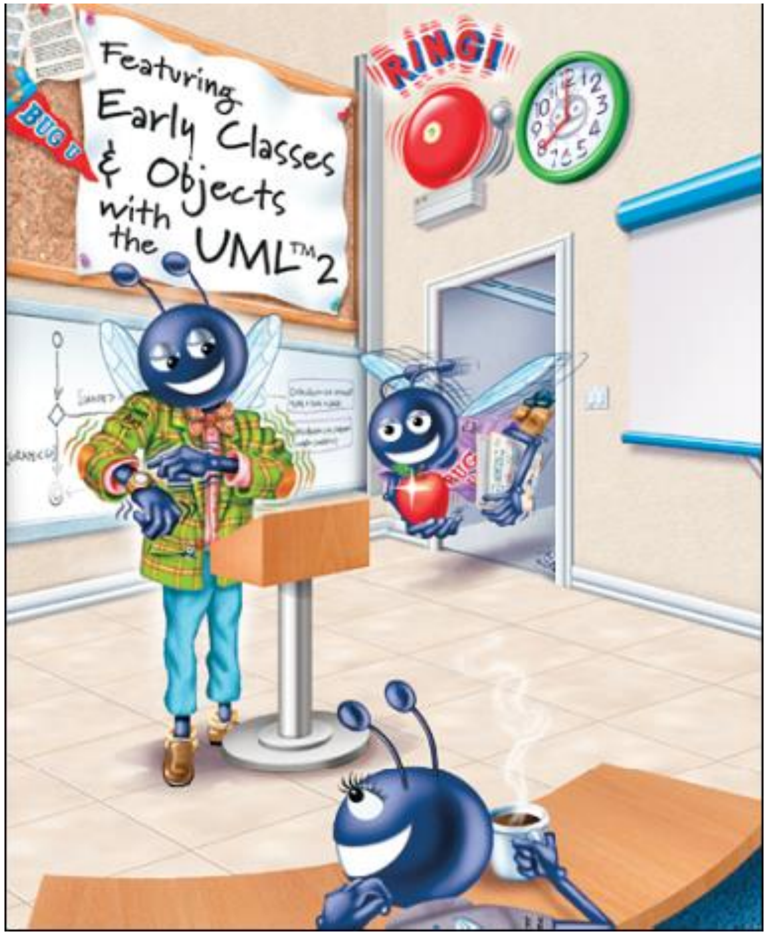
C++程序设计



上节课内容回顾

1. 利用随机数生成机制实现模拟技术
2. 标识符的存储类别和可见性
3. 函数的递归调用
4. 引用、内联函数、函数重载、函数模板

指出下面程序段中的错误并改正：

a)

```
float cube( float );
double cube( float number )
{
    return number * number * number;
}
```

指出下面程序段中的错误并改正：

b)

```
int randomNumber = srand();
```

c)

```
float y = 123.45678;
int x;
x = y;
cout << static_cast< float >( x );
```

指出下面程序段中的错误并改正：

d)

```
double square( double number )
{
   double number;
   return number * number;
}
```

指出下面程序段中的错误并改正：

e)

```
int sum( int n )
{
    if ( n == 0 )
          return 0;
    else
          return n + sum( n );
}
```

第六讲 数组与vector

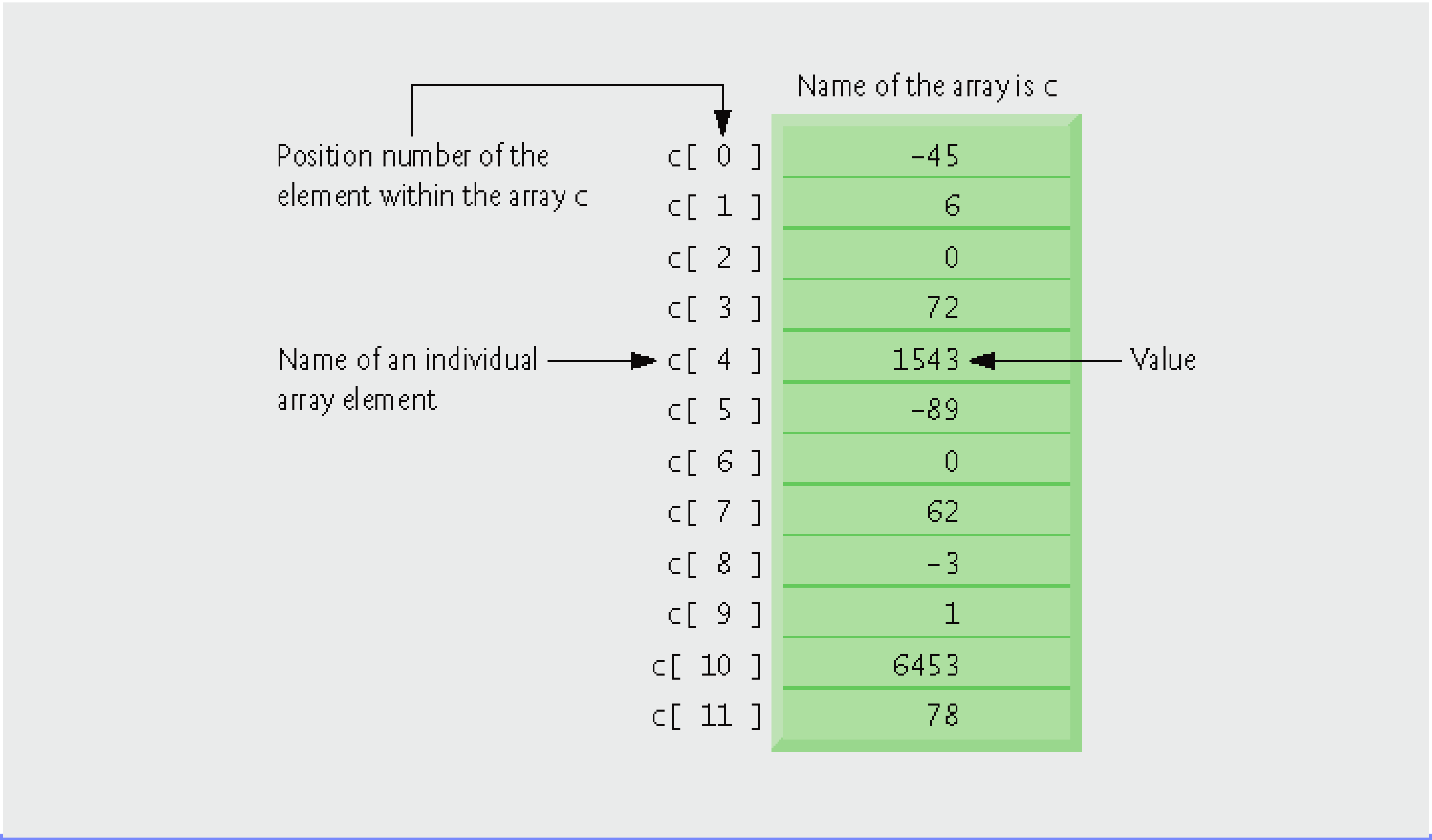
学习目标：

- 声明数组、初始化数组、引用数组中的元素
- 将数组传递给函数
- 多维数组
- 使用C++标准库类模板 vector

1. Introduction

- **Arrays**
  - 包含同一数据类型的数据结构
  - 占用一段连续的内存空间
  - 创建后大小不能改变
  - 通过索引的方式访问数组中的元素

1. Introduction



2. Declaring and Initializing Arrays

- Declaring an array
  - 类型、数组名、数组大小
    - 如： int c[12];
  - 数组大小为大于 0 的常整数

2. Declaring and Initializing Arrays

- 循环初始化数组成员

```
int n[ 10 ];

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

2. Declaring and Initializing Arrays

- 用初始化列表来初始化数组成员
  - 例：int n[] = { 10, 20, 30, 40, 50 };
  - 如果初始化列表的数据量小于数组长度，其余数组元素将被初始化为 0
    - 例：int n[10] = { 0 };
  - 如果初始化列表的数据量大于数组长度，产生编译错误

3. Examples Using Arrays

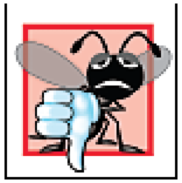
```
……
   const int arraySize = 10;

   int s[ arraySize ]; // array s has 10 elements

   for ( int i = 0; i < arraySize; i++ ) // set the values
       s[ i ] = 2 + 2 * i;
……
```

3. Examples Using Arrays

- **Constant variables**
  - ◈ const 修饰符，又称为常量或只读变量
  - ◈ 声明时必须进行初始化，且以后不能修改
  - ◈ 使用常量变量来声明数组长度使程序更加灵活，避免了“magic numbers”

3. Examples Using Arrays

性能提示：假如不是用执行时的赋值语句来初始化数组，而是在编译时用一个数组初始化列表来初始化数组，程序执行速度会更快。



常见编程错误：只有常量才可用于声明自动和静态数组的长度。不用常量会造成语法错误。

3. Examples Using Arrays

- 用字符数组来存储和处理字符串
  - char string1[] = { 'f', 'i', 'r', 's', 't', '\0' };
  - cin >> string1;
  - 以 ’ \0’ 结尾的字符数组可以通过 cout << 进行输出

4. Passing Arrays to Functions

- 向函数传递数组参数
  - int hourlyTemperatures[24];
  - 函数调用： modifyArray(hourlyTemperatures, 24);

- 接收数组作为参数的函数
  - void modArray(int b[], int arraySize);

4. Passing Arrays to Functions

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

4. Passing Arrays to Functions

- **const array parameters**
  - const 修饰符
  - 阻止被调用的函数修改数组值
  - 在函数体内数组元素为常量
  - 防止程序员意外修改数组元素

4. Passing Arrays to Functions

```
void tryToModifyArray( const int [] );


int main()
{
   int a[] = { 10, 20, 30 };

   tryToModifyArray( a );
   cout << a[ 0 ] << ' ' << a[ 1 ] << ' ' << a[ 2 ] << '\n';
   return 0;
}
```

5. Case Study: Class GradeBook Using an Array to Store Grades

- **static data members**
  - 也称为类变量 （所有对象共享）
  - 即使没有创建对象也可以访问
    - 类名::静态数据成员名

```
class GradeBook
{
public:
   const static int students = 10; // note public data

   GradeBook( string, const int [] );
   ……
private:
   string courseName;
   int grades[ students ];
};
```

```
GradeBook::GradeBook( string name, const int gradesArray[] )
{
  setCourseName( name ); // initialize courseName


  // copy grades from gradeArray to grades data member
  for ( int grade = 0; grade < students; grade++ )
    grades[ grade ] = gradesArray[ grade ];
}
```

```
#include "GradeBook.h"

int main()
{
  int gradesArray[ GradeBook::students ] =
    { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };

  GradeBook myGradeBook(
    "CS101 Introduction to C++ Programming", gradesArray );
  myGradeBook.displayMessage();
  myGradeBook.processGrades();
  return 0;
}
```

6. Searching Arrays with Linear Search

- 数组可存放大量数据
  - 查找指定值 (key value)
- Linear search （线性查找）
  - Compares each element of an array with a search key
  - Just as likely that the value will be found in the first element as the last
  - Works well for small or unsorted arrays

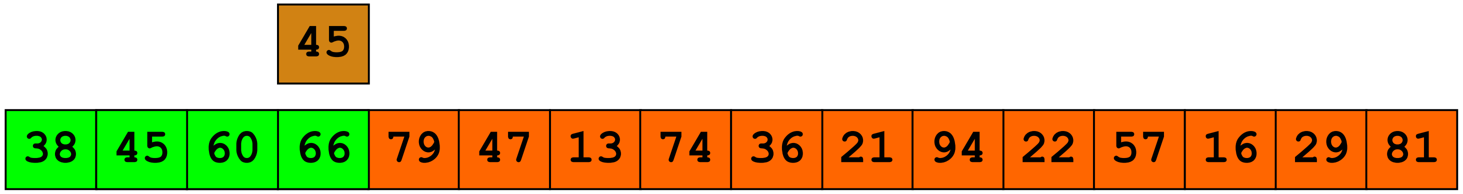
6. Searching Arrays with Linear Search

```
// compare key to every element of array until location is
// found or until end of array is reached; return subscript of
// element if key or -1 if key not found
int linearSearch( const int array[], int key, int sizeOfArray )
{
  for ( int j = 0; j < sizeOfArray; j++ )
    if ( array[ j ] == key ) // if found,
      return j; // return location of key
  return -1; // key not found
} // end function linearSearch
```

7. Sorting Arrays with Insertion Sort

- Sorting data
  - One of the most important computing applications
- Insertion sort （插入排序）
  - First iteration takes second element
  - If it is less than the first element, swap it with first element
  - Second iteration looks at the third element
  - Insert it into the correct position with respect to first two elements
  - …
  - At the ith iteration of this algorithm, the first i elements in the original array will be sorted

7. Sorting Arrays with Insertion Sort



the fourth iteration of this loop is shown here

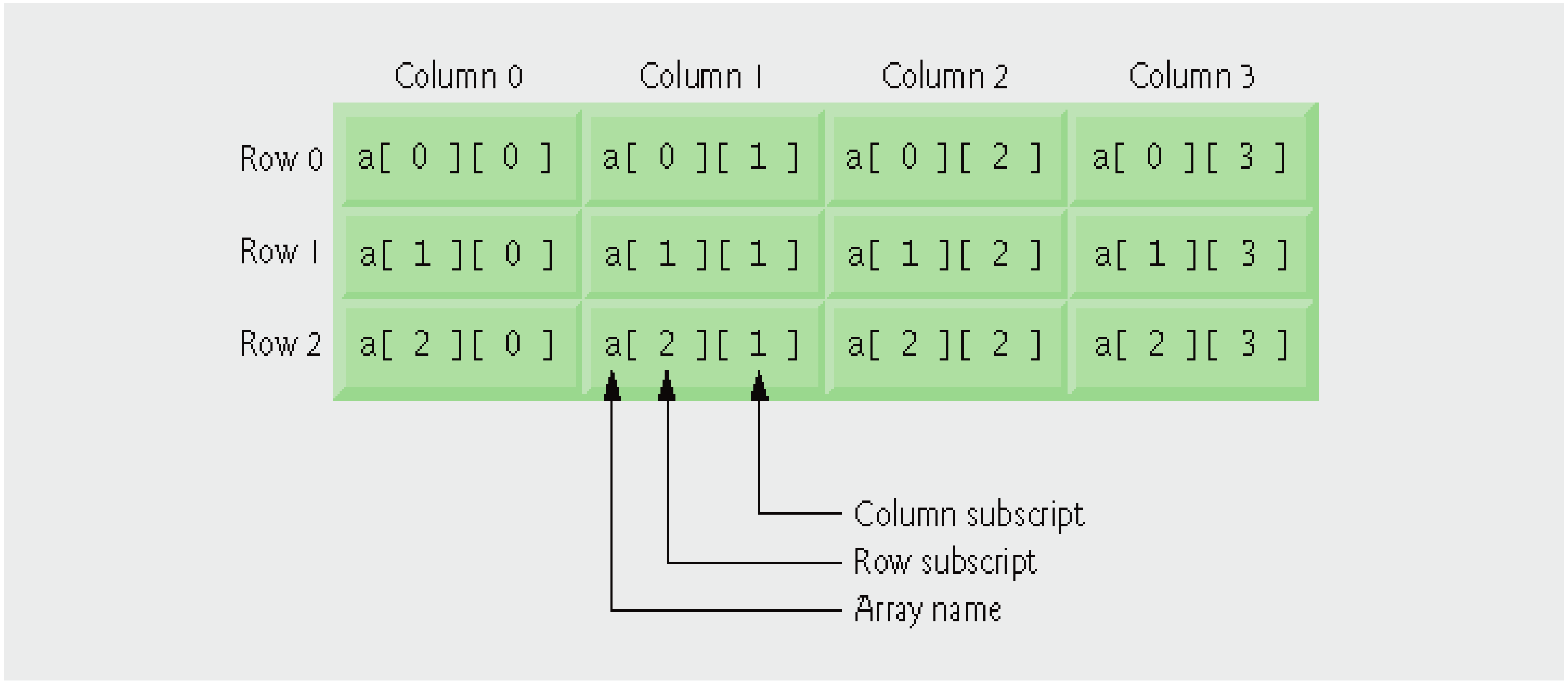
http://courses.cs.vt.edu/~csonline/Algorithms/Lessons/InsertionCardSort/insertioncardsort.swf

```
int main()
{
   const int arraySize = 10; // size of array a
   int data[ arraySize ] = { 34, 56, 4, 10, 77, 51, 93, 30, 5, 52 };
   int insert; // temporary variable to hold element to insert

   cout << "Unsorted array:\n";
   ...
   for ( int next = 1; next < arraySize; next++ )
   {
      insert = data[ next ]; // store the value in the current element

      int moveItem = next; // initialize location to place element
      // search for the location in which to put the current element
      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
   ...
```

8. Multidimensional Array

- 二维数组
  - 表示二维表格中的值
  - 下标为 [x][y]
- 声明并初始化二维数组
  - int b[2][2] = { { 1, 2 }, { 3, 4 } };
  - int b[2][2] = { { 1 }, { 3, 4 } };

8. Multidimensional Array



Two-dimensional array with three rows and four columns.

8. Multidimensional Array

- 二维数组参数
  - 第一维数组（行）的大小不是必须的
  - 第二维数组（列）的大小是必须的
  - 例： void printArray(const int a[][3]);

8. Multidimensional Array

- 二维数组的处理
  - ◈例：

```
total = 0;
for ( row = 0; row < 3; row++ )
    for ( col = 0; col < 4; col++ )
      total += a[ row ][ col ];
```

9. Case Study: Class GradeBook Using a Two-Dimensional Array

- Class GradeBook
  - 一维数组：存放一次考试多个学生的成绩
  - 二维数组：存放多个学生多次考试的成绩
    - 行代表一个学生多次考试的成绩
    - 列代表多个学生一次考试的成绩

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

   GradeBook( string, const int [][ tests ] );

   ......
   double getAverage( const int [], const int );

   ......
private:
   string courseName; // course name for this grade book
   int grades[ students ][ tests ]; // two-dimensional array of grades
};
```

```
GradeBook::GradeBook( string name, const int gradesArray[][ tests ] )
{
  setCourseName( name );

  // copy grades from gradeArray to grades
  for ( int student = 0; student < students; student++ )
    for ( int test = 0; test < tests; test++ )
      grades[ student ][ test ] = gradesArray[ student ][ test ];
}
```

```
int main()
{
  int gradesArray[ GradeBook::students ][ GradeBook::tests ] =
    { { 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++ Programming", gradesArray );
  myGradeBook.displayMessage();
  myGradeBook.processGrades();
  return 0;
}
```

10. C++ Standard Library Class Template vector

- C-style pointer-based arrays
  - 有以下不足之处
    - 没有越界检查
    - 两个数组之间不能进行比较和其他逻辑运算
    - 数组之间不能使用赋值运算符进行赋值

10. C++ Standard Library Class Template vector

- Class template vector
  - 可以用来定义各种数据类型
    - ◈vector< *type* >
    - ◈缺省的所有 vector 中的元素被初始化为 0
  - 成员函数 size 得到数组的长度
  - vector 对象之间可以进行比较和其他逻辑运算
  - vector 对象之间可以通过赋值运算符进行赋值

10. C++ Standard Library Class Template vector

- **vector 的成员函数 at**
  - 用来访问某一个元素
  - 执行边界检查
    - 当索引无效时抛出异常
    - 使用 “[]” 访问时不进行边界检查

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

void outputVector( const vector< int > & ); // display the vector
void inputVector( vector< int > & ); // input values into the vector

int main()
 {
   vector< int > integers1( 7 ); // 7-element vector< int >
   vector< int > integers2( 10 ); // 10-element vector< int >

   cout << "Size of vector integers1 is " << integers1.size()
      << "\nvector after initialization:" << endl;
   outputVector( integers1 );

   cout << "\nSize of vector integers2 is " << integers2.size()
      << "\nvector after initialization:" << endl;
   outputVector( integers2 );
```

```
cout << "\nEnter 17 integers:" << endl;
inputVector( integers1 );
inputVector( integers2 );

……

if ( integers1 != integers2 )
   cout << "integers1 and integers2 are not equal" << endl;

vector< int > integers3( integers1 ); // copy constructor

……

cout << "\nAssigning integers2 to integers1:" << endl;
integers1 = integers2; // integers1 is larger than integers2
```

```
   ......
   if ( integers1 == integers2 )
     cout << "integers1 and integers2 are equal" << endl;

   ......
   integers1[ 5 ] = 1000;

   ......
   cout << "\nAttempt to assign 1000 to integers1.at( 15 )" << endl;
   integers1.at( 15 ) = 1000; // ERROR: out of range
   return 0;
}
```

```
void outputVector( const vector< int > &array )
 {
   size_t i; // declare control variable

   for ( i = 0; i < array.size(); i++ )
   {
      cout << setw( 12 ) << array[ i ];
      ......
   } // end for
   ......
} // end function outputVector

void inputVector( vector< int > &array )
 {
   for ( size_t i = 0; i < array.size(); i++ )
      cin >> array[ i ];
} // end function inputVector
```

思考：

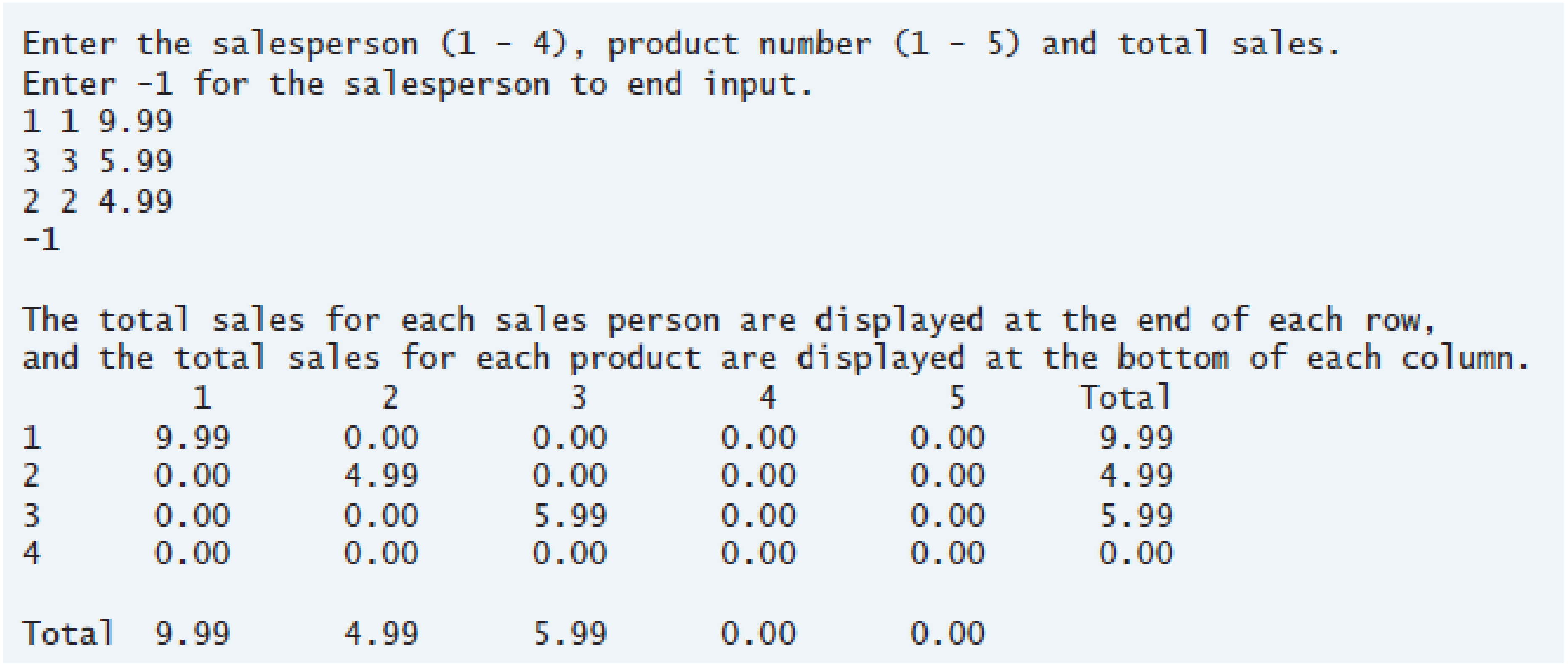
- 运行P274页，二维成绩数组的GradeBook类，观察打印结果
- 思考P277页，OutputGrades()函数的运行过程

思考题：

- 一家公司有4名销售员（1到4），他们销售5种不同产品（1到5），每名销售员在表格中记录每种售出的产品，每份表格包含以下信息：销售员编号、产品号、当天所售产品的总金额。
- 编写程序，根据产品和销售人员，将销售总额相加。在处理完所有销售信息后，以表格形式打印。其中列代表产品，行代表销售员。

思考题：

- 示例输出



```
Enter the salesperson (1 - 4), product number (1 - 5) and total sales.
Enter -1 for the salesperson to end input.
1 1 9.99
3 3 5.99
2 2 4.99
-1

The total sales for each sales person are displayed at the end of each row,
and the total sales for each product are displayed at the bottom of each column.
             1         2         3         4         5        Total
1         9.99      0.00      0.00      0.00      0.00        9.99
2         0.00      4.99      0.00      0.00      0.00        4.99
3         0.00      0.00      5.99      0.00      0.00        5.99
4         0.00      0.00      0.00      0.00      0.00        0.00

Total     9.99      4.99      5.99      0.00      0.00
```