

Contents

1	Diving In – The Visual Studio IDE	1
2	A Note on Programs in General	2
3	Diving In – C++	2
3.1	C++ as an <i>Actual Language</i>	2
3.2	Programming: Like telling a Foreign Butler what to do	3
3.3	Programming as <i>Problem Solving</i>	3
3.4	Ideal Software	4

1 Diving In – The Visual Studio IDE

- We'll be learning C++, and using the MS Visual C++ compiler and IDE for writing our programs
 - MS Visual Studio is *very popular* in industry for writing C++ programs
 - As such, *we're not just learning a language, but also Visual Studio!*
 - What's an IDE?
 - Stands for *Integrated Development Environment*
 - Basically, it's a piece of software that tries to help you write software
 - IDE's are intended to make it very easy to write, compile, run, and debug programs
-

- Of course, not everything succeeds at their goals
 - IDE's can often complicate matters tremendously
 - For the first 2 weeks, experience tells me that most of you (and myself) will have countless problems that can all be attributed to *Visual Studio*, and not C++!
 - **NOTE:** This is common with any IDE, not just Visual Studio
 - If you're having "weird" problems, it's probably a problem with the way Visual Studio (or your "project") is setup, and *not* your actual program
-

- To try and help everyone get over this "hump" quicker, I've put together an exhaustive visual reference, which can be found on the course website.
 - I'll go through this live in a minute, and discuss some of the common problems
-

2 A Note on Programs in General

- First, keep in mind that the actual C++ source code is just a file
 - A plain-text file, with no formatting (color, font specification, tables/figures, etc.)
 - It's just text!
 - Most programming editors (including the one we'll use) add *syntax highlighting* to make source code a bit easier to read
 - Keep in mind that the colors you see are not part of the code itself – the IDE is doing all of that
-

- Once you have your source code written in a file, you'll *compile* the file into an actual program
 - If your file has any errors in it (perhaps you speld someting wrong), the compiler will complain, and will not be able to produce a program
 - The complaints from the compiler will be listed, with hints at where your errors may be at
 - Once your code successfully compiles, you'll have a working program that can be run
-

3 Diving In – C++

A quick note about your programs...

- Most programs you are probably familiar with have a *GUI*... A Graphical User Interface
 - You use your mouse to move through menus and “do stuff”
 - Although almost all modern software has a GUI, we'll be doing something more basic at first
 - Instead of a GUI, we'll be running things at a command line
 - Don't worry, *once you know the basics of writing software, creating a GUI for your program is relatively easy*
-

3.1 C++ as an *Actual Language*

- As stated multiple times, when programming, we are instructing the computer to do something
- We instruct the computer to do this via a very specific language
- The language is similar to a natural language (English, French, etc.) except for the following:
 1. **The vocabulary is *extremely* small.** There are only a handfull of keywords that are basic to the language itself. We can, as programmers, *extend* this vocabulary, however.
 2. **The grammar is *extremely* precise.** Pieces of the language fit together only in a very specific fashion.
 3. **There is no “slang”.** That is, if you use something not in the vocabulary or use improper grammar, you don't get a kinda-sorta program. Rather, the compiler will not recognize the code as C++! It will complain about any errors, and *not* produce a program!

A note on the last point. . .

- The fact that the compiler only recognizes *absolutely* valid C++ may seem cumbersome, or even scary
 - But it's much more of a benefit than anything
 - First, this will make learning C++ very easy. . . if you use any incorrect vocabulary or bad grammar, the compiler will let you know immediately
 - Second, anytime you make such a mistake, the compiler will tell you roughly what the error is and where it's at!
 - The errors reported by the compiler are *usually* pretty accurate, although they can occasionally be wildly off the mark.
-

3.2 Programming: Like telling a Foreign Butler what to do

It's good to think of programming in the following way. . .

- You have a butler who is extremely fast, and capable of doing almost anything you can dream
 - All you have to do is tell the butler what you want
 - **Problem:** the butler speaks a language you do not know
 - **Worse:** the butler tolerates no error in speech
 - **Slightly Better:** If you tell the butler to do something but there's an error in your speech, the butler lets you know what/where the error is
-

3.3 Programming as *Problem Solving*

- Learning a programming language is extremely easy
 - Learning to write programs to solve problems, however, is a little more difficult
 - Note the following. . .
-

1. **Understanding.** Before we can tell a computer to solve a problem, **we must first understand what the problem is for ourselves!**
You think this would be obvious, but it's one of the biggest causes of errors in modern software!
 2. **Design.** Once we understand the problem, **we must figure out how to solve the problem on our own, before we could ever think about telling a computer to solve it!**
If you can't solve the problem, then you're not going to be able to tell the computer how to solve it!
 3. **Implementation.** Only once we understand the problem and how to solve it can we begin to tell a computer, via programming, **how it can solve the problem.**
You must understand the problem and how to solve it in a very precise manner, as you must be precise when telling a computer to do anything!
 4. **Testing.** Nobody is perfect. Just because your program compiles and runs **does not mean it's doing the correct thing!**
We must test our programs *exhaustively* before we even think they are finished.
-

3.4 Ideal Software

There's an old saying in computer science...

- *Ideal software is designed by philosophers...*
- *implemented by mathematicians...*
- *and tested by idiots.*