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

C++ 三级

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	A	B	C	B	A	C	C	A	B	D	D	B	C	B	B

第 1 题 关于计算机的数据编码，下面说法正确的是（ ）。

- A. 机器数 `1000 1101B` 的真值可能是 `-13`（如果它是原码）或 `141`（如果它是无符号数）等数，这取决于我们如何解释它。
- B. 计算机中，所有数据最终都以二进制、八进制、十六进制的形式存储和运算。
- C. 字节（`Byte`）是计算机中最小的数据单位。位（`bit`）是计算机中最小的存储单位。
- D. 计算机中，`1k` 字节是 `1000` 字节的意思。

第 2 题 计算机厂商为了计算方便，一般采用 `1000` 进制。如果我们买的厂商标注的是 `1 TB` 的硬盘，它实际的存储容量是（ ）。

- A. $1000 \times 1000 \times 1000 \times 1000 \div 1024 \div 1024 \div 1024\text{b} = 931\text{Gb}$
- B. $1000 \times 1000 \times 1000 \times 1000 \div 1024 \div 1024 \div 1024\text{B} = 931\text{GB}$
- C. $1024 \times 1024 \times 1024 \times 1024 \div 1000 \div 1000 \div 1024\text{B} = 1049\text{GB}$
- D. $1000 \times 1024 \times 1024 \times 1024 \div 1024 \div 1024 \div 1024\text{b} = 977\text{Gb}$

第 3 题 低 4 位、高 4 位压缩技术，适用于数据仅使用字节的一部分（如仅用低 4 位）的场景。字节结构：一个字节为 8 位，分为高 4 位（高位）和低 4 位（低位）。当数据是十六进制数（$0 \sim 15$，即 `0x0` 到 `0xF`），每个值仅需 4 位表示，高 4 位全为 `0`。将两个相邻的 4 位值合并为一个字节。四个数据 `0x1`、`0x2`、`0x3`、`0x4` 采用上述压缩技术压缩以后是（ ）。

- A. `12D`、`34D`
- B. `12Q`、`34Q`
- C. `12H`、`34H`
- D. `00010011B`、`00110101B`

第 4 题 关于计算机编码中反码和补码，下面说法错误的是（ ）。

- A. 负数的补码，一个快速方法是从右往左扫描正数的二进制形式，遇到第一个 `1` 之后，左边的所有位都取反。
- B. 对于一个 n 位的二进制数：最大表示范围：$[-(2^{n-1})-1, +(2^{n-1}-1)]$。
- C. 反码减法可以统一为加法。符号位可以直接参与运算。
- D. 反码表示中，0 的表示不唯一：`0000 0000B` 和 `1111 1111B`。

第 5 题 一种加密方式是字符数组与密钥 `KEY`、运算方式分开传输，比如字符数组 `char text[4] = {'G', 'E', 'S', 'P'};` 由一种传输方式发送，密钥 `KEY = 2026` 通过另一种发送方式发送，运算方式 `char function[4] = {'|', '-', '^', '+'};` 又是另一种发送方式发送。三种数据都到达目的地以后，分别进行例如 `'G' | 6`、`'E' - 2`、`'S' ^ 0`、`'P' + 2` 等计算，来得到相应的真实内容，上述 `GESP` 通过这种加密方式，加密以后最终的内容是（ ）。

- A. `GCSR`
- B. `RSCG`
- C. `GCSA`
- D. `BCSR`

第 6 题 关于位运算，下列说法错误的是（ ）。

- A. 找唯一数：数组中唯一出现一次的数，其余出现两次，全部异或结果即为该数。例如：数组 `[5, 7, 9, 7, 5]`（唯一数是 `9`）。
- B. 交换两个数：`a ^= b; b ^= a; a ^= b;`（无需临时变量）。
- C. 将二进制位整体左移 n 位，高位溢出舍弃，低位补 `0`；等价于 `num` 乘以 2^n。
- D. 对每一个二进制位取反，包括符号位，简单运算规则是 `~n = -n - 1`。

第 7 题 关于字符串和字符数组，下列说法正确的是（ ）。

- A.

```
char str[] = "GESP";
int len1 = sizeof(str);
int len2 = strlen(str);
```

上面程序能够正确执行，`len1` 与 `len2` 相等。

- B.

```
char str1[4] = "GESP";
char str2[4] = {'G', 'E', 'S', 'P'};
```

这段程序将能够正确执行。

- C.

```
char str2[4] = {'G', 'E', 'S', 'P'};
strcpy(str2, "HELLO,GESP");
cout << str2 << endl;
```

这段程序即使能够运行，但是存在覆盖数组以外的内存空间的行为，可能会引起严重错误。

- D.

```
char dest[4] = {'G', 'E', 'S', 'P'};
char src[] = "HELLO";
strcat(dest, src);
cout << dest << endl;
```

这段程序能够正确执行，不存在数组越界行为。

第 8 题 计算机中的 `2 KB` 等于多少 `bit`（ ）。

- A. `16384`
- B. `20000`

- [ ] **C.** `2000`
- [ ] **D.** `2048`

第 9 题 在 C++ 中，对于 32 位有符号整数 `int` 类型数据 `n`，关于按位取反运算符 `~`，下列说法正确的是（ ）。

- [ ] **A.** `~6` 的结果是 `5`。
- [ ] **B.** 按位取反满足公式 `~n = -n - 1`。
- [ ] **C.** `~0` 的结果是 `1`。
- [ ] **D.** `~(-2)` 的结果是 `-1`。

第 10 题 关于计算机中的二进制编码表示，下列说法错误的是（ ）。

- [ ] **A.** 原码是最直观的一种有符号数表示方法。最高位（最左边的位）为符号位：`0` 表示正数，`1` 表示负数，其余位为数值位（真值的绝对值）。
- [ ] **B.** 补码完美解决了原码和反码的缺陷，是现代计算机中表示有符号整数的标准方式。正数的补码与其原码、反码相同；负数的补码是将其对应正数的原码按位取反（得到反码），然后加 `1`。
- [ ] **C.** 计算补码的一个更快的技巧：从右往左扫描正数的二进制形式，遇到第一个 `1` 之后，左边的所有位都取反。
- [ ] **D.** 对于一个 n 位的二进制数，补码最大表示范围为 $[-2^{n-1}, +2^{n-1}]$。

第 11 题 下面选项中提到的变量都是正整数，关于位运算，下面说法错误的是（ ）。

- [ ] **A.** `num & 1`，结果为 `1` 则奇数，`0` 则偶数（仅看最低位）。
- [ ] **B.** `num & 0xFF` 保留低 8 位。
- [ ] **C.** `num & b` 的结果一定小于等于 `num`。
- [ ] **D.** 若 `num` 左移导致高位溢出（如超过整型范围），结果符合乘法规律。

第 12 题 `a=7, b=3, c=14, d=15, e=8`，对于运算表达式 `!a << b & c ^ d | e` 的结果是（ ）。

- [ ] **A.** `0`
- [ ] **B.** `15`
- [ ] **C.** `7`
- [ ] **D.** `14`

第 13 题 关于 `string` 的成员函数，下面说法错误的是（ ）。

- [ ] **A.** `size()`：返回字符串长度（字符个数，不含 `'\0'`）。
- [ ] **B.** `length()` 与 `size()` 功能完全一致，返回字符串长度。
- [ ] **C.** `empty()`：判断字符串是否为空（非空返回 `true`，空返回 `false`）。
- [ ] **D.** `s.append(s2, 0, 3);` 从 `s2` 下标 `0` 开始，截取 `3` 个字符。

第 14 题 以下数组定义，符合 C++ 语法的是（ ）。

- [ ] **A.** `int [10] a;`
- [ ] **B.** `int b['&'];`
- [ ] **C.** `int c[*];`
- [ ] **D.** `double d[10.0];`

第 15 题 现在有一个数，请你分别判断它们是否可能是二进制、八进制、十进制、十六进制。例如，`6AFF` 就只可能是十六进制，而 `1011` 则是四种进制皆有可能。输入 `N`（保证 $1 \leq N \leq 1000$），表示有 `N` 个数让你进行判断，接下来输入 `N` 个字符串（保证所有字符串长度不超过 `10`），判断可能是四个进制当中的哪个进制数。输出 `N` 行，每行 `4` 个数，用空格隔开，分别表示给定的字符串是否可能表示一个二进制数、八进制数、十进制数、十六进制数。使用 `1` 表示可能，使用 `0` 表示不可能。下面程序横线处可以满足这个要求的是（ ）。

```
#include <iostream>
using namespace std;
int main() {
    int n = 0;
    cin >> n;
    for (int i = 0; i < n; i++) {
        char str[11];
        cin >> str;
        char max = '0';
        for (int i = 0; str[i] != '\0'; i++)
            if (str[i] > max)
                max = str[i];
        ----------------------------------------
    }
    return 0;
}
```

- A. `cout << (max >= '1') << " " << (max >= '7') << " " << (max >= '9') << " " << (max >= 'F') << endl;`
- B. `cout << (max <= '1') << " " << (max <= '7') << " " << (max <= '9') << " " << (max <= 'F') << endl;`
- C. `cout << (max = '1') << " " << (max = '7') << " " << (max = '9') << " " << (max = 'F') << endl;`
- D. `cout << (max < '1') << " " << (max < '7') << " " << (max < '9') << " " << (max < 'F') << endl;`

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 反码运算时，产生的进位需要循环进位，即最高位产生的进位要加回到结果的最低位。

第 2 题 `-11` 的补码的一种计算方式是从右往左扫描正数的二进制形式，遇到第一个 `1` 之后，左边的所有位都取反。

第 3 题 一个 8 位的二进制数补码，最大的表示范围是从 -127 到 $+127$。

第 4 题 判断某个数是否是质数，枚举范围可适当缩小（遍历到 $\sqrt{i}$ 而非 i），提升效率。

第 5 题 如果 `a` 为 `int` 类型的变量，且表达式 `((a & 1) == 1)` 的值为 `true`，则说明 `a` 是奇数。

第 6 题 十六进制数 `CCF` 对应的二进制数、八进制数、十进制数分别是：`110011001111`、`6317`、`3269`。

第 7 题 下列程序如果能够正确执行，那么输出的结果是 `GESP`。

```cpp
int main() {
    string name = "GESP";
    cout << name[false] << name[true] << name[1 << 1] << name[7 >> 1] << endl;
}
```

第 8 题 某个初学 C++ 的学生，在对照参考程序写了一个程序以后，信心满满地进行编译，他敲进编译器的全部代码如下，这个程序能够正常编译运行。

```cpp
1 #include<bits/stdc++.h>
2 using namespace std;
3 int main() {
4     cout << "Hello world!" << endl;
5     return 0;
6 }
```

第 9 题 代码逐行复刻手工运算步骤，中间不能增加任何判断逻辑，否则就不属于模拟算法。

第 10 题 判断一个 `int` 型数字 `s[i]` 是不是在 `0` 和 `9` 之间（可以是 `0` 和 `9`），判断条件可以写为 `if (s[i] >= '0' && s[i] <= '9')`。

3 编程题（每题 25 分，共 50 分）

3.1 编程题 1

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

3.1.1 题目描述

小杨同学有一串数字，想把它们变成另一串数字，这个过程叫做**加密**。

他有一本**密码本**，密码本告诉你：
每个数字应该变成哪个数字。

数字一共有 10 个：
`0`、`1`、`2`、`3`、`4`、`5`、`6`、`7`、`8`、`9`

密码本会依次告诉你：
`0` 要变成什么
`1` 要变成什么
`2` 要变成什么
……
`9` 要变成什么

请你按照密码本，把原来的每个数字都换成新的数字，然后输出。

3.1.2 输入格式

输入共有 3 行。
第一行：一个整数，表示有多少个数字需要加密；
第二行：这些需要加密的数字；
第三行：密码本，一共 10 个数字。

这 10 个数字的意思是：
第 1 个数字：表示 `0` 加密后变成什么；
第 2 个数字：表示 `1` 加密后变成什么；
第 3 个数字：表示 `2` 加密后变成什么；
……
第 10 个数字：表示 `9` 加密后变成什么。

3.1.3 输出格式

输出加密后的数字。

也就是：把输入第二行里的每个数字，都按照输入第三行的密码本换掉后输出。

3.1.4 样例

3.1.5 输入样例 1

```
7
0 2 0 3 4 1 9
9 0 1 2 3 4 5 6 7 8
```

3.1.6 输出样例 1

```
9 1 9 2 3 0 8
```

3.1.7 样例解释

第二行要加密的数字是：
`0 2 0 3 4 1 9`

第三行密码本是：
`9 0 1 2 3 4 5 6 7 8`

它的意思是：
`0` 变成 `9`
`1` 变成 `0`
`2` 变成 `1`
`3` 变成 `2`
`4` 变成 `3`
`5` 变成 `4`
`6` 变成 `5`
`7` 变成 `6`
`8` 变成 `7`
`9` 变成 `8`

所以：
`0` 变成 `9`
`2` 变成 `1`
`0` 变成 `9`
`3` 变成 `2`

`4` 变成 `3`
`1` 变成 `0`
`9` 变成 `8`

最后得到：
`9 1 9 2 3 0 8`

3.1.8 数据范围

需要加密的数字个数不超过 20000 个，且均为 0 到 9；密码本中的数字不重复，且均为 0 到 9。

3.1.9 参考程序

```cpp
#include <iostream>
using namespace std;

int a[20010];
int keys[11];

int main() {
    int n;
    cin >> n;
    for (int i = 0; i < n; ++i)
        cin >> a[i];
    for (int i = 0; i < 10; ++i)
        cin >> keys[i];
    for (int i = 0; i < n; ++i)
        cout << keys[a[i]] << ' ';
    cout << endl;
    return 0;
}
```

3.2 编程题 2

- **试题名称**：字符转换
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.2.1 题目描述

小杨同学有一串字符，里面可能有：

- 大写字母，比如 `A`、`B`、`C`
- 小写字母，比如 `a`、`b`、`c`
- 数字，比如 `0`、`1`、`2`

现在小杨同学想把这串字符变一变，规则如下：

1. 如果是大写字母，就变成对应的小写字母；
2. 如果是小写字母，就变成对应的大写字母；
3. 如果是数字，就变成 `*`。

请你按照这个规则，帮小杨把整串字符转换好。

3.2.2 输入格式

输入一共有 2 行。
第一行：一个整数，表示这串字符一共有多少个字符。
第二行：一串连续的字符，中间没有空格。

3.2.3 输出格式

输出转换后的字符。

注意：
输出时字符之间不要加空格。

3.2.4 样例

3.2.5 输入样例 1

```
5
aBc98
```

3.2.6 输出样例 1

```
AbC**
```

3.2.7 样例解释

原来的字符是：

```
aBc98
```

从左到右一个一个看：

1. `a` 是小写字母，所以变成 `A`；
2. `B` 是大写字母，所以变成 `b`；
3. `c` 是小写字母，所以变成 `C`；
4. `9` 是数字，所以变成 `*`；
5. `8` 是数字，所以变成 `*`。

所以最后输出：

```
AbC**
```

3.2.8 数据范围

字符个数不会超过 1000 个。

每个字符只会是大写字母、小写字母或数字。

3.2.9 参考程序

```cpp
#include <iostream>
using namespace std;

char s[1010];

int main() {
    int n;
    cin >> n;
    cin >> s;
    for (int i = 0; i < n; ++i) {
        char c = s[i];
        if ('a' <= c && c <= 'z')
            cout << (char)(c - ('a' - 'A'));
        else if ('A' <= c && c <= 'Z')
            cout << (char)(c + ('a' - 'A'));
        else
            cout << '*';
    }
    cout << endl;
    return 0;
}
```