开始写C++程序

2021/02

Why learning C++

- Structural programming language
  - Only include 3 syntactic structures
    - Sequential, branching, loop, (function call)
  - But is functionally complete
- How to make programming easier?
  - A relatively direct mapping between human mind and program code
    - How? Enable you to describe objects in your program
  - A better way to manage data and functions in your code
    - How? Enable your to write modular code

Compound data type 复合数据类型

- Need to define an object before using it.
- How to define an object? By describing its
  - Useful data (properties/values)
  - The functionalities it provides (methods/ functions)
  - In an abstract data type
- A class (类) contains the definition of an object or similar objects
  - The module (container) for all definitions

Class 类 and object 对象

- Write a class in C++

```
1 class Student
2 {
3 public:
4       char name[20];
5       int id;
6       double age; // = 18.5
7 };
```

Class and object

- Define an object (a variable) of the class

```
 8 Student s;
 9 strcpy(s.name, "Cong Liu");
10 s.id = 12340001;
11 s.age = 18.5;
```

Class and object

- An object use a consecutive block of memory
  - The size is the sum of the sizes of the member variables
  - The member variables defined first locates in the front of this memory
  - What is the size of an object of the following class

```
1 class Student
2 {
3 public:
4       char name[20];
5       int id;
6       double age;
7 };
```

Class and object

- Assignment of an object as a whole

```
1  void some_function(Student student) {
2      ...
3  }
4
5  void some_function2(Student & student2) { //传tom的地址
6      student2.age += 1; // will change tom.age
7  }
8
9  int main() {
10     Student tom;
11     strcpy(tom.name, "Tom");
12     tom.id = 12345678;
13     tom.age = 18;
14
15     Student copy_of_tom;
16     copy_of_tom = tom; // will copy all member variables
17
18     some_function(tom); // will copy to student
19     some_function2(tom); // will pass the address of tome to student2
20 }
```

Class and object

- Member functions 成员函数
  - As a module, a class function as a container for
    - The objects' member variables
    - And the functions defined specifically for these objects
  - Organize code better by grouping related things together
    - Easier reading and faster searching
  - Next, we will use an example to show how C++ do this

Class and object

- Member
- functions
- C

```
1  struct Student
2  {
3      int id;
4      double age;
5  };
6
7  void read(Student & this_obj) {
8      cin >> this_obj.id;
9      cin >> this_obj.age;
10 }
11
12 void print(const Student * this_obj) {
13     cout << this_obj->id << endl;
14     cout << this_obj->age << endl;
15 }
```

const使得更为安全 防止改变引入的对象

Class and object

- Member
- functions
- C
- C++

```c
struct Student
{
    int id;
    double age;
};

void read(Student & this_obj) {
    cin >> this_obj.id;
    cin >> this_obj.age;
}

void print(const Student * this_obj) {
    cout << this_obj->id << endl;
    cout << this_obj->age << endl;
}
```

```c
class Student // class
{
public:
    int id;
    double age;

    // the first object parameter is hidden
    // which is default as 'Student * this'
    void read() { 省略的参数默认叫this 保留字
        cin >> this->id;
        cin >> this->age;
    }

    // 'const' is for 'const Student * this'
    void print() const {
        cout << this->id << endl;
        cout << this->age << endl;
    }

}; // class contains the functions
```

Class and object

- Calling member functions

```
 1 class Student
 2 {
 3 public:
 4     void read() { ... }
 5     void print() const { ... }
 6 };
 7
 8 class School {
 9 public:
10     void print() const { ... }
11 };
12
13 int main() {
14     Student tom;
15     School sysu;
16     tom.print();
17     sysu.print();
18 }
```

Special member functions

- A flexible way in C++: use a function to initialize objects of a class.
  - This function is call a constructor 构造函数

```
1  class Student
2  {
3  public:
4      char name[20];
5
6      // constructor
7      Student() {
8          strcpy(this->name, "NO_NAME");
9      }
10 }
11
12 int main() {
13     Student some_one, another;
14     cout << some_one.name << endl;
15     cout << another.name << endl;
16 }
```

自动调用

Special member functions

- Similarly, C++ provide destructor 释构函数 to finalize objects

```
 1 class Student
 2 {
 3 public:
 4     Student() {
 5         cout << "constructor" << endl;
 6     }
 7     // destructor
 8     ~Student() {
 9         cout << "destructor" << endl;
10     }
11 }
12
13 int main() {
14     Student some_one; // will call constructor here
15 } // will call destructor here
```

Special member functions

- An example

```
1  class Student
2  {
3  public:
4      char * name;
5      Student() {
6          this->name = new char[20];
7          strcpy(this->name, "NO_NAME");
8      }
9      void assignName(char newName[]) {
10         delete[] this->name;
11         this->name = new char[strlen(newName) + 1];
12         strcpy(this->name, newName);
13     }
14     ~Student() {
15         delete [] this->name;
16     }
17 }
18
19 int main() {
20     Student some_one;
21     some_name.assignName("Mark Zuckerberg");
22 }
```

Operator overriding 操作符重写

- Operators are defined
  - For most basic types
  - But not for objects
- Operator overriding is to define operators
  - Involving any object
  - Using functions
- Operator functions are matched according to
  - The name of the operator
  - The types of the operands

操作符重载

Operator overriding

- Example

```
1  class Student
2  {
3  public:
4      int gpa;
5  };
6
7  bool operator > (Student & s1, Student & s2) {
8      return s1.gpa > s2.gpa;
9  }
10
11 int main() {
12     Student s1;
13     Student s2;
14     s1.gpa = 100;
15     s2.gpa = 90;
16     cout << (s1 > s2);
17 }
```

操作符函数

Operator overriding

- Example 2:

```
1  class Student
2  {
3  public:
4       int gpa;
5
6       bool operator > (Student & s2) {
7            return this->gpa > s2.gpa;
8       }
9
10 };
11
12 int main() {
13      Student s1;
14      Student s2;
15      s1.gpa = 100;
16      s2.gpa = 90;
17      cout << (s1 > s2);
18 }
19
```

课后问题

- 怎样的函数作为对象函数
- 把一个普通函数变为对象函数需要那些改动
- 对象函数与普通函数的区别
- 构造函数和普通对象函数的区别
- 构造函数和释构函数的异同