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

C++ 四级

2025 年 06 月

1 单选题（每题 2 分，共 30 分）

第 1 题 在C++中，声明一个指向整型变量的指针的正确语法是（ ）。

- A. `int* ptr;`
- B. `*int ptr;`
- C. `int ptr*;`
- D. `ptr int;`

第 2 题 下面的函数接收一个3行4列的二维数组并输出其中元素，则横线上**不能**填写（ ）。

```
void printArray(________) {
    for (int i = 0; i < 3; ++i)
        for (int j = 0; j < 4; ++j)
            std::cout << arr[i][j] << " ";
}
```

- A. `int arr[3][4]`
- B. `int arr[][4]`
- C. `int (*arr)[4]`
- D. `int** arr`

第 3 题 在C++中，`int arr[3][4]` 和 `int* arr = new int[12]` 均可模拟一个3行4列的二维数组。关于这两种方式，下面说法错误的是（ ）。

- A. `int arr[3][4]` 在栈上分配空间，适合数组较小的情况；
- B. `int* arr = new int[12]` 在堆上分配空间，数组较大时也适用；
- C. 这两种方式申请的内存空间都是连续的。
- D. 这两种方式申请的内存都能自动释放。

第 4 题 关于以下 C++代码，说法正确的是（ ）。

```cpp
int main() {
    greet();
    return 0;
}

void greet() {
    cout << "Hello!" << endl;
}
```

- [ ] **A.** 正确编译并输出 `Hello!`
- [ ] **B.** 编译错误：找不到函数 `greet()`
- [ ] **C.** 编译警告但可以运行
- [ ] **D.** 链接错误

第 5 题 在C++中，如果希望通过函数修改传入的结构体对象的内容，应该使用哪种参数传递方式?

- [ ] **A.** 值传递或引用传递
- [ ] **B.** 值传递或指针传递
- [ ] **C.** 引用传递或指针传递
- [ ] **D.** 仅指针传递

第 6 题 以下哪个选项正确描述了C++中形参和实参的区别?

- [ ] **A.** 形参是函数调用时传递给函数的具体值，实参是函数定义中声明的变量。
- [ ] **B.** 形参是函数定义中声明的变量，实参是函数调用时传递给函数的具体值。
- [ ] **C.** 形参和实参在函数调用时是完全相同的。
- [ ] **D.** 形参只在函数内部可见，实参在函数外部可见。

第 7 题 运行如下代码会输出（ ）。

```cpp
int value = 100;

void print1() {
    int value = 50;
    cout << value << " ";
    cout << ::value << " ";
}

void print2() {
    cout << value << " ";
}

print1();
print2();
```

- [ ] **A.** `100 100 100`
- [ ] **B.** `50 50 50`
- [ ] **C.** `50 100 100`
- [ ] **D.** `50 50 100`

第 8 题 小杨在整理一副扑克牌的所有红心扑克牌，使其从小到大排列。他的做法是：最开始抓到第1张扑克牌被认为已经排好序；然后抓第2张扑克牌，将其插入至有序部分的正确位置；不断循环步骤，每次将新抓到扑克牌插入至有序部分，直至抓完所有扑克牌，这样抓牌结束时就完成了扑克牌的排序。小杨这种整理扑克牌的方式与（ ）排序的方式最接近。

- [ ] **A.** 冒泡排序
- [ ] **B.** 插入排序
- [ ] **C.** 选择排序
- [ ] **D.** 直接排序

第 9 题 以下哪种情况是使用插入排序的合适场景？

- [ ] **A.** 数据量非常大，且乱序严重
- [ ] **B.** 希望获得稳定排序，但不要求实时性
- [ ] **C.** 数据几乎有序，只需少量调整
- [ ] **D.** 想在交换次数最少的前提下排好大数组

第 10 题 以下关于递推算法基本思想的描述，正确的是（ ）。

- [ ] **A.** 递推算法通过将问题分解为相互独立的子问题来解决。
- [ ] **B.** 递推算法从已知的基础情况出发，通过某种关系逐步推导出更大规模问题的解。
- [ ] **C.** 递推算法通常用于穷举所有可能的解决方案。
- [ ] **D.** 递推算法适用于在每一步做出局部最优选择以达到全局最优。

第 11 题 给定如下算法，其时间复杂度为（ ）。

```
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 12 题 下述斐波那契数列计算的时间复杂度是（ ）。

```cpp
int fibonacci(int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;
    return fibonacci(n - 1) + fibonacci(n - 2);
}

```

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

第 13 题 关于下面 C++ 程序的描述，（　）最准确。

```cpp
ifstream in("data.txt");
string line;
while (getline(in, line)) {
    cout << line << endl;
}
```

- [ ] **A.** 将从标准输入读取每行，并输出到屏幕
- [ ] **B.** 程序无法运行，因为 `getline` 只能读取 `cin`
- [ ] **C.** 将 data.txt 中的每一行读取并输出到屏幕
- [ ] **D.** 程序将创建 data.txt 并写入默认文本

第 14 题 在C++中，异常处理机制（try-catch块）的主要目的是( )。

- [ ] **A.** 提高程序的运行速度。
- [ ] **B.** 在程序发生运行时错误时，提供一种结构化的错误处理方式。
- [ ] **C.** 确保程序在编译时没有错误。
- [ ] **D.** 减少程序的内存占用。

第 15 题 为了提高冒泡排序的效率，如果某轮“冒泡”中没有执行任何交换操作，说明数组已经完成排序，可直接返回结果，则两条横线上分别应该填写（ ）。

```cpp
void bubbleSortWithFlag(vector<int> &nums) {
    for (int i = nums.size() - 1; i > 0; i--) {
        bool flag;
        ________________    // 在此处填入代码

        for (int j = 0; j < i; j++) {
            if (nums[j] > nums[j + 1]) {
                swap(nums[j], nums[j + 1]);
                ____________________________    // 在此处填入代码
            }
        }
        if (!flag)
            break;
    }
}
```

- [ ] **A.**

```
flag = false;
flag = false;
```

☐ **B.**

```
flag = false;
flag = true;
```

☐ **C.**

```
flag = true;
flag = false;
```

☐ **D.**

```
flag = true;
flag = true;
```

2　判断题（每题 2 分，共 20 分）

第 1 题 下面C++代码正确声明了一个返回 `int` 类型、接受两个 `int` 参数的函数。

```
int add(int, int);
```

第 2 题 下面C++代码的输出是 `15` 。

```
void foo(int x) {
    x += 5;
}
int main() {
    int a = 10;
    foo(a);
    cout << a << endl;
}
```

第 3 题 下面c++代码在一个结构体中又定义了别的结构体。这种结构嵌套定义的方式语法不正确。

```
#include <string>
#include <vector>

using namespace std;

struct Library {
    struct Book {
        struct Author {
            string name;
            int birthYear;
        };

        string title;
        int year;
        Author author;
    };

    string name;
    vector<Book> books;
```

```
};
```

第 4 题 在C++中，相比于值传递，使用引用传递作的优点可以直接操作和修改原始变量，避免数据拷贝，提高效率。

第 5 题 下面这段代码不合法，因为每一行都必须显式初始化3个元素。

`int arr[2][3] = {{1, 2}, {3}};`

第 6 题 以下程序中使用了递推方式计算阶乘（$n! = 1 \times 2 \ldots \times n$），计算结果正确。

```cpp
int factorial(int n) {
    int res = 1;
    for (int i = 0; i < n; ++i) {
        res *= i;
    }
    return res;
}
```

第 7 题 无论初始数组是否有序，选择排序都执行 $O\left(n^2\right)$ 次比较

第 8 题 以下C++代码，尝试对有 `n` 个整数的数组 `arr` 进行排序。这个代码实现了选择排序算法。

```cpp
for (int i = 0; i < n - 1; ++i) {
    int minIndex = i;
    for (int j = i + 1; j < n; ++j) {
        if (arr[j] < arr[minIndex])
            minIndex = j;
    }
    if (minIndex != i)
        swap(arr[i], arr[minIndex]);
}
```

第 9 题 如果一个异常在 try 块中抛出但没有任何 catch 匹配，它将在编译时报错。

第 10 题 下面C++代码实现将 `Hello` 写入 `data.txt` 。

```cpp
ofstream out("data.txt");
out << "Hello";
out.close();
```

3 编程题（每题 25 分，共 50 分）

3.1 编程题 1

- **试题名称**：画布裁剪
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.1.1 题目描述

小 A 在高为 h 宽为 w 的矩形画布上绘制了一幅画。由于画布边缘留白太多，小 A 想适当地裁剪画布，只保留画的主体。具体来说，画布可以视为 h 行 w 列的字符矩阵，其中的字符均为 ASCII 码位于 33 ~ 126 之间的可见字符，小 A 只保留画布中由第 x_1 行到第 x_2 行、第 y_1 列到第 y_2 列构成的子矩阵。

小 A 将画布交给了你，你能帮他完成画布的裁剪吗?

3.1.2 输入格式

第一行，两个正整数 h, w，分别表示画布的行数与列数。

第二行，四个正整数 x_1, x_2, y_1, y_2，表示保留的行列边界。

接下来 h 行，每行一个长度为 w 的字符串，表示画布内容。

3.1.3 输出格式

输出共 $x_2 - x_1 + 1$ 行，每行一个长度为 $y_2 - y_1 + 1$ 的字符串，表示裁剪后的画布。

3.1.4 样例

3.1.4.1 输入样例 1

```
3 5
2 2 2 4
.....
.>_<.
.....
```

3.1.4.2 输出样例 1

```
>_<
```

3.1.4.3 输入样例 2

```
5 5
1 2 3 4
AbCdE
fGhIk
LmNoP
qRsTu
VwXyZ
```

3.1.4.4 输出样例 2

```
Cd
hI
```

3.1.5 数据范围

对于所有测试点，保证 $1 \le h, w \le 100$，$1 \le x_1 \le x_2 \le h$，$1 \le y_1 \le y_2 \le w$。

3.2 编程题 2

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

3.2.1 题目描述

体育课上有 n 名同学排成一队，从前往后数第 i 位同学的身高为 h_i，体重为 w_i。目前排成的队伍看起来参差不齐，老师希望同学们能按照身高从高到低的顺序排队，如果身高相同则按照体重从重到轻排序。在调整队伍时，每次只能交换相邻两位同学的位置。老师想知道，最少需要多少次交换操作，才能将队伍调整成目标顺序。

3.2.2 输入格式

第一行，一个正整数 n，表示队伍人数。

接下来 n 行，每行两个正整数 h_i 和 w_i，分别表示第 i 位同学的身高和体重。

3.2.3 输出格式

输出一行，一个整数，表示最少需要的交换次数。

3.2.4 样例

3.2.4.1 输入样例 1

```
5
1 60
3 70
2 80
4 55
4 50
```

3.2.4.2 输出样例 1

```
8
```

3.2.4.3 输入样例 2

```
5
4 0
4 0
2 0
3 0
1 0
```

3.2.4.4 输出样例 2

```
1
```

3.2.5 数据范围

对于所有测试点，保证 $1 \leq n \leq 3000$，$0 \leq h_i, w_i \leq 10^9$。