

****SHANGHAI JIAO TONG UNIVERSITY

**C程序设计语言**

目 录

[1 基本概念 2](#_Toc51770396)

[2 类型、运算符和表达式（位运算比较有趣） 2](#_Toc51770397)

[3 控制流 2](#_Toc51770398)

[4 指针与数组 4](#_Toc51770399)

# 基本概念

# 类型、运算符和表达式（位运算比较有趣）

1. 外部变量与static初始化为0
2. atoi：数字字符串转换为数字
3. 隐式算数转换会进行低转高，但float不会自动转为double
4. (double)a 强制类型转换。若函数形参为float，也会对传入的int数进行强制类型转换
5. &用于屏蔽某些位，flags&0x111可以屏蔽除最低三位外的数字；|可以置位，flags|1可以即最低位置1；~（求反码，1变0,0变1）可以清除所需的位，比如flags&~1111可以清除1234位数字，变为0。

eg:getbits(unsigned int x,ingt p,int n)//将x从p位置开始的n位取出对齐

{return (x>>(p-n+1)) &~(~0<<n);}前面一部是是将目标位移到右边，后一部分是将原数前面的所有数置0

x&=(x-1)可以把x的最右边的一个1去掉，0x11100-1=0x11011,0x11100&0x11011=0x11000即将最后一位去掉

改写：

int bitcounts(unsigned x)//统计其中1的数量

{

int b=0;

while(x!=0)

{

x&=(x-1);

++b;

}

}

# 控制流

1. if的嵌套尽量用{}
2. ,表达式，for循环中经常使用 for(i=0;j=1;j<5;++j,++i)
3. goto语句转向某个位置，对于退出深层嵌套很有用

for(;;)

for(;;)

{if() goto found;}//退出两层for循环

found:

{}

1. 不允许函数之内定义函数
2. extern double val[]只是声明一个外部数组，但是并不分配内存，所以不需要说明数组规模
3. 外部static常用与说明变量和说明函数，静态函数只能用在函数说明所在的文件之内。内部static变量只属于特定函数，不管函数是否调用都存在
4. 寄存器变量register提醒编译程序所在变量所在频率高，则将其存在寄存器里面，访问更快。只适用于自动变量以及函数的形参。不论寄存器变量是不是存在寄存器里面，其地址都不能访问
5. 初始化：
   1. 没有显示初始化时，外部变量和static初始化为0，自动变量和register没初始化（初值是“垃圾”）。
   2. 外部变量和register的初始化符都必须是常量表达式，初始化只做一次
   3. 自动变量和register初始化符可为常量表达式或者函数调用等任意表达式
   4. int a[]={1,2,3}

int b[3]={1}其他为0

char pa[]=”he” char pa[]={‘h’,’e’,’\0’}结束符占一位

1. #define 名字 替换文本

替换文本可以为多行，但是每行末尾加\

变元做多次操作得小心： #define square(x) x\*x（×） 调用square(z+1)出问题

1. #undef 标识符 后面取消以前定义的宏定义，后面则不能调用这个宏，如果标识符未被定义为宏，则忽略此语句
2. 宏定义的形式参数不能用带引号的字符串替换。但是，如果在替换文本中以#为前缀，那么其将被由实际边缘替换的参数扩展成待引号的字符串。例如：

#define dprint(expr) printf(#expr “=%g\n”,expr)

调用dprint(x/y)，其会被扩展printf(“x/y””=%g\n”,x/y),即为printf(“x/y=%g\n”,x/y)

1. 条件包含：

在预处理阶段，可以进行条件控制。例如下面代码可以根据电脑版本包含不同的头文件

#if SYSTEM==SYSV

#define HDR “sysv.h”

#elif SYSYTEM==BSD

#define HDR “bsd.h”

#else

#define HDR “default.h”

#endif

#include HDR

需要测试一个文件是否已经定义

#ifndef HDR

#define HDR

#endif

# 指针与数组

1. 取址&只用于内存中的对象，不能操作表达式、常量、寄存器变量
2. 指针运算比数组下标运算更快，

int a[10];

int \*pa;

pa=a;//pa=&a[0]

\*(pa+1);//a[1],\*(a+1)

但是数组和指针还是有区别，指针是变量，所以pa=a和pa++合法，但是数组名称不是变量，所以a=pa，a++非法

1. 所有的指针运算都会自动考虑它所指向的对象的大小

有效指针运算：同类型指针之间赋值；指针值加减整数；指向同组数组元素的指针建的减或者比较比较运算；指针赋0或与0做比较。其他都非法

int a[10];

int \*p=a,\*q=a+6;

q-p+1表示位于pq之间的元素个数

1. 字符指针与数组

char amessage[]=”hello”;

char \*pa=amessage;//pa指向amessage的首地址，并没有复制字符串

1. 指针数组与指向指针的指针

char \*pa[10];//是一个有10个元素（假如为文本）的一维数组，且数组中每个元素都是指向char型字符的指针。如果我们要交换实际的文本行的时候，只需要交换指针里面的值即可，而不必实际去移动真正的文本。

指针数组的每行可以有不同的长度，比如字符串指针数组，每一个指向的字符串长度都可以不一样。

1. 二维数组：

int a[2][3];

F(int a[2][3]);//F(int a[][3];F((\*a)[13])

必须要括号，否则\*a[13]表示有13个指针的指针数组，因为[]优先级比\*高

1. 指向函数的指针

任何类型的指针都可以转换为void\*，并且当它转换回原来的类型时不会丢失信息。

char \*pa[]; void \*pa[]

int (\*comp)(void\*,void\*；表示comp是一个指向函数的指针，\*comp是一个函数，参数为两个void\*；这个括号很重要，否则变成返回一个指向int类型的指针的函数

调用：(\*comp)(v[i],v[j])

1. int (\*daytab)[13];//指向由13个整数类型元素组成的一维数组的指针

int \*daytab[13] 由13个指向证书类型对象的指针组成的一维数组

char (\*(\*x())[])()

x:返回值为指向一维数组的指针的函数，该一维数组由指向返回字符类型

char (\*(\*x[3])())[5]

x:由3个指向函数的指针组成的一维数组，该函数返回指向由5个字符组成的一维数组的指针、

# 结构

1. struct point{

int x;

int y;

};

定义对象：struct point p1{1,1};（struct必须写）

struct point \*pa;

1. 结构体成员运算符(. ->)函数调用的()以及下标[]优先级最高。