第1章 绪 论

1.1 本章知识点

数据结构主要研究非数值计算的问题，即如何合理地组织数据、高效地处理数据。它是一门研究非数值计算程序设计中的操作对象以及它们之间的关系和操作的学科，是一门综合性的计算机专业基础课，前驱课程为程序设计语言与离散数学，后续课程有操作系统、数据库等。结构化程序设计使得人们认为程序设计的实质就是对所处理的问题选择一种合适的数据结构，并在此基础上设计一种好的算法（程序 = 数据结构 + 算法）。

基本概念：

数据（**Data**）——客观事物的符号表示，是所有能输入到计算机中并被计算机程序存储、加工处理的符号的总称。

数据元素（**Data element**）——数据的基本单位，也称结点（node）或记录（record）。

数据项（**Data item**）——组成数据元素的、有独立含义的、不可分割的数据最小单位，也称域（Field）。

数据对象（**Data Object**）——性质相同的数据元素的集合，是数据的一个子集。

数据结构（**Data Structure**）——相互之间存在一种或多种特定关系的数据元素的集合。

逻辑结构（**Logical structure**）——数据元素间抽象化的相互关系，与数据的存储无关，独立于计算机，是从具体问题抽象出来的数学模型。有以下类型：

集合（**Set** ）——数据元素间除“同属于一个集合”约束外，无其它关系；

线性结构（**Linear**）——一对一关系；

树形结构（**Tree**）——一对多关系；

图形结构（**Graph**）——多对多关系；

存储结构（**Storage structure**）——数据对象在计算机存储器中的存储表示，也称物理结构。主要有以下类型：

顺序存储结构（**Sequential storage structure**）——借助元素在存储器中的相对位置来表示数据元素间的逻辑关系；

链式存储结构（**linked storage structure**）——借助指示元素存储地址的指针表示数据元素间的逻辑关系；

数据类型（**Data type**）——一个值的集合以及定义在这个值集合上的一组操作的

总称。

抽象数据类型（**Abstract data type**）——一般指由用户定义的，用以表示应用问题的数学模型，以及定义在该模型上的一组操作的总称（数据对象 + 数据关系 + 基本操作）；

类 C 语言——伪码的一种，精选了 C 语言的一个核心子集，同时做了若干扩充修改，以增强语言的描述功能。

算法（**Algorithm**）——为了解决某类问题而规定的一个有限长的操作序列，必须满足 5 个特征：有穷性、确定性、可行性、输入、输出。评价算法优劣的标准有（算法设计的要求）：正确性、可读性、健壮性、高效性。

时间复杂度（**Time complexity**）——指程序运行从开始到结束所需要的时间度量 T（n）。但要精确地计算 T（n）是困难的，因此引入渐进时间复杂度在数量上估计一个算法的执行时间，使用大 O 记号表示的算法的时间复杂度，称为算法的渐进时间复杂度。通常用 O（1）表示常数计算时间。常见的渐进时间复杂度有：

$$O(1) < O(\log 2n) < O(n) < O(n\log 2n) < O(n^2) < O(n^3) < O(2^n)$$

空间复杂度（**Space complexity**）——指程序运行从开始到结束所需的辅助存储量 S（n）。

1.2 习题解析

1. 简述下列概念：数据、数据元素、数据项、数据对象、数据结构、逻辑结构、存储结构、抽象数据类型。

【解答】请参考 1.1 本章知识点

2. 试举一个数据结构的例子，叙述其逻辑结构和存储结构两方面的含义和相互关系。

【解答】如一个位数为 200 位的长整数，常用的程序设计语言都无对应的数据类型来定义它。采用数据结构的观点，可以把该位数为 200 位的长整数的看做是由 200 个 1 位十进制整数构成的序列，对应的逻辑结构就是线性结构，存储结构可以采用顺序存储结构（位数固定）或链式存储结构（位数不固定，如 210 位的长整数）。

3. 简述逻辑结构的四种基本关系并画出它们的关系图。

【解答】逻辑结构的四种基本关系为：

集合（Set）——数据元素间除“同属于一个集合”约束外，无其它关系；

线性结构（Linear）——一对一关系；

树形结构（Tree）——一对多关系；

图形结构（Graph）——多对多关系；

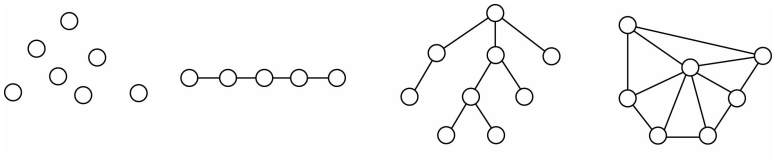


图 1 – 1　逻辑结构的四种基本形态

4. 存储结构由哪两种基本的存储方法实现?

【解答】存储结构可以由顺序存储结构和链式存储结构这两种基本的存储方法实现。

5. 选择题

(1) 在数据结构中，从逻辑上可以把数据结构分成（　　）。

A. 动态结构和静态结构　　B. 紧凑结构和非紧凑结构

C. 线性结构和非线性结构　　D. 内部结构和外部结构

【解答】选 C。数据结构的逻辑结构除了可以分为：集合、线性结构、树形结构、图形结构，还可以分为线性结构和非线性结构，其中非线性结构包含：集合、树形结构、图形结构。

(2) 与数据元素本身的形式、内容、相对位置、个数无关的是数据的（　　）。

A. 存储结构　　B. 存储实现

C. 逻辑结构　　D. 运算实现

【解答】选 C。存储结构是数据对象在计算机存储器中的存储表示，与数据元素本身的形式、内容、相对位置、个数均有关；存储实现实际也是存储结构的意思；运算实现即算法，也与问题规模即据元素的个数有关。

(3) 通常要求同一逻辑结构中的所有数据元素具有相同的特性，这意味着（　　）。

A. 数据具有同一特点

B. 不仅数据元素所包含的数据项的个数要相同，而且对应数据项的类型要一致

C. 每个数据元素都一样

D. 数据元素所包含的数据项的个数要相等

【解答】选 B。相同特性指的就是数据元素本身所包含的数据项的类型与个数。

(4) 以下说法正确的是（　　）。

A. 数据元素是数据的最小单位

B. 数据项是数据的基本单位

C. 数据结构是带有结构的各数据项的集合

D. 一些表面上很不相同的数据可以有相同的逻辑结构

【解答】选 D。数据元素是数据的基本单位，数据项是数据的最小单位，A 与 B 选项错误；数据结构是相互之间存在一种或多种特定关系的数据元素的集合，不是数据项的集合，C 选项错误。

(5) 以下与数据的存储结构无关的术语是（　　）。

A. 顺序队列　　B. 链表　　C. 有序表　　D. 链栈

【解答】选 C。存储结构指数据对象在计算机存储器中的存储表示，也称物理结构。主要有以下类型：顺序存储结构和链式存储结构。A 选项“顺序队列”为顺序存储结构；B 选项“链表”为链式存储结构；D 选项“链栈”为链式存储结构；C 选项“有序表”指的是按值大小排列而成的线性表，并未涉及数据的存储结构，因此选 C。

(6) 以下数据结构中，（　　）是非线性数据结构

A. 树　　B. 字符串　　C. 队　　D. 栈

【解答】选 A。非线性结构包含：集合、树形结构、图形结构。字符串、队、栈均为特殊的线性结构。

6. 试分析下面各程序段的时间复杂度。

（1）
```
x = 90; y = 100;
while(y > 0)
if(x > 100)
{x = x - 10; y - - ; }
else x + + ;
```

【解答】该程序段的各语句执行次数是固定的，与问题规模 n 无关，因此时间复杂度为：O（1）。

（2）
```
for (i = 0; i < n; i + + )
    for (j = 0; j < m; j + + )
        a[i][j] = 0;
```

【解答】该程序段的 a［i］［j］ =0 语句执行了 n * m 次，算法执行时间为 T（n） = n * m，因此时间复杂度为：O（n * m）。

（3）
```
s = 0;
for i = 0; i < n; i + + )
  for(j = 0; j < n; j + + )
    s + = B[i][j];
sum = s;
```

【解答】该程序段的 s = 0 和 sum = s 各执行了一次，s + = B［i］［j］语句执行了 n * n 次，算法执行时间为 T（n） =1 + n * n + 1，因此时间复杂度为：O（n * n）即 O（n^2）。

（4）
```
i = 1;
while(i < = n)
    i = i*3;
```

【解答】该程序段的 i = 1 执行了一次，i = i * 3 语句执行了 $\log_3 n$ 次，算法执行时间为 T（n） =1 + $\log_3 n$，因此时间复杂度为：O（$\log_3 n$）。

（5）
```
x = 0;
for(i = 1; i < n; i + + )
  for (j = 1; j < = n - i; j + + )
    x + + ;
```

【解答】该程序段的 x = 0 执行了一次，x + + ；语句执行了 $\sum_{i=1}^{n}\sum_{j=1}^{n-i}1 = n(n-1)/2$ 次，算法执行时间为 T（n） =1 + n（n - 1）/2，因此时间复杂度为：O（n^2）。

（6）
```
x = n; //n > 1
y = 0;
while(x≥(y + 1)*(y + 1))
```

```
y++;
```

【解答】该程序段的x=n；y=0；各执行了一次，y++语句执行了$\sqrt{n-1}$次，算法执行时间为T（n）$=2+\sqrt{n-1}$，因此时间复杂度为：O（$\sqrt{n}$）。

1.3 自测题

1. 填空题

（1）_____是数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理。

（2）_____指的是解决问题的有限运算序列。

（3）从逻辑关系上讲，数据结构主要分为_____、_____、_____和_____。

（4）数据的存储结构主要有_____和_____两种基本方法，不论哪种存储结构，都要存储两方面的内容：_____和_____。

（5）算法具有五个特性，分别是_____、_____、_____、_____、_____。

（6）算法的描述方法通常有_____、_____、_____和_____四种。

（7）在一般情况下，一个算法的时间复杂度是_____的函数。

（8）设待处理问题的规模为n，若一个算法的时间复杂度为一个常数，则表示成数量级的形式为_____，若为$n*\log_2 5n$，则表示成数量级的形式为_____。

（9）算法在发生非法操作时可以作出处理的特性称为_____。

2. 选择题

（1）顺序存储结构中数据元素之间的逻辑关系是由（　　）表示的，链接存储结构中的数据元素之间的逻辑关系是由（　　）表示的。

A. 线性结构　　B. 非线性结构　　C. 存储位置　　D. 指针

（2）假设有如下遗产继承规则：丈夫和妻子可以相互继承遗产；子女可以继承父亲或母亲的遗产；子女间不能相互继承。则表示该遗产继承关系的最合适的数据结构应该是（　　）。

A. 树　　B. 图　　C. 线性表　　D. 集合

（3）算法指的是（　　）。

A. 对特定问题求解步骤的一种描述，是指令的有限序列

B. 计算机程序

C. 解决问题的计算方法

D. 数据处理

（4）下面（　　）不是算法所必须具备的特性。

A. 有穷性　　B. 确切性　　C. 高效性　　D. 可行性

（5）算法分析的目的是（　　），算法分析的两个主要方面是（　　）。

A. 找出数据结构的合理性　　B. 研究算法中输入和输出的关系

C. 分析算法的效率以求改进　　D. 分析算法的易读性和文档性

E. 空间性能和时间性能　　F. 正确性和简明性

G. 可读性和文档性　　H. 数据复杂性和程序复杂性

3. 判断题

（　　）（1）算法的时间复杂度都要通过算法中的基本语句的执行次数来确定。

（　　）（2）每种数据结构都具备三个基本操作：插入、删除和查找。

（　　）（3）所谓数据的逻辑结构指的是数据之间的逻辑关系。

（　　）（4）逻辑结构与数据元素本身的内容和形式无关。

（　　）（5）基于某种逻辑结构之上的基本操作，其实现是唯一的。

4. 分析以下各程序段，并用大 O 记号表示其执行时间。

（1）
```
i = 1; k = 0
while (i < n - 1)
{
  k = k + 10*i;
  i + +;
}
```

（2）
```
i = 1; k = 0;
do
{
  k = k + 10*i;
  i + +;
}while(i < = n)
```

（3）
```
i = 1; j = 0
while(i + j < = n)
  ij(i > j)j + +;
    else i + +;
```

（4）
```
y = 0;
while((y + 1)*
  y = y + 1;
```

（5）
```
for(i = 1; i < = n; i + +)
  for(j = 1; j < = i; j + +)
    for(k = 1; k < = j; k + +)
      x + +;
```

5. 设有数据结构（D，R），其中 D = {1，2，3，4，5，6}，R = {（1，2），（2，3），（2，4），（3，4），（3，5），（3，6），（4，5），（4，6）}。试画出其逻辑结构图并指出属于何种结构。

6. 将下列函数按它们在 n 时的无穷大阶数，从小到大排列。

n，$n-n^3+7n^5$，$n\log n$，$2^{n/2}$，n^3，$\log_2 n$，$n^{1/2}+\log_2 n$，$(3/2)^n$，$n!$，$n^2+\log_2 n$

7. 对下列用二元组表示的数据结构，试分别画出对应的逻辑结构图，并指出属于何种结构。

（1）A = （D，R），其中 D = {a1，a2，a3，a4}，R = { }

（2）B = （D，R），其中 D = {a，b，c，d，e，f}，R = {（a，b），（b，c），（c，d），（d，e），（e，f）}

（3）C = （D，R），其中 D = {a，b，c，d，e，f}，R = {（d，b），（d，g），（b，a），（b，c），（g，e），（g，h）}

（4）D = （D，R），其中 D = {1，2，3，4，5，6}，R = {（1，2），（1，4），（2，3），（2，4），（3，4），（3，5），（3，6），（4，6）}

自测题参考答案

1. 填空题答案

（1）数据元素；

（2）算法；

（3）集合，线性结构，树结构，图结构；

（4）顺序存储结构，链接存储结构，数据元素，数据元素之间的关系；

（5）输入，输出，有穷性，确定性，可行性；

（6）自然语言，程序设计语言，流程图，伪代码；

（7）问题规模；

（8）O（1），O（$n\log_2 n$）；

（9）健壮性

2. 选择题答案

（1）C，D

（2）B

（3）A

（4）C

（5）C，E

3. 判断题答案

（1）错。时间复杂度要通过算法中基本语句执行次数的数量级来确定。

（2）错。如数组就没有插入和删除操作。此题注意是每种数据结构。

（3）对。

（4）对。

（5）错。基本操作的实现是基于某种存储结构设计的，因而不是唯一的。

4.（1）T（n）=O（n）。

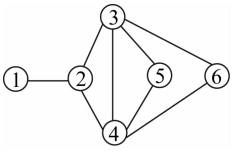
（2）T（n）=O（n）。

（3）T（n）=O（n）。

（4）T（n）=O（$\sqrt{n}$）。

（5）T（n）=O（n^3）。

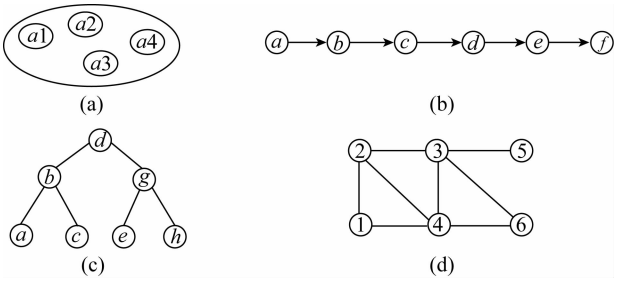
5. 其逻辑结构图如下图所示，它是一种图结构。



6. $\log_2 n$，$n^{1/2}+\log_2 n$，n，$n\log_2 n$，$n^2+\log_2 n$，n^3，$n-n^3+7n^5$，$2^{n/2}$，$(3/2)^n$，$n!$

7.（1）属于集合，其逻辑结构图如（a）所示；

（2）属于线性结构，其逻辑结构图如（b）所示；
（3）属于树结构，其逻辑结构图如（c）所示；
（4）属于图结构，其逻辑结构图如（d）所示。



1.4 算法实现

【算法 1.1】求两个 n 阶矩阵的乘积

```
//求两个 3 阶矩阵的乘积
#include <iostream>
using namespace std;

void matproduct(int a[][3], int b[][3], int c[][3], int n)
{
    for(int i = 0; i < n; ++i)
        for(int j = 0; j < n; ++j)
        {
            c[i][j] = 0;
            for(int k = 0; k < n; ++k)
                c[i][j] = c[i][j] + a[i][k] *b[k][j];
        }
}
int main()
{
    int a[3][3] = {{1, 1, 1}, {2, 2, 2}, {3, 3, 3}};
    int b[3][3] = {{4, 4, 4}, {4, 5, 5}, {6, 6, 6}};
    int c[3][3] = {0};
    matproduct(a, b, c, 3);
    for(int i = 0; i < 3; ++i)
```

```
        {
            for(int j = 0; j < 3; + + j)cout < < c[i][j] < < '\t';
            cout < < endl;
        }
        return 0;
    }
```

【算法 1.2】交换 x 和 y 的值

```
//交换 x 和 y 的值
#include < iostream >
using namespace std;

void main()
{
    int x, y, temp;
    cout < < "请输入 x 和 y 的值,用空格隔开:";
    cin > > x > > y;
    temp = x;
    x = y;
    y = temp;
    cout < < "x = " < < x < < "  y = " < < y < < endl;
}
```

【算法 1.3】变量计数之一

```
//变量计数之一
#include < iostream >
using namespace std;

void main()
{
    int x = 0, y = 0, n;
    cout < < "请输入 n 的值:";
    cin > > n;
    for(int k = 1; k < = n; k + + )
        x + + ;
    for(int i = 1; i < = n; i + + )
        for(int j = 1; j < = n; j + + )
            y + + ;
    cout < < "x = " < < x < < "  y = " < < y < < endl;
}
```

【算法 1.4】变量计数之二

```
//变量计数之二
#include <iostream>
using namespace std;

void main()
{
    int x = 1, n;
    cout << "请输入 n 的值:";
    cin >> n;
    for(int i = 1; i <= n; i ++)
        for(int j = 1; j <= n; j ++)
            for(int k = 1; k <= n; k ++)
                x ++;
    cout << "x = " << x << endl;
}
```

【算法 1.5】顺序查找

```
//顺序查找
#include <iostream>
using namespace std;

int Search(int a[], int n, int e)
{
    for(int i = 0; i < n; i ++)
        if(a[i] == e)return i + 1;
    return 0;
}
void main()
{
    int a[10] = {1,2,3,4,5,6,7,8,9,10};
    int e, pos;
    cout << "请输入要查找的元素的值:";
    cin >> e;
    pos = Search(a,10, e);
    if (pos == 0)cout << "查找失败!" << endl;
    else cout << "元素 " << e << " 在第" << pos << "位置。" << endl;
}
```

【算法 1.6】复数的抽象数据类型实现

```
//复数的运算实现
#include <iostream>
using namespace std;

typedef struct Complex{
    float Realpart;
    float Imagepart;
}Complex;

void Creat(Complex &C,float x,float y)
{
    C.Realpart = x;
    C.Imagepart = y;
}

float GetReal(Complex C)
{
    return C.Realpart;
}

float GetImage(Complex C)
{
    return C.Imagepart;
}

Complex Add(Complex C1, Complex C2)
{
    Complex sum;
    sum.Realpart = C1.Realpart + C2.Realpart;
    sum.Imagepart = C1.Imagepart + C2.Imagepart;
    return sum;
}

Complex Sub(Complex C1, Complex C2)
{
```

```
    Complex difference;
    difference. Realpart = C1. Realpart - C2. Realpart;
    difference. Imagepart = C1. Imagepart - C2. Imagepart;
    return difference;
}

void main()
{
    Complex C1,C2,sum,difference;
    Creat(C1, 3, 4);
    Creat(C2, 2, 3);
    sum = Add(C1, C2);
    difference = Sub(C1, C2);
    cout < < "复数 3 + 4i 与 2 + 3i 的和是:" < < GetReal(sum) < < " + " < < GetImage
            (sum) < < "i" < < endl;
    cout < < "复数 3 + 4i 与 2 + 3i 的差是:" < < GetReal(difference) < < " + " < < GetIm-
            age(difference) < < "i" < < endl;
}
```

1.5 实验 1：预备实验——C 语言程序编程、调试的方法与结构体的应用

1.5.1 实验目的与内容

目的：

（1）掌握 C 语言程序的编写规则与结构体的定义与应用方法；

（2）掌握 Visual C + + 6.0 编译器的使用方法；

（3）掌握算法时间复杂度的分析方法。

内容：

（1）使用 Visual C + + 6.0 编译器验证 1.4 算法实现中的【算法 1.1】求两个 n 阶矩阵的乘积、【算法 1.2】交换 x 和 y 的值、【算法 1.3】变量计数之一、【算法 1.4】变量计数之二、【算法 1.5】顺序查找、【算法 1.6】复数的抽象数据类型实现

（2）编译或运行出错则利用 Visual C + + 6.0 编译器调试程序的方法找出错误所在；

（3）分析【算法 1.1】求两个 n 阶矩阵的乘积、【算法 1.2】交换 x 和 y 的值、【算法 1.3】变量计数之一、【算法 1.4】变量计数之二、【算法 1.5】顺序查找 这个五个程序是时间复杂度。

1.5.2　实验步骤

（1）实验准备：复习 C 语言课程相关编程方法；

（2）双击桌面上的 Visual C + + 6.0 快捷图标或从开始菜单找到 Visual C + + 6.0，打开编程环境，如下图所示：

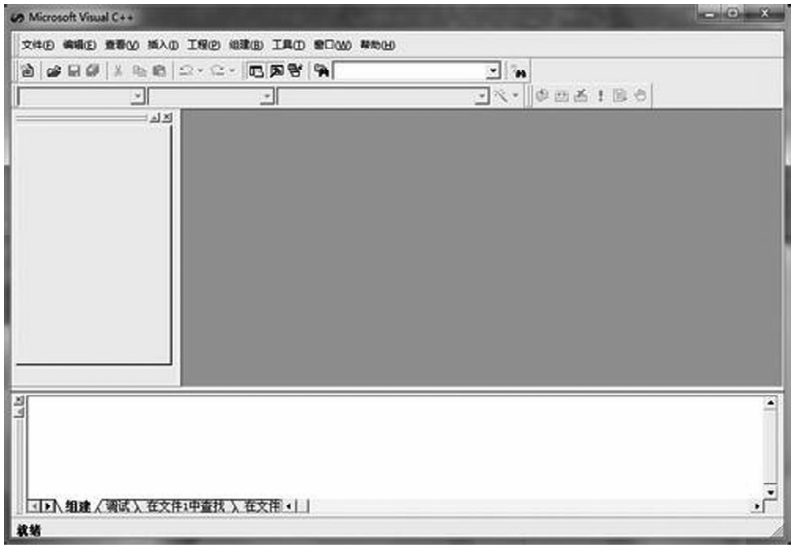


图 1－2　Visual C + + 6.0 编程环境

（3）在打开的界面中，单击“文件”菜单项。选择里面的子菜单“新建”，单击此项，得到如下图所示的选项卡（也可以直接使用快捷键：Ctrl + N）：

图 1－3　Visual C + + 6.0 新建菜单项

（4）在打开的新建选项卡中，单击“文件”，然后选中“文件”选项卡中的“C + + source file”一栏，在右边的“文件名”文本框中，对要新建的 C + + 源文件命名，如“Demo. cpp”，具体详细操作如下图所示：

图 1 - 4　Visual C + + 6.0 新建“文件”菜单项

（5）在上述操作的基础上，保存文件到指定的文件夹。如保存到桌面的 Demo 文件夹中，如下图所示：

图 1 - 5　Visual C + + 6.0 存储文件选择目录对话框

（6）选择路径之后，单击“确定”按钮，得到如下图所示界面，在右边的编辑区内，您就可以开始编写 C/ + + 语言代码了。

图 1 - 6 Visual C + + 6.0 代码编辑区

（7）开始代码的编辑。下面，编写【算法 1.1】求两个 n 阶矩阵的乘积的 C 语言代码，代码编辑如下如所示：

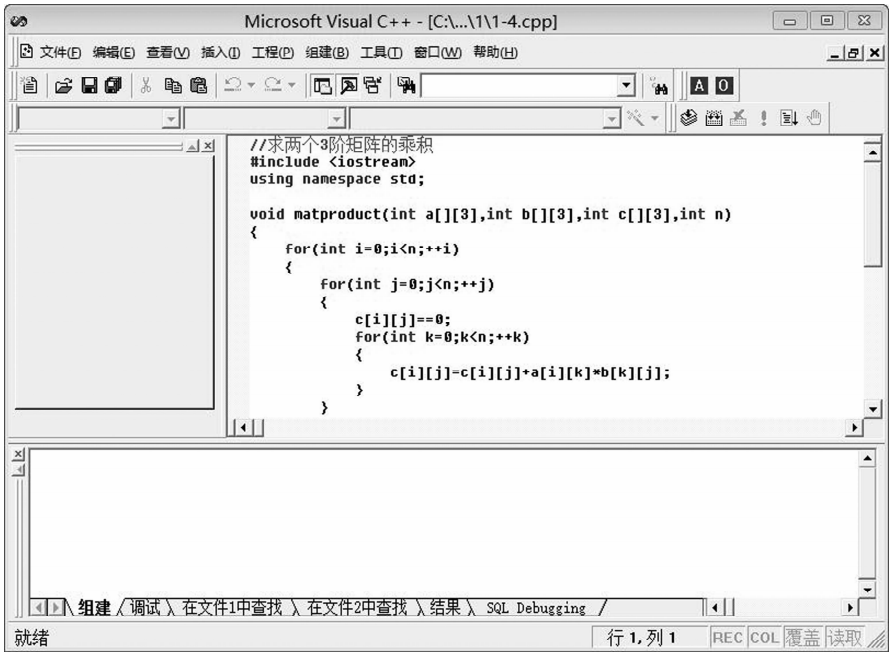


图 1 - 7 Visual C + + 6.0 代码编写完成

（8）编译。对程序进行编译，点击工具栏的“编译”图标（或者按下快捷键：Ctrl + F7），如下图所示：

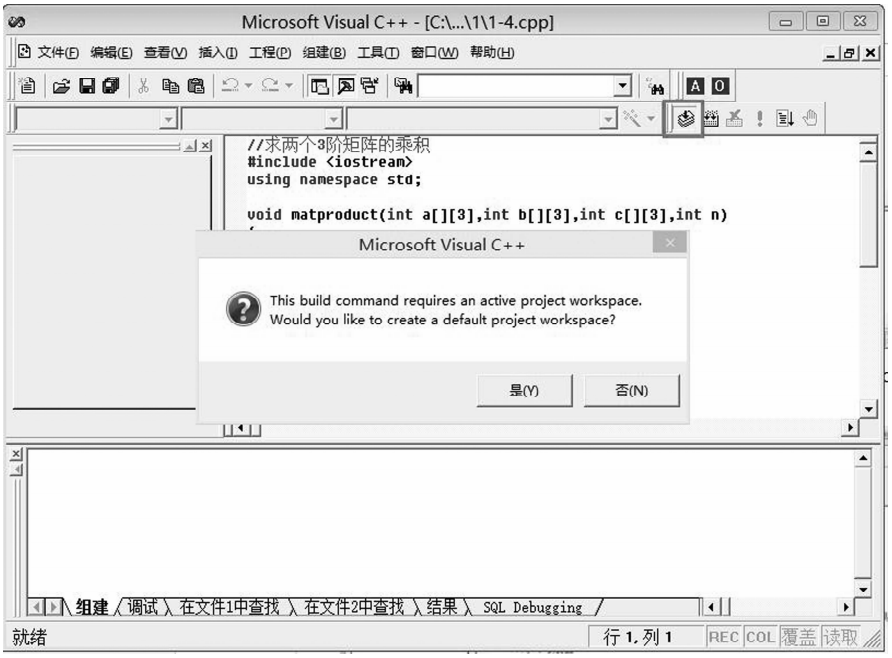


图 1－8　Visual C + + 6.0 编译

（9）组建。对程序进行链接，点击工具栏的“组建”图标（或者按下快捷键：F7），如下图所示：

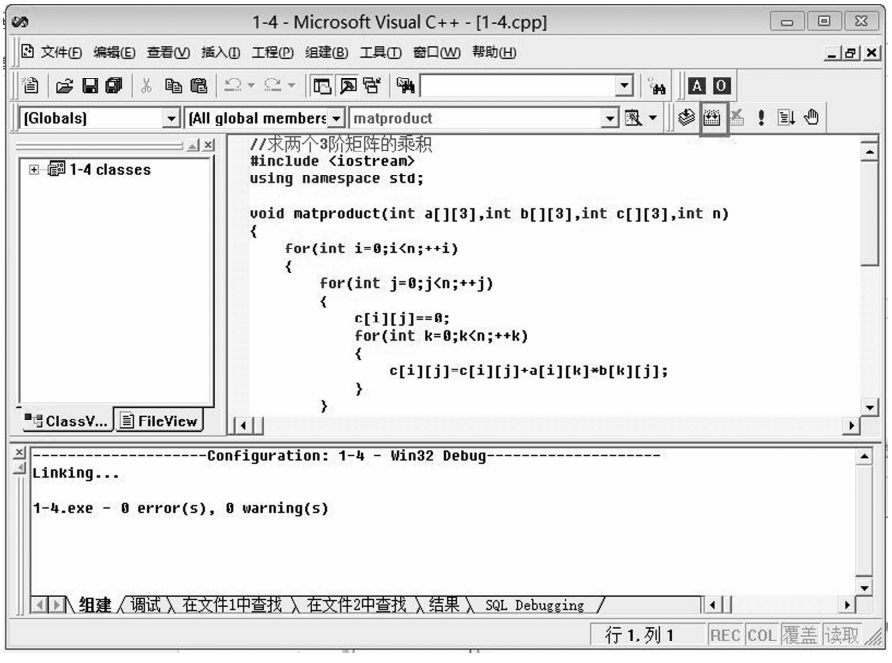


图 1－9　Visual C + + 6.0 组建

（10）执行。接下来可以查看程序的运行结果了，点击工具栏中的“执行”图标（或者按下快捷键：Ctrl + F5），程序执行的结果如下图所示：

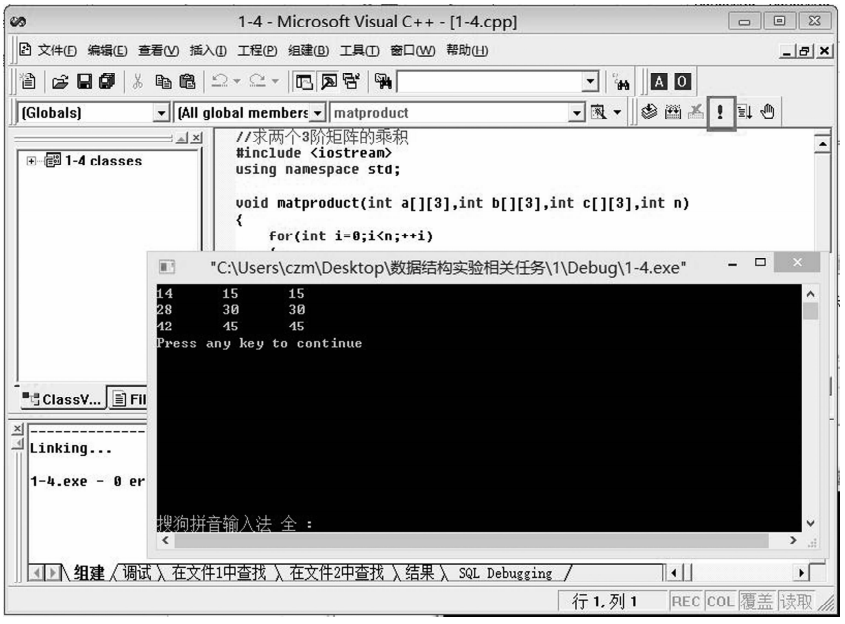


图 1－10　Visual C + + 6.0 执行

（11）当程序编译错误时，可通过编译器下部的信息区查看相应的编译错误（主要是C 程序的语法错误），从上往下逐条定位，订正，再执行编译（Ctrl + F7），直到无错误为止，如下图所示（程序里的“c［i］［j］ = =0”语句后忘记输入“;”）：

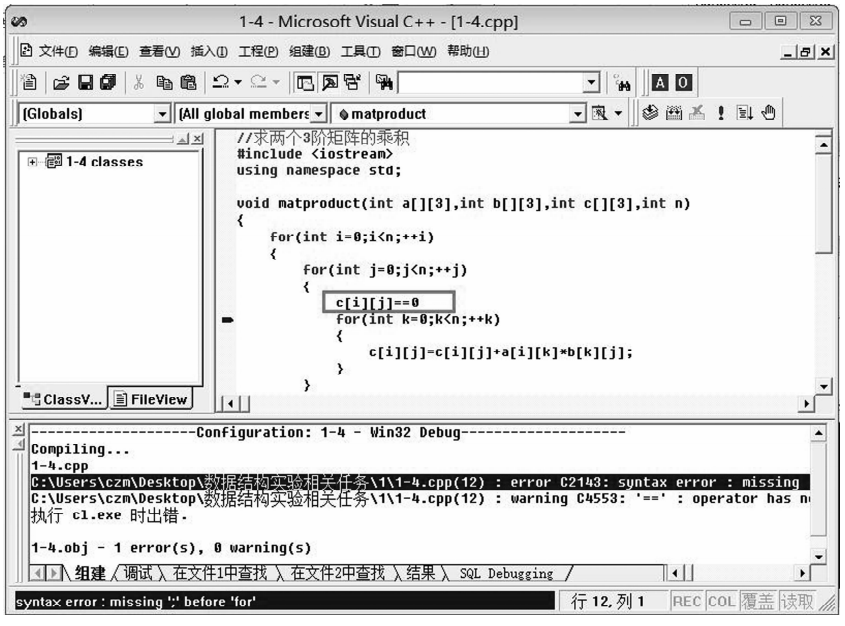


图 1－11　Visual C + + 6.0 检查编译错误

（12）如果能出运行窗口，但运行结果不正确，可进入 Debug（调试）模式，逐行执行程序：

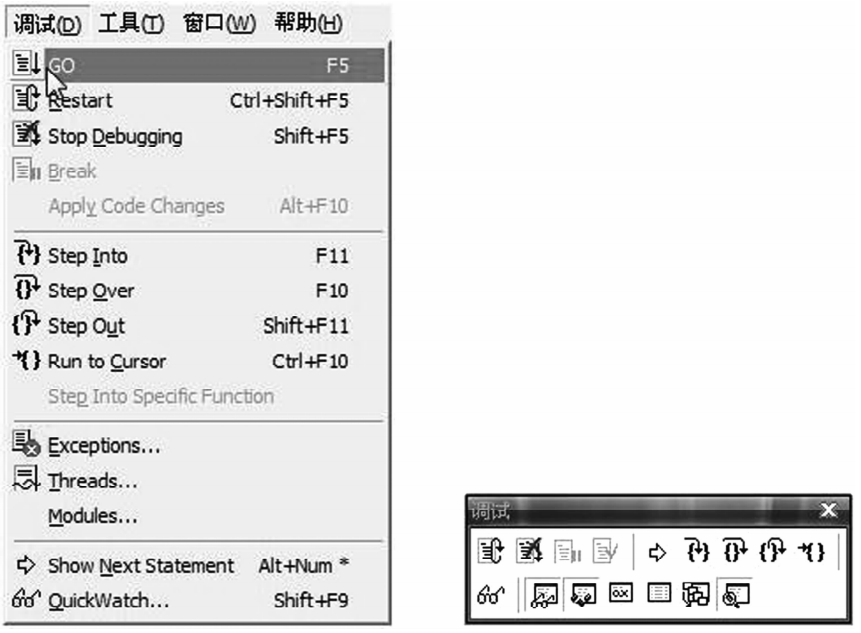


图 1－12　Visual C＋＋ 6.0 调试菜单与快捷工具栏

然后在执行窗口或者 Visual C＋＋ 6.0 编译器内存监控区查看每次执行的结果，直到发现错误语句修改正确为止。

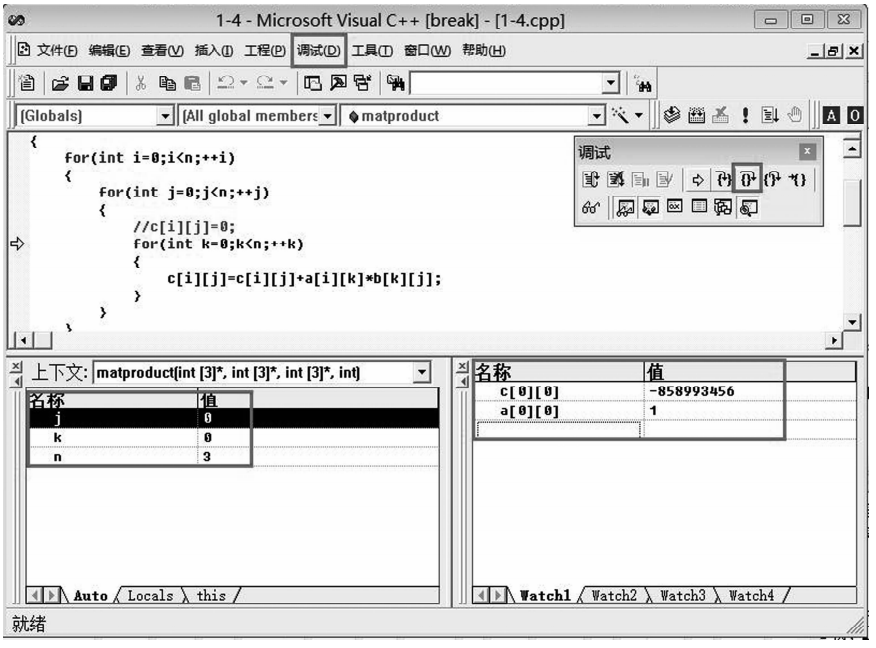


图 1－13　Visual C＋＋ 6.0 监控区

如上图所示，就是程序没有对 c［i］［j］进行初始化为 0 的操作，虽然可运行程序，但是结果是错误，使用调试模式，单步执行到箭头所示语句，发现 i，j，k 变量变化正确，然后在右侧 Watch1 检测区填入 c［0］［0］，发现值为一个未被初始化的数，从而发现程序未初始化的程序逻辑错误。

常用功能：

- **Restart**（Ctrl + shift + F5）：此 debugger 功能将从程序的开始（第一有效行）处全速执行，而不是从当前所跟踪的位置开始调试，这时所有变量的当前值都将被丢弃，debugger 会自动停在程序的 main（）开始处。这时如果选择 **Step Over**（F10）就可以逐步执行 main（）函数了。
- **Stop Debugging**（Shift + F5）：此 debugger 功能将终止（所有）调试，并返回到常规编辑状态。
- **Break**：此功能将在调试过程中的 debugger 当前位置挂起程序的执行，然后就可以在调试状态一修改程序的代码，接着可以用 **Apply Code Changes**（Alt + F10）来应用修改的代码到正在调试的程序当中。如果，当前（需要，待）可以（从 DOS 等窗口）输入值，挂起后将不能再输入。
- **Apply Code Changes**（Alt + F10）：此功能可以在程序正在调试程序过程中应用（挂起）修改后的源代码．如，选择 **Break** 功能并修改代码后，只要选择 **Apply Code Changes**（Alt + F10）就能将修改后的代码应用到正在调试的程序当中。
- **Show Next Statement**（Alt + Num *）：此功能将显示程序代码的下一条语句，如果源代码中找不到，则在 Disassembly 窗口中显示语句。（当在 Disassembly 窗口中显示时，可以单击 Disassembly 返回到源代码窗口。）
- **Step Into**（F11）：此功能可以单步进入到在调试过程中所跟踪的调用函数的语句的函数内部．如，当前语句是"d. Display（）"，选择 **Step Into**（F11）后，Debugger 将进入 Display（）函数内部并停在 Display（）函数内部的第一条语句上。（此时，就可以 **Step Over**（F10）对 Display（）函数进行单步调试了。）
- **Step Over**（F10）：此功能可以单步对所在函数单步调试，如果调试的语句是一个调用函数的语句时，Debugger 将全速执行所调用的函数，单步（一步）通过所调用的函数，Debugger 停该调用语句的下一条语句上。
- **Step Out**（Shift + F11）：此功能将使 Debugger 切换回全速执行到被调用函数结束，并停在该函数调用语句的下一条语句上。当确定所调用的函数没有问题时可以用这个功能全速执行被调用函数。
- **Run to Cursor**（Ctrl + F10）：此功能将全速执行到包含插入点光标所在的行，可以作为在插入点光标处设置常规断点的一种选择。（注意，当光标处不是一个有效的执行语句时此功能将不起作用。）
- **Go**（F5）：此功能将全速执行程序直到遇到一个断点或程序结束，或直到程序暂停等待用户输入。注意，此功能最能有效的调试循环，常将断点设置在循环体内，重复的按 F5 全速执行循环体可以测试循环过程中的产生的变化。

● Step Into Specific Function：此功能可以可以单步通过程序中的指令，并进入指定的函数调用，此功能对于函数的嵌套层不限。

（13）关闭当前的工作空间，然后继续新建“C + + Source File”，输入【算法 1.2】交换 x 和 y 的值、【算法 1.3】变量计数之一、【算法 1.4】变量计数之二、【算法 1.5】顺序查找、【算法 1.6】复数的抽象数据类型实现的示例程序进行验证。

（14）分析上述验证的这五个程序（【算法 1.1】～【算法 1.5】）的时间复杂度。

（15）撰写实验报告。

1.5.3 实验报告

_____学院实验报告（软件类）

实验名称	实验 1 预备实验——C 语言程序编程、调试的方法；结构体的应用	实验类型	基础 □ 设计 □ 综合 □	实验日期	年 月 日

实验目的：

（1）掌握 C 语言程序的编写规则与结构体的定义与应用方法；

（2）掌握 Visual C + + 6.0 编译器的使用方法。

（3）掌握算法时间复杂度的分析方法。

实验内容：

（1）使用 Visual C + + 6.0 编译器验证 1.4 算法实现中的【算法 1.1】求两个 n 阶矩阵的乘积、【算法 1.2】交换 x 和 y 的值、【例 1.3】变量计数之一、【例 1.4】变量计数之二、【算法 1.5】顺序查找、【算法 1.6】复数的抽象数据类型实现；

（2）编译或运行出错则利用 Visual C + + 6.0 编译器调试程序的方法找出错误所在；

（3）分析【算法 1.1】求两个 n 阶矩阵的乘积、【算法 1.2】交换 x 和 y 的值、【例 1.3】变量计数之一、【例 1.4】变量计数之二、【算法 1.5】顺序查找 这个五个程序的时间复杂度。

实验过程及分析：

一、实验预习与准备

1. 参考资料：

2. 软硬件环境：win 7 + Visual C + + 6.0

3. 相关知识点：

二、编写程序（只需写出最主要的核心代码，不足地方可另加纸写）

【算法 1.1】求两个 n 阶矩阵的乘积

```
for(int i = 0; i < n; + + i)
    for(int j = 0; j < n; + + j)
    {
        c[i][j] = 0;
        for(int k = 0; k < n; + + k)
            c[i][j] = c[i][j] + a[i][k]*b[k][j];
    }
```

【算法 1.2】交换 x 和 y 的值

……

【算法 1.3】变量计数之一

……

续表

实验名称	实验1 预备实验——C语言程序编程、调试的方法；结构体的应用	实验类型	基础 □ 设计 □ 综合 □	实验日期	年 月 日

【算法1.4】变量计数之二
【算法1.5】顺序查找
【其他】复数的抽象数据类型实现

三、编译运行程序（不足地方可另加纸写）

序号	出错表现或提示	原因	解决方法
1	error C2143：syntax error ：missing ‘;’ before ‘for’	漏写语句结束符“;”	在c［i］［j］=0语句后补上“;”
2			
3			
4			
5			

四、运行结果与时空复杂度分析

【算法1.1】求两个n阶矩阵的乘积、【算法1.2】交换x和y的值、【算法1.3】变量计数之一、【算法1.4】变量计数之二、【算法1.5】顺序查找、【算法1.6】复数的抽象数据类型实现 六个程序在VC6.0中编写完成，并通过编译，运行结果正确。

时空复杂度分析结果：

【算法1.1】求两个n阶矩阵的乘积

$T(n)=O(n^3)$

【算法1.2】交换x和y的值

$T(n)=O(\ )$

【算法1.3】变量计数之一

$T(n)=O(\ )$

【算法1.4】变量计数之二

$T(n)=O(\ )$

【算法1.5】顺序查找

$T(n)=O(\ )$

实验体会：

教师评语及成绩：

实验指导教师签名：

年 月 日