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

C++ 二级

2025 年 12 月

1 单选题（每题 2 分，共 30 分）

第 1 题 近日，空中客车公司表示，约6000架空客A320系列飞机需要紧急更换一种易受太阳辐射影响的飞行控制软件。空客表示，在对一起飞行事故分析后的结果显示，强烈的太阳辐射可能会损坏飞行控制系统所需的关键数据，导致判断失误，进而引发飞行异常。那这里的飞行控制系统执行判断的部件最可能是下面的( )。

- A. 辐射传感器
- B. 处理器
- C. 内存单元
- D. 输出设备

第 2 题 小明最近为了备考GESP，开始看B站上关于网络知识的视频。其中提到计算机网络系统有不同的划分标准，那他平时上学所在的教学楼内的网络是一个（ ）。

- A. PAN
- B. LAN
- C. MAN
- D. WAN

第 3 题 下面有关C++变量的说法，正确的是( )。

- A. 不可以用 `for` 作为变量名，因为 `for` 是C++的关键字（保留字）。
- B. `_tnt` 不可以是变量名，因为变量名的第一个字符必须是英文字母。
- C. `_tnt_` 不可以是变量名，因为最后一个字符容易与减号混淆。
- D. 可以用 `printf` 作为变量名，因为 `printf` 是关键字，但这不是好习惯，因为 `printf` 有约定的功能与含义。

第 4 题 一个小数是0.123123123……无限循环，其小数点后1位是1，后2位是2，依此类推，求第N位的值。横线处应填入的代码是( )。

```
int N;
cin >> N;
remainder = _____________;

if (remainder == 0)
    cout << 1;
else if (remainder == 1)
    cout << 2;
else
    cout << 3;
```

- [ ] **A.** `N % 3`
- [ ] **B.** `(N - 1) % 3`
- [ ] **C.** `N / 3`
- [ ] **D.** `(N - 1) / 3`

第 5 题 某同学执行C++代码时 `printf("%g\n", (3 + 3.1415926535));` 输出 `6.14159`，其原因最可能是( )。

- [ ] **A.** C++中整数和浮点数相加时，整数会被转换为浮点数，而某些十进制小数无法精确表示为二进制小数，从而产生某些舍入误差。
- [ ] **B.** C++的 `printf` 函数在输出浮点数时根据格式有默认小数点位数，因此输出了较少的位数。
- [ ] **C.** `3.1415926535` 是一个无限循环小数，在计算机中无法精确表示。
- [ ] **D.** 由于计算机CPU的运算错误导致。

第 6 题 某单位工号的编码规则：编码总长5位，均为数字，前4位依次整除以3其值累加之和除以10的余数为第5位数字。如某工号为76587，前4位分别整除以3后，商分别为2、2、1、2，其累加之和为7，除以10的余数为7，故第5位为7。下面代码依次输入前4位后，两个横线处分别应填的是(　　)。

```
int rst = 0, N;
for (int i = 0; i < 4; i++){
    cin >> N;
    rst += ___________; // L1
}
cout << ___________; // L2
```

- [ ] **A.**

```
N % 3
rst / 10
```

- [ ] **B.**

```
N % 3
rst % 10
```

- [ ] **C.**

```
N / 3
rst / 10
```

- [ ] **D.**

```
N / 3
rst % 10
```

第 7 题 下面的C++代码执行后的输出是( )。

```
for (int i = -2; i < 2; i++)
    if (i % 2)
        printf("%d#",i);
```

- [ ] **A.** `-1#1#`
- [ ] **B.** `-1#0#1#`
- [ ] **C.** `-2#-1#1#`
- [ ] **D.** `-2#-1#1#2#`

第 8 题 下面的C++代码执行后其输出是( )。

```
int cnt = 0, N;
for (int i =1; i < 10; i += 2)
    for (int j =0; j < i; j++)
       cnt += 1;
cout << cnt;
```

- [ ] **A.** `100`
- [ ] **B.** `55`
- [ ] **C.** `45`
- [ ] **D.** `25`

第 9 题 下面C++代码执行后其输出是( )。

```
int i,j;

for (i = 1; i < 12; i++){
    if (i % 2 == 0)
        continue;

    for (j = 0; j < i; j++)
        if (i * j % 2 == 0)
            break;
    if(j >= i)
        cout << i * j << " ";
}
if(i >= 12)
    cout << (i * j);
```

- [ ] **A.** `0 0`
- [ ] **B.** `11`
- [ ] **C.** `0`
- [ ] **D.** `0 11`

第 10 题 与下面C++输出效果不一致的代码是( )。

```
int i;
for (i = 0; i < 10; i++)
    cout << i;
```

A.

```
int i = 0;
while (i < 10){
    cout << i;
    i += 1;
}
```

B.

```
int i = 0;
while (i < 10){
    i += 1;
    cout << i;
}
```

C.

```
int i = 0;
while (true){
    cout << i;
    i += 1;
    if (i >= 10)
        break;
}
```

D.

```
int i = 0;
while (true){
    if (i >= 10)
        break;
    cout << i;
    i += 1;
}
```

第 11 题 下面C++代码执行后输出是（　）。

```
int num = 0;
while (num <= 5){
    num += 1;
    if (num % 3)
        continue;
    printf("%d#",num);
}
if(num > 5)
    printf("%d", num);
```

A. `3#6#`

B. `3#6#6`

C. `1#2#3#4#5#6#`

D. `1#2#3#4#5#6#6`

第 12 题 下面C++代码执行后，其输出是（　）。

```
int cnt = 0;
for (int i = 0; i <5; i++)
    for (int j =i; j < 4; j++)
        cnt += 1;
cout << cnt;
```

- [ ] **A.** `9`
- [ ] **B.** `10`
- [ ] **C.** `14`
- [ ] **D.** `20`

第 13 题 漂亮数的定义是：如果N能被M整除，或者某位是M，或者N的每位数之和能被M整除，则说N是M的漂亮数。如果三个条件都满足，则是完整漂亮数。123是3的完整漂亮数，因为123能被3整除，也含有3，其每位数之和为6也能被3整除。下面的代码用于判断N是否为M的完整漂亮数并输出。相关说法正确的是（　）。

```
int N, M, Flag, Sum, num;
cout << "请输入N，不等于0的正整数：";
cin >> N;
cout << "请输入M：M必须大于1小于9：";
cin >> M;

Sum = 0; // 记录各位数之和
Flag = 0; // 假设记录N不含有M

while (N != 0){
    num = N % 10;

    Sum += num;
    if (num == M)
        Flag = 1;

    N /= 10;
}
if ((N % M == 0) && (Flag == 1) && (Sum % M == 0)) // L1
    printf("%d 是 %d 的完整漂亮数", N, M);
else
    printf("%d 不是 %d 的完整漂亮数", N, M);
```

- [ ] **A.** 代码能完成题目设定目标。
- [ ] **B.** 在 `while` 循环中N最终将变成0，因此L1行代码中 `N % M` 将总是满足条件，可以在 `while` 前增加一行代码 `int old_num = N;` ，并将L1开始这4行代码中的 `N` 都改为 `old_num` 就可以做出正确的判定。
- [ ] **C.** `while` 循环中 `if` 判断语句可以增加 `else` 子句，其内容为 `Flag = 0` 。
- [ ] **D.** 如果先后输入 `0` 和 `3` ，则肯定会输出 `0是3的完整漂亮数` 。

第 14 题 阅读下面的C++代码。执行后如输入 `5` ，其输出的字符图形是（　）。

```
int n,i,j,k;
cin >> n;
for (i = 0; i < n; i++){
    for (j = 0; j < n-i-1; j++)
        cout << " ";
    for (k = 0; k < 2*i+1; k++)
        cout << "*";
    cout << endl;
}
```

- [ ] **A.**

```
*****
 ****
  ***
   **
    *
```

☐ **B.**

```
    *
   ***
  *****
 *******
*********
```

☐ **C.**

```
*
**
***
****
*****
```

☐ **D.**

```
    *
   **
  ***
 ****
*****
```

第 15 题 某学校举办“十佳歌手大奖赛”，经过选拔最终参赛选手有25人，评委10人，最终计分规则去掉一个最高分去掉一个最低分作为该参赛选手的最终得分，并输出该得分。如果单个评委可给满分为10分，则相关说法正确的是（ ）。

```
float total_score, max_score, min_score, now_score;
for (int i = 0; i < 25; i++){
    max_score = 0;  // 记录最高分
    min_score = 10; // 记录最低分
    total_score = 0; // 记录总分

    for (int j = 0; j < 10; j++){
        cin >> now_score; // 录入评委打分

        max_score = max(max_score, now_score); // L1
        min_score = min(min_score, now_score); // L2

        total_score += now_score;
    }

    cout << (total_score - max_score - min_score);
}
```

☐ **A.** 程序总体逻辑错误。因为要去掉最高分和最低分，需要排序，而程序没有相关代码。

☐ **B.** 内层循环和外层循环之间的三行代码也就是 `max_score = 0` 开始的三行代码应该移动到外层循环外。

☐ **C.** L1和L2标记的两行代码可以分别改为简单的 `if` 语句或 `? :` 语句。

☐ **D.** `total_score += now_score` 不可以更改为 `total_score = total_score + now_score` 。

2 判断题（每题 2 分，共 20 分）

第 1 题 鸿蒙是华为公司开发的一款操作系统，那么它能够将正确的源程序翻译成目标程序，并运行。（ ）

第 2 题 C++表达式 `5 < 10 && 20` 对应的逻辑值为 `true` 。( )

第 3 题 C++表达式 `10 / 0.333333 == 10 / (1 / 3)` 的值为 `true`。（ ）

第 4 题 下面C++代码中N是整数，执行时无论输入负整数、0或正整数，其输出都将是 `0`。（ ）

```
cin >> N;
while (N)
    N /= 10;
cout << N;
```

第 5 题 下面的C++代码执行后，其输出是 `4 0`。（ ）

```
int a,b;
a = 4;
b = a == 5;
cout << a << ' ' << b;
```

第 6 题 C++代码中对表达式 `('Z' - 'A') < ('z' - 'A')` 的结果输出为 `0`。（ ）

第 7 题 下面的C++代码可以用于判断正整数N的位数（即几位数，如123是3位数，12为2位数）。（ ）

```
int N, N10, i;
cin >> N;
N10 = 10, i = 1;
while (1) {
    if (N % N10 == N) {
        printf("%d 是 %d 位数", N, i);
        break;
    }
    i++, N10 *= 10;
}
```

第 8 题 计算交叉加减的结果，形如 `1-2+3-4+5-……` 。下面C++代码中的变量都是整型，则将 `Flag = -Flag` 改为 `Flag -= Flag` 效果相同。（ ）

```
cin >> N;
Flag = -1;
tnt = 0;
for (i = 1; i < N + 1; i++) {
    Flag = -Flag;
    tnt += Flag * i;
}
cout << tnt;
```

第 9 题 下列C++代码段执行后将输出 `55`。（ ）

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

第 10 题 下面C++代码执行后输出如下，因为代码 `printf("\n")` 没有任何可读内容，删除不影响输出效果。（ ）

```
 1  2  3  4  5  6  7  8  9
 2  4  6  8 10 12 14 16 18
 3  6  9 12 15 18 21 24 27
 4  8 12 16 20 24 28 32 36
 5 10 15 20 25 30 35 40 45
 6 12 18 24 30 36 42 48 54
 7 14 21 28 35 42 49 56 63
 8 16 24 32 40 48 56 64 72
 9 18 27 36 45 54 63 72 81
```

```
for (int i = 1; i < 10; i++) {
    for(int j = 1; j < 10; j++)
        printf("%3d", i * j);
    printf("\n");
}
```

3 编程题（每题 25 分，共 50 分）

3.1 编程题 1

- **试题名称**：环保能量球
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.1.1 题目描述

小杨最近在玩一个环保主题的游戏。在游戏中，小杨每行走 1 公里就可以获得 1 点“环保能量”。

为了激励玩家，游戏设置了“里程奖励”：小杨每行走 x 公里，游戏就会额外奖励 1 点能量。

现在已知小杨总共行走了 n 公里，请你帮他计算，他一共能获得多少点环保能量?

3.1.2 输入格式

第一行包含一个正整数 t，代表测试数据组数。

对于每组测试数据：

- 第一行包含一个正整数 n，代表行走的公里数。
- 第二行包含一个正整数 x，代表奖励触发的间隔。

3.1.3 输出格式

对于每组测试数据，输出一个整数，代表小杨获得的环保能量总数。

3.1.4 样例

3.1.4.1 输入样例

```
3
5
2
10
3
2
5
```

3.1.4.2 输出样例

```
7
13
2
```

3.1.4.3 样例解释

- 对于第 1 组数据，$n = 5, x = 2$：小杨行走获得 5 点能量。此外，他在第 2 公里和第 4 公里时各获得 1 点额外奖励，总共 $5 + 2 = 7$ 点。
- 对于第 2 组数据，$n = 10, x = 3$：行走获得 10 点。他在第 3、6、9 公里时各获得 1 点额外奖励，总共 $10 + 3 = 13$ 点。
- 对于第 3 组数据，$n = 2, x = 5$：行走获得 2 点。由于行走路程不足 5 公里，没有额外奖励，总共 2 点。

3.1.5 数据范围

对于全部数据，保证：$1 \leq t \leq 100$，$1 \leq n, x \leq 1000$。

3.2 编程题 2

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

3.2.1 题目描述

小杨在探险时发现了一张神奇的矩形地图，地图有 H 行和 W 列。每个格子的坐标是 (r, c)，其中 r 表示行号从 1 到 H，c 表示列号 1 到 W。

小杨听说地图中隐藏着一些“黄金格”，这些格子满足一个神秘的数学挑战：当格子坐标 (r, c) 代入特定的不等式关系成立时，该格子就是黄金格。具体来说，黄金格的条件是：$\sqrt{r^2 + c^2} \leq x + r - c$ 。

例如，如果参数 $x = 5$ ，那么格子 $(4, 3)$ 就是黄金格。因为左边坐标平方和的平方根 $\sqrt{4^2 + 3^2}$ 算出来是 5，而右边 $5 + 4 - 3$ 算出来是 6，5 小于等于 6，符合条件。

3.2.2 输入格式

三行，每行一个正整数，分别表示 H, W, x。含义如题面所示。

3.2.3 输出格式

一行一个整数，代表黄金格数量。

3.2.4 样例

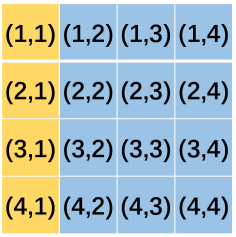
3.2.4.1 输入样例 1

```
4
4
2
```

3.2.4.2 输出样例 1

```
4
```

3.2.4.3 样例解释



图中标注为黄色的四个格子是黄金格，坐标分别为 $(1,1)$，$(2,1)$，$(3,1)$，$(4,1)$。

3.2.5 数据范围

对于所有测试点，保证给出的正整数不超过 1000。