# 《数据结构与 STL 》

## 教师手册

ver 0.1

肖波 徐雅静 莫骁 肖晶

2010 年 9 月

目录

习题 1.....
习题 2.....
习题 3.....
习题 4.....
习题 5.....
习题 6.....
习题 7.....
习题 8.....

习题 1

1. 填空题

（1）（_____）是指数据之间的相互关系，即数据的组织形式。通常人们认为它包含三个方面的内容，分别为数据的（_____）、（_____）及其运算。

答案：数据结构 逻辑结构存储结构

（2）（_____）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处理。

答案：数据元素

（3)数据元素之间的不同逻辑关系代表不同的逻辑结构,常见的逻辑结构有(_____)、(_____)、(_____）和（_____)。

答案：集合 线形结构 树结构图结构

（4）数据的存储结构考虑的是如何在计算机中存储各个数据元素，并且同时兼顾数据元素间的逻辑关系。基本的存储结构通常有两大类：（_____）和（_____)。

答案：顺序存储结构 链式存储结构

（5）通常一个问题可以有多种不同的算法，但每个算法必须满足 5 个准则：输入、输出、(_____)、(_____）和（_____)。

答案：有穷性 确定性 可行性

（6）通常通过衡量算法的（_____）复杂度和（_____）复杂度来判定一个算法的好坏。

答案：时间 空间

（7）常见时间复杂性的量级有：常数阶 O（_____)、对数阶 O（_____)、线性阶 O（_____)、线性对数阶 O（_____)、平方阶 O（_____)、和指数阶 O（_____)。通常认为，当问题规模较大时，具有（_____）量级的算法是不可计算的。

答案：1 logn n nlogn n2 指数

（8）STL 提供的标准容器有顺序容器、(_____）和（_____)。

答案：排序容器 哈希容器

（9）算法可认为是 STL 的精髓，所有算法都是采用（_____）的形式提供的。

答案：函数模版

（10)通常认为 STL 由空间管理器、迭代器、泛函、适配器、(_____)和(_____)等六部分构成，其中前面四部分服务于后面两部分。

答案：容器 算法

2. 选择题

（1）以下结构中，（ ）属于线性结构。

A. 树 B. 图 C. 串 D. 集合

（2）算法描述的方法有很多种，常常将（ ）称为算法语言。

A. 自然语言 B. 流程图 C. 伪代码 D. 程序设计语言

（3）现实生活中的家族谱，可认为是一种（ ）结构。

A. 树 B. 图 C. 集合 D. 线性表

（4）手机中存储的电话号码簿，可认为是一种（ ）结构。

A. 树 B. 图 C. 集合 D. 线性表

（5）NP 问题是（ )。

A. 非多项式时间问题，即非 P 问题 B. 非确定性多项式时间问题

C. P 问题的子集 D. 与 P 问题不相交的

（6）以下（ ）不属于 STL 的顺序容器。

A. 向量（vector） B. 列表（list） C. 队列（queue） D.双端队列（deque）

（7）下面带有@ 标记的语句的频度(n>10)是（ ）。

```
for(int i=0;i<n-1;i++)
for(int j=i+1;j<n;j++)
@cout<<i<<j<<endl;
```

A. n*(n-1)/2 B. n*n/2 C. n*(n+1)/2 D.不确定

分析：

```
2 1 2
0 1 0
1 1 ( 1)
2
n n n
i j i i
  .
n i n n
. . .
= = + =
.
Σ Σ =Σ . . =
```

3. 分析以下程序段的时间复杂度。

（1）

```
for (i=l; i<=n; i++) {
k++;
for (j=1; j<=n; j++)
m += k;
}
```

（2）

```
for (i=l; i<=n; i++)
k++;
for (j=1; j<=n; j++)
m += k;
```

（3）

```
i=1;
while (i<=n)
i *= 2;
```

（4）

```
i=1;
while (i<=n)
i += 2;
```

（5）

```
k=100,i=10;
do {
if (i<n) break;
i++;
}while (i<k);
```

（6）

```
for (i=0; i<100; i++)
for (j=0; j<i; j++)
sum += j;
```

（7）

```
y=0;
while (y*y*y <= n)
```

```
y++;
 (8) int i=0;
while(i<n && a[i]!=k) i++;
return i==n?-1:i ;
```

5

答案:

(1) O(n2)

(2) O(n)

(3) O(logn)

(4) O(n)

(5) O(1)

(6) O(1)

(7) O(n1/3)

(8) O(n)

4. 将整数设计为一个类，将整数相关的常见数学运算设计为类的接口并进行实现，如求与给定值的最大公约数、最小公倍数、枚举所有因子等。

解:

```
#include "math.h"
#include "vector"
using std::vector;
/定义自然数类
class NaturalNumber{
public:
NaturalNumber(unsigned long int n=0):num(n){}
unsigned long int GreatestCommonDivisor(NaturalNumber &n);//求解最大公约数
unsigned long int LeaseCommonMultiple(NaturalNumber & n);//求解最小公约数
int GetFactors(vector <unsigned long int> & factors);//求所有因子，存储在 factors
中，函数返回因子个数
unsigned long int GetNumber(){return num;}
//„„其它外部接口
private:
unsigned long int EUCLID(NaturalNumber & nn);//欧几里德算法求解最大公约数
unsigned long int num;//存储真正的自然数
};
/返回欧几里德算法求解最大公约数
unsigned long int NaturalNumber :: EUCLID(NaturalNumber & nn)
{
unsigned long int m = (num > nn.num) ? num : nn.num;//较大的自然数赋值给 m
unsigned long int n = (num < nn.num) ? num : nn.num;//较小的自然数赋值给 n
unsigned long int r = m % n;
while (r != 0){
m = n; n = r; r = m % n;
}
return n;
```

```
}
/返回最大公约数
unsigned long int NaturalNumber :: GreatestCommonDivisor(NaturalNumber & nn)
6
{
return EUCLID(nn);
}
/返回最小公倍数
unsigned long int NaturalNumber :: LeaseCommonMultiple(NaturalNumber & nn)
{
unsigned long int temp = EUCLID(nn);
return num * nn.GetNumber() / temp;
}
int NaturalNumber :: GetFactors( vector <unsigned long int> & factors )
{
int t=0;
int m = sqrt((double)num);
vector <unsigned long int> bigfactors;
for (unsigned long int i=1;i<m;i++)
{
if ( 0 == num%i ) {t+=2; factors.push_back(i);bigfactors.push_back(num/i);}
}
if ( m*m == num ) { t++; factors.push_back(m);}
vector <unsigned long int> ::iterator it=bigfactors.end();
if (bigfactors.size()) do
{
it--;
factors.push_back(*it);
}while (it!=bigfactors.begin());
return t;
}
void main()
{
NaturalNumber p(1);
int xx = p.GreatestCommonDivisor(NaturalNumber(120));
int yy = p.LeaseCommonMultiple(NaturalNumber(120));
vector <unsigned long int> f;
int t = p.GetFactors(f);
}
```

7

习题 2

1. 填空题

（1）在一个单链表中，已知每个结点包含 data 和 next 两个域，q 所指结点是 p 所指结点的

直接前驱，若在 q 和 p 之间插入 s 所指结点，则执行（_____）和（_____）操
作。
答案：q->next = s; s->next 或ps->next=q->next; q->next = s;
（2）表长为 n 的顺序表，当在任何位置上插入或删除一个元素的概率相等时，插入一个元素所需移动元素的平均个数为（_____），删除一个元素需要移动元素的平均个数为（_____）。
答案：n/2 (n-1)/2
（3）表长为 0 的线性表称为（_____）。
答案：空表
（4）动态内存管理是操作系统的基本功能之一，其作用是响应用户程序对内存的（_____）和（_____）请求。
答案：申请 释放
（5）顺序表多采用（_____）实现的，是一种随机存取结构，对表中任意结点存取操作的时间复杂度为（_____）。而查找链表中的结节，需要从头指针起顺着链扫描才能得到，平均时间复杂度为（_____）。因此，若线性表的操作主要是进行查找，很少进行插入或删除操作时，采用（_____）表比较合适。
答案：数组 0(1) 0(n)顺序
（6）在链表某个位置上进行插入和删除操作，只需要修改（_____）即可，而无须移动大量元素，操作的时间复杂度为（_____）。而在顺序表中进行插入和删除操作，往往要移动大量元素，平均移动元素的数目为（_____），平均时间复杂度为（_____）。因此，若对线性表进行频繁的插入和删除操作时，采用（_____）表相对合适。若插入和删除主要发生在表头和表尾，则采用（_____）表更为合适。
答案：指针 0(1) 表长的一半 0(n) 链 循环链
（7）静态链表一般采用（_____）存储的链表结构。
答案：数组
（8）对于 32 位计算机环境，若单链表中的数据类型为整性，则其存储密度为（_____），
而若为双链表，则存储密度为（_____）。若采用顺序表存储数据，则其存储密度为（_____）。
答案：50% 33% 1
（9）向量是最常用的容器，STL 中向量使用（_____）实现，因此向量具有（_____）表的所有特点，可以快速随机存取任意元素。
答案：数组 顺序
（10）操作系统在运行之初，有一块很大的连续内存块供用户程序申请，通常称之为内存池或（_____）。
答案：堆
（11）循环链表与单链表的区别仅仅在于其尾结点的链域值不是（_____），而是一个指向（_____）的指针。
答案：NULL( 或空指针) 头结点
2. 选择题
8
（1）线性表的顺序存储结构是一种（ ）的存储结构，线性表的链式存储结构是一种（ ）的存储结构。

A. 随机存取 索引存取 B. 顺序存取 随机存取
C. 随机存取 顺序存取 D. 索引存取 散列存取
（2）在双向链表 p 所指结点之前插入 s 所指结点的操作是（ ）。
A. p->left=s; s->right=p; p->left->right =s; s->left=p->left;
B. p->right=s; p->right->left=s; s->left=p; s->right=p->right;
C. s->right=p; s->left=p->left; p->left=s; p->left->right =s;
D. s->right=p; s->left=p->left; p->left->right =s; p->left=s;
（3）若链表是利用 C++ 指针来存储结点的地址，结点空间的分配和收回都是由操作符 new 和 delete动态执行的，则称该链表为（ ）链表
A. 单向 B. 双向 C. 静态 D. 动态
（4）将线性表存储到计算机中可以采用多种不同的方法，按顺序存储方法存储的线性表称为
（ ），按链式存储方法存储的线性表称为（ ）。
A. 数组单链表 B. 顺序表链表
C. 向量静态链表 D. 静态链表动态链表
（5）（ ）是 STL 中线性表的链式存储形式，STL 标准库中一般采用（ ）
实现。
A. vector数组 B. list单链表
C. list双向循环链表 D. vector单链表
（6）顺序表的类型定义可经编译转换为机器级。假定每个结点变量占用 k(k>=1)个字节，b 是顺序表的第一个存储结点的第一个单元的内存地址，那么，第 i 个结点 ai 的存储地址为（ ）。
A. b+ki B. b+k(i-1) C. b+k(i+1) D. b-1+ki
（7）在循环链表中，若不使用头指针而改设为尾指针（rear），则头结点和尾结点的存储位置分别是（ ）。
A. rear和 rear->next->next B. rear->next和rear
C. rear->next->next和 rear D. rear和 rear->next
（8）有时为了叙述方便，可以对一些概念进行简称，以下说法错误的是（ ）。
A. 将“指针型变量”简称为“指针”
B. 将“头指针变量”简称为“头指针”
C. 将“修改某指针型变量的值” 简称为“修改某指针”
D. 将“p 中指针所指结点” 简称为“P 值”
（9）以下说法错误的是（ ）。
A. 对循环链表来说，从表中任一结点出发都能通过向前或向后操作而扫描整个循环链表。
B. 对单链表来说，只有从头结点开始才能扫描表中全部结点。
C. 双链表的特点是找结点的前趋和后继都很容易。
D. 对双链表来说，结点*P 的存储位置既存放在其前趋结点的后继指针域中，也存放在它的后继结点的前趋指针域中。
（10）以下说法正确的是（ ）。
A. 顺序存储方式的优点是存储密度大、且插入、删除运算效率高。
B. 链表的每个结点中都恰好包含一个指针。
C. 线性表的顺序存储结构优于链式存储结构。
D. 顺序存储结构属于静态结构，链式结构属于动态结构。

（11）单链表中，增加头结点的目的是为了（ ）

A. 使单链表至少有一个结点 B. 标示表结点中首结点的位置

C. 方便运算的实现 D. 说明单链表是线性表的链式存储实现

3. 程序选择题

（1）已知 L 指向带表头结点的非空单链表的头结点，且 P 结点既不是首结点，也不是尾结

点，试从下列提供的答案中选择合适的语句序列：

a.删除 P 结点的直接后继结点的语句序列是（_____）

b.删除 P 结点的直接前驱结点的语句序列是（_____）

c.删除 P 结点的语句序列是（_____）

d.删除首结点的语句序列是（_____）

e.删除尾结点的语句序列是（_____）

(1) delete Q; (8) P->next = P->next->next

(2) Q = P (9) P = P->next

(3) L = L->next (10) while(P->next != Q) P = P->next;

(4) P = L (11) while(P != NULL) P = P->next;

(5) Q = P->next (12) while(Q->next != NULL) { P = Q ; Q = Q->next;}

(6) P->next = P (13) while(P->next->next != NULL) P = P->next;

(7) P = P->next->next (14) while(P->next->next != Q) P = P->next;

答案: a. 5 8 1

b. 2 4 14 5 8 1

c. 2 4 10 8 1

d. 4 5 8 1

e. 4 13 5 8 1

（2）已知 p 结点是某双向链表的中间结点，试从下面语句((1)~(18)中选择合适的语句序列，

完成 a～e 要求的操作。

a. 在 p 结点后插入 s 结点的语句序列是_____。

b. 在 p 结点前插入 s 结点的语句序列是_____。

c. 删除 p 结点的直接后继结点的语句序列是_____。

d. 删除 p 结点的直接前驱结点的语句序列是_____。

e. 删除 p 结点的语句序列是_____。

10

(1) p->next = p->next->next; (10) p->prior->next = p;

(2) p->prior = p->prior->prior; (11) p->next->prior = p;

(3) p->next = s; (12) p->next->prior = s;

(4) p->prior = s; (13) p->prior->next = s;

(5) s->next = p; (14)p->next->prior = p->prior;

(6) s->prior = p; (15) q=p->next;

(7) s->next = p->next; (16) q=p->prior;

(8) s->prior = p->prior; (17) delete p;

(9) p->prior->next = p->next; (18) delete q;

答案:

a. 7 6 12 3

b. 5 8 13 4

c. 15 1 11 18

d. 16 2 10 18

e. 14 9 17

4. 分析以下各程序段的执行结果。

（1）程序段一

```
vector <int> ivec(2,100);
ivec.push_back (3);
ivec.insert (ivec.begin (),10);
vector <int>::iterator it = ivec.begin();
ivec.erase (++it);
ivec.pop_back();
ivec.insert(ivec.end(),20);
for ( it = ivec.begin (); it != ivec.end(); it++ ) cout << *it <<" ";
cout << endl;
```

答案:

10 100 20

分析:开始时容器中有 100 100, 然后为 100 100 3 10 100 100 ,3 10 100 3 10 100, 10 100 20

（2）程序段二

```
int a[]={1,2,3,4,5};
vector <int> ivec(a, a+5);
cout << "1.size: " << ivec.size() << endl;
ivec.resize(100);
cout << "2.size: " << ivec.size() << endl;
for (int j=0; j<95; j++) ivec.insert(ivec.end(),j);
cout << "3.size: " << ivec.size() << endl;
ivec.resize(100);
ivec.reserve (20);
cout << "4.size: " << ivec.size() << endl;
```

答案：1.size:5

2.size:100

11

3.size:195

4.size:100

（3）程序段三

```
int a[]={1,2,3,4,5};
list <int> ilist(3,2);
ilist.assign(a,a+5);
ilist.pop_back (); //1 2 3 4
ilist.insert (ilist.end(),7); //1 2 3 4 7
ilist.pop_front (); //2 3 4 7
ilist.front ()=20; //20 3 4 7
ilist.sort(); //3 4 7 20
```

```
for ( list<int>::iterator it = ilist.begin (); it != ilist.end(); it++ )
cout << *it <<" ";
cout << endl;
```

答案：3 4 7 20

5. 算法设计

（1）分别编程实现顺序表类和链表类，并设计完整的测试程序对每个接口进行测试。

（2）设 A= （a1,a2,a3,....)和B=(b1,b2,..,bm)是两个线性表（假定所含数据元素均为整数)。

若 n=m 且 ai=bi(i=1,.. 则称 A=B;若 ai=bi(i=1,..且 aj+1<bj+1（j<n<=m ），则称 A<B;在其

他情况下均称 A>B 。试编写一个比较 A 和 B 的算法，当 A<B,A=B 或 A>B 是分别输出-1,0

或者 1。

（3）假设有两个按数据元素值递增有序排列的线性表 A 和 B，均以单链表作存储结构。编写算法将 A 表和 B 表归并成一个按元素值递减有序（即非递增有序，允许值相同）排列的线性表 C，并要求利用原表(即 A 表和 B 表的)结点空间存放表 C。

（4）试分别以顺序表和单链表作为存储结构，各写一个实现线性表的就地（即使用尽可能少的附加空间）逆置的算法，在原表的存储空间内将线性表（a1，a2,，„„，　an-1，an)逆置

为（an，an-1，„„，　a2，a1)。

（5）假设在长度大于 1 的循环链表中，既无头结点也无头指针。s 为指向链表中某个结点的指针，试编写算法删除结点*s 的前趋结点。

（6）已知一单链表中的数据元素含有三类字符(即:字母字符、数字字符和其它字符)。试编写算法，构造三个循环链表，使每个循环链表中只含同一类的字符，且利用原表中的结点空间作为这三个表的结点空间(头结点可另辟空间)。

（7）Josephus 环问题：任给正整数 n、k，按下述方法

可得排列 1，2，„„，　n 的一个置换：将数字 1，2，„„，

n 环形排列（如图 2-36 所示)，按顺时针方向从 1 开始

计数，计满 k 时输出该位置上的数字（并从环中删去

该数字)，然后从下一个数字开始继续计数，直到环中

所有数字均被输出为止。例如，n=10，k=3 时，输出

的置换是 3，6，9，2，7，1，8，5，10，4。试编写

一算法，对输入的任意正整数 n、k，输出相应的置换

数字序列。

1

2

3

n

n-1

n-2

图 2-36 Josephus 环



习题 3

1. 填空题

（1）栈的进出原则是（_____），队列的进出原则是（_____）。
答案：后进先出（LIFO） 先进先出（FIFO）
（2）设 32 位计算机系统中，空栈 S 存储 int型数据，栈顶指针为 1024H 。经过操作序列 push(1)，push(2)，pop，push(5)，push(7)，pop，push(6)之后，栈顶元素为（_____），栈底元素为（_____），栈的高度为（_____），输出序列是（_____），栈顶指针为（_____）H 。
答案：6 1 3 2,7 1030
（3）两栈共享存储空间，其数组大小为 100，数组下标从 0 开始。top1 和 top2 分别为栈 1
和栈 2 的栈顶元素下标，则栈 1 为空的条件为（_____），栈 2 为空的条件为（_____），栈 1 或栈 2 满的条件为（_____）。
答案：top1==-1 top2==100 top1+1==top2
（4）一个队列的入队顺序是 1234，则队列的输出顺序是（_____）。
答案：1234
（5）设循环队列数组大小为 100，队头指针为 front，队尾指针为 rear，约定 front指向队头元素的前一个位置，该位置永远不存放数据。则入队操作时，修改 rear=(_____)，出队操作修改 front=(_____)，队空的判别条件为（_____），队满的判别条件为(_____)。若 front=20，rear=60，则队列长度为(_____)，若 front=60,rear=20，则队列长度为（_____）。
答案：(rear+1)%100 (front+1)%100 rear==front (rear+1)%100=front 40 60
（6）朴素模式匹配算法中，每个串的起始下标均为 1，变量 i=100，j=10，分别表示主串和模式串当前比较的字符元素下标，若本次比较两字符不同，则 i 回溯为（_____），j 回溯为（_____）。
答案：92 1
（7）用循环链表表示的队列长度为 n，若只设头指针，则出队和入队的时间复杂度分别为（_____）和（_____）。
答案：O(1) O(n)
2. 选择题
（1）将一个递归算法改为对应的非递归算法时，通常需要使用（ ）。
A. 数组 B. 栈 C. 队列 D. 二叉树
（2）四个元素 1、2、3、4 依次进栈，出栈次序不可能出现（ ）情况。
A. 1 2 3 4 B. 4 1 3 2 C. 1 4 3 2 D. 4 3 2 1
（3）设循环队列中数组的下标范围是 1~n，其头尾指针分别为 f 和 r，则其元素个数为（ ）。
A. r-f B. r-f+1 C. (r-f) mod n +1 D. (r-f+n) mod n
（4）10 行 100 列的二维数组 A 按行优先存储，其元素分别为 A[1][1] ~ A[10][100]，每个元
素占 4 字节，已知 Loc(A[6][7])=10000H，则 Loc(A[4][19])=（ ）。
13
A. FC11H B. 9248H C. 2F00H D. FD10H
（5）设有两个串 s1 和 s2，求 s2 在 s1 中首次出现的位置的运算称为（ ）。
A. 连接 B. 模式匹配 C. 求子串 D. 求串长
（6）为了解决计算机主机和键盘输入之间速度不匹配问题，通常设置一个键盘缓冲区，该缓冲区应该是一个（ ）结构。

A. 栈 B. 队列 C. 数组 D. 线性表

（7）STL 中的双端队列为（ ）。

A. 顺序容器 B. 容器适配器 C. 迭代器适配器 D. 泛函适配器

（8）STL 中的（ ）允许用户为队列中的元素设置优先级。

A. 队列适配器 B. 双端队列 C. 优先级队列适配器 D. 栈适配器

（9）string类型不支持以（ ）的方式操作容器，因此不能使用 front back和 pop_back 操作。

A. 线性表 B.队列 C. 栈 D. 串

3. 算法设计

（1）设计一个算法判断算数表达式的圆括号是否正确配对。

（2）假定用带头结点的循环链表表示队列，并且只设置一个指针指向队尾元素，试设计该队列类，完成相应的入队、出队、置空队、求队长等操作接口。

（3）设计算法把一个十进制数转换为任意指定进制数。

（4）设有一个背包可以放入的物品重量为 S，现有 n 件物品，重量分别为 w1，w2，„„，wn 。问能否从这 n 件物品中选择若干件放入此背包，使得放入的重量之和正好为 S。如果存

在一种符合上述要求的选择，则称此背包问题有解，否则此问题无解，试用递归和非递归两种方法设计解决此背包问题的算法。

背包问题分析：

背包问题是一个经典的 NP 问题，它既简单形象容易理解，又在某种程度上能够揭示动态规划的本质，故不少教材都把它作为动态规划部分的第一道例题。本题目是最简单的 01 背包问题，除此之外，还有许多由此衍生出来的很多复杂的背包问题。

本题中，最容易想到的就是假定背包中已放入了部分物品，现将第 i 件物品试着放入背包中，如果可以放进去，背包的重量在原来的基础上增加了 wi；如果不可以放进去，说明加入后太重了，换下一件物品。如果所有的剩余物品都不能放入，说明以前放入的物品不合适，拿出上一次放入的物品，继续试剩余的物品。

递归解法：

设背包函数为 knapsack(int s, int n)，参数 int s为剩余重量，int n为剩余物品数，返回值表示背包分配是否成功。

（1） 如果 s==0，表示分配成功，返回 1；

（2） 如果 s<0 或者 n<0，表示太重，或者物品分配完毕，返回 0；

（3） 执行 knapsack( s- wi, n-1)测试当前这件物品放入是否成功。

（3.1） 如果成功，说明当前这件物品放入刚好最终分配成功。

（4） 返回 knapsack( s , n-1)说明当前物品不合适，减小剩余物品数，继续测试。

测试代码：

```
/*简单的背包问题递归解*/
#include"stdio.h"
#define N 6 /*物品数量*/
14
#define S 8 /*背包大小*/
int W[N+1]={0,1,2,3,4,5,6}; /*数据，各物品重量，W[0]不使用*/
/*
背包函数
knapsack()
```

```
参数
int s剩余重量
int n剩余物品数
返回
int背包分配是否成功
*/
int knapsack(int s,int n)
{
if(s == 0分配结束，成功*/
return 1;
if(s < 0 || (s > 0 && n没有可用空间，或者物品分配完毕*/
return 0;
if( knapsack(s - W[n] , n 递归*/{/*
printf("%-4d",W[n])输出*/
return 1;
}
return knapsack(s , n - 1);
}
int main()
{
if(knapsack(S , N递归调用*/
("\nOK!\n");
else
("Failed!");
return 1;
}/*main*/
```

非递归解法：

一件一件的物品往包（即栈）里放，发现有问题，拿出来，放其他的物品。

（1）i=1

（2）从第 i 件到第 n 件测试每件物品，对于第 j 次循环，测试第 j 件物品

（2.1）如果该物品可以放，入栈

（2.2）若栈的容量刚好达到要求，成功，打印栈元素。

（2.3）继续测试 j+1 件物品

（3）若没有成功

（3.1）若栈空，返回失败

15

（3.2）将栈顶物品（设第 k 个）出栈

（3.3）令 i=k+1，返回（2）

代码：

```
#include"stdio.h"
#define N 6物品数量*/
#define S 8背包大小*/
int W[N+1]={0,1,2,3,4,5,6数据， 各物品重量，W[0]不使用*/
int stack[1000]={0};
```

```
int value=0;
int size=0;
knapsackstack()
{
int i=1;
while (1)
{
for (int j=i;j<=N;j++)
{
if (value+W[j]<=S) {stack[size++]=j; value+=W[j];}
if (value==S){
printf得到一组解：");
for (int p=0;p<size;p++) printf("%d ",W[stack[p]]);
printf("OK!\n");
break;
}
else if (value>S) break;
}
if (size==0) {printf("Finished!");exit(0);}
i=stack[--size]+1;
value-=W[i-1];
}
}
void main()
{
knapsackstack();
}
```



习题 4

1. 填空题

（1）一般来说，数组不执行（_____）和（_____）操作，所以通常采用（_____）方法来存储数组。通常有两种存储方式：（_____）和（_____）。

答案：删除 插入 顺序存储行优先存储列优先存储

（2）设 8 行 8 列的二维数组起始元素为 A[0][0]，按行优先存储到起始元素下标为 0 的一维
数组 B 中，则元素 B[23]在原二维数组中为（_____）。若该二维数组为上三角矩阵，按行优先压缩存储上三角元素到起始元素下标为 0 的一维数组 C 中，则元素 C[23]即为原矩
阵中的（_____）元素。

答案：A[2][7] A[3][5]

（3）设二维数组 A 为 6 行 8 列，按行优先存储，每个元素占 6 字节，存储器按字节编址。已知 A 的起始存储地址为 1000H，数组 A 占用的存储空间大小为（_____）字节，数
组 A 的最后一个元素的下标为（_____），该元素的第一个字节地址为（_____）