C++ Notes

PART-1

Source code

1. We will be making use of code blocks as our IDE.
2. For setting up the most basic c++ code, open code blocks and go to file>new>project>C++......This sets up a basic c++ project that prints out “Hello world”
3. This is that program:

   ```
   #include <iostream> //preprocessor directive (include these files)

   using namespace std; //standard library

   int main() //main function
   {
   cout << "Hello world!" << endl;
   return 0;
   }
   ```

   Analysis of the main function:
   1. cout (read as c-out) - It actually prints out some string or any other data on the screen.
   2. << - string insertion operator
   3. endl - End of line statement (This just moves the cursor to a new line, \n can also be used)
      endl is not important as the following program can also print out “Hello world”:

      ```
      #include <iostream>

      using namespace std;

      int main()
      ```

```
{
        cout << "Hello world!"  ;
        return 0;
}
```

4. The main function must always return 0

4. **Printing text on the screen:**

Use << endl whenever you feel like moving to the next line. However, \n can also be used.

5. **Variables**

   Eg:

   int tuna=6;

6. **Taking user input:**

   cin takes user input.

   >> is extraction operator

   We make the use of:

   Program for adding numbers:

```
#include <iostream>

using namespace std;

int main()
{
        int a,b,c;
        cout << "Enter two numbers:";
        cin >> a;
        cin >> b;
        c=a+b;
        cout << "The sum is:" << c << endl;
}
```

7. **if statements**

   Eg1:

```
if (3>1)
{
```

```
cout << “I am awesome”;
}
```
Eg2:
```
if (3>=3){
cout >> “No bud”;
}
```
Eg3:
```
if(1=<3){
}
```
Note: != is for not equals to.

8. **Functions**

   >prototype of a function

   ```
   return type name_of_function(function_parameters)
   {
   statements;
   }
   ```

   Eg:

   ```
   #include <iostream>

   using namespace std;

   void printmeout()
   {
   	cout << "This is inside the function";

   }

   int main()
   {
   	cout << "I am going to use a function:" << endl;
   	printmeout();
   	return 0;
   }
   ```

Note: The flow in which a C++ program is read is from top to bottom. If we would have declared the function after calling it, it would give an error. However, the function prototype can be used.

I.e.

```
#include <iostream>

using namespace std;

void printmeout(); // function prototype

int main()
{
        cout << "I am going to use a function:" << endl;
        printmeout();
        return 0;
}

void printmeout()
{
        cout << "This is inside the function";

}
```

9. Classes and objects (Object-oriented programming)

Classes are meant to add similar functions under one category.
For example: if someone is dealing with some temperature-related functions than they can place these functions under the same class.

class declaration:

```
class name_of_class () {
public:
        void f1()
                {
                }
        int f2(int a, int b)
        {
        statements;
        }
private:
        void f3(){
                }

}
```

Public vs private?
Public objects in a class are universally available to be used i.e. global objects. They can be used in the main function also. However, private objects are available only in the scope of this class.

Calling out the public functions of a class using objects?

```
int main()
{
name_of_class  name_an_object; // Separated by space

// calling the public function using the dot operator
name_of_object.name_of_function; //name_of_object.f1;

}
```

Real example:

```
#include <iostream>

using namespace std;

// Declare a class and define function inside the class
class BuckysClass{
        public: // You can use the function outside of the class.
        void coolSaying(){
        cout << "preachin to the choir" << endl;
        }

};

int main()
{
        BuckysClass buckysObject; // Create an object from
BuckysClass
        buckysObject.coolSaying();
        return 0;
}
```

10. Using variables in classes

1. **Wrong method**

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

using namespace std;

class adityasclass
{
        public:
        string name; //this is a variable
};


int main()
{
        adityasclass ao;
        ao.name = "aditya";
        cout << ao.name;
        return 0;
}
```

Why this is the wrong method?
Making class variables public is not a good practice. Rather make a private class variable and make a public function inside the scope of the same class.

2. **Right method**

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

using namespace std;

class adityasclass
{
        public:
        void setname(string x){
        name=x;
        }
        string printname(){
        return name;
        }
        private:
        string name; //this is a variable


};


int main()
{
        adityasclass ao;
        ao.setname("Sir Jadeja");
        cout << ao.printname();
        return 0;
}
```

11. Constructors (special function)

- Never have a return type
- Mainly used to assign values
- Just type in the constructor name, it is always the same as the class name.
- The constructor is called as soon as an object of the same class is created inside the main
- Generally, constructors aren't used to printing stuff out.

Eg:

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

using namespace std;

class adityasclass
{
        public:
        adityasclass(){
        cout << "Man of steel and burnt with wrath";
        }

        void setname(string x){
        name=x;
        }
        string printname(){
        return name;
        }

        private:
        string name; //this is a variable


};


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

Output: Man of steel and burnt with wrath

- Use of constructors to assign “name” from the previous program to assign a value (the actual use of constructors).

  Eg:

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

```
using namespace std;

class adityasclass
{
        public:
        adityasclass(string z){
        setname(z);
        }

        void setname(string x){
        name=x;
        }
        string printname(){
        return name;
        }

        private:
        string name; //this is a variable


};


int main()
{
        //thus by using constructor we are passing
on a string in the object itself
        adityasclass ao("Cmon, matey");
        cout << ao.printname();
        return 0;
}
```

- Using two objects to assign the same value (They do not overwrite)

Eg:

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

using namespace std;

class adityasclass
{
        public:
        adityasclass(string z){
        setname(z);
        }

        void setname(string x){
        name=x;
        }
        string printname(){
        return name;
        }

        private:
        string name; //this is a variable


};


int main()
{
        //thus by using constructor we are
passing on a string in the object itself
        adityasclass ao("Cmon, matey");
        cout << ao.printname();
```

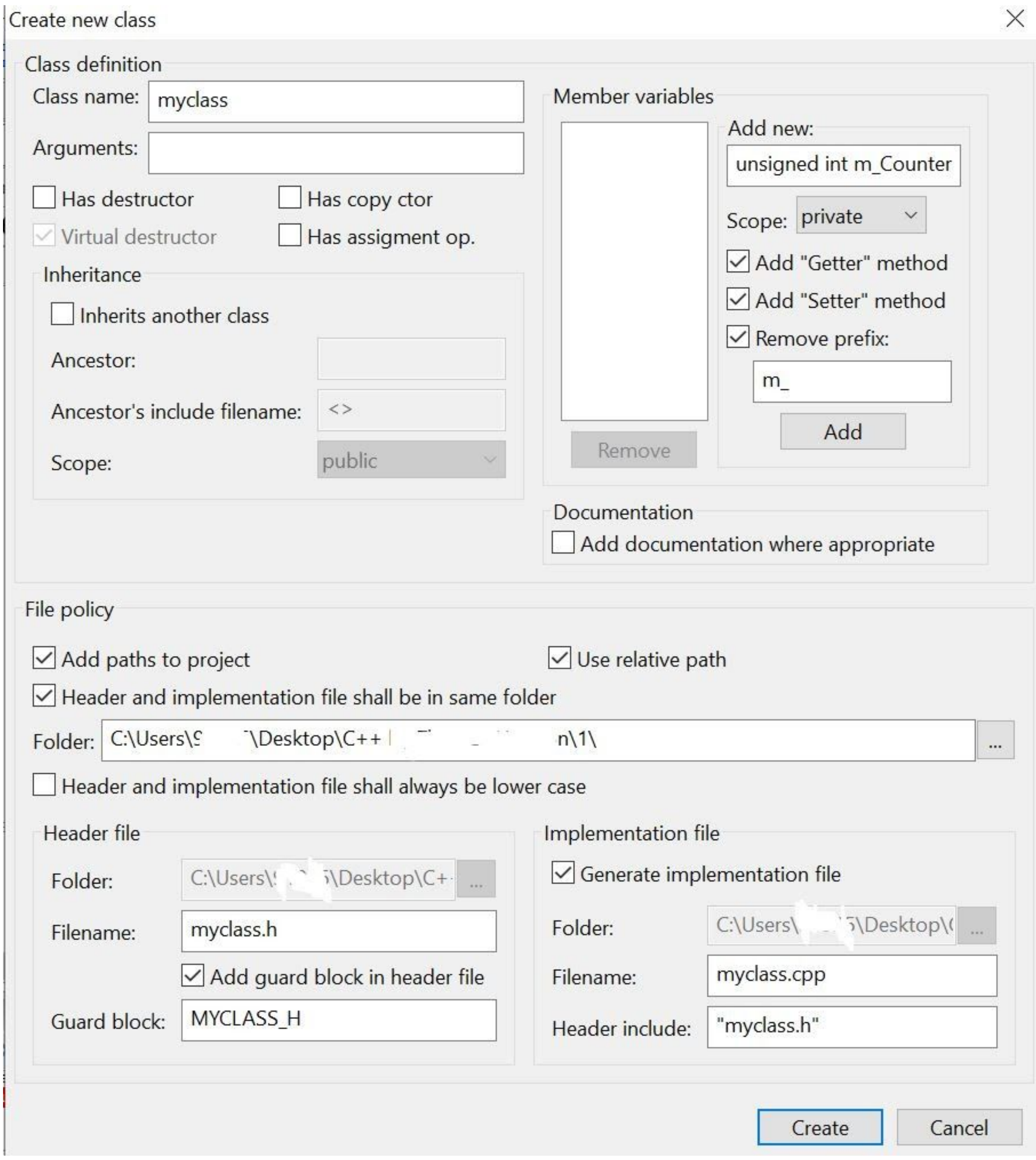
```
    adityasclass ao1("sally mcsalad");
    cout << ao1.printname();

    return 0;
}
```

Output:

```
Cmon, mateysally mcsalad
```

12. Placing classes in separate files



Create new class

Class definition

Class name: myclass

Arguments:

☐ Has destructor ☐ Has copy ctor

✓ Virtual destructor ☐ Has assigment op.

Inheritance

☐ Inherits another class

Ancestor:

Ancestor's include filename: <>

Scope: public

Member variables

Add new: unsigned int m_Counter

Scope: private

✓ Add "Getter" method

✓ Add "Setter" method

✓ Remove prefix: m_

Add

Remove

Documentation

☐ Add documentation where appropriate

File policy

✓ Add paths to project ✓ Use relative path

✓ Header and implementation file shall be in same folder

Folder: C:\Users\S \Desktop\C++ n\1\

☐ Header and implementation file shall always be lower case

Header file

Folder: C:\Users\ 5\Desktop\C+

Filename: myclass.h

✓ Add guard block in header file

Guard block: MYCLASS_H

Implementation file

✓ Generate implementation file

Folder: C:\Users\ 5\Desktop\(

Filename: myclass.cpp

Header include: "myclass.h"

Create Cancel

Go to file>new>class

Make sure that your desktop shows this and check and uncheck accordingly. A couple of files would be created one header file and another a CPP file.

We will put all the classes, function prototype and functions in the header file.