Recitation Class 10 for VG101

Date: 2012 / 12 / 03

Wang Qian

C++ V.S. C

- Library:

C	C++
	iostream
	fstream
	string
math	cmath
string	cstring

- Do not forget: using namespace std;

C++ V.S. C

- Data type:
- Type casting:
  - More like a function.
  - double pi = 3.14;
  - int a = int(pi);
- string:
  - #include <string>
  - Operators: = and +
- bool:
  - Only take a memory of 1 byte.
  - Have two values: true and false.

C++ V.S. C

- Input and output:
  - cin >> var >> var;
  - cout << var << var;
  - getline(cin,str);   (#include <string>)
  - e.g.

```
printf("The %d-th Fibonacci Number is %d\n",6,8);
cout << "The “ << 6
        << "-th Fibonacci Number is “
        << 8 << endl;
```

C++ V.S. C

- Pass by reference
- Recall how we design the swap function in C.
- In C++, we do it in this way:
  - swap(x,y);
  - void swap(int &x,int &y)

```
{
        int t = x;
        x = y;
        y = t;
}
```

POP V.S. OOP

- POP:
  - Procedure Oriented Programing
  - More like our thinking strategy
- OOP:
  - Object Oriented Programming
  - More related to the real world
  - It is more efficient for complicated programing
  - Features:
    - Abstractness
    - Data Hiding and Encapsulation
    - Reusability
    - Polymorphism
    - Inheritance

POP V.S. OOP

- Let’s focus on the following example:
  - Suppose that there are several students taking the course VG101. The final grade will depend on assignments, labs and exams. Of course, they have different weights. We want to use a program to assist us at the end of the semester with dealing with the data.
  - Now, Let’s have a look at the two different thinking strategies, POP and OOP.
  - Notice that we are talking about the thinking strategy instead of programming itself!

POP

- What procedures are required in this task?
  - Get the total number of students.
  - Get the name of the students.
  - Get the ID of the students.
  - Get the grade for labs.
  - Get the grade for assignments.
  - Get the grade for exams.
  - Get the weights for labs, assignments and exams.
  - Decide the final grade.
  - Decide the rank of the students according to their final grade.

POP

- How to implement these procedures?
- We can use several function.
  - void getTot(int *n);
  - void getName(char *name, const int n);
  - void getID(char *id, const int n);
  - void getGrade(const int idx, int *item);
  - void getWeight(int *w);
  - void calcFinal(int *fnl, const int *w, const int n);
  - void sort(struct Student *stu, const int n);

POP

- What kind of data structure is preferred?
  - We may use a structure.
  - struct Class {

```
  int labW,assignW,midW,finalW;
};
```

  - struct Student {

```
      char *name, * id;
      int lab[10],assignment[10],mida,midb,final;
      int grade;
};
```

OOP

- In Object Oriented Programming, we consider the simple objects first, then to the more specific organization, such as functions.
- This is called “bottom-up programming”.
- What objects are required in this task?
  - Course and student.
- What kind of properties do they have?
  - Course: name, students, weights.
  - Student: name, id, grades.

OOP

- How to implement these objects?
- Let's consider the class Student first.

  -
    ```
    class Student {
          string name,id;
          int lab[3],assign[3],mid,final;
          double grade;
    public:
          Student() {};            // constructor
          ~Student() {};           // destructor
    };
    ```
  - We still need some functions.

OOP

- What about the class Course?
  - class Course {

```
        string name;
        int tot;
        double labW,assignW,midW,finalW;
        Student *stu;
public:
        Course() {};
        ~Course() {};
};
```

  - Of course, we still need some functions, too.
  - It is better to put the prototype in a header file, and the definition in a separate source file. (Not required in this course)

OOP

- The complete version can be found on SAKAI.
- Key words about the data hiding:
- private:
  - Never accessible to other functions or classes.
- public:
  - Accessible to other functions or classes.
- protected:
  - Will be introduced in the future (if needed).

OOP

- Calculation is not interfered by the user.
  - E.g. sortGrade();
- Interactive mode between user and computer
  - Initialization
  - E.g. vg101.setCourse();
  - Update
  - E.g. vg101.updateData();
  - Report
  - E.g. vg101.getData();

OOP

- Data hiding still works for member functions.
  - class Course {

```
        …
        void sortGrade();
    public:
        void setCourse(void);
        void updateData(void);
        void getData(void);
    };
```

- Prototype is similar to what we saw before.

OOP

- Use "::" to show the corresponding class.
  - E.g.

```
void Course::updateData(void)
void Student::setStudent(void)
```

- In the same function, we can use the unqualified name.
- Also, we can use the private member functions in it.
  - E.g.

```
void Course::getData(void) {
        sortGrade();
        ……
}
```

Inheritance

- Today, only the concept will be introduced.
- bass class v.s. derived class
- E.g.
  - ElementaryMember:
    - 10% discount
    - One membership point for consuming one RMB.
    - Free drinks.
  - AdvancedMember:
    - All the rights for elementary member.
    - Free snacks.
    - Physical fitness test once a month.

Polymorphism

- Still, only the concept will be introduced.
- Polymorphic member function method.
- E.g.
  - ElementaryMember:
    - 10% discount
    - One membership point for consuming one RMB.
    - Free drinks.
  - AdvancedMember:
    - 20% discount
    - Three membership points for consuming one RMB.
    - Free drinks.
    - Free snacks.
    - Physical fitness test once a month.