notifications.

Security Extensions: There are add-ons that allow the developer to increase the security of their application by providing various security features such as authenticating users, encrypting data and managing permissions. These extensions can protect the application from unauthorized access and data loss.

Cloud Integration: There are add-ons that allow the developer to integrate their applications with the cloud by providing features like connecting to cloud storage services, managing cloud instances, and using cloud-based services . These extensions allow the developer to take advantage of cloud technology to increase the scalability, availability and efficiency of their applications.

It's important to note that add-ons can affect the performance of Visual Basic and that it's important to carefully choose which add-ons to install to ensure they support the needs of the project and the way the developer works. It's also good practice to periodically review the installed add-ons and remove those that are no longer used to improve Visual Basic's performance.

imprint

Author: **Michael Lappenbusch**

E-mail: admin@perplex.click

Homepage: https://www.perplex.click

Release year: 2023