vg101: Introduction to Computer Programming

RC 8

CHEN Xiwen

2019/7/19 (FRI)

Welcome to C++

- Almost all the aspects of C are preserved.
- Data type: `bool` , `string`.
- Headers: C++ headers.
- Use `namespace` to avoid conflicts of function names. (e.g., `using namespace std;`)

  1. Print without `namespace`:

     ```
     #include <iostream>
     int main() {
         std::cout << "Print without namespace.\n";
         return 0;
     }
     ```

  2. Print with `namespace`

     ```
     #include <iostream>
     using namespace std;
     int main() {
         cout << "Print with namespace.\n";
         return 0;
     }
     ```

- Object oriented language.

Special Features

- **Strings**

  ```
  string a = "Hello, ";
  string b = "world!";
  string c = a + b;
  ```

  See *cppreference* for more available functions.

- **Dynamic memory**

  ```
  int* a = new int;
  int* b = new int[10];

  delete a;
  delete[] b;
  ```

  The way of deleting the memory should match the way of allocating the memory.

- **I/O**

  1. Standard input/output

     `cout` and `cin`

     ```cpp
     #include <iostream>
     using namespace std;
     int main() {
         int a, b;
         cin >> a >> b;
         cout << "Input: " << a << ", " << b << endl;
         return 0;
     }
     ```

  2. File I/O (`fstream`) `[demo_fstream.cpp]`

     - Open a file stream for:
       1. reading: `ifstream f_i("input.txt");`
       2. writing: `ofstream f_o("output.txt");`
     - Read data from input stream:
       1. `operator>>` : extract formatted data
       2. `int_type get()` : extract characters
       3. `basic_istream& read(char_type* s, std::streamsize count)` : extract blocks of characters
       4. `getline(char_type* s, std::streamsize count, char_type delim)` : extract characters until `delim` is met
     - Write data to output stream:
       1. `operator<<` : insert formatted data
       2. `basic_ostream& put(char_type ch)` : insert a character
       3. `basic_ostream& write(const char_type* s, std::streamsize count)` : insert blocks of characters
       4. `std::basic_ostream& flush()` : write uncommited changes to the underlying output sequence

  ```cpp
  int main() {
      ifstream f_i("input.txt");
      ofstream f_o("output.txt");

      string buffer;
      char str[20];
      f_i.getline(str, 20 * sizeof(char), '|');
      cout << str << endl;
      while (f_i >> buffer) {
          f_o << buffer << endl;
          cout << "Written: " << buffer << endl;
      }
      return 0;
  }
  ```

- **Overloading**

1. Operator `./demo/[demo_operator.cpp]`

   Comparing string lengths.

   ```cpp
   bool operator>(string a, string b) {
       return a.length() > b.length();
   }
   ```

2. Function

   ```cpp
   int add(int a, int b) {
       cout << "Integer addition.\n";
       return a + b;
   }
   double add(double a, double b) {
       cout << "Double addition.\n";
       return a + b;
   }
   int main() {
       int a = 1,
           b = 3;
       double c = 1.2,
              d = 3.4;
       cout << add(a, b) << endl << endl;
       cout << add(c, d) << endl << endl;
       cout << add(a, c) << endl << endl;
       cout << add(c, a) << endl << endl;
       return 0;
   }
   ```

- **Class**

  > Class
  >
  > |------- Attributes
  >
  > |------- Methods
  >
  > Instance: a realization of a class: call class functions (and access attributes)

- **Object oriented programming**

  1. Procedural programming: complete tasks following a procedure sequentially in a program
  2. Object oriented programming: render some data and functions to different objects, each object manages its own data and complete tasks

  e.g., In the *OneCard* game, how does a player play a card?

  1. Procedural: `void play_card(player_t* player, game_t* game_state) {};`
  2. Object oriented: `void Player::play_card(game_t* game_state) {};`

Class

`./demo/[demo_cls.cpp, dmeo_inheritence.cpp]`

- Declaring a class: (in `.h` files)

  ```cpp
  class ClassName {
  ```

```cpp

private:
    // private data;
    // private functions;
public:
    // constructor;
    ClassName();
    // destructor;
    // called automatically at the end of the object lifetime;
    ~ClassName();
    // public data;
    // public functions;
    void public_f1(args);
    void public_f2(args);

};
```

- Implementing a class: (in `.cpp` files that include the corresponding `.h` files)

```cpp
ClassName::ClassName() {
    // construct a class;
}
ClassName::~ClassName() {
    // destructor;
    // free dynamically allocated memories if necessary;
}
void ClassName::public_f1() {
    // do something;
}
void ClassName::public_f2() {
    // do something;
}
```

- Use a class (instantiation and calling member functions)

```cpp
int main() {
    // constructor is called when an instance is declared;
    ClassName class_instance;
    ClassName class_instance2(3);
    class_instance.public_f1();
    class_instance.public_f2();
    return 0;
}
```

- Inheritance
  1. Field types
     - Public
     - Private
     - Protected

Access	public	protected	private
Same Class	T	T	T
Derived Class	T	T	F
Other Classes	T	F	F

2. Syntax

```cpp
class DerivedClass : public BaseClass1, public BaseClass2 ... {
    // attributes and methods;
};
```

3. Polymorphism: classes `C1`, `C2`, `C3` are all inherited from the base class `C`, but can be extended in terms of attributes and methods.
4. `virtual` key word: override member functions even there is no compile-time information about the actual type of the class.

Exercises

1. Implement the array data type as a class in C++, which has the following member functions. `./ex/[array.h, array.cpp, main.cpp]`

```cpp
class Array {
    protected:
        int _card;
        int _size;
        int* _arr;
    public:
        Array(int init_size = 10);
        ~Array();
        int get_size();
        int get_card();
        // return true if not modified the size;
        bool insert(int elt);
        // return the number of elements removed;
        // remove all the elements that match;
        int remove(int elt);
        void print();
}
```

2. A set is a special array that does not contain any duplicated element. Implement the set data type in C++ as a derived class of array.
3. What is the relationship between an ordered array and a set? Implement the ordered array data type.

General Suggestions after Midterm 2

1. Pay more attention to the lecture slides, especially the questions at the end of each chapter.
2. In part A of the exam, make wise use of time. Do not spend too much time on modifying your language when a few key words are sufficient.
3. Practice yourself on homework, labs and project.

4. Use JOJ.