C++面试题

1、C++为什么要提出引用？（引用和指针的区别？）

引用变量在功能上等于一个常量指针（到底是常量指针还是指针常量？有的书上写的常量指针，有的博客里写的指针常量，那就不纠结这个问题吧。总之这里指的是顶层 const），即一旦指向某一个单元就不能在指向别处。

（1）我们在用指针的使用经常犯的错误是什么？

①操作野指针；

②不知不觉改变了指针的值，而后还以为该指针正常。

如果我们要正确的使用指针，我们不得不人为地保证这两个条件。而引用的提出就是解决这个问题。

引用区别于指针的特性是：

① 必须初始化（保证不是野指针）；

② 一个引用永远指向他初始化的那个对象（保证指针值不变）。

所以引用的提出就是：让人为地保证这两个条件变成让编译器保证。这样可以减少错误的产生。

（2）程序中使用指针，程序的可读性差；而引用本身就是目标变量的别名，对引用的操作就是对目标变量的操作。

（3）用引用传递函数的参数，能保证参数传递中不产生副本，提高传递的效率，且通过 const 的使用，保证了引用传递的安全性。

2、使用 const 而不是#define 来定义常量？

①const 常量有数据类型，而宏常量没有数据类型。编译器可以对前者进行类型安全检查，而后者仅仅进行字符替换，没有类型安全检查，并且在字符替换过程中会产生一些意料不到的错误。

②const 方法可以很方便地用于复合类型。

如 const int a[3]={1,2,3};

③const 标识符遵循变量的作用域原则，可以创建作用域为全局、名称空间、数据块的常量。

3、C++左值和右值的区别？

左值：lvalue（locator value）代表一个在内存中占有确定位置的对象（换句话说就是有一个地址）。

右值：rvalue 通过排他性来定义，每个表达式不是 lvalue 就是 rvalue。因此从上面的 lvalue 的定义，rvalue 是在不在内存中占有确定位置的表达式。

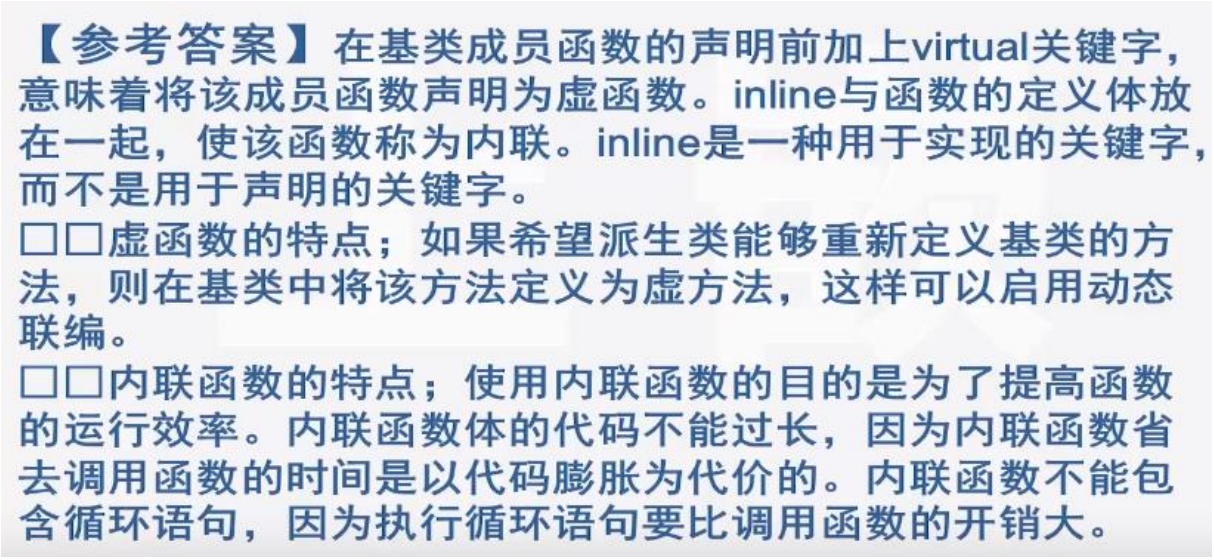
4、C++是不是类型安全的？

类型安全：不同类型的数据不能随意的转化来进行操作，java 作为第二代面向对象高级语言，即是类型安全的，C 和 C++便不是。

例如：就 C++而言，我们可以把 0 作为 false，非零作为 true。一个函数就算是 bool 类型的，我们还是可以返回 int 类型，并且自动把 0 转化成 false，非零转化成 true。

相比之下 java 不能把 int 类型转化成 bool 类型。

5、C++中 virtual 和 inline 的含义分别是？



virtual 的作用主要有两点：

1、为了实现多态

2、为了构建纯虚函数（抽象类）

6、C++中 Const 关键字的作用？（只读）

【参考答案】const关键字至少有下列n个作用：

（1）欲阻止一个变量被改变，可以使用const关键字。在定义该const变量时，通常需要对它进行初始化，因为以后就没有机会再去改变它了；

（2）对指针来说，可以指定指针本身为const，也可以指定指针所指的数据为const，或二者同时指定为const；

（3）在一个函数声明中，const可以修饰形参，表明它是一个输入参数，在函数内部不能改变其值；

（4）对于类的成员函数，若指定其为const类型，则表明其是一个常函数，不能修改类的成员变量；

（5）对于类的成员函数，有时候必须指定其返回值为const类型，以使得其返回值不为“左值”。

7、VC 中，编译工具条内的 Debug 与 Release 选项是什么含义？

【参考答案】Debug 通常称为调试版本，它包含调试信息，并且不作任何优化，便于程序员调试程序。Release 称为发布版本，它往往是进行了各种优化，使得程序在代码大小和运行速度上都是最优的，以便用户很好地使用。Debug带有大量的调试代码，运行时需要相应的运行库，发布模式程序紧凑不含有调试代码和信息，直接可以运行（如果不需要运行库）

8、C++中的静态局部变量？

（1）静态局部变量在静态存储区内分配存储单元。在程序运行期间都不释放。

（2）对静态局部变量是在编译时赋初值的，即只赋值一次，在程序运行时它已有初值，以后每次调用函数时不再重新赋初值而只是保留上一次函数调用结束时的值。

（3）静态局部变量，如果定义的时候没有赋初值，编译时自动赋初值 0 或者空字符。

（4）静态局部变量在函数调用结束后仍然存在，但其他函数不能引用它，在其他函数中，它是"不可见"的。

9、C++中，sizeof(void)返回值是什么？为什么编译器不让他为 0？

答：返回值为 1，若一个类 A 中没有任何变量和成员函数，sizeof(A)返回值也为 0，因为声明它时，需要一个占位符，而 C++中最小内存单元为 1 个字节。

10、C++中函数模板和类模板的区别？

函数模板：C++中函数模板即函数的形参类型和返回值类型都可以自己指定，且函数模板的实例化是由编译程序在函数调用时自动完成。

类模板：C++中类模板即是类的成员函数和成员变量都可以自己指定（不止有函数，还有成员变量），且类模板的实例化必须由程序员显示的实现。

11、C++，重载函数调用依据是什么？

答：依据主要是函数名+参数列表（类型和长度），首先会匹配参数列表绝对匹配的那

个函数，即参数长度和类型都相同；若绝对匹配不成功，则进行数据转换，即参数列表长度相同，类型不同，如形参定义为 double 类型，实际传参为 int 类型，int 类型能够转换为 double。

12、C++，基类的析构函数若不为虚函数，会带来什么问题？

答：如下情况，delete a 时调用的是父类 A 的析构函数，会导致数据残留。

```
class B:class A{};
A &a = new B(); #  父类 A 创建一个引用指向子类 B
delete a;
```

13、C++，深拷贝，浅拷贝，和拷贝构造函数相关问题？

深拷贝：即拷贝了指针也拷贝了资源

浅拷贝：只拷贝了指针，两个指针指向同一片资源

注：拷贝构造函数默认是浅拷贝，拷贝构造仅仅只是完成了值拷贝，导致两个指针指向了同一块内存区域。随着程序的运行结束，又去调用析构函数，先是 s2 去调用析构函数，释放了它指向的内存区域，接着 s1 又去调用析构函数，这时候析构函数企图释放一块已经被释放的内存区域，程序将会崩溃，所以引入了深拷贝。

14、C++中，i++与++i，左值与右值？

1、首先说左值和右值的定义：

变量和文字常量都有存储区，并且有相关的类型。区别在于变量是可寻址的（addressable）对于每一个变量都有两个值与其相联：

1).它的数据值，存储在某个内存地址中。有时这个值也被称为对象的右值（rvalue,读做 are-value）.我们也可认为右值的意思是被读取的值（read value）。文字常量和变量都可被用作右值。

2).它的地址值——即存储数据值的那块内存的地址。它有时被称为变量的左值。能作为左值的必须满足两个条件，其一是能寻到其空间地址，其二是对该空间具有写权限。

2、a++的意思是先复制一份临时数据出来参与周边环境的运算，再自加变量 a，可见 a++用来参与运算的是一份复制出来的临时数据，这个数据是临时存在而没有固定地址的，不是一个真正的变量。++a 的意思是先自加变量 a，再将变量放到周边环境参与运算，那么++a 用来参与运算的是有具体地址的变量，所以++a 是可以作为左值使用的。

1）a++返回一个临时变量
2）++a 返回变量的引用

15、C++中，三元表达符？

注：C++中三元表达符，":"前后必须是相同的数据类型。

15、C++中，所有运算符都能重载吗？

【参考答案】不能被重载的运算符

在 C++运算符集合中，有一些运算符是不允许被重载的。这种限制是出于安全方面的考虑，可防止错误和混乱。

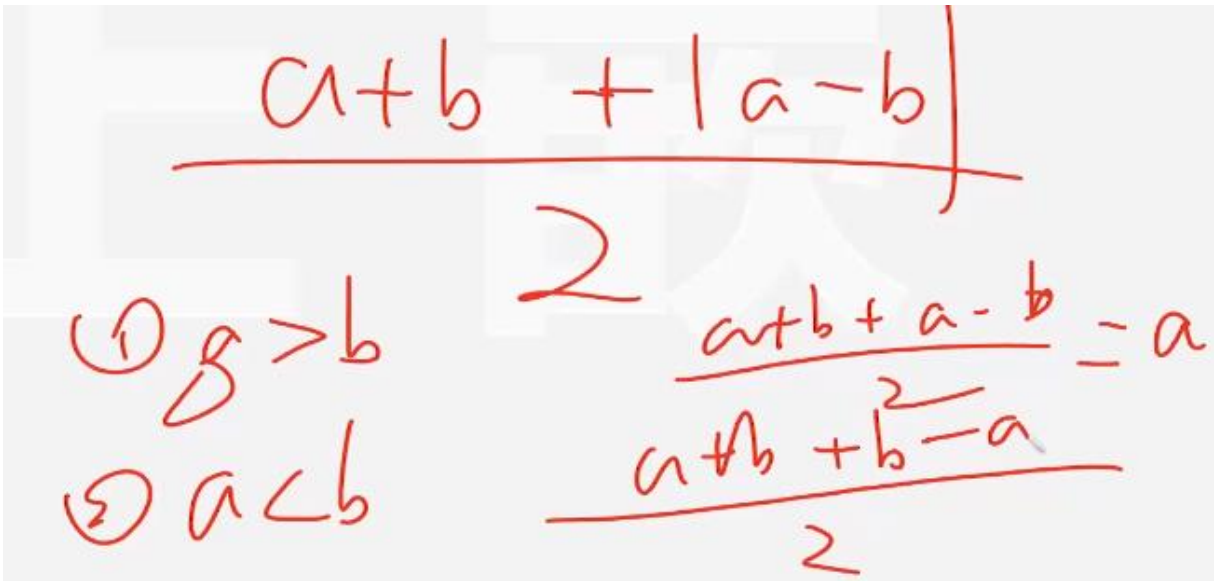
（1）不能改变 C++内部数据类型（如 int,float 等）的运算符。

（2）不能重载‘.’，因为‘.’在类中对任何成员都有意义，已经成为标准用法。

（3）不能重载目前 C++运算符集合中没有的符号，如#,@,$等。原因有两点，一是难以理解，二是难以确定优先级。

（4）对已经存在的运算符进行重载时，不能改变优先级规则，否则将引起混乱。

16、不使用任何判断语句，找出 a，b 中的最大值？



17、C++的空类，默认产生哪些成员函数？

```
class Empty
{
  public:
    Empty(); //缺省构造函数
    Empty（const Empty& ）; //拷贝构造函数
    ~Empty(); //虚构函数
    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
}
```

18、C++中声明一个 static 成员变量有什么用？

【参考答案】在C++类的成员变量被声明为static（称为静态成员变量），意味着它为该类的所有实例所共享，也就是说当某个类的实例修改了该静态成员变量，也就是说不管创建多少对象，static修饰的变量只占有一块内存。其修改值为该类的其它所有实例所见；而类的静态成员函数也只能访问静态成员（变量或函数）。static是加了访问控制的全局变量，不被继承。

注：类的静态成员（函数）只能通过类名来访问，而静态成员函数只能访问静态成员（变量或函数）

19、模板类和类模板？

区别：类模板是 C++中的一种机制，主要是为了实现泛型编程，而模板类则是一种数据类型，它正是通过类模板的机制实现的。

C++中模板类的作用：（主要是复用性）

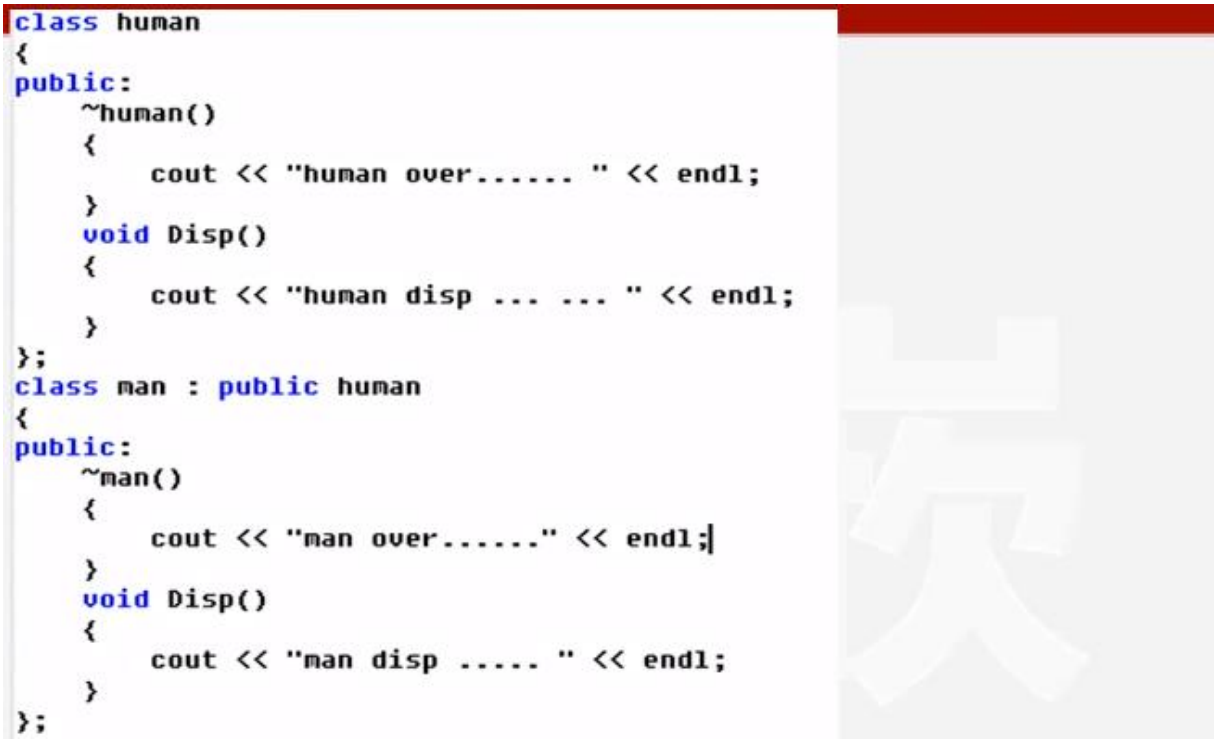
(1) 可用来创建动态增长和减小的数据结构
(2)它是类型无关的，因此具有很高的可复用性。
(3)它在编译时而不是运行时检查数据类型，保证了类型安全
(4)它是平台无关的，可移植性
(5)可用于基本数据类型

20、C++中哪些函数不能声明为虚函数？

另一种问法：C++中哪些函数不能被继承？

普通函数（非成员函数），构造函数，内联成员函数、静态成员函数、友元函数。
（1）虚函数用于基类和派生类，普通函数所以不能
（2）构造函数不能是因为虚函数采用的是虚调用的方法，允许在只知道部分信息的情况的工作机制，特别允许调用只知道接口而不知道对象的准确类型的方法，但是调用构造函数即使要创建一个对象，那势必要知道对象的准确类型。
（3）内联成员函数的实质是在调用的地方直接将代码扩展开
（4）继承时，静态成员函数是不能被继承的，它只属于一个类，因为也不存在动态联编等
（5）友元函数不是类的成员函数，因此也不能被继承

21、C++中虚函数（成员函数和析构函数）？

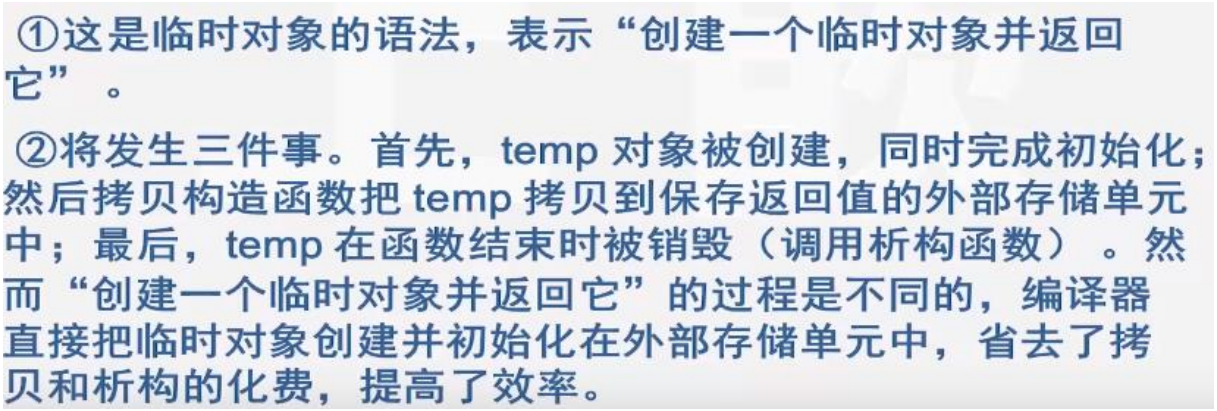


```
class human
{
public:
    ~human()
    {
        cout << "human over...... " << endl;
    }
    void Disp()
    {
        cout << "human disp ... ... " << endl;
    }
};
class man : public human
{
public:
    ~man()
    {
        cout << "man over......" << endl;
    }
    void Disp()
    {
        cout << "man disp ..... " << endl;
    }
};
```

#类的成员函数，默认定义都是为非虚函数

```
int main(){
    human *p = new man();
    p->Disp(); #  调用父类 human 的 Disp()函数
    delete p;   #  程序会报错，调用父类的析构函数，释放空间不完全
    return 0;
}
```

22、匿名对象，return String(s1+s2) 和 String temp(s1+s2);return temp;一样吗？



23、C++异常处理？

24、C++全局变量？

特点：

□ 作用域：全局可见。

全局变量（外部变量）是在函数外部定义的，它的作用域为从变量的定义处开始，到本程序文件的末尾。

注：通常把超出一个函数的作用域称为全局作用域，其他几种（如块作用域）不超出一个函数的作用域称为局部作用域。

□ 存储空间：静态存储区

系统会在执行时将全局变量分配在静态存储区，在程序执行期间，对应的存储空

间不会释放，一直到程序结束才会释放。

注：一个程序在内存中占用的存储空间可以分为 3 个部分：程序区（存放可执行程序的代码）、静态存储区（存放静态变量）、动态存储区（存放动态变量）。

□ 优先度：全局变量优先度低于局部变量

当全局变量和局部变量重名时，会屏蔽全局变量，局部优先。

优点：使用全局变量程序运行时速度会快一点，因为内存不需要再分配。

缺点：使用全局变量会占用更多的内存，因为其生命期长。

25、C++static 关键字？

特点：用来控制存储方式和可见性

① 存储空间：静态存储区（控制变量的存储方式）

静态变量存储在静态存储区（存储在静态存储区的变量，如果不显式地对其进行初始化，系统会将其初始化为 0），在程序执行期间，对应的存储空间不会释放，一直到程序结束才会释放。

static 控制变量的存储方式体现在局部变量上。局部变量存储在动态存储区，在局部变量定义前加上 static，该局部变量就变成了局部静态变量，局部静态变量存储在静态存储区，即使函数调用结束，它占用的存储空间也不会释放，但其他函数不能引用该局部静态变量。当下一次调用该函数时，不会再次对该局部静态变量进行初始化，它的值是上一次函数调用结束时的值。

对全局变量而言，存储方式没有什么改变，因为全局变量和全局静态变量都存储在静态存储区。

② 作用域：（控制变量、函数的可见性）

static 控制变量的可见性体现在全局静态变量和静态函数上。

全局变量默认具有外部链接性，作用域是整个工程。使用 static 关键字可以限制

全局变量的作用域，全局静态变量的作用域仅限本文件，它对在其他文件不可见，也就是说不能在其他文件中引用该全局静态变量，但其他文件中可以定义和它名字相同的变量，不会发生冲突。

在函数的返回类型前加上 static 关键字，函数即被定义为静态函数。静态函数与普通函数的不同在于，它只能在声明它的文件当中可见，不能被其它文件使用，其它文件中可以定义相同名字的函数，不会发生冲突。

局部静态变量的作用域与局部变量的作用域相同，其作用域都是从定义开始到函数或程序块结束为止。

26、C++和 C 的区别？

设计思想上：

C++是面向对象的语言，而 C 是面向过程的结构化编程语言，C++在 C 的基础上增加了类。

语法上：

① C++具有封装、继承和多态三种特性。

② C++相比 C，增加了许多类型安全的功能，比如强制类型转换。

③ C++支持范式编程，比如模板类、函数模板等。

④ 在 C++中，引用是一个经常使用的概念。引用型变量是其他变量的一个别名，我们可以认为他们只是名字不相同，其他都是相同的。

⑤ 在 C++语言中，仍然支持 malloc()和 free()来分配和释放内存，同时增加了 new 和 delete 来管理内存。

⑥ C++支持函数重载，允许有相同的函数名，不过它们的参数类型不能完全相同，这样这些函数就可以相互区别开来。而这在 C 语言中是不允许的。

27、程序的内存模型分为那几个区域？

（1）代码区（.text 段）：存放代码。

（2）静态存储区（.data 段+.bss 段）：存放全局变量、静态变量、常量。其中初始化过的放在.data 段，未初始化过的放在.bss 段。

（3）堆区：由程序员根据需要自己申请空间（使用 new 或 malloc），需要自己回收这些空间（使用 delete 或 free），如果没有回收，程序结束时操作系统会回收这些空间。

（4）共享库的内存映射区域：动态链接库是在程序运行时才被链接的，这一块空间是为动态链接库准备的。

（5）栈区：存放函数的参数、返回值、局部变量等。

（6）内核虚拟内存空间：这段地址空间只能在内核态时才能访问，一般情况下进程运行在用户态，当发生中断时（比如线程切换、程序异常），就会进入内核态，处理完中断后，再回到用户态。

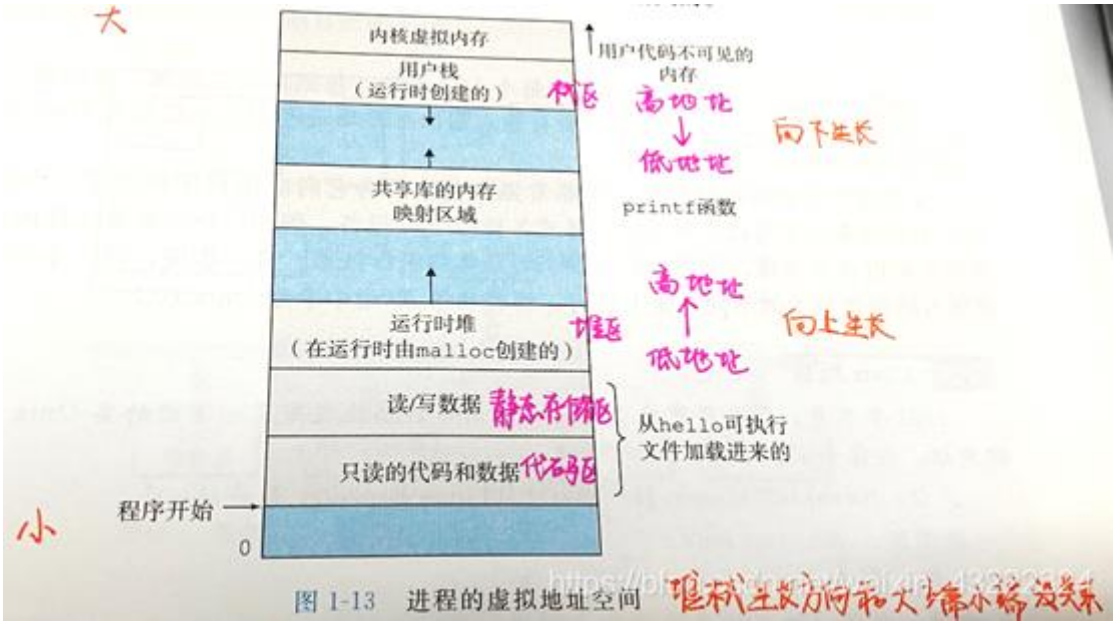


图 1-13　进程的虚拟地址空间

28、malloc/free 和 new/delete 的区别？

malloc/free	new/delete
C 标准库中的函数	C++中的操作符（关键字）
malloc 内存分配成功返回 void*，需要将 void*指针强制转换成我们需要的类型	new 内存分配成功时，返回的是对象类型的指针，类型严格与对象匹配，无须进行类型转换
malloc 需要显式地指出（自己算）所需内存的字节数	使用 new 申请内存分配时无须指定内存块的大小（编译器会根据数据类型计算）
malloc 分配内存失败时返回 NULL	new 内存分配失败时，会抛出异常
malloc 申请空间，但不会调用构造函数（因为它没有构造函数）	new 申请空间+调用构造函数
free 释放空间，但不会调用析构函数（因为它没有析构函数）	delete 释放空间+调用析构函数
new/delete 底层也是用 malloc/free 来实现申请和释放内存的。	

29、解决哈希冲突的方法有哪些?

（1）开放地址法

当发生地址冲突时，按照某种方法继续探测哈希表中的其他存储单元，直到找到空位置为止。（除留取余法等）

（2）再哈希法

当发生冲突时，使用第二个、第三个、哈希函数计算地址，直到无冲突时。缺点：计算时间增加。

（3）建立一个公共溢出区

除了原本的存储空间外，另外开辟一个存储空间用以存储发生冲突的记录。

（4）链地址法

哈希表每个结点后跟一个链表用来存放冲突的值。

30、C++内存对齐?

各种数据类型所占字节数:

char	bool	short	int	float	double	long long
1	1	2	4	4	8	8

其他:

数组	指针	对象	函数
数组大小*数据类型大小。 例: char a[7]; 占 7 个字节	与计算机位数有关。 32 位，指针 4 字节; 64 位，指针 8 字节;	有虚函数占用 4 个字节(也就是一个指针的大小）。	

一个类占多少字节?

（1） 空类：1 字节。（为什么？）

（2） 只含有函数（也含构造和虚构函数）：1 字节，因为函数不占用类空间。

（3） 有虚函数：无论有多少个虚函数，都占 4 个字节（一个指向虚函数表的指针，64 位系统指针是 8 个字节）。

（4） 只有数据成员：

例如：class A{

int a; char b; int c;

}

占 12 个字节，因为要考虑到内存对齐问题，这里以 4 字节对齐。

（5） 类中的静态成员变量不占类的内存，静态成员变量存储在静态存储区，并且静态成员变量的初始化必须在类外初始化，且只能初始化一次。

（6） 类与类之间也要内存对齐。比如一个类占 18 个字节，那个编译器会为它补齐 2 个字节，占 0~19 的内存，下一个类从偏移量为 20 的地址开始存放。

为什么要内存对齐?

比如以 4 字节对齐为例，一个 int 类型 4 字节的数据如果放在内存地址 2 开始的位置，那么这个数据占用的内存地址为 2、3、4、5，那么这个数据就被分为了 2 个部分，一个部分在地址 0～3 中，另外一部分在地址 4～7 中，由于 32 位 CPU 一次读取

4 个字节，所以，CPU 会分 2 次进行读取，先读取地址 0～3 中内容，再读取地址 4～7 中数据，最后 CPU 提取并拼接出正确的 int 类型数据。那么如果我们把这个 int 类型 4 字节的数据放在从地址 0 开始的位置，CPU 只要进行一次读取就可以得到这个 int 类型数据了。

因此内存对齐可以让内存存取更有效率。

31、大端存储、小端存储？

大端存储：

数据的尾端字节存储在内存的高地址。即头端字节存储在内存的低地址，数据依序增加存储。

小端存储：

数据的尾端字节存储在内存的低地址。即头端字节存储在内存的高地址，数据依序递减存储。

比如，一个 int 型数据用十六进制表示 0x 00 01 02 03，一个 16 进制数字=2^4，占 4 位，2 个十六进制数字占 8 位即 1 个字节，内存中是一个字节一个字节存放的。

大端存储（更符合日常理解）	小端存储（更适合计算机）
（高位）0x 00 01 02 03（低位）	（高位）0x 00 01 02 03（低位）
低地址 高地址	高地址 低地址

扩展：如何判断自己的计算机是大端还是小端？

32、C++中，一个程序从源代码到可执行程序的过程？

预处理、编译、汇编、链接（指静态链接）。

一、 预处理

C 语言的宏替换和文件包含的工作，不归入编译器的范围，而是交给独立的预处理器。主要处理源代码文件中的以“#”开头的预编译指令。处理规则见下：

1、删除所有的#define，展开所有的宏定义。

2、处理所有的条件预编译指令，如“#if”、“#endif”、“#ifdef”。

3、处理“#include”预编译指令，将文件内容替换到它的位置，这个过程是递归进行的，文件中包含其他文件。

4、删除所有的注释，“//”和“/**/”。

5、保留所有的#pragma 编译器指令，编译器需要用到他们，如：#pragma once 是为了防止有文件被重复引用。

6、添加行号和文件标识，便于编译时编译器产生调试用的行号信息，和编译时产生编译错误或警告是能够显示行号。

二、编译

把预处理之后生成的 xxx.i（C）或 xxx.ii（C++）文件，进行一系列词法分析、语法分析、语义分析及优化后，生成相应的汇编代码文件。

三、 汇编

将汇编代码转变成机器可以执行的指令(机器码文件，二进制)。汇编过程是根据汇编指令和机器指令的对照表一一翻译过来，汇编过程由汇编器 as 完成。

经汇编之后，产生目标文件(与可执行文件格式几乎一样)xxx.o(Windows 下)、xxx.obj(Linux 下)。但是，经过预编译、编译、汇编之后，生成机器可以执行的目标文件之后，还有一个问题——变量 a 和数组 arr 的地址还没有确定。这就需要链接器来搞定啦。

四、链接（这里讲的链接，严格说应该叫静态链接。）

链接操作最重要的步骤就是将函数库中相应的代码组合到目标文件中。链接就是将不同部分的代码和数据收集和组合成一个单一文件的过程，也就是把不同目标文件合并成最终可执行文件的过程。当然，务必知道：这个过程不涉及内存。

补充：链接可以分为三种情形

1，编译时（compile time）链接，也就是我们常说的静态链接，是在源代码被翻译成机器代码时完成的；

2，装入时（load time，加载时）链接，也就是程序被加载器加载到内存时；

3，运行时（run time）链接，应用程序执行时。

33、类中的 public, protect, private ?

（1）public 修饰的变量和函数在类的内部和外部都可以访问；

（2）private 修饰的变量和函数只有在类的内部可以访问；

（3）protected 修饰的变量和函数在类的内部可以访问，还可以在派生类中访问。如果类没有派生出其他类，那么 protected 和 private 是完全相同的，protected 和 private 一样只能在类的内部访问，不能在类的外部访问。

public protect private 继承

继承是使代码可以复用的重要手段，也是面向对象程序设计的核心思想之一。简单的说，继承是指一个对象直接使用另一对象的属性和方法。C++中的继承关系就好比现实生活中的父子关系，继承一笔财产比白手起家要容易得多。继承的方式有三种分别为公有继承（public）,保护继承（protect），私有继承（private）。

继承方式	基类的 public 成员	基类的 protected 成员	基类的 private 成员	继承引起的访问控制关系变化概括
public 继承	仍为 public 成员	仍为 protected 成员	不可见	基类的非私有成员在子类的访问属性都不变
protected 继承	变为 protected 成员	变为 protected 成员	不可见	基类的非私有成员都成为子类的保护成员
private 继承	变为 pirvate 成员	变为 private 成员	不可见	基类中的非私有成员都成为子类的私有成员

34、struct 和 class 区别 ?

C++中保留了 C 语言的 struct 关键字，并且加以扩充。在 C 语言中，struct 只能包含成员变量，不能包含成员函数。而在 C++中，struct 类似于 class，既可以包含成员变量，又可以包含成员函数。

C++中的 struct 和 class 基本是通用的，唯有几个细节不同：

（1）使用 class 时，类中的成员默认都是 private 属性的；而使用 struct 时，结构体中的成员默认都是 public 属性的。

（2）class 继承默认是 private 继承，而 struct 继承默认是 public 继承。

（3）class 可以使用模板，而 struct 不能。

35、C++内存泄漏

概念：内存泄漏是指你向系统申请分配内存进行使用（new/malloc），然后系统在堆内存中给这个对象申请一块内存空间，但当我们使用完了却没有归系统（delete），导致这个不使用的对象一直占据内存单元，造成系统将不能再把它分配给需要的程序。

危害：内存泄露最明显最直接的影响就是导致系统中可用的内存越来越少。直到所有的可用内存用完最后导致系统无可用内存而崩溃。

预防办法：用智能指针、正确的 new 和 delete

36、C++数组越界

概念：对数组访问超出了原来的数组定义的范围，使用了超过有效范围的偏移量，这就是数组下标越界。c 和 c++语言中数组下标越界，编译器是不会检查出错误的，但是实际上后果可能会很严重。

危害：当出现越界时，由于无法得知被访问空间存储的内容及功能，所以会出现不可预知后果，可能程序崩溃，可能运算结果非预期。

预防办法：在编程时要避免指针越界访问，对于用变量作为参数访问元素时，需要控制好变量范围。如果变量值由外部因素决定，那么访问前先对合法性做判断，防止越界。

37、C++11 版本内容，以及智能指针相关

查链接、这个挺重要的。