注意：绿色是被挑选出来的题

橙色是要注意的

红色是有疑问的

1. const符号常量：

（1）const char \*p

（2）char const \*p

（3）char \* const p

如果const位于星号的左侧，则const就是用来修饰指针所指向的变量，即指针指向为常量。

如果const位于星号的右侧，const就是修饰指针本身，即指针本身是常量。

1. 析构函数和虚函数的用法和作用？

析构函数的作用是当对象生命期结束时释放对象所占用的资源。析构函数用法：析构函数是特殊的类成员函数

它的名字和类名相同，没有返回值，没有参数不能随意调用也没有重载。只是在类对象生命期结束时有系统自动调用。

虚函数用在继承中，当在派生类中需要重新定义基类的函数时需要在基类中将该函数声明为虚函数，作用为使程序支持动态联遍。

1. 堆和栈的区别

栈（stack）：由编译器自动分配释放，存放函数的参数值，局部变量的值等。其操作方式类似于数据结构中的栈。

堆：一般由程序员分配释放，若不释放，程序结束时可能由OS回收。注意它与数据结构中的堆是两回事，分配方式类似。

1. 头文件的作用是什么？
   1. 通过头文件来调用库功能。在很多场合，源代码不便（或不准）向用户公布，只要向用户提供头文件和二进制的库即可。用户只需要按照头文件中的接口声明来调用库功能，而不必关心接口怎么实现的。编译器会从库中提取出相应的代码。
   2. 头文件能加强类型安全检查。如果某个接口被实现或被使用时，其方式与头文件中的声明不一致，编译器就会指出错误，这一简单的规则能大大减轻程序员调试、改错的负担。
2. 内存的分配方式有几种？
   1. 从静态存储区域分配。内存在程序编译的时候已经分配好，这块内存在程序的整个运行期间都存在。如全局变量。
   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。程序在运行的时候用malloc或new申请任意多少的内存，程序员自己负责在何时用free或delete释放内存。动态内存的生存期由我们决定，使用非常灵活。
3. BOOL、float、指针变量与”零值比较的if语句”。

BOOL:If(flag)或if(!flag)

Float:const float EPSINON = 0.00001;

If((x>=.EPSINON)&&(x<=EPSINON)) 说明：不可将浮点变量用”＝＝”或”！＝”与数字比较，应该设法转化成”>=”或”<=”此类形式。

指针\*p：if(p==NULL) if(p!=NULL)

1. 以下为Windows NT下的32位C++程序，请计算sizeof的值

Char str[] = “Hello”;

Char \*p = str;

Int n = 10

请计算：

Sizeof(str) = 6

Sizeof(p) = 4

Sizeof(n)=2

Void Func(char str[100])

{

    请计算：sizeof(str) = 4

}

Void \*p = malloc(100)

请计算：sizeof(p) = 4

1. #include <filename.h>和#include “filename.h”有什么区别

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中开始搜索filename.h

1. Const有什么用途
   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。
2. 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”?

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 fee(int x,inty);

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”来解决名字匹配的问题。

1. 内存思考题

Void GetMemory(char \*p)

{

        P = (char \*)malloc(100);

}

Void Test(void)

{

        Char \*str = NULL;

        GetMemory(str);

        Strcpy(str,”hello world”);

        Printf(str);

}//函数内的变量是独立于main的，对其改变不会影响main的变量

请问运行Test函数会有什么样的结果？

程序会崩溃，因为GetMemory并不能传递动态内存，Test函数中的str一直是NULL。

Strycpy(str,”hello world”);将使程序崩溃。

Char \*GetMemory(void)

{

        Char p[] = “hello world”;

        Return p;

}

Void Test(void)

{

        Char \*str = NULL;

        Str = GetMemory();

        Printf(str);

}

请问Test函数会有什么样的结果？

可以是乱码。因为GetMemory返回的是指向”栈内存”的指针，该指针的地址不是NULL，但其原现的内容已经被清除，新内容不知。

Void GetMemory2(char \*\*p,int num)

{

        \*P = (char \*)malloc(num);

}

Void Test(void)

{

        Char \*str = NULL;

        GetMemory(&str,100);

        Strcpy(str,”hello”);

        Printf(str);

}

请问运行Test函数会有什么样的结果？

* 1. 能够输出hello
  2. 内存泄露

Void Test(void)

{

        Char \*str = (char \*) malloc(100);

        Strcpy(str,”hello”);

        Free(str);

        If(str != NULL)

        {

            Strcpy((str,”world”);

            Printf(str);

}

}

请问运行Test函数会有什么样的结果？

篡改动态内存区的内容，后果难以预料，非常危险。

因为free(str);之后，str成为野指针，if(str!=NULL)语句不起作用。

1. 头文件的作用是什么？
   1. 通过头文件来调用库功能。在很多场合，源代码不便（不准）向用户公布，只要向用户提供头文件和二进制的库即可。用户只需要按照头文件中的接口声明来调用库功能，而不必关心接口怎么实现的。编译器会从库中提取相应的代码。
   2. 头文件能加强类型安全检查。如果某个接口被实现或被使用时，其方式与头文件中的声明不一致，编译器就会指出错误，这一简单的规则能减轻程序员调试、改错的负担。
2. C++里面晃是不所有的动作都是main()引起的？如果不是，请举例

在运行C++程序时，通常从main()函数开始执行。因此如果没有main()，程序将不完整，编译器将指出未定义main()函数。

例外情况：**如，在windows编程中，可以编写一个动态 连接库（DLL）模块，这是其他windows程序可以使用的代码。由于DLL模块不是独立的程序，因此不需要main()。用于专用环境的程序――如机器人中的控制器芯片――可能不需要main()，但常规的独立程序都需要main().**

1. 引用与指针的区别：
   1. 引用总是指向某个对象，定义时没有初始化是错误的；
   2. 给引用赋值是修改引用所关联的对象的值，所关联的对象不变。
2. 变量的声明和定义有什么区别

从编译原理上来说，声明是仅仅告诉编译器，有个某类型的变量会被使用，但是编译器并不会为它分配任何内存。而定义就是分配了内存。

1. Sizeof和strlen区别

1、Sizeof操作符的结果类型是size\_t，它在头文件中typedef为unsigned int类型。该类型保证能容纳实现所建立的最大对象的字节大小。

2、Sizeof是运算符，strlen是函数

3、Sizeof可以用类型做参数，strlen只能用char\*做参数，且必须是以换行符”\0″结尾的。

4、Strlen的结果要在运行的时候才能计算出来，是用来计算字符串的长度，不是类型占内存的大小。

5、数组做sizeof的参数不退化，传递给strlen就退化为指针；

1. C中malloc与new的区别
   1. new是C++中的操作符，malloc是C中的一个函数；
   2. new不上是分配内存，而且会调用类的构造函数，同理delete会调用类析构函数，而malloc则只分配内存，不会进行初始化类成员的工作，同样free也不会调用析构函数。
   3. 内存泄露对于malloc或者new都可以检查出来的，区别在于new可以指明那个文件的那一行，而malloc没有这些信息。
   4. New和malloc效率比较

New有三个字母，malloc有六个字母

New可以认为是malloc加构造函数的执行。

New出来的指针是直接带类型信息。

而malloc返回的都是void指针。

1. 关键字static在C和C++中的区别
   1. 在C语言中，主要体现在静态全局变量、静态局部变量和静态函数。
   2. 在C++中，主要体现在静态数据成员和静态成员函数。
2. 简述#define #endif 和#ifndef的作用

#define指示接受一个名字并定义该名字为预处理器变量；

#ifndef检测指定的预处理变量是否定义；

#endif预处理未定义

1. 实现双向链表删除一个节点P，在节点P后插入一个节点，写出这两个函数;

答：双向链表删除一个节点P

Template<class type> void list<type>::delnode(int p)

{

        int k = 1;

        listnode<type> \*ptr,\*t;

        ptr = first;

        While(ptr->next!=NULL&&k!=p)

        {

            ptr = ptr->next;

            k++;

}

t = ptr->next;

cout <<”你已经将数据项”<<t->data<<”删除”<<endl;

ptr->next = ptr->next->next;

length–;

delte t;

}

在节点P后插入一个节点：

Template<class type> bool list<type>::insert(type t,int p)

{

        Listnode<type> \*ptr;

        Ptr = first;

        int k = 1;

        while(ptr != NULL && k < p)

{

    ptr = ptr->next;

    k++;

}

If(ptr == NULL && k!=p)

{

    return false;

}

else

{

    Listnode<type> \*tp;

    tp = new listnode<type>;

    tp->data = t;

    tp->next = ptr->next;

    ptr->next = tp;

    length++;

    return true;

}

}

1. sizeof的使用场合

sizeof操作符一个主要用途是存储分配和I/O系统那样的例程进行通信。可以查看某种类型的对象在内存中所占单位字节；

在动态分配一对象时，可以让系统指导要分配多少内存；

便于一些类型的扩充。

1. 以下代码有什么问题[C++]

struct Test

{

        Test(int){};

        Test(){};

        void fun(){};

};

void main(void)

{

        Test a(1);

a.fun();

        Test b();

        b.fun();

}

Test b();//定义了一个函数 b不是一个类对象。

1. 以下代码有什么问题？

Cout <<(true?1:”I”<<endl;

1：”1″不是同一类型

:前后必须是同一类型or能隐式转换的类型

1. 以下三条输出语句分别输出什么？

char str1[] = “abc”;

char str2[] = “abc”;

const char str3[] = “abc”;

const char str4[]] = “abc”;

const char\* str5 = “abc”;

const char\* str6 = “abc”;

cout << boolalpha <<(str==str2)<<endl;

cout <<boollalpha<<(str3==str4)<<endl;

cout <<boollalpha<<(str5==str6)<<endl;

分别输出false,false,true。Str1和str2都是字符数组，每个都有其自己的存储区，它们的值则是各存储区的首地址，不符；str3和str4同上，只是按const语义，它们所指向的数组区不能修改。Str5和str6并非数组而是字符指针，并不分配存储区，其后”abc”以常量形式存于静态数据区，而它们自己仅是指向该区首地址的指针，相符。

1. C++中的空类，默认产生哪些类成员函数？

class Empty

{

        public:

            Empty();//缺省构造函数

            Empty(const Empty&);//拷贝构造函数

            ~Empty();//析构函数

            Empty&operator=(const Empty&);//赋值运算符

            Empty\* operator&();    //取址运算符

            Const Empty\* operator&() const;//取址运算符 const

};

1. 以下代码能够编译通过吗，为什么？

unsigned int const size1 = 2;

char str1[size1];

unsigned int temp = 0;

cin >>temp;

unsigned int const size2 = temp;

char str2[size2];

str2定义出错，size2非编译器期间常量，而数组定义要求长度必须编译期常量。

1. 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。

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

{

        assert((strDest!=NULL)&&(strSrc!=NULL));

        char \*address = strDest;

        while((\*strDest ++=\*strSrc++)!=’\0′)

{

    NULL;

}

return address;

}

strycpy能把strSrc的内容复制到strDest，为什么还要char\*类型的返回值？

为了实现链式表达式。

如：int length = strlen(strcpy(strDest,”hello world”));

1. 写出下面这段代码的输出

#include <iostream>

Int main(void)

{

        int a,b,c,d;

        a = 10;

        b = a++;

        c = ++a;

        d = 10 \* a++;

        printf(“b,c,d:%d,%d,%D”,a,b,c);

        return 0;

}

10,12,120;

1. 编写类String的构造函数、析构函数和赋值函数

class String

{

        public:

            String(Const char \*str = NULL);//普通构造函数

            String(const String &other);    //拷由构造函数

            ~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

};

//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];

            \*m\_data = ‘\0′;

}

else

{

    int length = strlen(str);

    m\_data = new char[length + 1];

    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];

        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];

        strcpy(m\_data,other.m\_data);

        return \*this;

}

1. 全局变量可不可以定义在可被多个.C文件包含的头文件中？为什么？

可以，在不同的C文件中以static形式来声明同名全局变更。可以在不同的C文件夹中声明同名的全局变量，前提是其中只能有一个C文件中对此变量赋初值，此时连接不会出错。

1. 对于一个频繁使用的短小函数，在C语言中应用什么实现，在C++中应用什么实现？

C用宏定义，C++用inline

1. 指出下面代码的输出，并解释为什么

main()

{

        int a[5] = {1,2,3,4,5};

        int \*ptr = (int \*)(&a+1);

        printf(“%d,%d”,\*(a+1),\*(ptr-1));

}

输出：2，5

\*(a+1)就是a[1]，\*(ptr-1)就是a[4]，执行结果是2，5；

&a+1不是首地址+1，系统会认为加一个a数组的偏移，是偏移了一个数组的大小（本例是5个int）

int \*ptr = (int \*)(&a+1);则ptr实际是&(a[5])，也就是a+5

原因如下：

&a是数组指针，其类型为int(\*)[5];

而指针加1要根据指针类型加一定的值，不同类型的指针+1之后增加的大小不同，a是长度为5的int数组指针，所以要加5\*sizeof(int)

所以ptr实际是a[5]

但是ptr与(&a+1)类型是不一样的（这点很重要）

    所以ptr-1只会减去sizeof(int\*)

    a，&a的地址是一样的，但意思不一样，a是数组首地址，也就是a[0]的地址，&a是对象（数组）首地址，a+1是数组下一元素的地址，即a[1]，&a+1是下一个对象的地址，即a[5].

1. 以下代码中的两个sizeof用法有问题吗？

sizeof如用于数组，只能测出静态数组的大小，无法检测动态分配的或外部数组大小。注意：数组名作为函数参数时，退化为指针。数组名作为sizeof()参数时，数组名不退化，因为sizeof不是函数。

1. 请问以下代码有什么问题

int main()

{

        char a;

        char \*str = &a;

        strcpy(str,”hello”);

        printf(str);

        return 0;

}

没有为str分配内存空间，将会发生异常

问题出在将一个字符串复制进一个字符变量指针所指地址。虽然可以正确输出结果，但因为越界进行内在读写而导致程序崩溃。

Strcpy的在库函数string.h中，程序的主要程序在于越办进行内存读写导致程序崩溃。

2.

const char\* s = “AAA”;

Printf(“%s”,s);

S[0] = ‘B’;

Printf(“%s”,s);

“AAA”是字符串常量，S是指针，指向这个字符串常量，所以声明s的时候就有问题。

Const char\* s =”AAA”,然后又因为是常量，所以对S[0]的赋值操作是不合法的。

1. Char szstr[10];

Strcpy(szstr,”0123456789″);

产生什么结果？为什么？

正常输出，长度不一样，会造成非法的OS，覆盖别的内容

1. 交换两个变量的值，不使用第三个变量。即a=3,b=5，交换之后a=5,b=3;

两种解法，一种是用算术算法，一种是用^（异或）

a = a+b;

b = a-b;

a = a-b;

or

a = a^b;//只能对int ,char

b = a^b;

a = a^b;

or

a ^= b ^=a;

1. int a = 256;

char d =a;

pint(“%d”,d+1);

1

Char类型的变量赋值范围是0~255.当把256赋值给a时，超出了a的有效取值范围。此时a的实际值为0.

1. 将”引用”作为函数返回值类型的格式、好处和需要遵守的规则

格式：类型标识符 &函数名（形参列表及类型说明）{//函数体}

格式：在内存中不产生被返回值的副本：（注意：正是因为这点原因，所以返回一个局部变量的引用是不可取的。因为随着该局部变量生存期的结束，相应的引用也会失效。）

注意事项：

* 1. 不能返回局部变量的引用。主要原因是局部变量会在函数返回时被销毁，因此被返回的引用就成为了”无所指的”引用，程序会进入未知状态。
  2. 不能返回函数内部new分配的内存的引用。虽然不存在局部变量的被动销毁问题，可对于这种情况（返回函数内部new分配内存的引用），又面临其它尴尬的局面。如，被函数返回的引用只是作为一个临时变量出现，而没有被赋予一个实际的变量，那么这个引用所指向的空间（由new分配）就无法释放。
  3. 可以返回类成员的引用，但最好是const。主要原因是当对象的属性是与某种业务规则相关联时，其赋值常常与某些其它属性或者对象的状态有关，因此有必要将赋值操作封装在一个业务规则当中。如果其它对象可以获得该属性的非常量引用（或指针），那么对该属性的单纯赋值就会破坏业务规则的完整性。

1. 多态的作用：
   1. 隐藏实现细节，使得代码能够模块化；扩展代码模块，实现代码重用；
   2. 接口重用：为了类在继承和派生的时候，保证使用家庭中任一类的实例的某一属性时的正确调用。

