CCF 编程能力等级认证
Grade Examination of Software Programming

C++ 四级

2026 年 03 月

1 单选题（每题 2 分，共 30 分）

题号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
答案	B	B	B	B	A	B	B	B	B	B	B	C	B	B	A

第 1 题 执行下面程序后，输出为（ ）。

```
int f(int x = 2){
    return x * 3;
}

int main(){
    cout << f() << " " << f(4);
}
```

- A. 2 12
- B. 6 12
- C. 6 4
- D. 12 6

第 2 题 执行下面代码后，输出为（ ）。

```
int main() {
    int a = 5;
    int* p = &a;
    int** q = &p;
    **q += 7;
    cout << a << " " << *p;
}
```

- A. 5 5
- B. 12 12
- C. 12 5
- D. 5 12

第 3 题 已知：

```
int a[3][4] = {
    {1,2,3,4},
    {5,6,7,8},
    {9,10,11,12}
};
int (*p)[4] = a;
```

则表达式 `*(*(p + 2) + 1)` 的值为（ ）。

- [ ] **A.** 6
- [ ] **B.** 10
- [ ] **C.** 9
- [ ] **D.** 11

第 4 题 执行下面程序后，输出为（ ）。

```
void fun(int a, int &b, int *c){
    a += 1;
    b += 2;
    *c += 3;
}

int main(){
    int x = 1, y = 1, z = 1;
    fun(x, y, &z);
    cout << x << " " << y << " " << z;
}
```

- [ ] **A.** `2 3 4`
- [ ] **B.** `1 3 4`
- [ ] **C.** `2 1 4`
- [ ] **D.** `1 1 1`

第 5 题 执行下面程序后输出为（ ）。

```
int x = 3;
void f(int& x){
    x += 2;
}
int main(){
    int x = 10;
    f(x);
    cout << x << " " << ::x;
}
```

- [ ] **A.** `12 3`
- [ ] **B.** `10 5`
- [ ] **C.** `12 5`
- [ ] **D.** `10 3`

第 6 题 下列关于结构体初始化的写法，正确的是（ ）。

- [ ] **A.**

```
struct Point { int x, y; };

Point p = (1,2);
```

- [ ] **B.**

```
struct Point { int x, y; };

Point p = {1,2};
```

C.

```
struct Point { int x, y; };

Point p = new Point(1,2);
```

D.

```
struct Point { int x, y; };

Point p = <1,2>;
```

第 7 题 执行下面代码后输出为（ ）。

```
struct S { int a; int b; };

void g(S s){ s.a += 10; }
void h(S& s){ s.b += 10; }

int main(){
    S s{1,2};
    g(s);
    h(s);
    cout << s.a << " " << s.b;
}
```

A. `11 12`

B. `1 12`

C. `11 2`

D. `1 2`

第 8 题 关于递推算法的描述，正确的是（ ）。

A. 递推表现为函数自己调用自己

B. 递推从已知初值出发，利用递推关系逐步推出后续结果

C. 递推只能用于指数复杂度问题

D. 递推一定需要回溯

第 9 题 执行 `climb(6)` 的返回值为（ ）。

```
int climb(int n){
    if(n <= 2) return n;
    int a = 1, b = 2, c = 0;
    for(int i = 3; i <= n; i++){
        c = a + b;
        a = b;
        b = c;
    }
    return c;
}
```

A. 8

B. 13

C. 5

D. 10

第 10 题 某排序算法对如下数据排序（按 score 升序），则下面关于该排序算法稳定性的描述中，说法正确的是（ ）。

初始：`(90,'A'), (90,'B'), (80,'C'), (90,'D')`

排序后：`(80,'C'), (90,'A'), (90,'B'), (90,'D')`

- [ ] **A.** 不稳定，因为出现了相同分数
- [ ] **B.** 稳定，因为相同 score 的相对顺序保持为 A 在 B 前、B 在 D 前
- [ ] **C.** 不稳定，因为 C 跑到前面了
- [ ] **D.** 无法判断

第 11 题 下面代码试图把数组按升序进行“插入排序”，横线处应填写（ ）。

```
void ins(int a[], int n){
    for(int i = 1; i < n; i++){
        int key = a[i];
        int j = i-1;
        while(j >= 0 && __________){
            a[j+1] = a[j];
            j--;
        }
        a[j+1] = key;
    }
}
```

- [ ] **A.** `a[j] < key`
- [ ] **B.** `a[j] > key`
- [ ] **C.** `a[j+1] > key`
- [ ] **D.** `a[j] == key`

第 12 题 下列代码段的时间复杂度为（ ）。

```
int cnt=0;
for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
        if( (i+j) % 3 == 0) cnt++;
    }
}
```

- [ ] **A.** $O(n)$
- [ ] **B.** $O(n \log n)$
- [ ] **C.** $O(n^2)$
- [ ] **D.** $O(2^n)$

第 13 题 下面哪种方式不能实现将字符串 `Welcome to 2026!` 输出重定向到文件 `log.txt` （ ）。

- [ ] **A.**

```
freopen("log.txt", "w", stdout);
cout << "Welcome to 2026!" << endl;
fclose(stdout);
```

B.

```
std::ofstream outFile("log.txt");
cout << "Welcome to 2026!" << endl;
outFile.close();
```

C.

```
ofstream log_file("log.txt");
streambuf* org_cout = cout.rdbuf();
cout.rdbuf(log_file.rdbuf());
cout << "Welcome to 2026!" << endl;
cout.rdbuf(org_cout);
```

D.

```
std::ofstream outFile("log.txt");
outFile << "Welcome to 2026!" << endl;
outFile.close();
```

第 14 题 执行下面程序，输出结果是（ ）。

```
int divi(int a,int b){
    if(b==0) throw 0;
    return a/b;
}

int main(){
    try{
        cout << divi(10,0);
    }catch(const char* msg){
        cout << "A";
    }catch(int){
        cout << "B";
    }
}
```

A. `A`

B. `B`

C. 程序崩溃

D. 无输出

第 15 题 下列函数实现排行榜中单个元素的位置调整（类似插入排序的相邻搬移）。当某玩家分数增加，需将其向前移动时，`while` 循环的条件应为（ ）。

```
struct Player{ int score; };
void up(Player players[], int n, int idx){
    Player cur = players[idx];
    int i = idx;
    while( ____________________ ){
        players[i] = players[i-1];
        i--;
    }
    players[i] = cur;
}
```

A. `i > 0 && cur.score > players[i-1].score`

B. `i > 0 && cur.score < players[i-1].score`

C. `i < n-1 && cur.score > players[i+1].score`

☐ **D.** `i < n-1 && cur.score < players[i+1].score`

2　判断题（每题 2 分，共 20 分）

题号	1	2	3	4	5	6	7	8	9	10
答案	√	√	×	×	√	×	√	×	×	×

第 1 题 下面代码执行结束时，变量 `a` 的值变成 15。

```
void add10(int &x) { x += 10; }

int main() {
    int a = 5;
    add10(a);
}
```

第 2 题 引用一旦绑定某个变量，就不能再绑定其他变量。（）

第 3 题 执行下面代码，输出结果为 `5` 。

```
int main() {
    int a[2][3];
    cout << &a[1][2] - &a[0][1] << endl;
    return 0;
}
```

第 4 题 下面程序可以正常编译并输出 `10` 。

```
int calc(int x, int y = 10);
int calc(int x) { return x * 2; }
int calc(int x, int y) { return x * y; }

int main() {
    cout << calc(5);
}
```

第 5 题 下面程序执行后输出 `2010` 。

```
int x = 10;
void f() { int x = 20; cout << x; }

int main() {
    f();
    cout << x;
}
```

第 6 题 在 C++ 中，如果声明了一个指针变量但没有显式初始化，该指针会自动被初始化为 `nullptr` 。

第 7 题 下面代码没有语法错误。

```
struct GameCharacter {
    string name;
    int level;
    float position_x;
    float position_y;

    struct Equipment {
        string weapon;
        int attack_bonus;
        int defense_bonus;
    } equipment;

    struct Skill {
        string name;
        int damage;
    } skills[8];
    int skill_count;
};
```

第 8 题 下面程序能够把 `Hello` 写入 `data.txt` 文件中。

```
ofstream fout("data.txt");
cout << "Hello";
fout.close();
```

第 9 题 由于选择排序和插入排序的时间复杂度均为 $O(n^2)$，在任何实际场景下两者的性能表现几乎相同，可以互相替代。

第 10 题 下面用递推方式计算斐波那契数列第 `n` 项的程序，时间复杂度是 $O(2^n)$。

```
int fib(int n) {
    if (n <= 1) return n;
    int f0 = 0, f1 = 1, cur = 0;
    for (int i = 2; i <= n; i++) {
        cur = f0 + f1;
        f0 = f1;
        f1 = cur;
    }
    return cur;
}
```

3 编程题（每题 25 分，共 50 分）

3.1 编程题 1

- **试题名称**：山之谷
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.1.1 题目描述

现有一片山地，可以视为一个 N 行 M 列的网格图，第 i 行 j 列的海拔为 $h_{i,j}$。

如果一个单元格的海拔不高于其所有相邻单元格（**相邻包括上、下、左、右、左上、右上、左下、右下**，最多 8 个方向）的海拔，则称该单元格为山谷。

请你数一数该片山地中有多少山谷。

3.1.2 输入格式

第一行包含 2 个整数 N, M，表示山地的大小。

之后 N 行，每行包含 M 个整数 $h_{i,1}, h_{i,2}, \cdots, h_{i,M}$，表示海拔。

3.1.3 输出格式

输出 1 行，包含 1 个整数 C，表示山谷的数量。

3.1.4 样例

3.1.4.1 输入样例

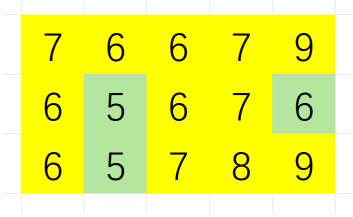
```
3 5
7 6 6 7 9
6 5 6 7 6
6 5 7 8 9
```

3.1.4.2 输出样例

```
3
```

3.1.5 样例解释

样例 1 如图所示，绿色单元格代表山谷：



3.1.6 数据范围

保证 $1 \leq N, M \leq 100$，$1 \leq h_{i,j} \leq 10^5$。

3.1.7 参考程序

```cpp
#include <iostream>

using namespace std;

int main() {
    int n, m;
    int h[105][105];

    cin >> n >> m;
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= m; j++)
            cin >> h[i][j];
    for(int i = 0; i <= max(n, m) + 1; i++)
        h[i][0] = h[0][i] = h[i][m + 1] = h[n + 1][i] = 1e9;

    int ans = 0;
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= m; j++) {
            bool ok = true;
            for(int i2 = i - 1; i2 <= i + 1; i2++)
                for(int j2 = j - 1; j2 <= j + 1; j2++)
                    if(h[i][j] > h[i2][j2]) {
                        ok = false;
                        break;
                    }
            ans += ok;
        }
    cout << ans;
    return 0;
}
```

3.2 编程题 2

- **试题名称**：礼盒排序
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.2.1 题目描述

商店推出了许多礼盒，每个礼盒中包含 **k 件商品**，每件商品都有一个价格。

现在需要对这些礼盒进行排序，排序规则如下：

1. 先按**礼盒总价格从小到大**排序；
2. 如果总价格相同，按**礼盒中最贵商品的价格从小到大**排序；
3. 如果仍然相同，按**礼盒中最便宜商品的价格从小到大**排序；
4. 如果仍然相同，按**礼盒编号从小到大**排序。

请输出排序后的礼盒编号。

3.2.2 输入格式

第一行包含两个整数 n 和 k，分别表示礼盒数量和每个礼盒中商品的数量。

接下来 n 行，每行包含 k 个整数，第 i 行表示第 i 个礼盒中各商品的价格。

3.2.3 输出格式

输出一行，包含排序后的礼盒编号（编号从 1 开始），用空格分隔。

3.2.4 样例

3.2.4.1 输入样例

```
4 3
3 5 2
4 1 5
2 2 4
3 4 3
```

3.2.4.2 输出样例

```
3 4 2 1
```

3.2.5 样例解释

4 个礼盒分别为：

编号	商品价格	总价	最大值	最小值
1	3 5 2	10	5	2
2	4 1 5	10	5	1
3	2 2 4	8	4	2
4	3 4 3	10	4	3

排序过程：

1. 按总价排序，3号礼盒总价最小；
2. 其余总价均为 10，再按最大值排序，4号最大值更小；
3. 1号和2号最大值相同，再按最小值排序，2号更小。

最终顺序为：3 4 2 1

3.2.6 数据范围

保证 $1 \le n \le 10^3$，$1 \le k \le 10$，商品价格 $\le 10^4$。

3.2.7 参考程序

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

struct Combo {
    int sum, mx, mn, id;
};

bool cmp(const Combo &a, const Combo &b) {
    if (a.sum != b.sum) return a.sum < b.sum;
    if (a.mx != b.mx) return a.mx < b.mx;
    if (a.mn != b.mn) return a.mn < b.mn;
    return a.id < b.id;
}

int main() {
    int n, k;
    cin >> n >> k;

    vector<Combo> v(n);

    for (int i = 0; i < n; i++) {
        v[i].sum = 0;
        v[i].mx = -1;
        v[i].mn = 1e9;
        v[i].id = i + 1;

        for (int j = 0; j < k; j++) {
            int x;
            cin >> x;
            v[i].sum += x;
            v[i].mx = max(v[i].mx, x);
            v[i].mn = min(v[i].mn, x);
        }
    }

    sort(v.begin(), v.end(), cmp);

    for (int i = 0; i < n; i++) {
        cout << v[i].id;
        if (i + 1 < n) cout << " ";
    }
    cout << endl;

    return 0;
}
```