# 第六章 数组 指针与字符串

6.1 数组

6.2 指针

6.3 动态内存分配

6.4 用vector创建数组对象

6.6 字符串

小结

# 6.1.4 对象数组

- 声明:

  类名　数组名[元素个数]；

  Point　p1[10];

- 访问方法,通过下标访问:

  数组名[下标].成员名

  p1[0].move(1,1);

# 对象数组初始化

- 数组中每一个元素对象被创建时，系统都会调用类构造函数初始化该对象。
- 通过初始化列表赋值。

  例：Point a[2]={Point(1,2),Point(3,4)};
- 如果没有为数组元素指定显式初始值，数组元素便使用默认值初始化（调用缺省构造函数）。

# 例6-3 对象数组应用举例

```
//Point.h
#ifndef _POINT_H
#define _POINT_H

class Point {       //类的定义
public:             //外部接口
  Point();
  Point(int x, int y);
  ~Point();
  void move(int newX,int newY);
  int getX() const { return x; }
  int getY() const { return y; }
  static void showCount();       //静态函数成员
private:                         //私有数据成员
  int x, y;
};
#endif       //_POINT_H
```

# 例6-3 (续)

```
//Point.cpp
#include <iostream>
#include "Point.h"
using namespace std;
Point::Point() {
  x = y = 0;
  cout << "Default Constructor called." << endl;
}
Point::Point(int x, int y) : x(x), y(y) {
  cout << "Constructor called." << endl;
}
Point::~Point() {
  cout << "Destructor called." << endl;
}
void Point::move(int newX,int newY) {
  cout << "Moving the point to (" << newX << ", " << newY << ")" <<
  endl;
  x = newX;
  y = newY;
}
```

# 例6-3 (续)

```
//6-3.cpp
#include "Point.h"
#include <iostream>
using namespace std;

int main() {
  cout << "Entering main..." << endl;
  Point a[2];
  for(int i = 0; i < 2; i++)
      a[i].move(i + 10, i + 20);
  cout << "Exiting main..." << endl;
  return 0;
}
```

# 例6-3 (续)

运行结果:

```
Entering main...
Default Constructor called.
Default Constructor called.
Moving the point to (10, 20)
Moving the point to (11, 21)
Exiting main...
Destructor called.
Destructor called.
```

# 6.2.11 对象指针

- 声明形式

  类名 *对象指针名；

  例: Point a(5,10);

  Piont *ptr;

  ptr=&a;

- 通过指针访问对象成员

  对象指针名->成员名

  ptr->getx();

# 动态申请内存操作符 new

- new 类型名T（初始化参数列表）
- 功能：在程序执行期间，申请用于存放T类型对象的内存空间，并依初值列表赋以初值。
- `Point *ptr1 = new Point;`
- `Point *ptr1 = new Point(1,2);`

# 释放内存操作符delete

- delete 指针p
- 功能：释放指针p所指向的内存。p必须是new操作的返回值。

```
 Point *ptr1 = new Point;
//调用缺省构造函数
 delete ptr1;

 Point *ptr2 = new Point(1,2);
//调用缺省构造函数
 delete ptr1;
```

# 申请和释放动态数组

- 分配：new 类型名T [ 数组长度 ]
  - 数组长度可以是任何表达式，在运行时计算
- 释放：delete[] 数组名p
  - 释放指针p所指向的数组。p必须是用new分配得到的数组首地址。

```
Point *ptr=new Point[2];//创建对象数组
delete[] ptr;        //删除整个对象数组
```

# 用vector创建数组对象

- 为什么需要vector？
  - ✓将动态数组封装，自动创建和删除
  - ✓数组下标越界检查
- vector动态数组对象的定义
  - ✓vector<元素类型> 数组对象名(数组长度);
  - ✓例：vector<int> arr(5)

    建立大小为5的int数组

# vector数组对象的使用

- 对数组元素的引用
  - ✓与普通数组具有相同形式：
  - ✓数组对象名 [ 下标表达式 ]
- 获得数组长度
  - ✓用size函数
  - ✓数组对象名.size()

# 例6-20 vector应用举例

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

//计算数组arr中元素的平均值
double average(const vector<double> &arr)
{
  double sum=0;
  for (int i=0;i<arr.size();i++)
     sum+=arr[i];
  return sum/arr.size();
}
```

# 例6-20 (续)

```
int main()
{
   int n,i;
   cout<<"n = ";
   cin>>n;
   vector<double> arr(n);//创建数组对象
   for (i=0;i<n;i++)
      cin >> arr[i];
   cout <<"Average="<<average(arr)<<endl;
   return 0;
}
```

# 6.6.1 用字符数组存储和处理字符串

- 字符串常量（例："program"）
  - ✓const char *STRING1 = "program";
- 字符串变量
  - ✓char str[8] = { 'p', 'r', 'o', 'g', 'r', 'a', 'm', '\0' };
  - ✓char str[8] = "program";
  - ✓char str[] = "program";

| p | r | o | g | r | a | m | \0 |
|---|---|---|---|---|---|---|---|

# 用字符数组表示字符串的缺点

- 用字符数组表示字符串的缺点

  ✓执行连接、比较等操作，调用库函数，很麻烦

- 解决方法

  ✓使用字符串类string表示字符串

  ✓string实际上是对字符数组操作的封装

# 6.6.2 string类

- 常用构造函数
  - ✓string(); //缺省构造函数，建立一个长度为0的串
  - ✓string(const char *s); //常量初始化string类的对象
  - ✓string(const string& rhs); //拷贝构造函数
- 例：
  - ✓string s1; //建立一个空字符串
  - ✓string s2 = “abc”; //用常量建立一个初值
  - ✓string s3 = s2;//执行拷贝构造函数

# 6.6.2 string类(续)

- 常用操作符
  - s + t　　将串s和t连接成一个新串
  - s = t　　用t更新s
  - s == t　　判断s与t是否相等
  - s != t　　判断s与t是否不等
  - s < t　　判断s是否小于t（按字典顺序比较）
  - s <= t　　判断s是否小于或等于t （按字典顺序比较）
  - s[i]　　访问串中下标为i的字符
- 例：
  - string s1 = "abc", s2 = "def";
  - string s3 = s1 + s2;　　//结果是"abcdef"
  - bool s4 = (s1 < s2);　　//结果是true
  - char s5 = s2[1];　//结果是'e'

# 小结

- 主要内容
  - ✓数组、指针、动态存储分配、指针与数组、指针与函数、字符串
- 达到的目标
  - ✓理解数组、指针的概念，掌握定义和使用方法，掌握动态存储分配技术，会使用string类。