

全国计算机技术与软件专业技术资格（水平）考试

2008 年下半年 程序员 下午试卷（B）

（考试时间 14:00~16:30 共 150 分钟）

请按下述要求正确填写答题纸

1. 在答题纸的指定位置填写你所在的省、自治区、直辖市、计划单列市的名称。
2. 在答题纸的指定位置填写准考证号、出生年月日和姓名。
3. 答题纸上除填写上述内容外只能写解答。
4. 本试卷共 7 道题，试题一至试题四是必答题，试题五至试题七选答 1 道。
每题 15 分，满分 75 分。

试题号	一~四	五~七
选择方法	必答题	选答 1 题

5. 解答时字迹务必清楚，字迹不清时，将不评分。
6. 仿照下面例题，将解答写在答题纸的对应栏内。

例题

2008 年下半年全国计算机技术与软件专业技术资格（水平）考试日期是 (1) 月 (2) 日。

因为正确的解答是“12 月 21 日”，故在答题纸的对应栏内写上“12”和“21”（参看下表）。

例题	解答栏
(1)	12
(2)	21

试题一（共 15 分）

阅读以下说明和流程图，填补流程图中的空缺（1）～（5），将解答填入答题纸的对应栏内。

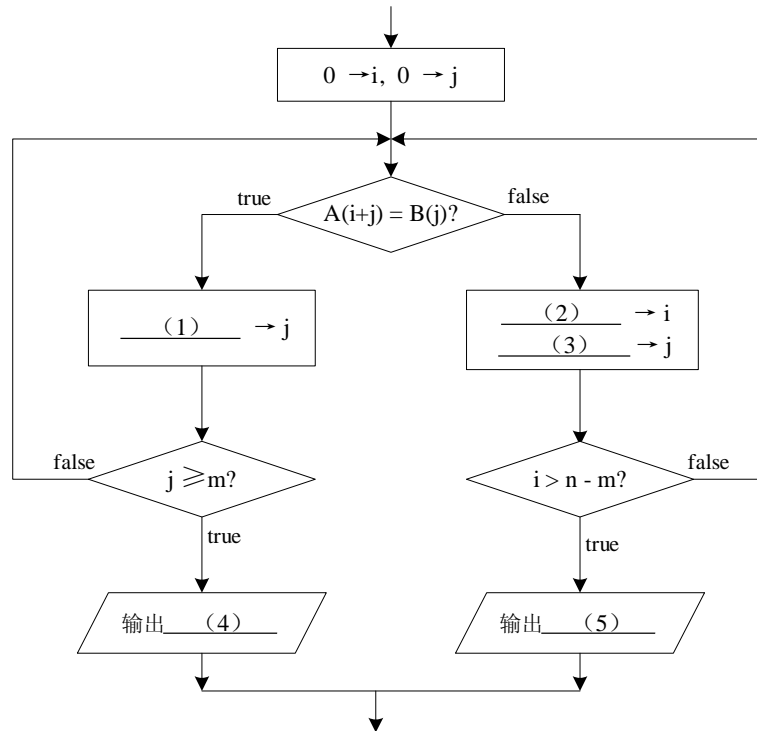
[说明]

下面流程图的功能是：在已知字符串 A 中查找特定字符串 B，如果存在，则输出 B 串首字符在 A 串中的位置，否则输出-1。设串 A 由 n 个字符 A(0)、A(1)、...、A(n-1) 组成，串 B 由 m 个字符 B(0)、B(1)、...、B(m-1) 组成，其中 $n \geq m > 0$ 。在串 A 中查找串 B 的基本算法如下：从串 A 的首字符 A(0) 开始，取子串 A(0)A(1)...A(m-1) 与串 B 比较；若不同，则再取子串 A(1)A(2)...A(m) 与串 B 比较，依次类推。

例如，字符串“CABBRFFD”中存在字符子串“BRF”（输出 3），不存在字符子串“RFD”（输出-1）。

在流程图中，i 用于访问串 A 中的字符 ($i=0, 1, \dots, n-1$)，j 用于访问串 B 中的字符 ($j=0, 1, \dots, m-1$)。在比较 A(i)A(i+1)...A(i+m-1) 与 B(0)B(1)...B(m-1) 时，需要对 A(i) 与 B(0)、A(i+1) 与 B(1)、...、A(i+j) 与 B(j)、... 逐对字符进行比较。若发现不同，则需要取下一个子串进行比较，依此类推。

[流程图]



试题二（共 15 分）

阅读以下说明和 C 程序代码，将应填入 (n) 处的字句写在答题纸的对应栏内。

[说明]

下面 C 程序代码的功能是：对于输入的一个正整数 n ($100 \leq n < 1000$)，先判断其是否是回文数（正读反读都一样的数）。若不是，则将 n 与其反序数相加，再判断得到的和数是否为回文数，若还不是，再将该和数与其反序数相加并进行判断，依此类推，直到得到一个回文数为止。例如，278 不是回文数，其反序数为 872，相加后得到的 1150 还不是回文数，再将 1150 与其反序数 511 相加，得到的 1661 是回文数。

函数 `int isPalm(long m)` 的功能是：将正整数 m 的各位数字取出存入数组中，然后判断其是否为回文数。若 m 是回文数则返回 1，否则返回 0。

[C 程序代码]

```
#include <stdio.h>
#include <stdlib.h>
int isPalm(long m)
{ /*判断 m 是否为回文数*/
    int i = 0, k = 0;
    char str[32];
    while (m > 0) { /*从个位数开始逐个取出 m 的各位数字并存入字符数组 str*/
        str[k++] = (1) + '0';
        m = m / 10;
    }
    for(i = 0; i < k/2; i++) /*判断 str 中的 k 个数字字符序列是否是回文*/
        if (str[i] != str[(2)]) return 0;
    return 1;
}
int main()
{
    long n, a, t;
    printf("input a positive integer:"); scanf("%ld",&n);
    if (n < 100 || n >= 1000) return -1 ;
    while((3)) { /*n 不是回文数时执行循环*/
        printf("%ld -> ", n);
        for(a = 0, t = n; t > 0; ) { /*计算 n 的反序数并存入 a*/
            a = (4) *10 + t % 10; t = t / 10;
        } /*end of for*/
        n = (5); /*与反序数求和*/
    } /*end of while*/
    printf("%ld\n",n);
    system("pause"); return 0;
}
```

试题三（共 15 分）

阅读以下说明和 C 函数，将应填入 (n) 处的字句写在答题纸的对应栏内。

[说明]

已知某二叉树的非叶子结点都有两个孩子结点，现将该二叉树存储在结构数组 **Ht** 中。结点结构及数组 **Ht** 的定义如下：

```
#define MAXLEAFNUM 30
struct node{
    char ch;          /*当前结点表示的字符，对于非叶子结点，此域不用*/
    char *pstr;      /*当前结点的编码指针，非叶子结点不用*/
    int parent;      /*当前结点的父结点，为 0 时表示无父结点*/
    int lchild,rchild;
    /*当前结点的左、右孩子结点，为 0 时表示无对应的孩子结点*/
};
struct node Ht[2*MAXLEAFNUM]; /*数组元素 Ht[0]不用*/
```

该二叉树的 *n* 个叶子结点存储在下标为 1~*n* 的 **Ht** 数组元素中。例如，某二叉树如图 3-1 所示，其存储结构如图 3-2 所示，其中，与叶子结点 **a** 对应的数组元素下标为 1，**a** 的父结点存储在 **Ht[5]**，表示为 **Ht[1].parent=5**。**Ht[7].parent=0** 表示 7 号结点是树根，**Ht[7].lchild=3**、**Ht[7].rchild=6** 分别表示 7 号结点的左孩子是 3 号结点、右孩子是 6 号结点。

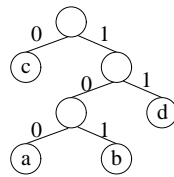


图 3-1 二叉树示意图

下标	ch	parent	lchild	rchild
1	a	5	0	0
2	b	5	0	0
3	c	7	0	0
4	d	6	0	0
5		6	1	2
6		7	5	4
7		0	3	6

图 3-2 结构数组 **Ht** 内容示意图

如果用“0”或“1”分别标识二叉树的左分支和右分支（如图 3-1 所示），从根结点开始到叶子结点为止，按所经过分支的次序将相应标识依次排列，可得到一个 0、1

序列，称之为对应叶子结点的编码。例如，图 3-1 中 a、b、c、d 的编码分别是 100、101、0、11。

函数LeafCode(Ht[],n)的功能是：求解存储在Ht中的二叉树中所有叶子结点(n个)的编码，叶子结点存储在Ht[1]~Ht[n]中，求出的编码存储区由对应的数组元素pstr域指示。

函数LeafCode从叶子到根逆向求叶子结点的编码。例如，对图 3-1 中叶子结点a求编码的过程如图 3-3 所示。

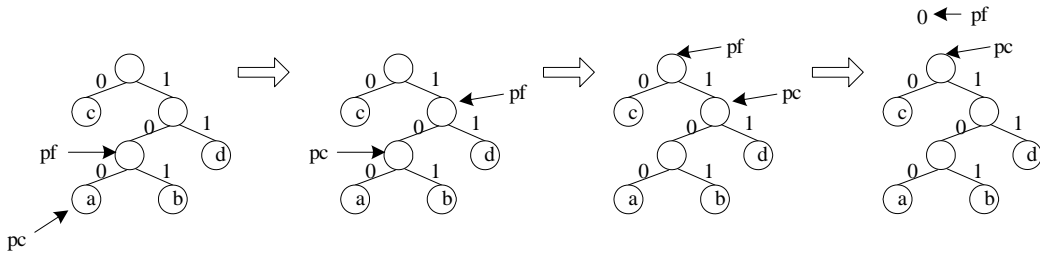


图 3-3 从叶子到根求结点编码示意图

```
typedef enum Status {ERROR, OK} Status;
```

[C函数]

```
Status LeafCode(struct node Ht[], int n)
```

```
{
    int pc, pf; /*pc 用于指出树中的结点， pf 则指出 pc 所对应结点的父结点*/
    int i, start;
    char tstr[31] = {"\0"}; /*临时存储给定叶子结点的编码，从高下标开始存入*/
    for(i = 1; __(1)__; i++) { /*对所有叶子结点求编码， i 表示叶结点在 HT 数组中的下标*/
        start = 29;
        pc = i;    pf = Ht[i].parent;
        while (pf != __(2)__) { /*没有到达树根时，继续求编码*/
            if (__(3)__.lchild == pc) /*pc 所表示的结点是其父结点的左孩子*/
                tstr[--start] = '0';
            else
                tstr[--start] = '1';
            pc = __(4)__;    pf = Ht[pf].parent; /*pc 和 pf 分别向根方向回退一层*/
        } /* end of while */
        Ht[i].pstr = (char *) malloc(31-start);
        if (!Ht[i].pstr) return ERROR;
        strcpy(Ht[i].pstr, __(5));
    } /* end of for */
    return OK;
} /* end of LeafCode */
```

试题四（共 15 分）

阅读以下说明和 C 函数代码，回答问题并将解答写在答题纸的对应栏内。

[说明]

著名的菲波那契数列定义式为

$$f_1 = 1 \quad f_2 = 1 \quad f_n = f_{n-1} + f_{n-2} \quad (n = 3, 4, \dots)$$

因此，从第 1 项开始的该数列为 1,1,2,3,5,8,13,21,...。函数 fib1 和 fib2 分别用递归方式和迭代方式求解菲波那契数列的第 n 项（调用 fib1、fib2 时可确保参数 n 获得一个正整数）。

[C 函数代码]

```
long fib1(int n)
{
    if (n <= 2)
        return 1;
    else
        fib1(n) = fib1(n-1) + fib1(n-2);
}
```

```
long fib2(int n)
{
    long f1=1,f2=1; int i;
    long f;
    for(i=3; i<=n; i++) {
        f = f1 + f2;
        f1 = f2; f2 = f;
    }
    return f;
}
```

[问题 1]（6 分）

函数 fib1 和 fib2 存在错误，只需分别修改其中的一行代码即可改正错误。

(1) 函数 fib1 不能通过编译，请写出 fib1 中错误所在行修改正确后的完整代码；

(2) 函数 fib2 在 $n \leq 2$ 时不能获得正确结果，请写出 fib2 中错误所在行修改正确后的完整代码。

[问题 2]（3 分）

将函数 fib1 和 fib2 改正后进行测试，发现前 46 项都正确，而第 47 项的值是一个负数，请说明原因。

[问题 3]（6 分）

函数 fib1、fib2 求得菲波那契数列第 n 项 ($n > 40$) 的速度并不相同，请指出速度慢的函数名，并简要说明原因。

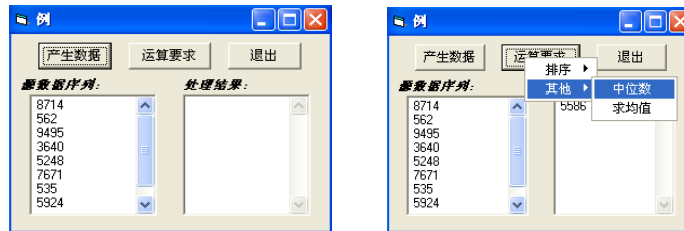
从下列 3 道试题（试题五至试题七）中任选 1 道解答。如
果解答的试题数超过 1 道，则题号小的 1 道解答有效。

试题五（共 15 分）

阅读以下应用说明、属性设置以及 Visual Basic 程序代码，将解答写在答题纸的对应栏内。

[应用说明]

本应用运行时，由用户输入一个正整数 n 后自动产生 n 个正整数，然后按照用户的指定要求对该组数进行处理。该应用的运行界面如下图所示：



- 窗体中有两个文本框（txtSrc，txtObj）、两个标签（lblSrc，lblObj）、三个命令按钮（cmdGendat，cmdProc，cmdQuit）和一个弹出式菜单（procMenu，初始时不可见）。
- 文本框 txtSrc（由标签 lblSrc 提示）用于显示产生的数据，文本框 txtObj（由标签 lblObj 提示）用于显示处理结果，要求每行显示一个整数。
- 程序启动时，命令按钮 cmdProc（运算要求）不可用。点击命令按钮 cmdGendat（产生数据）后，提示用户输入一个 n 的值并生成 n 个正整数存入数组元素 a(1)~a(n)，然后将数据逐行显示在 txtSrc 中，并设置命令按钮 cmdProc 可用。
- 点击命令按钮 cmdProc（运算要求）后弹出菜单。选择菜单项并单击后，进行相应处理并将结果显示在 txtObj 中，同时将 lblObj 的标题改为该菜单项表示的处理命令。
弹出式菜单“运算要求”的结构如下表所示：

标题	名称	层次
运算要求	procMenu	1
排序	Sorting	2
递增排列	Ascend	3
递减排列	Descend	3
找特殊数	SpecNum	2
中位数	MidNum	3
求均数	AvgNum	3

一个整数序列的中位数指对该序列进行非递减（增）排列后最中间位置上的元素。若序列长度为偶数，则取中间两个元素的平均值为其中位数。

[属性设置]

为实现单击命令按钮 cmdProc 后弹出“运算要求”菜单（procMenu），设计时需将 procMenu 的 (1) 属性设置成 false。

供 (1) 选择的属性： Default Enabled ScaleMode Style Visible

[Visual Basic 程序代码]

```
Dim a() As Integer, n As Integer
Private Sub Form_Load()
    txtSrc.Text = "": txtObj.Text = "": (2) = False
End Sub
Private Sub cmdGendat_Click() '生成正整数序列并存入数组a
    On Error GoTo Error_handler
    n = InputBox$("请输入数组元素个数: ", "输入序列长度")
    If (n < 1) Then
        MsgBox "输入数据错误!", vbOKOnly, "提示: "
        GoTo Error_handler:
    End If
    ReDim a(n) As Integer
    s = ""
    For i = 1 To n '将生成的正整数存入a(1)~a(n)中
        a(i) = Int(Rnd * 10000): s = s & Str$(a(i)) & vbCrLf
    Next
    txtSrc.Text = s
    (3) '设置运算要求命令按钮可用
Error_handler:
End Sub

Private Sub cmdProc_Click()
    PopupMenu procMenu
End Sub

Private Sub MidNum_Click() '求中位数
    lblObj.Caption = MidNum.Caption & ":"
    For i = 1 To round((n + 1)/2) '用选择排序法对数组a进行部分排序
        a(0) = a(i):k = i 'a(0)用作临时变量, 暂存第i次选出的最小元素
        For j = i + 1 To n
            If a(j) < a(0) Then
                a(0) = a(j): k = (4)
            End If
        Next
        If k <> i Then
            a(k) = a(i): a(i) = a(0)
        End If
    Next
    If n / 2 - n \ 2 > 0 Then 'n为奇数时, 取中间一个数
        txtObj.Text = Str$(a((5)))
    Else 'n为偶数时, 取中间两个数的平均值
        txtObj.Text = Str$(Int((a(n \ 2) + a(n \ 2 + 1)) / 2))
    End If
End Sub
'其他代码略
```

试题六（共 15 分）

阅读以下说明和 C++ 代码，将应填入 (n) 处的字句写在答题纸的对应栏内。

[说明]

C++ 标准模板库中提供了 vector 模板类，可作为动态数组使用，并可容纳任意数据类型，其所属的命名空间为 std。vector 模板类的部分方法说明如下表所示：

方法	含义
push_back(k)	向 vector 对象的尾部添加一个元素 k
begin()	返回一个迭代器对象，该对象指向 vector 中的第一个元素
end()	返回一个迭代器对象，该对象指向 vector 中的最后一个元素
empty()	测试 vector 对象是否为空
erase(ptr)	删除 vector 中 ptr 指向的元素

[C++ 代码]

```
#include <iostream>
#include <vector>
using namespace (1);
typedef vector<(2)> INTVECTOR;
const int ARRAY_SIZE = 6;
void ShowVector(INTVECTOR &theVector);
int main(){
    INTVECTOR theVector;
    // 初始化 theVector，将 theVector 的元素依次设置为 0 至 5
    for (int cEachItem = 0; cEachItem < ARRAY_SIZE; cEachItem++){
        theVector.push_back((3));
        ShowVector(theVector); // 依次输出 theVector 中的元素
        theVector.erase(theVector.begin() + 3);
        ShowVector(theVector);
    }
    void ShowVector(INTVECTOR &theVector) {
        if (theVector.empty()) {
            cout << "theVector is empty." << endl;    return;
        }
        INTVECTOR::iterator (4);
        for (theIterator = theVector.begin(); theIterator != theVector.end(); theIterator++){
            cout << *theIterator;
            if (theIterator != theVector.end()-1) cout << ", ";
        }
        cout << endl;
    }
}
该程序运行后的输出结果为：
0, 1, 2, 3, 4, 5
(5)
```

试题七（共 15 分）

阅读以下说明和 Java 代码，将应填入 （n） 处的字句写在答题纸的对应栏内。

[说明]

java.util 库中提供了 Vector 模板类，可作为动态数组使用，并可容纳任意数据类型。该类的部分方法说明如下表所示：

方法名	含义
add(k)	向 vector 对象的尾部添加一个元素 k
removeElementAt(i)	删除序号为 i 的元素（vector 元素序号从 0 开始）
isEmpty()	判断 vector 对象是否含有元素
size()	返回 vector 对象中所包含的元素个数

[Java 代码]

```
import （1） ;
public class JavaMain {
    static private final int （2） = 6;
    public static void main(String[] args){
        Vector<Integer> theVector = new Vector<（3）>();
        // 初始化 theVector，将 theVector 的元素设置为 0 至 5
        for (int cEachItem = 0; cEachItem < ARRAY_SIZE; cEachItem++)
            theVector.add(（4）);

        showVector(theVector); // 依次输出 theVector 中的元素
        theVector.removeElementAt(3);
        showVector(theVector);
    }
    public static void showVector(Vector<Integer> theVector){
        if (theVector.isEmpty()) {
            System.out.println("theVector is empty.");
            return;
        }
        for (int loop = 0; loop < theVector.size(); loop++) {
            System.out.print(theVector.get(loop));
            System.out.print(", ");
        }
        System.out.println();
    }
}
```

该程序运行后的输出结果为：

0, 1, 2, 3, 4, 5

（5）