Chapter 7

Arrays and Vectors

- To use the array data structure to represent a set of related data items
- To declare arrays, initialize arrays and refer to the individual elements of arrays.
- To pass arrays to functions.
- Basic searching and sorting techniques.
- To declare and manipulate multidimensional arrays.
- To use C++ Standard Library class template array, vector.

Topics

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

7.1 Introduction

6.8 Case Study: Game of Chance and Introducing enum

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

7.1 Introduction

❖ C++数据类型

- 数据类型
  - 基本数据类型 (系统提供)
    - 整数类型: int
    - 浮点类型: float, double
    - 字符类型: char
    - 布尔类型: bool
    - 空值类型: void
  - 构造数据类型 (用户定义)
    - 枚举类型
    - 数组类型
    - 结构和联合类型
    - 指针类型
    - 引用类型
  - 抽象数据类型 (用户定义)
    - 类
    - 派生类

Topics

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

7.2 Built-In 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 Built-In 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 Built-In 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**

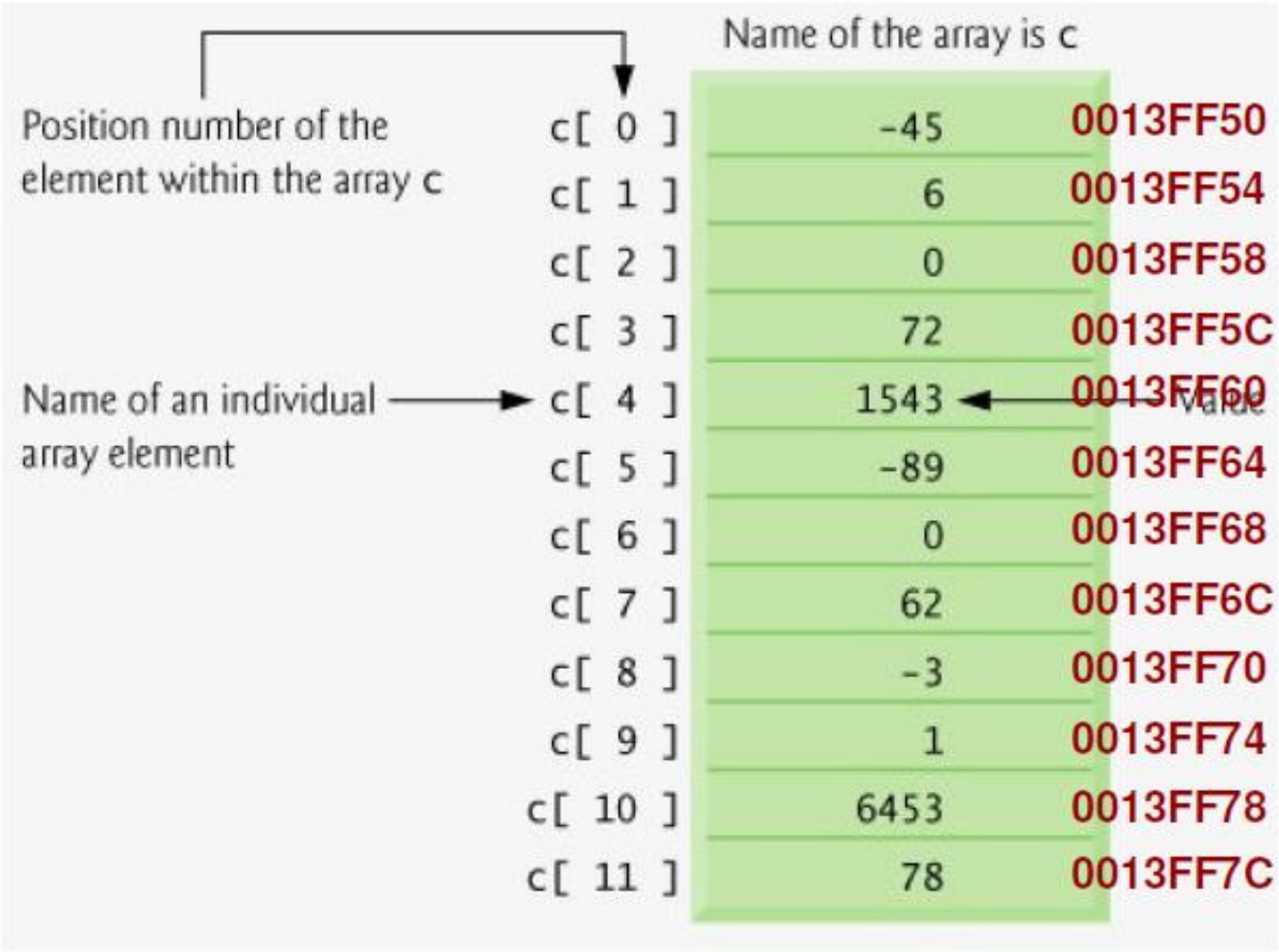
  const int arraysize = 10;

  int s[arraysize];

  // 常量变量, 声明时必须进行初始化!

7.2 Built-In Arrays

- 连续的存储区域: **int c[12];**



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

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

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

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

7.3 Examples Using Built-In 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 Built-In Arrays

- 2、求数组数据的分布

```
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};
```

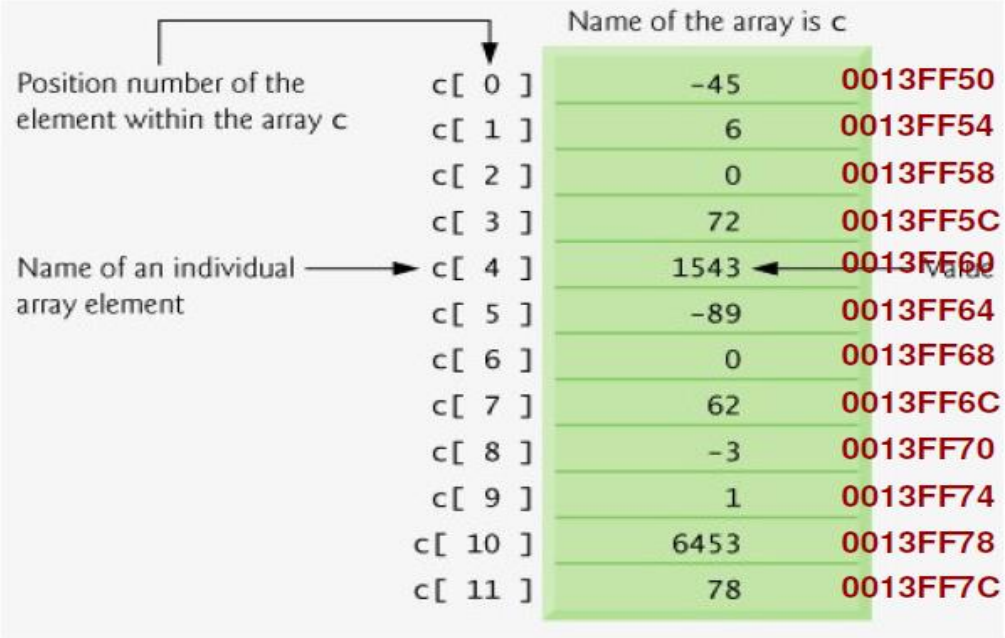
Topics

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

7.4 Passing Built-In 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 Buil Funct

- 函数原型

  void modifyArray(int b[ ], int)

  void modifyElement(int e);

- 函数调用

  const int s=5;

  int a[s]={0,1,2,3,4};

  modifyArray(a, s);

  modifyElement(a[1]);

```
void modifyArray(int b[], int size)
{
    for(int i=0;i<size;i++)
        b[i]*=2;
}
```

```
void modifyElement(int e)
{
    e*=2;
}
```

```
int main()
{
    const int s=5;
    int a[s]={0,1,2,3,4};
    modifyElement(a[1]);
    for(int i=0;i<s;i++)
        cout<<a[i];
    cout<<endl;

    modifyArray(a,s);
    for(int i=0;i<s;i++)
        cout<<a[i];
    cout<<endl;
}
```

Topics

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

7.5 Searching Built-In 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;
}
```

7.5 Searching Built-In 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 < sizeOf
        if (array[i] == key)
            return i;
    return -1;
}
```

```
template <class T>
int 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;

    system("pause");

}
```

7.5 Searching Built-In 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

7.5 Searching Built-In 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;
}
```

7.5 Searching Built-In 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)
            return biSearch(array, low, mid - 1, key);
        else
            return biSearch(array, mid + 1, high, key);
}
```

Topics

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

7.6 Sorting Built-In 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)

7.6 Sorting Built-In Arrays with Insertion Sort

右移元素

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

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

```
// Fig. 7.20: fig07_20.cpp
const int arraySize = 10; // siz
int data[ arraySize ] = { 34, 56
int insert; // temporary variab

// insertion sort, loop over the
for ( int next = 1; next < arraySize; next++ )
{
   insert = data[ next ]; // store the value in the
   int moveItem = next; // initialize location to

   // search for the location in which to put the
   while ( ( moveItem > 0 ) && ( data[ moveItem - 1 ] > insert ) )
   {
      // shift element one slot to the right
      data[ moveItem ] = data[ moveItem - 1 ];
      moveItem--;
   } // end while

   data[ moveItem ] = insert; // place inserted element into the array
} // 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

Topics

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

7.7 Introduction of arrays

- **Declaring arrays**
  - **array<type, arraySize> arrayName;**
- **Examples**
  - **array<int, 5> a;**
  - **const size=10;**
  - **array<char, size> b;**

7.7 Introduction of arrays

- Initializing of arrays
  - array<int, 5> a={0,1,2,3,4};
  - array<char, 3> b={‘a’, ‘b’, ‘c’};
  - const size=10;
  - array<double, size> c;
  - for(int i=0; i<c.size(); i++)
    c[i]=1.1*i;
  - array< int, 5 > items = { 1, 2, 3, 4, 5 };
  - array< int, 5 > newArray = items;

7.7 Introduction of arrays

❑ Using arrays to Summarize Survey Results

```
int main()
{
    // define array sizes
    const size_t responseSize = 20; // size of array responses
    const size_t frequencySize = 6; // size of array frequency

    // place survey responses in array responses
    const array< unsigned int, responseSize > responses =
    { 1, 2, 5, 4, 3, 5, 2, 1, 3, 1, 4, 3, 3, 3, 2, 3, 3, 2, 2, 5 };

    // initialize frequency counters to 0
    array< unsigned int, frequencySize > frequency = {};

    .
    .
    .

    .
    .

} // end main
```

```
Rating          Frequency
     1                  3
     2                  5
     3                  7
     4                  2
     5                  3
```

7.7 Introduction of arrays

❑ array作为参数传递

```
#include <iostream>
#include <array>

void f(array<int, 5> a)
{
    for (int i = 0; i < a.size(); i++)
        a[i] *= 2;
}

int main()
{
    array<int, 5> items = { 1,2,3,4,5 };
    for (int i=0;i<items.size();i++ )
        cout << items[i] << " ";
    cout << endl;

    f(items);

    for (int i=0;i<items.size();i++)
        cout << items[i] << " ";
    cout << endl;
}
```

```
1 2 3 4 5
1 2 3 4 5
请按任意键继续. . .
```

7.7 Introduction of arrays

Range-Based for statement

```
items before modification: 1 2 3 4 5
items after modification: 2 4 6 8 10
```

```
int main()
{
    array< int, 5 > items = { 1, 2, 3, 4, 5 };

    // display items before modification
    cout << "items before modification: ";
    for (int item : items)
        cout << item << " ";

    // multiply the elements of items by 2
    for (int& itemRef : items)
        itemRef *= 2;

    // display items after modification
    cout << "\nitems after modification: ";
    for (int item : items)
        cout << item << " ";

    cout << endl;
} // end main
```

Topics

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

7.8 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.

```
Welcome to the grade book for
CS101 Introduction to C++ Programming!

The grades are:

Student  1:  87
Student  2:  68
Student  3:  94
Student  4: 100
Student  5:  83
Student  6:  78
Student  7:  85
Student  8:  91
Student  9:  76
Student 10:  87

Class average is 84.90
Lowest grade is 68
Highest grade is 100

Grade distribution:
  0-9:
10-19:
20-29:
30-39:
40-49:
50-59:
60-69: *
70-79: **
80-89: ****
90-99: **
  100: *
```

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

```
class GradeBook // GradeBook.h
{
public:
    GradeBook( string, const int [ ] );
private:
    int grades[ 10 ]; // array of student grades
};  array< int, 10> grades;
```

- 10: Magic Number, 幻数, 不提倡使用

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

```
class GradeBook // GradeBook.h
{
public:
    const static int students = 10;
    GradeBook( string, const int [ ] );
private:
    int grades[ students ];
};   array< int, students> grades;
```

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

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

```
class GradeBook
{
public:
    // constant -- number of students who took the test
    static const size_t students = 10; // note public data

    // constructor initializes course name and array of grades
    GradeBook(const string&, const array< int, students >&);

    void setCourseName(const string&); // set the course name
    string getCourseName() const; // retrieve the course name
    void displayMessage() const; // display a welcome message
    void processGrades() const; // perform operations on the grade data
    int getMinimum() const; // find the minimum grade for the test
    int getMaximum() const; // find the maximum grade for the test
    double getAverage() const; // determine the average grade for the test
    void outputBarChart() const; // output bar chart of grade distribution
    void outputGrades() const; // output the contents of the grades array
private:
    string courseName; // course name for this grade book
    array< int, students > grades; // array of student grades
}; // end class GradeBook
```

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

```
// constructor initializes courseName and grades array
GradeBook::GradeBook(const string& name,
    const array< int, students >& gradesArray)
    : courseName(name), grades(gradesArray)
{
} // end GradeBook constructor
```

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

```
// perform various operations on the data
void GradeBook::processGrades() const
{
    // output grades array
    outputGrades();

    // call function getAverage to calculate the average grade
    cout << setprecision(2) << fixed;
    cout << "\nClass average is " << getAverage() << endl;

    // call functions getMinimum and getMaximum
    cout << "Lowest grade is " << getMinimum() << "\nHighest grade is "
        << getMaximum() << endl;

    // call function outputBarChart to print grade distribution chart
    outputBarChart();
} // end function processGrades
```

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

```
int GradeBook::getMinimum() const
{
    int lowGrade = 100; // assume lowest grade is 100

    // loop through grades array
    for (int grade : grades)
    {
        // if current grade lower than lowGrade, assign it to lowGrade
        if (grade < lowGrade)
            lowGrade = grade; // new lowest grade
    } // end for

    return lowGrade; // return lowest grade
} // end function getMinimum
```

```
// output bar chart displaying grade distribution
void GradeBook::outputBarChart() const
{
    cout << "\nGrade distribution:" << endl;

    // stores frequency of grades in each range of 10 grades
    const size_t frequencySize = 11;
    array< unsigned int, frequencySize > frequency = {}; // init to 0s

    // for each grade, increment the appropriate frequency
    for (int grade : grades)
        ++frequency[grade / 10];

    // for each grade frequency, print bar in chart
    for (size_t count = 0; count < frequencySize; ++count)
    {
        // output bar labels ("0-9:", ..., "90-99:", "100:" )
        if (0 == count)
            cout << "  0-9: ";
        else if (10 == count)
            cout << "  100: ";
        else
            cout << count * 10 << "-" << (count * 10) + 9 << ": ";

        // print bar of asterisks
        for (unsigned int stars = 0; stars < frequency[count]; ++stars)
            cout << '*';

        cout << endl; // start a new line of output
    } // end outer for
} // end function outputBarChart
```

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

```
#include <array>
#include "GradeBook.h" // GradeBook class definition
using namespace std;

// function main begins program execution
int main()
{
    // array of student grades
    const array< int, GradeBook::students > grades =
       { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };
    string courseName = "CS101 Introduction to C++ P

    GradeBook myGradeBook( courseName, grades );
    myGradeBook.displayMessage();
    myGradeBook.processGrades();
 } // end main
```

```
Welcome to the grade book for
CS101 Introduction to C++ Programming!

The grades are:

Student  1:  87
Student  2:  68
Student  3:  94
Student  4: 100
Student  5:  83
Student  6:  78
Student  7:  85
Student  8:  91
Student  9:  76
Student 10:  87

Class average is 84.90
Lowest grade is 68
Highest grade is 100

Grade distribution:
  0-9:
10-19:
20-29:
30-39:
40-49:
50-59:
60-69: *
70-79: **
80-89: ****
90-99: **
  100: *
```

Q & A

- 设计一个正整数集合类，最多有10个元素，支持集合的显示，元素个数统计，元素的增加，判断某个元素是否属于该集合。

```
class Set
{
public:
    static const size_t num = 10;
    Set(const array< int, num >&);

    void displaySet();
    void addElement(int e);
    bool containElement(int e);
    int size();

private:
    array< int, num > set;
};
```

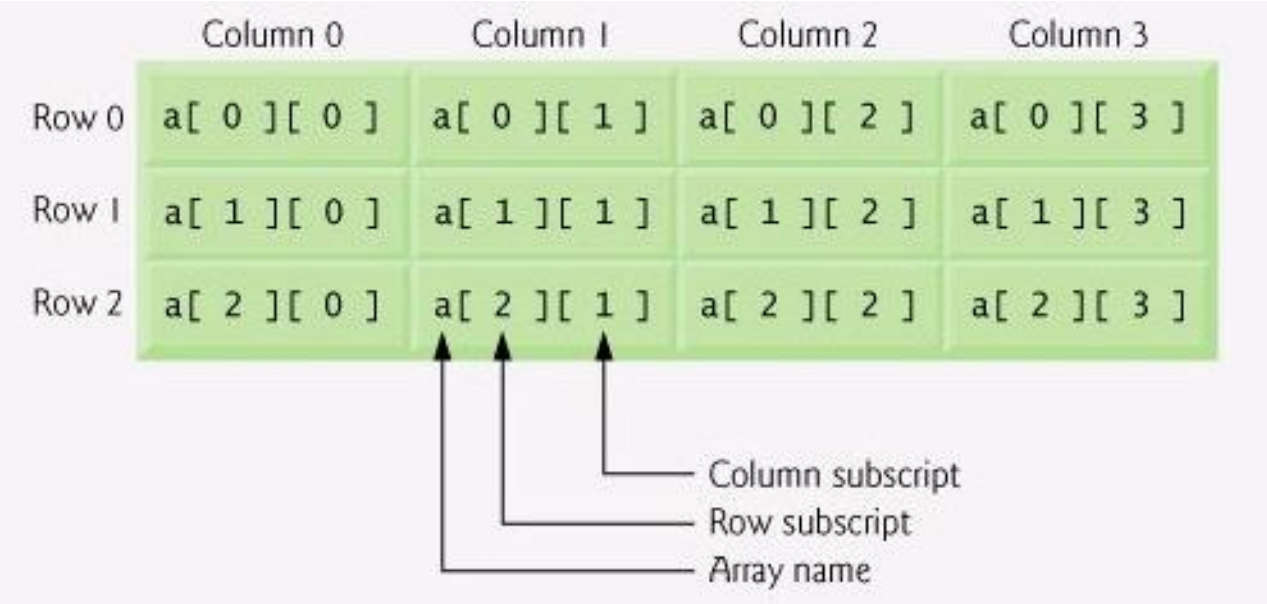
```
int main() {
    array<int, 10> a={-1,0,2,3,3,4};
    Set set(a);
    set.displaySet();
    set.addElement(5);
    set.displaySet();
    return 0;
}
```

```
{2,3,4}
{2,3,4,5}
```

Topics

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

7.9 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];**

7.9 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

7.9 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

7.9 Multidimensional Arrays

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

  只有第一维的size可以省略

7.9 Multidimensional Arrays

- 元素访问

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

7.9 Multidimensional Arrays

- 例1. 求n*n矩阵下三角元素的和。
- 例2. 给定两个n*n矩阵，求矩阵加法。
- 例3. 给定两个n*n矩阵，求矩阵乘法。

Q & A

```
int getElement( const int array[ ], int index )
{
    return array[index];
}
int main( )
{
    int s[ ][3] = {{1}, {2, 3}, {4, 5, 6}};
    for ( int m = 0; m < 3; m++ ){
        for ( int n = 0; n < 3; n++ )
            cout << s[m][n] << " ";
        cout << endl;
    }
    int sum = 0;
    for ( int i = 0; i < 3; i++ ){
        sum += getElement( s[ i ],  i );
    }
    cout << sum << endl;
    return 0;
}
```

Topics

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

7.10 Case Study: Class GradeBook

```
const size_t rows = 2;
const size_t columns = 3;
void printArray( const array< array< int, columns >, rows> & );

int main()
{
   array< array< int, columns >, rows > array1 = { 1, 2, 3, 4, 5, 6 };
   array< array< int, columns >, rows > array2 = { 1, 2, 3, 4, 5 };

   cout << "Values in array1 by row are:" << endl;
   printArray( array1 );

   cout << "\nValues in array2 by row are:" << endl;
   printArray( array2 );
} // end main

// output array with two rows and three columns
void printArray( const array< array< int, columns >, rows> & a )
{
   // loop through array's rows
   for ( auto const &row : a )
   {
      // loop through columns of current row
      for ( auto const &element : row )
         cout << element << ' ';

      cout << endl; // start new line of output
   } // end outer for
} // end function printArray
```

7.10 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]

array< array< int, 3>, 10> grades;

- 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（P257）

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

```
class GradeBook
{
public:
   // constants
   static const size_t students = 10; // number of students
   static const size_t tests = 3; // number of tests

   // constructor initializes course name and array of grades
   GradeBook( const string &,
      array< array< int, tests >, students > & );

   void setCourseName( const string & ); // set the course name
   string getCourseName() const; // retrieve the course name
   void displayMessage() const; // display a welcome message
   void processGrades() const; // perform operations on the grade data
   int getMinimum() const; // find the minimum grade in the grade book
   int getMaximum() const; // find the maximum grade in the grade book
   double getAverage( const array< int, tests > & ) const;
   void outputBarChart() const; // output bar chart of grade distribution
   void outputGrades() const; // output the contents of the grades array
private:
   string courseName; // course name for this grade book
   array< array< int, tests >, students > grades; // 2D array
}; // end class GradeBook
```

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

```
// find minimum grade in the entire gradebook
int GradeBook::getMinimum() const
{
   int lowGrade = 100; // assume lowest grade is 100

   // loop through rows of grades array
   for ( auto const &student : grades )
   {
      // loop through columns of current row
      for ( auto const &grade : student )
      {
         // if current grade less than lowGrade, assign it to lowGrade
         if ( grade < lowGrade )
            lowGrade = grade; // new lowest grade
      } // end inner for
   } // end outer for

   return lowGrade; // return lowest grade
} // end function getMinimum
```

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

```
// determine average grade for particular set of grades
double GradeBook::getAverage( const array<int, tests> &setOfGrades ) const
{
   int total = 0; // initialize total

   // sum grades in array
   for ( int grade : setOfGrades )
      total += grade;

   // return average of grades
   return static_cast< double >( total ) / setOfGrades.size();
} // end function getAverage
```

7.10 Case Study: Cla

```
The grades are:

             Test 1  Test 2  Test 3  Average
Student  1       87      96      70    84.33
Student  2       68      87      90    81.67
Student  3       94     100      90    94.67
Student  4      100      81      82    87.67
Student  5       83      65      85    77.67
Student  6       78      87      65    76.67
Student  7       85      75      83    81.00
Student  8       91      94     100    95.00
Student  9       76      72      84    77.33
Student 10       87      93      73    84.33
```

```
// output the contents of the grades array
void GradeBook::outputGrades() const
{
   cout << "\nThe grades are:\n\n";
   cout << "             "; // align column heads

   // create a column heading for each of the te
   for ( size_t test = 0; test < tests; ++test )
      cout << "Test " << test + 1 << "  ";

   cout << "Average" << endl; // student average column heading

   // create rows/columns of text representing array grades
   for ( size_t student = 0; student < grades.size(); ++student )
   {
      cout << "Student " << setw( 2 ) << student + 1;

      // output student's grades
      for ( size_t test = 0; test < grades[ student ].size(); ++test )
         cout << setw( 8 ) << grades[ student ][ test ];

      // call member function getAverage to calculate student's average;
      // pass row of grades as the argument
      double average = getAverage( grades[ student ] );
      cout << setw( 9 ) << setprecision( 2 ) << fixed << average << endl;
   } // end outer for
} // end function outputGrades
```

7.10 Case Study Using a Two-D

```
// function main begins program e
int main()
{
   // two-dimensional array of st
   array< array< int, GradeBook::
      { 87, 96, 70,
        68, 87, 90,
        94, 100, 90,
        100, 81, 82,
        83, 65, 85,
        78, 87, 65,
        85, 75, 83,
        91, 94, 100,
        76, 72, 84,
        87, 93, 73 };

   GradeBook myGradeBook(
      "CS101 Introduction to C++
   myGradeBook.displayMessage();
   myGradeBook.processGrades();
} // end main
```

```
Welcome to the grade book for
CS101 Introduction to C++ Programming!

The grades are:

             Test 1  Test 2  Test 3  Average
Student  1       87      96      70    84.33
Student  2       68      87      90    81.67
Student  3       94     100      90    94.67
Student  4      100      81      82    87.67
Student  5       83      65      85    77.67
Student  6       78      87      65    76.67
Student  7       85      75      83    81.00
Student  8       91      94     100    95.00
Student  9       76      72      84    77.33
Student 10       87      93      73    84.33

Lowest grade in the grade book is 65
Highest grade in the grade book is 100

Overall grade distribution:
  0-9:
10-19:
20-29:
30-39:
40-49:
50-59:
60-69: ***
70-79: ******
80-89: ***********
90-99: *******
  100: ***
```

Topics

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

7.11 Introduction to C++ Standard Library ClassTemplate vector

Built-In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, Ch18) : 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.**

7.11 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.

7.11 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

7.11 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

7.11 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";
```

7.11 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

7.11 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**) if its argument is an invalid subscript. By default, this causes a C++ program to terminate.**

程序解读（P261）

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
- 了解C++标准类模板vector

Homework

- 实验必选题目:
- 13(实验手册Ex1), 16(实验手册Ex2), 21, 27(实验手册Ex4), 31, 32, 实验手册Ex3
- 实验任选题目:
- 24, 25, 33