CO1404 : Introduction to Programming

Week 2 – Variables, Operators and Conditions

Lab Summary

This lab worksheet helps you write your first programs. Students who have experience programming should find it straightforward at first. However, there will be advanced questions later on. There's a lot of reading in this worksheet as we introduce the tools required. Later worksheets will tend to be shorter, with more work for you to do!

Remember: If you have any questions or require assistance, please get your tutors attention. We are all here to help!

Objectives

This Lab aims to achieve the following objective(s) below:

PO.1: Introduce the development environment and programming language you will be using for this module.

Lab Activity

This weeks lab activity begins with three precursor tasks (Variable Declarations, Variables and Operators and if statements). These will help you complete stages 1 and 2 marked in the lab sheet. **Stage 1** is the ***absolute minimum*** point you should reach. Hopefully you can reach it in the lab session, if not use the drop-in help sessions and your own time to catch up. Reaching stage 2 suggests you are on target for a top class mark for the module. Set your goals sensibly - don't just aim for the minimum or you may struggle to pass the module.

Disclaimer : *The "Advanced Challenges" are intended as fun exercises designed to push your knowledge to the limits. I encourage you all to give them ago, ask questions if you get stuck but most importantly don't stress about them.*

Variables Declarations

For each of the following variable declarations, decide if it is:

- **Good**: *A valid declaration with a readable variable name (see lecture).*
- **Poor**: *A valid declaration, but a poor choice of variable name*
- **Invalid**: *The declaration has an error - it will not compile*

```
int numberStudents;
int totalCash = 210.50;
double num = 45.5;
string myID = "G1423";
string myName = "Bob"
int 1stPlaceScore;
float wall_length;
float wall height;
double pi = 3.14159;
Float AccountBalance;
char firstLetter = 'A';
char secondLetter = "B";
```

Write your answers in a document, paper, or into the worksheet and save it later for reference. It will help you during your revision. Check and explain your answers with the tutor.

Variables and Operators

1. Create a new Visual Studio C# Console project called **CalculateTax**. Refer to last week's worksheet if you have forgotten how to create a new project.
2. Copy the code below into the correct place in the shell code. Again, refer to last week's lecture / lab or ask the tutor if you are not sure where to put the code:

```
double itemPrice;
string inputString

Console.Write("Enter the item price: ");
inputString = Console.ReadLine();
itemPrice = double.Parse(inputString);

double itemTax = 8.5;
int totalPrice = itemPrice + itemTax;

Console.WriteLine("Tax on the item is {0}", itemTax);
Console.WriteLine("Total item price is {0}", totalprice);
```

```
Console.ReadLine();
```

3. Now you maybe wondering, "The code you told me to copy has errors". Unfortunately, it does. Sorry about that. Fix the errors so that the program will run.

4. The program adds a fixed amount of tax to the price, which isn't very realistic. Update the value of `itemTax` to be `20%` of `itemPrice`. This involves a calculation using addition and division. Try to do this on your own, but if you get stuck there's an example in the lecture.

5. After displaying the total value with tax, add some extra code to the program to ask the user `"How many items do you want to buy?"`. Read an integer from the user for their answer, then calculate the `total cost` (`totalPrice * number of items`). Output a final message in this style:

   ```
   The price for 4 items is 57.60
   ```

6. Add some sensible comments to your code. To comment your code you use `// <some comment>`. For example:

   ```
   double reactorCoreTemp;

   if (reactorCoreTemp > 20.5)
   {
     Warning.redAlert("We have a big problem"); // Sounds the alarm.
   }
   else
   {
     Console.Writeline("Reactor Core is operating at {0}", reactorCoreTemp);
   }
   ```

   If you are unsure about your comments, please check them with your tutor.

An `if` Statement

1. Create a new Visual Studio C# Console project called **BirthYear** and copy the code below into the correct place in the shell code:

```
int year;
string inputString;

Console.Write("Enter your year you were born: ");
inputString = Console.ReadLine();
year = int.Parse(inputString);

if (year % 4 == 0);
{
  Console.WriteLine("You were born in a leap year");
}

Console.ReadLine();
```

2. Look carefully for the single warning that Visual Studio highlights. This mistake was highlighted in the lecture. Fix this simple bug (if you don't fix it then the program will run, but it will say every year is a leap year - try running it first it if you wish).

3. Run the program and it will tell you whether your birth year was a leap year. It works by seeing if there is a 0 remainder after dividing the year by 4, which is the same as saying that the year divides by 4 exactly - the main rule for leap years (actually there are other rules for leap years but they don't affect any recent years).
4. Add an `else` statement to display the message `"You were not born in a leap year"` appropriately. Check the lecture notes to see where to put an else statement (alternatively, have another look at the Reactor Core example in the previous section) . Be precise and try to line things up in the same way as the lecture.

5. Add another `if-else` statement that checks if the user was born in the same year as yourself. It should display `You were born in the same year as me!` or `You weren't born in the same year as me`.

Advanced Challenge

Can you work out how to update your code to say one of `You're older than me`, `You're younger than me` or `You're the same age as me`?

STAGE 1

This stage consists of three elements: ***A Common Error: Integer divided by Integer***, ***Currency Conversion***, ***Maximum Number***.

A Common Error: Integer divided by Integer

1. Create a new Visual Studio C# Console project called **IntegerDivision**.

2. Copy the code below into the correct place in the shell code:

```
double half = 1/2;
Console.Write("One half (1/2) = {0}", half);
Console.ReadLine();
```

3. Run the program. That's strange... it says `1/2 = 0` .

   Why does this happen? The computer calculates the division `1/2`, but as both 1 and 2 are integers (there's no decimal point) it thinks you want an integer as an answer. So it rounds the result 0.5 down (to the floor) to 0. The computer does this *before* it sees that the result is being put into a double (remember: *computers are stupid*).

   🔎: On a different point, this is one of the rare times that C# will convert types automatically, it converts its integer answer 0 to a double without giving you an error.

4. Fix the problem by writing `1.0` and `2.0` instead. This shows that they are **floating point values** and indicates to C# that you want a floating point answer. The program will run correctly now.

   This issue can often catch you out and cause unexpected bugs. It usually happens when you are programming a formula. Here's a standard question for novice programmers where you must be careful about this problem:

5. Remove that code and write a program that asks the user for a temperature in Celsius (an integer). Convert this temperature to Fahrenheit using the formula:

```
Fahrenheit = Celsius * (9/5) + 32
```

   Tip : Watch the brackets. Display the result as a `double` (example 25C = 77.00F).

Currency Conversion

1. Create a new Visual Studio C# Console project called **CurrencyConverter**.

2. Write a program that inputs from the user an amount in pounds (£) and converts it to an amount in dollars ($), displaying the result. At first assume that the exchange rate is $1.5 for £1. Check your result carefully and make sure you are converting in the correct direction.

3. Update your program so the user can first enter the exchange rate to use instead of just using 1.5.

4. Extend the program further by first prompting the user `"Type 1 to convert from £ to $, or type 2 to convert from $ to £`. Then read the user's choice (as an integer). Use an `if-else` statement that tests if this integer is equal to 1 as its condition. In the first block of the if statement, put the code to convert from £ to $, and in the second block (`else`) add code to convert the other way, from $ to £. The code to convert the other way is almost the same.

   Advanced Challenge

   Clearly there is a lot of similar code between the if and else blocks here. Try to reduce the amount of duplicate code to the minimum. You can get it down to only one line in each block of the 'if-else' statement, although it will need some thought.

Maximum Number

1. Write a program that asks the user to input five different integers. Using `if` statements work out which is the maximum of the five numbers and display it. You will need several `if` statements to do this exercise. Ask your tutor for advice if necessary.

If you have completed all three elements, you have successfully completed stage one

---

STAGE 2

This stage consists of a single element: ***Sorting***. This exercise is designed to very difficult but you have already reached the first class stage for this lab sheet so challenges here are supposed to be very tough. You will gain experience just by thinking about this exercise.

Sorting (!)

1. Write another program that asks the user to input five different integers. Using `if` or `if-else` statements ***only***, display the numbers in increasing order. E.g. user inputs: `5`, `2`, `11`, `1`, `4`. Then the program outputs: `1` , `2`, `4`, `5`, `11`.

   Hint : *Possible approach: Compare the 2nd, 3rd, 4th and 5th values in turn with the 1st value. If any value is less than the 1st value then swap it with the first value. Think carefully how to swap two variables. Now do the same comparing/swapping with values 3,4 & 5 against value 2. Then compare/swap 4, 5 against 3, then finally 5 against 4. There's a pattern there. Ten 'if' statements in total. Good luck!*