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

C++ 一级

2026 年 09 月

1 单选题（每题 2 分，共 30 分）

第 1 题 第二届世界人形机器人运动会开幕式上，机器人方阵通过控制程序来完成整场表演。表演前，工程师们使用集成开发环境（IDE）编写并调试方阵的控制程序。下列关于 IDE 功能的描述，错误的是（　　）。

- A. 可以提供代码自动补全，提高编程效率
- B. 可以提供辅助信息，帮助排查程序逻辑错误
- C. 可以直接控制机器人电机转动
- D. 支持语法高亮，使代码更易阅读

第 2 题 现代计算机自问世至今，已经经历了四个时代，即：电子管计算机（1946—1958 年）、晶体管计算机（1958—1964 年）、中小规模集成电路计算机（1965—1970 年）以及大规模和超大规模集成电路计算机（1971 年至今）。那么这里划分计算机时代所采用的主要依据是（　　）。

- A. 运算速度
- B. 存储容量
- C. 体积大小
- D. 核心元器件

第 3 题 下列 C++ 代码执行的输出是（　　）。

```
int a = 4 + -2 * 3;
cout << a;
```

- A. `-2`
- B. `6`
- C. `18`
- D. 输出错误提示信息

第 4 题 下列 C++ 表达式与 `2 * - 2 % 3` 的值相同的是（　　）。

- A. `(2 * - 2) % 3`
- B. `2 * (- 2 % 3)`
- C. `2 * - (2 % 3)`
- D. `2 % - 2 * 3`

第 5 题 如下 C++ 代码执行后，其输出是（　　）。

```
int a = 3;
int b = 4;
printf("%d#%d\n", a, b);
```

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

```
3
4
```

第 6 题 下列 C++ 代码执行后的输出是（　　）。

```
int a = 8;
int b = 10;
cout << (a % b) << " " << (a * 1 / b) << " " << (a * 1.0 / b);
```

- [ ] **A.** `8 0.8 0`
- [ ] **B.** `0 0.8 0`
- [ ] **C.** `8 0 0.8`
- [ ] **D.** `0 0 0.8`

第 7 题 执行下面代码后，`b` 的值会是（　　）。

```
int b = 8;
++b *= 2 + 1;
```

- [ ] **A.** `18`
- [ ] **B.** `17`
- [ ] **C.** `19`
- [ ] **D.** `27`

第 8 题 执行如下 C++ 代码，其输出是（　　）。

```
int i, cnt = 0;
for (i = 0; i < 10; i++)
    cnt += 1;
cout << i << ' ' << cnt;
```

- [ ] **A.** `10 10`
- [ ] **B.** `9 9`
- [ ] **C.** `10 9`
- [ ] **D.** `9 10`

第 9 题 执行如下 C++ 代码，变量都是整型，则其输出是（　　）。

```cpp
int cnt, i;
cnt = 0;
for (i = -10; i < 0; i++)
    cnt += 1;
cout << (i * cnt);
```

- [ ] **A.** `-10`
- [ ] **B.** `-9`
- [ ] **C.** `0`
- [ ] **D.** `9`

第 10 题 如下 C++ 代码执行输出是（　　）。

```cpp
int i;
for (i = 1; i < 5; i++) {
    if (i % 3 == 0) {
        printf("%d+", i);
        continue;
    } else {
        printf("%d-", i);
        break;
    }
}
printf("%dEND*", i);
```

- [ ] **A.** `1-1`
- [ ] **B.** `1 - 1 END*`
- [ ] **C.** `1+1-2-2END*`
- [ ] **D.** `1-1END*`

第 11 题 如下 C++ 代码执行时，下列说法错误的是（　　）。

```cpp
int N, tot;
cin >> N;
tot = 0;

while (N > 0) {
    cout << N << "#";
    tot += 1;
    N -= 1;
}

cout << tot << "-END";
```

- [ ] **A.** 如果输入 `3`，输出是：`3#2#1#3-END`
- [ ] **B.** 如果输入 `-10`，输出是：`0-END`
- [ ] **C.** 如果输入 `0`，输出是：`0-END`
- [ ] **D.** 如果输入 `1`，输出是：`1#0-END`

第 12 题 下列 C++ 代码执行时如输入 `35`，输出如下图所示。横线处应填入的代码是（　　）。

```
35 34 33 32 31 30 29 28 27 26
25 24 23 22 21 20 19 18 17 16
15 14 13 12 11 10  9  8  7  6
 5  4  3  2  1
```

```
int N, tot, i;
cin >> N;

tot = 0;
for (i = 0; i < N; i++) {
    printf(__________);
    tot = tot + 1;
    if (tot % 10 == 0)
        printf("\n");
}
```

- [ ] **A.** `"%3d", N`
- [ ] **B.** `"%3d", N - i`
- [ ] **C.** `N - i`
- [ ] **D.** `N`

第 13 题 下面 C++ 代码执行后输出是（　　）。

```
int k = 0;
while (k < 8) {
    k += 2;
    continue;
    if (k > 5)
        break;
    cout << k << "#";
}
if (k >= 8)
    cout << "FINISH";
```

- [ ] **A.** `FINISH`
- [ ] **B.** `2#4#6#FINISH`
- [ ] **C.** `2#4#FINISH`
- [ ] **D.** `2#4#6#8#FINISH`

第 14 题 在下列 C++ 代码中，如果 `N` 和 `M` 均输入大于 `0` 的整数，则可以取得正整数 `N` 的从右到左第 `M` 位置的数。个位数为第 `1` 位置。则横线处应填入（　　）。

```
int N, M, M10;

cout << "请输入正整数:";
cin >> N;
cout << "从右到左第几位? ";
cin >> M;

M10 = 1;
for (; M > 1; M--)
    M10 *= 10;
cout << _____________; // L1
```

- [ ] **A.** `N / M10 % 10`
- [ ] **B.** `N / M10 / 10`
- [ ] **C.** `N % M10 % 10`
- [ ] **D.** `N % M10 / 10`

第 15 题 求解鸡兔同笼。在如下 C++ 代码中，`tou` 代表鸡和兔的头总数，`jiao` 代表鸡和兔的脚总数。先后两个输入均为正整数。关于该程序下面说法准确的是（　　）

```
int tou, jiao, tu_tou, ji_tou;

cout << "请输入鸡和兔的头总数：";
cin >> tou;
cout << "请输入鸡和兔的脚总数：";
cin >> jiao;

for (tu_tou = 0; tu_tou < tou + 1; tu_tou++) { // L1
    ji_tou = tou - tu_tou;
    if (tu_tou * 4 + ji_tou * 2 == jiao) {
        printf("兔的数量=%d,鸡的数量=%d\n", tu_tou, ji_tou);
        break;
    }
}
if (tu_tou == tou) // L2
    printf("无解\n");
```

- ☐ **A.** 能求解鸡兔同笼。
- ☐ **B.** `ji_tou` 没有初始化，在 `L1` 行代码前增加 `ji_tou = 0;` 即可求解鸡兔同笼。
- ☐ **C.** 将 `L1` 行的 `tu_tou < tou + 1` 改为 `tu_tou <tou` 即可求解鸡兔同笼。
- ☐ **D.** 将 `L2` 行的 `tu_tou == tou` 改为 `tu_tou == tou + 1` 即可求解鸡兔同笼。

2　判断题（每题 2 分，共 20 分）

第 1 题 C++ 程序执行 `cin` 或 `scanf` 语句时通常是使用键盘等输入设备来输入某个变量的值。

第 2 题 如果 `n` 为 `3` 位数的正整数，则 C++ 表达式 `n % 100` 将能得到最高位。如 `n` 为 `123`，则该表达式的值为 `1`。

第 3 题 下面 C++ 代码执行后将输出 `20`。

```
int i;
for (i = 0; i < 20; i += 3)
    if (i % 3 != 0)
        continue;
cout << i;
```

第 4 题 下列 C++ 代码执行时如果先后输入 `10` 和 `10`，则将输出 `10小于10`。

```
int N, M;

cin >> N >> M;

if (N > M)
    printf("%d大于%d", N, M);
else
    printf("%d小于%d", N, M);
```

第 5 题 如下 C++ 代码如果输入大于 `0` 的整数，则可以求 `1` 到 `N` 之和。

```cpp
int N, tot;

cout << "请输入正整数: ";
cin >> N;
tot = 0;

while (N > 0) {
    tot += N;
    N += -1;
}
cout << tot;
```

第 6 题 在 C++ 中，表达式 `2 % 2` 与 `(-2) % 2` 的值相同。

第 7 题 试图编译并执行下面 C++ 代码，会输出 `2` 。

```cpp
float z = 12.0;
z = z / 4;
z--;
cout << z++;
```

第 8 题 在 C++ 代码中，如果两个变量 `__` 和 `____` 都只包含下划线(`_`)，但数目不同，则可能导致编译器无法识别，而报告变量重复定义。

第 9 题 C++ 的变量 `X` 是 `double` 型，则表达式 `X = 27 % 12` 的结果不可预测。

第 10 题 下面 C++ 代码执行后输出的是 `3` 。

```cpp
int ans = 0;
for (int idx = 1; idx <= 7; idx += 2) {
    if (idx == 5)
        continue;
    if (idx * idx > 30)
        break;
    ans += idx;
}
cout << ans;
```

3 编程题（每题 25 分，共 50 分）

3.1 编程题 1

- **试题名称**：新龟兔赛跑
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.1.1 题目描述

自从上次龟兔赛跑，兔子因为轻敌而惨败后，一直心有不甘，因此有了新一届龟兔赛跑。

乌龟和上次一样，将会以恒定的速度往终点爬行，题目将给出乌龟从起点到终点所需时间；
兔子并没有吸取教训，只是打算在终点前少睡一会儿，题目将给出兔子的速度，和兔子打算在终点前睡觉的时间；

起点和终点之间的距离固定为 1000.0，请帮忙计算乌龟和兔子谁会获胜？

3.1.2 输入格式

输入包含 3 行，每行一个浮点数：

- 第 1 行的浮点数表示乌龟从起点到终点所需时间；
- 第 2 行的浮点数表示兔子的速度；
- 第 3 行的浮点数表示兔子打算在终点前睡觉的时间。

3.1.3 输出格式

输出包含 2 行：

- 第 1 行，如果乌龟获胜，输出 `turtle`；如果兔子获胜，输出 `rabbit`；如果平局，输出 `tie`；
- 第 2 行，输出兔子到达终点所需的总时间（含兔子睡觉的时间），保留两位小数。

3.1.4 样例

3.1.5 输入样例 1

```
20.0
100.0
11.0
```

3.1.6 输出样例 1

```
turtle
21.00
```

3.1.7 样例解释 1

兔子跑完全程所需时间是 $1000.0 \div 100.0 = 10.0$，而兔子要睡觉 11.0 单位时间，所以兔子一共需要 21.0 单位时间才能到达终点，这大于乌龟所需时间。

3.1.8 输入样例 2

```
20.0
100.0
9.0
```

3.1.9 输出样例 2

```
rabbit
19.00
```

3.1.10 样例解释 2

与样例 1 类似，兔子跑完全程所需时间是 $1000.0 \div 100.0 = 10.0$，但兔子仅睡觉 9.0 单位时间，所以兔子一共需要 19.0 单位时间就能到达终点，这小于乌龟所需时间。

3.1.11 数据范围

所有输入均有且仅有一位小数，且是不超过 1000.0 的正数。

本题默认给出的数值均为龟兔世界中的标准单位，不需要进行任何单位换算。

路程 = 速度 × 时间

3.2 编程题 2

- **试题名称**：棋盘上的奖赏
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.2.1 题目描述

相传古印度舍罕王为了奖赏宰相西萨发明国际象棋，西萨提的任何要求舍罕王都愿意满足。西萨只求在棋盘第 1 格放 1 粒麦子，第 2 格放 2 粒麦子，之后每一格都放前一格两倍的麦子，以此类推直到第 64 格。国王觉得这奖赏微不足道，立刻命人取粮。

然而麦子的数量飞速增长：到第 20 格已需上百万粒，到第 40 格就耗尽了全国库存，而后面仍有 24 格。最终国王发现，即便倾尽天下粮食也填不满棋盘。

聪明的你知道前 n 格总共需要多少粒麦子吗?

3.2.2 输入格式

一行，一个整数 n。

3.2.3 输出格式

一行，一个整数，表示前 n 格总共需要的麦子粒数。

3.2.4 样例

3.2.5 输入样例 1

```
3
```

3.2.6 输出样例 1

```
7
```

3.2.7 输入样例 2

```
10
```

3.2.8 输出样例 2

```
1023
```

3.2.9 数据范围

对于 40% 的测试点，保证 $1 \leq n \leq 10$。

对于所有测试点，保证 $1 \leq n \leq 30$。