一、请填写 BOOL , float, 指针变量 与“零值”比较的 if 语句。

提示：这里“零值”可以是 0, 0.0 , FALSE 或者“空指针”。例如 int 变量 n 与“零值”比较的 if 语句为：

```
if ( n == 0 )
if ( n != 0 )
```

以此类推。

请写出 BOOL flag 与“零值”比较的 if 语句： **if (flag)** **if (!flag)**
请写出 float x 与“零值”比较的 if 语句： **const float EPSINON = 0.00001;** **if ((x >= - EPSINON) && (x <= EPSINON)**
请写出 char *p 与“零值”比较的 if 语句： **if (p == NULL)** **if (p != NULL)**

二、简答题

1、头文件中的 ifndef/define/endif 干什么用？

答：防止该头文件被重复引用。

2、#include <filename.h> 和 #include “filename.h” 有什么区别？

答：对于#include <filename.h> ，编译器从标准库路径开始搜索 filename.h

对于#include “filename.h”，编译器从用户的工作路径开始搜索 filename.h

3、const 有什么用途？（请至少说明两种）

答：（1）可以定义 const 常量

（2）const 可以修饰函数的参数、返回值，甚至函数的定义体。被 const 修饰的东西都受到强制保护，可以预防意外的变动，能提高程序的健壮性。

4、在 C++ 程序中调用被 C 编译器编译后的函数，为什么要加 extern “C”声明？

答：C++语言支持函数重载，C 语言不支持函数重载。函数被 C++编译后在库中的名字与 C 语言的不同。假设某个函数的原型为： `void foo(int x, int y);`

该函数被 C 编译器编译后在库中的名字为_foo，而 C++编译器则会产生像_foo_int_int 之类的名字。

C++提供了 C 连接交换指定符号 extern“C”来解决名字匹配问题。

5、 内存分配方式有几种？

答：[1]从静态存储区域分配。内存在程序编译的时候就已经分配好，这块内存在程序的整个运行期间都存在。例如全局变量，static 变量。

[2]在栈上创建。在执行函数时，函数内局部变量的存储单元都可以在栈上创建，函数执行结束时这些存储单元自动被释放。栈内存分配运算内置于处理器的指令集中，效率很高，但是分配的内存容量有限。

[3]从堆上分配，亦称动态内存分配。动态内存的生存期由程序员决定，使用非常灵活，但如果在堆上分配了空间，就有责任回收它，否则运行的程序会出现内存泄漏，频繁地分配和释放不同大小的堆空间将会产生堆内碎块。

6、基类的析构函数不是虚函数，会带来什么问题？

答：如果析构函数不被声明成虚函数，则编译器实施静态绑定，在删除基类指针时，只会

调用基类的析构函数而不调用派生类析构函数，这样就会造成派生类对象析构不完全。所以，将析构函数声明为虚函数是十分必要的。（virtual table）

7、阐述 C++中多态的实现机制？

答：在基类的函数前加上 virtual 关键字，在派生类中重写该函数，运行时将会根据对象的实际类型来调用相应的函数。如果对象类型是派生类，就调用派生类的函数；如果对象类型是基类，就调用基类的函数。

8、在 C/C++中 static 有什么用途？（请至少说明两种）

答： [1]在函数体，一个被声明为静态的变量在这一函数被调用过程中维持其值不变。

[2]在模块内（但在函数体外），一个被声明为静态的变量可以被模块内所用函数访问，但不能被模块外其它函数访问。它是一个本地的全局变量。

[3]在模块内，一个被声明为静态的函数只可被这一模块内的其它函数调用。那就是，这个函数被限制在声明它的模块的本地范围内使用

9、将“引用”作为函数参数有哪些特点？

答：[1] 传递引用给函数与传递指针的效果是一样的。这时，被调函数的形参就成为原来主调函数中的实参变量或对象的一个别名来使用，所以在被调函数中对形参变量的操作就是对其相应的目标 对象（在主调函数中）的操作。

[2] 使用引用传递函数的参数，在内存中并没有产生实参的副本，它是直接对实参操作；而使用一般变量传递函数的参数，当发生函数调用时，需要给形参分配存储单元，形参变量是实参变量的 副本；如果传递的是对象，还将调用拷贝构造函数。因此，当参数传递的数据较大时，用引用比 用一般变量传递参数的效率和所占空间都好。

[3] 使用指针作为函数的参数虽然也能达到与使用引用的效果，但是，在被调函数中同样要给形参分配存储单元，且需要重复使用”*指针变量名”的形式进行运算，这很容易产生错误且程序的阅 读性较差；另一方面，在主调函数的调用点处，必须用变量的地址作为实参。而引用更容易使用，更清晰。

10、请简述以下两个 for 循环的优缺点

<table>
<tr>
<td>

```
// 第一个
for (i=0; i<N; i++)
{
  if (condition)
      DoSomething();
  else
      DoOtherthing();
}
```

</td>
<td>

```
// 第二个
if (condition)
{
    for (i=0; i<N; i++)
      DoSomething();
}
else
{
    for (i=0; i<N; i++)
      DoOtherthing();
}
```

</td>
</tr>
<tr>
<td>优点：程序简洁

缺点：多执行了 N-1 次逻辑判断，并且打断了循环“流水线”作业，使得编译器不能对循环进行优化处理，降低了效率。</td>
<td>优点：循环的效率高

缺点：程序不简洁</td>
</tr>
</table>

三、填空题

1、在定义类对象的语句执行时，系统在建立每个对象的过程中将自动调用该类的(**构造函数**)使其初始化。

2、当一个类对象被撤消时将自动调用该类的(**析构函数**)。

3、对基类数据成员的初始化是通过执行派生类构造函数中的(**成员初始化列表**)来实现的。

4、对一个类中的数据成员的初始化可以通过构造函数中的(**赋值**)实现，也可以通过构造函数中的(**初始化列表**)实现。

5、在一个派生类中，对基类成员、类对象成员和非类对象成员的初始化次序是先(**基类成员**)，后(**类对象成员**)，最后为(**非类对象成员**)。

6、当撤消一个含有基类和类对象成员的派生类对象时，将首先完成(**自己**)的析构函数定义体的执行，接着完成(类对象)的析构函数定义体的执行，最后完成(**基类**)的析构函数定义体的执行。

7、设 px 是指向一个类动态对象的指针变量，则执行“delete px;”语句时，将自动调用该类的(**析构函数**)。

8、当一个类对象离开它的作用域时，系统将自动调用该类的(**析构函数**)。

9、假定一个类对象数组为 a[n]，当离开它的作用域时，系统自动调用该类析构函数的次数为(**n**)。

10、假定 AB 为一个类，则执行“AB a[10];”语句时，系统自动调用该类构造函数的次数为(**10**)。

11、假定用户没有给一个名为 AB 的类定义构造函数，则系统为其隐含定义的构造函数为(**AB(){}**)。

12、假定用户没有给一个名为 AB 的类定义析构函数，则系统为其隐含定义的析构函数为(**~AB(){}**)。

13、若需要把一个函数“void f();”定义为一个类 AB 的友元函数，则应在类 AB 的定义中加入一条语句：(**friend void f();**)

14、若要把一个类 AB 定义为一个类 CD 的友元类，则应在类 CD 的定义中加入一条语句：(**friend class AB;**)

15、假定类 AB 中有一个静态整型成员 bb，在类外为它进行定义初始化为 0 时，所使用的语句为(**int AB:bb = 0;**)。

16、假定类 AB 中有一个公用属性的静态数据成员 bb，在类外不通过对象名访问该成员 bb 的写法为(**AB::bb**)

17、当类中一个字符指针成员指向具有 n 个字节的存储空间时，它所能存储字符串的最大长度为(**n-1**)

18、假定 AB 这一个类，则该类的拷贝构造函数的声明语句为(**AB(AB &ab)**)。

19、对类对象成员的初始化是通过执行构造函数中的(**初始化**)完成的。

20、对于类中的定义的成员，其隐含访问的权限为(**private**)，对于结构中定义的成员，其隐含访问权限为(**public**)。

21、一个类的友元函数或友元类能够通过成员操作符访问该类的(**私有成员**)。

22、假定要对类 AB 定义加号操作符重载成员函数，实现两个 AB 类对象的加法，并返回相加结果，则该成员函数的声明语句为：(**AB operator + (AB &a)**)。

23、在 C＋＋流类库中，根基类为(**ios**)。

24、在 C＋＋流类库中，输入流类和输出流类的名称分别为(**istream**)和(**ostream**)。

25、若要在程序文件中进行标准输入输出操作，则必须在开始的#include 命令中使用（ **iostream.h** ）头文件。

26、若要在程序文件中进行文件输入输出操作，则必须在开始的#include 命令中使用（ **fstream.h** ）头文件。

27、当从字符文件中读取回车和换行两个字符时，被系统看作为一个（**换行符** ）。

28、当使用 ifstream 流类定义一个流对象并打开一个磁盘文件时，文件的隐含打开方式为(**ios::in**),当使用 ofstream 流类定义一个流对象并打开一个磁盘文件时，文件的隐含打开方式为（ **ios::out** ）。

29、当需要使用 istream 流类定义一个流对象并联系一个字符串时，应在文件开始使用#include 命令，使之包含（ **stream.h** ）。

四、代码分析题

```
class A
{
      public:
            A(int aa = 0):a(aa) {a = aa;}
            ~A(){cout<<"deconstructor A!"<<a<<endl;
      private:
             int a;
};
class B:public A
{
      public:
            B(int aa=0, int bb = 0):A(aa){b=bb;)
            ~B(){cout<<"deconstructor B!"<<b<<endl;}
      private:
             int b;
};

int main()
{
    B x(5);
    B y(6,7);
    system("pause");
    return 0;
}
```

```
class A
{
      public:
            A(int x)
            {
                  a = new int(x);
                  cout<<"constructor!"<<*a<<endl;
            }
            ~A()
            {
                delete a;
                cout<<"deconstructor!"<<endl;
            }
      private:
             int *a;
};

int main()
{
    A x(9), *p;
    p = new A(12);
    delete p;
    system("pause");
    return 0;
}
```

上述两段代码的返回结果是？

```
deconstructor B!7
deconstructor A!6
deconstructor B!0
deconstructor A!5
```

```
constructor!9
constructor!12
deconstructor!
deconstructor!
```

五、编写 strcpy 函数

已知 strcpy 函数的原型是

char *strcpy(char *strDest, const char *strSrc);

其中 strDest 是目的字符串，strSrc 是源字符串。

（1） 不调用 C++/C 的字符串库函数，请编写函数 strcpy

```
char *strcpy(char *strDest, const char *strSrc);
{
    assert((strDest!=NULL) && (strSrc !=NULL));
    char *address = strDest;
    while( (*strDest++ = * strSrc++) != '\0' )
       NULL ;
    return address ;
}
```

（2）strcpy 能把 strSrc 的内容复制到 strDest，为什么还要 char * 类型的返回值？

答：为了实现链式表达式。

例如　　int length = strlen(strcpy(strDest, “hello world”));

编写类 String 的构造函数、析构函数和赋值函数

已知类 String 的原型为：

```
class String
{
  public:
    String(const char *str = NULL); // 普通构造函数
    String(const String &other);        // 拷贝构造函数
    ~ String(void);                     // 析构函数
    String & operate =(const String &other);    // 赋值函数
  private:
    char    *m_data;                // 用于保存字符串
};
```

请编写 String 的上述 4 个函数。

```
// String 的析构函数
String::~String(void)
{
    delete [] m_data;
    // 由于 m_data 是内部数据类型，也可以写成 delete m_data;
}

// String 的普通构造函数
String::String(const char *str)
{
    if(str==NULL)
    {
        m_data = new char[1];    // 若能加 NULL 判断则更好
        *m_data = ‘\0’;
    }
    else
    {
        int length = strlen(str);
        m_data = new char[length+1];  // 若能加 NULL 判断则更好
        strcpy(m_data, str);
    }
}
// 拷贝构造函数
String::String(const String &other)
{
    int length = strlen(other.m_data);
    m_data = new char[length+1];      // 若能加 NULL 判断则更好
```

```
    strcpy(m_data, other.m_data);
}
// 赋值函数
String & String::operate =(const String &other)
{
    // (1) 检查自赋值
    if(this == &other)
        return *this;

    // (2) 释放原有的内存资源
    delete [] m_data;

    // （3）分配新的内存资源，并复制内容
    int length = strlen(other.m_data);
    m_data = new char[length+1];         // 若能加 NULL 判断则更好
    strcpy(m_data, other.m_data);

    // （4）返回本对象的引用
    return *this;
}
```

思考题：

You are the proud owner and the only employee of Auto Repair Inc. When you started the company, you chose not to charge by the hour, but based on the complexity of the repairs. Some repairs are really complex but take very little time. Others may require hours, but are so simple that their list price is low.

At the moment, business is great. There are more broken cars than you could possibly repair. You want to generate as much revenue as possible, so you'll need to decide which cars to repair.

For the purpose of the assignment, each repair consumes only your time - you can, for example, ignore the cost of spare parts.

Example Data
Available hours: *8*

Cars:
ABC-123 3h 100€
DEF-456 5h 120€
GHI-789 4h 80€
ZZZ-999 1h 50€

Result: ABC-123, GHI-789, ZZZ-999