Structure and Class in C++

Sachin Dev
CSE,BUET

Structures

```
struct student{
    string name;
    int marks;
};
```

Structures

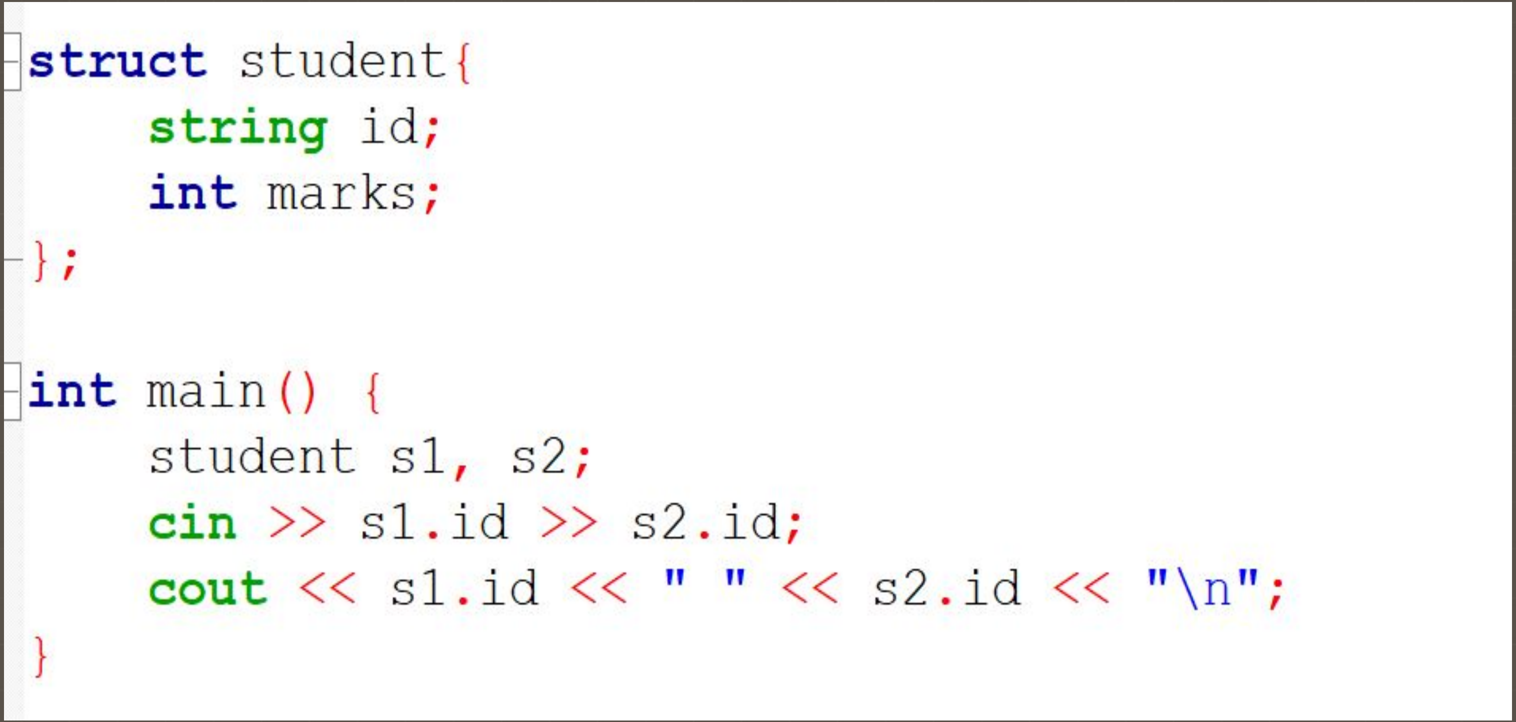
A composite data-type maker.

Suppose you want to have a new data-type named student

It will have two variable one is a string type variable called name, and other is marks

Structure allows to bind one or more data type together and make a complex variable.

Also allows to make multiple instances of those variables.



```cpp
struct student{
    string id;
    int marks;
};

int main() {
    student s1, s2;
    cin >> s1.id >> s2.id;
    cout << s1.id << " " << s2.id << "\n";
}
```

Declaration style as before. But to access one of the attributes of the variable we need to use a dot(.) symbol.

Like the id of student1 can be accessed by student1.id

A better form of structure is class

It allows many facilities regarding private, public, protected variables and methods etc.

Declaration of class looks something like this

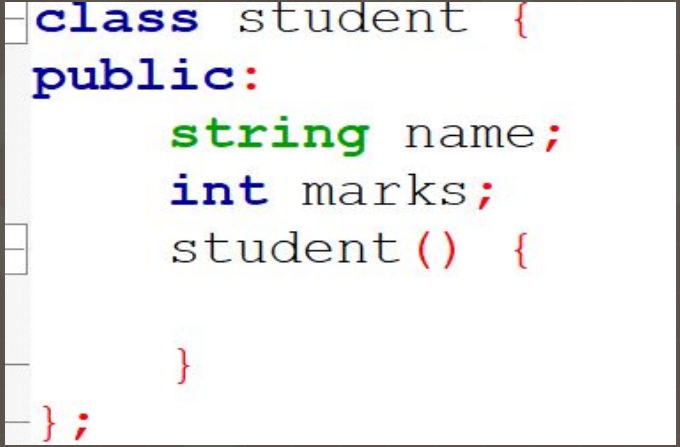
```
class classname {
        /// your code here

};
```

To create an object of a class you need to write the class name and then object name of your choice.

a sample declaration: *classname* var1;

Use of public attributes.

The methods or variables you want to access from outside of the class you can use public keyword for this.



```
class student {
public:
        string name;
        int marks;
        student() {

        }
};
```

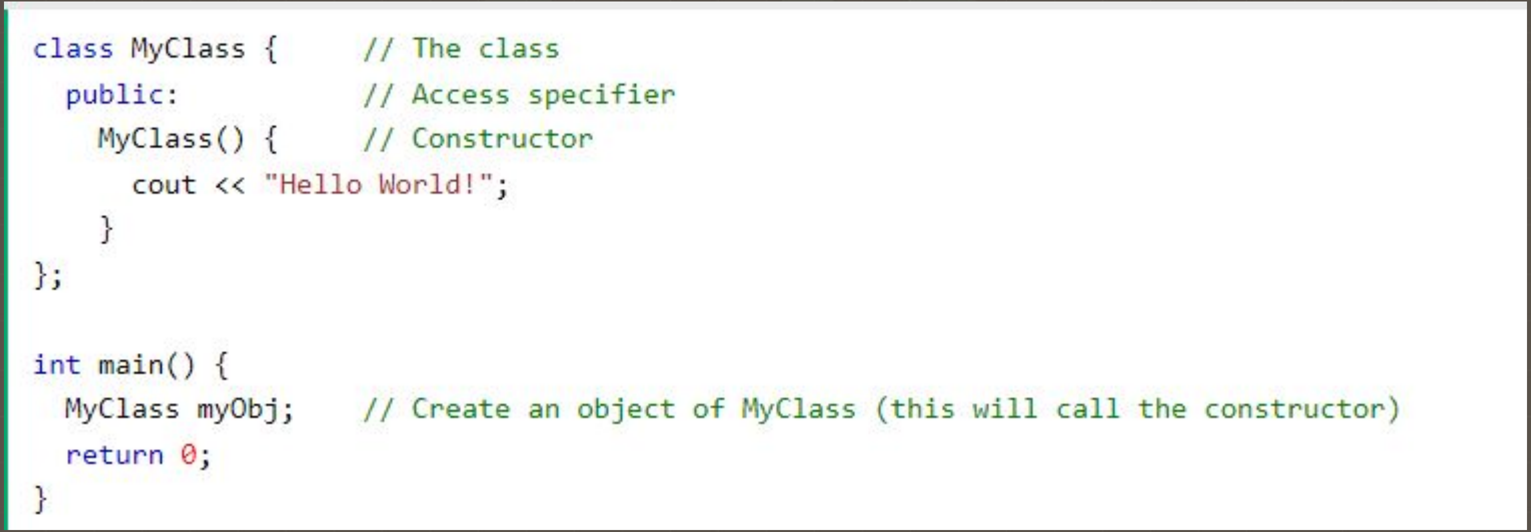
The methods or variables you don't want to use from outside of the class you can mark them as private attributes using keyword private.

private:

Constructors

A constructor in C++ is a special method that is automatically called when an object of a class is created.

To create a constructor, use the same name as the class, followed by parentheses `()`:

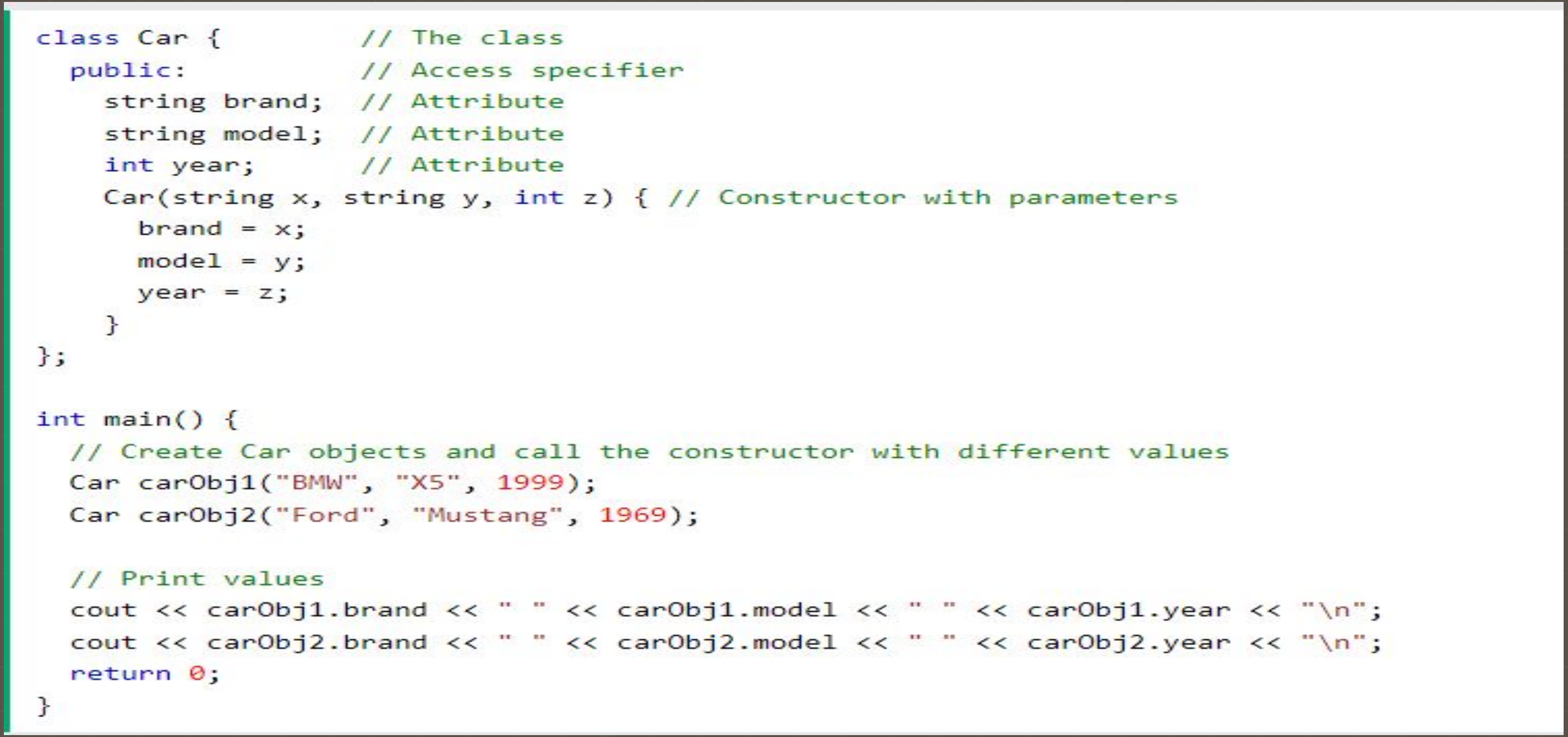


```
class MyClass {     // The class
  public:           // Access specifier
    MyClass() {     // Constructor
      cout << "Hello World!";
    }
};

int main() {
  MyClass myObj;    // Create an object of MyClass (this will call the constructor)
  return 0;
}
```

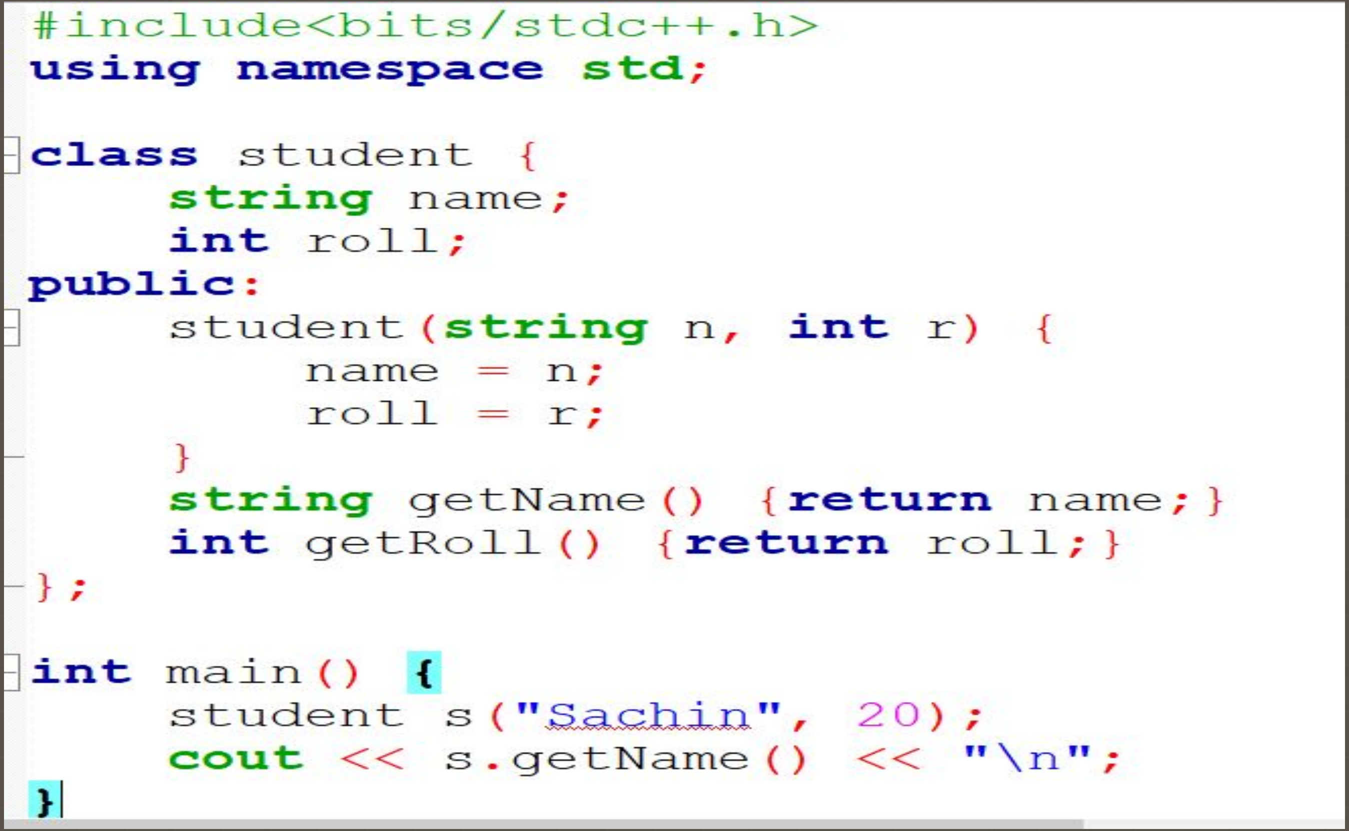
Constructor Parameters

Constructors can also take parameters (just like regular functions), which can be useful for setting initial values for attributes.

The following class have `brand`, `model` and `year` attributes, and a constructor with different parameters. Inside the constructor we set the attributes equal to the constructor parameters (`brand=x`, etc). When we call the constructor (by creating an object of the class), we pass parameters to the constructor, which will set the value of the corresponding attributes to the same:



```cpp
class Car {         // The class
  public:           // Access specifier
    string brand;  // Attribute
    string model;  // Attribute
    int year;      // Attribute
    Car(string x, string y, int z) { // Constructor with parameters
      brand = x;
      model = y;
      year = z;
    }
};

int main() {
  // Create Car objects and call the constructor with different values
  Car carObj1("BMW", "X5", 1999);
  Car carObj2("Ford", "Mustang", 1969);

  // Print values
  cout << carObj1.brand << " " << carObj1.model << " " << carObj1.year << "\n";
  cout << carObj2.brand << " " << carObj2.model << " " << carObj2.year << "\n";
  return 0;
}
```



```cpp
#include<bits/stdc++.h>
using namespace std;

class student {
    string name;
    int roll;
public:
    student(string n, int r) {
        name = n;
        roll = r;
    }
    string getName() {return name;}
    int getRoll() {return roll;}
};

int main() {
    student s("Sachin", 20);
    cout << s.getName() << "\n";
}
```

What would happen if the constructor was declared before public: keyword?

What would happen if we wanted to get the value of name by using .name?

What do we need to do if we want to set name/roll outside of the constructor?

One task:

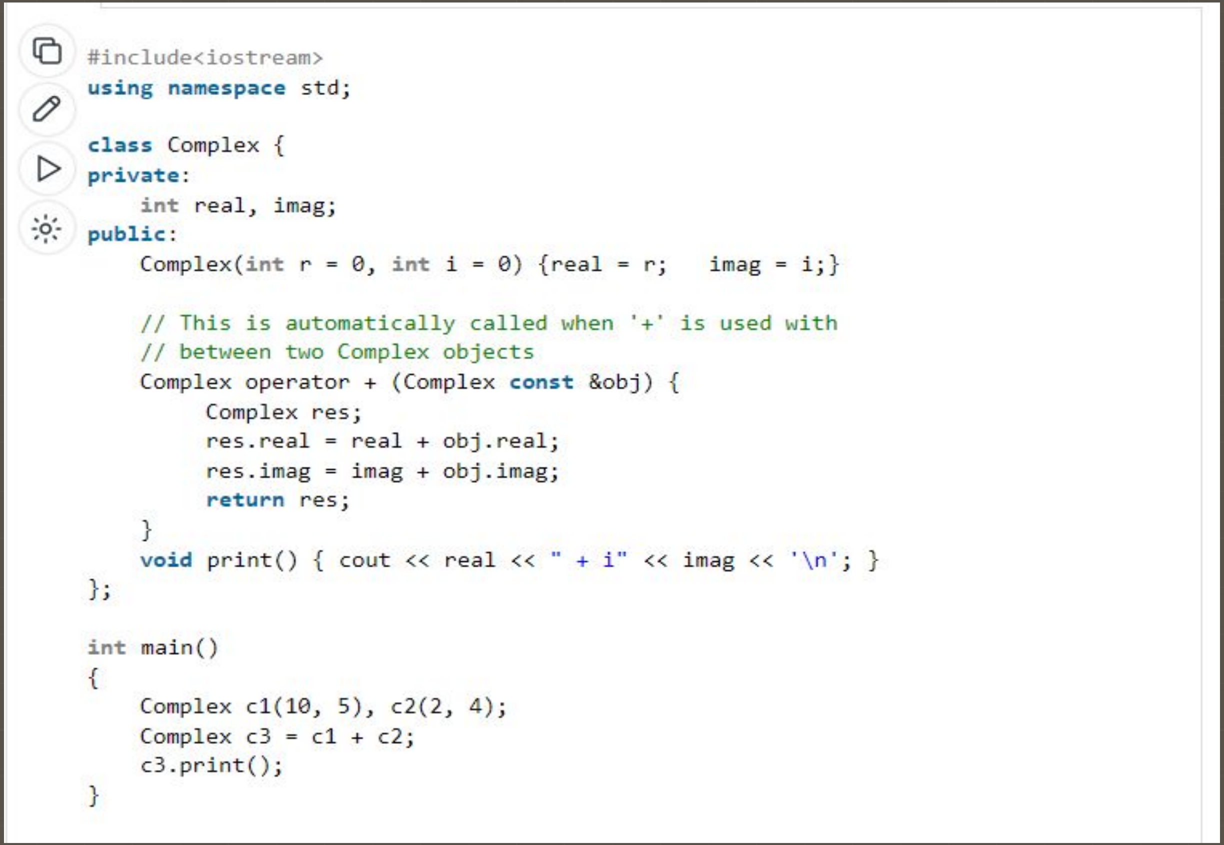
You have two complex numbers and you have to add them

sounds simple?

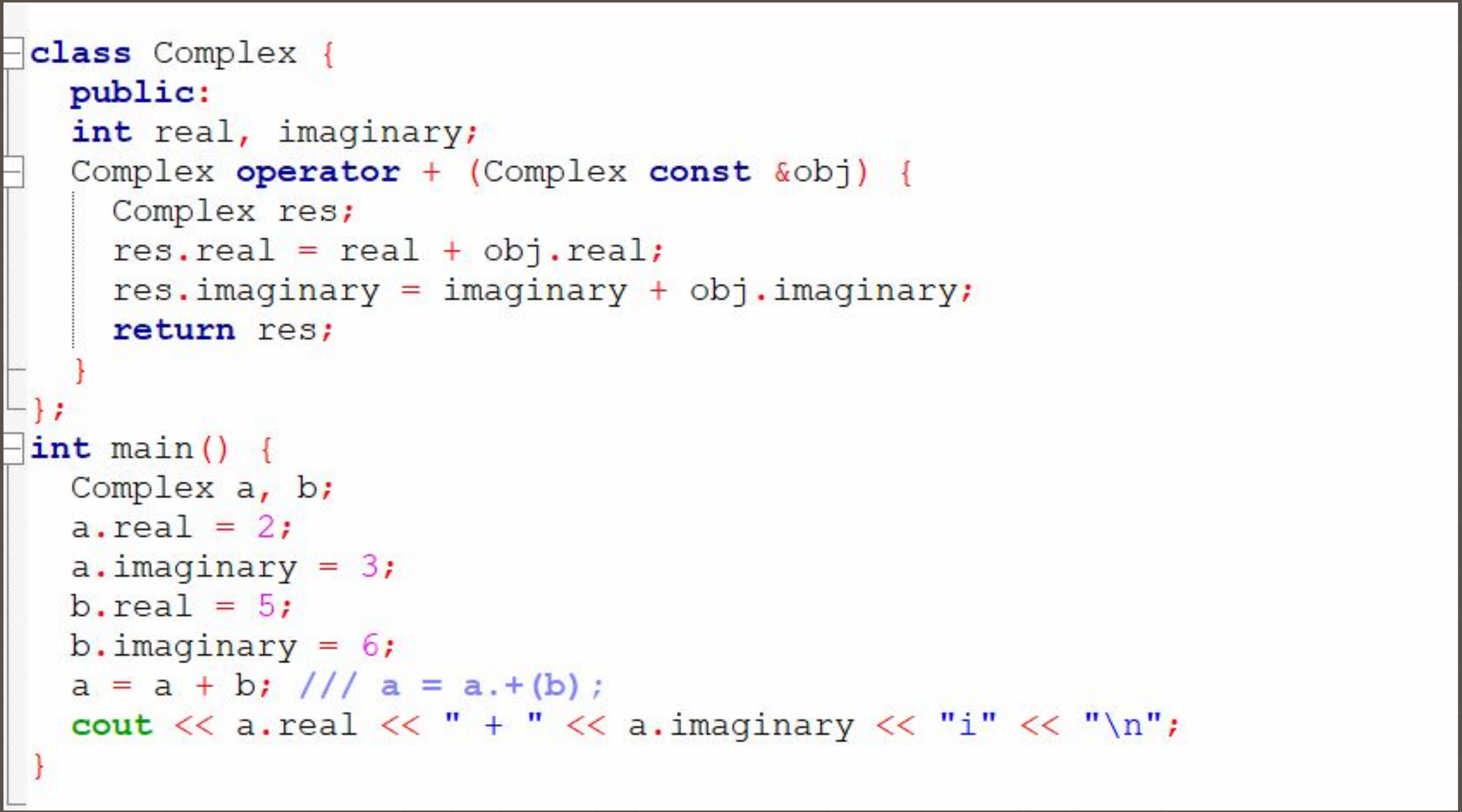
Let complex number is a class and you are given two of its instances.

You basically want to do something so that complex_number1 + complex_number2 gives you the result of addition that means the summation of the two complex number.

Operator overloading????



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

class Complex {
private:
    int real, imag;
public:
    Complex(int r = 0, int i = 0) {real = r;   imag = i;}

    // This is automatically called when '+' is used with
    // between two Complex objects
    Complex operator + (Complex const &obj) {
         Complex res;
         res.real = real + obj.real;
         res.imag = imag + obj.imag;
         return res;
    }
    void print() { cout << real << " + i" << imag << '\n'; }
};

int main()
{
    Complex c1(10, 5), c2(2, 4);
    Complex c3 = c1 + c2;
    c3.print();
}
```



```
class Complex {
  public:
  int real, imaginary;
  Complex operator + (Complex const &obj) {
    Complex res;
    res.real = real + obj.real;
    res.imaginary = imaginary + obj.imaginary;
    return res;
  }
};
int main() {
  Complex a, b;
  a.real = 2;
  a.imaginary = 3;
  b.real = 5;
  b.imaginary = 6;
  a = a + b; /// a = a.+(b);
  cout << a.real << " + " << a.imaginary << "i" << "\n";
}
```

https://ideone.com/0wqKY8

what will this code do?

Forming a linked list?

Templates in C++

https://www.geeksforgeeks.org/templates-cpp/

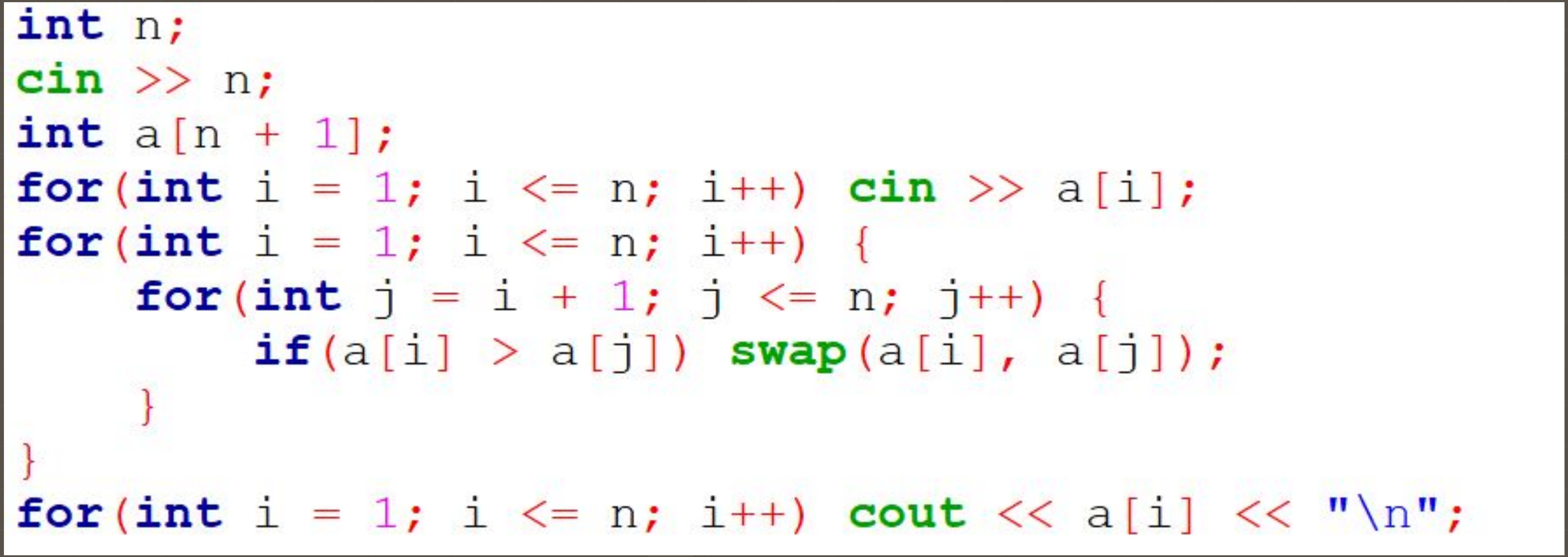
More explanation in class

Sorting: Placing the variables of an array or vector ascending or descending depending on our choice

Suppose arr[] = {5, 4,1, 2, 3}

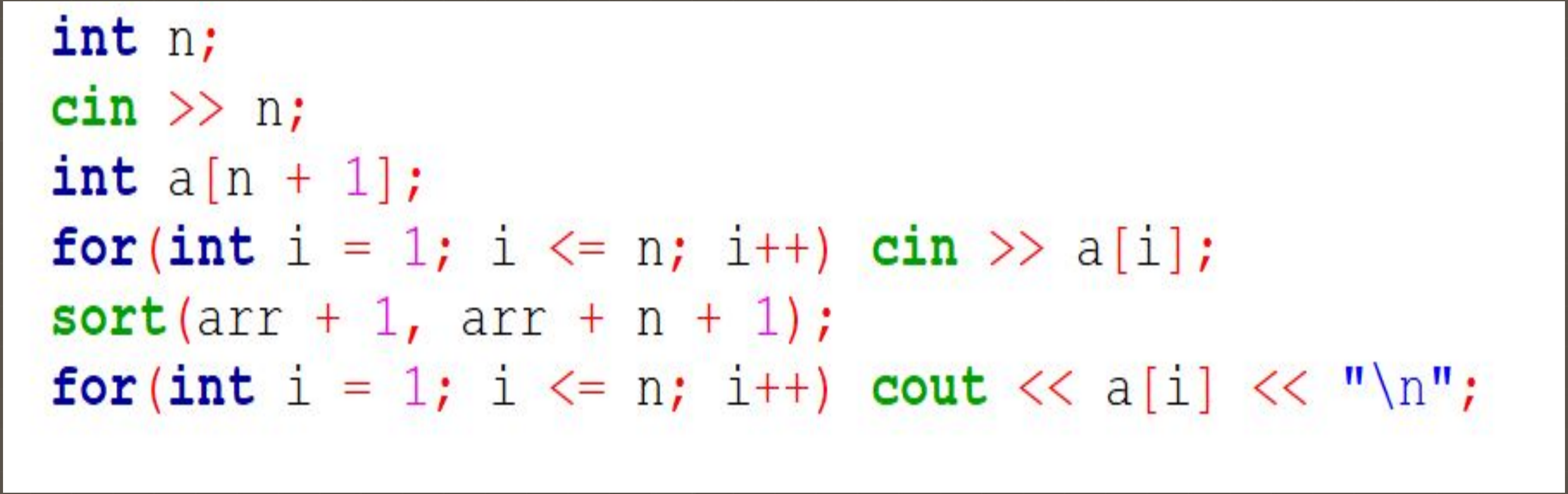
if we want to sort it in ascending order the array would be arr[] = {1, 2, 3, 4, 5}

Bubble sort:



```
int n;
cin >> n;
int a[n + 1];
for(int i = 1; i <= n; i++) cin >> a[i];
for(int i = 1; i <= n; i++) {
    for(int j = i + 1; j <= n; j++) {
        if(a[i] > a[j]) swap(a[i], a[j]);
    }
}
for(int i = 1; i <= n; i++) cout << a[i] << "\n";
```

STL sort :



```
int n;
cin >> n;
int a[n + 1];
for(int i = 1; i <= n; i++) cin >> a[i];
sort(arr + 1, arr + n + 1);
for(int i = 1; i <= n; i++) cout << a[i] << "\n";
```