



C++ 三级

2026 年 09 月

1 单选题（每题 2 分，共 30 分）

第 1 题 关于计算机的数据编码，下面说法不正确的是（ ）。

- ☐ A. 数字 5，不管采用原码、反码、补码，它的 8 位二进制形式都是 $(0000\ 0101)_2$ 。
- ☐ B. 计算机科学中，单位换算默认 $1\text{KB} = 1024\text{B}$ ，而非 1000B 。
- ☐ C. 原码是最直观的一种有符号数表示方法。
- ☐ D. 反码运算中，最高位进位要抛掉。

第 2 题 关于进制转换，下面说法不正确的是（ ）

- ☐ A. 数字 1101B 可以分解为：
 $1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 8 + 4 + 0 + 1 = 13$ （十进制）
- ☐ B. 数字 42Q 可以分解为：
 $4 \times 8^1 + 2 \times 8^0 = 32 + 2 = 34$ （十进制）
- ☐ C. 数字 2BEH 可以分解为：
 $2 \times 16^2 + 11(\text{B}) \times 16^1 + 14(\text{E}) \times 16^0 = 512 + 176 + 14 = 702$ （十进制）
- ☐ D. $345\text{Q} = 0\text{E3H}$ （通常写作 E3H ）

第 3 题 下面程序的运行结果是（ ）

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int x = 1;
5      for (int i = 0; i = 0; i <= 5) {
6          x++;
7      }
8      cout << x << endl;
9      return 0;
10 }
```

- ☐ A. 1
- ☐ B. 10
- ☐ C. 15
- ☐ D. 程序编译错误

第4题 关于计算机编码中补码，下面说法错误的是（ ）。

- ☐ A. 正数的补码：与其原码、反码相同。
- ☐ B. 负数的补码：将其对应的正数的原码按位取反（得到反码），然后加1。
- ☐ C. 8位的补码，最大表示数据是128，最小表示数据是-128。
- ☐ D. 负数的补码，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5题 下面程序执行的结果是（ ）。

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int x = 2026;
5      int count = 0;
6      while (x > 0) {
7          count++;
8          x = x & (x - 1);
9      }
10     cout << count << endl;
11 }
```

- ☐ A. 2026
- ☐ B. 8
- ☐ C. 16
- ☐ D. 32

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。
例子：数组 [5, 7, 9, 7, 5]（唯一数是 9）
- ☐ B. 若左移导致高位溢出（如超过整型范围），结果不再符合乘法规律。
- ☐ C. 有符号整数左移溢出时结果确定。
- ☐ D. $n+2$ 位二进制数 $A_1A_2\cdots A_n00$ ，一定是4的倍数（最后两位00是二进制的00）。

第7题 计算 $3 \& 7 \mid 15 \wedge \sim 11$ （ $a=3, b=7, c=15, d=11$ ）：（ ）。

- ☐ A. -9
- ☐ B. -1
- ☐ C. -7
- ☐ D. -5

第8题 下面函数是一个加密解密的函数，横线处应该填入的程序段是（ ）。

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main() {
5      int key;
6      cin >> key;
7      string msg = "Hello GESP";
8      cout << msg << endl;
9      for (int i = 0; i < msg.size(); i++) {
10         msg[i] = msg[i] ^ key;
11     }
12     cout << msg << endl;
13     -----
14     cout << msg << endl;
15     return 0;
16 }

```

☐ A.

```

1  for (int i = 0; i < msg.size(); i++) {
2      msg[i] = msg[i] | key;
3  }

```

☐ B.

```

1  for (int i = 0; i < msg.size(); i++) {
2      msg[i] = msg[i] >> key;
3  }

```

☐ C.

```

1  for (int i = 0; i < msg.size(); i++) {
2      msg[i] = msg[i] ^ key;
3  }

```

☐ D.

```

1  for (int i = 0; i < msg.size(); i++) {
2      msg[i] = msg[i] & key;
3  }

```

第9题 下列程序是分别判断输入的 N 个数是否可能是二进制、八进制、十进制、十六进制，横线处应该填入的是 ()

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int n = 0;
5      cin >> n;
6      for (int i = 0; i < n; i++) {
7          char str[11];
8          cin >> str;
9          char max = '0';
10         for (int i = 0; str[i] != '\0'; i++)
11             if (str[i] > max)
12                 max = str[i];
13         -----
14     }
15     return 0;
16 }

```

- ☐ 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;`

第 10 题 下面的程序是位掩码技术实现权限的组合管理。首先定义四个权限常量，分别对应读、写、修改、删除，每个权限占用整数的一个独立二进制位。初始化权限掩码 `mask` 为 `0`，代表没有任何权限。如果想实现如下的操作：添加读权限；添加写权限；清除读权限，其他位不变；添加删除权限。则横线处应该填入的是（ ）。

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      const int READ = 1 << 0;
6      const int WRITE = 1 << 1;
7      const int MODIFY = 1 << 2;
8      const int DELETE_ = 1 << 3;
9      int mask = 0;
10     -----
11     return 0;
12 }
```

☐ A.

```
1  mask = mask | READ;
2  mask = mask | WRITE;
3  mask = mask & (~READ);
4  mask = mask | DELETE_;
```

☐ B.

```
1  mask = mask & READ;
2  mask = mask & WRITE;
3  mask = mask | (~READ);
4  mask = mask & DELETE_;
```

☐ C.

```
1  mask = mask | READ;
2  mask = mask & WRITE;
3  mask = mask & (~READ);
4  mask = mask | DELETE_;
```

☐ D.

```
1  mask = mask | READ;
2  mask = mask | WRITE;
3  mask = mask | (~READ);
4  mask = mask | DELETE_;
```

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`，对于正整数，等价于对 `num` 做整数除以 2，向下取整
- ☐ B. `num | 0x0F`，可以把 `num` 的低 4 位全部置为 1，高位保持不变

- ☐ C. `num ^ num` 的结果一定等于 0
- ☐ D. 若 `num` 右移时低位被移出丢弃，结果一定符合除法数学规律

第 12 题 已知: `a=0, b=2, c=11, d=7, e=5` , 表达式 `!a & b << c ^ d | e` 的结果是 ()

- ☐ A. 5
- ☐ B. 13
- ☐ C. 7
- ☐ D. 28

第 13 题 关于 `string` 的成员函数, 下面说法错误的是 ()

- ☐ A. `size()` 返回字符串的字符个数, 不包含末尾的 `'\0'` ; `length()` 和 `size()` 功能完全等价
- ☐ B. `s.empty()` : 字符串为空时返回 `true` , 字符串不为空返回 `false`
- ☐ C. `s.substr(pos, len)` , 当省略第二个参数 `len` 时, 代表从 `pos` 位置一直截取到字符串末尾
- ☐ D. `s.erase(2)` 表示删除字符串中下标为 2 的这一个字符

第 14 题 以下数组定义, 符合 C++ 语法的是 ()

- ☐ A. `int a[] = {1, 2, 3.0};`
- ☐ B. `int b['A'];`
- ☐ C. `int c[ ];`
- ☐ D. `float d[10.0];`

第 15 题 三进制有个奇妙的应用, 有 1000 个酒缸, 恰好 1 缸有毒。毒酒喝下后 24 小时毒发死亡; 无毒酒无影响。总共有 48 小时, 可以做两轮实验, 每轮可以重新安排老鼠喝酒。有三种情况, 这只老鼠两轮都不喝这缸酒; 这只老鼠第一轮喝这缸酒; 这只老鼠第二轮喝这缸酒, 要保证找出毒酒, 最少需要多少只老鼠? ()

- ☐ A. 5
- ☐ B. 7
- ☐ C. 10
- ☐ D. 9

2 判断题 (每题 2 分, 共 20 分)

第 1 题 对于计算机编码运算, 补码做加法运算时, 如果最高位产生进位, 需要把该进位循环加到结果的最低位。

第 2 题 求 -13 的补码, 可以对正数 13 的二进制, 从右向左扫描, 遇到第一个 1 之后, 右边所有位全部取反, 左边保持不变。

第 3 题 一个 8 位二进制原码表示的有符号数, 可表示的数值范围是 -128 到 +127。

第 4 题 这段代码执行后, 能够将数组中所有大于 0 的元素赋值为 0。

```

1  int a[] = {2, -5, 7, 0, -1};
2  int n = sizeof(a) / sizeof(a[0]);
3  for (int num : a) {
4      if (num > 0) {
5          num = 0;
6      }
7  }

```

第5题 a 是 int 有符号整型，若 $(a \& 1) == 1$ 结果为 true，则 a 一定是正奇数。

第6题 char str[4] = "GESP"; 将在内存空间中给 str[4] 分配 5 个字节。

第7题 判断字符串中字符是否是 '0' 或者 '1'，可以写条件：if(str[i]==0 || str[i]==1)。

第8题 下面程序执行后，将输出 98。

```

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int a = 'a', b = 'b';
5      if (a > 0 || b++ > 0) {
6      }
7      cout << b;
8  }

```

第9题 枚举可以在保证不漏解的前提下缩小枚举范围，不是必须遍历全部。

第 10 题 下面流程图逻辑：初始化 $\text{sum} \leftarrow 0$, $i \leftarrow 0$ ；当 $i < n$ 成立时，执行 $\text{sum} \leftarrow \text{sum} + a[i]$ ，再执行 $i \leftarrow i + 1$ ；循环结束输出 sum 。该流程图可以正确计算一维数组 $a[0] \sim a[n]$ 全部元素的总和。

