Object Oriented Software Development

Introduction to Computers and C++ Programming

**Preamble**

In this course, you will learn to write C++ programs to solve different problems.  From time-to-time this task may become frustrating and difficult.  However, with patience and practice, you will gain more experience and things will get easier.  The difficulty in using a computer to solve problems comes from the fact that you need to somehow ask a computer to do things for you.  Imagine, sitting in front of the computer and asking her to add two numbers, “Hey you add 2 to 4 and tell me what the answer is.”  But we do not normally talk directly to a computer, not yet. Instead, we communicate with it through an operating system (OS).

The communication with the OS is done via the program that you write.  Our program is a set of instructions that a computer will follow.  These instructions are not in English but we understand them very well, because they are written in a high-level language.  These instructions will, at some point, get to the machine much differently in machine language.  The machine language is a low-level language.

**How does a computer solve a problem?**

Given the information above, let’s look at an analogy using and architect in a foreign land.

Suppose you are an architect who knows how to design a house and you just got a job in Japan and want to build a house there.  You don't know the Japanese language but you have access to a translator.

So… you write, on a piece of paper, a set of instructions in English and give this to the translator.  The translator will bring you a copy of your instructions in Japanese the next day.  This would happen if the translator could understand everything that you have written.  If there is something that doesn’t make sense to the translator (syntax error), the translator would halt his/her job and would let you know that there was something that must be fixed before he/she could give you a correct final translation.  The possible errors could be caused by a set of syntactically incorrect instructions or could be due to missing syntax.

If you have no error, then you would have a translation in your hand a bit later.  When you look at the translated instructions set, you don't understand it, after all it is written in a different language. But you are not worried about this, there is someone who can read these instructions and complete the house for you.

The next thing you need to do is to find a Japanese architect to build the house.  If you are lucky (for now we assume you are), the translator and the architect may be the same person but you need to get the translation and resubmit it to him/her for execution.

The next thing that should happen is that the Japanese architect will bring all the resources together and will build the house for you.  If all your measurements are correct and all the requests are valid, you will get a good house. Sometimes you may have made a request that is impossible to complete.  An invalid request may not necessary be due to a wrong syntax.  For example, you have asked the builder to build a dog house with height 0.  Although that dimension is a valid dimension on the paper, it is not practical.

So you may have asked the builder to build something that is not possible, in which case he/she will let you know at the time that he/she is executing the instruction set (run-time error).

A third type of error that may happen is due to a mistake that you have made when you designed your algorithm or when you were writing the set of instructions.  For example, you may have written "keep the distance between each steps at 1 foot" but you meant to say "keep the distance between each steps at 1/2 foot". In that case, the builder will build have the steps placed based on what he/she sees on the instruction set.  This kind of error is called logic error.

We can compare some of the characters and parts of this story to what we will see in a programming class:

|  |  |
| --- | --- |
| You know how to design a house | You know how to write a C++ program |
| The set of instructions you write | Your C++ program |
| The Japanese translator | The C++ compiler |
| The translated version of your program | The Executable file produced by the compiler |
| The Japanese architect | The C++ compiler |
| Resources | Linker, input data,  .... |
| The house | The output from your program |

You may find other similarities as well.

In general, you will go through several steps before you can get an output from your program;

1. First you will use an editor to type your program
2. then you will use a compiler (in our case a C++ compiler) to compile your program and to get an executable file
3. then, at last, you will run the executable file to obtain the output.

The most important part of a programmer's job is **solving the problem first**.  It is much harder to solve a problem than to translate your ideas to a specific programming language.  Thus, one should first think about a method and develop an algorithm to solve the problem.  An algorithm is a sequence of precise instructions, which results in a solution.  The keyword here is precision.  If your algorithm has ambiguity in it, then you will not get the correct answer.

You will get frustrated, annoyed maybe even angry at times… However, if you work hard you will find this course very challenging, rewarding, satisfying, and dare I say it… FUN!

**Problem Solving**

Problem solving is usually broken down into two major phases:

1. Problem solving phase
2. Implementation phase.

Phase 1 - In the first phase, you will take three steps:

* **Step I:** You will define the problem that you want to solve, clearly.
* **Step II:** You will design an algorithm that is precise and very well thought to solve the problem,
* **Step III:** You will test your algorithm on paper.  Your algorithm should work correctly, before you can write a program for it.

Phase 2 - In this phase, you will take two steps:

* **Step I:** Translate your algorithm to C++ language.  If you have a correct and precise algorithm, the translation should be almost line-by-line.  This translation must be correct and free of:
* Syntax errors, which are the errors resulted from incorrect use of the programming language syntax or violation of syntax rules.
* Computations that are not possible, such as dividing by 0
* Errors made by the programmer.  Such errors are those made by using wrong signs or arithmetic operators.
* **Step II:** Test the program to make sure it produces the correct results.  Make sure your test cases are different.  The only way to correctly test a program is to have many different test cases.

**Synopsis**

Before going through these exercises, please read Chapter 1 of the required reading. At the end of these laboratory exercises you should:

* Learn some basics about computer systems
* Learn about problem solving and programming
* Learn about software life cycle
* Illustration of the problem solving and programming - A simple C++ program

**Questions**

1. Find out the name of operating system that you will be using for this course.

In general, for the whole CSCCORE2I course, I’ll be using my MacBook Air, so apple will be used, however, I learnt last year that you can bootcamp your Mac so that it also runs Windows, which is useful for Visual Studio 2019 which we used last year to program in C. This year I’ll be doing the same, using Visual Studio 2019 to code C++.

1. Find out what type of text editor or Integrated Development Environment is recommended by the lecturer for this course.

They recommend Visual Studio 2019.

1. What are the steps that you need to take to create a program?
   1. Click on Visual Studio 2019.
   2. Click “continue without code”.
   3. Click “File” -> “New” -> “Project”.
   4. Click “Empty Project”.
   5. Click “Next”.
   6. Save to the location you want it saved to.
   7. Click “Create”.
2. What do you think would be the most suitable statement to complete the flowchart that is shown in Figure 1.

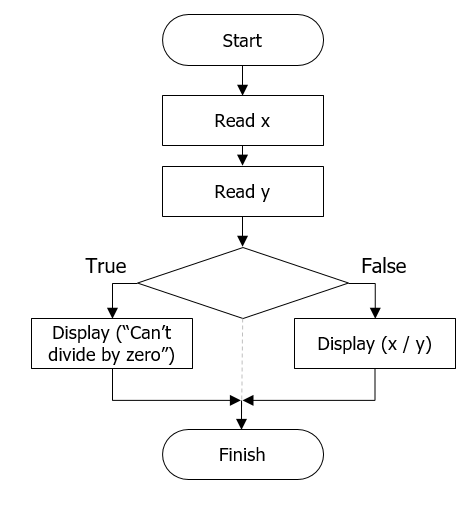


Figure 1- Conditional If

The most suitable statement to fit into this conditional statement diamond would be “y == 0”.

1. What do you think would be the most suitable statement to complete the flowchart that is shown in Figure 2.

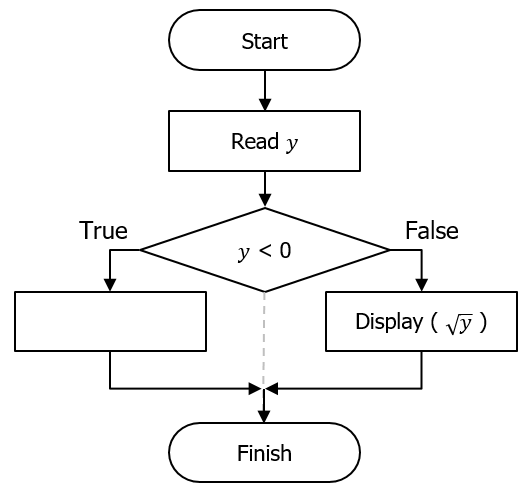


Figure 2 - Conditional If

The most suitable statement to complete the flowchart shown in figure 2 would be “display(‘Can’t find the square root of a number smaller than 0’)”.

1. Which one of these two are more difficult to do, problem-solving or programming? Explain your answer.

The most difficult out of problem-solving and programming, I would say, is problem-solving, because you’ve got to abstract the problem is such a way that the computer will be able to understand it. The writing of the program is simply translation.

1. How are you going to communicate with the computer to solve a problem?

To communicate with a computer to solve a problem, I am going to code.

1. Suppose you are helping the university registrar office with the registration process. You are to send students to six different halls depending on the first letter of their surname and the balance that has appeared on their bills. Here are the criteria you will use to separate them:
   * Students with balance zero and their surname begins with letters:
   * A-E in Hall 3, F-J in Hall 2, L-O in Hall 8, P-R in Hall 10, and S-Z in Hall 12.
   * Students with a non-zero balance go to Hall 18. These students can go back to register once they have a zero balance on their bills.

Create a flowchart that shows your solution to the given task.

The answer to this question is in my folder as I’d already drawn it on paper.

1. A grocery store sells many cases of soft drink every day.  In each case, there are 12 bottles and the store profits 20 pence per bottle. We want to compute the profit that the store makes every day for selling the soft drinks.  In addition to this we want to program to make an annual forecast; i.e. if we sold *x* amount of drinks every day for a year how much profit would be made?

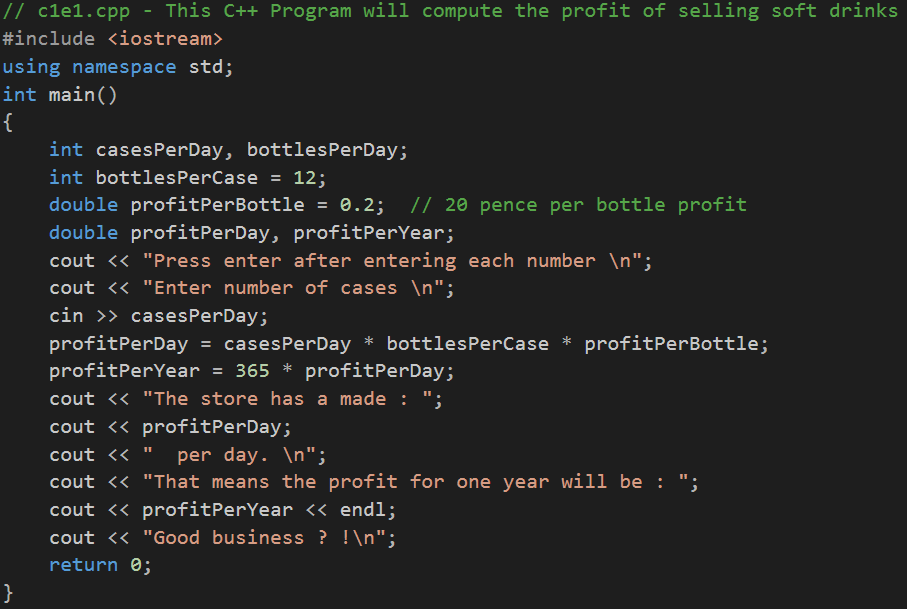
You are to produce a flowchart and Structured English that addresses the above problem.

The answer to this question is again in my folder as I’d done it previously.

**Programming Exercises**

We will now write a program in C++

1. Open Visual Studio and click new project, enter the following:
   * Name your project c1e1
   * Select Win32 Console Application
2. You are presented with an Application Wizard, select Next and ensure the following:  
   * Radio button Console Application is selected
   * Precompiled header is unchecked
   * Security Development Lifecycle (SDL) checks is unchecked
   * Empty project is checked
3. On the right-hand side of the IDE you should see a Solution Explorer, if you do not, use the View menu to select Solution Explorer.
4. In the Solution Explorer, right mouse click and select the Option Add🡪New Item.
5. Select C++ File (.cpp) and click Add.
6. In the coding window, type the code that is over the page.
7. Once you have typed the code, press Ctrl + F5, if a message dialog box appears stating that This project is out of date then select Yes to build it.



**Code Explained**

On the first line of the program you have:

// c1e1.cpp - This C++ Program will compute the profit of selling soft drinks

The // tells the compiler that the line is only a comment and does not participate in the computing.  Comments are added for readability and/or to describe parts of a program. The c1e1.cpp filename is Chapter 1, Exercise 1.

On the next line you have:

#include <iostream>

This is called an include directive.  It tells the compiler where to find information about some of the items that are used in your program. For example: cout*,* <<*,* >>, and cin in the above program.  There are other libraries that you will use to include other items.  Note that the directive always begins with ‘#**’**.

using namespace std;

This line will tell the compiler that the names defined in iostream are to be interpreted in a “standard way”.

int main( )   
{

Let’s just say that this marks the beginning of your main program.  Actually, intis used for type “integer”, main is a function name and ( ) will mark the boundary of parameters.  Note that the ‘{‘ marks the beginning of the main function and } marks the end of it.  In general, remember that for every open ‘{‘, you should have a corresponding ‘}’.

In the next four lines:

int casesPerDay, bottlesPerDay;

int bottlesPerCase = 12,

double profitPerBottle = 0.2;  // 20 pence per bottle profit

double profitPerDay, profitPerYear;

We will declare the variables that we plan to use and we also define their type.  We look at different variable types in the future labs.  In the above four lines int is used to declare variables of type “integer”.  On the second line we not only define the bottlesPerCase as an integer, we also initialized that to 12. On the 3rd and 4th lines, we have defined variables of type double.  Note that each instruction is ended with a ‘;’.

On the following two lines:

cout << “Press enter after entering each number \n”;   
cout << “Enter number of cases \n”;

we have used cout to display a message on the screen.  The *cout* statement will allow us to direct data from a variable out to the screen.

On the other hand, in the following line the cin directs data from the keyboard into a variable.

cin >> casesPerDay;

It is important but very easy to remember what the direction of << and >> is.  Note that when we send data to the screen, we send it to cout so the direction must be <<, and when we send data from the keyboard to a variable using cin the direction must be toward the variable >> .

In the sample program, we have performed some calculations:

profitPerDay = casesPerDay \* bottlesPerCase \* profitPerBottle;   
profitPerYear = 365 \* profitPerDay;

In the first line we have multiplied, ‘\*’, casesPerDay by bottlesPerCase by profitPerBottle, and stored the result, =, into profitPerDay.  We will learn about different arithmetic operators in future labs.  Explain what you have done in the second line.

**What could have gone wrong?**

**1) Syntax Error**

A Syntax error is the result of violation of the syntax (the grammar rules) of the programming language that you use.  For instance, if you forget to put ‘;’ at the end of a C++ instruction, your compiler will not correctly compile and will display an error on the screen.  To see the type of error, you can remove one of the ‘;’ a try to compile the program.  Another example of this type of error is if you do not have a paired open ‘{‘ and close ‘}’ set of braces.

**2) Logic Error**

The logic error will be the result of incorrect translation of your algorithm when you were writing the program.  This error will not be detected by the computer and the only way to find it is to test the program carefully after it is completed.  For instance, if in the following statement:

profitPerDay = casesPerDay \* bottlesPerCase \* profitPerBottle; **//correct**

which is the correct statement for computing the profitPerDay, but if by mistake, we use ‘+’ instead of ‘\*’, then we will have:

profitPerDay = casesPerDay + bottlesPerCase \* profitPerBottle;**//wrong**

We will get an answer, but that answer is not correct.  The error is the result of the mistake in the translation of our algorithm. Instead of ‘\*’ we have used ‘+’.

**3) Run-time Error**

A run-time error is detected when we run a program.  This type of error is mostly related to numeric calculations.  For example, a computer cannot compute the square root of negative number.

**Conclusion**

Congratulations, in this lab, you learned about some the basic steps you need to take to solve a problem using a computer program.  Also, you learned some of the syntax in C++. If you get through the lab exercises quickly why not try the supplementary exercises, but be warned, they are not as straight forward as they might appear. If you struggled and did not get through the lab exercises during the laboratory time, don’t worry, all you need to do is practise. You must ensure all lab exercises are completed and uploaded to Moodle on a weekly basis. Each laboratory exercise contributes to your overall mark.

**Submission Instructions**

1. Upload the file c1e1.cpp to the Moodle Link: Chapter 1 Lab Exercises

**Supplementary**

1. Modify program c1e1.cpp so that it uses 22 pence profit per bottle in the calculations.  This time in your output, display both the number of bottles sold and the profit for one day, one year, and 10 years.  Call your new program c1e2.cpp.
2. Rather than just output the pence, output any pounds. For example the output might be: This means the profit for one year will be 102 pounds and 40 pence.
3. Rather than using words use the £ symbol, for example, the output might be: This means the profit for one year will be £102.40.