C & C++
LAB ASSIGNMENT

Counting Change

Dan McElroy
June 2018

Hello programmers. Look at all the change I have. I want you to write a program that is going to input the number of nickels, and dimes , and quarters and figure out the total. The best part is that you get to write this program all on your own. But you get to use the paycheck program as a reference. You can use the Mac or the PC. You can even use the online compiler.

List of Topics

- Project definition
- Develop an algorithm
- Review the paycheck project
- Develop some test data and test the program
- Document the project (lab report)

The video is divided into several sections to cover the Coin Counting project. The video discusses:
Project definition
How to develop an algorithm
We will review the paycheck project
Develop some test data and test the program
AND Document the project (the lab report)

Before You Even Start

I suggest that you create a new folder on your computer or storage device for the CountingChange project.

You can call your project:

CountingChange, TotalOfCoins, CoinCounter

You can choose any name you wish, but make sure you do not use any spaces in your project name.

I suggest that you create a new folder on your computer or storage device for the CountingChange project.
You can call your project:
CountingChange, TotalOfCoins, CoinCounter
You can choose any name you wish, but make sure you do not use any spaces in your project name.

1 - Project Definition

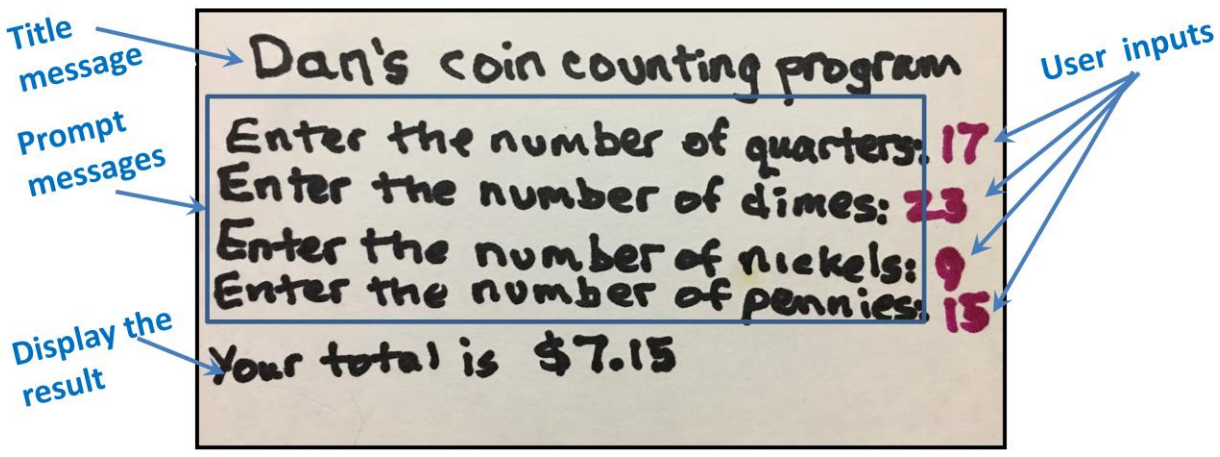
Here is the project definition.

Write a program that counts change. The program should ask for the number of quarters, the number of dimes, the number of nickels and the number of pennies. Then the program should tell the user how much money there is, expressed in dollars.

You must use constants for the value of each coin type, and display a title at the top of the screen when the program first runs.

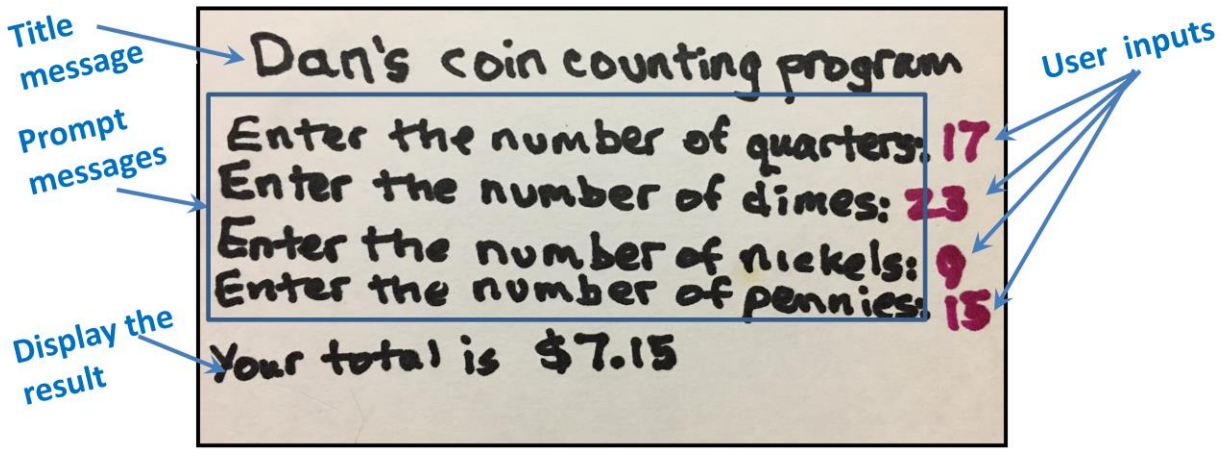
Here is the project definition.
Write a program that counts change. The program should ask for the number of quarters, the number of dimes, the number of nickels and the number of pennies. Then the program should tell the user how much money there is, expressed in dollars.
You must use constants for the value of each coin type, and display a title at the top of the screen when the program first runs.

1 – Outline How the Screen Should Look



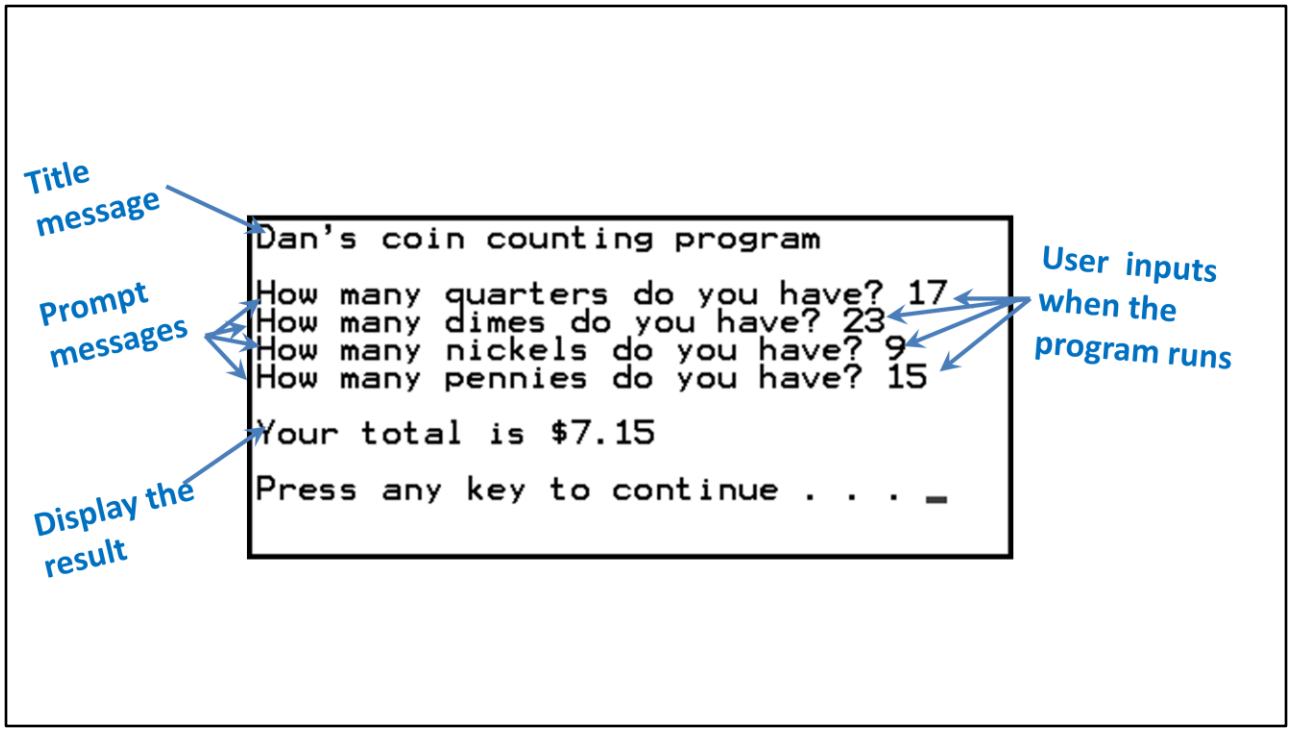
The first thing you should do when you are given a programming task is to figure out what the output should look like. Draw it out on paper. As the project progresses, don't worry if the actual program output does not look exactly like your original plan. Plans change.

1 – Outline How the Screen Should Look



Put a title at the top of the output display that identifies your name and the program, followed by a blank line. Your program does not need to display anything in colors. I am only using red on my program outline to highlight the user inputs. These values are to be read from the keyboard when the program runs, instead of being displayed as part of the prompt messages.

When all of the inputs are read, display another blank line and then the total of all the coins in dollars.



Let's see how the program looks when it is run. There is the title line at the top of the display. I am going to input 17 quarters, 23 dimes, 9 nickels and 15 pennies. The program shows me that I have $7.15.

Your program should look something like this when it runs. The text of your program does not need to look exactly like mine. Instead of a prompt message that looks like, "How many quarters do you have? " – your program could say, "Number of quarters? ", or "Input the number of quarters: "

The output could even say, "You have …" and then how much money instead of "Your total is …" Feel free to customize the text that is displayed, but be polite.

3 – CountingChange – Develop Algorithm

INPUT	PROCESS	OUTPUT
quarters **dimes** **nickels** **pennies**	Display Title message Read **quarters** from keyboard Read **dimes** from keyboard Read **nickels** from keyboard Read **pennies** from keyboard **total = quarters*0.25 + dimes*0.10 + nickels*0.05 + pennies*0.01** Display **total**	**Title message** **Total message**

In a hierarchy, input, process, output chart, also called a HIPO chart we want to define the inputs and the output.

Once we know the inputs and the outputs, we then need to decide how to get from input all the way to the output. We're going to read the number of quarters, dimes, nickels and pennies. Compute the total of all the coins and then display that value on the screen.

Variables, Constants and Data Types

This program is only using numeric data for the number of coins, their values and the total value of all the coins.

We need to choose which of the numeric data types to use in the program. The **int** data type can only be used for whole numbers. The **int** data type is a great choice for the number of each coin type. We can't input half a quarter, or even a penny. But we will need to use the double data type for the value of each coin and the total value of all the coins expressed in dollars.

This program is only using numeric data for the number of coins, their values and the total value of all the coins. We need to choose which of the numeric data types to use in the program. The **int** data type can only be used for whole numbers. The **int** data type is a great choice for the number of each coin type. We can't input half a quarter, or even a penny. But we will need to use the double data type for the value of each coin and the total value of all the coins expressed in dollars.

Variables, Constants and Data Types

If we define an **int** variables for quarters, dimes, nickels and pennies, and define a **double** variable for total, then the total can be computed as:

total = quarters*0.25 + dimes*0.10 + nickels*0.05 + pennies*0.01;

If we define **int** variables for quarters, dimes, nickels and pennies, then the total can be computed as:

total = quarters*0.25 + dimes*0.10 + nickels*0.05 + pennies*0.01;

Variables, Constants and Data Types

Part of the project definition states that a constant must be used for the coin values. In that case, we need to define a constant for the value of a quarter, etc.

```
total = quarters*QUARTER_VALUE + dimes*DIME_VALUE
      + nickels*NICKEL_VALUE + pennies*PENNY_VALUE;
```

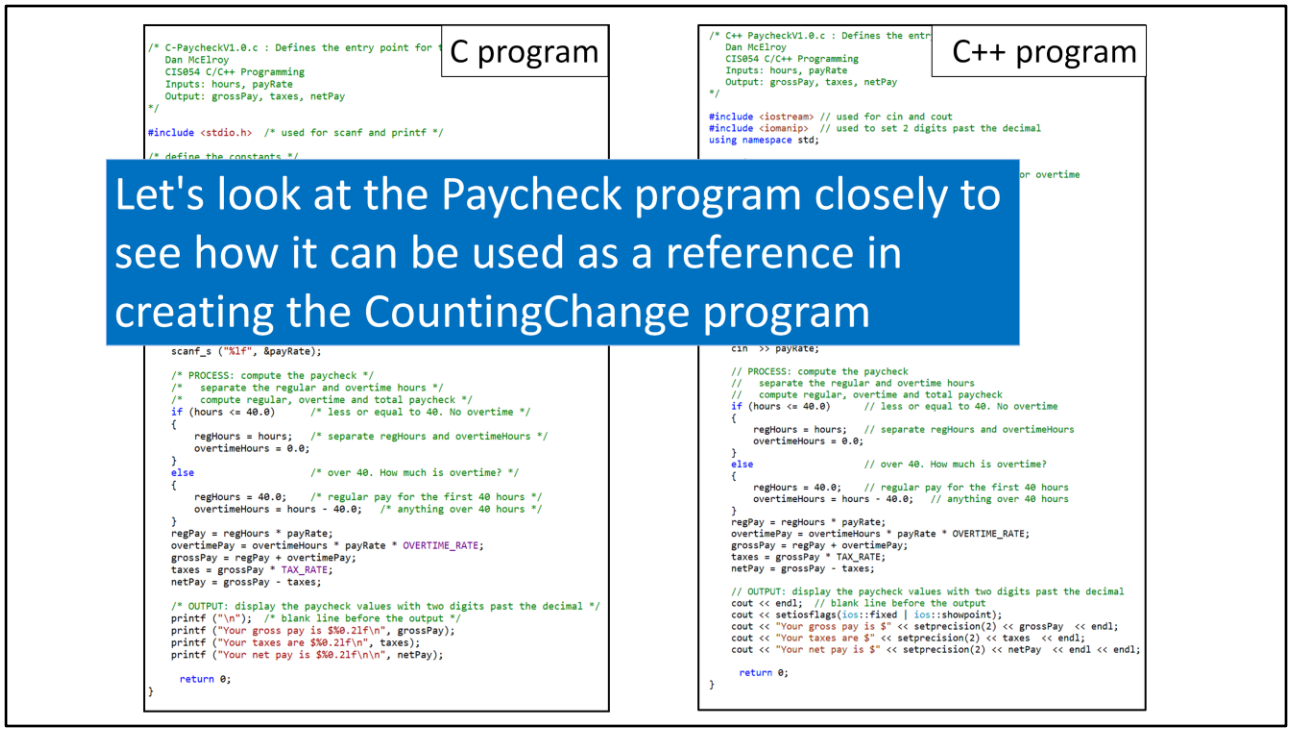
NOTE: Since the statement does not end until the semicolon ; is reached, the statement can cover more than one line.

Let's review how to define constants in C and C++.

Part of the project definition states that a constant must be used for the coin values. In that case, we need to define a constant for the value of a quarter, etc.

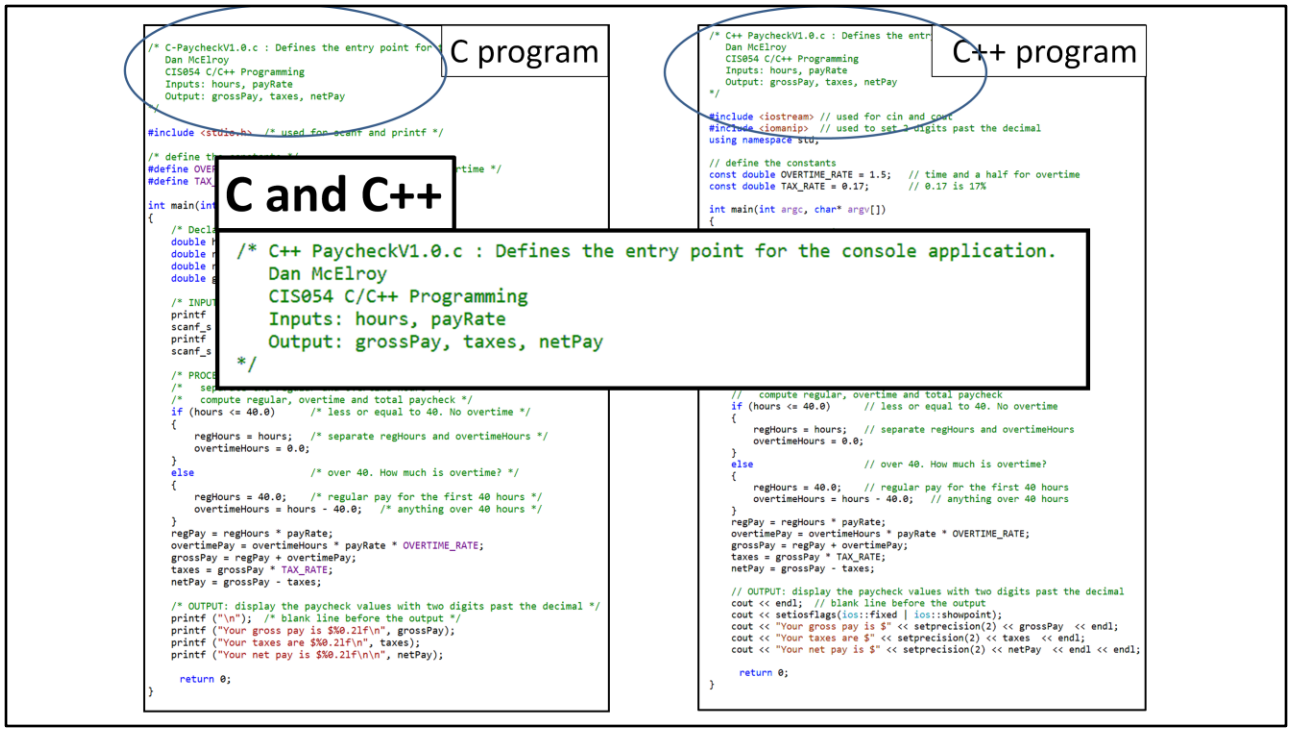
```
total = quarters*QUARTER_VALUE +
dimes*DIME_VALUE
      etc.
```

NOTE: Since the statement does not end until the semicolon ; is reached, the statement can cover more than one line.



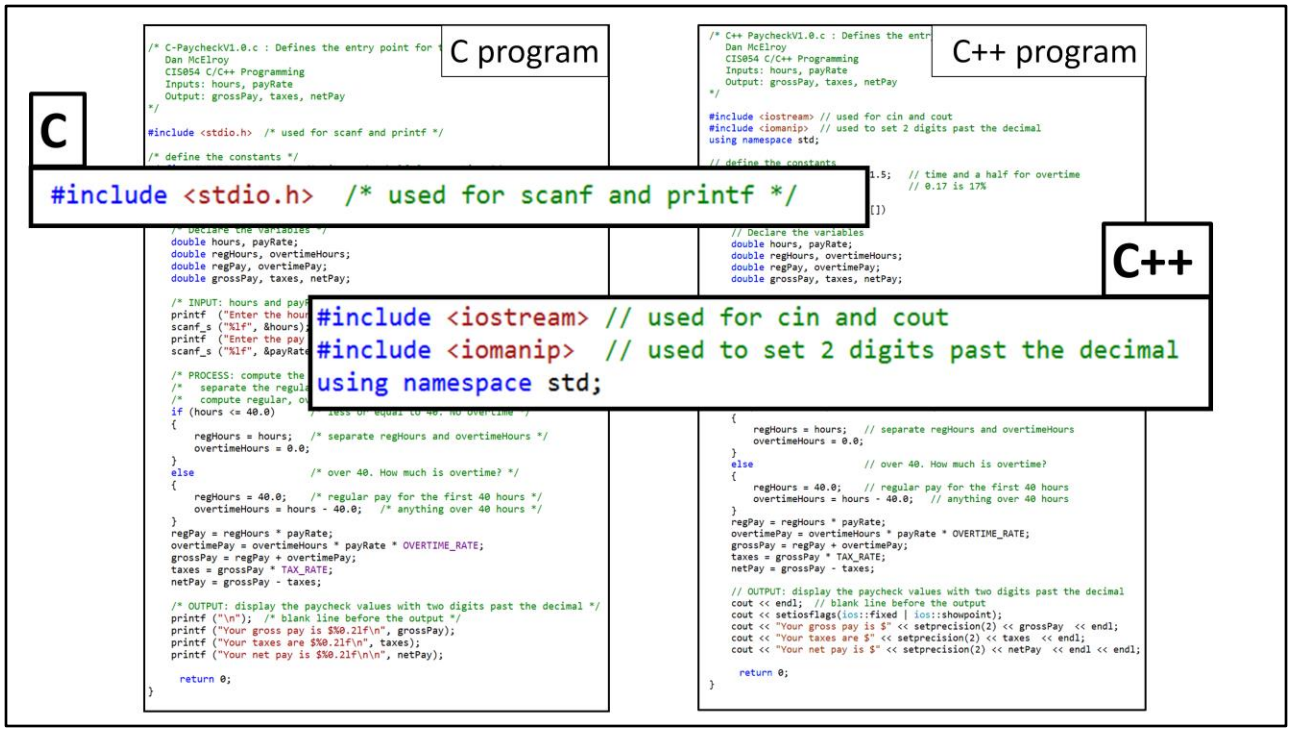
I am now going to review the Paycheck program. You can use this program as a reference when you develop the CountingChange program. Choose to do this project either using the C language or C++. You can get the source code for the paycheck program in either C or C++ from the class website.

If the text is too small on this slide, don't worry about reading it yet. I will zoom in for different parts during the discussion.



The title block at the top of the program file uses the C-language style comments, starting with a /* and ending with a */

The comment block identifies what is in the file as well as the inputs and outputs. Some companies want the programmer's name in the title block, others don't. I want your name in the title block and I will look for it when I grade your projects. Although not shown here, another couple of things that are usually placed in the title block are the version number and the date. This can help determine the latest version of the code when there are multiple files that are not the same.



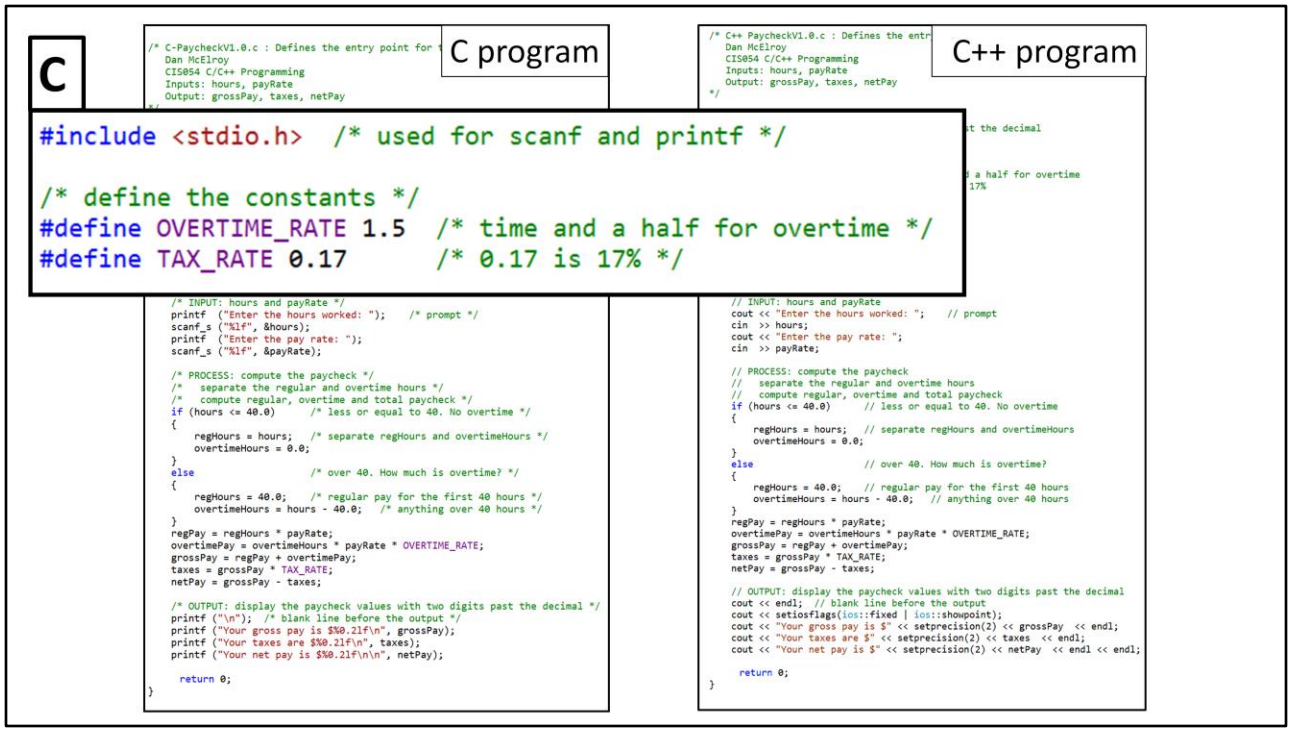
The #include statements brings in additional code that helps the compiler build your program. Look carefully, there is no semicolon ; at the end of the #include statement. Some other programming languages use the word **Import** or **Imports** instead of #include.

The C-program uses the **#include <stdio.h>** command to provide access for the console input and output functions **scanf** and **printf**. The keyboard and display are the console.

The C++ version of the program uses **#include <iostream>** to give access to the console through **cout** and **cin**.
The **#include <iomanip>** gives access to the **setiosflags** and **setprecision** at the end of the program to make a display with two digits past the decimal.
using namespace std; makes the coding much easier. Otherwise we would need to type **std::** in front of each **cout**, **cin** and some other commands. There is a semicolon ; at the end of using namespace std;

Look at the C++ style comments // C++ style comments start with the // and end

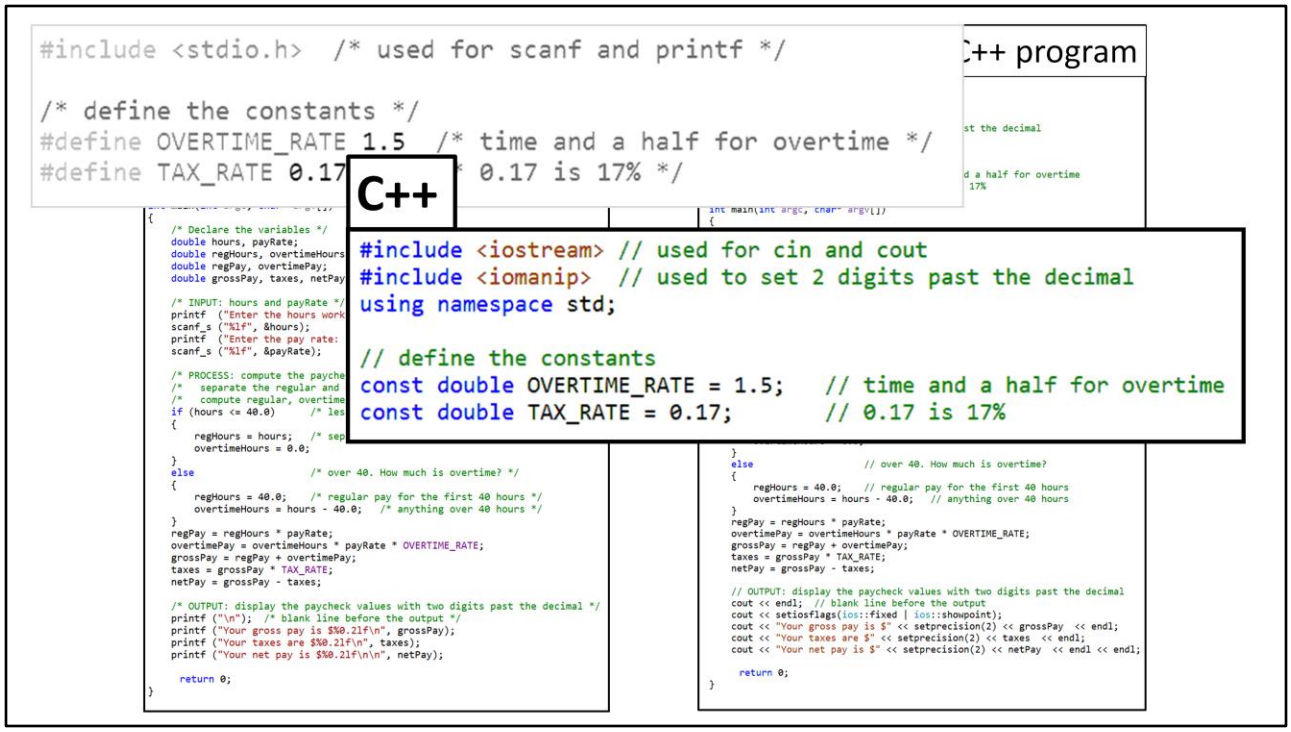
at the end of the line. Although this style of comment was introduced with C++, most modern C compilers accept both types of comments.



The #define statement is a pre-processor directive that can be used in C to define constants. It causes a text replacement while the program is being compiled. This would be similar to the Find-and-Replace feature in a word processor. Anywhere that the characters OVERTIME_RATE appear in the program, they would be replaced with the characters 1.5

Because the #define is a text replacement, it is important that neither the equal sign = nor the semicolon character ; are used in the #define statement. Otherwise those extra characters would also be added into the program before it is compiled.

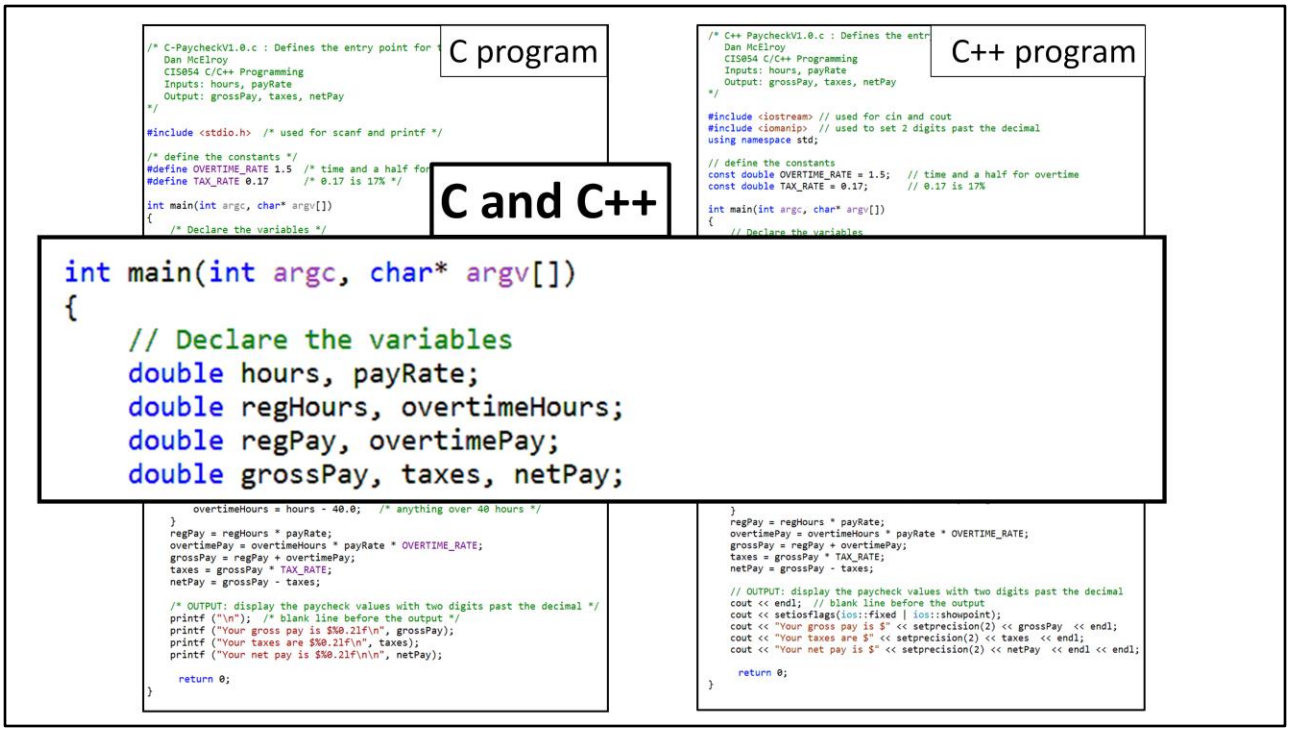
Although not required, it is common practice to use all UPPER_CASE characters and the underscore _ when defining constants. This helps make them easier to recognize when looking at the code later.



Compare the difference in how C and C++ declare a constant.

Instead of a text replacement, we are actually creating data with its associated data type, and using the equal-sign = to assign it a value. The semicolon is used at the end of the assignment statement.

By using **const double OVERTIME_RATE = 1.5;** we now have the value 1.5 stored in memory.



The Paycheck program is using the **double** data type for all its variables. This is the place in the CountingChange program where you need to define the **int** variables for quarters, dimes, nickels and pennies. You also need to define a **double** for the total.

Prompt and Input a Number

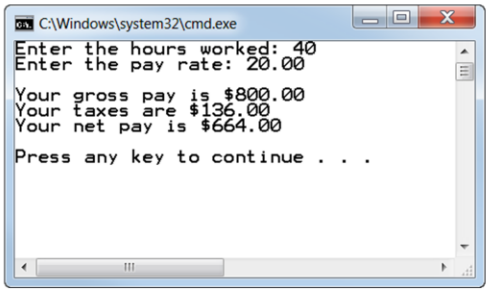
```
/* C-PaycheckV1.0.c : Defines the entry point for the console applicati
   Dan McElroy
   CIS054 C/C++ Programming
   Inputs: hours, payRate
   Output: grossPay, taxes, netPay
*/

#include <stdio.h>  /* used for scanf and printf */

/* define the constants */
#define OVERTIME_RATE 1.5  /* time and a half for overtime */
#define TAX_RATE 0.17      /* 0.17 is 17% */

int main(int argc, char* argv[])
{
    /* Declare the variables */
    double hours, payRate;
    double regHours, overtimeHours;
    double regPay, overtimePay;
    double grossPay, taxes, netPay;

    /* INPUT: hours and payRate */
    printf  ("Enter the hours worked: ");    /* prompt */
    /* your compiler may need scanf instead of scanf_s */
    scanf_s ("%lf", &hours);
    printf  ("Enter the pay rate: ");
    scanf_s ("%lf", &payRate);

    /* PROCESS: compute the paycheck */
    /*   separate the regular and overtime hours */
    /*   compute regular, overtime and total paycheck */
    if (hours <= 40.0)      /* less or equal to 40. No overtime */
    {
        regHours = hours;   /* separate regHours and overtimeHours */
        overtimeHours = 0.0;
    }
    else                    /* over 40. How much is overtime? */
    {
        regHours = 40.0;    /* regular pay for the first 40 hours */
        overtimeHours = hours - 40.0;   /* anything over 40 hours */
    }
    regPay = regHours * payRate;
    overtimePay = overtimeHours * payRate * OVERTIME_RATE;
    grossPay = regPay + overtimePay;
    taxes = grossPay * TAX_RATE;
    netPay = grossPay - taxes;

    /* OUTPUT: display the paycheck values with two digits past the decimal */
    printf ("\n");  /* blank line before the output */
    printf ("Your gross pay is $%0.2lf\n", grossPay);
    printf ("Your taxes are $%0.2lf\n", taxes);
    printf ("Your net pay is $%0.2lf\n\n", netPay);

     return 0;
}
```

C

```
/* INPUT: hours and payRate */
printf  ("Enter the hours worked: ");    /* prompt */
/* your compiler may need scanf instead of scanf_s */
scanf_s ("%lf", &hours);
```

C++

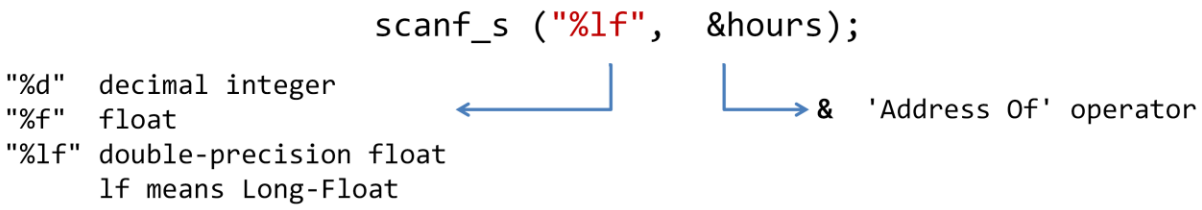
```
// INPUT: hours and payRate
cout << "Enter the hours worked: ";    // prompt
cin  >> hours;
```



C-programs use the **printf** statement to output to the console screen and **scanf** or **scanf_s** to read from the console keyboard.

C++ programs use **cout** to output to the console screen and **cin** to read from the console keyboard.

scanf Format String



The first parameter in a call to scanf is the **format string**. It is surrounded by double-quotes. The format string tells scanf what type of data is expected to be typed at the console. The Paycheck program is using **"%lf"** because it is expecting data that may have digits past the decimal, and the data is going to be stored in a variable whose data type is **double**, also known as long-float.

Assuming that **quarters, dimes, nickels** and **pennies** are declared as type **int** then use **%d** inside the scanf format string instead of **%lf**.

```c
/* C-PaycheckV1.0.c : Defines the entry point for the console application.
   Dan McElroy
   CIS054 C/C++ Programming
   Inputs: hours, payRate
   Output: grossPay, taxes, netPay
*/

#include <stdio.h>  /* used for scanf and printf */

/* define the constants */
#define OVERTIME_RATE 1.5  /* time and a half for overtime */
#define TAX_RATE 0.17      /* 0.17 is 17% */

int main(int argc, char* argv[])
{
    /* Declare the variables */
    double hours, payRate;
    double regHours, overtimeHours;
    double regPay, overtimePay;
    double grossPay, taxes, netPay;

    /* INPUT: hours and payRate */
    printf  ("Enter the hours worked: ");    /* prompt */
    /* your compiler may need scanf instead of scanf_s */
    scanf_s ("%lf", &hours);
    printf  ("Enter the pay rate: ");
    scanf_s ("%lf", &payRate);

    /* PROCESS: compute the paycheck */
    /*   separate the regular and overtime hours */
    /*   compute regular, overtime and total paycheck */
    if (hours <= 40.0)      /* less or equal to 40. No overtime */
    {
        regHours = hours;   /* separate regHours and overtimeHours */
        overtimeHours = 0.0;
    }
    else                    /* over 40. How much is overtime? */
    {
        regHours = 40.0;    /* regular pay for the first 40 hours */
        overtimeHours = hours - 40.0;   /* anything over 40 hours */
    }
    regPay = regHours * payRate;
    overtimePay = overtimeHours * payRate * OVERTIME_RATE;
    grossPay = regPay + overtimePay;
    taxes = grossPay * TAX_RATE;
    netPay = grossPay - taxes;

    /* OUTPUT: display the paycheck values with two digits past the decimal */
    printf ("\n");  /* blank line before the output */
    printf ("Your gross pay is $%0.2lf\n", grossPay);
    printf ("Your taxes are $%0.2lf\n", taxes);
    printf ("Your net pay is $%0.2lf\n\n", netPay);

     return 0;
}
```

Compute

netPay, taxes and grossPay

```c
regPay = regHours * payRate;
overtimePay = overtimeHours * payRate * OVERTIME_RATE;
grossPay = regPay + overtimePay;
taxes = grossPay * TAX_RATE;
netPay = grossPay - taxes;
```

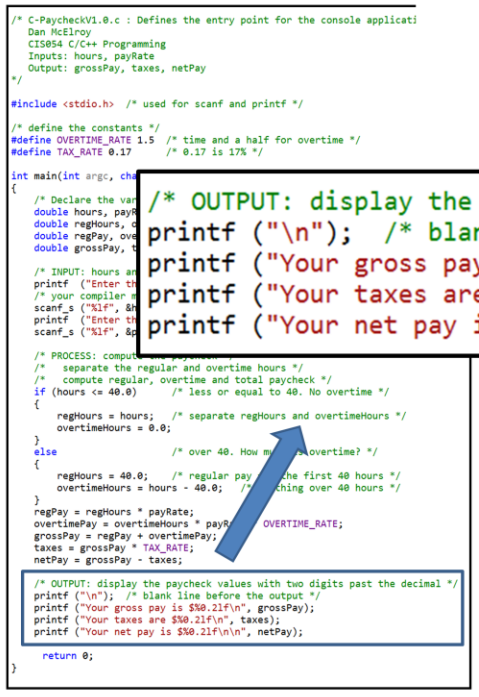
The computations for the total value of all the coins is much easier than the computation in the Paycheck program. All you need to do is add up the

number of each coin times its value and save it in the total.

OUTPUT using C

grossPay, taxes, netPay

```
/* OUTPUT: display the paycheck values with two digits past the decimal */
printf ("\n");  /* blank line before the output */
printf ("Your gross pay is $%0.2lf\n", grossPay);
printf ("Your taxes are $%0.2lf\n", taxes);
printf ("Your net pay is $%0.2lf\n\n", netPay);
```

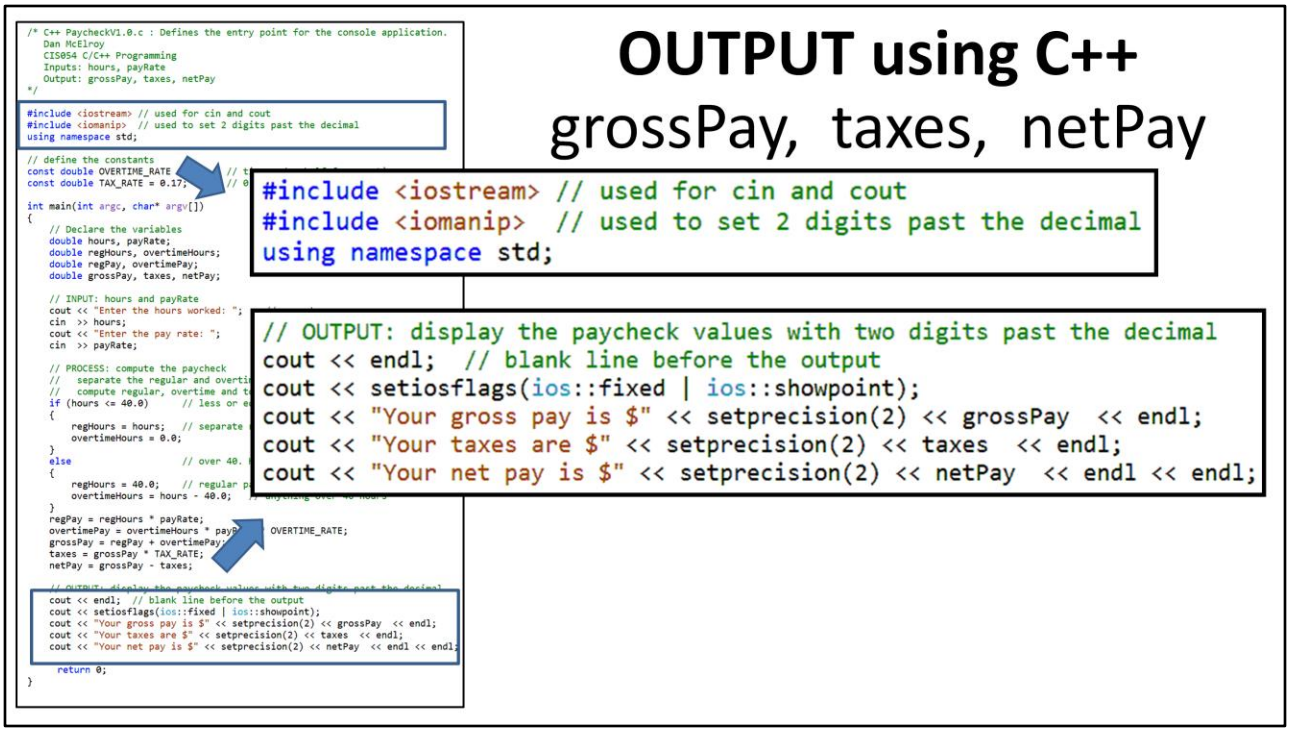



```
("Your pay is $%0.2lf\n", pay);
```

Review the C and C++ Paycheck programs on how they display the paycheck with two decimal places. You only need a blank line before displaying the total. Two new line commands \n\n create a blank line before the C program closes.

OUTPUT using C++

grossPay, taxes, netPay



If writing a C++ program, make sure you type the last letter of **endl** as a lower-case L because endl means end of line.

Using **endl** twice moves the cursor down two lines.

Test the Program

Here are the values for the number of quarters, dimes, nickels and pennies that I have chosen. I want you to choose different values.

quarters	dimes	nickels	pennies	total
17	23	9	15	$7.15
17	23	0	15	$6.70
17	23		15	???
17	23	nine	15	???

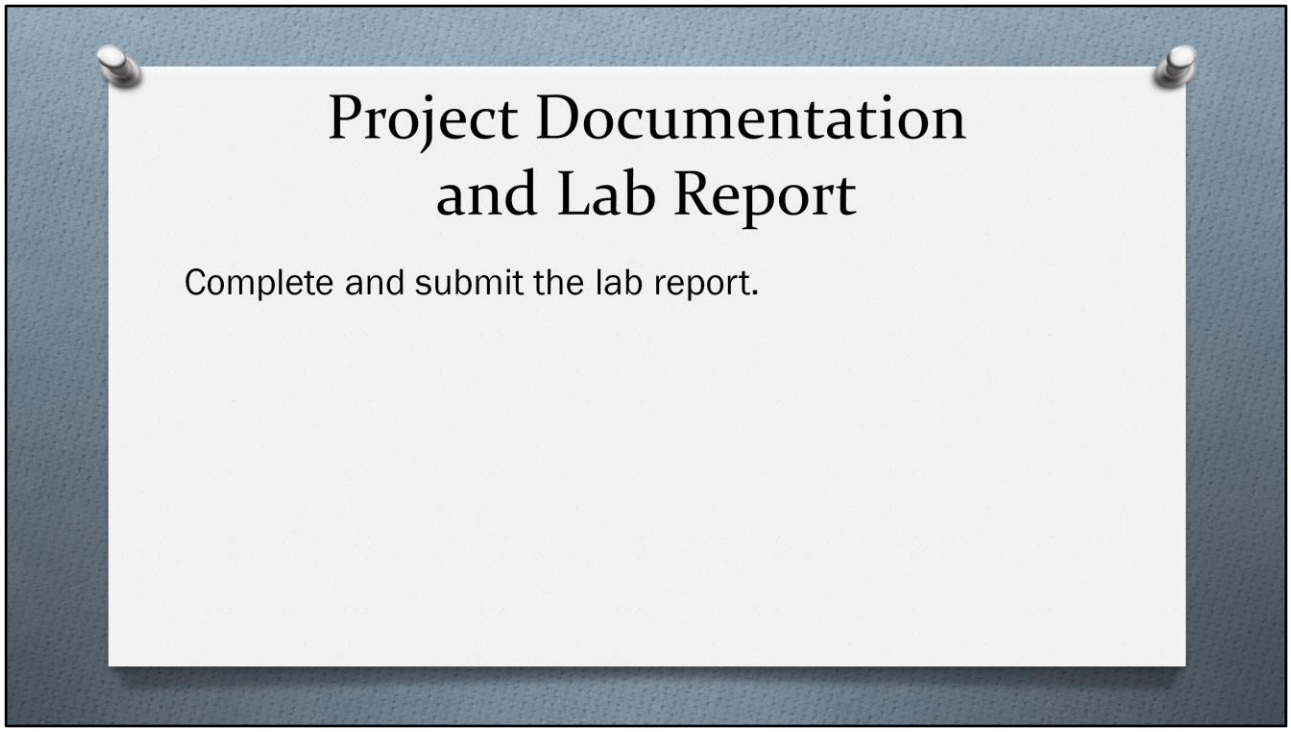
Here are the values for the number of quarters, dimes, nickels and pennies that I have chosen. I want you to choose different values. The first two rows should work ok. I want you to test the program when at least one of the values is entered as zero.

Test the Program

Although we have not covered how to process unexpected inputs yet, I want you to test the program and find out what happens if either nothing is entered or letters are entered for nickels.

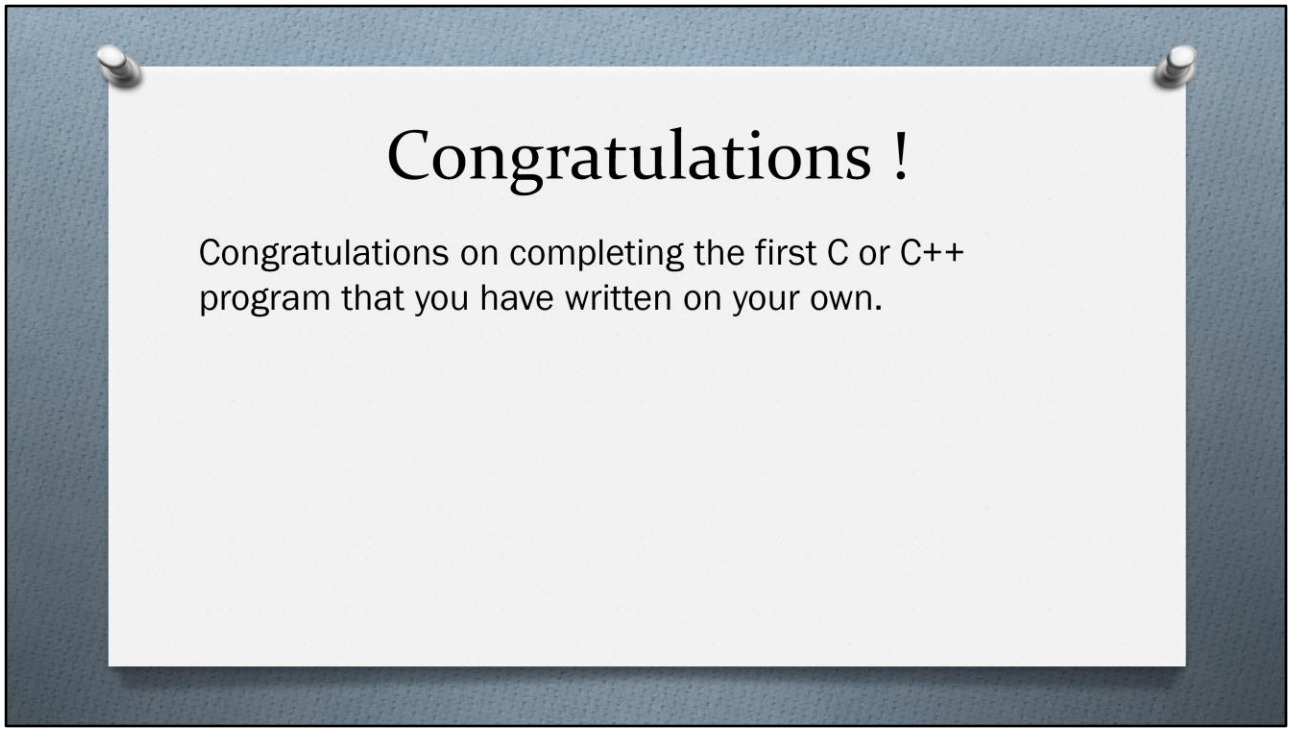
quarters	dimes	nickels	pennies	total
17	23	9	15	$7.15
17	23	0	15	$6.70
17	23		15	???
17	23	nine	15	???

Although we have not covered how to process unexpected inputs yet, I want you to test the program and find out what happens if either nothing is entered or letters are entered for nickels.



Project Documentation and Lab Report

Complete and submit the lab report.



Congratulations on completing the first C or C++ program that you have written on your own.

I look forward to seeing you in the next module.

Bye for now.