

复旦大学计算机科学技术学院

2019~2020 学年第二学期期末考试试卷

☒ A 卷 ☐ B 卷 ☐ C 卷

课程名称: 面向对象程序设计 课程代码: COMP130135

开课院系: 计算机科学技术学院 考试形式: ☐ 开卷 ☒ 闭卷

姓名: _____ 学号: _____ 专业: _____

(本试卷答卷时间为 120 分钟, 答案必须写在答题纸上, 做在试卷上无效)

我已知悉学校与考试相关的纪律以及违反纪律的后果, 并将严守纪律, 不作弊, 不抄袭, 独立答题。

学生(签名): _____

题号	一	二	三	四	总分
分值	24	20	24	32	100

一、选择题(每题只选择一个答案; 如选多个, 不计分。每题 3 分, 总分 24 分)

1. 以下哪个不是 C++ 语言的关键字_____。
A. statics
B. inline
C. cout
D. struct
2. 类 Demo 的复制构造函数的原型是_____。
A. Demo& Demo(Demo& rhs)
B. Demo(Demo& rhs)
C. Demo& Demo(const Demo& rhs)
D. Demo(const Demo& rhs)
3. 以下定义有效的是_____。
1) const std::string hello = "Hello";
 const std::string message = hello + ", world" + "!";
2) const std::string world = ", world";
 const std::string message = "Hello" + world + "!";
3) const std::string exclam = "!";
 const std::string message = "Hello" + ", world" + exclam;
4) const std::string message = "Hello" + ", world" + "!";
A. 1) B. 1), 2) C. 1), 2), 3) D. 1), 2), 3), 4)

4. 以下关于函数形参和实参的描述中，错误的是_____。
- A. 非引用类型做函数形参，函数调用时，实参向形参传递值
 - B. 引用类型做函数形参，函数调用时，可以避免复制实参的开销
 - C. 非 `const` 引用作函数形参，实参可以是 `const` 引用
 - D. `const` 引用做函数形参，实参可以是非 `const` 引用
5. 以下关于模板函数的描述中，错误的是_____。
- A. 模板函数是 C++ 实现泛型函数的语言设施。
 - B. 定义模板函数时，并不知道模板参数的具体类型。
 - C. 编译程序时，模板参数的类型并不明确；运行程序时，模板参数的类型才能确定。
 - D. 模板参数不同，则对应的实例函数不同。
6. 以下关于类的保护标签的描述中，错误的是_____。
- A. 类的保护标签定义了其后成员的可访问性。
 - B. `public` 保护标签下定义类成员，仅类的外部可以访问。
 - C. `private` 保护标签下定义类成员，仅类本身可以访问。
 - D. 保护标签可按任意顺序出现，亦可多次出现。
7. 以下关于构造函数的描述中，正确的有_____项。
- 1) 构造函数是类的特殊的成员函数，定义了类的对象如何初始化。
 - 2) 编写程序时，不能像调用其它成员函数那样显式调用构造函数，创建类的对象时会自动调用。
 - 3) 构造函数必须定义在 `public` 保护标签下。
 - 4) 构造函数名与类名相同，且没有返回值类型。
- A. 1 B. 2 C. 3 D. 4
8. 如果有如下定义：
- ```
static string const TIOBE[] = {"java", "C", "python", "c++", "c#",
"Visual Basic", "javascript", "PHP", "SQL", "R"};
string Lan[10];
```
- 且已包含相关头文件和 `using` 声明，则以下程序语句能正确运行的是\_\_\_\_\_。
- A. `TIOBE[0][0] = 'J';`
  - B. `TIOBE[9] = "Go";`
  - C. `copy(TIOBE, TIOBE+10, Lan);`
  - D. `sort(TIOBE, TIOBE+10);`

## 二、程序阅读题（每题 5 分，共 20 分）

1. 以下程序运行时，如果输入：1 2 3 1 1 2 3，则输出为：\_\_\_\_\_

|                                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                        |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>#include &lt;iostream&gt; #include &lt;list&gt;  using std::list; using std::cin; using std::cout; using std::endl;  int main(){     list&lt;int&gt; li;     int i;      while(cin &gt;&gt; i)         li.push_back(i);</pre> | <pre>list&lt;int&gt;::iterator iter; for(iter = li.begin(); iter != li.end(); iter++)     if(*iter == 1)         iter = li.erase(iter);  for(iter = li.begin(); iter != li.end(); iter++)     cout &lt;&lt; *iter &lt;&lt; " "; cout &lt;&lt; endl;  return 0; }</pre> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

2. 以下程序的运行结果是：\_\_\_\_\_

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                                                           |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>#include &lt;iostream&gt;  using std::cout;  class Item { public:     int cnt;     Item() :cnt(0) {} };  class Store { public:     Store() {         itm = new Item();         cnt++;     }     Store(const Store&amp; p) {         itm = new Item();         itm-&gt;cnt = p.itm-&gt;cnt++;         cnt++;     }     Store&amp; operator=(const Store&amp; p) {         if (this != &amp;p) {             delete itm;             itm = new Item();             itm-&gt;cnt = p.itm-&gt;cnt++;         }         return *this;     } }</pre> | <pre>int getItem() {     return itm-&gt;cnt; } static int getcnt() {     return cnt; } ~Store() {     cnt--;     delete itm; } private:     Item * itm;     static int cnt; };  int Store::cnt = 0;  int main() {     Store p1, p2;     p2 = p1;     Store p3 = p2;     cout &lt;&lt; p1.getItem() + p2.getItem()     + p1.getcnt() + p2.getcnt();      return 0; }</pre> |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

3. 以下程序的运行结果是：

|                                                                                                                                                                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>#include &lt;iostream&gt;  using std::ostream; using std::cout;  class Point {     friend ostream&amp; operator&lt;&lt;(ostream&amp;, const Point&amp;); public:     Point(int i = 0, int j = 0) :x(i), y(j) {}     Point&amp; operator+=(Point p){         this-&gt;x += p.x;         this-&gt;y += p.y;         return *this;     } private:     int x, y; };</pre> | <pre>Point operator+(const Point&amp; p1, const Point&amp; p2){     Point p = p1;     p += p2;     return p; }  ostream&amp; operator&lt;&lt;(ostream&amp; os, const Point&amp; p){     os &lt;&lt; '(' &lt;&lt; p.x &lt;&lt; ", " &lt;&lt; p.y &lt;&lt; ')';     return os; }  int main(){     Point p1(1, 2), p2 = 1, p3 = 2;     p3 += p1 + p2;     cout &lt;&lt; p3;      return 0; }</pre> |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

4. 以下程序的运行结果是：

|                                                                                                                                                                                                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>#include &lt;iostream&gt;  using std::cout;  class X { public:     X(int i = 0) :x(i) {}     int add(int i)const     { return x + i; }     virtual int multiply(int i, int j = 0)const     { return x * i; } protected:     int x; };</pre> | <pre>class Y :public X { public:     Y(int i = 0, int j = 0) :X(i), y(j) {}     int add(int i, int j)const     { return x + i + y + j; }     int multiply(int i, int j)const     { return x * j + y * i; } private:     int y; };  int main(){     Y y(5);     const X&amp; x = y;     cout &lt;&lt; x.add(5) &lt;&lt; ' ' &lt;&lt; x.multiply(5) &lt;&lt; ' ';     cout &lt;&lt; y.add(5, 5) &lt;&lt; ' ' &lt;&lt; y.multiply(5, 5);      return 0; }</pre> |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

### 三、程序填空题（每空 3 分，共 24 分）

下面的代码实现合并区间的功能。给定区间[15,18]，[2,6]，[5,10]，[1,3]，合并后得到区间[1,10]，[15,18]。源代码中区间使用类 Interval 实现；函数 merge\_interval 实现了合并区间的功能。必要的头文件和 using 语句已经略去。

```

class Interval{
public:
 int start, end;
 Interval__(1)_____ ;
};

typedef vector<Interval> Intervals;

bool compare_interval(const Interval&a, const Interval&b){
 __(2)_____ ;
}

Intervals merge_intervals(Intervals & intervals){
 Intervals merged;
 if (__(3)_____)
 return merged;

 sort(__(4)_____);
 for(std::size_t idx = 0;__(5)_____ ; ++idx){
 const Interval& interval = intervals[idx];
 if (__(6)_____)
 merged.back().end = std::max(interval.end, merged.back().end);
 else
 __(7)_____ ;
 }

 return merged;
}

int main(){
 Intervals vec = {
 Interval(15, 18), Interval(2, 6),
 Interval(5, 10), Interval(1, 3)
 };
 Intervals merged = merge_intervals(vec);
 for(std::size_t idx = 0; idx<merged.size(); ++idx){
 __(8)_____ ;
 std::cout << "[" << interval.start << "," << interval.end << "]" << ",";
 };

 return 0;
}

```

#### 四、编程题（共 32 分）

1.（12 分）编写模板函数 `selectSort` 实现对区间 `[begin,end)` 之间的进行选择排序。

选择排序的基本思想是：首先在整个选出序列中最小的元素，将它与序列的第一个交换位置；然后再次从余下的序列中选出最小的结点，将其与序列的第二个元素交换位置；...；直到整个序列完成排序。

2.（20 分）按要求用 C++ 实现表示公司的类 `Company`，表示雇员的类 `Employee`，表示经理的类 `Manager` 和表示程序员的类 `Programmer`。每个公司有若干位雇员，雇员分为两类：经理和程序员。公司为每位雇员发放工资和分红，每位雇员根据工作时长、加班时长和股票数量计算工资。经理的单位时长工作工资是 200，加班无收入，有奖金收入；程序员的单位时长工作工资是 150，单位时长加班工资是 300，无股票收入。以下为具体要求：

1) 类 `Company` 包含一个记录雇员信息的变量 `ep`，以及增加雇员的函数 `AddEmployee(Employee *e)`，减少雇员的函数 `RemoveEmployee(string name)`，输出所有雇员姓名和收入的函数 `Print`，以及析构函数。

2) 类 `Employee` 是经理类 `Manager` 和程序员类 `Programmer` 的基类，包含一个记录雇员姓名的变量 `n`，一个雇员姓名为参数的构造函数，一个计算雇员收入的虚函数 `salary`，以及其虚析构函数。

3) 类 `Manager` 包含一个记录工作时间的变量 `hours` 和记录奖金的变量 `bonus`，一个以姓名、工作时长和奖金为参数的构造函数，以及计算雇员收入的函数 `salary`。

4) 类 `Programmer` 包含一个记录工作时长的变量 `hours` 和加班时长的变量 `overtime`，一个以姓名、工作时长和加班时长为参数的构造函数以及计算雇员收入的函数 `salary`。

测试程序如下：

```
int main(){
 Company cp;
 cp.AddEmployee(new Manager("Wang",20,500));
 cp.AddEmployee(new Programmer("Chen",20,10));
 cp.AddEmployee(new Programmer("Zhao",30,0));
 cp.RemoveEmployee("Chen");
 cp.AddEmployee(new Programmer("Lin",15,5));
 cp.Print();
 return 0;
}
```

输出：

```
Wang:4500
Zhao:6000
Lin:4500
```