



C++ 四级

2026 年 09 月

1 单选题（每题 2 分，共 30 分）

第 1 题 小杨使用指针修改计数器的值。执行下面程序后，输出结果是（ ）。

```
1 int count = 8;  
2 int *p = &count;  
3 *p += 4;  
4 cout << count << " " << *p;  
5 return 0;
```

- ☐ A. 8 8
- ☐ B. 8 12
- ☐ C. 12 12
- ☐ D. 12 8

第 2 题 关于下面指针声明的说法，正确的是（ ）。

```
1 int a = 10;  
2 int b = 20;  
3 const int *p = &a;
```

- ☐ A. 可以通过 p 修改 a 的值
- ☐ B. 可以令 p = &b
- ☐ C. p 的指向和所指向的值都不能修改
- ☐ D. p 必须始终指向 a

第 3 题 小杨用二维数组记录仓库货物数量。执行下面代码后，变量 x 的值是（ ）。

```
1 int goods[3][4] = {{2, 4, 6, 8}, {10, 12, 14, 16}, {18, 20, 22, 24}};  
2 int (*p)[4] = goods;  
3 int x = *(*p + 1) + 2;
```

- ☐ A. 12
- ☐ B. 14
- ☐ C. 20
- ☐ D. 22

第 4 题 下面函数用于将一个 3 行 5 列二维数组的第 r 行元素全部加 1，横线处正确的形参写法是（ ）。

```

1 void addOne(_____, int r) {
2     for (int j = 0; j < 5; j++) {
3         arr[r][j]++;
4     }
5 }

```

- ☐ A. int arr[][]
- ☐ B. int **arr
- ☐ C. int arr[][5]
- ☐ D. int arr[5][]

第5题 执行下面程序后，输出结果是（ ）。

```

1 int score = 60;
2
3 void update(int &score) {
4     score += 5;
5 }
6
7 int main() {
8     int score = 80;
9     update(score);
10    cout << score << " " << ::score;
11    return 0;
12 }

```

- ☐ A. 85 60
- ☐ B. 80 65
- ☐ C. 85 65
- ☐ D. 80 60

第6题 执行下面程序后，输出结果是（ ）。

```

1 struct Device {
2     int id;
3     int state;
4 };
5
6 void reset(Device d) {
7     d.state = 0;
8 }
9
10 void start(Device &d) {
11     d.state += 1;
12 }
13
14 int main() {
15     Device d{7, 2};
16     reset(d);
17     start(d);
18     cout << d.id << " " << d.state;
19     return 0;
20 }

```

- ☐ A. 7 0
- ☐ B. 7 1

☐ C. 7 2

☐ D. 7 3

第7题 小杨定义了结构体数组，并使用指针访问其中的元素。执行下面代码后输出的是（ ）。

```
1 struct Book {
2     string name;
3     int pages;
4 };
5
6 int main() {
7     Book books[2] = {"C++", 120}, {"Math", 150};
8     Book *p = books + 1;
9     p->pages += 10;
10    cout << books[1].name << " " << books[1].pages;
11    return 0;
12 }
```

☐ A. C++ 120

☐ B. Math 150

☐ C. Math 160

☐ D. C++ 160

第8题 关于冒泡排序、插入排序和选择排序，下列说法正确的是（ ）。

☐ A. 三种排序算法的最坏时间复杂度都是 $O(n)$

☐ B. 冒泡排序只能从小到大排序，不能从大到小排序

☐ C. 插入排序每次将一个待排序元素插入前面已经有序的序列中

☐ D. 选择排序每轮只需要比较一次就能确定最小元素

第9题 某机器人每次可以向前移动1格或2格，到达第 n 格的方法数由下面函数计算。ways(6) 的返回值是（ ）。

```
1 int ways(int n) {
2     if (n <= 2)
3         return n;
4     int a = 1, b = 2, c = 0;
5     for (int i = 3; i <= n; i++) {
6         c = a + b;
7         a = b;
8         b = c;
9     }
10    return b;
11 }
```

☐ A. 8

☐ B. 10

☐ C. 13

☐ D. 21

第10题 对一组 struct student 的学生按成绩（score）升序排序。排序前后的数据如下。关于该排序的稳定性，判断正确的是（ ）。

```

1 struct student {
2     int score;
3     char id;
4 };

```

排序前：

```

1 (90, 'A'), (80, 'B'), (90, 'C'), (80, 'D')

```

排序后：

```

1 (80, 'B'), (80, 'D'), (90, 'C'), (90, 'A')

```

- ☐ A. 稳定，因为所有成绩已经按升序排列
- ☐ B. 稳定，因为分数相同不会影响排序结果
- ☐ C. 不稳定，因为相同成绩的 (90, 'A') 和 (90, 'C') 的相对顺序发生了改变
- ☐ D. 无法判断，因为没有给出排序算法的代码

第 11 题 下面代码使用插入排序将数组按升序排列，横线处应填写（ ）。

```

1 void insertionSort(int a[], int n) {
2     for (int i = 1; i < n; i++) {
3         int key = a[i];
4         int j = i - 1;
5         while (j >= 0 && _____) {
6             a[j + 1] = a[j];
7             j--;
8         }
9         a[j + 1] = key;
10    }
11 }

```

- ☐ A. $a[j] < \text{key}$
- ☐ B. $a[j] > \text{key}$
- ☐ C. $a[j] == \text{key}$
- ☐ D. $a[j + 1] > \text{key}$

第 12 题 下面代码的时间复杂度是（ ）。

```

1 int countPairs(int a[], int n) {
2     int cnt = 0;
3     for (int i = 0; i < n; i++) {
4         for (int j = i + 1; j < n; j++) {
5             if (a[i] + a[j] == 100) {
6                 cnt++;
7             }
8         }
9     }
10    return cnt;
11 }

```

- ☐ A. $O(1)$
- ☐ B. $O(n^3)$
- ☐ C. $O(n)$

☐ D. $O(n^2)$

第 13 题 假设文件 data.txt 的内容如下:

```
1 | Blue Sky
```

执行下面程序后, 输出结果是 ()。

```
1 | int main() {  
2 |     ifstream fin("data.txt");  
3 |     string a, b;  
4 |     fin >> a >> b;  
5 |     cout << b << "-" << a;  
6 |     fin.close();  
7 |     return 0;  
8 | }
```

☐ A. Blue-Sky

☐ B. Sky-Blue

☐ C. Blue Sky

☐ D. Sky Blue

第 14 题 执行下面程序后, 输出结果是 ()。

```
1 | int main() {  
2 |     try {  
3 |         int age = -1;  
4 |         if (age < 0)  
5 |             throw age;  
6 |         cout << "A";  
7 |     } catch (const char *msg) {  
8 |         cout << "B";  
9 |     } catch (int value) {  
10 |         cout << "C" << value;  
11 |     }  
12 |     return 0;  
13 | }
```

☐ A. A

☐ B. B

☐ C. C-1

☐ D. 程序崩溃

第 15 题 下面函数使用冒泡排序将数组按升序排列。为了在数组已经有序时提前结束, 两处横线应分别填写 ()。

```

1 void bubbleSort(int a[], int n) {
2     for (int i = n - 1; i > 0; i--) {
3         bool changed = _____;
4         for (int j = 0; j < i; j++) {
5             if (a[j] > a[j + 1]) {
6                 int t = a[j];
7                 a[j] = a[j + 1];
8                 a[j + 1] = t;
9                 changed = _____;
10            }
11        }
12        if (!changed)
13            break;
14    }
15 }

```

- ☐ A. false , true
- ☐ B. true , false
- ☐ C. false , false
- ☐ D. true , true

2 判断题（每题 2 分，共 20 分）

第 1 题 执行下面程序后，变量 a 的值为 15。

```

1 int a = 10;
2 int *p = &a;
3 *p += 5;

```

第 2 题 一个函数必须在调用之前既声明又定义。

第 3 题 下面二维数组在内存中按行优先连续存储，因此 $*(*(a + 1) + 0)$ 的值为 5。

```

1 int a[2][4] = {{1, 2, 3, 4}, {5, 6, 7, 8}};

```

第 4 题 执行下面程序后会输出 20。

```

1 void change(int x) {
2     x = 20;
3 }
4
5 int main() {
6     int x = 10;
7     change(x);
8     cout << x;
9     return 0;
10 }

```

第 5 题 下面结构体初始化语句是合法的。

```

1 struct Point {
2     int x;
3     int y;
4 };
5
6 Point p{3, 4};

```

第6题 对于按升序实现的稳定插入排序，移动元素的条件通常应为 `a[j] >= key`，这样能够保证相等元素的相对顺序不变。

```

1 while (j >= 0 && a[j] >= key) {
2     a[j + 1] = a[j];
3     j--;
4 }

```

第7题 下面递推程序计算 $n!$ 。当 $n = 4$ 时，返回值为 24。

```

1 int factorial(int n) {
2     int result = 1;
3     for (int i = 1; i <= n; i++) {
4         result *= i;
5     }
6     return result;
7 }

```

第8题 下面两层循环的时间复杂度是 $O(n^2)$ 。

```

1 for (int i = 0; i < n; i++) {
2     for (int j = 1; j < n; j *= 2) {
3         cout << i + j;
4     }
5 }

```

第9题 假设文件能够正常打开，下面程序会把 `Welcome` 写入 `log.txt`。

```

1 int main() {
2     ofstream fout("log.txt");
3     fout << "Welcome";
4     fout.close();
5     return 0;
6 }

```

第10题 执行下面程序时，`catch (int e)` 能够捕获由 `throw "Error"` 抛出的异常，因此程序输出 `Caught`。

```

1 int main() {
2     try {
3         throw "Error";
4     } catch (int e) {
5         cout << "Caught";
6     }
7     return 0;
8 }

```