面向对象与流

陈云帆

数据结构与算法 补充内容——C++特性简介

http://www.jpk.pku.edu.cn/pkujpk/course/sjjg

补充内容 面向对象与流

- 类与对象
  - 类的概念及基本语法
  - 默认函数——构造、析构、复制构造、赋值与取址
  - 特殊成员——this指针
  - 模板类
- 流
  - 标准输入输出流
  - 流操纵算子
  - 文件输入输出流

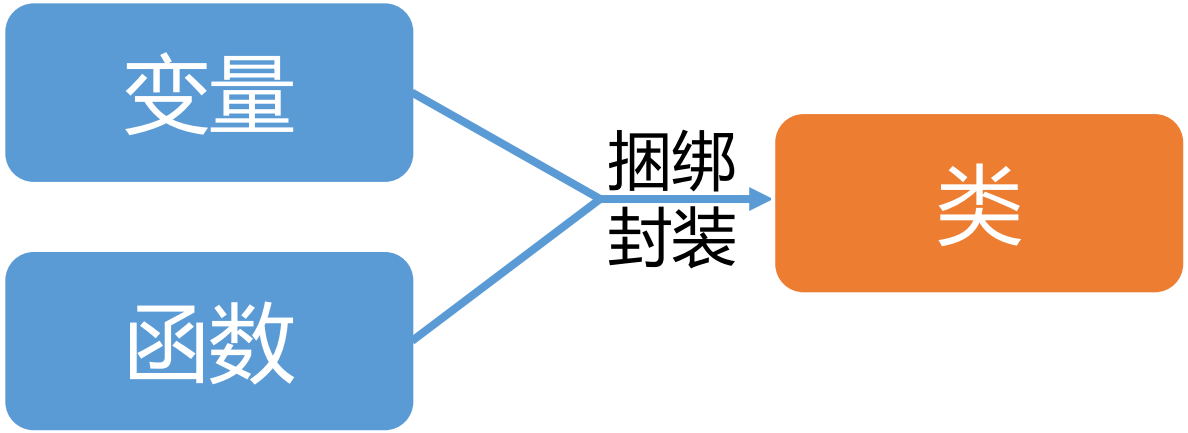
类的概念

- 一个类别、数据类型——万物皆对象

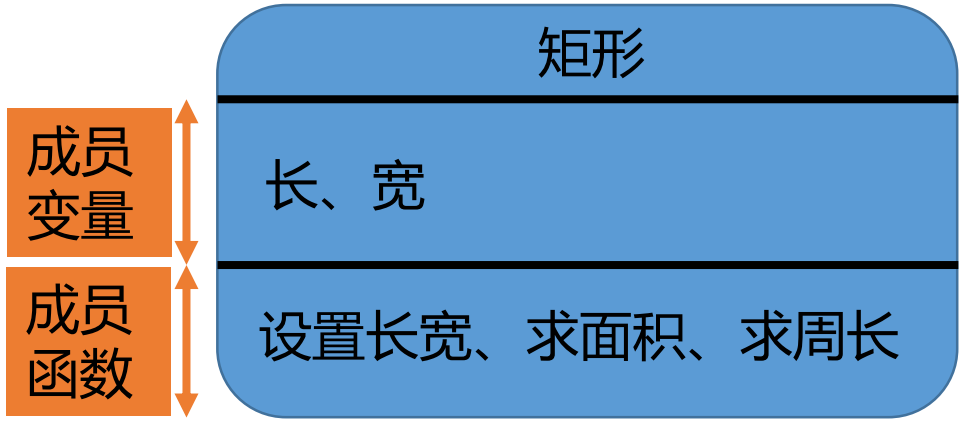
- 属性：亮度、电量、运营商……
- 方法：开关机、调整亮度、发送短信……

类的概念

- 共同特点（变量）→构成数据结构
- 归纳行为（函数）→操作数据结构（抽象）



定义类



- 看上去像 “带函数的结构体”

```
class Rectangle{
        public:
                int w, h;
                int Area() {
                        return w * h;
                }
                int Perimeter(){
                        return 2 * ( w + h);
                }
                void Init( int w_,int h_ ) {
                        w = w_;  h = h_;
                }
};//必须有分号
```

使用类

```
int main( ) {
        int w,h;
        Rectangle  r;  // r是一个对象
        cin >> w >> h;
        r.Init( w,h);
        cout << r.Area() << endl << r. Perimeter();
        return 0;
}
```

使用类

Rectangle r1, r2;

对象名.成员名

r1.w = 5; r2.Init(5,4);

指针->成员名

Rectangle * p1=&r1, p2=&r2;

p1->w = 5; p2->Init(5,4);

引用名.成员名

CRectangle & rr = r2;

rr.w = 5; rr.Init(5,4);

声明、定义分离

```
class Rectangle{
public:
        int w,h;
        int Area();     //成员函数仅在此处声明
        int Perimeter() ;
        void Init( int w_,int h_ );
};
int Rectangle::Area()  {   return w * h;      }
int Rectangle::Perimeter() {  return 2 * ( w + h);  }
void Rectangle::Init( int w_,int h_ ) {  w = w_;  h = h_;}
```

访问权限

private	私有成员，只能在成员函数中访问
public	公有成员，可以在任何地方访问
protected	保护成员，暂不介绍

```
class className {
    private:
      私有属性和函数
    public:
      公有属性和函数
    protected:
      保护属性和函数
};
```

- 在类的成员函数内部，能够访问：
  当前对象的全部属性、函数；
  同类其它对象的全部属性、函数。

参考文献

- 北京大学 郭炜、刘家瑛《程序设计实习》
  https://www.coursera.org/course/pkupop
- Prata, S. (2011). C++ primer plus. Addison-Wesley Professional.

数据结构与算法

谢谢聆听

国家精品课“数据结构与算法”
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

张铭，王腾蛟，赵海燕
高等教育出版社，2008. 6。“十一五”国家级规划教材

面向对象与流

陈云帆

数据结构与算法 补充内容——C++特性简介

http://www.jpk.pku.edu.cn/pkujpk/course/sjjg

补充内容 面向对象与流

- 类与对象
  - 类的概念及基本语法
  - 默认函数——构造、析构、复制构造、赋值与取址
  - 特殊成员——this指针
  - 模板类
- 流
  - 标准输入输出流
  - 流操纵算子
  - 文件输入输出流

默认方法

```
class cellphone{
```

- 每个物体都有诞生和消亡——构造函数与析构函数

```
public:
    cellphone();
    ~cellphone();
```

- 可以被复制

```
    cellphone(const cellphone&);
    cellphone & operator=(const cellphone&);
```

- 可以取地址

```
    cellphone * operator&();
    const cellphone* operator&() const;
private:
    int electricity; };
```

特殊成员——this指针

并非对象的成员，是常量指针
每个对象可以使用 this 指针访问自己的地址
非 static 成员函数调用时，this 指针为隐式参数

用途：防止自赋值、返回以连续调用

特殊成员——this指针

```
class Complex {
        float real, imag;
        public:
                Complex * ReturnAddress ( ) {
                        return this;
                }//  c.ReturnAddress ()等效于 & c
                float ReturnReal() {
                        return this -> real;//等效于return real
                }
};
```

参考文献

- 北京大学 郭炜、刘家瑛《程序设计实习》
  https://www.coursera.org/course/pkupop
- Prata, S. (2011). C++ primer plus. Addison-Wesley Professional.

数据结构与算法

谢谢聆听

国家精品课“数据结构与算法”
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

张铭，王腾蛟，赵海燕
高等教育出版社，2008. 6。“十一五”国家级规划教材

面向对象与流

陈云帆

数据结构与算法 补充内容——C++特性简介

http://www.jpk.pku.edu.cn/pkujpk/course/sjjg

补充内容 面向对象与流

- 类与对象
  - 类的概念及基本语法
  - 默认函数——构造、析构、复制构造、赋值与取址
  - 特殊成员——this指针
  - 函数模板与类模板
- 流
  - 标准输入输出流
  - 流操纵算子
  - 文件输入输出流

函数模板

实际问题中的需要：
对不同类型数据可用的排序函数 sort

```
template<class T>
return-type sort(...T...)
```

函数模板

一个实际的输出函数：

```
template<class T>
void print( const T array[], int size){
    int i;
    for ( i =0; i<size; i++) cout<<array[i];
    return;
}
```

int a[10]; print(a,10);

函数模板

一个实际的输出函数：

```
template<class T1, class T2>
void print(T1 arg1, T2 arg2 , string s, int k)
    { cout<<arg1<<s<<arg2<<k<<endl;  return; }
```

类模板

- 为了多快好省地定义出一批相似的类,可以定义类模板,然后由类模板生成不同的类

- 数组是一种常见的数据类型，元素可以是：
  - 整数
  - 字符串
  - ……

类模板：在定义类的时候给它一个/多个参数,这个/些参数表示不同的数据类型。在调用类模板时，指定参数，由编译系统根据参数提供的数据类型自动产生相应的模板类。

类模板的定义

```
template <class T>//类模板的首部，声明类模板的参数
class Carray{
    T *ptrElement;
    int size;
public:
    Carray(int length);
    ~ Carray();
    int len();
    void setElement(T arg, int index);
    T getElement(int index);
};
```

使用类模板声明对象

```
Carray<int> arrayInt(50), *ptrArrayInt;
    //创建一个元素类型为int的Carray模板类，并声明该模板类
      的一个对象、以及一个指针。
```

不同模板参数产生的模板类，不是同一个类

参考文献

- 北京大学 郭炜、刘家瑛《程序设计实习》
  https://www.coursera.org/course/pkupop
- Prata, S. (2011). C++ primer plus. Addison-Wesley Professional.

数据结构与算法

谢谢聆听

国家精品课“数据结构与算法”
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

张铭，王腾蛟，赵海燕
高等教育出版社，2008. 6。“十一五”国家级规划教材

面向对象与流

陈云帆

数据结构与算法 补充内容——C++特性简介

http://www.jpk.pku.edu.cn/pkujpk/course/sjjg

补充内容 面向对象与流

- 类与对象
  - 类的概念及基本语法
  - 默认函数——构造、析构、复制构造、赋值与取址
  - 特殊成员——this指针
  - 模板类
- 流
  - 标准输入输出流
  - 流操纵算子
  - 文件输入输出流

标准输入流

cin >> x;

- 读入整型数时以第一个非数字为终结
- 读入字符串时以第一个空格、tab 或换行符为终结

```
cin.getline (str, len, ch);      // 读入一个字符串
                                 // ch被从流中提出，但不存入str
ch = cin.get();                  // 读入一个单独的字符
cin.ignore (len, ch);            // 忽略一串字符, ch同上
```

标准输入流

```
cin >> x;
```

判断读入结束：

```
int x;
while(cin>>x){
      .....
}
return 0;
```

键盘读入时用ctrl-z结束，文件读入时读到文件末尾

标准输出流

```
cout << y;
```

cout 输出到标准设备
cerr 输出错误信息
clog 输出错误日志

标准输出流

```
cout << y;
```

- 输出一个字符：

  cout .put('A').put('a');

流操纵算子

`cout << y;`

- 整型数

```
int n = 10;
cout << n << endl;
cout << hex << n << endl
     << dec << n << endl
     << oct << n << endl;
```

流操纵算子

- 浮点数

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
        double x = 1234567.89,y = 12.34567;
        int n = 1234567;
        int m = 12;
        cout << setprecision(6) << x << endl
                << y << endl << n << endl << m;
}
```

流操纵算子

- 浮点数

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
        double x = 1234567.89,y = 12.34567;
        int n = 1234567;
        int m = 12;
        cout << setiosflags(ios::fixed) <<
                setprecision(6) << x << endl
                << y << endl << n << endl << m;
}
```

流操纵算子

- 设置域宽

```
cin.width(5);
cin >> string;
cout << string << endl;
cin >> string;
cout << string << endl;
```

输入：1234567890

文件输入输出

- 基本操作与cin和cout相同
  - ifstream fin; ofstream fout;
  - fin.open (“input.txt”); fout.open (“output.txt”,ios::out);
  - fin >> …
  - fout << …
- 打开文件选项
  - ios::out　输出到文件, 删除原有内容
  - ios::app　输出到文件, 保留原有内容，总是在尾部添加
  - ios::ate　输出到文件, 保留原有内容，可以在文件任意位置添加

文件输入输出

- 文件指针操作

```
ofstream  fout("a1.out",ios::ate);
long location = fout.tellp();
        //取得写指针的位置
location = 10L;
fout.seekp(location);
        // 将写指针移动到第10个字节处
fout.seekp(location,ios::beg); //从头数location
fout.seekp(location,ios::cur); //从当前位置数location
fout.seekp(location,ios::end); //从尾部数location
```

输入文件指针操作为 tellg, seekg

参考文献

- 北京大学 郭炜、刘家瑛《程序设计实习》
  https://www.coursera.org/course/pkupop
- Prata, S. (2011). C++ primer plus. Addison-Wesley Professional.

数据结构与算法

谢谢聆听

国家精品课“数据结构与算法”
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

张铭，王腾蛟，赵海燕
高等教育出版社，2008. 6。“十一五”国家级规划教材