Chapter 7

Arrays and Vectors

- To use the array data structure to represent a set of related data items. (利用数组结构表示一组相关的数据项)
- To use arrays to store, sort and search lists and tables of values. (利用数组存储、排序与查找序列表数值)
- To declare arrays, initialize arrays and refer to the individual elements of arrays. (声明、初始化数组, 访问数组中的单个元素)
- To pass arrays to functions. (传递数组给函数)
- Basic searching and sorting techniques. (基本的搜索和排序方法)
- To declare and manipulate multidimensional arrays.(声明和使用多维数组)
- To use C++ Standard Library class template vector.(使用C++标准类模板vector)

Topics

- **7.1 Introduction**
- 7.2 Arrays
- 7.3 Examples Using Arrays
- 7.4 Passing Arrays to Functions
- 7.5 Case Study: Class GradeBook Using an Array to Store Grades
- 7.6 Searching Arrays with Linear Search
- 7.7 Sorting Arrays with Insertion Sort
- 7.8 Multidimensional Arrays
- 7.9 Case Study: Class GradeBook Using a Two-Dimensional Array
- 7.10 Introduction to C++ Standard Library ClassTemplate vector

7.1 Introduction

6.8 Case Study: Game of Chance and Introducing enum

```
1.  // Fig. 6.9: fig06_09.cpp, Roll a six-sided die 6,000,000 times.
2.  int main()
3.  {
4.     int frequency1 = 0; // count of 1s rolled
5.     int frequency2 = 0; // count of 2s rolled
6.     int frequency3 = 0; // count of 3s rolled
7.     int frequency4 = 0; // count of 4s rolled
8.     int frequency5 = 0; // count of 5s rolled
9.     int frequency6 = 0; // count of 6s rolled
10.    int face; // stores most recently rolled value
11.    // summarize results of 6,000,000 rolls of a die
12.    for ( int roll = 1; roll <= 6000000; roll++ )
13.    {
14.       face = 1 + rand() % 6; // random number from 1 to 6
15.       // determine roll value 1-6 and increment appropriate counter
16.       switch ( face )
17.       {
18.          case 1: ++frequency1; break; // increment the 1s counter
19.          case 2: ++frequency2; break; // increment the 2s counter
20.          case 3: ++frequency3; break; // increment the 3s counter
```

7.1 Introduction

- C++数据类型

7.1 Introduction

- 构造数据类型

  根据已定义的一种或多种数据类型构造出的新数据类型. 即: 构造数据类型可以分解成若干个"成员"或"元素", 每个"成员"都属于一种已定义数据类型.

- **Array(数组)**

Structures of related data items with same data type, 相同数据类型的元素(Element)组成的序列, 用于描述和存取同一性质的成批数据.

- 声明? 初始化? 赋值? 使用?

Topics

- 7.1 Introduction
- **7.2 Arrays**
- 7.3 Examples Using Arrays
- 7.4 Passing Arrays to Functions
- 7.5 Case Study: Class GradeBook Using an Array to Store Grades
- 7.6 Searching Arrays with Linear Search
- 7.7 Sorting Arrays with Insertion Sort
- 7.8 Multidimensional Arrays
- 7.9 Case Study: Class GradeBook Using a Two-Dimensional Array
- 7.10 Introduction to C++ Standard Library ClassTemplate vector

7.2 Arrays-概念

- 数组:

  相同数据类型元素的列表, 每项称为数组元素, 具有相同数组名, 根据index(索引, 或称subscript下标)来访问.

- 一个下标: 一维数组

- 两个下标: 二维数组

7.2 Arrays-数组的声明

type arrayName [arraySize];

- **arrayName**: 数组名, 必须是标识符
- **type**: 数组元素类型, 可以是非引用类型外的任何数据类型
- **[ ]**: 方括号运算符
- **arraySize**: 数组元素个数, 必须是大于0的整数常量

7.2 Arrays-数组的声明

type arrayName [arraySize];

- **arraySize: 数组元素个数, 必须是大于0的整数常量(字符型、枚举型、整型等)**
- **直接常量**

  int s[10]; int s['a'];
- **符号常量**

  const int arraysize = 10;

  int s[arraysize];

  // 常量变量, 声明时必须进行初始化!

7.2 Arrays-数组的声明

- const int arraysize;

  arraysize = 10;

  int s[arraysize];

1. const object must be initialized if not extern
2. you cannot assign to a variable that is const

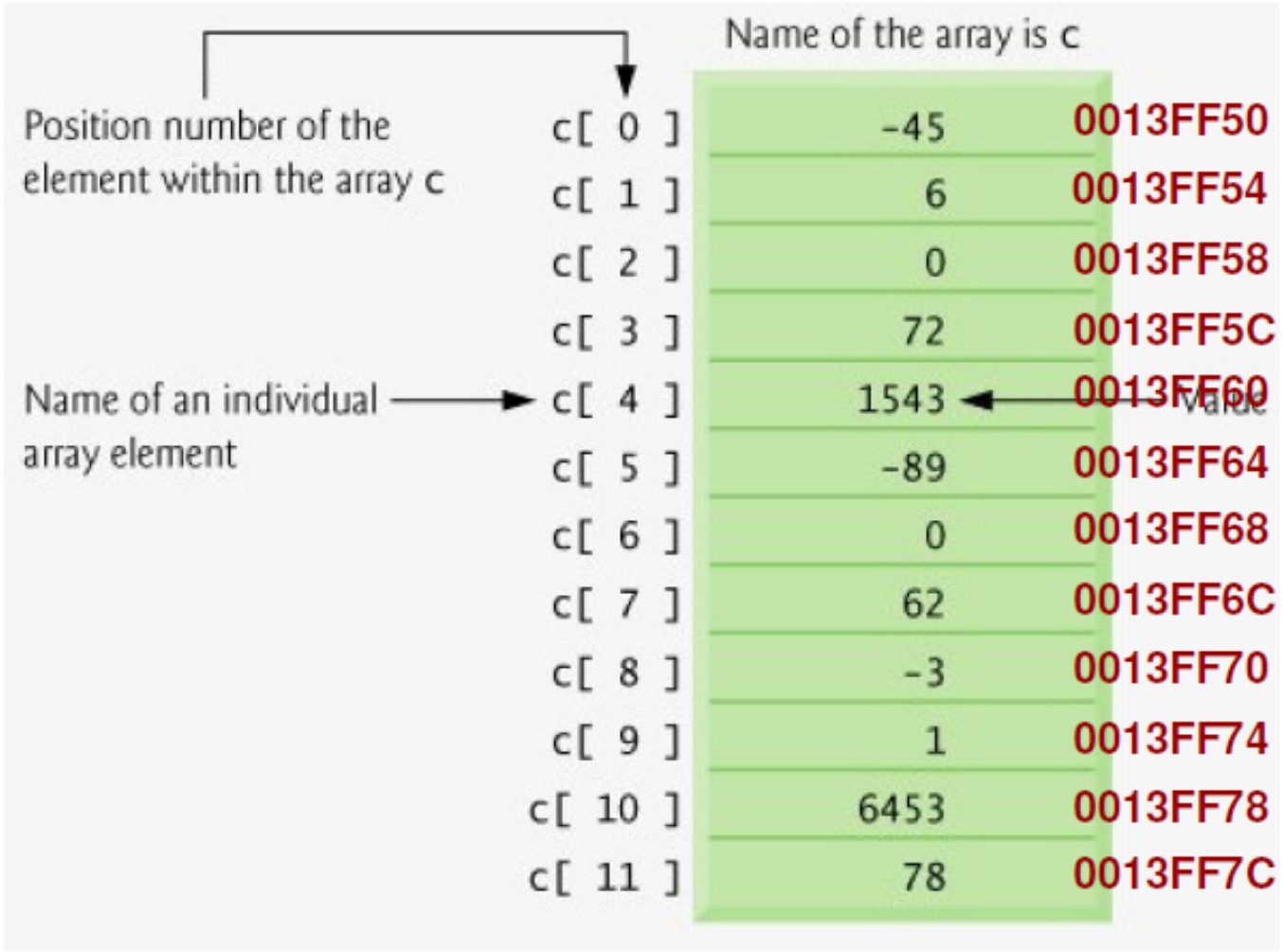
- int arraysize = 10;

  int s[arraysize];

expected constant expression

7.2 Arrays

- 连续的存储区域: int c[12];

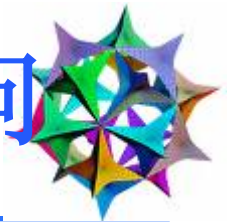


7.2 Arrays-数组元素的访问

- int c[10];
- 可视为一系列相同类型“变量”的序列
- “变量名”为c[index], 即数组名[索引/下标]

- index可以是任何值为≥0且<arraysize的整数值的表达式(变量、常量或函数调用等)
  - 0-based, 第一个元素为c[0]
  - 最后一个元素为c[9]

7.2 Arrays-数组元素的访问

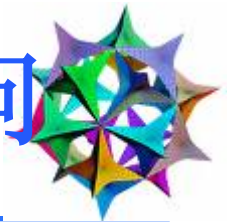


- int c[10];

- 数组元素可以当普通变量使用

```
a=1; b=2; c[ a + b ] += 2;
x = c[ 6 ] / 2;
x = c[ c[3] ];
cout << c[ 0 ] + c[ 1 ] + c[ 2 ] << endl;
```

7.2 Arrays-数组元素的访问



- int c[10];
- c[10] = 6; c[11] = 9;
- 没有编译错误, 属于逻辑错误!
- C++不进行边界检测, 程序员务必确保数组访问不越界!
- 结果: 覆盖相邻内存区域中的值!

c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]	c[7]	c[8]	c[9]	i=6	j=9

7.2 Arrays-数组初始化和赋值

(1)采用Initializer List进行初始化

- int c[10] = {32, 27, 64, 1, 95, 14, 90, 70, 60, 37};
- int c[5] = { 32 }; // 初始化列表不足补0, 即:
- int c[5] = {32, 0, 0, 0, 0}
- int c[ ] = { 1, 2, 3, 4, 5 };

  即: int c[5] = { 1, 2, 3, 4, 5 };

- int c[ ]; // unknown size
- int c[5] = { 32, 27, 64, 18, 95, 14 };
- // error C2078: too many initializers

7.2 Arrays-数组初始化和赋值

- (2)采用循环语句对每个数组元素赋值

```
int n[ 10 ];
for ( int i = 0; i < 10; i++ )
   n[ i ] = 0;

int n[ 10 ];
for ( int i = 0; i < 10; i++ )
   cin>>n[ i ];
```

7.2 Arrays-静态局部数组

static int array[3]; // Static Local Array

- **Starage Class:** 均为静态存储类别
- 初始化: 只进行一次初始化, 如果没有显式初始化, 那么自动初始化为全0
- 保值: 函数调用结束后, 值依然存在.

- 全局数组?

Topics

- 7.1 Introduction
- 7.2 Arrays
- **7.3 Examples Using Arrays**
- 7.4 Passing Arrays to Functions
- 7.5 Case Study: Class GradeBook Using an Array to Store Grades
- 7.6 Searching Arrays with Linear Search
- 7.7 Sorting Arrays with Insertion Sort
- 7.8 Multidimensional Arrays
- 7.9 Case Study: Class GradeBook Using a Two-Dimensional Array
- 7.10 Introduction to C++ Standard Library ClassTemplate vector

```
1.  // Fig 6.9, p198
2.  int frequency1 = 0, frequency2 = 0, frequency3 = 0,
3.      frequency4 = 0, frequency5 = 0, frequency6 = 0;
4.  int face; // stores most recently rolled value
5.  for ( int roll = 1; roll <= 6000000; roll++ )
6.  {
7.      face = 1 + rand() % 6; // random number from 1 to 6
8.      switch ( face )
9.      {
10.       case 1:  ++frequency1; break;
11.       case 2:  ++frequency2; break;
12.       case 3:  ++frequency3; break;
13.       case 4:  ++frequency4; break;
14.       case 5:  ++frequency5; break;
15.       case 6:  ++frequency6; break;
16.       default: cout << "Program should never get here!";
17.     } // end switch
18. } // end for
```

```
1.  int frequency[6] = {0};
2.  int face; // stores most recently rolled value
3.  for ( int roll = 1; roll <= 6000000; roll++ )
4.  {
5.        face = 1 + rand() % 6; // random number from 1 to 6
6.        switch ( face )
7.        {
8.           case 1:  ++frequency[0]; break;
9.           case 2:  ++frequency[1]; break;
10.          case 3:  ++frequency[2]; break;
11.          case 4:  ++frequency[3]; break;
12.          case 5:  ++frequency[4]; break;
13.          case 6:  ++frequency[5]; break;
14.          default: cout << "Program should never get here!";
15.       } // end switch
16. } // end for
```

```
1.  int frequency[7] = {0};
2.  int face; // stores most recently rolled value
3.  for ( int roll = 1; roll <= 6000000; roll++ )
4.  {
5.        face = 1 + rand() % 6; // random number from 1 to 6
6.        switch ( face )
7.        {
8.           case 1:  ++frequency[1]; break;
9.           case 2:  ++frequency[2]; break;
10.          case 3:  ++frequency[3]; break;
11.          case 4:  ++frequency[4]; break;
12.          case 5:  ++frequency[5]; break;
13.          case 6:  ++frequency[6]; break;
14.          default: cout << "Program should never get here!";
15.       } // end switch
16. } // end for
```

7.3 Examples Using Arrays

```
1. int frequency[7] = {0};
2. int face; // stores most recently rolled value
3. for ( int roll = 1; roll <= 6000000; roll++ )
4. {
5.       face = 1 + rand() % 6; // random number from 1 to 6
6.       ++frequency[ face ];
7. } // end for
```

```
1. int frequency[7] = {0};
2. for ( int roll = 1; roll <= 6000000; roll++ )
3. {
4.       frequency[ 1 + rand() % 6 ]++;
5. } // end for
```

7.3 Examples Using Arrays

```
1.  // Fig 7.10, P260
2.  const int arraysize = 7;
3.  int frequency[ arraysize ] = { };
4.
5.  for ( int roll = 1; roll <= 6000000; roll++ )
6.      frequency[ rand()%6 + 1 ]++;
```

7.3 Examples Using Arrays

❑1、求数组的和

```
using std::cout;
using std::endl;

int main()
{
   const int arraySize = 10; // constant variable indicating size of array
   int a[ arraySize ] = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };
   int total = 0;

   // sum contents of array a
   for ( int i = 0; i < arraySize; i++ )
      total += a[ i ];

   cout << "Total of array elements: " << total << endl;

   return 0; // indicates successful termination
} // end main
```

```
Total of array elements: 849
```

7.3 Examples Using Arrays

- 2、求数组的分布

```
const int number=10;
int a[number]={88, 39, 24, 0, 44, 100, 93, 91,
89, 78}
```

- 3、求两个数组的公共元素

```
const int number=5;
int a[number]={1, 2, 3, 4, 5};
int b[number]={3, 4, 5, 6, 7};
```

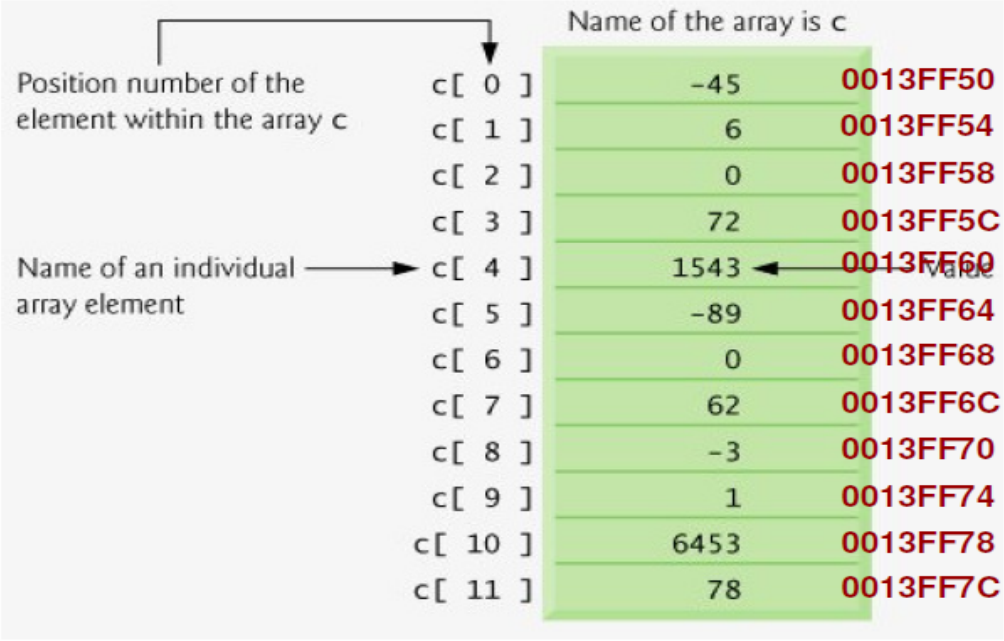
Topics

- 7.1 Introduction
- 7.2 Arrays
- 7.3 Examples Using Arrays
- **7.4 Passing Arrays to Functions**
- 7.5 Case Study: Class GradeBook Using an Array to Store Grades
- 7.6 Searching Arrays with Linear Search
- 7.7 Sorting Arrays with Insertion Sort
- 7.8 Multidimensional Arrays
- 7.9 Case Study: Class GradeBook Using a Two-Dimensional Array
- 7.10 Introduction to C++ Standard Library ClassTemplate vector

7.4 Passing Arrays to Functions

函数参数传递的两种方式:

- Pass-by-Value, 传值
- Pass-by-Reference, 传引用
  - Reference Parameter, 引用参数
  - Pointer Parameter, 指针参数



7.4 Passing Arrays to Functions

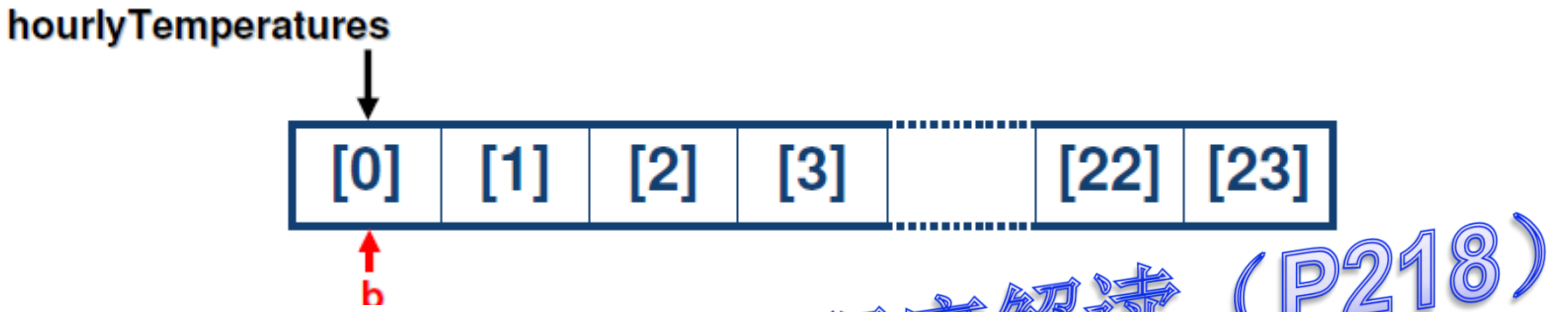
- 函数原型

  void modifyArray(int b[ ], int);

- 函数调用

  int hourlyTemperatures[24];

  modifyArray(hourlyTemperatures, 24);



程序解读（P218）

7.4 Passing Arrays to Functions

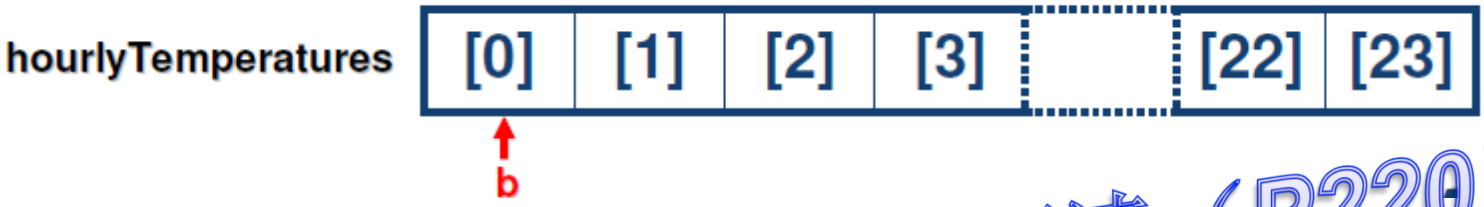
- 函数原型

  void modifyArray(const int b[ ], int);

- 函数调用

  modifyArray(hourlyTemperatures, 24);

- **b**依然指向**hourlyTemperatures**数组首元素，但**const**限定通过**b**不能修改数组元素



程序解读（P220）

Topics

- 7.1 Introduction
- 7.2 Arrays
- 7.3 Examples Using Arrays
- 7.4 Passing Arrays to Functions
- **7.5 Case Study: Class GradeBook Using an Array to Store Grades**
- 7.6 Searching Arrays with Linear Search
- 7.7 Sorting Arrays with Insertion Sort
- 7.8 Multidimensional Arrays
- 7.9 Case Study: Class GradeBook Using a Two-Dimensional Array
- 7.10 Introduction to C++ Standard Library ClassTemplate vector

7.5 Case Study: Class GradeBook Using an Array to Store Grades

- 需求: 对学生成绩进行各种统计分析,如最高分、最低分、成绩分布等.

4.6 Formulating Algorithms: Counter-Controlled Repetition(计数器控制的循环)

```
53    total = 0; // initialize total
54    gradeCounter = 1; // initialize loop counter
55
56    // processing phase
57    while ( gradeCounter <= 10 ) // loop 10 times
58    {
59       cout << "Enter grade: "; // prompt for input
60       cin >> grade; // input next grade
61       total = total + grade; // add grade to total
62       gradeCounter = gradeCounter + 1; // increment by 1
63    } // end while
```

7.5 Case Study: Class GradeBook Using an Array to Store Grades

```
1.  class GradeBook // GradeBook.h
2.  {
3.  public:
4.     GradeBook( string, const int [ ] );
5.  private:
6.     int grades[ 10 ]; // array of student grades
7.  };
```

- 10: Magic Number, 幻数, 不提倡使用

7.5 Case Study: Class GradeBook Using an Array to Store Grades

```
1.  class GradeBook // GradeBook.h
2.  {
3.  public:
4.      const int students = 10;
5.      GradeBook( string, const int [ ] );
6.  private:
7.      int grades[ students ];
8.  };
```

- **ERROR**: 类定义中不能对数据成员进行显式初始化. 头文件中类的定义仅是声明, 并未真正分配内存空间, 因此不能赋值.

7.5 Case Study: Class GradeBook Using an Array to Store Grades

```
1.  class GradeBook // GradeBook.h
2.  {
3.  public:
4.     const static int students = 10;
5.     GradeBook( string, const int [ ] );
6.  private:
7.     int grades[ students ];
8.  };
```

- **OK**: 特例, **Only static const integral data members can be initialized within a class.**

7.5 Case Study: Class GradeBook Using an Array to Store Grades

- 数据成员：

① 普通数据成员

② const数据成员

③ static数据成员

④ static const 数据成员

	non-static	static
non-const	普通成员①	静态成员③
const	常量成员②	静态常量④

7.5 Case Study: Class GradeBook Using an Array to Store Grades

数据成员: 普通, const, static, static const

① 普通数据成员

- 在类定义中只能声明, 不能初始化
- 可在构造函数中赋值

② const数据成员

- 在对象整个生存期中都不能改变, 在类定义中只能声明, 不能初始化
- Conflict: const数据必须初始化
- 必须在构造函数初始化列表中初始化(chap 10)

7.5 Case Study: Class GradeBook Using an Array to Store Grades

数据成员: 普通, const, static, static const

③ static数据成员

- 意义: 在类的所有对象之间共享, 称为class variable(类变量)
- public类变量可以直接通过类名+::访问
- 在类定义中只能声明, 不能初始化
- 在类外部给出定义和初始化

```
// test.h
class Test{
public:
    Test();
    static int num;
    void displayMessage();
};
```

```
// test.cpp
#include "test.h"
#include <iostream>
int Test::num = 0;
Test::Test() {
   num++;
}
void Test::displayMessage(){
   cout<<num<<" ";
}
```

```
// main.cpp
#include "test.h"
int main()
{
   cout<<"Init = " <<Test::num<< endl;
   Test tt1; tt1.displayMessage();
   Test tt2; tt2.displayMessage();
   Test tt3; tt3.displayMessage();
   cout<<"End = "<<Test::num<< endl;

   return 0;
}
```

```
Init = 0
1
2
3
End = 3
```

7.5 Case Study: Class GradeBook Using an Array to Store Grades

数据成员: 普通, const, static, static const

④ static const数据成员

- 特殊的static数据成员, 一般在类定义中只能声明、不能初始化, 须在类外部给出定义和初始化
- 但如果是整数数据成员(integral), 则可在类定义中声明并初始化(ISO C++)

7.5 Case Study: Class GradeBook Using an Array to Store Grades

```
// test.h
class Test{
public:
    Test();
    const static double num;
};
```

```
// test.cpp
#include "test.h"
#include <iostream>
const double Test::num = 5;
Test::Test() {
    cout << fixed;
    cout << num << endl;
}
```

```
// main.cpp
#include "test.h"
int main()
{
    Test t;
    cout<< Test::num << endl;

    return 0;
}
```

7.5 Case Study: Class GradeBook Using an Array to Store Grades

```
1.  class GradeBook// GradeBook.h
2.  {
3.  public:
4.     const static int students = 10;
5.     GradeBook( string, const int [ ] );
6.  private:
7.     int grades[ students ]; // array of student grades
8.  };
```

- OK: Only static const integral data members can be initialized within a class.

7.5 Case Study: Class GradeBook Using an Array to Store Grades

```
#define STUDENTS 10
class GradeBook
{
public:
   GradeBook( string, const int [ ] );
private:
   int grades[ STUDENTS ];
};
```

程序解读（P221）

Topics

- 7.1 Introduction
- 7.2 Arrays
- 7.3 Examples Using Arrays
- 7.4 Passing Arrays to Functions
- 7.5 Case Study: Class GradeBook Using an Array to Store Grades
- **7.6 Searching Arrays with Linear Search**
- 7.7 Sorting Arrays with Insertion Sort
- 7.8 Multidimensional Arrays
- 7.9 Case Study: Class GradeBook Using a Two-Dimensional Array
- 7.10 Introduction to C++ Standard Library ClassTemplate vector

7.6 Searching Arrays with Linear Search

- 判断数组中是否有含有与某个关键值(Key value)相等的元素, 查找过程称为搜索(searching).
- 线性搜索(Linear Search)
  - 将要查找的关键值与数组中每个元素逐个比较
  - 平均情况, 查找关键值需与一半元素相比较
  - 适合于小型数组和未排序的数组

```
int linearSearch(const int array[], int key, int sizeOfArray)
{
    for (int i = 0; i < sizeOfArray; i++)
        if (array[i] == key)
            return i;
    return -1;
}
```

程序解读（P225）

7.6 Searching Arrays with Linear Search

- 如何设计模板适用不同数据类型?

```
int linearSearch(const int array[], int key, int sizeOfArray)
{
    for (int i = 0; i < size
        if (array[i] == key)
            return i;
    return -1;
}
```

```
template <class T>
T linearSearch(const T array[], T key, int sizeOfArray)
{
    for (int i = 0; i < sizeOfArray; i++)
        if (array[i] == key)
            return i;
    return -1;
}

int main()
{
    const int size = 100;
    int a[size];
    double b[size];
    for (int i = 0; i < size; i++)
        a[i] = i * 2;
    for (int i = 0; i < size; i++)
        b[i] = i * 1.1;
    cout << linearSearch(a, 8, size)<<endl;
    cout << linearSearch(b, 8.8, size) << endl;
}
```

7.6 Searching Arrays with Linear Search

- 二分搜索（Binary-Search）
  - 采用分治策略
  - 假设数组已升序排序，将要查找的关键值与数组中间比较：如相等则找到；如小于则继续在数组左半部搜索；如大于则继续在数组右半部搜索
  - 只适用于已排序数组

7.6 Searching Arrays with Linear Search

❑二分搜索（Binary-Search）-迭代法

```
int biSearch(const int array[], int low, int high, int key)
{
    while (low <= high)
    {
        int mid = (low + high) / 2;
        if(array[mid]==key)
            return mid;
        else if (array[mid]>key)
                high = mid - 1;
             else
                low = mid + 1;
    }
    return -1;
}
```

7.6 Searching Arrays with Linear Search

❑二分搜索（Binary-Search）-递归法

```
int biSearch(const int array[], int low, int high, int key)
{

    if (low>high)
        return -1;

    int mid = (low + high) / 2;

    if (array[mid] == key)
        return mid;
    else if (array[mid]>key)
            biSearch(array, low, mid - 1, key);
        else
            biSearch(array, mid + 1, high, key);
}
```

Topics

- 7.1 Introduction
- 7.2 Arrays
- 7.3 Examples Using Arrays
- 7.4 Passing Arrays to Functions
- 7.5 Case Study: Class GradeBook Using an Array to Store Grades
- 7.6 Searching Arrays with Linear Search
- **7.7 Sorting Arrays with Insertion Sort**
- 7.8 Multidimensional Arrays
- 7.9 Case Study: Class GradeBook Using a Two-Dimensional Array
- 7.10 Introduction to C++ Standard Library ClassTemplate vector

7.7 Sorting Arrays with Insertion Sort

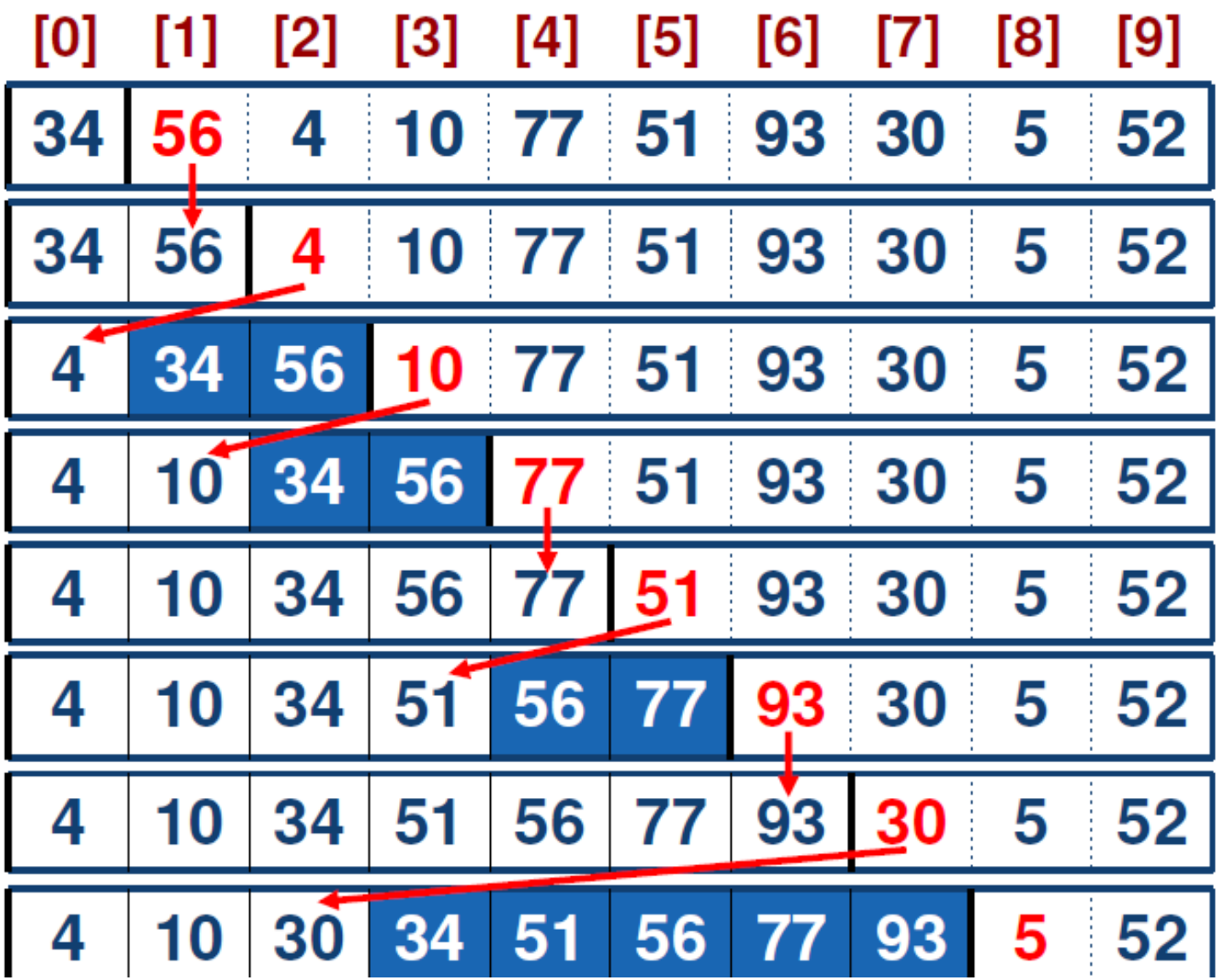
排序(Sorting)

- 将要排序的内容存储在一维数组中, 然后根据某个排序算法交换(Swap)它们的位置, 使它们的值由小到大(或反之)排列.

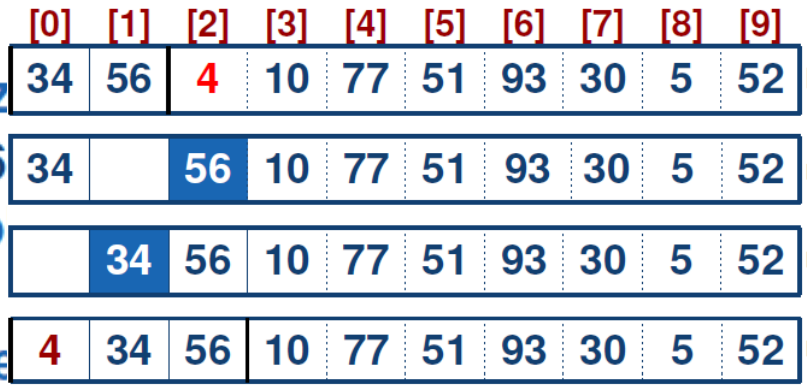
排序算法

- 插入排序(Insertion Sort)
- 选择排序(Selection Sort)
- 快速排序(Quick Sort)
- 冒泡排序(Bubble Sort)

7.7 Sorting Arrays with Insertion Sort



```
1.  // Fig. 7.20: fig07_20.cpp
2.  const int arraySize = 10; // siz
3.  int data[ arraySize ] = { 34, 56
4.  int insert; // temporary variab
5.
6.  // insertion sort, loop over the
7.  for ( int next = 1; next < arraySize; next++ )
8.  {
9.     insert = data[ next ]; // store the value in the
10.    int moveItem = next; // initialize location to
11.
12.    // search for the location in which to put the
13.    while ( ( moveItem > 0 ) && ( data[ moveItem - 1 ] > insert ) )
14.    {
15.       // shift element one slot to the right
16.       data[ moveItem ] = data[ moveItem - 1 ];
17.       moveItem--;
18.    } // end while
19.
20.    data[ moveItem ] = insert; // place inserted element into the array
21. } // end for
```



数组中从第2个元素开始，取每个元素与其前面的元素相比较，找到其应插入位置.

while循环实现当前待插元素(insert)插入位置之前的元素均向右移动一个位置

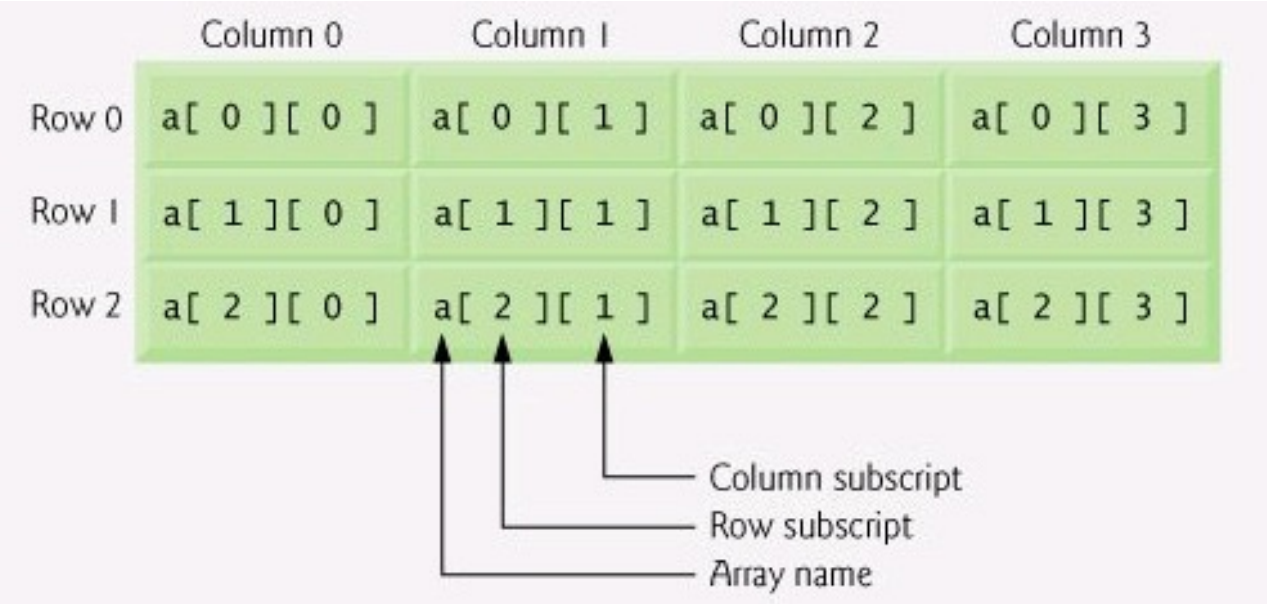
7.7 Sorting Arrays with Insertion Sort

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	
4	34	56	10	77	51	93	30	5	52	insert=10; next=3
4	34		56	77	51	93	30	5	52	moveItem = 3 56 >10
4		34	56	77	51	93	30	5	52	moveItem = 2 34 > 10
4	10	34	56	77	51	93	30	5	52	moveItem = 1 4 < 10

Topics

- 7.1 Introduction
- 7.2 Arrays
- 7.3 Examples Using Arrays
- 7.4 Passing Arrays to Functions
- 7.5 Case Study: Class GradeBook Using an Array to Store Grades
- 7.6 Searching Arrays with Linear Search
- 7.7 Sorting Arrays with Insertion Sort
- **7.8 Multidimensional Arrays**
- 7.9 Case Study: Class GradeBook Using a Two-Dimensional Array
- 7.10 Introduction to C++ Standard Library ClassTemplate vector

7.8 Multidimensional Arrays



- 正确: **int a[m][n];**
- 其中m、n为大于0的整数常量, 称m*n数组,
- 可以视为m个一维数组(n个元素)的数组
- 错误: **int b[m, n];**

7.8 Multidimensional Arrays

- 逐行、顺序存储

  int a[3][4];

a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2][0]	a[2][1]	a[2][2]	a[2][3]

a[0][0]	-45	0013FF50
a[0][1]	2	0013FF54
a[0][2]	6	0013FF58
a[0][3]	178	0013FF5C
a[1][0]	45	0013FF60
a[1][1]	65	0013FF64
a[1][2]	7	0013FF68
a[1][3]	1	0013FF6C
a[2][0]	0	0013FF70
a[2][1]	-5	0013FF74
a[2][2]	7	0013FF78
a[2][3]	345	0013FF7C

7.8 Multidimensional Arrays

- ✓ int a[2][3] = {{ 1, 2, 3 }, { 4, 5, 6 }};
- ✓ int a[2][3] = {1, 2, 3, 4, 5, 6};
- ✓ int a[2][3] = {{1}, {4, 5, 6}};

  1 0 0 4 5 6
- ✓ int a[2][3] = {1, 2};

  1 2 0 0 0 0
- ✓ int a[ ][3] = {{1, 2, 3}, {4, 5, 6}}; // a[2][3]
- ✓ int a[ ][3] = {1, 2, 3, 4, 5, 6}; // a[2][3]
- ✓ int a[ ][3] = {{1, 2}, {4, 5, 6}};

  a[2][3], 1 2 0 4 5 6
- ✓ int a[ ][3] = {1};

  a[1][3], 1 0 0

7.8 Multidimensional Arrays

- int a[2][ ] = {1, 2, 3, 4, 5, 6};

  只有第一维的size可以省略

7.8 Multidimensional Arrays

- 元素访问

```
for ( int nRow = 0; nRow < nNumRows; nRow++)
    for ( int nCol = 0; nCol < nNumCols; nCol++)
        anArray[nRow][nCol] = 0;
```

Topics

- 7.1 Introduction
- 7.2 Arrays
- 7.3 Examples Using Arrays
- 7.4 Passing Arrays to Functions
- 7.5 Case Study: Class GradeBook Using an Array to Store Grades
- 7.6 Searching Arrays with Linear Search
- 7.7 Sorting Arrays with Insertion Sort
- 7.8 Multidimensional Arrays
- **7.9 Case Study: Class GradeBook Using a Two-Dimensional Array**
- 7.10 Introduction to C++ Standard Library ClassTemplate vector

7.9 Case Study: Class GradeBook Using a Two-Dimensional Array

- 10个学生* 3次考试

87, 96, 70,
68, 87, 90,
94, 100, 90,
.....
87, 93, 73

10 X 3数组
int grades[10][3]

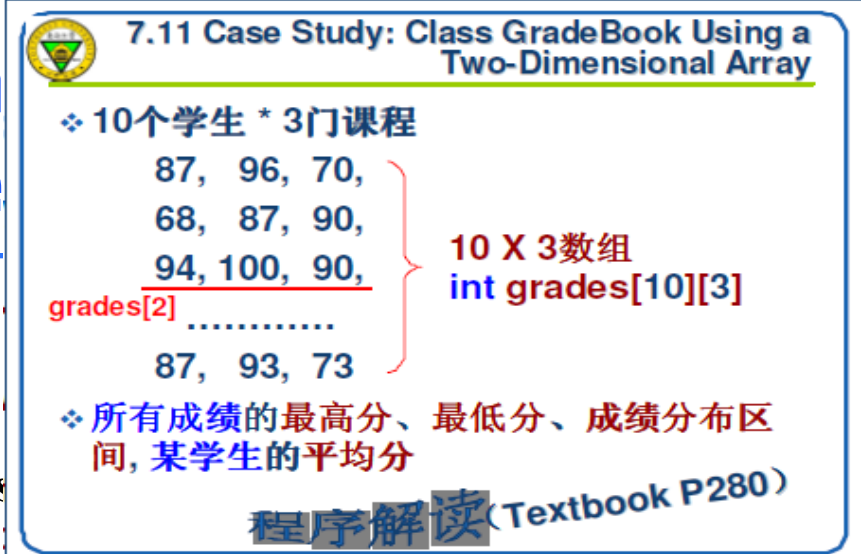
- 所有成绩的最高分、最低分、成绩分布区间，某学生的平均分

程序解读（P231）

7.9 Case Study: Class GradeBook Using a Two-Dimensional Array

```
1.  double GradeBook::getAverage(
2.                                  const int setOfGrades[ ],
3.                                  const int grades )
4.  {
5.     int total = 0; // initialize total
6.
7.     for ( int grade = 0; grade < grades; grade++ )
8.        total += setOfGrades[ grade ];
9.
10.    return static_cast< double >( total ) / grades;
11. }
```

7.9 Case Study: Cla
Using a Two-Dime




```
1. void GradeBook::outputGrades()
2. {
3.   cout << "\nThe grades are:\n\n";
4.   cout << “            "; // align column heads
5.
6.    for ( int test = 0; test < tests; test++ ) // 输出列表头
7.          cout << "Test " << test + 1 << " ";
8.   cout << "Average" << endl; // student average column heading
9.
10.  for ( int student = 0; student < students; student++ )
11.  {
12.         cout << “Student ” << setw( 2 ) << student + 1; // 输出行表头
13.
14.         for ( int test = 0; test < tests; test++ )
15.            cout << setw( 8 ) << grades[ student ][ test ];
16.
17.         double average = getAverage( grades[ student ], tests );
18.         cout << setw( 9 ) << setprecision( 2 ) << fixed << average << endl;
19.  }
20. }
```

Q & A

```
1.  int getElement( const int array[ ], int index )
2.  {
3.      return array[index];
4.  }
5.  int main( )
6.  {
7.      int s[ ][3] = {{1}, {2, 3}, {4, 5, 6}};
8.      for ( int m = 0; m < 3; m++ ){
9.          for ( int n = 0; n < 3; n++ )
10.             cout << s[m][n] << " ";
11.         cout << endl;
12.     }
13.     int sum = 0;
14.     for ( int i = 0; i < 3; i++ ){
15.         sum += getElement( s[ i ],  i );
16.     }
17.     cout << sum << endl;
18.     return 0;
19. }
```

Topics

- 7.1 Introduction
- 7.2 Arrays
- 7.3 Examples Using Arrays
- 7.4 Passing Arrays to Functions
- 7.5 Case Study: Class GradeBook Using an Array to Store Grades
- 7.6 Searching Arrays with Linear Search
- 7.7 Sorting Arrays with Insertion Sort
- 7.8 Multidimensional Arrays
- 7.9 Case Study: Class GradeBook Using a Two-Dimensional Array
- **7.10 Introduction to C++ Standard Library ClassTemplate vector**

7.10 Introduction to C++ Standard Library ClassTemplate vector

数组使用存在的不足:

- 使用数组很容易越界
- 作为参数传递时, 必须传递其首地址和数组大小
- 两个数组不能够直接用“关系”运算符进行比较
- 两个数组不能够直接作赋值运算

解决方案:

- C++ Standard Library中的vector (向量)
- STL中的类模板(Class Template, Ch14) : 类中的某些数据成员、函数参数或者函数返回值的类型可以由用户指定.

7.10 Introduction to C++ Standard Library ClassTemplate vector

标准模板库(STL, Standard Template Library)

- ANSI / ISO C++标准库(C++ Standard Library)的一个重要组成部分, 包含了诸多在计算机科学领域里常用的基本数据结构和基本算法, 是具有工业强度的、高效的C++程序库.

- vector: 同一种类型的对象的集合, 每个对象都有对应的整数索引值. 由于vector可以包含其他对象, 因此被称为容器.

7.10 Introduction to C++ Standard Library ClassTemplate vector

(1)头文件和using声明

- #include <vector>
- using std::vector;

(2)vector变量定义的几种方式

- vector<int> v2(v1); // v2是v1的一个副本
- vector<int> v3(n, i); // n个元素, 初始值为i
- vector<int> v4(n); // n个元素, 取缺省值0

7.10 Introduction to C++ Standard Library ClassTemplate vector

(3)成员函数

- **v.empty()**

  返回类型： **bool**, 如果v为空, 返回**true**, 否则返回**false**

- **v.size()**

  返回类型： **size_t** (即unsigned int)， 返回向量v中元素的个数

7.10 Introduction to C++ Standard Library ClassTemplate vector

(3)成员函数

- **v.push_back (const T& x);**
  - **Add element at the end**

```
vector<int> myvector;
int myint;

std::cout << "Please enter some integers (enter 0 to end):\n";

do {
    cin >> myint;
    myvector.push_back(myint);
} while (myint);

std::cout << "myvector stores " << int(myvector.size()) << " numbers.\n";
```

7.10 Introduction to C++ Standard Library ClassTemplate vector

(4)常用运算操作

- **v1 = v2**

  赋值操作, v1的元素替换为v2元素副本

- **v1== v2**

  v1和v2比较, 判断元素是否完全相等

- **v[n]**
  - 返回向量v中下标为n的元素
  - 通过下标读/写已经存在的元素
  - C++未对n作越界判断

7.10 Introduction to C++ Standard Library ClassTemplate vector

解决方案: 另一个成员函数

- v.at(int n)
  - 返回向量v中下标为n的元素
  - provide bounds checking, 边界判断
  - throws an exception (异常, Ch16) if its argument is an invalid subscript. By default, this causes a C++ program to terminate.

程序解读（P236）

Summary

- 利用数组结构表示一组相关的数据
- 利用数组存储、排序与查找序列或表的数值
- 声明数组、初始化数组、引用数组中的元素
- 传递数组给函数
- 基本的查找和排序方法
- 线性查找
- 插入排序
- 声明和使用多维数组
- 使用C++标准类模板vector

Homework

- 实验必选题目:
- 11, 12, 15, 22, 29, 31, 33, 37, 40
- 实验任选题目:
- 23, 24, 25, 26, 27, 28, 30
- 作业题目(Homework):
- 18, 21