



C++ 五级

2023 年 9 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

第 1 题 近年来，线上授课变得普遍，很多有助于改善教学效果的设备也逐渐流行，其中包括比较常用的手写板，那么它属于哪类设备？（ ）。

- A. 输入
- B. 输出
- C. 控制
- D. 记录

第 2 题 如果 `a` 和 `b` 均为 `int` 类型的变量，且 `b` 的值不为 0，那么下列能正确判断“`a` 是 `b` 的 3 倍”的表达式是（ ）。

- A. `(a >> 3 == b)`
- B. `(a - b) % 3 == 0`
- C. `(a / b == 3)`
- D. `(a == 3 * b)`

第 3 题 如果变量 `a` 和 `b` 分别为 `double` 类型和 `int` 类型，则表达式 `(a = 6, b = 3 * (7 + 8) / 2, b += a)` 的计算结果为（ ）。

- A. 6
- B. 21
- C. 28
- D. 不确定

第 4 题 有关下面 C++ 代码说法错误的是（ ）。

```

1 // sumA()和sumB()用于求从1到N之和
2 #include <iostream>
3 using namespace std;
4 int sumA(int n) {
5     int sum = 0;
6     for (int i = 1; i < n + 1; i++)
7         sum += i;
8     return sum;
9 }
10 int sumB(int n) {
11     if (n == 1)
12         return 1;
13     else
14         return n + sumB(n - 1);
15 }
16 int main() {
17     int n = 0;
18     cin >> n;
19     cout << sumA(n) << " " << sumB(n) << endl;
20     return 0;
21 }

```

- A. sumA() 用循环方式求从 1 到 N 之和， sumB() 用递归方式求从 1 到 N 之和。
- B. 默认情况下，如果输入正整数 1000，能实现求从 1 到 1000 之和。
- C. 默认情况下，如果输入正整数 100000，能实现求从 1 到 100000 之和。
- D. 一般说来，sumA() 的效率高于 sumB()。

第 5 题 下面C++代码以递归方式实现字符串反序，横线处应填上代码是（ ）。

```

1 //字符串反序
2 #include <iostream>
3 #include <string>
4 using namespace std;
5 string sReverse(string sIn) {
6     if (sIn.length() <= 1) {
7         return sIn;
8     } else {
9         return _____ // 此处填写代码
10    }
11 }
12 int main() {
13     string sIn;
14     cin >> sIn;
15     cout << sReverse(sIn) << endl;
16     return 0;
17 }

```

- A. sIn[sIn.length() - 1] + sReverse(sIn.substr(0, sIn.length() - 1));
- B. sIn[0] + sReverse(sIn.substr(1, sIn.length() - 1));
- C. sReverse(sIn.substr(0, sIn.length() - 1)) + sIn[sIn.length() - 1];
- D. sReverse(sIn.substr(1, sIn.length() - 1)) + sIn[sIn.length() - 1];

第6题 印度古老的汉诺塔传说：创世时有三根金刚柱，其中一柱从下往上按照大小顺序摞着64片黄金圆盘，当圆盘逐一从一柱借助另外一柱全部移动到另外一柱时，宇宙毁灭。移动规则：在小圆盘上不能放大圆盘，在三根柱子之间一次只能移动一个圆盘。下面的C++代码以递归方式实现汉诺塔，横线处应填入代码是（ ）。

```
1  #include <iostream>
2  using namespace std;
3  // 递归实现汉诺塔，将N个圆盘从A通过B移动C
4  // 圆盘从底到顶，半径必须从大到小
5  void Hanoi(string A, string B, string C, int N) {
6      if (N == 1) {
7          cout << A << " -> " << C << endl;
8      } else {
9          Hanoi(A, C, B, N - 1);
10         cout << A << " -> " << C << endl;
11         _____; // 此处填写代码
12     }
13 }
14 int main() {
15     Hanoi("甲", "乙", "丙", 3);
16     return 0;
17 }
```

- A. Hanoi(B, C, A, N - 2)
- B. Hanoi(B, A, C, N - 1)
- C. Hanoi(A, B, C, N - 2)
- D. Hanoi(C, B, A, N - 1)

第7题 根据下面C++代码的注释，两个横线处应分别填入（ ）。

```

1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  using namespace std;
5
6  bool isOdd(int N) {
7      return N % 2 == 1;
8  }
9  bool compare(int a, int b) {
10     if (a % 2 == 0 && b % 2 == 1)
11         return true;
12     return false;
13 }
14 int main() {
15     vector<int> lstA; // lstA是一个整型向量
16     for (int i = 1; i < 100; i++)
17         lstA.push_back(i);
18     // 对lstA成员按比较函数执行结果排序
19     sort(lstA.begin(), lstA.end(), _____); // 此处填写代码1
20
21     vector<int> lstB;
22     for (int i = 0; i < lstA.size(); i++) // lstB成员全为奇数
23         if (_____ ) // 此处填写代码2
24             lstB.push_back(lstA[i]);
25
26     cout << "lstA: ";
27     for (int i = 0; i < lstA.size(); i++)
28         cout << lstA[i] << " ";
29     cout << endl;
30
31     cout << "lstB: ";
32     for (int i = 0; i < lstB.size(); i++)
33         cout << lstB[i] << " ";
34     cout << endl;
35     return 0;
36 }

```

- A. compare 和 isOdd(lstA[i])
- B. compare(x1,y1) 和 isOdd
- C. compare 和 isOdd
- D. compare(x1,y1) 和 isOdd(lstA[i])

第8题 有关下面代码正确的是 ()。

```

1 // 在C++语言中，可以通过函数指针的形式，将一个函数作为另一个函数的参数。
2 // 具体来说：bool checkNum(bool (*Fx)(int), int N); 声明了一个函数，
3 // 其第一个参数是函数指针类型，指向一个接收一个int参数且返回值为bool的函数。
4 #include <iostream>
5 using namespace std;
6
7 bool isEven(int N) {
8     return N % 2 == 0;
9 }
10 bool checkNum(bool (*Fx)(int), int N) {
11     return Fx(N);
12 }
13 int main() {
14     cout << checkNum(isEven, 10) << endl;
15     return 0;
16 }

```

- A. checkNum() 函数定义错误。
- B. 将 isEven 作为 checkNum() 参数将导致错误。
- C. 执行后将输出 1。
- D. 运行时触发异常。

第9题 有关下面C++代码正确的是（ ）。

```

1 #include <iostream>
2 using namespace std;
3
4 bool isOdd(int N) {
5     return N % 2 == 1;
6 }
7 int Square(int N) {
8     return N * N;
9 }
10 bool checkNum(bool (*Fx)(int), int x) {
11     return Fx(x);
12 }
13 int main() {
14     cout << checkNum(isOdd, 10) << endl; // 输出行A
15     cout << checkNum(Square, 10) << endl; // 输出行B
16     return 0;
17 }

```

- A. checkNum() 函数定义错误。
- B. 输出行 A 的语句将导致编译错误。
- C. 输出行 B 的语句将导致编译错误。
- D. 该代码没有编译错误。

第10题 下面代码执行后的输出是（ ）。

```

1  #include <iostream>
2  using namespace std;
3
4  int jumpFloor(int N) {
5      cout << N << "#";
6      if (N == 1 || N == 2) {
7          return N;
8      } else {
9          return jumpFloor(N - 1) + jumpFloor(N - 2);
10     }
11 }
12 int main() {
13     cout << jumpFloor(4) << endl;
14     return 0;
15 }

```

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

第 11 题 下面代码中的 `isPrimeA()` 和 `isPrimeB()` 都用于判断参数 N 是否素数，有关其时间复杂度的正确说法是 ()。

```

1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  bool isPrimeA(int N) {
6      if (N < 2)
7          return false;
8      for (int i = 2; i < N; i++)
9          if (N % i == 0)
10             return false;
11     return true;
12 }
13 bool isPrimeB(int N) {
14     if (N < 2)
15         return false;
16     int endNum = int(sqrt(N));
17     for (int i = 2; i <= endNum; i++)
18         if (N % i == 0)
19             return false;
20     return true;
21 }
22 int main() {
23     cout << boolalpha;
24     cout << isPrimeA(13) << " " << isPrimeB(13) << endl;
25     return 0;
26 }

```

- A. `isPrimeA()` 的最坏时间复杂度是 $O(N)$ ，`isPrimeB()` 的最坏时间复杂度是 $O(\log N)$ ，`isPrimeB()` 优于 `isPrimeA()`。

- B. `isPrimeA()` 的最坏时间复杂度是 $O(N)$, `isPrimeB()` 的最坏时间复杂度是 $O(N^{\frac{1}{2}})$, `isPrimeB()` 优于 `isPrimeA()`。
- C. `isPrimeA()` 的最坏时间复杂度是 $O(N^{\frac{1}{2}})$, `isPrimeB()` 的最坏时间复杂度是 $O(N)$, `isPrimeA()` 优于 `isPrimeB()`。
- D. `isPrimeA()` 的最坏时间复杂度是 $O(\log N)$, `isPrimeB()` 的最坏时间复杂度是 $O(N)$, `isPrimeA()` 优于 `isPrimeB()`。

第 12 题 下面代码用于归并排序, 其中 `merge()` 函数被调用次数为 ()。

```

1  #include <iostream>
2  using namespace std;
3
4  void mergeSort(int * listData, int start, int end);
5  void merge(int * listData, int start, int middle, int end);
6
7  void mergeSort(int * listData, int start, int end) {
8      if (start >= end)
9          return;
10     int middle = (start + end) / 2;
11     mergeSort(listData, start, middle);
12     mergeSort(listData, middle + 1, end);
13     merge(listData, start, middle, end);
14 }
15 void merge(int * listData, int start, int middle, int end) {
16     int leftSize = middle - start + 1;
17     int rightSize = end - middle;
18
19     int * left = new int[leftSize];
20     int * right = new int[rightSize];
21     for (int i = 0; i < leftSize; i++)
22         left[i] = listData[start + i];
23     for (int j = 0; j < rightSize; j++)
24         right[j] = listData[middle + 1 + j];
25
26     int i = 0, j = 0, k = start;
27     while (i < leftSize && j < rightSize) {
28         if (left[i] <= right[j]) {
29             listData[k] = left[i];
30             i++;
31         } else {
32             listData[k] = right[j];
33             j++;
34         }
35         k++;
36     }
37
38     while (i < leftSize) {
39         listData[k] = left[i];
40         i++;
41         k++;
42     }
43     while (j < rightSize) {
44         listData[k] = right[j];
45         j++;
46         k++;
47     }
48     delete[] left;
49     delete[] right;
50 }
51 int main() {
52     int lstA[] = {1, 3, 2, 7, 11, 5, 3};
53     int size = sizeof(lstA) / sizeof(lstA[0]);
54
55     mergeSort(lstA, 0, size - 1); // 对lstA执行归并排序
56
57     for (int i = 0; i < size; i++)
58         cout << lstA[i] << " ";
59     cout << endl;
60 }

```

A. 0
return 0;

B. 1

C. 6

D. 7

第 13 题 在上题的归并排序算法中, mergeSort(listData, start, middle); 和 mergeSort(listData, middle + 1, end); 涉及到的算法为 ()。

A. 搜索算法

B. 分治算法

C. 贪心算法

D. 递推算法

第 14 题 归并排序算法的基本思想是 ()。

A. 将数组分成两个子数组, 分别排序后再合并。

B. 随机选择一个元素作为枢轴, 将数组划分为两个部分。

C. 从数组的最后一个元素开始, 依次与前一个元素比较并交换位置。

D. 比较相邻的两个元素, 如果顺序错误就交换位置。

第 15 题 有关下面代码的说法正确的是 ()。

```
1  #include <iostream>
2
3  class Node {
4  public:
5      int Value;
6      Node * Next;
7
8      Node(int Val, Node * Nxt = nullptr) {
9          Value = Val;
10         Next = Nxt;
11     }
12 };
13
14 int main() {
15     Node * firstNode = new Node(10);
16     firstNode->Next = new Node(100);
17     firstNode->Next->Next = new Node(111, firstNode);
18     return 0;
19 }
```

A. 上述代码构成单向链表。

B. 上述代码构成双向链表。

C. 上述代码构成循环链表。

D. 上述代码构成指针链表。

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 TCP/IP的传输层的两个不同的协议分别是UDP和TCP。

第 2 题 在特殊情况下流程图中可以出现三角框和圆形框。

第 3 题 找出自然数 N 以内的所有质数，常用算法有埃氏筛法和线性筛法，其中埃氏筛法效率更高。

第 4 题 在C++中，可以使用二分法查找链表中的元素。

第 5 题 在C++中，通过恰当的实现，可以将链表首尾相接，形成循环链表。

第 6 题 贪心算法的解可能不是最优解。

第 7 题 一般说来，冒泡排序算法优于归并排序。

第 8 题 C++语言中的 `qsort` 库函数是不稳定排序。

第 9 题 质数的判定和筛法的目的并不相同，质数判定旨在判断特定的正整数是否为质数，而质数筛法意在筛选出范围内的所有质数。

第 10 题 下面的C++代码执行后将输出 0 5 1 6 2 3 4 。

```
1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4
5  bool compareModulo5(int a, int b) {
6  |     return a % 5 < b % 5;
7  }
8  int main() {
9  |     int lst[7];
10 |     for (int i = 0; i < 7; i++)
11 |         lst[i] = i;
12
13 |     // 对序列所有元素按compareModulo5结果排序
14 |     sort(lst, lst + 7, compareModulo5);
15
16 |     for (int i = 0; i < 7; i++)
17 |         cout << lst[i] << " ";
18 |     cout << endl;
19 |     return 0;
20 }
```