**Lecture 2: Chapter 2: Introduction to C Programming – Mon Aug 23 or Tues Aug 24**

**Announcements**

Reading:

* Chapter 2

Assignments:

* Assign: Assignment #1 – due on **Aug 30** (MW class) or **Aug 31** (TR class) **(no late assignments accepted)**

**Today’s Goals**

1. Setting up your C Programming Environment
2. Setting up first program in Eclipse
3. Write program to display information
4. Write program to receive information

**Today’s Terminology**

**Terminology**

* Preprocessor Directive
  + Lines in a C program that are processed by the preprocessor **before** compilation
  + Any line that begins with a #
  + We will use these when including header files
* Function
  + Functions contain statements that perform a task
  + C programs contain a “special” function called “main
    - **int** **main**(**void**)
  + The “main” function is where a C program starts execution
  + Note, in C we use the term **function**, in Java we use the term **method**
* Blocks
  + Code that is between an opening curly brace and closing curly brace - { }
* Statements
  + Statements represent actions
  + Statements must end with a semicolon
* Header File
  + A file with the .h extension
  + Contains C declarations that can be shared with source files
* Source File
  + A file with the .c extension
* String
  + Also called a character string
* Reserved Words (Keywords)
  + Words that have a specific meaning in the language and cannot be used for other purposes
  + int, float, void, if, while
* Identifier
  + An identifier is **basically a name** chosen by the programmer for variables, constants, functions
  + Follows rules specified by the language
* Variable
  + Used to store a value that may change
  + Variables are given **names** based on the rules of the language
  + Variables have a **type** to indicate the type of values that can be stored in the variable
* Variable Definitions
  + A statement that defines a variable giving it a name and a data type
  + **int** numberOfStudents;
* Data type
  + Specifies the type of data stored in a variable such as integer
* Assignment Statement
  + Way to get a value into a variable
  + variable = expression;
* Arguments
  + The values that are specified when calling a function
  + Values being sent to a function

**Setting up Eclipse Development Environment on Personal Computer**

**Development Environment**

* Details for Eclipse are documented in a separate document on Blackboard
* **Note:** I had an older version of Eclipse and the CDT (C/C++ Development Toolkit) so some of the issues I ran into might have been corrected. A new version of the CDT was just releases in July 2017.

**First C Program in Eclipse**

**What we do in C**

1. Type in source code using editor – using Eclipse in this example
   1. Create a project
   2. Create a source **.c** file
      1. C program file names should end with the **.c** extension
   3. Build source code
      1. Building is done in Eclipse
   4. Run
      1. Running translates the machine code into something that the platform understands

**Simple program to get started**

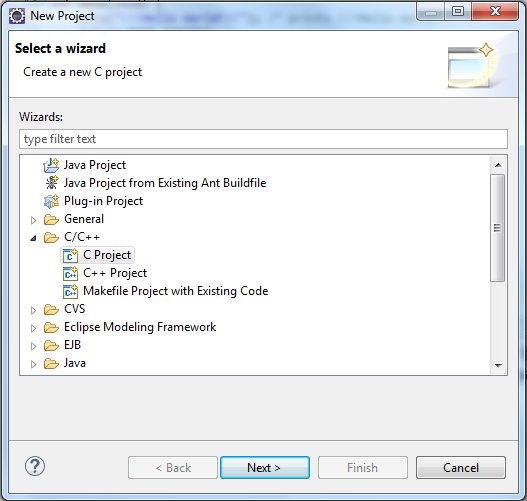
* Quick look at programming from a general view
* Not a lot of details
* Everything will be covered in more details later

**Using Eclipse to write 1st program**

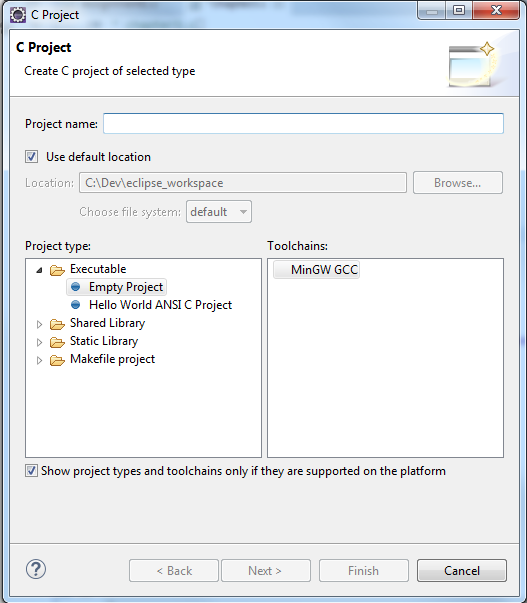
1) Start Eclipse

2) Create a C Project

* **File->New-> Project** to display the New Project dialog
  + Select **C project** and click next



* The **C Project** dialog will appear

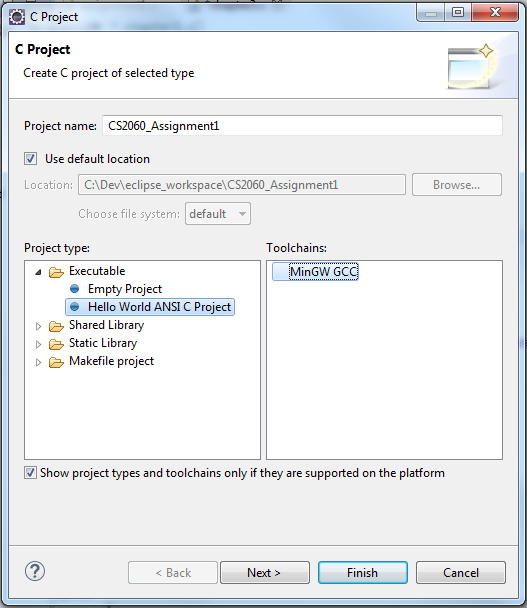


* Set the following
  + Project name - give your project a name – **CS2060\_Assignment1**
  + Default location - should be set to the workspace you selected when started Eclipse
  + Project type – **Hello World ANSI C Project**
  + Toolchain – **MinGW GCC**

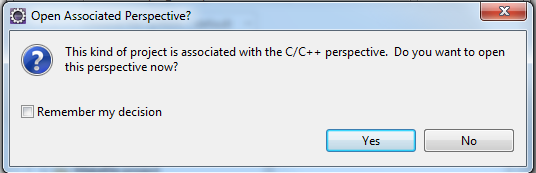
Note: Toolchain is a set of tools – compilers, linker, assembler - intended to build your project.

There could be several listed here – we want to use the MinGW GCC toolchain we installed.

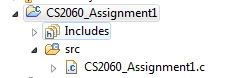
* + Click "Finish" to create the project



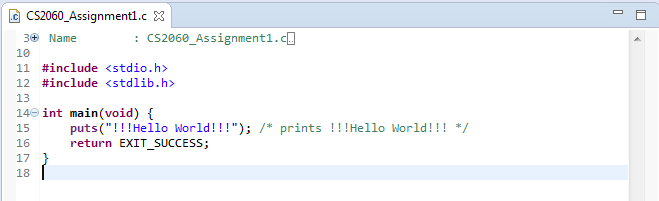
* It might now ask you if you want to switch perspective
* Click **Yes** since you are now working in C, not Java



* In Eclipse, you will see in Project Explorer pane



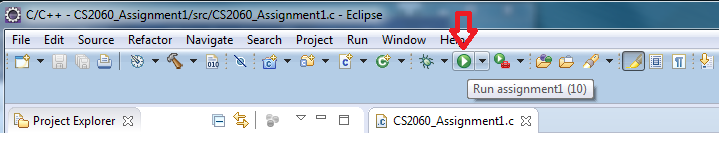
And in the right editor window the code for source code file **CS2060\_Assignment1.c**



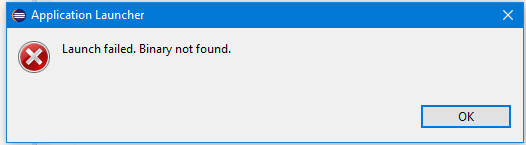
* Note: Go to workspace directory on your hard drive you will see your project **CS2060\_Assignment1**

3) Build Project

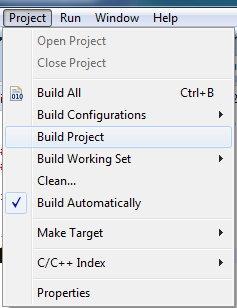
* If you are used to Java, normally you would now click the Run button



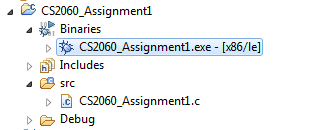
* But with C/C++ if you Run you will see launch error because you need to first **build** the program



* To build, go to menu and select **Project** -> **Build Project** or click on build icon 



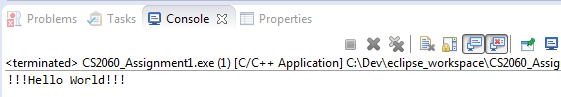
* This will build your program and generate a binary (.exe) – this is added under the Program Explorer



* If the build runs into any problems, you will see these in the Problems view.
* **\*\*\*Note: Issues with Norton\*\*\***
  + I ran into problems with Norton when I attempted to build my project.
  + I’ve documented these issues below in the **Eclipse Issues** section

4) Run

* To run go to menu and select **Run** -> **Run** or click on the run icon 
* You should see the output in the console

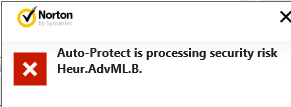


**Eclipse Notes – A Few Tips**

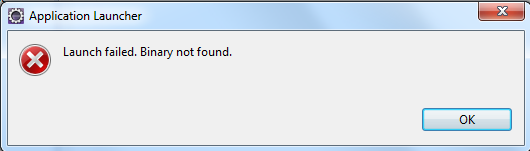
* To find the “Welcome” screen – the one with Overview and Tutorials
  + Help -> Welcome
  + For tips and basic information click **Workbench Basics**
  + Next click on **Getting Started,** then **Basic Tutorial**
* Use the C perspective
  + Window -> Open Perspective -> Other -> C/C++
  + A “perspective” is just a layout of views
  + A “perspective” controls what you see in certain menus and toolbar
* If lose a view, click on **Window -> Show View**
* Clicking in a view makes it active

**Eclipse Issues**

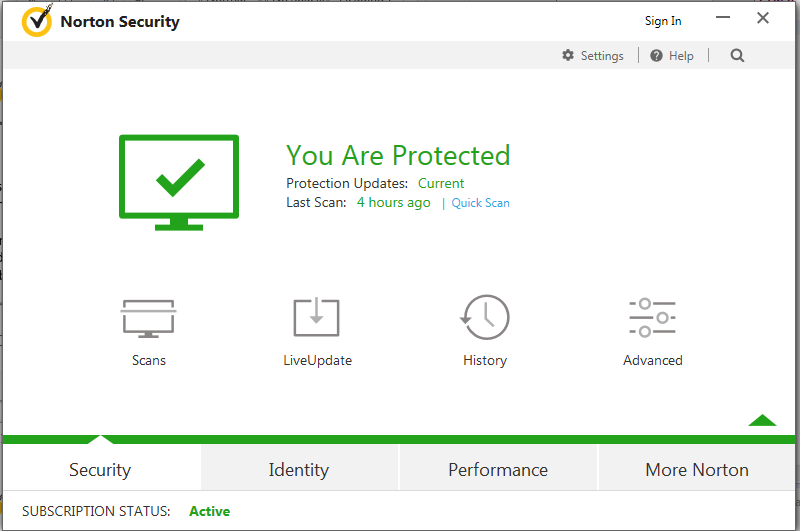
* **Eclipse doesn’t recognize changes**
  + Eclipse seems to have issues refreshing when changes are made and sometimes old issues remain displayed on the screen.
  + To fix this problem do the following
    - Make your changes
    - Save the file
    - Select Project -> Clean…
    - Select the project to clean
    - Select Project -> Build Project
    - Select Run
* **Issues with Norton**
* Norton was giving me problems with step 3
* When building the project, Norton would display this message and remove the exe file!



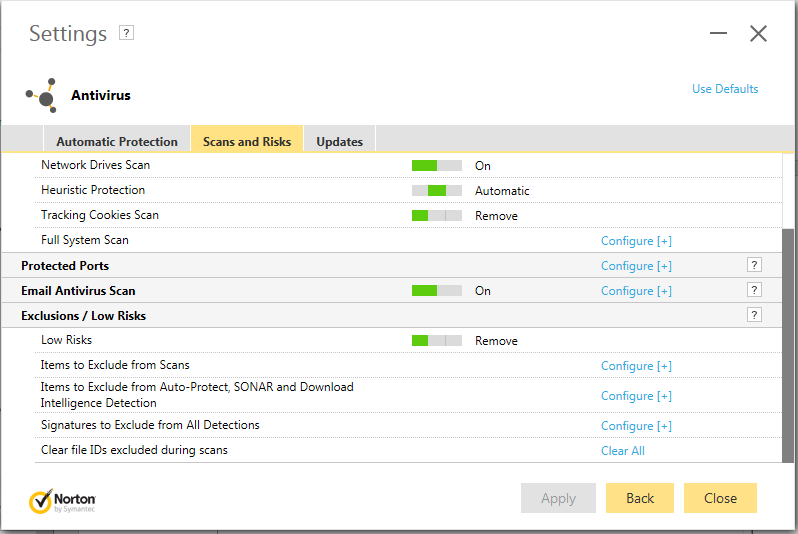
* When I tried to run the program, I would see this error because the .exe was gone!



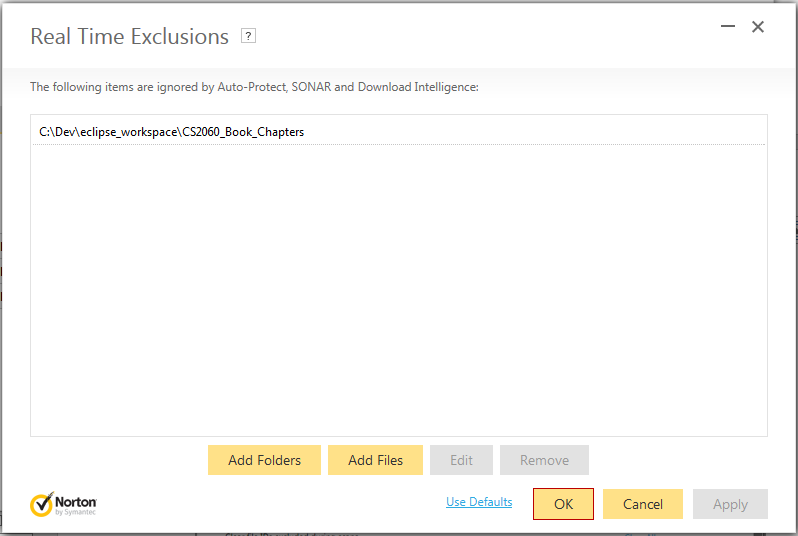
* Norton determined the .exe was bad and removed it
* If you have this issue, do the following to fix it:
  + Open Norton
  + Click **Settings - > Antivirus**



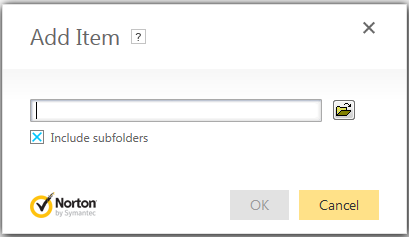
* + On Settings screen select **Scans and Risks** tab



* + Scroll down to Exclusions/Low Risks
  + For **Items to exclude from Auto-Protect, SONAR and Download Intelligence Detection** - click on “Configure +”
  + In **Real Time Exclusions** window - click Add Folders



* + Click on browse button and find folder you want to exclude from this insanity!



* + Go back to menu and select **Project -> Build Project**
  + Your .exe file should no longer disappear

**A Simple C Program - Details**

**Simple Program**

**#include** <stdio.h>

// All functions need to have a comment describing their purpose except main!

**int** **main**(**void**)

{

// This is one way to create a comment in C

**printf** ("This is my first C programming class \n");

/\* This is another way to create a comment in C \*/

**printf** ("And this is my first C program");

}

**What These Lines of Code Mean – In English Please**

**Line Purpose of the line**

**1** **#include** <stdio.h>

**#include is a preprocessor directive**

Include the contents of the standard input/output header file

Tells the preprocessor to include the contents of the **stdio.h** file in this program

Stdio.h contains standard input/output library functions such as **printf**

Use angle brackets since stdio.h is a standard library file

**2 Blank Line**

Blank lines are ignored by the compiler

Your programs must contain white-space to make them easy to read

**3** // All functions need to have a comment describing their purpose except main!

**Comment**

Comments are used to document your programs

Comments are ignored by the complier

Two ways to create comments double slash // or the pair /\* and \*/

If you use // the complier ignores everything following the // on that line

If you use /\* and \*/ the complier ignores EVERYTHIN between /\* and \*/

**4 int** **main**(**void**)

**Define the main function**

This is the point where the program starts to execute (run).

You can have many functions in your program

You can have ONLY one main function

Functions can return values - **int** indicates that our main function returns an integer

Functions can receive values - **(void)** indicates that our main function receives no values

**5 {**

**Opening curly brace**

Used to start another block - in this case the "main function block"

There must be a matching closing curly brace - it is in line 11

**6** // This is one way to create a comment in C

**Comment**

Comments must be used to document sections of code

If you use // the complier ignores everything following the // **on that line**

This comment style is the preferred style

**7 printf** ("This is my first C programming class \n");

**Executable statement**

**printf** is an executable statement that gives you the ability to display information

In this case, the character string in quotation marks will be displayed

The \n is used to create a new line – it’s like pressing the enter key when writing in Word

Statements must end with a semicolon

Note: **printf** is a standard library function – it already exists – it is found in the **stdio.h** file

The linker not the compiler knows where this function is located

**8 Another blank Line**

**9** /\* This is another way to create a comment in C \*/

**Another comment**

Shows the other way to write a comment

If you use /\* and \*/ the complier ignores EVERYTHIN between /\* and \*/

**10 printf** ("And this is my first C program");

**Another executable statement**

**11 }**

**Closing curly brace**

Closes the "Main block"

**Program Output**

This is my first C programming class

And this is my first C program

**C Notes**

* Statements must end with a semicolon
* Reserved words
  + Certain words are reserved in C - they are called **reserved words** or **keywords**
  + They cannot be used for any other purpose and their meaning cannot be changed
* **puts** is a function that was already created - it’s part of the C library available for your use.

**Errors**

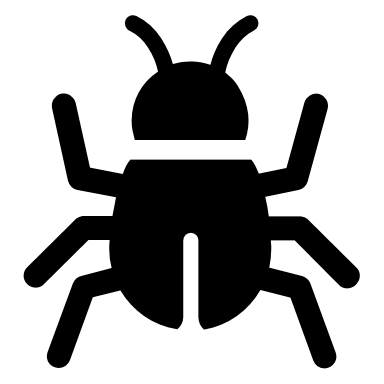
* Syntax errors
  + When you violate the syntax of the language
  + For example, when you leave off a semicolon, have a typo in your code, leave off a curly brace, etc.
  + Detected by the complier
    - shown in Eclipse as a yellow "?" - hover over "?" for information
* Runtime errors
  + Occur when the code is running
  + Causes code crashes - will see message in console and possibly a yellow diamond
  + For example, if you attempt to divide by 0 your code will have a runtime error
* Logic errors
  + Occurs when your code does not produce the expected result

**Another Simple C Program - Reading Input from the Console**

**To display information to the outside world**

* **printf** => sends output to console

**To obtain information from the outside world**

* **scanf =>** read input from the keyboard
  + Input from keyboard is the default
* **Note**: there is a bug in Eclipse when using scanf ****
  + The console has buffer problems in Windows
  + When you include scanf in your program and run it, the printf statements are not executed until after the scanf
  + This makes it look like you are not prompting the user!
  + To fix this issue, add the following line to your code:

**setvbuf** (stdout, NULL, \_IONBF, 0);

**Steps for obtaining input**

* Ask user for input – this is called *prompting the user*
* Use **scanf** library function to read user input
* Do something with the input received from the user

**Reading Input Example**

**#include** <stdio.h>

// All functions need to have a comment describing their purpose

**int** **main**(**void**) {

// There is a bug in Eclipse when you use scanf function

// The Eclipse console has buffer problems on Windows

// The problem is the printf statements are not executed until after the scanf

// Add the following line of code at the start of your program

**setvbuf** (stdout, NULL, \_IONBF, 0);

// Variable to store the user input

**int** numberOfStudents;

// Prompt the user for the number of students in the class

**printf** ("Enter the number of students in the class\n");

// Read the integer value the user enters

**scanf** ("%d", &numberOfStudents);

// Display the value the user entered

**printf** ("Can you believe %d students showed up for class.\n",

numberOfStudents);

}

**What These New Lines of Code Mean**

**Line Purpose of the line**

**10 setvbuf** (stdout, NULL, \_IONBF, 0);

**Workaround for issue in Eclipse**

Include the **setvbuf** statement to force Eclipse to handle scanf properly

**13 int** numberOfStudents;

**Variable Definition**

Must declare all variables before their use

Give variables a data type and meaningful name

**16 printf** ("Enter the number of students in the class\n");

**Executable statement**

**printf** is an executable statement that gives you the ability to display information

In this case, prompting user for input

Displays prompt then positions cursor on the beginning of the next line

**19 scanf** ("%d", &numberOfStudents);

**Executable statement**

**scanf** is an executable statement that gives you the ability to obtain information from user

When this statement is executed, program stops and waits for user input

User entered value will be stored in memory location associated with variable **numberOfStudents**

Note: **scanf** is a standard library function – it already exists – it is found in the **stdio.h** file

The linker not the compiler knows where this function is located

**22 printf** ("Can you believe %d students showed up for class.\n",numberOfStudents);

**Executable statement**

**printf** is an executable statement that gives you the ability to display information

In this case, displaying information as well as the value stored in a variable

The **%d** indicates that an integer value will be displayed

In this case the value stored in **numberOfStudents** is displayed

**Program Output**

Enter the number of students in the class

32

Can you believe 32 students showed up for class.

**Notes**

* **int** numberOfStudents;
  + Defines a variable named **numberOfStudents**
  + Sets aside a location in memory called **numberOfStudents**
  + The **int** indicates that integer values can be stored in that location
* **printf** – function for output – default is console
* **scanf** – function for input – default is keyboard
* scanf arguments
  + “%d”
    - This is called a **format control string**
    - Indicates the type of data the user should enter
    - The “d” indicates an integer value is expected
  + &numberOfStudents
    - The ampersand is called the **address operator**
    - You combine & with variable to indicate memory address where the variable **numberOfStudents** is stored
    - Including the & is a bit different from other languages