Today in CS161

- ***Week #2***
  - **Solving Problems with Computers**
    - What are Algorithms
    - Write a short **Algorithm**
- ***Preparing to write programs***
  - On Unix and Mac systems
- ***Assignments***
  - Walk through **Homework #1**
  - Ethics

Solving Problems w/ Computers

- Programs are...
  - an expression of a series of instructions that the computer must perform
  - written in precise languages, called **programming languages**
- Programming languages...
  - translate your ideas into specific language that the computer will understand
  - C, C++, Java, Pascal, Visual Basic, Cobol, Fortran

Solving Problems w/ Computers

- The most difficult part of programming is figuring out how to design a method to solve a problem
- Only then do we translate this into C++!
- Therefore, start by writing an **algorithm** once you understand the problem

Solving Problems w/ Computers

- An **algorithm** is a sequence of step by step instructions for performing some task -- usually for computation
- An **algorithm** must...
  - accomplish the task
  - be clear and easy to understand
  - define the sequence of steps needed to accomplish the task in the order specified

An Algorithm is…

- **Algorithm** – an ordered set of actions designed to accomplish a certain task. The method for long division is an algorithm, so are cooking recipes, filling out an application, or going through a sequence of motions required to throw a curveball.
- Algorithms are vital as the complexity of our problems increase. Think about making a cake without a recipe. We may quickly lose our way or forget a vital ingredient. Same goes for programming. It has always taken me far longer to write code without an algorithm, throw it away, and write it again than it has to write the algorithm first, desk check it, and then finally sit down and turn it in the code.

Algorithms

- It is best to **ignore** the details of your programming language (C++) when trying to solve a problem
- Instead, figure out the steps you need to go thru to solve the problem
- Write these steps down in English
- These steps are called the algorithm!

Algorithms

- Think of your algorithm as a tool for creating the instructions for solving a problem....as if you were to tell them to another person.
- Remember an algorithm is a sequence of step by step instructions for performing some tasks

Convert Inches to MM

We will create an algorithm in class:

Important Terms…

- **Program** – is a sequence of instructions in a language (we will be using a high level language C++ for most situations in this course). A program is used to implement one or more algorithms.
- **Source code** is the text of a program that is created by a programmer. It is legible to human eyes but the computer cannot execute it directly.
- **Compiler** is a special program that takes source code and translates it into object code that the computer that you are using can understand and execute.

Important Terms…

- **Object code** is a program that has been translated into machine-readable form by the compiler. It consists of sequences of binary numbers and therefore cannot be read by us.
- **Function** is a logically self consistent and somewhat independent part of a program. Maybe it is something that needs to happen over and over and you don’t want to write it over and over. Every C++ program has at least one function called main.

Important Terms…

- **Library** is a collection of functions that can be added to a program with ease. C++ includes many built-in libraries that allows others to use them (so we don't have to recreate the wheel)
- **Bug** is a mistake in a program. Many years ago, an inspect flew into a computer and caused a malfunction; the programmer who discovered it coined the term and it has been used ever since. There are three main types of bugs: syntax…like making a typo, grammatical…not using concepts in an order that makes sense, and logical – where the use is valid but the results are not what we were looking form

Today in CS161

- ***Next Topic:***
  - **Solving Problems with Computers**
    - Walk through the Tic Tac Toe Algorithm
- ***Getting ready for Creating Programs***
  - Turn the Inches to MM algorithm into a program!
- ***Preparing to write programs***
  - See Examples of creating programs with Dev C++ and Unix

Tic Tac Toe Algorithm

- Step A: Display a Message letting the user know the:
  - Welcome and Rules
- Step B: Initialize the Game with an empty board
  - Display Board with 2 vertical and 2 horizontal lines
  - Clear the Board making sure there are no X's or O's in the game positions. Clear all memory that will be needed so that it is an empty slate.
  - Select a Player to start: Give them a choice of an X or an O
- Step C: Start Playing until there is a Winner or Cat Scratch
  - Display a Message:
    - Tell the correct player to select a location
  - Choose Mouse Location (Wait for the mouse button to be hit)
  - Find the Location on the Board
  - Check to see if that Location is Available
    - If available, Move Piece and Switch Players
    - Otherwise display a message to retry
  - If it was a Cat Scratch, start back with step B.
- Display a winner's message and end the game!

Creating programs…

- The first step of program development is to write an algorithm.
- Think of your algorithm as a tool for creating the instructions for solving a problem....as if you were to tell them to another person.
- Remember an algorithm is a sequence of step by step instructions for performing some tasks

Here is the algorithm we did…

- Convert inches to millimeters
  - First understand the problem
    - where do the inches come from (the user)
    - what is the math needed for the conversion
      - mm = 25.4 times inches
    - how do we want to display the results
      - `2in convert to 50.8mm`

Convert inches to millimeters

- Next, write the algorithm
  - Step 1: Welcome the user
    - tell them what to expect
    - tell them the purpose of the program

Convert inches to millimeters

- Step 2:
  - Get the number of inches from the user
    - display a **prompt** asking the user to enter
    - **read** in the number of inches
    - Remove the newline from the input buffer
    - display what was read **(echo)**
    - ask the user if this is really correct **(confirm)**
    - if not, repeat this step until the user is satisfied

Convert inches to millimeters

- Continuing with Steps 3 and 4:
  - Convert the number of inches to mm
    - mm = 25.4 times inches
  - Display the results
  - Provide a sign-off message

Convert inches to millimeters

- The next step is to turn this into a C++ program!
- All programs have the following “form”

```
#include <iostream>
using namespace std;
//header comments...
int main()
{
  //program body goes here...
  return 0;
}
```

Convert inches to millimeters

```
#include <iostream>
using namespace std;
//  ********************************
//   Karla S. Fant
//   CS161 Programming Assignment #0
//   Purpose of this program is to convert
//   inches entered in by the user into
//   millimeters and display the results
//   ********************************
int main() {
```

(Different Kind of Comment...)

```
#include <iostream>
using namespace std;
/*  ********************************
     Karla S. Fant
     CS161 Programming Assignment #0
     Purpose of this program is to convert
     inches entered in by the user into
     millimeters and display the results
     ********************************  */
int main() {
```

Convert inches to millimeters

```
//Define variables
float inches;  //to save # inches
float mm;      //to save the result

//Step #1, welcome the user
cout <<“Welcome! We will be converting”
    <<“ inches to mm today” <<endl;
```

(A different way to do this...)

```
//Define variables
float inches,  //to save # inches
      mm;      //to save the result

//Step #1, welcome the user
cout <<"Welcome! We will be converting";
cout <<" inches to mm today" <<endl;
```

(NOTE: `endl` is `end` followed by a letter `l`)

Convert inches to millimeters

```
//Step #2, Get the input (prompt, read)
cout <<“Please enter the number of inches”
     <<“ that you wish to convert:  “;

cin >> inches;          //read the # inches
cin.get();              //remove the newline

//echo what was entered
cout <<“You entered: “ <<inches <<“in”
       <<endl;
```

Convert inches to millimeters

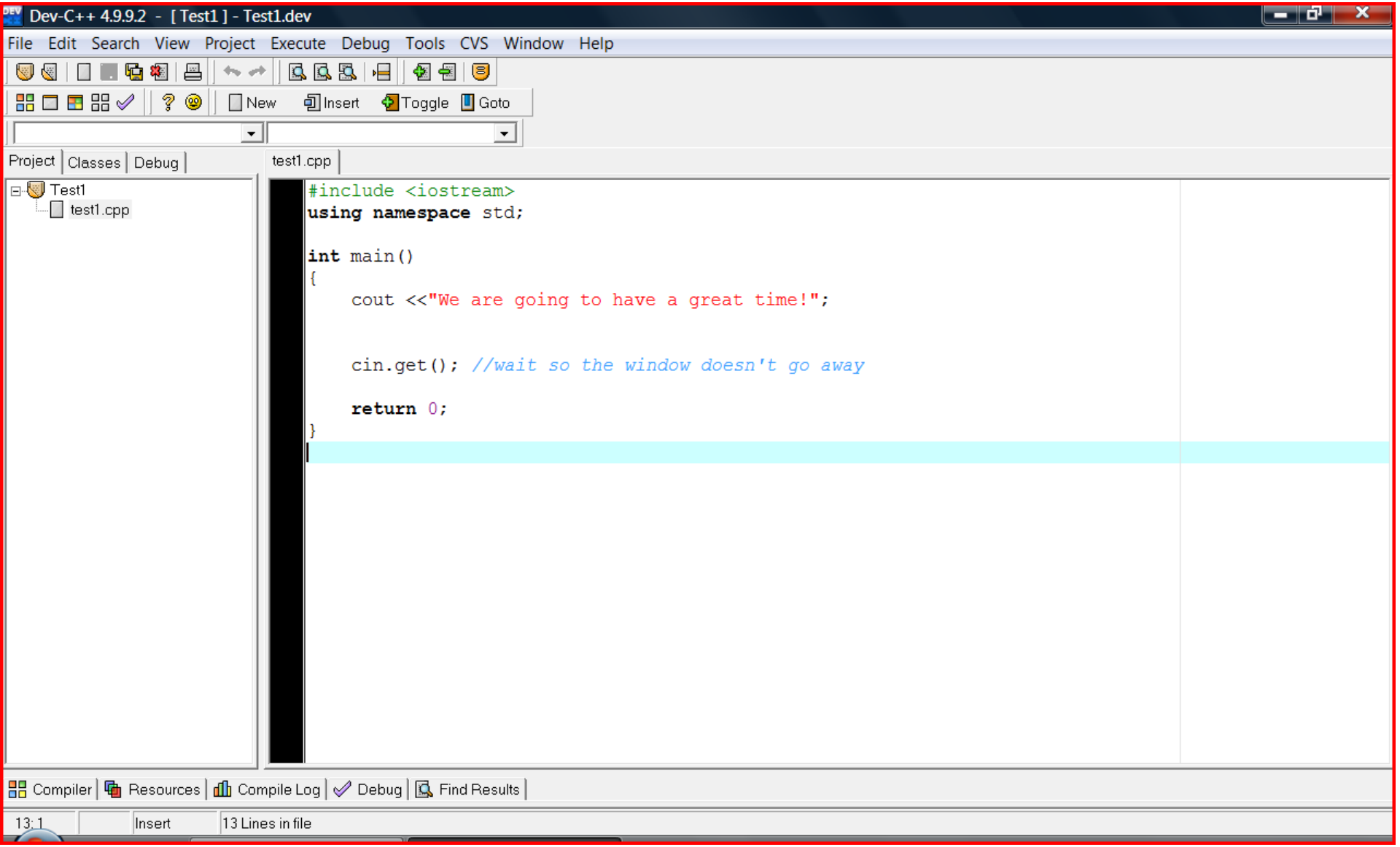
```
//Step #3 Convert inches to millimeters
mm = 25.4 * inches;

//Step #4 Display the results
cout <<inches <<“in converts to “
     <<mm <<“mm” <<endl;

//Step #5 Sign off message
cout <<“Thank you for using CONVERT”
     <<endl;
return 0;
}
```

Preparing to Write Programs:

- The following slides show some examples of Dev-C++ windows
- The first displays just a text window
- The second has both text and graphics windows

- Dev-C++ is for PCs
- Your other choice is to use Unix

- With Dev-C++ you have to be very careful not to leave newlines in the input buffer. It will be a good habit for everyone to get into….removing newlines!

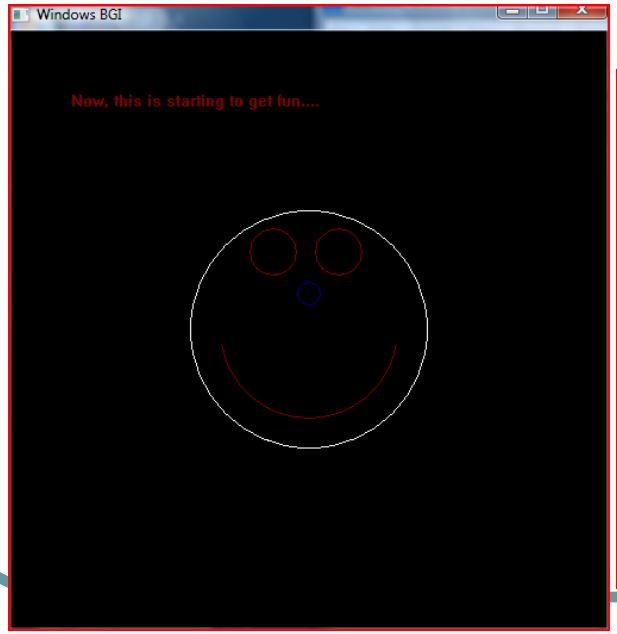

Dev-C++ 4.9.9.2 - [Test1] - Test1.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Classes Debug
Test1
test1.cpp
test1.cpp
#include <iostream>
using namespace std;
int main()
{
cout <<"We are going to have a great time!";
cin.get(); //wait so the window doesn't go away
return 0;
}
Compiler Resources Compile Log Debug Find Results
13: 1
Insert
13 Lines in file

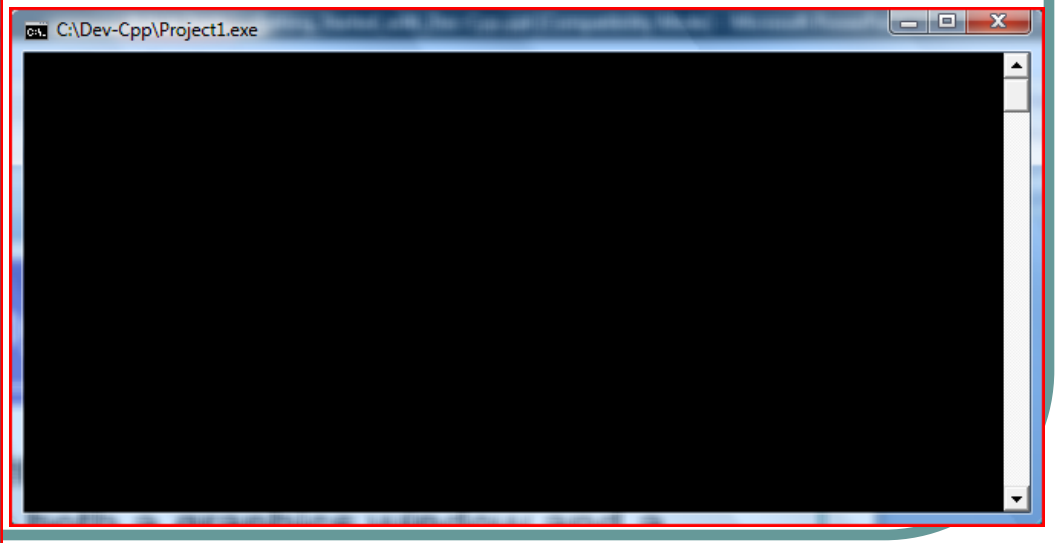
When we execute the program…

- A program like this will have one “window” pop up when we run the program. We call this the console window:

When we do Graphics too…

- When we do graphics we will have both a graphics window *and a* console window:





What's up with newlines???

- With Dev-C++ you have to be very careful not to leave newlines in the input buffer. It will be a good habit for everyone to get into….removing newlines!
- Let me explain…
- With Dev-C++, they want the "window" that is open showing your computer program running to stay active (and visible) until you hit that final enter key (also called a newline).

- Sounds like a good plan.
- You get to see the screen with the result of your program until you are done…by hitting enter! Yes!

Newlines…when they happen…

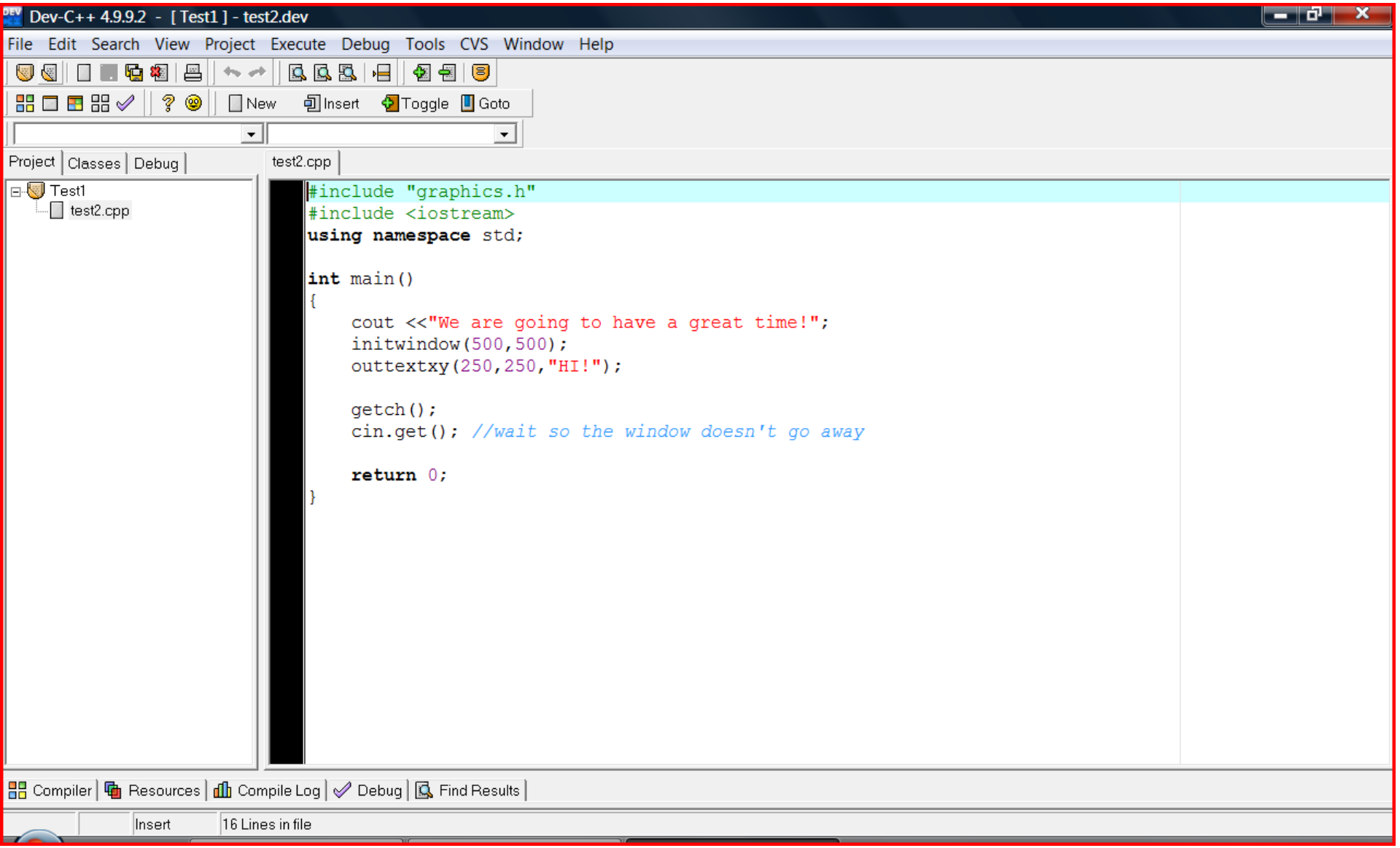
- But, the problem comes in since we are working in what is called “batch” mode versus “event driven” mode for input
- Input means that the program is going to receive information from the user…something will be typed in at the keyboard for example
  - In batch mode, nothing actually goes into the system until the user hits enter (i.e., a newline causes the information that has been typed in the “input buffer” to become available for your program to use
  - So, after we are prompted for input, the user types something in, then they have to hit enter to get that data to be available for the computer.

Newlines…there is a catch!

- So what happens to that enter?
- It becomes a newline that sits in the input buffer waiting to be used (read) by your program
- If your program doesn't use it…and the program ends – guess what?
  - The system will see that newline and think that you are done with the window that is up on the screen
  - So FLASH…you won't see the results of your program running
  - Is it a bug? Nope.
  - What can I do? … Get rid of the newlines each and every time you read something into your program!!!!

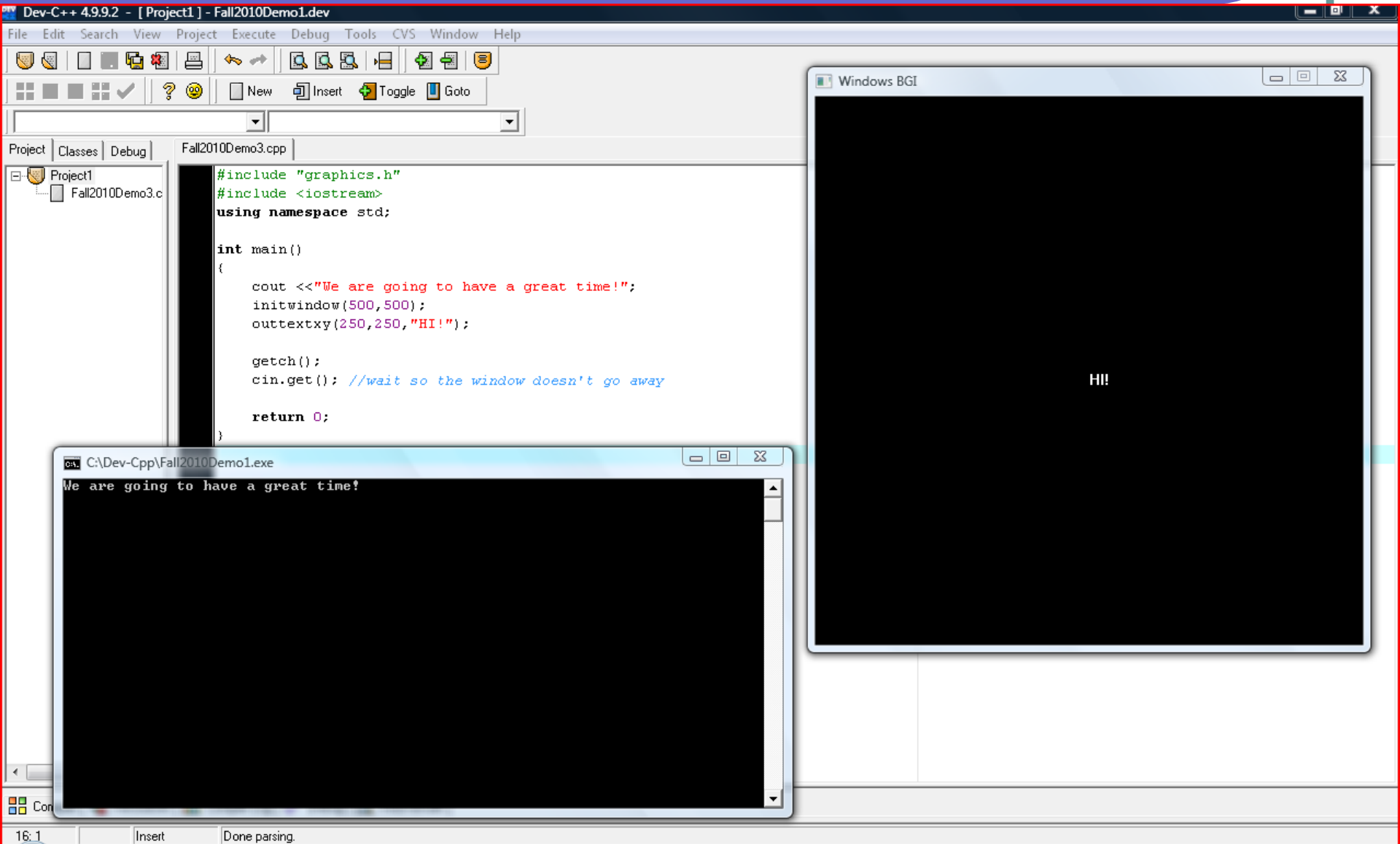
Now…add some graphics

- With Dev C++ we can do regular C++ or we can add a graphics window and draw as well
- Again, graphics is extra and not required in this course
- To start you have to open a graphics window and decide what the size is:
  - This window is from 0,0 (upper left) to 500,500 (lower right)
  - You can make this bigger…we will see this in class
- To output text we use either:
  - outtextxy(x,y, “text to display”);
  - Or, outtext(“text at the current position”);


Dev-C++ 4.9.9.2 - [Test1] - test2.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Classes Debug
test2.cpp
Test1
test2.cpp
#include "graphics.h"
#include <iostream>
using namespace std;
int main()
{
cout <<"We are going to have a great time!";
initwindow(500,500);
outtextxy(250,250,"HI!");
getch();
cin.get(); //wait so the window doesn't go away
return 0;
}
Compiler Resources Compile Log Debug Find Results
Insert
16 Lines in file

```
#include "graphics.h"
#include <iostream>
using namespace std;

int main()
{
    cout <<"We are going to have a great time!";
    initwindow(500,500);
    outtextxy(250,250,"HI!");

    getch();
    cin.get(); //wait so the window doesn't go away

    return 0;
}
```

When we execute the program…



Now…with comments…

```
//This goes to the text window (dialog box)
cout <<"We are going to have a great time!";

 //This is the size (max x, max y) of the graphics window:
 initwindow(500,500);

 //cout does not go to the graphics window…outtext and
 //outtextxy do. 0,0 (origin) is at the upper left corner!
 outtextxy(250,250,"HI!");

 getch();    //this is what we need to do to wait so the graphics
             //window does not go away….hit any key

 cin.get(); //wait so the text (dialog) window doesn't go away
 return 0;
}
```

Program Style

- The Style of your program is important because by doing it cleanly, you can create programs that are easier to read and easier tocorrect
- **Style includes...**

  ...indentation

  ...grouping like elements

  ...using blank lines

  ...variables and program names

Now…think about “style”

- The following is the same program as before but with comments explaining what is happening!

- Things to know about this program:
  - You have to use the #include <iostream>
  - And, using namespace std;
  - In order to do input from the keyboard and output to the screen

- cout is pronounced “see out”
- cin is pronounced “see in”

Poor Program Style

```
#include <iostream>
using namespace std;
int main() { float celsius; float fahrenheit; cout <<"Please enter”
  <<“ temperature in Celsius: " <<endl; cin >>celsius; cin.get();
    fahrenheit = (celsius * 9.0/5.0) +32.0;cout <<celsius;cout <<"
    Celsius  =  " <<fahrenheit;cout <<" Fahrenheit"; cout<<endl;
    return 0;}
```

Better Program Style

```
#include <iostream>
using namespace std;
//This program converts temperatures……
int main()
{
        float celsius;    //temp in celsius
        float fahr;       //temp in Fahrs

        //Read in the temperature in celsius
        cout << "Enter temp in Celsius: “ << endl;
        cin >> celsius;           cin.get();
```

Better Program Style

```
        //Convert celsius to fahrenheits
        fahr = (celsius * 9.0 / 5.0) + 32.0;

        //Print the results
        cout << celsius << " Celsius  =  "  << fahr;
        cout << " Fahrenheit" << endl;

        cout <<“Please hit ENTER when finished:”;
        cin.get();

        return 0;
}
```

Compare the next two slides….

- Look at the next two slides and come up with your comments on the benefits and drawbacks of the style used

- **Comment on each of these areas:**

  ...understandable?

  ...readable?

  ...are like areas grouped together?

  ...what could be better?

```
#include <iostream>
using namespace std;

int main()
{
    cout <<"We are going to have a great time!";


    cin.get(); //wait so the window doesn't go away

    return 0;
}
```

```
//Test Program #1
//Written by Karla Fant for CS161

//The purpose of this program is to show how to display
//a welcome message to the user

//These first two lines make sure that we can perform Input and Output
#include <iostream>
using namespace std;

//This is where the heart of the program is…you will always have the following line:
int main()
{
   //"see out" is how we can output a message to the screen
  // The << is called the "insertion" operator; anything inside the " " is displayed
   cout <<"We are going to have a great time!";

   //On PC's you need to wait for the user to hit enter otherwise the screen just flashes
   //and you won't see anything at all! cin.get() will wait for the user to hit Enter….
   cin.get(); //wait so the window doesn't go away

   return 0;    //return "success" back to the operating system!
}
```

Again …Compare another

- Again, look at the next two slides and come up with your comments on the benefits and drawbacks of the style used
- **Comment on each of these areas:**

  ...are there ethical implications of a poorly styled program?

  …does the program tell the reader what is going to happen?

  …do we have to read English or C++ to figure it out?

  …what would happen if the C++ was more complex? Yikes

```
 #include <iostream>
using namespace std;
// This is a simple program
// Written by: Beth Miller

int main()
{
        cout << "We are going to have a great time!!!!!!!!!!!!!!!!!!";
        cout << endl <<endl;
        cout <<"WOWWWWWWWWWWWWWWWWWWWW"
             <<endl;
        cout << endl <<endl <<endl;
        cout << "HIT ENTER TO END THIS SESSION";
        cin.get();
        return 0;
}
```

Same example w/ Poor Style

```
#include <iostream>
using namespace std;
int main(){cout<<"We are going to have a great time!!!!!!!!!!!!!!!!!!";
cout<<endl<<endl;cout<<"WOWWWWWWWWWWWWWWWWWWWWWWW"<<endl;cout<<endl<<endl <
cout<<"HIT ENTER TO END THIS SESSION";cin.get();return 0;
}
```