C语言学习总结

# 函数

1、递归优缺点

优点：实现简单

缺点：需要消耗更多计算机资源，如内存，调用时间

2、函数原型与已定义的字符常量（源代码组织）

三种类型：头文件、函数定义、主程序

函数原型与已定义的字符常量要放在头文件中，方便组织维护。（备注：#include “xxxx.h” 指令中的双引号表示包含的头文件位于当前目录，也就是源程序所在的目录）。

3、c语言中没有引用

因此如果要在被调用函数中修改调用函数中的变量，就必须要想被调用函数传递调用函数中的变量。

如果要修改调用函数的多个变量，并把被调函数中的多个值传回调用函数，则必须使用指针；

4、地址运算法&

5、间接运算符\*（解引用运算符）

6、函数的参数

一种是给函数传递变量的值，一种是给函数传递变量的地址；函数形参和函数内部定义的局部变量都是属于函数私有的。

7、函数相关的概念

函数形式、形式参数、实际参数、传递值、函数返回类型、函数原型（函数签名）

8、函数原型、函数定义

# 数据与指针

备注：”存储类别”概念

1、数组声明、数组初始化

2、关键字：const、static

3、指针涉及的量：指针的值、指针指向地址的值、指针的地址

4、指针的8种基本操作

赋值：把地址赋给指针

解引用：获取指针指向地址的值（备注：不要解引用未初始化的指针；也就是当不清楚指针指向时，不要直接将一个值赋给指针的解引用）

取址：获取变量（指针也是一个变量）的地址

指针与整数相加：

递增指针：

指针减去一个整数：

指针递减：

指针求差：

比较：

5、指针分类

指针数组、函数指针、指向指针的指针数组、指向函数的指针数组

6、const

const变量、const数组、const指针、指向const的指针

备注：const修饰的是它后面的符号

注意：c语言中允许将一个const指针赋给非const指针，但是在c++中就不允许；

7、指针概念辨析

int (\*p)[3]; #这里p相当于二维数组的数组名, [3]指的是二维数组的第二维元素个数；所以p是一个指向具有三个整数元素的数组的指针；

int \*\*p; #这里p是一个指向int类型指针的指针

8、数组初始化与指针初始化

数组初始化是将每个元素拷贝到数组，指针初始化是将第一个元素的地址拷贝给指针；

9、辨析int \*p[3] 与int (\*p)[3]

前者是有三个int类型的指针，后者是一个指向具有三个元素的数组的指针；

# 字符串

1、字面量

整数字面量：eg: 5;字符串字面量：eg：“qwerty”；

2、指针数组与二维数组（用字符串字面量初始化指针数组和二维字符数组）

指针数组：int \*p[3]; 优点是存储空间比二维数组少，

二维数组int a[3][40]; 优点是可以修改数组中元素

3、gets、puts、fgets、fputs

gets、puts不安全，因为它将一个数组名转化为一个指针，从而只知道数组的首地址，不知道数组的结束地址，因此可能访问或者修改其他地址；前者在读入字符串后，会自动去掉换行符，后者在输出时会自动添加换行符；puts不安全是因为它必须要读到空字符才会结束读取并添加换行符；

fgets，fputs是安全的行读取函数，因为它添加了一个参数，来限定可以读取（输出）的字符个数；fgets不会自动去掉末尾的换行符，fputs不会在字符串末尾自动添加换行符；

# 存储类别、链接和内存管理

auto, extern, static, register, const, volatile, restricted, \_Thread\_local, \_Atomic

rand(), srand(), time(), malloc(), calloc(), free()

1、左值、标识符

标识符时软件指定硬件内存中的对象的方式；

左值：指定对象的表达式成为左值

指定对象方式：1、标识符；2、表达式；

2、存储期、作用域、链接

存储期指对象在内存中保留了多长的时间。

3、作用域指程序可以访问标识符的区域；一个C变量的作用域可以分为：块作用域、函数作用域或文件作用域；

块是用一对花括号括起来的代码区域；

翻译单元：每个翻译单元对应一个源文件和它所包含的文件

4、链接：C变量具有三种链接属性：内部链接、外部链接、无链接

外部链接变量可以在文件程序中使用，内部链接变量只能在一个翻译单元中使用。加了static关键字的变量就是内部链接变量，仅限于一个翻译单元。

作用域和链接描述了标识符的可见性。

5、存储期描述了标识符访问对象的生存期。C对象有四种存储期：静态存储期、线程存储期、自动存储期、动态分配存储期。

静态存储期：这种存储期的变量在程序执行期间一直都存在。文件作用域变量都具有静态存储期。使用关键字static定义的块作用域变量也具有静态存储期；



自动变量

寄存器变量

块作用域的静态变量：指在程序的执行过程中，变量在内存中原地不动；这种变量具有块作用域，静态存储期。

外部链接的静态变量：具有文件作用域、外部链接、静态存储期。使用extern关键字再次声明（extern是用于再次声明外部链接变量的（全局变量）），就可以在定义该变量的文件之外的其他文件中使用该变量了。

备注：外部变量的作用域是从声明出到文件结尾。

6、声明：定义式声明和引用式声明

7、五种存储类型

外部变量

内部变量

块作用域自动变量

块作用域静态变量

寄存器变量

8、存储类别与函数

外部函数、静态函数、内联函数

9、ANSI C类型限定符

恒常性（constancy）：const

易变性（volatility）：volatile

restrict

\_Atomic

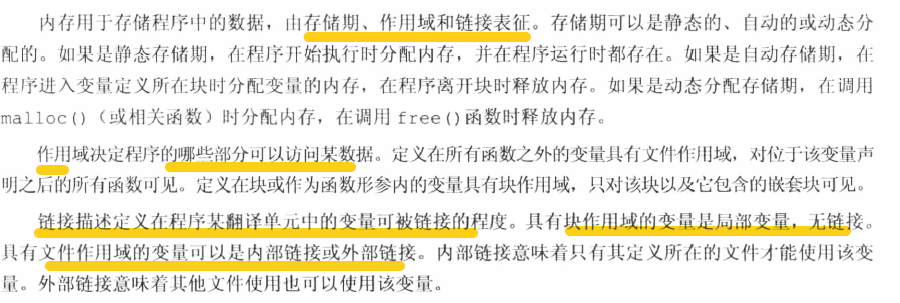
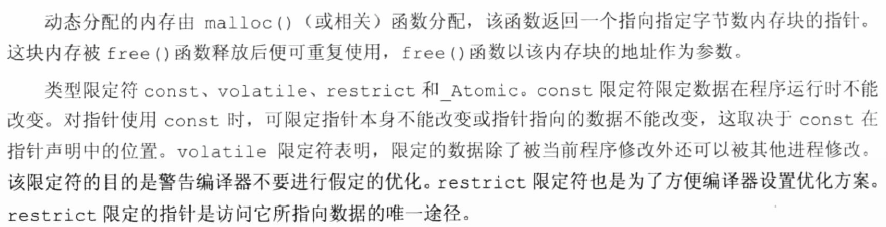
const:指变量不可修改；const与\*结合的辨析：const在\*之前是指指针所指的变量不可修改（指针指的变量不可改），const在\*之后是指指针的值不可修改（地址不可改）。

全局变量使用const: ①在一个文件声明式定义，在另一个文件声明式引用；②直接在头文件中声明式定义，在其他文件中包含该头文件。这种方式必须在头文件中使用static来定义全局变量，避免每个包含该头文件的文件中都声明式定义该变量。

volatile:该限定符告诉计算机，除了程序可以修改该变量外，代理也可以改变该变量的值。通常被用于硬件地址以及其他程序或者同时运行的线程中共享数据。这个限定符使得：当使用volatile 声明的变量的值的时候，系统总是重新从它所在的内存读取数据，而不使用寄存器中的缓存的值。（这个关键字告诉编译器不要做某部分代码的计算优化，避免出错）

restrict：这个关键字允许编译器优化某部分代码以更好的支持计算。该关键字只能用于指针（因此restrict必然是放在\*后面的），表明该指针是访问数据对象的唯一且初始的方式

-Atomic：并发程序设计把程序执行分成可以同时执行的多个线程。这引发了问题：如何管理访问相同数据的不同线程。C11通过包含可选的头文件stdatomic.h和threads.h提供一些可选的管理方法。

# 结构和其他数据形式

关键字：struct、union、typedef

运算符：. 与 –>

结构体：定义、创建、初始化（可以类似于数组一样初始化）

函数参数与结构体：①传递结构的成员；②传递结构的地址（指针）；③传递结构；作为函数参数。

允许将一个结构赋给相同类型的另一个结构；然而数组却不可以。

结构和结构指针的选择：结构指针优缺点：执行快，只需传递一个地址，但缺点是无法保护数据；把结构作为参数优缺点：函数处理的是结构的副本，保护了原始数据，缺点是较老版本的实现无法处理这样的代码，传递结构浪费时间和存储空间。

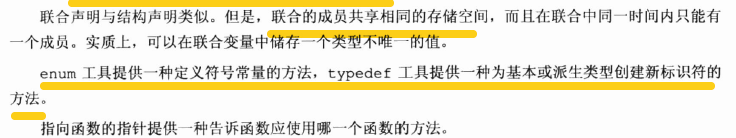
备注：结构体中的成员指针只能用来管理那些已分配和在别处分配的字符串。

联合：指的是同一块地址，能存储多种类型的变量，但是在某一时刻只能存储一种确定的类型。

备注：. 成员运算符与->间接成员运算符

函数、函数指针、函数作为形式参数:

函数指针（eg: funcP）的使用方式：1、(\*funcP)(参数)；2、funcP(参数)；



# 位操作

位运算符

&，|，~，^，<<， >>，

应用：掩码（&），打开位（|），关闭位（&），切换位（^），检查位（&）；

位字段：通过一个结构声明来建立；

通过位运算法或者位字段可以压缩存储空间

对齐特性：

\_Alignof： 用于查询一个类型或者一个变量的对齐值；

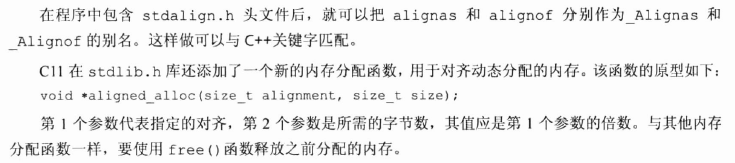
size\_t d\_align=\_Alignof(float);

\_Alignas： 用于制定一个变量或者一个类型的对齐值；

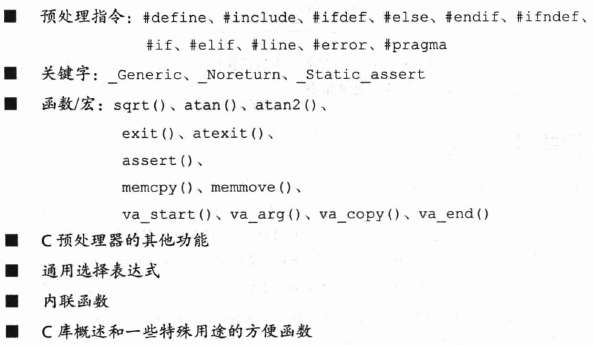
\_Alignas(double) char c1; // 指字符变量c1的对齐值为8

\_Alignas说明符可以强制执行数据存储区上的对齐要求。

对齐概念：是由于处理器与存储器之间的数据交互方式是一次读（写）32位（32位机），对于一个int型数据，为了能一次读取，减少读（写）时间，就需要它是4的倍数，这样能一次性读取，不然就需要两次或以上的读取。



# 预处理器和C库



#define pi 3.14; //用于定义明示常量

**宏的概念：** 明示常量名即为宏。

宏：①类对象宏；②类函数宏。

符号常量的价值：助记、易更改、可移植

**记号：**把宏的替换体看做记号型字符串，而不是字符型字符串；

在宏定义中：字符型字符串、记号型字符串

在#define中使用参数，也就是创建类函数宏

**#运算符**：可以把记号转换成字符串；#x就是转换子串“x”这个形参名，称为字符串化；

**##运算法**：将两个记号组合成一个记号

**变参宏：**… 与 \_\_VA\_ARGS\_\_



**宏和函数的选择：**有些编程任务既可以用带参数的宏完成也可以用函数完成；

宏和函数的选择实际上是时间和空间的权衡。宏生成内联代码，即在程序中生成语句，调用多少次就插入多少行语句，因此消耗更多空间；而函数则只有一个副本，但是运行必须跳转至函数内部，再返回主程序，因此函数消耗更多时间；

**文件包含**：#include

#include<stdio.h> //尖括号告诉预处理器在标准系统目录中去查找改文件；

#include“mystuff.h” //双引号告诉预处理器在当前目录中查找该文件；

**头文件常用形式：**明示常量、宏函数、函数声明、结构模板定义、类型定义；声明外部变量供其他文件共享；使用具有文件作用域、内部链接、const限定符的变量和数组，static意味着每个包含该头文件的文件都将获得一份副本，从而不需要在一个文件中进行定义式声明，在其他文件中进行引用式声明；

**#define:宏定义，主要包括常量定义，宏函数定义；**

**#include：文件包含，主要包含引入头文件；**

**其他指令：#undef, #if, #ifdef,#ifndef, #else, #endif, #error, #pragma**

**#undef：**用于取消以定义的#define指令；如果想使用一个名称，但不确定之前是否使用过，就可以用#undef指令取消该名称的定义，然后重新定义；

**条件编译：**可以让程序更容易移植，通过改变文件开头几个关键的定义，即可根据不同的系统设置不同的值和不同的文件；

**#if, #ifdef, #else, #endif：**创建条件编译，也就是使用这些指令告诉编译器根据编译是的条件执行或者忽略信息块；

#ifndef作用：防止相同的宏被重复定义；防止多次包含一个头文件；

C标准头文件使用#ifndef技巧来避免重复包含。这需要确保测试的标识符没有在别处定义；解决办法是：使用文件名作为标识符、使用大写字母、下划线代替文件名中的点字符、用下划线作前缀或后缀；

#if指令很像C语言中的if，#if后面跟整型常量表达式；可以使用C语言中的关系运算符合逻辑运算符；

新的编译器可以用#if defined (VAX) 代替 #ifdef VAX。

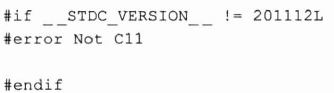
**预定义宏：C标准预定义的宏，不可取消，宏是全局的**

**预定义标识符：**

**#line指令重置\_\_LINE\_\_和FILE\_\_宏报告的行号和文件名；**



**#error指令让预处理器放出一条错误消息，该消息包含指令中的文本**



#pragma：pragma(编译指示)，目的是用于：通过命令行参数或者IDE菜单修改编译器的一些设置；

泛型选择（c11） \_Generic ：泛型编程指那些没有特定类型，但是一旦指定一种类型就可以转换成指定类型的代码；泛型表达式不是预处理指令但是在泛型编程中它常用作#define宏定义的一部分；**泛型选择语句和swich类似，只是前者用表达式的类型来匹配标签，而后者用表达式的值来匹配标签**。

**备注：宏必须定义为一个逻辑行。**

**内联函数： 函数说明符：inline；存储类别说明符：static**

**函数调用过程：建立调用、传递参数、跳转到函数代码、运行、返回；**

**内联函数会用内联代码替换函数调用；**

**内联函数创建：**标准规定具有**内部链接**的函数可以成为内联函数，同时内联函数的定义与调用该函数的代码必须在**同一个文件**中。通常，内联函数应该定义在首次使用它的文件中，所以内联函数相当于函数原型。

由于一般情况下内联函数都具有内部链接，因此如果程序有多个文件都需要使用某个内联函数，那么这些文件都必须包含该内联函数的定义。因此简单的方法是将**内联函数定义放在头文件中**。

**函数说明符：inline, \_Noreturn**

**Inline:内联函数**

**\_Noreturn: 表明调用完成后函数不返回主调函数。比如exit()函数，一旦调用exit()函数，它就不会返回主调函数。**

C库

可以在不同的位置找到库函数：①头文件；②库文件；

不同的系统搜索这些函数的方法不同：①自动访问；②文件包含：包含头文件；③库包含：编译链接时，指定库选项；

**类型变体**：C11新增的泛型选择表达式定义一个泛型宏，根据参数类型选择最合适的函数版本。利用关键字\_Generic定义泛型宏，自动依据参数类型来选择不同函数。

C99标准提供的tgmath.h头文件中定义了泛型类型宏，可以依据参数类型的不同选择不同的函数。

这里需要理解：使用宏函数（类函数宏）与函数的区别。（二者的名字是相同的）；在包含定义了泛型类型宏的头文件后，如果要调用函数而不是宏函数，可以用圆括号把被调用的函数名括起来。同时还可以通过函数指针的方式来调用函数（而不是类函数宏）；

**通用工具库**

通用工具库包含各种函数：随机数生成器、查找和排序函数、转换函数、内存管理函数。这些函数原型都在stdlib.h头文件中。

exit()与atexit(\*funct):exit是退出函数，在main函数中使用，相当于return，但是在其他函数中使用会终止整个程序；atexit(\*funct)则是接收一个函数指针，在退出时，会执行函数指针所指的函数。

qsort()函数：以指针、size\_t类型、函数指针为参数；

c与c++中的void\* 区别：void\*指无类型的指针，也即可以指向任意类型的指针。两种语言都可以将任意类型的指针赋给void\*指针，但是c++要求把void\*指针赋给任何类型的指针时，必须使用强制类型转换。

断言库

断言库assert.h头文件支持的断言库时一个辅助调试程序的小型库。由assert()宏组成，接受一个整型表达式作为参数。

**assert()宏**：参数是一个条件表达式或者逻辑表达式。当表达式为假时就用assert()语句终止程序。

使用assert()好处：自动标识文件和出问题的行号，还有一种无须修改代码就能开启或关闭assert()的机制。

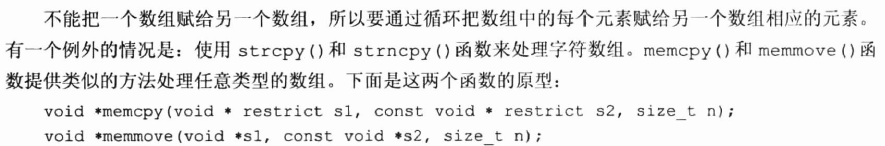
#define NDEBUG //这个宏可以禁用assert();

**\_Static\_assert()宏**：

可以导致程序无法通过编译，assert()可以导致正在运行的程序终止。

\_Static\_assert()接受两个参数：整型常量表达式、字符串。如果第一个参数为假，则编译器会显示字符串，并停止编译该程序。

**string.h库的memcpy()和memmove()**



两个函数都不知道数据的类型，他们只负责从一个位置把一些字节拷贝到另一个位置。

可变参数：stdarg.h

宏可以接受可变数量的参数：通过…和\_VA\_ARGS\_\_来实现

宏的泛型选择：\_Generic()来实现

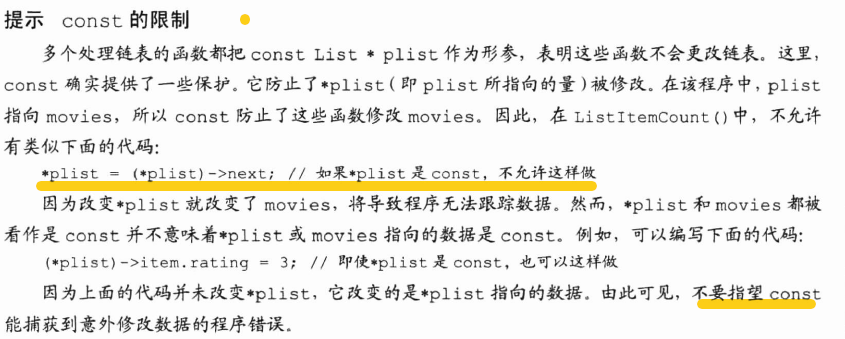
函数的可变参数：通过…来实现；

需要注意的是：**这些函数原型应该有一个形参列表，其中至少有一个形参（最右边的形参表示省略号部分代表参数的个数）和一个省略号。**

# 高级数据结构

**备注：**free()操作的参数是一个指针，该函数会释放该指针所指向的内存区块；原则上free()会使当前指针指向的内容不可用，因此对于链表节点的释放，需要先保存下一个节点的指针，再释放当前节点。

**备注：**const限定只能限定一层的不可更改，如果是多层的，则其他层是可以更改的（比如：const point \*p;p是一个指向类型point的指针，这个p指向的point类型的变量的值不可更改，但是当point是一个指针类型时，point类型变量指向的变量就是可以修改的 ）



程序项目设计：首先是概念模型设计，设计好接口，然后再设计细节；这样将用户接口和代码细节分开的程序，更容易理解和更新；

类型：属性和操作

定义新类型的方法：

1. 提供类型和相关操作的抽象描述；
2. 开发一个实现ADT的编程接口；
3. 编写代码实现接口；

链表：是一个能储存一系列项且可以对其进行所需要操作的数据对象；

建立抽象、建立接口、使用接口、实现接口；

定义抽象、定义接口、实现接口、测试；

# 总结

**C运算符**：算术运算符、关系运算符、赋值运算符、逻辑运算符、条件运算法（？：）、与指针有关运算符（&和\*）、符号运算符（+和-）、结构和联合运算符（.和->）、按位运算符、混合运算法（sizeof）

**C关键字**：

基本数据类型关键字:void,

存储类别关键字：auto, extern, static, register, \_thread\_local

限定符关键字：const, volatile, restrict (主要是对变量操作相关的限制), \_Alignas(bits\_num)(用于变量存储的位对齐限定), \_Alignof(类型名或者变量名)用于获取参数的在存储中的对齐位数。

函数限定关键字:inline:内联函数, \_Noreturn:表明函数调用完成后不返回主调函数；

**宏相关的关键字**：

#line:用于修改行号和文件名；#error：让预处理翻出一条错误信息；

#pragma:用于设定编译器

\_Generic() ：用于泛型选择，依据类型选择特定的值；

…和\_\_VA\_ARGS\_\_:用于变参宏

**流程关键字**：for, do, while, if, else, switch, case, break, continue, goto

**结构与联合**：struct, union

**枚举**：enum

**C常用库**：stdio.h, stdlib.h, math.h, stdbool.h, assert.h, complex.h, ctype.h, errno.h, fenv.h, float.h, inttypes.h, iso646.h, locale.h, setjmp.h, signal.h, stdalign.h, stdarg.h, stdatomic.h, stddef.h, stdint.h, stdnoreturn.h, string.h, tgmath.h, thread.h, time.h, uchar.h, wchar.h, wctype.h,

**表达式、语句和流程**