Basics

April 20, 2018

1 Create your first program

```
In [1]: #include <iostream>

         int main()
        {
             std::cout << "Hello world, I am ready for C++"; // prints out string
             return 0; // returns 0
             /* We can write comments like
             that for multiple lines*/
             // We can write one-liners like that
        }
```

The first line we encounter has a hash at the start of the line. Any line that has a hash sign at the start is a preprocessor directive. include <iostream> After the hash sign we have the word include. There are several preprocessor directives available in C++, but include is the one we see and use the most often.

Include means add the declarations of the given library. In this case we are adding the declarations of the iostream library.

The final detail of this line .... The brackets..

The brackets say “Look for this file in the directory where all the standard libraries are stored”. C++ also allows us to specify the library name using double quotes.

```
In [ ]: #include "main.hpp"
```

The double quotes say “look in the current directory, if the file is not there, then look in the directory where the standard libraries are stored”.

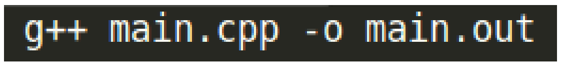
If we change the brackets to double quotes, for this case, the program still compiles. We will see later in the course a situation where using the correct enclosure (Brackets or quotes) makes a difference.

The next section is the main section of the program. We will discuss this in more detail in the next concept.

```
In [ ]: int main()
         {
              std::cout << "Hello world, I am ready for C++";
              return 0;

          }
```

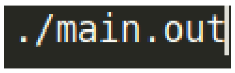
2 How to compile a program:

The first thing is to compile the file by running: 

```
g++ main.cpp -o main.out
```

- *g++* is for the C++ compiler
- *main.cpp* is the name of the file
- *-o* is the command that you define the output
- *main.out* is the name of the output

Having done that, you will have created a RUN file :

Then run this file with: 

```
./main.out
```

3 Using Namespace

Writing std:: can be a pain. So C++ actually offers a shortcut for writing cout.
Before the start of the main function, put in the command "using namespace std;"

```
In [ ]: using namespace std;
        int main()
        {
        }
```

This tells the compiler to assume we are using the standard library, so we don't have to write std::.

I will add there is some controversy about using namespace. When the commands are not explicitly defined, there is a possibility that when your code is added to a large project, your code might reference a command from a different library.

In this class, its use is up to you. Sometimes I will use namespace and sometimes I will not.

4 Print formatting

4.1 Escape sequences

The most common ones are: - newline -tab

```
In [8]: int integer = 456;

        std::cout<<"The variable value is: "<<integer;
```

```
The variable value is: 456
```

```
In [11]: std::cout<<"The variable value is: "<<"\n"<<integer;
```

```
The variable value is:
456
```

```
In [15]: std::cout<<"The variable value is: "<<"\t"<<integer;
```

```
The variable value is:          456
```

We can also format the output by using the iomanip library.
Include it as #include .

4.2 Iomainp

Once it is included, you can format output using the iomanip library. For example, we can set the width of an output using the setw command.

```
In [16]: #include <iomanip>

         std::cout<<"\n\nThe text without any formating\n";
         std::cout<<"Ints"<<"Floats"<<"Doubles"<< "\n";
         std::cout<<"\nThe text with setw(15)\n";
         std::cout<<"Ints"<<std::setw(15)<<"Floats"<<std::setw(15)<<"Doubles"<< "\n";
         std::cout<<"\n\nThe text with tabs\n";
         std::cout<<"Ints\t"<<"Floats\t"<<"Doubles"<< "\n";
```

```


The text without any formating
IntsFloatsDoubles

The text with setw(15)
Ints         Floats        Doubles


The text with tabs
Ints        Floats        Doubles
```

A sample solution to the Format Output Program.
/Formatting Output Goal: practice using cout to format output to console Print the variables in three columns: **Ints, Floats, Doubles /

```
In [ ]: #include <iostream>
         #include <iomanip>

         int main()
         {
             int a = 45;
             float b = 45.323;
             double c = 45.5468;
```

```
    int aa = a + 9;
    float bb = b + 9;
    double cc = c + 9;
    int aaa = aa + 9;
    float bbb = bb + 9;
    double ccc = cc + 9;

    std::cout<<"print with set width = 10\n";
    std::cout<<"Ints"<<std::setw(10);
    std::cout<<"Floats"<<std::setw(10);
    std::cout<<"Doubles"<<std::setw(10) << "\n";

    std::cout<< a;
    std::cout<< std::setw(12)<< b;
    std::cout<< std::setw(10)<< c << "\n";

    std::cout<< aa;
    std::cout<< std::setw(12)<< bb;
    std::cout<< std::setw(10)<< cc << "\n";

    std::cout<< aaa;
    std::cout<< std::setw(12)<< bbb;
    std::cout<< std::setw(10)<< ccc << "\n\n";

    std::cout<<"print with tabs\n";
    std::cout<<"Int"<<"\tFloats"<<"\tDoubles\n";
    std::cout<< aaa<<"\t"<< bbb<<"\t"<< ccc <<"\n";

    return 0;
}
```

5 Size of variable types

In [ ]:
```
#include <iostream>
using namespace std;
int main(){
    cout<<"short size = "<<sizeof(short)<<"\n";
    cout<<"long size = "<<sizeof(long)<<"\n";
    cout<<"char size = "<<sizeof(char)<<"\n";
    cout<<"float size = "<<sizeof(float)<<"\n";
    cout<<"double size = "<<sizeof(double)<<"\n";
    cout<<"bool size = "<<sizeof(bool)<<"\n";
}
```

6 Define Constants

6.1 Define Constants

In C++ we can define a variable as a constant. Meaning, its value does not change for the life of the program.

We use the keyword 'const' to define a constant.

```
In [ ]: #include <iostream>
        using namespace std;

        int main()
        {
            const int weightGoal = 100;
            cout<<"WeightGoal = "<<weightGoal<<"\n";
            weightGoal = 200;
            cout<<"WeightGoal = "<<weightGoal<<"\n";
            return 0;
        }
```

With this statement we have set the integer weightGoal to 100. It cannot be changed during the program. If you want to change the value of weightGoal, you will have to edit the source code and recompile it.

6.2 Enumerated Constants

C++ also allows for enumerated constants. This means the programmer can create a new variable type and then assign a finite number of values to it. Here is the form of the enum keyword: enum type$_namevalue1, value2, value3, ..object_names;$

```
In [ ]: enum MONTH {Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec};
```

In this example the enum variable MONTH has twelve possible values. These 12 values translate into 12 integer values.

```
In [ ]: Jan = 0
        Feb = 1
        //etc.
```

```
In [ ]: /*Enum example*/

        #include <iostream>

        using namespace std;

        int main()
        {
            //define MONTHS as having 12 possible values
            enum MONTHS {Jan, Feb, Mar, Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec};
            //define bestMonth as a variable type MONTHS
            MONTHS bestMonth;

            //assign bestMonth one of the values of MONTHS
            bestMonth = Jan;

            //now we can check the value of bestMonths just
            //like any other variable
            if(bestMonth == Jan)
            {
                cout<<"I'm not so sure January is the best month\n";
            }
            return 0;
        }
```

7 Input / Output

7.1 I/O Files

As with other programming languages, we can read and write files.
File IO Steps:

- Include the library
- Create a stream (input, output, both)
  - ofstream myfile; (for writing to a file)
  - ifstream myfile; (for reading a file)
  - fstream myfile; (for reading and writing a file)
- Open the file myfile.open("filename");
- Write or read the file
- Close the file myfile.close();

In the next quiz you are going to see the output stream and the input stream in action. Then I want you to play with this program by adding text to the input.txt file and then changing the ifstream and ofstream commands to fstream . Make sure that the text you write is read by the program.

```
In [ ]: #include <iostream>
        #include <fstream>
        #include <string>
        using namespace std;

        int main () {
            string line;
            //create an output stream to write to the file
            //append the new lines to the end of the file
            ofstream myfileI ("input.txt", ios::app);
```

```
    if (myfileI.is_open())
    {
        myfileI << "\nI am adding a line.\n";
        myfileI << "I am adding another line.\n";
        myfileI.close();
    }
    else cout << "Unable to open file for writing";

    //create an input stream to read the file
    ifstream myfileO ("input.txt");
    //During the creation of ifstream, the file is opened.
    //So we do not have explicitly open the file.
    if (myfileO.is_open())
    {
        while ( getline (myfileO,line) )
        {
            cout << line << '\n';
        }
        myfileO.close();
    }

    else cout << "Unable to open file for reading";

    return 0;
}
```

Before you begin this quiz, I want to talk about the file called input.txt.

Our program reads and writes to this file. You can look at this file, you will see it has text in it. Our program will first write to this file, then read from it. Normally, if a program writes to a file we can open that file and see the added text in the file. We can't do that for input.txt. We won't see the added text in the file.

But we know it is writing to the file because we can see it being read by the program. We can also change input.txt and we will see the program reading the new text.

7.2 User Input

In C++ we use std::cout for writing to the console.

And we have std::cin for reading from the console.

```
In [ ]: /*This program accepts inputs from the input.txt file*/

        #include <iostream>
        #include <string>


        int main()
        {
            int year = 0;
            int age = 0;
            std::string name = " ";
            //print a message to the user
            std::cout<<"What year is your favorite? ";

            //get the user response and assign it to the variable year
            std::cin >> year;

            //output response to user
            std::cout<<"How interesting, your favorite year is "<<year<<"!\n";

            //print a message to the user
            std::cout<<"At what age did you learn to ride a bike? ";

            //get the user response and assign it to the variable age
            std::cin >> age;

            //output response to user
            std::cout<<"How interesting you learned to ride at "<<age<<"!\n";

            std::cout<<"What is your name? ";
            std::cin>>name;
            std::cout<<"Hello "<<name<<" !\n";
            return 0;
        }
```

However there is an issue with std::cin! It cannot read strings that contain space!

7.3 String Input

So, we now know that std::cin will not retrieve strings that have a space in them. It will see the space as the end of the input. We will obviously need a method to enter strings.

C++ has a function called **getline**. You can find detailed information at this link.

std::getline(std::cin, variableName);

The basic form of getline is:

getline: it will *retrieve* characters from the std::cin *source and stores* them in the variable called variableName. It will retrieve all characters until the newline or "" is detected.

```
In [ ]: /*Goal: practice std::cin for strings
        **Write a program that prompts two users for their
        **name, address, and phone number.
        **Print the information to the console in the following format:
        **name
        **\/t\/t address
        **\/t\/tphone number
        */

        /* We take the data from a txt file with the following data:
```

```
Imogene Penelope Freely
         2343 South View Road
         (408)435-3221
Sandy Beaches
         1123 Pebble Creek Road
         (408)546-5432
*/
#include <iostream>
#include <string>

int main()
{

    std::string name;
    std::string address;
    std::string phone_number;
    std::getline(std::cin, name);
    std::getline(std::cin, address);
    std::getline(std::cin, phone_number);
    std::cout<<name<<'\n';
    std::cout<<"\t"<<"\t"<<address<<"\n";
    std::cout<<"\t"<<"\t"<<phone_number<<"\n";
    return 0;
}
```

In [ ]:
```
// A fancier answer:


/* We take the data from a txt file with the following data:
Imogene Penelope Freely
         2343 South View Road
         (408)435-3221
Sandy Beaches
         1123 Pebble Creek Road
         (408)546-5432
*/

#include<iostream>
#include<string>

int main()
{
    std::string name1, address1, phoneNo1;
    std::string name2, address2, phoneNo2;

    // get user 1 information
    std::cout<<"Used1 what is your name?\n";
    std::getline(std::cin, name1);
    std::cout<<"User1 what is your address?\n";
    std::getline(std::cin, address1);
    std::cout<<"User1 what is your phone number?\n";
    std::getline(std::cin, phoneNo1);

    // get user2 information
    std::cout<<"Used2 what is your name?\n";
    std::getline(std::cin, name2);
    std::cout<<"User2 what is your address?\n";
    std::getline(std::cin, address2);
    std::cout<<"User2 what is your phone number?\n";
    std::getline(std::cin, phoneNo2);

    // print information

    std::cout<<"\n\n"<<name1<<"\n";
    std::cout<<"\t\t"<<address1<<"\n";
    std::cout>>"\t\t"<<phoneNo1<<"\n";

    std::cout<<"\n\n"<<name2<<"\n";
    std::cout<<"\t\t"<<address2<<"\n";
    std::cout>>"\t\t"<<phoneNo2<<"\n";

    return 0;
}
```

7.4 Stringstream library

There is one further aspect of string inputs that you might find handy.

First we will need to include the Stringstream library. Then use getline to get the string from the user Then we will convert the string variable to a numeric variable.

```
In [ ]: //step 1: Include the Stringstream library

        #include<sstream>

        //step 2: Use getline to get the string from the user

        std::getline(std::cin, stringVariable);

        //step 3: Convert the string variable to float variable

        std::stringstream(stringVariable) >> numericVariable;
```

```
In [ ]: // Example

        #include <iostream>
        #include <string>
```

```
#include <sstream>

int main ()
{
    std::string stringLength;
    float inches = 0;
    float yardage = 0;

    std::cout << "Enter number of inches: ";
    std::getline (std::cin,stringLength);
    std::stringstream(stringLength) >> inches;
    std::cout<<"You entered "<<inches<<"\n";
    yardage = inches/36;
    std::cout << "Yardage is " << yardage;
    return 0;
}
```

In [ ]:
```
/*
 **Prompt the user for the length of a room.
 **Then prompt for the width of the room.
 **Print out the area of the room.
 */

 #include <iostream>
 #include <string>
 #include <sstream>

 int main ()
 {
   std::string stringLength, stringWidth;
   float length = 0;
   float width = 0;
   float area = 0;

   std::cout << "Enter the length of the room: ";
   //get the length as a string
   std::getline (std::cin,stringLength);
   //convert to a float
   std::stringstream(stringLength) >> length;
   //get the width as a string
   std::cout << "Enter width: ";
   std::getline (std::cin,stringWidth);
   //convert to a float
   std::stringstream(stringWidth) >> width;
   area = length * width;
   std::cout << "\nThe area of the room is: " << area << std::endl;
   return 0;
 }
```

8 Header Files

As we have seen we can include additional libraries in C++, we can also include our own libraries.
Traditionally, these files are called header files and they have an .hpp extension. Although any extension will work.

- Header files contain information about how to do a task.
- The main program contains information about what to do.

Let's see how a header file works with a simple hello world program.
We have a simple hello world program. We can test this, and the program runs.

```
In [ ]: /*
        This is the version without header files
        */
        #include <iostream>
        #include <string>

        using namespace std;

        int main()
        {
            cout<<"Hello, I use header files!";
            return 0;
        }
```

```
In [ ]: /*
        We will create a main.hpp file, where it will contain the following code:
        */

        #include <iostream>
        #include <string>

        using namespace std;

        /*
        The main.cpp file, will include the following code
        */

        #include "main.hpp"

        int main()
        {
            cout<<"Hello, I use header files!";
            return 0;
        }

        // Notice that I am using double quotes ("") instead of angle brackets. The main.hpp is
```

9 Review

In this chapter we will make an overall review

9.1 Debugging

We will debug a script that is full of errors.

```
In [ ]: // Initial script
        #include <main.hpp>

        void main ()
        {
            int FTemp = 0
            int CTemp = 0;

            cout >> "Enter a Farenheit temperature: ";
            cin << FTemp;

            CTemp = FTemp - 32 / (9/5);
            cout >> "\n <<FTemp >> " in Farenheit = "  >> CTemp >> in Celsius\n";
            return 0;
        }

        // main.hpp

        /*The header file*/

        #include<iostream>

        using namespace std;


        // input.txt
        32
```

```
In [ ]: // Debugged script

        #include "main.hpp"

        int main()
        {
            float FTemp = 0;
            float CTemp = 0;

            cout << "Enter a Farenheit temperature: ";
            cin >> FTemp;
```

```
    CTemp = (FTemp - 32.0)/ (9.0/5.0);
    cout << "\n" << FTemp << "in  Farenheit " << CTemp << " in Celsius\n";
    return 0;
}

// main.hpp

/*The header file*/

#include<iostream>

using namespace std;


// input.txt
32
```

9.2 Playground

We will create 4 files for a program.

- main.cpp - the main program
- main.hpp - the header file
- mainFunctions.cpp - a file for any functions
- input.txt - for user inputs. Delete the first line of the file, otherwise the line will be seen as an input.

```
In [ ]: // main.cpp

        /*Put your code here*/

        #include "main.hpp"
        #include "mainFunctions.cpp"

        int main()
        {
            return 0;
        }

        //main.hpp

        /*Put your code here*/

        #include "main.hpp"
        #include "mainFunctions.cpp"
```

```
int main()
{
    return 0;
}

// mainFunctions.cpp
/*functions can go here if you like*/


// input.txt
/*This file can be used to input user responses*/
```

Test Run will execute the following commands:

- Compile the code and create an executable file called main.out
  - **g++ main.cpp -o main.out**
- Execute the executable file using input.txt as the input
  - **./main.out input.txt**