		922337203685477.5807
String	1 Byte per character	0 to approximately 65500 characters 0 to 2E32 on 32 bit systems
Byte	1 Byte	0 to 255
Boolean	2 Bytes	True or False
Date	8 Bytes	January 1, 100 to December 31, 9999
Object	4 Bytes	Any object reference
Variant	16 Bytes + 1 Byte for each character	NULL, ERROR, Any numeric value upto the range of double or any character text, object or array

2.4 MODULES

Code in VB is stored in modules. There are 3 kinds of modules, Class modules, Form modules (Form module is a type of Class module) and Standard modules.

Simple applications can consist of just a single Form and all of the code in the application resides in that Form module. As our applications get larger and more sophisticated, we add additional Forms. Eventually, we might find that there are common codes we want to execute in several forms. We do not want to duplicate the code in all Forms, we create a separate module containing the procedure that implements the common code. This separate module should be a Standard module.

Each Standard, Class and Form module can contain:

- Declarations: We can place constant, type, variable and DLL procedure declarations at the module level of Form, Class or Standard modules.

- Procedures - A Sub, Function or Property procedure contains pieces of code that can be executed as a unit.

2.4.1 Form Modules

Form modules (.FRM extension) are the foundation of any VB application. They can contain graphical descriptions of the form and its controls, including their property settings. They can also contain form level declarations of types, constants, variables and external procedures, procedures that handle events and general procedures. Virtually everything that applies to class module applies to form module. Forms are just class modules that can have controls placed on them and display a form window. Forms are part of our application that are visible to users at run-time.

2.4.2 Standard Modules

Standard Modules (.BAS extension) are containers for procedures and declarations, commonly used by other parts of our application. They can contain global or module level declarations of types, constant, variables, external procedures and global procedures.

2.4.3 Class Modules

Class modules (.CLS extension) are foundation of object oriented programming in VB. We can write code in class modules to create new objects. These new objects can include our own customized properties and methods, although custom objects can not have their own events. All the properties and methods we create can also be used by other objects, in our application. We can also use the keyword, New to create multiple copies of our objects.

2.5 PROCEDURES

We can simplify programming tasks by breaking programs into smaller logical components. These components, called procedures can then become building blocks that let us enhance and extend VB.

Procedures are useful for condensing repeated or shared tasks, such as frequently used calculations, text and control manipulations and data base operations.

There are two major benefits of programming with procedures,

1. Procedures allow us to break programs into discrete logical units each of which can be debugged more easily than an entire program without procedures.
2. Procedures used in one program can act as building blocks for other programs, usually with little or no modification. There are several types of procedures used in VB.

A procedure can be a Sub, Function or Property procedure.

1. Sub procedures do not return a value.
2. Function procedures return a value.
3. Property procedures can return and assign values and set references to objects.

2.5.1 Sub Procedures

The syntax for a Sub procedure is

```
[ Private | Public] [ Static ] Sub procedurename (arguments)
        Statements
End Sub
```

Arguments are a list of argument names, separated by comma, if there is more than one. Each argument looks like a variable declaration and acts like a variable in the procedure.

Each time the procedure is called, the statement between Sub and End Sub are executed. Sub Procedures can be placed in standard modules, class modules and form modules. Sub procedures are by default Public in all modules, which means they can be called from anywhere in the application.

In VB it is useful to distinguish between two types of Sub Procedures, General Procedures and Event Procedures.

General Procedures

A General Procedure tells the application how to perform a specific task. Once a General Procedure is defined, it must be specifically invoked by the application. By contrast, an Event Procedure remains idle until called upon to respond to events caused by the user or triggered by the system.

Event Procedures

When an object in VB recognises that an event has occurred, it automatically invokes the event procedure with the name corresponding to the event. Because the name establishes an association between the object and the code, event procedures are attached to forms and functions.

- An event procedure for a control combines the control's actual name (specified in the name property), an underscore (_), and the event name. For instance if you want a command button named cmdQuit to invoke an event procedure when it is clicked, use the procedure cmdQuit_Click.
- An event procedure for a form combines the word Form and underscore and the event name. If we want a form to invoke an event procedure when it is clicked, we use the procedure Form_Click. Like controls forms do have

unique names, but they are not used in the names of event procedure. If we are using the MDI form, the event procedure combines the word MDIForm, an underscore, and the event name as in MDIForm_Load.

All event procedures use the same general syntax.

Syntax for a control event :

```
Private Sub controlname_eventname (arguments)
        Statementblock
End Sub
```

Syntax for a form event :

```
Private Sub form_eventname (arguments)
        Statementblock
End Sub
```

Although we can write event procedures from scratch, it is easier to use the code procedures provided by VB, which automatically includes the correct procedure's Subnames. We can select a template in the Code window by selecting an object from the Object box and then selecting a procedure from the procedure box.

It is also a good idea to change the names of our controls before we start writing event procedure for them.

2.5.2 Function Procedures

Visual Basic includes system-provided, or intrinsic function, like Sqr, Cos, Chr. In addition we can use the function statement to write our own Function procedures. The syntax for a function procedure is :

```
[Private| Public] [Static] Function Procedurename(Arguments)[As type ]
        Statements
End Function
```

Like a Sub procedure a Function procedure is a separate procedure that can take arguments, perform a series of statements, and change the value of its arguments. The arguments for a Function procedure work in exactly the same way as the arguments for a sub procedure. Aside from the Function keyword there are three differences between Sub and Function procedures:

- Generally we call a function by including the function procedure name and arguments in the right side of the larger statement or expression.
- Function procedures have data types just as variables do. This determines the type of the return value. (In the absence of an As clause, the type is the default Variant type)
- We return a value by assigning it to the procedure name itself. When the function procedure returns a value, this value can then become part of a larger expression.

For example, We could write a function that calculates the third side, or hypotenuse of a right angle triangle given the values for the other two sides.

```
Function Hypotenuse ( A,B)
        Hypotenuse= Sqr(A ^ 2 + B ^ 2)
End Function
```

We call a Function the same way we call any of the built in functions.

Example

```
Label1.Caption = Hypotenuse(CInt(Text1.Text),CInt(Text2.Text))
        X= Hypotenuse(Width, Height)
```

2.5.3 Property Procedures

We can use Property Procedures to create and manipulate custom properties.

- Property procedures can be used to create read only properties for forms, Standard Modules and Class Modules.
- Property Procedures should be used instead of public variables whenever we have code that must be executed when the property value is set.

When we create a Property procedure, the new property is a property of the module containing the procedure. VB provides three kinds of Property procedures as given below

Property procedure	Description
Property Let	A procedure that sets the value of a property
Property Get	A procedure that returns the value of a property
Property Set	A procedure that sets a reference to an object

Read Only Properties

We can use property procedures to add customize read only property to objects. This is useful in any situations where we want the property to be read-only for users and/or we want related code to be executed when the property is set.

2.6 CONTROL STRUCTURES

Control Structures allow us to control the flow of our programs execution. If left unchecked by control flow statements, the program's logic will flow through statements from left to right, and top to bottom. When some very simple programs can be written with only this uni-directional flow, and while some flow can be controlled by using operators to regulate precedence of operations, most of the power and utility of any programming language comes from its ability to change statement order with structures and loops.