C++的试卷上的一些问题

整理by 2015级计算机科学与技术 冉诗菡

1．要求通过函数来实现一种不太复杂的功能，并且要求加快执行速度，选用 A

A．内联函数　　B．重载函数　　C．内部函数　　D．函数模板

- 内联函数：从源代码层看，有函数的结构，而在编译后，却不具备函数的性质。内联函数不是在调用时发生控制转移，而是在编译时将函数体嵌入在每一个调用处。编译时，类似宏替换，使用函数体替换调用处的函数名。一般在代码中用inline修饰，但是能否形成内联函数，需要看编译器对该函数定义的具体处理。内联扩展是用来消除函数调用时的时间开销。它通常用于频繁执行的函数。 一个小内存空间的函数非常受益。
- 为了减少时间开销，如果在类体中定义的成员函数中不包括循环等控制结构，可以省略inline，C++系统会自动将它们作为内置(inline)函数来处理。
- 如果成员函数不在类体内定义，而在类体外定义，系统并不把它默认为内置(inline)函数，调用这些成员函数的过程和调用一般函数的过程是相同的。如果想将这些成员函数指定为内置函数，应当用inline作显式声明。如果在类体外定义inline函数，则必须将类定义和成员函数的定义都放在同一个头文件中(或者写在同一个源文件中)，否则编译时无法进行置换(将函数代码的拷贝嵌入到函数调用点)。

3．关于类和对象不正确的说法是C

A．类是一种类型，它封装了数据和操作　　B．对象是类的实例

C．一个类的对象只有一个　　D．一个对象必属于某个类

- 类是对象的抽象，而对象是类的具体实例。类是抽象的，不占用内存，而对象是具体的，占用存储空间。

4．下列有关构造函数的描述中，正确的是B

A．构造函数可以带有返回值　　B．构造函数的名字与类名完全相同

C．构造函数必须带有参数　　D．构造函数必须定义，不能缺省

- 一个类可以有多个构造函数 ，可根据其参数个数的不同或参数类型的不同来区分它们即构造函数的重载。
- 构造函数的命名必须和类名完全相同。
- 构造函数的功能主要用于在类的对象创建时定义初始化的状态。它没有返回值，也不

能用void来修饰。

- 构造函数不能被直接调用，必须通过new运算符在创建对象时才会自动调用
- 当一个类没有定义任何构造函数，C#编译器会为其自动生成一个默认的无参的构造函数。

5．假定a为一个整型数组名，则元素a[4]的字节地址为C

A．a+4　　B．a+8　　C．a+16　　D．a+32

- 因为int类型占四个字节

6．类中能访问静态成员的函数是B

A．虚函数　　B．静态成员函数　　C．构造函数　　D．析构函数

- 析构函数是析构对象

- 关于静态成员：
- 普通数据成员属于类的一个具体的对象，只有对象被创建了，普通数据成员才会被分配内存。而静态数据成员属于整个类，即使没有任何对象创建，类的静态数据成员变量也存在。
- 因为类的静态数据成员的存在不依赖与于任何类对象的存在，类的静态数据成员应该在代码中被显式地初始化，一般要在类外进行。
- 外部访问类的静态成员能直接通过类名来访问，例如：test::getCount()。虽然静态成员不属于类的某个对象，但是我们仍然可以使用类的对象、引用或指针来访问静态成员。
- 类的静态成员函数无法直接访问普通数据成员（可以通过对象名间接的访问），而类的任何成员函数都可以访问类的静态数据成员。
- 静态成员和类的普通成员一样，也具有public、protected、private3种访问级别，也可以具有返回值、const修饰符等参数。

8．关于友元函数的描述中，错误的是 B

A．友元函数不是成员函数

B．友元函数只能访问类中私有成员

C．友元函数破坏隐藏性，尽量少用

D．友元函数说明在类体内，使用关键字friend

- 友元函数是指某些虽然不是类成员却能够访问类的所有成员的函数（可以访问对象的

私有成员，但普通函数不行），但它不能直接访问类的成员，只能访问对象的所有成员，调用友元函数时，在实际参数中需要指出要访问的对象。

- 必须在类的说明中说明友元函数，说明时以关键字friend开头，后跟友元函数的函数原型，友元函数的说明可以出现在类的任何地方，包括在private和public部分；
- 注意友元函数不是类的成员函数，所以友元函数的实现和普通函数一样，在实现时不用"::"指示属于哪个类，只有成员函数才使用"::"作用域符号；
- 类与类之间的友元关系不能继承。
- 一个类的成员函数也可以作为另一个类的友元，但必须先定义这个类。

9．关于类模板的说法正确的是B

A．类模板的主要作用是生成抽象类

B．类模板实例化时，编译器将根据给出的模板实参生成一个类

C．在类模板中的数据成员具有同样类型

D．类模板中的成员函数没有返回值

- 类模板：
- 类模板是对一批仅仅成员数据类型不同的类的抽象
- 模板的类型参数由关键字class 或关键字typename 及其后的标识符构成。在模板参数表中关键字class 和typename 的意义相同。
- 类模板的使用实际上是将类模板实例化成一个具体的类，它的格式为：类名<实际的类型>。

- 抽象类：
- 在C++中，含有纯虚拟函数的类称为抽象类，它不能生成对象
- 抽象类是不完整的，它只能用作基类。
- 抽象类不能实例化。

11．假定类AB中有一个公有属性的静态数据成员`static int bb`；在类外不通过对象名给该成员bb赋值为10的写法：`int AB::bb=10;`

12．如果要把类B的成员函数`void fun( )`说明为类A的友元函数，则应在类A中加入语句：`friend void B::fun();`

- 注意void在B的前面 不要写成`friend B::void fun();`

13．执行下列程序`double a=3．1415926，b=3．14；cout<<setprecision(5)`

<<a<< ", " <<
setprecision(5)<<b<<endl；程序的输出结果是 3.1416, 3.14

- setprecision (4)
  四舍五入 保留四位有效数字 （并不是保留到小数点后四位） 如果位数不够的话
  并不会在末尾补0
  如果用了这个 最好是要记住prec 然后set了精度之后要再set回之前的prec精度

14．为了实现运行时的多态性，派生类需重新定义基类中的虚函数

- 虚函数与纯虚函数 在他们的子类中都可以被重写。它们的区别是：
  （1）纯虚函数只有定义，没有实现；而虚函数既有定义，也有实现的代码。
  纯虚函数一般没有代码实现部分，如
  virtual void print() = 0;
  而一般虚函数必须要有代码的实现部分，否则会出现函数未定义的错误。
  virtual void print()
  { printf("This is virtual function\n"); }
  （2）包含纯虚函数的类不能定义其对象，而包含虚函数的则可以。
- 虚函数：

15．在C++中有两种参数传递方式：传值和传引用
（类似c程里面的值传递和地址传递

16．已知int*p=NULL，使用new为指针P申请一个存储大小为10的存放int型的空间，代码为p = new int[10];

17．每个对象都是所属类的一个实例

18．函数重载时，编译系统会根据形参的类型或形参的个数来区分。

19．静态成员函数、友元函数、构造函数和析构函数中，不属于成员函数的是友元函数。

20．局部对象和全局对象中，在同一程序中全局对象生存期最长。

- 改错题很喜欢考的点：
-

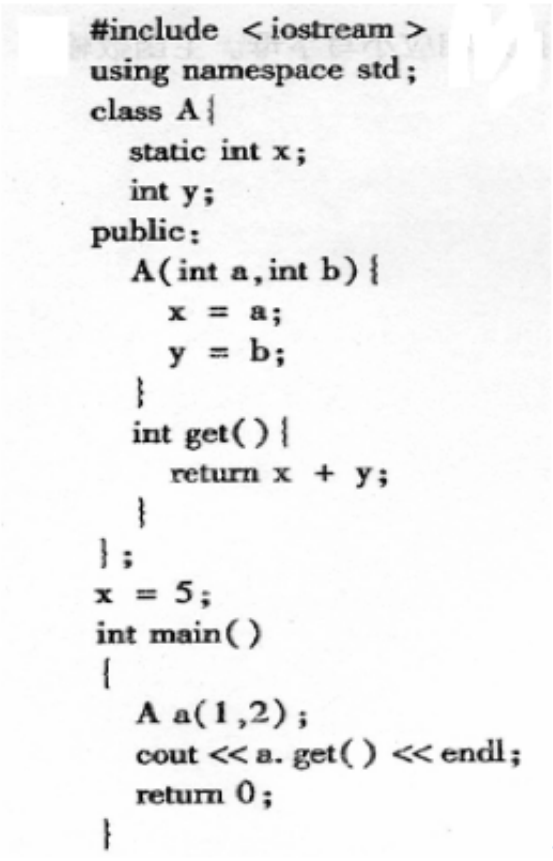
```
#include <iostream>
using namespace std;
class A{
  int x;
public:
  A(int a){
    x = a;
  }
  void set(int a){
    x = a;
  }
  void get(){
    cout << x << endl;
  }
};
int main()
{
  const A a(4);
  a.set(6);
  a.get();
  a.set(10);
  a.get();
  return 0;
}
```

a是常对象，不能更新。修改：将a修改为非常对象。

22

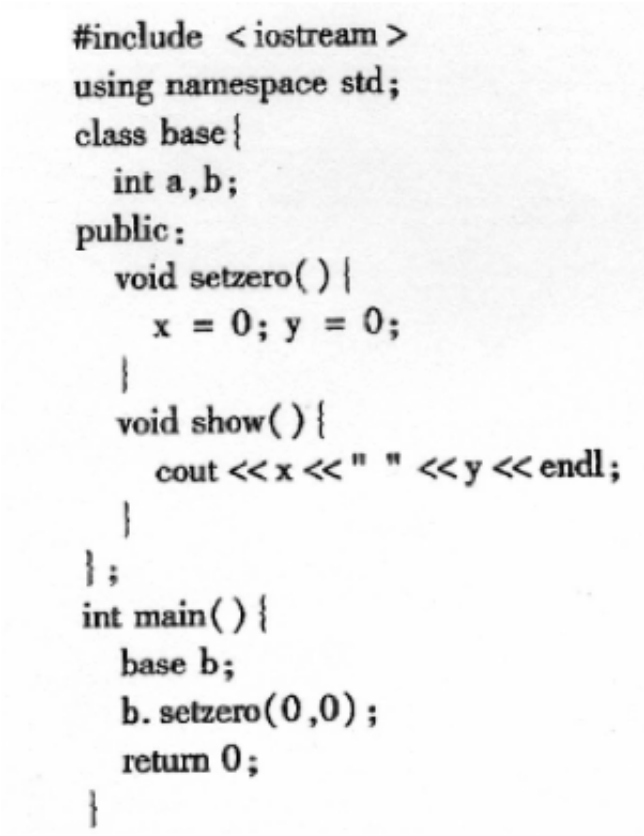
```
#include <iostream>
using namespace std;
class base{
  int *p;
public:
  base (int a){
    p = &a;
  }
  int get(){
    return p;
  }
};
int main()
{
base b(3);
cout << b. get();
}
```

构造函数中p = &a错，a是一个形参啊！所以是一个局部变量的 指针不能保存局部变量a的引用，因为当这个构造函数运行完了之后局部变量就会被释放掉的 指针就没有指的地方了；return p错，返回值与说明的类型不一致。

- 

```
#include <iostream>
using namespace std;
class A{
  static int x;
  int y;
public:
  A(int a,int b){
    x = a;
    y = b;
  }
  int get(){
    return x + y;
  }
};
x = 5;
int main()
{
  A a(1,2);
  cout << a.get() << endl;
  return 0;
}
```

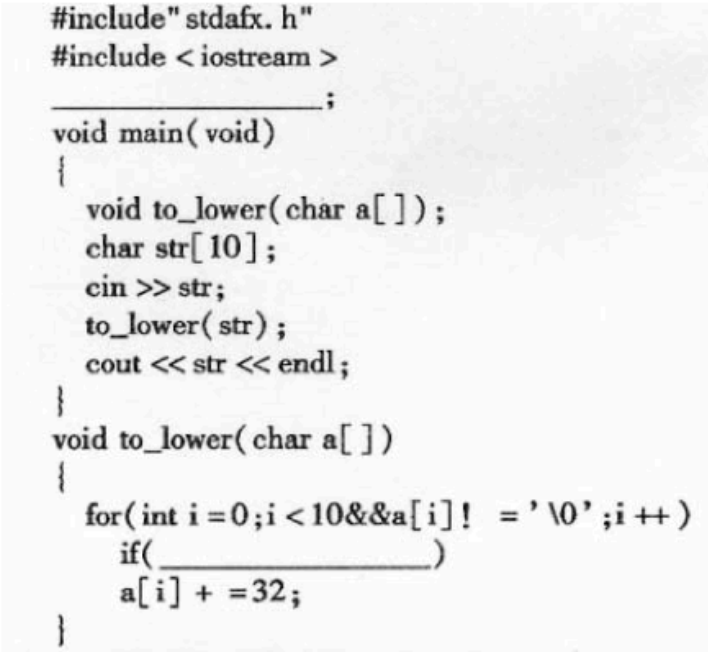
x为静态类变量，其赋值形式错误。修改：`int A::x=5;`

- 

```
#include <iostream>
using namespace std;
class base{
  int a,b;
public:
  void setzero(){
    x = 0; y = 0;
  }
  void show(){
    cout<<x<<" "<<y<<endl;
  }
};
int main(){
  base b;
  b.setzero(0,0);
  return 0;
}
```

b.setzero(0,0)错误，不存在这样样式的函数。修改：将其修改为b.setzero()

26. 编一个函数 to_ lower( )，实现将字符串中的大写字母转换成相应小写字母。主函数输入数据并输出结果。

- 

```
#include"stdafx.h"
#include<iostream>
_____;
void main(void)
{
  void to_lower(char a[]);
  char str[10];
  cin>>str;
  to_lower(str);
  cout<<str<<endl;
}
void to_lower(char a[])
{
  for(int i=0;i<10&&a[i]!='\0';i++)
    if(_____)
    a[i]+=32;
}
```

using namespace std;　　a[i]>='A'&&a[i]<='Z'

跟c里面的其实差不多诶 都适用ascii

```
void to_lower(char a[]);
```

传进去的是字符数组的地址

- private成员在派生类当中无法访问 要更改访问权限为public或者protected 这样对于派生类来说才是透明的

- 传进去的时候传的是地址还是引用还是对象
- 调用一个函数的时候，这个函数形参有两个，而调用的时候只有一个实参，可以使第二个带默认值。eg: void fun(int i,int j=0)
-