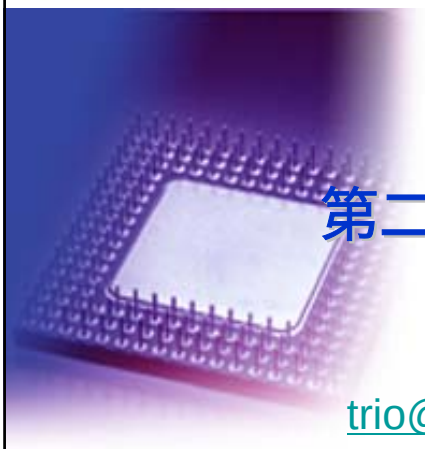
CNASIC



第二讲 C语言复习

凌 明

trio@seu.edu.cn

东南大学国家专用集成电路系统工程技术研究中心

CNASIC

还是先复习一下C吧！

www.cnasic.com

CNASIC

目　录

- C概述
- 数据类型、运算符、与表达式
- 逻辑运算和判断选取控制
- 循环控制
- 函数
- 预编译处理
- 指针
- 位运算

CNASIC

C的历史

- 在C语言诞生以前，系统软件主要是用汇编语言编写的。由于汇编语言程序依赖于计算机硬件，其可读性和可移植性都很差；但一般的高级语言又难以实现对计算机硬件的直接操作（这正是汇编语言的优势），于是人们盼望有一种兼有汇编语言和高级语言特性的新语言
- Ｃ语言是贝尔实验室于70年代初研制出来的，后来又被多次改进，并出现了多种版本。80年代初，美国国家标准化协会（ANSI），根据Ｃ语言问世以来各种版本对Ｃ语言的发展和扩充，制定了ANSI C标准（1989年再次做了修订）

CNASIC

C的特点

- ①语言简洁、紧凑，使用方便、灵活。共有３２个关键字，９种控制语句。
- ②运算符丰富，公有３４种运算符。
- ③数据结构丰富，数据类型有：整型、实型、字符型、数组、指针、结构体、共用体等。
- ④具有结构化的控制语句（如if…else、while、do…while、switch、for）
- **⑤语法限制不太严格，程序设计自由度大。**
- **⑥允许直接访问物理地址，能进行位（bit）操作，可以直接对硬件操作。**
- ⑦生成目标代码质量高，程序执行效率高。
- 可移植性好。

CNASIC

C的关键字

C语言的关键字共有32个：

（1）数据类型关键字（12个）：char, double, enum, float, int, long, short, signed, struct, union, unsigned, void

（2）控制语句关键字（12个）：break, case, continue, default, do, else, for, goto, if, return, switch, while

（3）存储类型关键字（4个）：auto, extern, register, static

（4）其它关键字（4个）：const, sizeof, typedef, volatile

CNASIC

C的语句

- 与其它高级语言一样，C语言也是利用函数体中的可执行语句，向计算机系统发出操作命令。按照语句功能或构成的不同，可将C语言的语句分为五类。

1.控制语句

控制语句完成一定的控制功能。C语言只有9条控制语句，又可细分为三种：

（1）选择结构控制语句
if()～else～, switch()～

（2）循环结构控制语句
do～while(), for()～, while()～, break, continue

（3）其它控制语句
goto, return

CNASIC

C的语句

2. 函数调用语句

函数调用语句由一次函数调用加一个分号（语句结束标志）构成。

例如，printf("This is a C function statement.");

3. 表达式语句

表达式语句由表达式后加一个分号构成。最典型的表达式语句是，在赋值表达式后加一个分号构成的赋值语句。

例如，“num=5 ”是一个赋值表达式，而“num=5;”却是一个赋值语句。

CNASIC

C的语句

4. 空语句

空语句仅由一个分号构成。显然，空语句什么操作也不执行。

5. 复合语句

复合语句是由大括号括起来的一组（也可以是1条）语句构成。例如：

```
main()
    { ……
        {……} /*复合语句。注意：右括号后不需要分号。*/
      ……
      }
```

复合语句的性质：

（1）在语法上和单一语句相同，即单一语句可以出现的地方，也可以使用复合语句。

（2）复合语句可以嵌套，即复合语句中也可出现复合语句。

CNASIC

C语言程序的总体结构

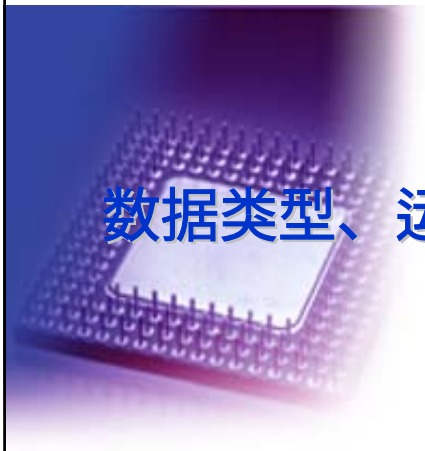
- 在C语言中，除实现*顺序*、*选择*和*循环*三种基本结构等的9条控制语句外，输入输出操作均由标准库函数（不是C语言的组成部分）来实现。
- 一个完整的C语言程序，是由一个main()函数（又称主函数）和若干个其它函数结合而成的，或仅由一个main()函数构成。

CNASIC

C程序的文件结构

- 一个基于C的软件系统一般由以下的文件构成：
  - 若干个C文件，每个C文件中包含若干个函数；
  - 若干个头文件，每个头文件包括一些数据结构的定义以及C函数的原型；
  - 若干个库文件，库文件是编译后的二进制文件，一般由若干个C文件编译后组成，其中包含了若干个函数的可执行代码，这些库文件中的函数可执行代码在链接的时候合并到最终的可执行文件中。
  - 一个或多个MAKE文件，Make文件描述了多个文件中的依赖关系以及生成最终可执行文件或库文件所需要的信息

CNASIC



数据类型、运算符、与表达式

- 1． C的数据类型

C的数据类型包括：整型、字符型、实型或浮点型（单精度和双精度）、枚举类型、数组类型、结构体类型、共用体类型、指针类型和空类型。

2． 常量与变量

常量其值不可改变，符号常量名通常用大写。变量其值可以改变，变量名只能由字母、数字和下划线组成，且第一个字符必须为字母或下划线。否则为不合法的变量名。变量在编译时为其分配相应存储单元。

3． 整型数据

整型常量的表示方法：十进制不用说了，八进制以0开头，如0123，十六进制以0x开头，如0x1e。
整型变量分为：基本型（int）、短整型（short int）、长整型（long int）和无符号型。
不同机器上各类数据所占内存字节数不同，一般int型为 2 个字节，long型为4个字节。

CNASIC

- 4． 实型数据

实型常量表示形式：十进制形式由数字和小数点组成（必须有小数点），如：0.12、.123、123.、0.0等。指数形式如123e3代表123×10的三次方。
实型变量分为单精度（float）和双精度（double）两类。在一般系统中float型占4字节，7位有效数字，double型占8字节，15~16位有效数字。

5． 字符型数据

字符变量用单引号括起来，如'a','b'等。还有一些是特殊的字符常量，如'\n','\t'等。分别代表换行和横向跳格。
字符变量以char 来定义，一个变量只能存放一个字符常量。
字符串常量是由双引号括起来的字符序列。这里一定要注意'a'和"a"的不同，前者为字符常量，后者为字符串常量，c规定：每个字符串的结尾加一个结束标志'\0'，实际上"a"包含两个字符：'a'和'\0'。

6． 数值型数据间的混合运算

整型、字符型、实型数据间可以混合运算，运算时不同类型数据要转换成同一类型再运算，转换规则：
char,short -> int -> unsigned -> long -> double <- float

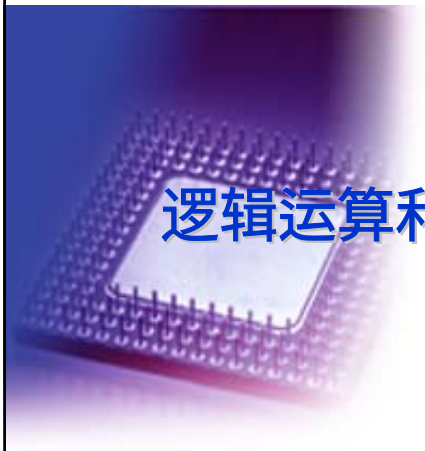
www.cnasic.com

CNASIC

- 7． 运算符和表达式

算数运算符（ + - * / % ）
关系运算符（> < == >= <= != ）
逻辑运算符（! && || ）
位运算符（<< >> ~ | ^ & ）
赋值运算符（ = ）
条件运算符（ ? : ）
逗号运算符（ , ）
指针运算符（ * & ）
求字节数（ sizeof ）
强制类型转换（类型）
分量运算符（ . -> ）
下标运算符（ [ ] ）
其它运算符（ 如函数调用运算符( ) ）
自增自减运算符（ ++ -- ）注意：++i和i++的不同之处，++i使用i之前先使i加１，i++使用i之后，使i加１。
逗号表达式的求解过程：先求解表达式１，再求解表达式２，整个表达式的值是表达式２的值。

CNASIC



逻辑运算和判断选取控制

www.cnasic.com

CNASIC

1． 关系运算符

- c提供６种关系运算符（> < <= >= == != ）前四种优先级高于后两种。
- 如果你不清楚运算符的优先级，最好的办法是－－括号！！

CNASIC

2． If语句

C提供了三种形式的if语句

- If(表达式) 语句
- If(表达式) 语句1 else 语句2
- If(表达式1) 语句1
  Else if(表达式2) 语句2
  …
  else 语句n

CNASIC

3． 条件运算符

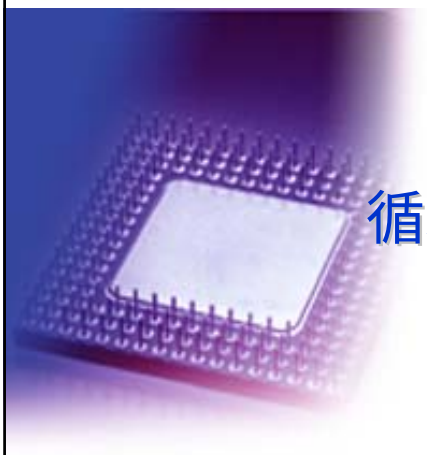
- (a>b)?a:b 条件为真，表达式取值a，否则取值b
- 编译器可以产生比if … else…更优化的编译结果！

CNASIC

4． Switch语句

```
Switch(表达式)
{
case 常量表达式１：语句１; break;
case 常量表达式２：语句2; break;
       …
case 常量表达式n：语句ｎ; break;
      default ：语句ｎ＋１;
}
```

CNASIC



循环控制

CNASIC

循环控制

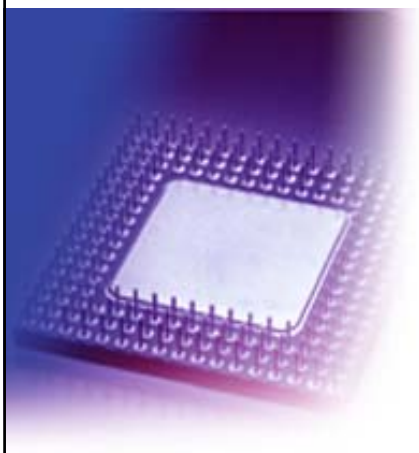
1． 几种循环语句

goto语句（现已很少使用）
while语句　先判断表达式后执行语句
do-while语句　先执行语句后判断表达式
for语句

2． Break语句和continue语句

Break语句用于跳出循环，continue用于结束本次循环。

CNASIC



函 数

CNASIC

- 1. 关于形参和实参的说明

① 在函数被调用之前，形参不占内存
② 实参可以是常量、变量或表达式
③ 必须指定形参的类型
④ 实参与形参类型应一致
⑤ 实参对形参的数据传递是“值传递”，即单向传递

2. 函数返回值

如果想让函数返回一个值，在函数中就要用return语句来获得，在定义函数时也要对函数值指定类型，如果不指定，默认返回整型。

- Int func1(int *p)

```
{
        int a, b;
        …
        *p = a;
        return b;
}

main()
{
        int c;
        func(&c);
        printf(“%d\n”, c);
}
```

- 3． 函数调用

1)注意在函数调用时实参和形参的个数、类型应一一对应。对实参表求值的顺序是不确定的，有的系统按自左至右，有的系统则按自右至左的顺序。这一点要注意。
2)函数调用的方式：函数语句，函数表达式，函数参数
3)如果主调函数和被调函数在同一文件中，并且主调函数在前，那么一般要在主调函数中对被调函数进行说明。除非：(1)被调函数的返回值类型为整型或字符型(2)被调函数出现在主调函数之前。
4)对函数的说明和定义是不同的，定义是指对函数功能的确立，包括指定函数名，函数值类型，形参及其类型、函数体等。说明则只是对已定义的函数返回值类型进行说明，只包括函数名、函数类型以及一个空的括弧，不包括形参和函数体。
5)c语言允许函数的递归调用（在调用一个函数的过程中又出现直接或间接的调用该函数本身）。

CNASIC

局部变量和全局变量

从变量作用域角度分，变量可分为局部变量和全局变量。
1)内部变量（局部变量）
在一个函数内定义，只在函数范围内有效的变量。
2)外部变量（全局变量）
在函数外定义，可以为本文件其它函数所共用，有效范围从定义变量的位置开始到本文件结束。建议尽量少使用全局变量，因为它在程序全部执行过程中都占用资源，而且使函数的通用性降低了。如果在定义外部变量之前的函数要想使用该外部变量，则应在该函数中用extern作外部变量说明。

www.cnasic.com

CNASIC

动态存储变量与静态存储变量

从变量值存在的时间（生存期）角度来分，可分为静态存储变量和动态存储变量。静态存储指在程序运行期间给变量分配固定的存储空间，动态存储指程序运行期间根据需要动态的给变量分配存储空间。

C语言中，变量的存储方法分为两大类：

静态存储类和动态存储类，具体包括：自动的（auto），静态的(static)，寄存器的(register)，外部的(extern)。

1) 局部变量的存储方式

函数中的局部变量如不作专门说明，都之auto的，即动态存储的，auto可以省略。局部变量也可以定义为static的，这时它在函数内值是不变的。静态局部变量如不赋初值，编译时系统自动赋值为0，动态局部变量如不赋初值，则它的值是个不确定的值。C规定，只有在定义全局变量和局部静态变量时才能对数组赋初值。为提高执行效率，c允许将局部变量值放在寄存器中，这种变量叫register变量，要用register说明。但只有局部动态变量和形式参数可以作为register变量，其它不行。

2) 全局变量的存储方式

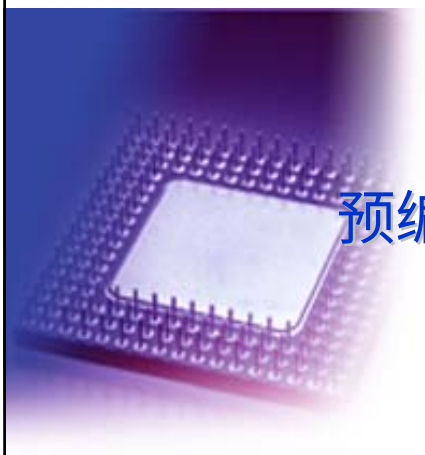
全局变量在函数外部定义，编译时分配在静态存储区，可以在程序中各个函数所引用。多个文件的情况如何引用全局变量呢？假如在一个文件定义全局变量，在别的文件引用，就要在此文件中用extern对全局变量说明，但如果全局变量定义时用static的话，此全局变量就只能在本文件中引用了，而不能被其它文件引用。

CNASIC

内部函数和外部函数

- 内部函数：只能被本文件中的其它函数调用，定义时前加static，内部函数又称静态函数。
- 外部函数：可以被其它文件调用，定义时前加extern，如果省略，则隐含为外部函数，在需要调用此函数的文件中，一般要用extern说明。

CNASIC



预编译处理

CNASIC

预处理

- c编译系统在对程序进行通常的编译之前，先进行预处理。c提供的预处理功能主要有以下三种：
  - 1）宏定义
  - 2）文件包含
  - 3）条件编译

CNASIC

1． 宏定义

- 不带参数的宏定义
  用一个指定的标识符来代表一个字符串，形式：#define 标识符 字符串
  几点说明：
  1） 宏名一般用大写
  2） 宏定义不作语法检查，只有在编译被宏展开后的源程序时才会报错
  3） 宏定义不是c语句，不在行末加分号
  4） 宏名有效范围为定义到本源文件结束
  5） 可以用#undef命令终止宏定义的作用域
  6） 在宏定义时，可以引用已定义的宏名

CNASIC

带参数的宏定义

- 定义形式：#define 宏名（参数表） 字符串
  这和函数有些类似，但他们是不同的：
  1） 函数调用时，先求实参表达式值，再代入形参，而宏只是简单替换，并不求值
  2） 函数调用是在程序运行时分配内存的，而宏展开时并不分配内存，也没有返回值的概念
  3） 对函数中的实参和形参都要定义类型，而且要求一致，宏名无类型，其参数也没有类型。
  4） 函数只有一个返回值，而宏可以得到几个结果
  5） 宏替换不占运行时间，只占编译时间，而函数调用占运行时间

CNASIC

2． 文件包含处理

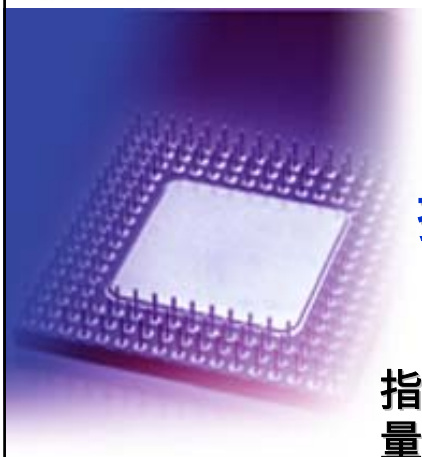
- #include “文件1” 就是将文件1的全部内容复制插入到#include位置，作为一个源文件进行编译。
- 在#include命令中，文件名可以用" "也可以用< >，假如现在file1.c中包含file2.h文件，" "表示系统先在file1.c所在目录中找file2.h，如果找不到，再按系统指定的标准方式检索目录，< >表示系统直接按指定的标准方式检索目录。所以用" "保险一点。

CNASIC

3． 条件编译

- 条件编译指不对整个程序都编译，而是编译满足条件的那部分。条件编译有以下几种形式：
- 1）#ifdef 标识符
  程序段１
  #else
  程序段２
  #endif
  它的作用：当标识符在前面已经被定义过（一般用#define），则对程序段１编译，否则对程序段２编译。
- ２）#ifndef 标识符
  程序段１
  #else
  程序段２
  #endif
  它的作用和#ifdef相反，当标识符没被定义过，对程序段１编译，否则对程序段２编译。
- ３）#if 表达式
  程序段１
  #else
  程序段２
  #endif
  它的作用：当表达式值为真（非０）时，对程序段１编译，否则对程序段２编译。

CNASIC



指 针

指针说白了就是地址。指针变量就是用来存放指针（地址）的变量。

CNASIC

变量的指针和指向变量的指针变量

- 读起来很拗口，说白了就是变量的地址和用来存放变量地址的地址变量。因为一个变量在编译的时候系统要为它分配一个地址，假如再用一个变量来存放这个地址，那么这个变量就叫做指向变量的指针变量，也就是用来存放变量地址的这么一个变量。所谓"指向"就是指存放××的地址，如指向变量的指针变量，"指向"就是指用来存放变量的地址，再如指向数组的指针变量，"指向"就是指存放数组的地址。只要理解了这个，指针也就不难了。另外，还有指向字符串的指针变量，指向函数的指针变量，指向指针的指针变量等。
  - 1) 指针变量的定义
    形式：类型标识符 *标识符 如：int *pointer;
    要注意两点：*表示pointer是个指针变量，在用这个变量的时候不能写成*pointer， *pointer是pointer指向的变量。一个指针变量只能指向同一个类型的变量。如上面pointer只能指向int型变量。
  - 2)指针变量的引用
    两个有关的运算符：
    & 取地址运算符 &a 就代表变量a的地址
    * 指针运算符 *a 就代表变量a的值

www.cnasic.com

CNASIC

数组的指针和指向数组的指针变量

- 数组的指针指数组的起始地址，数组元素的指针指数组元素的地址。
  1)指向数组元素的指针变量的定义与赋值
  定义和指向变量的指针变量定义相同，c规定数组名代表数组的首地址，即第一个数组元素地址。
  2)通过指针引用数组元素
  我们通常引用数组元素的形式是a[i]，如果用指针可以这样引用，*(a+i)，或定义一个指针变量p，将数组a的首地址赋给p，p=a;然后用*(p+i)引用。
  注意：指针变量 p 指向数组a首地址，则p++指向数组 a 的下一元素地址，即a[1]的地址。

CNASIC

字符串的指针和指向字符串的指针变量

- 1)字符串的表示形式
  c中字符串有两种表示形式：一种是数组，一种是字符指针
  char string[]="I love c!";
  char *str="I love c!";
  其实指针形式也是在内存中开辟了一个数组，只不过数组的首地址存放在字符指针变量str中，千万不要认为str是一个字符串变量。
  2)字符串指针作函数参数
  实际上字符串指针就是数组的首地址。

- 3)字符指针变量与字符数组的区别
  ① 字符数组由若干元素组成，每个元素存放一个字符，而字符指针变量只存放字符串的首地址，不是整个字符串
  ② 对数组初始化要用static，对指针变量不用。
  ③ 对字符数组赋值，只能对各个元素赋值，不能象下面这样：
  char str[14];
  str="I love c!";
  对指针变量可以，
  char *str;
  str="I love c!";
  注意：此时赋给str的不是字符，而是字符串首地址。

CNASIC

- ④ 数组在定义和编译时分配内存单元，而指针变量定义后最好将其初始化，否则指针变量的值会指向一个不确定的内存段，将会破坏程序。如：
  char *a;
  scanf("%s", a);这种方法是很危险的，应该这样：
  char *a, str[10];
  a = str;
  scanf("%s", a);这样字符指针就指向了一个确定的内存段。
  ⑤ 指针变量的值是可以改变的，而字符数组名所代表的字符串首地址却是不能改变的。

www.cnasic.com

CNASIC

函数的指针和指向函数的指针变量

一个函数在编译时被分配一个入口地址，这个入口地址就称为函数的指针。函数名代表函数的入口地址，这一点和数组一样。我们可以用一个指针变量来存放这个入口地址，然后通过该指针变量调用函数。如：假设有一个求两者较大的函数如下：int max(int x, int y);
当我们调用这个函数时可以这样：
int c;
c=max(a, b);这是通常调用方法，其实我们可以定义一个函数指针，通过指针来调用，如：
int (*p)(); //注意指向函数指针变量的定义形式
p=max; //此句就是将函数的入口地址赋给函数指针变量p
c=(*p)(a, b);
有些朋友可能对(*p)()不大理解，其实它的意思就是定义一个指向函数的指针变量p，p不是固定指向哪个函数的，而是专门用来存放函数入口地址的变量。在程序中把哪个函数的入口地址赋给它，它就指向哪个函数。但要注意，p不能象指向变量的指针变量一样进行p++,p-等无意义的操作。
既然p是一个指针变量，那么就可以作为函数的参数进行传递。其实函数的指针变量最常用的用途之一就是作为函数参数传递到其它函数。这也是c语言中应用的比较深入的部分了。

CNASIC

返回指针值的函数

- 我们知道，一个函数可以带回一个整型值、字符值、实型值等，函数还可以带回一个指针型的数据，即地址。这种函数的定义形式如下：
  类型标识符 *函数名(参数表) 如：int *a(x,y)返回一个指向整型的指针
- 使用这种函数的时候要注意：在调用时要先定义一个适当的指针来接收函数的返回值。这个适当的指针其类型应为函数返回指针所指向的类型。
  这样的函数比较难于理解，其实只要把它当做一般的函数来处理就容易了。当我们觉得指针难于理解的时候，就把它暂时当做整型来看，就好理解多了。

CNASIC

指针数组

- 指针数组无疑就是数组元素为指针，定义形式为：类型标识　*数组名[数组长度]
如:int *p[4]，千万不要写成int (*p)[4]，这是指向一维数组的指针变量。指针数组多用于存放若干个字符串的首地址，注意一点，在定义指针数组时初始化，如下：
static char *name[]={"Li jing","Wang mi","Xu shang"};
不要以为数组中存放的是字符串，它存放的是字符串首地址，这一点一定要注意。

CNASIC

指向指针的指针

- 说的明白一点，将一个指针再用一个变量来存放，那么这个变量就是指向指针的指针。定义如：char * *p;

CNASIC

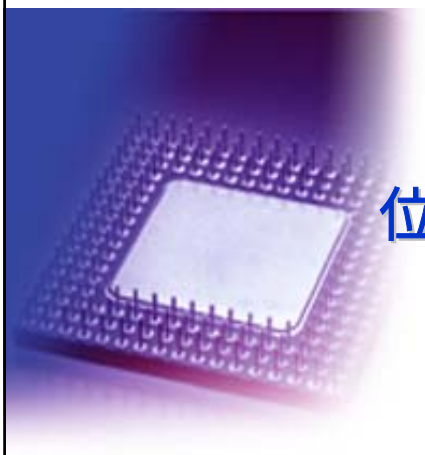
Void 指针

- ANSI新增了一种void *指针类型，即定义一个指针变量，但不指向任何数据类型，等用到的时候再强制转换类型。如：
  char *p1;
  void *p2;
  p1 = (char *)p2;
  也可以将一个函数定义成void *型，如：
  void *fun(ch1, ch2)
  表示函数fun返回一个地址，它指向空类型，如果需要用到此地址，也要对其强制转换。如（假设p1为char型）：
  p1=(char *)fun(c1,c2);

CNASIC

- 指针应该说是c语言中比较重要的概念，也是c语言的精华，它有很多优点，但用不好也会带来严重性的错误，这就需要我们多用，多练，慢慢的积累经验。

CNASIC



位 运 算

CNASIC

1)概述

- 所谓位运算是指进行二进制位的运算。在系统软件中，常要处理二进制位的问题。
c提供的位运算符有：
& 按位与
| 按位或
^ 按位异或
~ 取反
<< 左移
>> 右移
&对于将一个单元清零、取一个数中的某些指定位以及保留指定位有很大用途。
|常被用来将一个数的某些位置1。
^判断两个位值，不同为1，相同为0。常用来使特定位翻转等。
~常用来配合其它位运算符使用的，常用来设置屏蔽字。
<<将一个数的各二进制位全部左移，高位左移后溢出，舍弃不起作用。左移一位相当于该数乘2，左移n位相当于乘2n。左移比乘法运算要快的多。
>>右移时，要注意符号问题。对无符号数，右移时左边高位移入0，对于有符号数，如果原来符号位为0（正数），则左边移入0；如果符号位为1（负数），则左边移入0还是1要取决于系统。移入0的称为"逻辑右移"，移入1的称为"算数右移"。

CNASIC

2)位段

- 将一个字节分为几段来存放几个信息。所谓位段是以位为单位定义长度的结构体类型中的成员。如：
struct packed-data{
unsigned a:2;
unsigned b:6;
unsigned c:4;
unsigned d:4;
int I;
}data;
其中a,b,c,d分别占2位,6位,4位,4位。I为整型，占4 个字节。
对于位段成员的引用如下：
data.a = 2;等，但要注意赋值时，不要超出位段定义的范围。如位段成员a定义为2位，最大值为3，即(11)2，所以data.a=5;就会取5的两个低位进行赋值，就得不到想要的值了。
关于位段的定义和引用，有几点重要说明：
①若某一个段要从另一个字开始存放，可以定义：
unsigned a:1;
unsigned b:2;
unsigned :0;
unsigned c:3; (另一单元)
使用长度为0的位段，作用就是使下一个位段从下一个存储单元开始存放。
②一个位段必须存放在用一个存储单元中，不能跨两个单元。
③可以定义无名位段。如：
unsigned a:1;
unsigned :2; (这两位空间不用)
unsigned b:3;
④位段的长度不能大于存储单元的长度，也不能定义位段数组。