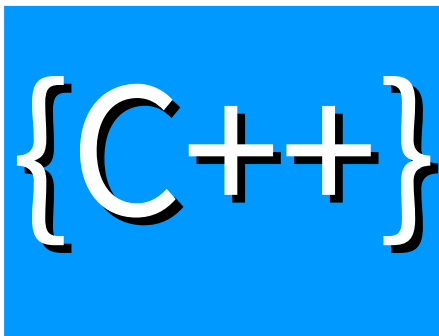


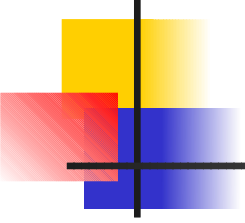


# Classes

---



Yang-Cheng Chang  
Yuan-Ze University  
[yczhang@saturn.yzu.edu.tw](mailto:yczhang@saturn.yzu.edu.tw)



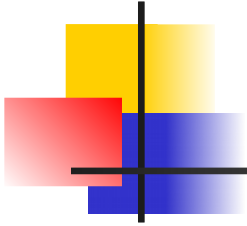
# Example: Rewrite the code to C++ Class

---

```
#include <iostream>
#include <string>
using namespace std;

struct person
{
    string name;
    int age;
};

int main()
{
    person a, b;
    a.name = "Calvin";
    b.name = "Hobbes";
    a.age = 30;
    b.age = 20;
    cout << a.name << ": " << a.age << endl;
    cout << b.name << ": " << b.age << endl;
    return 0;
}
```



# Rule of Three

---

- The code can be copied once, but that when the same code is used **three** times, it should be extracted into a new procedure.



# toString function

---

```
#include <iostream>
#include <string>
#include <sstream>
using namespace std;

struct person
{
    string name;
    int age;
};

string toString(person &pn)
{
    ostringstream stringStream;
    stringStream << "Name: " << pn.name << ", " << "Age: " << pn.age;
    return stringStream.str();
}

int main()
{
    person a, b;
    a.name = "Calvin";
    b.name = "Hobbes";
    a.age = 30;
    b.age = 20;
    cout << toString(a) << endl;
    cout << toString(b) << endl;
    return 0;
}
```

# The definition of class(.h)

Include guard

```
#ifndef PERSON_H
#define PERSON_H
// Demonstrates how to rewrite c++ struct to c++ class
```

attributes

```
using namespace std;
class person {
    private:
        // member variables
        string name;
        int age;
    public:
        // constructor function
        //person( string, int );
        person( string = "noname", int = 5 );
        // member functions
        void setName( string );
        void setAge( int );
        string toString();
};
```

member  
functions

```
#endif // PERSON_H
```

default parameters

# The definition of member functions(.cpp)

Include the definition of  
person class

class name::function name

```
#include <iostream>
#include <sstream>
#include "person.h"
using namespace std;

person::person(string nm, int ae)
{
    setName(nm);
    setAge(ae);
}

void person::setName( string nm )
{
    name = nm;
}

void person::setAge( int ae )
{
    age = ae;
}

string person::toString()
{
    ostringstream stringStream;
    stringStream << "Name: " << name << ", " << "Age: " << age;
    return stringStream.str();
}
```



# Class in a Separate Header File for Reusability

---

- .cpp files for source-code implementations
  - Class implementations
  - Main programs
  - Test programs
  - ...
- .h Header files
  - Separate files in which class definitions are placed.
  - Allow compiler to recognize the classes when used elsewhere.
  - Generally have .h filename extensions



# Creating an object of a Class

---

- Declaring a variable of a class type creates an object. You can have many variables of the same type (class).
  - Instantiation
- Once an object of a certain class is instantiated, a new memory location is created for it to store its data members and code
- You can instantiate many objects from a class type.
  - Ex) `Circle c; Circle *c;`



# Instance an object with default parameters

```
class person {  
    private:  
        // member variables  
        string name;  
        int age;  
    public:  
        // constructor function  
        //person( string, int );  
        person( string = "noname", int = 5 );  
        // member functions  
        void setName( string );  
        void setAge( int );  
        string toString();  
};
```

## Main program

```
#include <iostream>  
#include "person.h"  
  
int main(int argc, const char *argv[])  
{  
    person noname;  
    person tom("Tom", 20 );  
  
    cout << noname.toString() << endl;  
    cout << tom.toString() << endl;  
    return 0;  
}
```

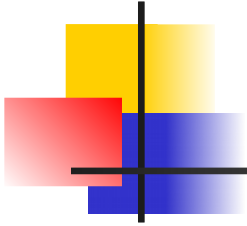
Instance an object with  
default parameters



# Assignment 1

---

- Create a class Rectangle with attributes
  - length and width are integer , and those default value are 1
  - FillCharacter, the specified character that will be used to draw the rectangle.
- Provide these member functions
  - **set** and **get** functions for the length and width attributes
    - The **set** functions should verify that length and width are larger than 1 and less than or equal to 20



# Assignment 1

---

- Two functions that calculate the perimeter and the area of the rectangle.
- **draw** function that displays the rectangle with FillCharacter
- **setFillCharacter** function to specify the character that will be used to draw the rectangle.

```
*****
*                                     *
*                                     *
*                                     *
*****
```