



# C++ 七级

2026 年 06 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	A	C	D	C	C	D	A	A	C	D	B	B	B	D

第 1 题 下列 C++ 代码的输出结果是（ ）。

```
1 #include <iostream>
2 #include <cmath>
3 using namespace std;
4 int main() {
5     cout << (int)(sqrt(50) + log2(8));
6     return 0;
7 }
```

- A. 9
- B. 10
- C. 11
- D. 12

第 2 题 下列关于 `<cmath>` 或 `<math.h>` 中的数学库函数的说法，正确的是（ ）。

- A. `sqrt(49)` 的返回值可以参与浮点运算。
- B. `log2(32)` 的返回值类型为 `int`。
- C. `pow(2, 5)` 的返回值类型一定为 `int`。
- D. `sin(90)` 的参数 90 表示 90 度。

第 3 题 下列关于 C++ 函数参数传递的说法，正确的是（ ）。

- A. 函数形参一定和实参使用同一块内存。
- B. 值传递时，在函数内修改形参一定会修改实参。
- C. 引用形参绑定到实参后，在函数内修改引用形参通常会影响到实参。
- D. 指针形参不能用于修改实参指向的数据。

第 4 题 有 5 个字符，它们出现的次数分别为 3、4、7、8、9。使用哈夫曼编码时，最小的带权路径长度 WPL 为（ ）。

- A. 62
- B. 64
- C. 67

D. 69

第5题 已知网格上每个网格点有一个数字， $a[i][j]$  表示第  $i$  行第  $j$  列处网格点上的数字。若  $dp[i][j]$  表示从网格左上角（第 0 行第 0 列）走到第  $i$  行第  $j$  列时能取得的最大数字和，且每次只能向右或向下移动。对于  $i > 0$  且  $j > 0$  的位置，正确的状态转移代码为（ ）。

- A.  $dp[i][j] = a[i][j] + \min(dp[i - 1][j], dp[i][j - 1])$
- B.  $dp[i][j] = \max(dp[i - 1][j - 1], dp[i][j])$
- C.  $dp[i][j] = a[i][j] + \max(dp[i - 1][j], dp[i][j - 1])$
- D.  $dp[i][j] = a[i][j] + dp[i - 1][j - 1]$

第6题 已知  $f[0] = 0$ ， $f[1] = 2$ ，并且对  $i \geq 2$  有  $f[i] = \max(f[i - 1], f[i - 2] + a[i])$ 。若  $a[1..5] = \{2, 7, 9, 3, 1\}$ ，则  $f[5]$  的值为（ ）。

- A. 10
- B. 11
- C. 12
- D. 22

第7题 下面代码是一维数组优化 0/1 背包的核心片段，其中  $w[i]$  表示第  $i$  件物品的重量， $v[i]$  表示第  $i$  件物品的价值。横线处应填入（ ）。

```
1 for (int i = 1; i <= n; i++) {
2     for (int c = W; c >= w[i]; c--) {
3         -----;
4     }
5 }
```

- A.  $dp[c] = \max(dp[c], dp[c + w[i]] + v[i])$
- B.  $dp[c] = \min(dp[c], dp[c - w[i]] + v[i])$
- C.  $dp[c] = dp[c - w[i]] + v[i]$
- D.  $dp[c] = \max(dp[c], dp[c - w[i]] + v[i])$

第8题 下面程序片段主要体现的算法思想是（ ）。

```
1 void dfs(int x, int y) {
2     vis[x][y] = true;
3     for (int k = 0; k < 4; k++) {
4         int nx = x + dx[k], ny = y + dy[k];
5         if (inside(nx, ny) && a[nx][ny] == 1 && !vis[nx][ny])
6             dfs(nx, ny);
7     }
8 }
```

- A. 泛洪算法
- B. 二分查找
- C. 贪心算法
- D. 归并排序

第9题 下列关于排序稳定性的说法，正确的是（ ）。

- A. 冒泡排序在只交换相邻逆序元素时是稳定排序

- B. 选择排序一定是稳定排序
- C. 快速排序一定是稳定排序
- D. 稳定排序一定会改变相等元素的相对顺序

第 10 题 无向图的边为  $(1, 2), (1, 3), (2, 4), (3, 4), (4, 5)$ 。从顶点 1 开始进行 BFS，每轮根据出队顶点，将与其相邻顶点按编号从小到大入队，则顶点 4 第一次入队时，队列的状态为 ( )。

- A. 1, 2, 3, 4
- B. 2, 3, 4
- C. 3, 4
- D. 3, 4, 5

第 11 题 一个长度为 11、下标为 0 到 10 的哈希表采用线性探测法处理冲突，哈希函数为  $h(x) = x \% 11$ 。依次插入 22、33、4、15、26，则 26 最终存放在下标 ( )。

- A. 0
- B. 4
- C. 5
- D. 6

第 12 题 关于哈希表处理冲突的方法，下列说法正确的是 ( )。

- A. 线性探测法发生冲突后，只能放弃插入该元素。
- B. 链地址法可以把哈希到同一位置的多个元素组织在同一个桶中。
- C. 只要哈希表长度是素数，就一定不会发生冲突。
- D. 开放定址法查找元素时不需要考虑冲突位置。

第 13 题 某算法需要枚举  $n$  个对象；对每个对象，还需要进行一次二分查找。若二分查找的对象规模也是  $n$ ，则该算法的时间复杂度通常为 ( )。

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

第 14 题 在升序数组中用二分查找第一个大于等于  $x$  的位置。若当前中点  $mid$  满足  $a[mid] < x$ ，下一步应 ( )。

- A. 令闭区间右边界变为  $mid - 1$
- B. 令闭区间左边界变为  $mid + 1$
- C. 立即返回  $mid$
- D. 交换  $a[mid]$  与  $x$

第 15 题 在如下网格中，# 表示不能经过的格子，. 表示可以经过的格子。从左上角走到右下角，每次只能向右或向下移动，不同路径共有 ( ) 条。

```

1 | . . . . .
2 | . # . # .
3 | . . . . .
4 | # . # . .
5 | . . . . .

```

- A. 5
- B. 6
- C. 7
- D. 8

## 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 使用 `cmath` 或 `math.h` 中的三角函数时，角度参数默认采用角度制。

第 2 题 使用 `cmath` 或 `math.h` 中的 `pow(2, 10)` 计算  $2^{10}$  时，由于参数均为整型 `int`，返回值类型也为整型 `int`。

第 3 题 0/1 背包使用一维数组优化时，容量从小到大枚举也能保证每件物品最多被选一次。

第 4 题 哈希表采用开放定址法时，即使哈希函数设计合理，也仍然可能发生冲突。

第 5 题 同一个图从同一个起点进行深度优先搜索，访问序列一定与邻接点的枚举顺序无关。

第 6 题 泛洪算法可以用递归 DFS 实现，但地图很大时可能由于递归层数过深导致调用栈溢出等运行时错误。

第 7 题 哈夫曼树中不存在度为 1 的结点。

第 8 题 冒泡排序的常见实现是稳定排序，选择排序也是。

第 9 题 在无权图中从起点执行 BFS 时，某个顶点第一次被访问到的层数等于起点到该顶点经过的最少边数。

第 10 题 在二维动态规划中，状态 `dp[i][j]` 的计算常常依赖其他状态，这些状态的计算必须在完成 `dp[i][j]` 的计算前完成。

## 3 编程题（每题 25 分，共 50 分）

### 3.1 编程题 1

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

#### 3.1.1 题目描述

小杨同学有一张包含  $n$  个结点的无向图  $G$ ， $G$  中的结点依次以  $1, 2, \dots, n$  编号。

小杨同学发现  $G$  中每个结点的度数都是 2。显然  $G$  中恰好有  $n$  条边。

小杨同学想为  $G$  中的结点染色，使得任意一条边两端的结点都有不同的颜色。

小杨同学想知道最少需要多少种颜色才能在满足条件的前提下为  $G$  染色。

### 3.1.2 输入格式

本题包含多组数据。

第一行，一个正整数  $t$ ，表示数据组数。

对于每组数据：

第一行，一个正整数  $n$ ，表示无向图  $G$  中的结点数。

接下来  $n$  行，每行两个正整数  $u_i, v_i$ ，表示一条连接结点  $u_i$  与  $v_i$  的无向边，整数之间以空格分隔。

保证  $G$  中没有重边与自环。

### 3.1.3 输出格式

对于每组数据：输出一行，一个整数，表示在满足条件的前提下为  $G$  染色需要的最少颜色数。

### 3.1.4 样例

#### 3.1.5 输入样例

```
1 4
2 6
3 1 6
4 2 1
5 3 2
6 4 3
7 5 4
8 6 5
9 6
10 1 3
11 3 5
12 5 1
13 2 4
14 4 6
15 6 2
16 3
17 1 2
18 2 3
19 3 1
20 5
21 1 4
22 2 5
23 3 1
24 4 2
25 5 3
```

#### 3.1.6 输出样例

```
1 | 2
2 | 3
3 | 3
4 | 3
```

### 3.1.7 数据范围

对于 40% 的测试点，保证  $\sum n \leq 500$ ， $\sum n$  指每个输入中多组数据的  $n$  的总和。

对于所有测试点，保证  $1 \leq t \leq 100$ ， $3 \leq n \leq 10^5$ ， $\sum n \leq 10^5$ 。保证  $G$  中没有重边与自环。

### 3.1.8 参考程序

```
1 #include <iostream>
2 #include <algorithm>
3 using namespace std;
4
5 int n;
6 int a[100010], b[100010], cnt[100010];
7
8 void solve() {
9     cin >> n;
10    for (int i = 1; i <= n; i++)
11        a[i] = b[i] = cnt[i] = 0;
12    for (int i = 1; i <= n; i++) {
13        int u, v;
14        cin >> u >> v;
15        b[u] = a[u];
16        a[u] = v;
17        b[v] = a[v];
18        a[v] = u;
19    }
20    bool flag = false;
21    for (int i = 1; i <= n; i++) {
22        if (cnt[i])
23            continue;
24        int u = i, last = a[u];
25        int v = a[u] + b[u] - last;
26        while (!cnt[v]) {
27            cnt[v] = cnt[u] + 1;
28            last = u;
29            u = v;
30            v = a[u] + b[u] - last;
31        }
32        if (cnt[i] % 2 != 0)
33            flag = true;
34    }
35    if (flag)
36        cout << 3 << endl;
37    else
38        cout << 2 << endl;
39 }
40
41 int main() {
42     int t;
43     cin >> t;
44     while (t--)
45         solve();
46     return 0;
47 }
```

## 3.2 编程题 2

- 试题名称: 消消乐
- 时间限制: 1.0 s
- 内存限制: 512.0 MB

### 3.2.1 题目描述

给定一个由  $n$  个整数构成的数组  $a = [a_1, \dots, a_n]$ 。每次你可以对数组  $a$  进行以下操作，直到数组  $a$  变为空：

- 指定  $a$  中的一个元素，获得该元素两侧相邻元素之和的分数，并将该元素从  $a$  中删去。

特别地，如果相邻元素不存在则该元素的值视为 0。例如，对于  $a = [1, 2, 3]$  可以进行以下操作：

- 指定元素 2，获得分数  $1 + 3$ ，删去 2 后  $a = [1, 3]$ ；
- 指定元素 1，获得分数  $0 + 3$ ，删去 1 后  $a = [3]$ ；
- 指定元素 3，获得分数  $0 + 0$ ，删去 3 后  $a$  变为空。

请问你能获得的分数总和最大是多少？

### 3.2.2 输入格式

第一行，一个正整数  $n$ ，表示数组长度。

第二行， $n$  个非负整数  $a_1, \dots, a_n$ ，表示数组  $a$  中的整数。

### 3.2.3 输出格式

输出一行，一个整数，表示能获得的最大分数总和。

### 3.2.4 样例

#### 3.2.5 输入样例 1

```
1 | 6
2 | 1 6 3 2 9 1
```

#### 3.2.6 输出样例 1

```
1 | 55
```

#### 3.2.7 输入样例 2

```
1 | 5
2 | 3 1415 926 53 58
```

#### 3.2.8 输出样例 2

```
1 | 5771
```

### 3.2.9 数据范围

对于 40% 的测试点，保证  $1 \leq n \leq 50$ ,  $0 \leq a_i \leq 10^3$ 。

对于所有测试点，保证  $1 \leq n \leq 100$ ,  $0 \leq a_i \leq 10^9$ 。

### 3.2.10 参考程序

```
1 #include <iostream>
2 #include <algorithm>
3 using namespace std;
4
5 int n;
6 int a[110];
7 long long f[110][110];
8
9 int main() {
10     cin >> n;
11     for (int i = 1; i <= n; i++)
12         cin >> a[i];
13     for (int i = 1; i <= n; i++)
14         for (int l = 1, r = i; r <= n; l++, r++)
15             for (int k = l; k <= r; k++)
16                 f[l][r] = max(f[l][r], f[l][k - 1] + f[k + 1][r] + a[l - 1] + a[r + 1]);
17     cout << f[1][n] << endl;
18     return 0;
19 }
```