1、设有矩阵A1（30\*35）、A2（35\*15）、A3（15\*5）、A4（5\*10），M=A1\*A2\*A3\*A4，下列组合计算M所需数乘次数最少的是：  
A、(A1(A2(A3A4)))    B、(A1((A2A3)A4))    C、((A1A2)(A3A4))    D、((A1(A2A3))A4)   E、(((A1A2)A3)A4)  
2、在32位机器上，有如下代码：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8476976)

1. **char** array[] = "abcdefg";
2. printf("%d\n",sizeof(array));        // 8
4. **char** \*p = "abcdefg";
5. printf("%d\n",sizeof(p));            // 4
7. void func(**char** p[10])
8. {
9. printf("%d\n",sizeof(p));        // 4
10. }
12. void func(**char** (&p)[10])
13. {
14. printf("%d\n",sizeof(p));        // 10
15. }
17. **int** main(void)
18. {
19. printf("%d\n",sizeof(**char**[2]));     // 2
20. printf("%d\n",sizeof(**char**&));       // 1
21. return 0;
22. }

char array[] = "abcdefg";

printf("%d\n",sizeof(array)); // 8

char \*p = "abcdefg";

printf("%d\n",sizeof(p)); // 4

void func(char p[10])

{

printf("%d\n",sizeof(p)); // 4

}

void func(char (&p)[10])

{

printf("%d\n",sizeof(p)); // 10

}

int main(void)

{

printf("%d\n",sizeof(char[2])); // 2

printf("%d\n",sizeof(char&)); // 1

return 0;

}

其输出结果依次为：  
A、8   4   4  10  2  1  
B、4   4   4  10  4  1  
C、8   4   4  10  2  4  
D、8   4   4  4  2  1  
E、8   4   4  10  4  4  
F、以上答案都不对  
3、

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8476976)

1. CONTAINER::iterator iter , tempIt;
2. for(iter = cont.begin() ; iter != cont.end() ; )
3. {
4. tempIt = iter;
5. ++iter;
6. cont.erase(tempIt);
7. }

CONTAINER::iterator iter , tempIt;

for(iter = cont.begin() ; iter != cont.end() ; )

{

tempIt = iter;

++iter;

cont.erase(tempIt);

}

假设cont是一个CONTAINER的示例，里面包含数个元素，那么当CONTAINER为：  
1、vector<int>  
2、list<int>  
3、map<int , int>  
4、deque<int>  
会导致上面的代码片段崩溃的CONTAINER类型是：  
A、1，4     B、2，3    C、1，3      D、2，4  
正确答案选择A（第1个、第4个都是线性的类型存储，所以会存在崩溃）  
4、以下代码

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8476976)

1. class classA
2. {
3. public:
4. classA()
5. {
6. clear();
7. }
8. virtual ~classA()
9. {
10. }
11. void clear()
12. {
13. memset(this , 0 , sizeof(\*this));
14. }
15. virtual void func()
16. {
17. printf("func\n");
18. }
19. };
20. class classB : public classA
21. {
22. };
24. **int** main(void)
25. {
26. classA oa;
27. classB ob;
28. classA \* pa0 = &oa;
29. classA \* pa1 = &ob;
30. classB \* pb = &ob;
32. oa.func();         // 1
33. ob.func();         // 2
34. pa0->func();       // 3
35. pa1->func();       // 4
36. pb->func();        // 5
38. return 0;
39. }

class classA

{

public:

classA()

{

clear();

}

virtual ~classA()

{

}

void clear()

{

memset(this , 0 , sizeof(\*this));

}

virtual void func()

{

printf("func\n");

}

};

class classB : public classA

{

};

int main(void)

{

classA oa;

classB ob;

classA \* pa0 = &oa;

classA \* pa1 = &ob;

classB \* pb = &ob;

oa.func(); // 1

ob.func(); // 2

pa0->func(); // 3

pa1->func(); // 4

pb->func(); // 5

return 0;

}

A、func              func        执行出错      执行出错        func    
B、执行出错         func        执行出错      执行出错        func   
C、执行出错      执行出错      执行出错      执行出错      执行出错  
D、func              func          func            func          func  
E、func              func        执行出错          func          func  
F、以上选项都不对  
5、在32位系统中

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8476976)

1. class CBase
2. {
3. public:
4. void func()
5. {
6. Print();
7. }
8. virtual void Print()
9. {
10. cout<<"CBase::Print"<<endl;
11. }
12. };
13. class CDerived : public CBase
14. {
15. public:
16. virtual void Print()
17. {
18. cout<<"CDerived::Print"<<endl;
19. }
20. };
22. **int** main(void)
23. {
24. CDerived c;
25. CBase \*p = &c;
26. return 0;
27. }

class CBase

{

public:

void func()

{

Print();

}

virtual void Print()

{

cout<<"CBase::Print"<<endl;

}

};

class CDerived : public CBase

{

public:

virtual void Print()

{

cout<<"CDerived::Print"<<endl;

}

};

int main(void)

{

CDerived c;

CBase \*p = &c;

return 0;

}

请问：  
sizeof(\*p)的值是多少？ （实际上求的就是一个成员变量的大小+一个虚表指针VPTR的大小）  
A、1    B、4     C、8    D、12  
p->Print(); 和 c.func();的输出分别是？  
A、CBase::Print     CBase::Print     B、CBase::Print     CDerived::Print  
C、CDerived::Print  CBase::Print    D、CDerived::Print  CDerived::Print  
6、

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8476976)

1. struct Thing
2. {
3. **int** valueA;
4. const **int** valueB;
5. };
6. Thing t;

struct Thing

{

int valueA;

const int valueB;

};

Thing t;

t的成员变量valueA 和 valueB 的值分别为：  
A、0  0     B、垃圾值  0     C、无法运行     D、垃圾值  垃圾值  
常成员变量valueB必须在构造函数中进行初始化  
7、for(int x = 0 , y = 0; !x && y <= 5 ; y++)语句执行循环的次数是：  
A、0        B、5        C、6        D、无数次  
8、在Windows 32位操作系统中，假设字节对齐为4，对于一个空的类A，sizeof(A)的值为（）  
A、0         B、1         C、2         D、4  
9、以下对函数指针的定义，哪些是正确的：  
A、typedef  int  (\*fun\_ptr)(int , int);       B、typedef  int  \*(fun\_ptr)(int , int);   
C、typedef  int  (fun\_ptr\*)(int , int);       D、typedef  \*int  (fun\_ptr)(int , int);   
10、在32位系统中，下面结构体

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8476976)

1. struct st
2. {
3. **char** ch , \*ptr;
4. union
5. {
6. **short** a , b;
7. unsigned **int** c : 2 , d : 1;
8. };
9. **bool** f;
10. struct st \*next;
11. };

struct st

{

char ch , \*ptr;

union

{

short a , b;

unsigned int c : 2 , d : 1;

};

bool f;

struct st \*next;

};

的大小是：  
A、14字节       B、16字节       C、20字节       D、24字节  
11、32位小端字节序的机器上，如下代码：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8476976)

1. **char** array[12] = {0x01 , 0x02 , 0x03 , 0x04 , 0x05 , 0x06 , 0x07 , 0x08};
2. **short** \*pshort = (**short** \*)array;
3. **int** \*pint = (**int** \*)array;
4. int64 \*pint64 = (int64 \*)array;
5. printf("0x%x , 0x%x , 0x%x , 0x%x", \*pshort , \*(pshort+2) , \*pint64 , \*(pint+2));

char array[12] = {0x01 , 0x02 , 0x03 , 0x04 , 0x05 , 0x06 , 0x07 , 0x08};

short \*pshort = (short \*)array;

int \*pint = (int \*)array;

int64 \*pint64 = (int64 \*)array;

printf("0x%x , 0x%x , 0x%x , 0x%x", \*pshort , \*(pshort+2) , \*pint64 , \*(pint+2));

输出结果为：  
A、0x201 , 0x403 , 0x807060504030201 , 0x0    B、0x201 , 0x605 , 0x807060504030201 , 0x0     
C、0x201 , 0x605 , 0x4030201 , 0x8070605       D、0x102 , 0x506 , 0x102030405060708 , 0x0    
E、0x102 , 0x304 , 0x1020304 , 0x5060708          F、0x201 , 0x605 , 0x4030201 , 0x6050403  
12、有关引用，下列说法错误的是：  
A、引用定义时必须初始化  
B、类的非静态引用成员变量不需要在构造函数中初始化  
C、数组可以有引用，但没有引用数组  
D、public派生对象可以初始化基类类型的引用  
E、整型数据的常引用可以直接用数值初始化  
F、以上选项都不是  
13、Windows PE文件装载到的地址为：  
A、0x0030000    B、0x0040000    C、任意地址   D、0x10000000      
14、以下哪些对象可用于Windows进程间通信：  
A、事件    B、临界区    C、互斥量    D、共享内存