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

C++ 四级

2025 年 12 月

1 单选题（每题 2 分，共 30 分）

第 1 题 小杨想让指针 `p` 指向整数变量 `x`，正确写法是（ ）。

- [ ] **A.**

```
int p = &x;
```

- [ ] **B.**

```
int *p = x;
```

- [ ] **C.**

```
int *p = &x;
```

- [ ] **D.**

```
p = *x;
```

第 2 题 小杨写了如下的指针接力程序，程序执行完后变量 `a`、`*p1` 和 `*p2` 的值分别是（ ）。

```
int a = 5;
int* p1 = &a;
int* p2 = p1;
*p2 = 10;
```

- [ ] **A.** `5 10 10`
- [ ] **B.** `5 10 15`
- [ ] **C.** `10 10 10`
- [ ] **D.** `5 5 10`

第 3 题 小杨用一个二维数组表示棋盘，其中 1 表示有棋子，0 表示没有棋子。他想知道第 2 行 第 3 列有没有棋子，可采用的代码是：（ ）。

```
int a[3][4] = {
    {1, 0, 1, 0},
    {0, 1, 0, 1},
    {1, 1, 0, 0}
};
```

- [ ] **A.** `cout << a[1, 2] << endl;`

- [ ] **B.** `cout << a[1][2] << endl;`
- [ ] **C.** `cout << a(1, 2) << endl;`
- [ ] **D.** `cout << a{1}{2} << endl;`

第 4 题 执行完下面的代码后，`*(p + 5)` 和 `arr[1][1]` 的值分别是（ ）。

```
int arr[3][4] = {{1,2,3,4}, {5,6,7,8}, {9,10,11,12}};
int* p = &arr[0][0];
```

- [ ] **A.** `5 6`
- [ ] **B.** `6 5`
- [ ] **C.** `5 5`
- [ ] **D.** `6 6`

第 5 题 执行完下面的代码后，`sum` 的值是（ ）。

```
int arr[2][3][2] = {
        {{1,2}, {3,4}, {5,6}},
        {{7,8}, {9,10}, {11,12}}
    };
int sum = 0;
for(int i = 0; i < 2; i++)
    for(int j = 0; j < 3; j++)
        for(int k = 0; k < 2; k++)
            if((i+j+k) % 2 == 0)
                sum += arr[i][j][k];
```

- [ ] **A.** `36`
- [ ] **B.** `39`
- [ ] **C.** `78`
- [ ] **D.** `30`

第 6 题 执行完下面的代码后，输出是（ ）。

```
int a = 1;

void test() {
    int a = 2;
    {
        int a = 3;
        a++;
    }
    a++;
    cout << a << " ";
}

int main() {
    test();
    cout << a;
    return 0;
}
```

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

- [ ] **D.** `4 2`

第 7 题 执行完下面的代码后，`a` 、`b` 和 `c` 的值分别是（ ）。

```
void byValue(int x) { x = 100; }
void byRef(int& x) { x = 200; }
void byPointer(int* x) { *x = 300; }

int main() {
    int a = 1, b = 2, c = 3;
    byValue(a);
    byRef(b);
    byPointer(&c);
    return 0;
}
```

- [ ] **A.** `100 200 300`
- [ ] **B.** `1 2 3`
- [ ] **C.** `1 200 300`
- [ ] **D.** `1 2 300`

第 8 题 运行如下代码会输出（ ）。

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

struct Rectangle {
    Point topLeft;
    Point bottomRight;
};

int main() {
    Rectangle rect = {{10, 10}, {20, 20}};
    rect.topLeft.x = 5;
    Point* p = &rect.bottomRight;
    p->y = 5;
    cout << rect.topLeft.x + rect.bottomRight.y;
    return 0;
}
```

- [ ] **A.** `10`
- [ ] **B.** `30`
- [ ] **C.** `15`
- [ ] **D.** `20`

第 9 题 给定函数 `climbStairs(int n)` 的定义如下，则 `climbStairs(5)` 的返回的值是（ ）。

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

- [ ] **A.** `5`
- [ ] **B.** `8`
- [ ] **C.** `13`
- [ ] **D.** `10`

第 10 题 对如下4个扑克牌进行排序，

```
struct Card {
    int value;
    char suit;  // 花色
};

Card cards[4] = {{5,'A'}, {3,'B'}, {5,'C'}, {3,'D'}};
```

使用某排序算法按value排序后，结果为: `{3,'D'}, {3,'B'}, {5,'A'}, {5,'C'}` ，则这个排序算法是稳定的吗?

- [ ] **A.** 稳定，因为相同 `value` 的元素相对顺序保持不变
- [ ] **B.** 不稳定，因为 `{3,'D'}` 出现在 `{3,'B'}` 之前
- [ ] **C.** 无法判断
- [ ] **D.** 稳定，因为结果是有序的

第 11 题 下面的函数 `selectTopK()` 实现从 `n` 个学生中选出前 `k` 名成绩最好的学生颁发奖学金（不需要对所有学生完全排序，只需要找出前 `k` 名），则横线上应填写（ ）。

```
struct Student {
    string name;
    int score;
};

void selectTopK(Student students[], int n, int k) {
    for (int i = 0; i < k; i++) {
        int maxIdx = i;
        for (____________________) {    // 在此处填入代码
            if (students[j].score > students[maxIdx].score) {
                maxIdx = j;
            }
        }

        if (maxIdx != i) {
            Student temp = students[i];
            students[i] = students[maxIdx];
            students[maxIdx] = temp;
        }
    }
}
```

- [ ] **A.** `int j = 0; j < n; j++`
- [ ] **B.** `int j = i + 1; j < n; j++`
- [ ] **C.** `int j = i; j < n; j++`
- [ ] **D.** `int j = 1; j <= n; j++`

第 12 题 某游戏的排行榜系统需要实时更新玩家分数。每次只有一个玩家的分数发生变化，排行榜已经是按分数降序排列的。现在需要将更新后的玩家调整到正确位置。下面的函数 `updateRanking()` 要实现上述功能，则两处横线上应分别填写（ ）。

```
struct Player {
    string name;
    int score;
};

// 玩家索引playerIdx的分数刚刚更新，需要调整位置
void updateRanking(Player players[], int size, int playerIdx) {
    Player updatedPlayer = players[playerIdx];

    if (playerIdx > 0 && updatedPlayer.score > players[playerIdx - 1].score) {
        int i = playerIdx;
        while (____________________) {  // 在此处填入代码
            players[i] = players[i - 1];
            i--;
        }
        players[i] = updatedPlayer;
    }
    else if (playerIdx < size - 1 && updatedPlayer.score < players[playerIdx + 1].score) {
        int i = playerIdx;
        while (____________________) {  // 在此处填入代码
            players[i] = players[i + 1];
            i++;
        }
        players[i] = updatedPlayer;
    }
}
```

- [ ] **A.**

```
i > 0 && updatedPlayer.score > players[i - 1].score
i < size - 1 && updatedPlayer.score < players[i + 1].score
```

- [ ] **B.**

```
i < size - 1 && updatedPlayer.score < players[i + 1].score
i > 0 && updatedPlayer.score > players[i - 1].score
```

- [ ] **C.**

```
i > 0 && updatedPlayer.score < players[i - 1].score
i < size - 1 && updatedPlayer.score < players[i + 1].score
```

- [ ] **D.**

```
i > 0 && updatedPlayer.score < players[i - 1].score
i < size - 1 && updatedPlayer.score > players[i + 1].score
```

第 13 题 给定如下算法，其时间复杂度为（ ）。

```cpp
bool f(int arr[], int n, int target) {
    for (int i = 0; i < n; i++) {
        int sum = 0;
        for (int j = 0; j < n; j++) {
            if (i & (1 << j)) {
                sum += arr[j];
            }
        }
        if (sum == target) return true;
    }
    return false;
}
```

- [ ] **A.** $O(n)$
- [ ] **B.** $O\left(n^2\right)$
- [ ] **C.** $O\left(n^3\right)$
- [ ] **D.** $O\left(2^n\right)$

第 14 题 执行下面 C++ 程序，会输出（　）。

```cpp

int main() {
    ofstream fout("test.txt");
    fout << "Happy" << endl;
    fout << "New Year";
    fout.close();

    ifstream fin("test.txt");
    string s1, s2;
    fin >> s1;
    getline(fin, s2);
    fin.close();

    cout << s1 << "|" << s2;
    return 0;
}
```

- [ ] **A.** `Happy|New Year`
- [ ] **B.** `Happy| New Year`
- [ ] **C.** `HappyNew Year|`
- [ ] **D.** `Happy|`

第 15 题 执行下面C++代码，会输出( )。

```cpp
int divide(int a, int b) {
    if(b == 0) throw "Division by zero";
    return a / b;
}

int main() {
    int result = 0;
    try {
        result = divide(10, 0);
        cout << "A";
    }
    catch(const char* msg) {
        cout << "B";
        result = -1;
    }
    cout << result;
    return 0;
}
```

- ☐ **A.** `A0`
- ☐ **B.** `B-1`
- ☐ **C.** `A10`
- ☐ **D.** 程序崩溃

2　判断题（每题 2 分，共 20 分）

第 1 题 小杨正在调试他的温度传感器程序，其中变量 `x` 保存当前温度。下面这段代码运行后，变量 `x` 的值变成了 `8`。

```cpp
int x = 5;
int *p = &x;
*p = *p + 3;
```

第 2 题 一个结构体不能包含另一个结构体。

第 3 题 在 C++ 中，定义如下二维数组：`int a[3][4];`，数组 `a` 在内存中是按行优先连续存放的，即 `a[0][0]`、`a[0][1]`、`a[0][2]`、`a[0][3]` 在内存中是连续的。

第 4 题 执行下面程序后，变量 `a` 的值会变成 `15`。

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

int a = 5;
add(a);
```

第 5 题 执行下面的C++代码，会输出 `8`，因为两个指针地址相差 `8` 个字节（假设 `int` 占 `4` 字节）。

```cpp
int arr[5] = {1, 2, 3, 4, 5};
int* p1 = arr;
int* p2 = arr + 2;

cout << p2 - p1;  // 输出结果
```

第 6 题 考虑用如下递推方式计算斐波那契数列，时间复杂度是 $O(n)$。

```
int n = 10;
int f[20];
f[0] = 0;
f[1] = 1;
for (int i = 2; i <= n; i++)
    f[i] = f[i - 1] + f[i - 2];
```

第 7 题 冒泡排序和插入排序都是稳定排序算法。

第 8 题 下面这段代码实现了选择排序算法。

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

第 9 题 下面代码可以正常编译并输出 `10`。

```
#include <iostream>
using namespace std;

int calculate(int x, int y = 10);

int main() {
    cout << calculate(5);  // 调用1
    return 0;
}

int calculate(int x, int y) {
    return x * y;
}

int calculate(int x) {  // 重载函数
    return x * 2;
}
```

第 10 题 执行下面代码会输出 `100`。

```
int main() {
    ofstream fout("data.txt");
    fout << 10 << " " << 20 << endl;
    fout << 30 << " " << 40;
    fout.close();

    ifstream fin("data.txt");
    int a, b, c, d;
    fin >> a >> b >> c >> d;
    fin.close();

    cout << a + b + c + d;
    return 0;
}
```

3 编程题（每题 25 分，共 50 分）

3.1 编程题 1

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

3.1.1 题目描述

小 A 有一张 M 行 N 列的地形图，其中第 i 行第 j 列的数字 a_{ij} 代表坐标 (i,j) 的海拔高度。

停机坪为一个 3×3 的区域且内部所有 9 个点的最大高度和最小高度之差不超过 H。

小 A 想请你计算出，在所有适合建造停机坪的区域中，区域内部 9 个点海拔之和最大是多少。

3.1.2 输入格式

第一行三个正整数 M,N,H，含义如题面所示。

之后 M 行，第 i 行包含 N 个整数 $a_{i1},a_{i2},\ldots,a_{iN}$，代表坐标 (i,j) 的高度。

数据保证总存在一个适合建造停机坪的区域。

3.1.3 输出格式

输出一行，代表最大的海拔之和。

3.1.4 样例

3.1.4.1 输入样例

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

3.1.4.2 输出样例

```
40
```

3.1.5 数据范围

对于所有测试点，保证 $1\le M,N\le 10^3, 1\le H, a_{ij}\le 10^5$。

3.2 编程题 2

- **试题名称**：优先购买
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.2.1 题目描述

小 A 有 M 元预算。商店有 N 个商品，每个商品有商品名 S、价格 P 和优先级 V 三种属性，其中 V 为正整数，且 V 越小代表商品的优先级越高。

小 A 的购物策略为：

- 总是优先买优先级最高的东西；
- 如果有多个最高优先级商品，购买价格最低的；
- 如果有多个优先级最高且价格最低的商品，购买商品名字典序最小的。

小 A 想知道能购买哪些商品。

3.2.2 输入格式

第一行两个正整数 M, N，代表预算和商品数。

之后 N 行，每行一个商品，依次为 S_i P_i V_i，代表第 i 个商品的商品名、价格、优先级。

数据保证不存在两个名字相同的商品。

3.2.3 输出格式

按照字典序从小到大的顺序，输出所有购买商品的商品名。

3.2.4 样例

3.2.4.1 输入样例

```
20 4
apple 6 8
bus 15 1
cab 1 10
water 4 8
```

3.2.4.2 输出样例

```
bus
cab
water
```

3.2.5 数据范围

对于所有测试点，保证 $1 \leq |S_i| \leq 10, 1 \leq M, P_i \leq 10^5, 1 \leq N \leq 10^3, 1 \leq V_i \leq 10$。商品名仅由小写字母组成且不存在两个相同的商品名。