C++ 基础知识面试版

1. C 和 C++ 区别?
2. C++ 中从代码到可执行文件中间经历的过程?
3. main 函数执行前和执行后的代码是什么?
4. C++ 程序内存分布?
5. C++ 的 struct 和 C 的 struct 的区别?
6. 形参和实参的区别?
7. 结构体内存对齐问题?
8. const 用法?
9. static 用法?

C 和 C++ 区别?

1. C 和 C++ 最大的区别是在于应用场景和设计思想上的区别
2. 出于对内存使用和执行效率的考虑，C 的特点是高效、精简；多用于操作系统和内核驱动；C++ 设计之初的目的就是为了把 C 繁杂的实现过程抽象为对象并且进行实例化管理；所以C++更适合做大型软件；概括的说 C 注重逻辑实现，C++ 注重程序的整体设计；
3. 设计思想的不同最终导致了现在 C++ 比 C 多了很多语法、关键字的不同以及类继承重载虚函数特性甚至是模板STL这些东西；

C++ 中从代码到可执行文件中间经历的过程?

1. 预处理阶段：处理以 # 号开头的预处理命令，生成 .i 或者 .ii 文件；
2. 编译阶段：对预编译阶段生成的文件进行词法分析、语法分析、语义分析和优化之后，生成汇编文件；
3. 汇编阶段：把汇编代码翻译生成机器码文件，生成 .obj 或者 .o 文件；
4. 链接阶段：对生成的 .obj 文件和 .o 文件和相关的文件关联起来，申城可执行程序；

main 函数执行前和执行后的代码是什么?

- main 函数执行之前主要是初始化系统资源：
    1. 设置栈指针；
    2. 初始化静态变量和全局变量，即 .data 段内容；
    3. 给未初始化的全部变量赋初值，即 .bss 段内容；
    4. 全局对象的初始化，全局对象会在 main 之前调用构造函数；
    5. 将 main 函数的参数 argc, argv 等传递给 mian 函数；
- main 函数之后：
    1. 全局对象的析构函数：
    2. 调用 atexit 注册的函数：

C++ 程序内存分布?

- 用户空间内存从高到低依次为 7 种不同的内存段： 0. 内核空间：
    1. 栈段：包括局部变量和函数调用的上下文；
    2. 文件映射段：包括动态库、共享内存等；
    3. 堆段：包括动态分配的的内存，从低地址开始向上增长；
    4. 未初始化数据段：包括未初始化的全局变量和静态变量；
    5. 已初始化数据段：包括已初始化的全局变量和静态变量；
    6. 程序文件段：包括二进制可执行代码；

C++ 的 struct 和 C 的 struct 的区别?

- C 语言中
    1. struct 是用户自定义数据类型，里面没有权限限制，
    2. 只是一些变量的集合体，可以封装数据但不能隐藏数据，而且成员不能是函数；
    3. 在一个结构标记声明后，必须在结构标记前加 struct 才能做结构类型名；
- C++ 中：
    1. 是抽象数据类型，支持成员函数定义，能继承，能实现多态；
    2. 有访问权限，可以和 class 一样有成员函数， 成员默认访问说明符为 public；
    3. 结构体标记可以直接作为结构体类型名使用；

形参和实参的区别?

1. 形参变量只有在被调用的时候才会被分配内存单元，在调用结束后会立刻释放内存单元，因此形参只能在函数内部有效；
2. 实参可以是常量、变量、表达式、函数等，无论实参是何种类型的值，在进行函数调用时，它们都必须是确定的值，以便把这些值传递给形参，因此应预先用赋值、输入等方式使形参获得确定的值；
3. 实参和形参必须在数量上、类型上和顺序上保持一致，否则会发生"类型不匹配"错误；
4. 函数调用时数据传送的方向是单向的，即只能把实参的值传递给形参，而不能把形参的值反向传递给实参，因此在函数调用的过程中，形参的值发生了改变不会影响到实参；
5. 当形参和实参都不是指针类型时，形参和实参是不同的变量，它们在内存中处于不同的位置；形参会将实参的值复制一份，在函数调用结束后形参被释放，而实参不会发生改变；

结构体内存对齐问题?

1. 结构体中的每个成员的相对结构体首地址的偏移量是对齐参数的整数倍，首位元素是对齐参数的 0 倍；
2. 结构体变量所占空间大小是对齐参数的整数倍，如有需要则在最后一个成员末尾填充字节达到要求；
3. 获取结构体成员相对于解结构体开头的字节偏移量： (unsigned long)(&(type.Member)) - (unsigned long)(&(type))

const 用法?

1. const 修饰变量、数组、指针、函数参数、函数返回值、引用都是将修饰对象声明为只读属性，不允许修改 const 所修饰的对象；
2. const 修饰类成员函数、类的作用都是限制在成员函数中堆类成员变量的修改；

- 常量指针：const 修饰 常量的指针：int const *p/const int *p, *p 值不可变；
- 指针常量：const 修饰 的指针指向常量：int *const p, p 值不可改变；

static 用法?

- 不考虑类时：
    1. 修饰全局变量或者函数的作用是隐藏：全局变量和函数对其他文件不可见；
    2. 修饰局部变量的作用是改变局部变量的生存期：作用域是局部内，但是生命周期等同于全局变量的生命周期；
- 考虑类时：
    1. 修饰成员变量：该变量只与类关联，不与类的对象关联，该类的所有对象共用该变量；
    2. 修饰成员函数：该函数没有 this 指针，无法访问类中的非 static 成员和变量；不能被声明为 const、虚函数、volatile；可以被非 static 成员函数任意访问；