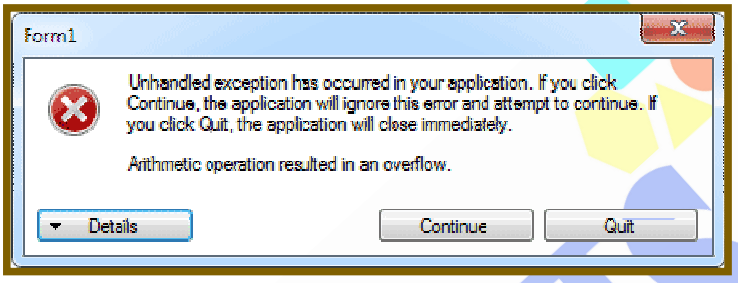
Module 1 - Error and Debugging Tools in Visual Basic

An error in programming refers to a problem or mistake in the code that prevents the program from working correctly. It can cause the program to:

- Fail to compile
- Crash during execution
- Produce incorrect results



Types of Error in Visual Basic

- **Syntax Error:** A mistake in the code's grammar that prevents it from compiling.
- **Runtime Error:** An error that occurs while the program is running, like dividing by zero.
- **Logical Error:** A flaw in the program's logic that produces incorrect results without crashing.
- In Visual Studio 2022, errors are usually shown in the **Error List window** with red or blue symbols.

Types of Errors by Context:

Type	What It Means	When It Happens
Syntax Error	The code is written incorrectly (bad grammar)	When you write the code
Runtime Error	The program runs but encounters a problem	When the program is running
Logical Error	The program runs but gives wrong results	When the logic is flawed

Visual Basic - Advanced

1. Syntax Errors
These are mistakes in the code structure or grammar.
• The compiler cannot understand what you've typed.
• These errors stop the program from running.

Examples:

Dim x = 5 ' ✅ Correct
Dim x = ' ❌ Syntax Error (missing value)

Visual Studio shows a red underline and a message like:
"Expression expected" or "Expected end of statement"

2. Runtime Errors
These occur while the program is running.
The code is correct in form, but something goes wrong when you execute it.

Examples:
• Trying to divide by zero
• Trying to open a file that doesn't exist
• Using a null object (nothing is assigned)

Example in VB.NET:

```
Dim num As Integer = 5
Dim result As Integer = num / 0 ' ❌ Runtime Error: Division by zero
```

The program crashes and may show a message like:
"System.DivideByZeroException"

3. Logical Errors
These errors occur when the program runs without crashing, but the output is incorrect because of a mistake in logic.
• These are the hardest to find because the program doesn't show any error message.

Examples:
• Using + instead of -
• Displaying the wrong value

Example:

```
Dim a As Integer = 10
Dim b As Integer = 5
```

Dim sum As Integer = a - b ' ❌ Logical Error (should be a + b)
The program runs, but gives wrong result (5 instead of 15)

How Visual Studio Helps You

- **Red underline** = Syntax error
- **Yellow warning** = Potential issue
- **Error List panel** = Shows all errors and warnings
- **Break mode** and **exception messages** help you find runtime issues

Debugging Tools in Visual Studio 2022

What is Debugging?

Debugging is the process of finding and fixing errors (bugs) in your program. Visual Studio has powerful tools to help you do this easily.
When you debug a program, you can:

- Pause the program at a certain line
- Check the value of variables
- Step through your code line by line
- See where and why the program crashed

Debugging Tools in Detail

1. Watch Window

✅ **What It Is:**
The **Watch Window** allows you to **monitor specific variables** and their values **while debugging**. You can choose which variables or expressions you want to "watch."

Why Use It:

- To keep an eye on important variables.
- To see how their values change as the program runs.

Features:

- You can **add any variable or expression manually**.
- Works even if the variable is not in the current scope (shows "not in scope").
- You can **watch multiple variables at once**.

How to Use:

1. Set a **breakpoint**.
2. Run the program (**F5**) — it will stop at the breakpoint.
3. Go to **Debug** → **Windows** → **Watch** → **Watch 1**.
4. Type variable names (like x, price, student.Name) in the Watch window.

2. Locals Window

What It Is:
The **Locals Window** shows **all variables that are currently in scope** (available) at the paused line.

Why Use It:
- To **automatically see all variables and their values** without typing them.
- Helps identify wrong or unexpected values easily.

Features:
- Lists **variable names**, **values**, and **data types**.
- Refreshes automatically as you step through the code.
- Useful for beginners who don't want to manually add watches.

How to Use:
1. Run the program with a **breakpoint** set.
2. When paused, go to **Debug → Windows → Locals**.
3. View all current variables and their values.

3. Immediate Window

What It Is:
The **Immediate Window** lets you **type and execute code manually** while debugging.

Why Use It:
- To test simple expressions or calculations.
- To change the value of variables on the spot.
- To call methods or check conditions without changing your code.

Features:
- Works **during break mode** (when the code is paused).
- Helps you debug without restarting the program.
- Can **assign values**, **print results**, or **evaluate logic**.

How to Use:
1. Run the program and pause at a **breakpoint**.
2. Go to **Debug → Windows → Immediate**.
3. Type and run commands like:

```
? x + 10        ' Shows result
x = 50          ' Changes the value of x
MsgBox("Test")  ' Runs the message box
```