7.1C/C++关键字

7.1.1 static静态变量有什么作用

C语言中

1Static修饰的全局变量与普通全局变量的区别:静态全局变量值初始化一次,这样做的目的是为了防止在其它文件单元中被引用.

2 Static局部变量和普通局部变量的区别在于,静态全局变量之初始化一次,下次使用的还是上一次的值.

3 Static函数与普通函数的区别,作用域不一样,静态函数只在一个源文件中有效,不能被其他源文件使用.

C++语言中,

静态成员变量,是类的成员,无论类的对象被定义了多少个,静态成员变量只有一份复制品,由该类型的所有对象共享访问.而普通成员变量,每个对象都有自己的复制品.

静态成员变量存储在全局数据区,定义时分配空间.

静态成员变量和普通成员变量一样遵循public\protected\private访问规则.

静态成员变量与全局变量相比的优势,作用域限制在本类中,不会与全局变量命名冲突.

private关键字可以限制访问规则.

不可以在头文件中定义静态变量.

如果在头文件中定义了静态变量,在引用该头文件的文件中都单独存在一个静态变量,会引起空间浪费或者程序错误.

7.1.2const有哪些作用

C语言中,定义变量为常类型以及修饰函数参数与返回值.

C++语言中,还可以修饰函数的定义,定义类的成员函数.

常类型的变量或对象的值是不能被更新的.

7.1.3 switch语句中case结尾是否必须添加break语句.

如果忽略了break语句,那么匹配case后面的所有情况都会被执行.

Switch(c)语句中c可以是int\long\char\unsigned int等类型.唯独不可以使float类型.

7.1.4 volatile在程序中有什么作用

当读取一个变量时,为提高存取速度,编译器优化时有时会先把变量读取到一个寄存器中,当以后再去取该变量值时,就直接从寄存器中取值,当变量值在本线程中改变时,会同时把新值复制到该寄存器中.

Volatile是一个类型修饰符,用来修饰被不同线程访问和修改的变量.被volatile类型定义的变量,系统每次用到它的时候都是直接从对应的内存中提取,而不会利用cache中原有数值,以适应它随时的变化,系统对这种变量的处理不做优化.

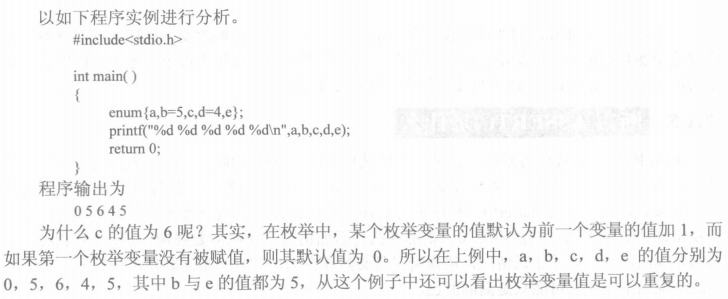
所以volatile一般用于修饰多线程间被多个任务共享的变量和并行设备硬件寄存器.

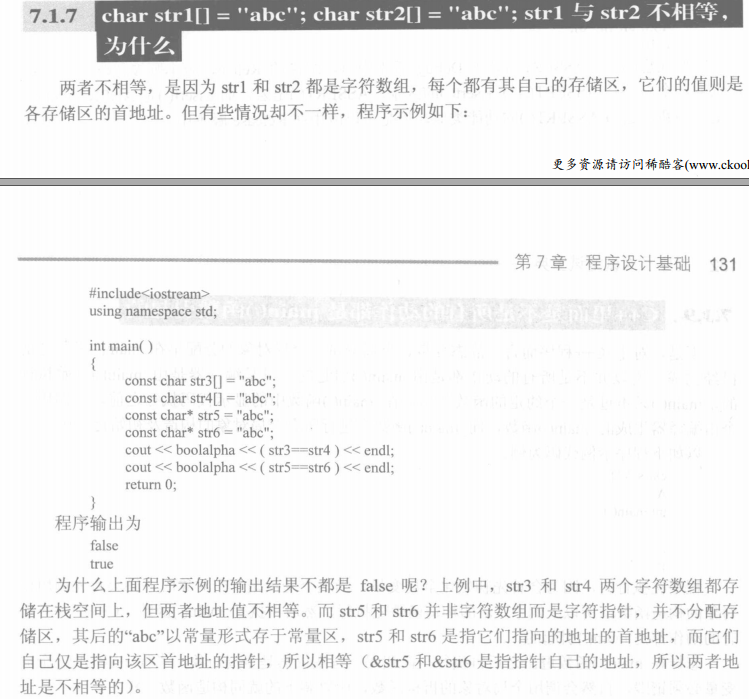
7.1.5断言assert

它是一个调试程序时经常使用的宏.定义在<assert.h>头文件中,通常用于判断程序中是否出现了非法的数据,在程序运行时他计算括号内的表达式的值,如果表达式的值为FALSE(0),程序报告错误,终止运行.

影响性能,在文件中插入#define NDEBUG来禁用assert.

7.1.6枚举





7.1.8main函数参数argc\argv含义.

Int argc表示命令行参数的数目

Char\*argv[]是指向字符串的指针数组

7.1.9C++main是不是所有的动作都是main函数引起的

静态变量\全局变量\全局对象的分配早在main函数之前完成,只是编译器是有main函数开始执行的.

在面函数中的显示代码之前,会调用一个由编译器生成的\_main()函数,而\_main()函数会进行所有全局对象的构造及初始化工作.

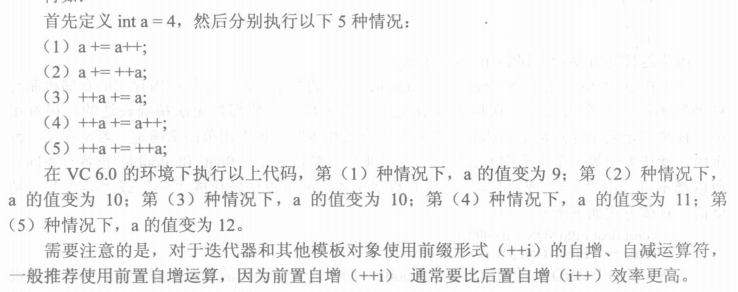
7.1.10\*p++与(\*p)++等价吗?

根据优先级顺序,不等价,前者先完成取值操作,然后对指针地址执行++操作,而后者为首先执行取值操作,然后对该值就行++运算.

7.1.11前置运算与后置运算有什么区别

以++操作为例,++a表示取a的地址,增加它的内容,然后把值放到寄存器中;

a++表示取a的地址,把它的值装入寄存器,然后增加内存中a的值.



7.1.12 a是变量,执行(a++)+=a语句是否合法

左值出现在等号左边,可以被改变,它是存储数据值的那块内容的地址,也被称为变量的地址,右值是指存储在某内存地址中的数据,也被称为变量的数据.

本体不合法,a++不能当做左值使用,++a可以当左值使用.

7.1.13 float\bool\int\指针变量与零值比较

Int类型

If(n==0)

If(n!=0)

Float类型在计算机中,大多数浮点数无法精确表达,很能用等号来判定两个浮点数是否相同.

const float EPSINON=0.00001;

If(x>=-EPSINON)&&(x<=EPSINON))

Bool类型

if(flag)

If(!flag)

指针类型

if(p==NULL)

If(p!=NULL)

7.1.14 new/delete malloc/free区别是什么

Malloc/free是C/C++语言的标准库函数,在C语言头文件stdlib.h.

New/delete是C++运算符.malloc/free无法满足动态对象的需求,对象在创建的同时要自动执行构造函数,对象销毁之前要自动执行析构函数,而malloc/free不在编译器控制范围内,无法执行构造函数和析构函数.

具体而言

1 new能够自动计算需要分配的内存空间,而malloc需要手工计字节数.

2 new/delete直接带具体类型的指针,malloc/free返回void类型的指针.

3 new调用类的构造函数,free调用函数的析构函数.

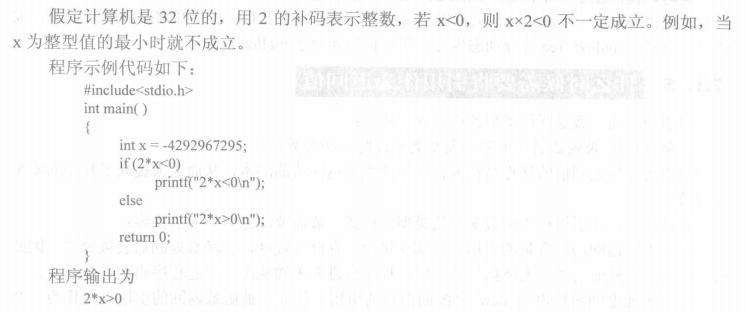
7.1.15 什么时候需要将引用作为返回值

Int& fun();

https://blog.csdn.net/lanyang123456/article/details/40866281

7.1.16 变量名为618Software是不合法的.

7.1.17 C语言中,整型变量x小于0,是否可知2x也小于0.



7.1.18 exit()和return的区别

return是语言级别的关键字,表示函数退出,堆栈的返回,

exit是系统函数,有函数库提供的,终止进程.

7.1.19 实现String函数

7.1.20 在C++中如何实现模板函数的外部调用

export实现模板函数的外部掉用,类似于extern关键字.

extern int n;

extern struct Point p;

extern class A a;

export template<class T> class Stack<int>s;

export template<class T>void f(T&t){}

7.1.21 在C++中,关键字explicit有什么作用

禁止隐式转换

一个参数的构造函数(或者除了第一个参数其余参数都有默认值的多参构造函数)一般具有两个功能,构造器和默认且隐含的类型转换操作符.而这并不是程序员本意,加上explicit关键词,指定这个构造器只能被明确的调用\使用,不能用作类型隐式转换.

7.1.22 C++中异常处理方法以及使用的关键字

try\catch\throw.

throw来抛出异常,try\catch来捕捉异常,并做相应的处理.

7.1.23 回调函数

回调函数就是被调用者回头调用的函数,它是一个通过函数指针调用的函数.

如果他函数的指针(地址)作为参数传递给另一个函数,当这个指针被用为调用它所指向的函数时,此时就可以称它为回调函数.

7.2 内存分配

7.2.1 内存分配的形式有哪些

C/C++编译的程序所占的内存一般分为以下几个部分.

1. 由符号起始的区块,block started by symbol,BBS段,用来存放程序中未初始化的全局数据和静态数据的一块内存区域.BBS段属于静态内存分配,程序结束后由系统自动回收.
2. 数据段data segment,存放程序中已经初始化的全局变量的一块内存区域,也属于静态内存分配.
3. 代码段 code segment/text segment,也叫文本段,通常指用来存放程序执行代码的一块区域.包括类成员函数和全局函数以及其他函数代码的一块内存区域.

这部分区域的大小在程序运行前就已经确定,并且内存区域通常是只读.某些结构也允许代码段可写.

1. 堆heap,用来存放进程运行中被动态分配的内存段,它的大小不固定.可动态扩张或者缩减.使用函数new\free或者malloc\delete来管理.分配方式类似于链表.
2. 栈stack,用来存储程序临时创建的局部变量,一般包括函数括号中定义的变量.除此之外,在函数被调用,其参数也会被压入发起调用的进程栈中,并且等到调用结束后,函数的返回值也会被存放回栈中.

栈由编译器自动分配释放,存放函数的参数\局部变量等值.其操作方式类似于数据结构中的栈.

7.2.2 什么是内存泄漏

堆是动态分配内存的,并且可以分配使用很大的内存,使用不好会产生内存泄漏.频繁使用malloc\delete会产生内存碎片.

所谓内存泄漏memory leak,是指由于疏忽或错误造成程序未能释放已经不再使用的内存.

并非指物理上的泄漏,而是指由于程序设计错误,市区了读卡改段内存的控制.

7.2.3 栈空间的最大值是多少

栈是向低地址拓展的数据结构,

堆是向搞地址拓展的.用链表的方式来存储,自然是不连续的,使用的是虚拟内存.

在win系统中,大小是2M,申请堆空间的大小一般小于2GB.

linux系统默认的栈控件大小是8M,通过ulimit -s来设置.

7.2.4 什么是缓冲区溢出

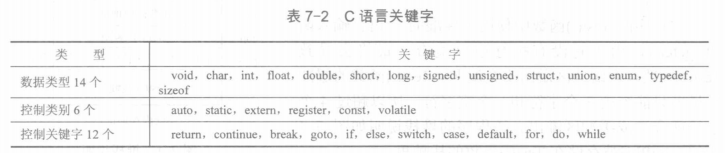
缓冲区是程序运行时机器内存中的一个连续块,它保证了给定类型的数据随着动态分配变量会出现问题.

缓冲区溢出是指当向缓冲区内填充数据位数超过了缓冲区自身的容量限制时,发生的溢出的数据覆盖在合法数据(数据\下一条指令的指针\函数返回地址等)上的情况.

最好的程序是不允许输入超过缓冲区长度的字符并检查数据长度,由于大多数程序都会假设数据长度总是与所分配的存储空间相当,进而存在缓冲区溢出隐患.

7.3 关键字

7.3.1 sizeof是关键字吗



sizeof是数据类型的关键字，不是函数。

7.3.2 strlen\sizeof

strlen(“\0”)=0 sizeof(“\0”)=2