

C++ 编程等级认证四级 2023 年 09 月理论 60 分钟

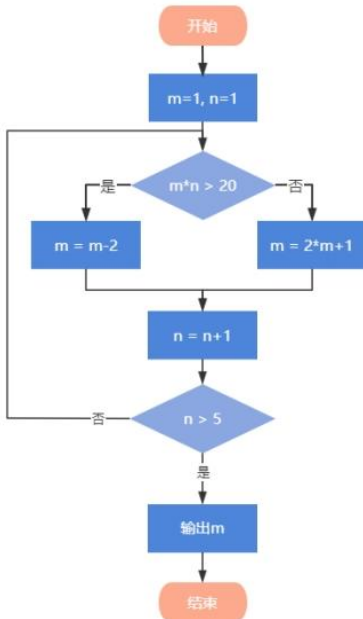
【单选题】(每题 2 分)

1、人们所使用的手机上安装的 App 通常指的是？()

- A、一款操作系统 B、一款应用软件 C、一种通话设备 D、以上都不对

正确答案：B

2、下列流程图的输出结果是？()



- A、9 B、7 C、5 D、11

正确答案：A

3、对包含 n 个元素的数组进行冒泡排序，平均时间复杂度一般为？()

- A、 $O(n)$
B、 $O(n\log n)$
C、 $O(n^2)$
D、以上都不正确

正确答案：C

4、下列关于 C++ 语言中指针的叙述，不正确的是？()

- A、可以定义指向 int 类型的指针。
B、可以定义指向自定义结构体类型的指针。
C、自定义结构体类型可以包含指针类型的元素。
D、不能定义指向 void 类型的指针，那没有意义。

正确答案：D

5、下列关于 C++ 语言中数组的叙述，不正确的是？()

- A、一维数组可以用来表示数列。
B、二维数组可以用来表示矩阵。
C、三维数组可以用来表示空间中物体的形状。
D、世界是三维的，所以定义四维数组没有意义。

正确答案：D

6、下列关于 C++ 语言中函数的叙述，正确的是？()

- A、函数调用前必须定义。 B、函数调用时必须提供足够的实际参数。 C、函数定义前必须声明。 D、函数声明只能写在函数调用前。

正确答案：B

7、下列关于 C++ 语言中函数的叙述，不正确的是？()

A、两个函数的声明可以相同。 B、两个函数的局部变量可以重名。 C、两个函数的参数可以重名。 D、两个函数可以重名。

正确答案：A

8、一个二维数组定义为 `char array[3][10];`，则这个二维数组占用内存的大小为？（ ）

A、10 B、30 C、32 D、48

正确答案：B

9、如果 `n` 为 `int` 类型的变量，一个指针变量定义为 `int *p = &n;`，则下列说法正确的是？（ ）

- A、指针变量 `p` 的值与变量 `n` 是相同的。
- B、指针变量 `p` 的值与变量 `n` 的地址是相同的。
- C、指针变量 `p` 指向的值为 `'n'`。
- D、指针变量 `p` 指向的值与变量 `n` 的地址是相同的。

正确答案：B

10、一个三维数组定义为 `long long array[6][6][6];`，则 `array[1][2][3]` 和 `array[3][2][1]` 在内存中的位置相差多少字节？（ ）

A、70 字节 B、198 字节 C、560 字节 D、无法确定

正确答案：C

11、如果 `a` 为 `int` 类型的变量，且 `a` 的值为 6，则执行 `a = ~a;` 之后，`a` 的值会是？（ ）

A、-6 B、6 C、-7 D、7

正确答案：C

12、一个数组定义为 `int a[5] = {1, 2, 3, 4, 5};`，一个指针定义为 `int *p = &a[2];`，则执行 `*p = a[1];` 后，数组 `a` 中的值会变为？（ ）

A、{1, 2, 2, 4, 5} B、{1, 3, 3, 4, 5} C、{1, 2, 3, 3, 5} D、{1, 2, 4, 4, 5}

正确答案：A

13、下列关于 C++ 语言中异常处理的叙述，正确的是？（ ）

- A、一个 `try` 子句可以有多个 `catch` 子句与之对应。
- B、如果 `try` 子句在执行时发生异常，就一定会进入某一个 `catch` 子句执行。
- C、如果 `try` 子句中没有可能发生异常的语句，会产生编译错误。
- D、`catch` 子句处理异常后，会重新执行与之对应的 `try` 子句。

正确答案：A

14、执行以下 C++ 语言程序后，输出结果是？（ ）

```
#include <iostream>
```

```
using namespace std;
```

```
int main() {
```

```
    int fib[10];
```

```
    fib[0] = 0;
```

```
    fib[1] = 1;
```

```
    for (int i = 2; i < 10; i++)
```

```
        fib[i] = fib[i - 1] + fib[i - 2];
```

```
    cout << fib[10] << endl;
```

```
    return 0;
```

```
}
```

A、0 B、5 C、55 D、无法确定。

正确答案：D

15、在下列代码的横线处填写（ ），完成对有 `n` 个 `int` 类型元素的数组 `array` 由小到大排序。

```
void BubbleSort(int array[], int n) {
```

```
    for (int i = n; i >= 2; i--)
```

```
        for ( _____ ) // 在此处填入代码
```

```
            if (array[j] > array[j + 1]) {
```

```
                int t = array[j];
```

```

        array[j] = array[j + 1];
        array[j + 1] = t;
    }
}

```

- A、int j = 1; j < n; j++
 B、int j = 0; j < n; j++
 C、int j = 0; j < i - 1; j++
 D、int j = 0; j < i; j++

正确答案：C

【判断题】(每题 2 分)

16、在 C++ 语言中，指针变量在逻辑上指向另一个变量在内存中的位置，指针变量本身不占用内存。()

正确答案：错误

17、对 N 个元素的数组执行插入排序算法，通常的时间复杂度是 $O(N^2)$ 。()

正确答案：正确

18、在 C++ 语言中，每个变量都有其作用域。()

正确答案：正确

19、在 C++ 语言中，在函数调用时，通过引用传递的参数不会复制实际参数，因此不会额外占用内存。()

正确答案：错误

20、在 C++ 语言中，可以通过定义结构体，定义一个新的数据类型。()

正确答案：正确

21、在 C++ 语言中，可以定义结构体类型的数组变量，定义结构体时也可以包含数组成员。()

正确答案：正确

22、如果希望记录 10 个最长为 99 字节的字符串，可以将字符串数组定义为 char s[10][100];。()

正确答案：正确

23、一个可能抛出异常的函数，调用它的位置没有在 try 子句中，会引起编译错误。()

正确答案：错误

24、== 和 := 都是 C++ 语言的运算符。()

正确答案：错误

25、通过使用文件重定向操作，可以将程序中输出到 cout 的内容输出到文件中，这是常用的记录程序运行日志的方法之一。()

正确答案：正确

【编程题】

26、(本题 25 分) 进制转换

时间限制：1.0 s

内存限制：128.0 MB

【问题描述】

N 进制数指的是逢 N 进一的计数制。例如，人们日常生活中大多使用十进制计数，而计算机底层则一般使用二进制。

除此之外，八进制和十六进制在一些场合也是常用的计数制（十六进制中，一般使用字母 A 至 F 表示十至十五；本题中，十一进制到十五进制也是类似的）。

在本题中，我们将给出 N 个不同进制的数。你需要分别把它们转换成十进制数。

【提示】

对于任意一个 L 位 K 进制数，假设其最右边的数位为第 0 位，最左边的数位为第 L-1 位，我们只需要将其第 i 位的数码乘以权

值 K^i ，再将每位的结果相加，即可得到原 K 进制数对应的十进制数。下面是两个例子：

(1) 八进制数 1362 对应的十进制数为 $1 \times 8^3 + 3 \times 8^2 + 6 \times 8^1 + 2 \times 8^0 = 754$ ；

(2) 十六进制数 3F0 对应的十进制数为 $3 \times 16^2 + 15 \times 16^1 + 0 \times 16^0 = 1008$ 。

【输入描述】

输入的第一行为一个十进制表示的整数 N 。接下来 N 行，每行一个整数 K ，随后是一个空格，紧接着是一个 K 进制数，表示需要转换的数。保证所有 K 进制数均由数字和大写字母组成，且不以 0 开头。保证 K 进制数合法。

保证 $N \leq 1000$ ；保证 $2 \leq K \leq 16$ ，保证所有 K 进制数的位数不超过 9。

【输出描述】

输出 N 行，每一个十进制数，表示对应 K 进制数的十进制数值。

【特别提醒】

在常规程序中，输入、输出时提供提示是好习惯。但在本场考试中，由于系统限定，请不要在输入、输出中附带任何提示信息。

【样例输入 1】

```
2
8 1362
16 3F0
```

【样例输出 1】

```
754
1008
```

【样例输入 2】

```
2
2 11011
10 123456789
```

【样例输出 2】

```
27
123456789
```

参考程序：

```
#include <iostream>
#include <cstring>
using namespace std;
int trans_digit(int k, char c) {
    if (c <= '9')
        return (c - '0');
    return (c - 'A' + 10);
}
long long trans(int k, char str[]) {
    int l = strlen(str);
    long long res = 0, pw = 1;
    for (int i = l - 1; i >= 0; i--) {
        res += pw * trans_digit(k, str[i]);
        pw *= k;
    }
    return res;
}
int main() {
    int n = 0;
    cin >> n;
    for (int t = 0; t < n; t++) {
        int k = 0;
        char str[10];
        cin >> k >> str;
        cout << trans(k, str) << endl;
    }
}
```

```
return 0;
}
```

27、(本题 25 分) 变长编码

时间限制：1.0 s

内存限制：128.0 MB

【问题描述】

小明刚刚学习了三种整数编码方式：原码、反码、补码，并了解到计算机存储整数通常使用补码。

但他总是觉得，生活中很少用到 $2^{31} - 1$ 这么大的数，生活中常用的 0~100 这种数也同样需要用 4 个字节的补码表示，太浪费了些。

热爱学习的小明通过搜索，发现了一种正整数的变长编码方式。这种编码方式的规则如下：

(1) 对于给定的正整数，首先将其表达为二进制形式。例如， $(0)\{10\} = (0)\{2\}$ ， $(926)\{10\} = (1110011110)\{2\}$ 。

(2) 将二进制数从低位到高位切分成每组 7 bit，不足 7 bit 的在高位用 0 填补。例如， $(0)\{2\}$ 变为 0000000 的一组， $(1110011110)\{2\}$ 变为 0011110 和 0000111 的两组。

(3) 由代表低位的组开始，为其加入最高位。如果这组是最后一组，则在最高位填上 0，否则在最高位填上 1。于是，0 的变长编码为 00000000 一个字节，926 的变长编码为 10011110 和 00000111 两个字节。

这种编码方式可以用更少的字节表达比较小的数，也可以用很多的字节表达非常大的数。例如，

987654321012345678 的二进制为 $(0001101\ 1011010\ 0110110\ 1001011\ 1110100\ 0100110\ 1001000\ 0010110\ 1001110)\{2\}$ ，于是它的变长编码为 (十六进制表示) CE 96 C8 A6 F4 CB B6 DA 0D，共 9 个字节。

你能通过编写程序，找到一个正整数的变长编码吗？

【输入描述】

输入第一行，包含一个正整数 N。约定 $0 \leq N \leq 10^{18}$ 。

【输出描述】

输出一行，输出 N 对应的变长编码的每个字节，每个字节均以 2 位十六进制表示 (其中，A-F 使用大写字母表示)，两个字节间以空格分隔。

【样例输入 1】

0

【样例输出 1】

00

【样例输入 2】

926

【样例输出 2】

9E 07

【样例输入 3】

987654321012345678

【样例输出 3】

CE 96 C8 A6 F4 CB B6 DA 0D

参考程序：

```
#include <iostream>
using namespace std;
void output_digit(int d) {
    if (d >= 10)
        cout << (char)('A' + d - 10);
    else
        cout << (char)('0' + d);
}
void output_code(int s) {
    output_digit(s >> 4);
```

```
        output_digit(s & 0x0f);
    }
int main() {
    long long n = 0;
    cin >> n;
    int split[10];
    int l = 0;
    while (n > 0) {
        split[l] = (int)(n & 0x7f);
        n >>= 7;
        l++;
    }
    for (int i = 0; i < l - 1; i++)
        split[i] |= 0x80;
    output_code(split[0]);
    for (int i = 1; i < l; i++) {
        cout << " ";
        output_code(split[i]);
    }
    cout << endl;
    return 0;
}
```