

GESP C++六级样题

(满分：100分 考试时间：180分钟)

学校：_____

姓名：_____

题目	一	二	三	总分
得分				

一、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

1. 以下不属于计算机输出设备的有 ()。

- A. 麦克风
- B. 音箱
- C. 打印机
- D. 显示器

2. 小明想了一个 1 到 100 之间的整数。你可以做多次猜测，每次猜测之后，如果你没有猜中，小明会告诉你，你猜的数比他想的数大还是小。你希望你在运气最坏的情况下花费最少的次数猜中，请问你运气最坏的情况下会猜几次？（包括最后猜中的那次）

- A. 6
- B. 7
- C. 8
- D. 100

3. 关于分治算法，下列说法错误的是（ ）。
- A. 分治算法的核心思想是分而治之，即把问题转化为多个规模更小的子问题求解。
 - B. 分治算法可以不使用递归实现。
 - C. 分治算法的时间复杂度是 $O(\log N)$ ，其中 N 表示问题的规模。
 - D. 二分法、快速排序等算法都是典型的使用分治思想的算法。
4. 下面关于 C++类的说法中，正确的是（ ）。
- A. 派生类不能和基类有同名成员函数，因为会产生歧义。
 - B. 派生类可以和基类有同名成员函数，派生类覆盖基类的同名成员函数。
 - C. 派生类可以和基类有同名成员函数，但是否覆盖同名成员函数需要取决于函数参数是否一致。
 - D. C++中派生类不继承基类的任何成员函数。
5. 关于下面 C++代码说法错误的是（ ）。

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  class Pet {
5  public:
6      string kind;
7      int age;
8      Pet(string _kind, int _age):kind(_kind), age(_age) {}
9  }
10
11 class Dog : public Pet {
12 public:
13     string color;
14     Dog(string _kind, int _age, string _color):Pet(_kind, _age), color(_color) {}
15 }
16
17 int main() {
18     auto dog = Dog("dog", 3, "white");
19     cout << "kind: " << dog.kind << endl;
20     cout << "age: " << dog.age << endl;
21     cout << "color: " << dog.color << endl; // 输出行A
22     return 0;
23 }
```

- A. Pet 类是基类，Dog 类是子类。

- B. Dog 类的构造函数中，将自动调用 Pet 类的构造函数。
- C. dog 是 Dog 类的实例。
- D. 最后一行（即输出行 A）输出代码会报错，因为 Pet 类中没有成员变量 color

6. 以下几个类定义中不能顺利通过编译的是（ ）。

```
1 class A {
2 public:
3     void func(int a, int b, int c) {};
4 };
5
6 class B {
7 public:
8     void func(int a, int b=1, int c=1){}
9 };
10
11 class C {
12 public:
13     void func(int a=3, int b, int c){}
14 };
15
16 class D {
17 public:
18     void func(int a=3, int b=1, int c=0){}
19 };
```

- A. class A
- B. class B
- C. class C
- D. class D

7. 关于运算符重载，下列说法正确的是（ ）。

- A. 对于下列代码中的两个运算符重载实现：虽然它们的参数类型都是 int 与 Test 的组合，这段代码仍可以通过编译。这是因为在两个重载实现中，int 与 Test 在参数中的顺序不同，编译器能够唯一地确定调用哪个运算符实现代码。

```

1  #include <iostream>
2  using namespace std;
3
4  class Test{
5  public:
6      int data;
7      Test(int d):data(d){}
8  };
9
10 Test operator +(const Test& a, const int& b) {
11     return Test(a.data + b);
12 }
13
14 Test operator +(const int& b, const Test& a) {
15     return Test(- a.data - b);
16 }
17
18 int main(){
19     Test obj(1);
20     int data = 2;
21     cout << (obj + data).data << endl;
22     cout << (data + obj).data << endl;
23     return 0;
24 }

```

B. A 的说法是错误的。这是因为加法满足交换律，因此即便调换 A 和 int 的顺序，加法运算的实现仍必须保持一致，编译器在编译时也会检查这一点。如将后一处 int 改为 double，则可通过编译。

C. A 和 B 的说法都是错误的。运算符重载时，所有参数以及返回值的类型必须完全相同。因此，即便是下列代码中的运算符重载实现，也不能通过编译。

```

1  class Test{
2  public:
3      int data;
4      Test(int d):data(d){}
5  };
6
7  Test operator +(const Test& a, const int& b) {
8      return Test(a.data + b);
9  }

```

D. A、B、C 的说法都是错误的。

8. 关于 C++程序的异常处理，以下选项中描述错误的是（ ）。

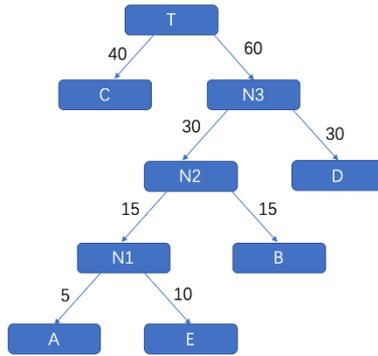
- A. 编程语言中的异常和错误是不同的概念。
- B. 异常一旦发生，程序便一定不能继续执行。
- C. 通过 try、catch 等保留字提供异常处理功能。
- D. 程序使用 throw 在任何地方抛出各种异常。

9. 有关下面 C++代码的说法，正确的是（ ）。

```
1  #include <iostream>
2  #include <cassert>
3
4  using namespace std;
5
6  class MoreData {
7      int* __data;
8      int head, tail, capacity;
9  public:
10     MoreData(int cap) {
11         capacity = cap;
12         __data = new int[capacity];
13         head = tail = 0;
14     }
15     MoreData& push(int val) {
16         assert(tail < capacity);
17         __data[tail++] = val;
18         return *this;
19     }
20     int pop() {
21         assert(head < tail);
22         return __data[--tail];
23     }
24     int size() {
25         return tail - head;
26     }
27 };
28
29 int main() {
30     auto myData = MoreData(100);
31     myData.push(4).push(5);
32     cout << myData.pop() << endl;
33     cout << myData.pop() << endl;
34     cout << myData.pop() << endl;
35     return 0;
36 }
```

- A. moreData 类可用于构造队列（queue）数据结构。
- B. myData.push(4).push(5);的连续 push()用法将导致错误。
- C. 前两个 cout << myData.pop() << endl;代码可以正确运行，分别输出 4 和 5。
- D. 最后一个 cout << myData.pop() << endl;代码可以通过编译，但不能正常运行，因为 pop 函数中的断言 assert(head < tail);会失败。

10. 如下图所示的哈夫曼树，按照哈夫曼编码规则，假设图中字符 C 的编码为 0，则 E 的编码为（ ）。



- A. 1111
- B. 1010
- C. 1101
- D. 1001

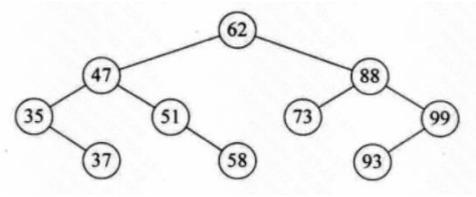
11. 下面有关格雷码的说法，错误的是（ ）。

- A. 格雷码是无权码。
- B. 格雷码是循环码。
- C. 011 和 100 是一组相邻的格雷码。
- D. 格雷码相邻的码组间仅有一位不同。

12. 在具有 $2N$ 个结点的完全二叉树中，叶子结点个数为（ ）。

- A. $N/2$
- B. $N - 1$
- C. N
- D. $N + 1$

13. 有关下图的二叉树，说法正确的有（ ）个。



- 是完全二叉树
- 是二叉搜索树
- 是平衡二叉树

- A. 0
- B. 1
- C. 2
- D. 3

14. 现希望存储整数 N 的所有质因子，请问其空间复杂度上界为是（ ）。

- A. $O(\log \log N)$
- B. $O(\log N)$
- C. $O(\sqrt{N})$
- D. $O(N)$

15. 下面 C++代码实现了某种排序算法，其中代码片段 `my_sort(arr, begin, i);`

`my_sort(arr, i + 1, end);`采用的算法思想是（ ）。

```

1 void my_sort(int arr[], int begin, int end) {
2     if (begin >= end - 1) return;
3     int i = begin, j = end - 1, x = arr[begin];
4     while (i < j)
5     {
6         while(i < j && arr[j] >= x)
7             j--;
8         if(i < j)
9             arr[i++] = arr[j];
10
11        while(i < j && arr[i] < x)
12            i++;
13        if(i < j)
14            arr[j--] = arr[i];
15    }
16    arr[i] = x;
17    my_sort(arr, begin, i);
18    my_sort(arr, i + 1, end);
19 }
  
```

- A. 递推
- B. 贪心
- C. 分治
- D. 搜索