**Chapter 1 Lab**

**Compiling, Errors, and Testing**

**Lab Objectives**

* Set up a project in Visual Studio.
* Be able to compile and execute a C++ program
* Be able to test a program
* Be able to debug a program with syntax and logic errors.
* Create a program from scratch by translating a pseudocode algorithm
* Be able to document a program

**Introduction**

The example we discussed in class calculates the user’s gross pay. It calculates the gross pay by multiplying the number of hours worked by hourly pay rate. However, it is not always calculated this way. What if you work 45 hours in a week? The hours that you worked over 40 hours are considered overtime. You will need to be paid time and a half for the overtime hours you worked.

In this lab, you are given a program which calculates user’s gross pay with or without overtime. You are to work backwards this time, and use pseudocode to write an algorithm from the Java code. This will give you practice with algorithms while allowing you to explore and understand a little Java code before we begin learning the Java programming language.

You will also need to test out this program to ensure the correctness of the algorithm and code. You will need to develop test data that will represent all possible kinds of data that the user may enter.

You will also be debugging a program. There are several types of errors. In this lab, you will encounter syntax and logic errors.

1. Syntax Errors—errors in the “grammar” of the programming language. These are caught by the compiler and listed out with line number and error found. You will learn how to understand what they tell you with experience. All syntax errors must be corrected before the program will run. If the program runs, this does not mean that it is correct, only that there are no syntax errors. Examples of syntax errors are spelling mistakes in variable names, missing semicolon, unpaired curly braces, etc.
2. Logic Errors—errors in the logic of the algorithm. These errors emphasize the need for a correct algorithm. If the statements are out of order, if there are errors in a formula, or if there are missing steps, the program can still run and give you output, but it may be the wrong output. Since there is no list of errors for logic errors, you may not realize you have errors unless you check your output. It is very important to know what output you expect. You should test your programs with different inputs, and know what output to expect in each case. For example, if your program calculates your pay, you should check three different cases: less than 40 hours, 40 hours, and more than 40 hours. Calculate each case by hand before running your program so that you know what to expect. You may get a correct answer for one case, but not for another case. This will help you figure out where your logic errors are.
3. Run time errors—errors that do not occur until the program is run, and then may only occur with some data. These errors emphasize the need for completely testing your program.

We will bring everything we have learned together by creating a program from an algorithm. Finally, you will document the program by adding comments. Comments are not read by the computer, they are for use by the programmer. They are to help a programmer document what the program does and how it accomplishes it. This is very important when a programmer needs to modify code that is written by another person.

# Task #1 Compile and Execute a Program

1. Open Visual Studio Community 2019.
2. Choose “Create a new project”
3. Create an “Empty Project”. Project name is Task 1, solution name is Lab 1. Choose where you want to create this project.
4. Download the Lab1.zip from Canvas and unzip.
5. Open another window in File Explorer and navigate to the Task 1 folder you have just created.
6. Copy **pay.cpp** into the Task 1 folder.
7. In Visual Studio, right click on the Source Files folder. Add an existing item **pay.cpp**.
8. Use **Ctrl + F5** to start the program without debugging. You should not receive any error messages.
9. When this program is executed, it will ask the user for input. You should calculate several different cases by hand. Since there is a critical point (40 hours) at which the calculation changes, you should test three different cases: the critical point, a number above the critical point, and a number below the critical point. You want to calculate by hand so that you can check the logic of the program. Fill in the chart below with your test cases and the result you get when calculating by hand.
10. Execute the program using your first set of data. Record your result. You will need to execute the program three times to test all your data. Note: you do not need to compile again. Once the program compiles correctly once, it can be executed many times. You only need to compile again if you make changes to the code.

|  |  |  |  |
| --- | --- | --- | --- |
| Hours | Rate | Pay (hand calculated) | Pay (program result) |
| 5 | 10.00 | 50.00 | 50.00 |
| 40 | 10.00 | 400.00 | 400.00 |
| 50 | 10.00 | 550.00 | 550.00 |

**Task #2 Debugging a C++ Program**

1. Right click on Solution in Visual Studio Solution Explorer.
2. Add a new project called Task 2 to your current solution.
3. Open another window in File Explorer and navigate to the Task 2 folder you have just created.
4. Right click on Task 2 in Visual Studio Solution Explorer.
5. Select Set as Start Up Project.
6. Right click on the Source folder under this project. Add an existing item and select **salesTax.cpp**. This file contains a simple C++ program that contains errors.
7. Start without debugging (Ctrl + F5). You should get a listing of syntax errors. Correct all the syntax errors, you may want to recompile after you fix some of the errors.
8. When all syntax errors are corrected, the program should compile. As in the previous exercise, you need to develop some test data. Use the chart below to record your test data and results when calculated by hand.
9. Execute the program using your test data and recording the results. If the output of the program is different from what you calculated, this usually indicates a logic error. Examine the program and correct logic error. Compile the program and execute using the test data again. Repeat until all output matches what is expected.
10. Try entering a word for the price. What happens? What kind of error would this be?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item | Price | Tax | Total(calculated) | Total (output) |
| Toothpaste | 2000 | 110 | 2110 | 4000110 |
| CrappyCar | 2 | 0.11 | 2.11 | 4.11 |

# Task #3 – Create a program from scratch

In this task you will create a new program that calculates gas mileage in miles per gallon. You will use string expressions, assignment statements, input and output statements to communicate with the user.

1. Right click on Solution in Visual Studio Solution Explorer.
2. Add a new project called Task 3 to your current solution.
3. Add a new file to the source folder called **mileage.cpp**.
4. Enter the standard template:

#include <iostream>

using namespace std;

int main ()

{

return 0;

}

1. Translate the algorithm below into C++, putting it in the main above the return 0 but after {. Don’t forget to declare variables before they are used. Each variable must be one word only (no spaces).

*Display a line indicating this program will calculate mileage*

*Display a prompt to user asking for miles driven*

*Read in miles driven*

*Display prompt to user asking for gallons used*

*Read in gallons used*

*Calculate miles per gallon by dividing miles driven by gallons used*

*Display miles per gallon along with appropriate labels*

1. Compile the program and debug, repeating until it compiles successfully.
2. Run the program and test it using the following sets of data and record the results:

|  |  |  |  |
| --- | --- | --- | --- |
| Miles driven | Gallons used | Miles per gallon (hand calculated) | Miles per gallon  (resulting output) |
| 2000 | 100 | 20 | 20 |
| 500 | 25.5 | 19.6078431372 | 19.6078 |
| 241.5 | 10 | 24.15 | 24.15 |
| 100 | 0 | Infinite | inf |

**Task #4 Documenting a C++ Program**

1. Compare the code listings of pay.cpp with mileage.cpp. You will see that pay.cpp has lines which have information about what the program is doing. These lines are called comments and are designated by the // at the beginning of the line.
2. Write a comment line at the top of the program which indicates the purpose of the program.
3. Write a second comment line at the top of the program with your name and today’s date.
4. Add comment lines after each variable declaration, indicating what each variable represents.
5. Add comment lines for each section of the program, indicating what is done in that section.
6. Finally add a comment line indicating the purpose of the calculation.

**Post Lab Questions**

1. What are the 3 types of errors? Give a specific example of 2 different kinds that you had to fix on the salesTax program (task 3).  
     
   Logic Error (errors in calculation with calculating the proper tax)  
   Syntax Error (errors that cause the code not to compile due to improper syntax)  
   Runtime Error (Errors that occur when a program runs infinitly)
2. Give an example of each of the following language elements that you saw in salesTax.
   1. Keywords  
      **double  
      int  
      string**
   2. Programmer-defined names  
      **total  
      TAX\_RATE**
   3. Operators  
      **\* and +**
   4. Punctuation  
      **;  
      <<  
      >>**
3. Explain what happens in the computer when you hit Start.  
     
   A computer reads codes line by line and compiles it into binary code that the machine turns into actual commands.