# C51如何使用指针变量

来源：网络 作者：无殇蝶舞 更新时间：2024-08-24

*第一篇：C51如何使用指针变量MOVX 就是用来读写外部RAM的MOVC 读写外部ROM可通过检测RI来判断RXD是否有信息C语言参考资料：指针类型和存储区的关系详解一、存储类型与存储区关系data- 可寻址片内rambdata- 可位寻...*

**第一篇：C51如何使用指针变量**

MOVX 就是用来读写外部RAM的MOVC 读写外部ROM

可通过检测RI来判断RXD是否有信息

C语言参考资料：

指针类型和存储区的关系详解

一、存储类型与存储区关系

data---> 可寻址片内ram

bdata---> 可位寻址的片内ram

idata---> 可寻址片内ram，允许访问全部内部ram

pdata---> 分页寻址片外ram(MOVX @R0)(256 BYTE/页)xdata---> 可寻址片外ram(64k 地址范围)

code---> 程序存储区(64k 地址范围),对应MOVC @DPTR

二、指针类型和存储区的关系

对变量进行声明时可以指定变量的存储类型如:

uchar data x和data uchar x相等价都是在内ram区分配一个字节的变量。

同样对于指针变量的声明，因涉及到指针变量本身的存储位置和指针所指向的存储区位置不同而进行相应的存储区类型关键字的使用如：

uchar xdata \* data pstr

是指在内ram区分配一个指针变量(“\*”号后的data关键字的作用)，而且这个指针本身指向xdata区(“\*”前xdata关键字的作用)，可能初学C51时有点不好懂也不好记。没关系，我们马上就可以看到对应“\*”前后不同的关键字的使用在编译时出现什么情况。

......uchar xdata tmp[10];//在外ram区开辟10个字节的内存空间，地址是外ram的0x0000－0x0009......第1种情况:

uchar data \* data pstr;pstr=tmp;

首先要提醒大家这样的代码是有bug的, 他不能通过这种方式正确的访问到tmp空间。为什么？我们把编译后看到下面的汇编

代码：

MOV 0x08,#tmp(0x00);0x08是指针pstr的存储地址

看到了吗！本来访问外ram需要2 byte来寻址64k空间，但因为使用data关键字(在“\*”号前的那个)，所以按KeilC编译环境来说

就把他编译成指向内ram的指针变量了，这也是初学C51的朋友们不理解各个存储类型的关键字定义而造成的bug。特别是当工程中的默认的存储区类为large时，又把tmp[10] 声明为uchar tmp[10] 时，这样的bug是很隐秘的不容易被发现。

第2种情况:

uchar xdata \* data pstr;pstr = tmp;

这种情况是没问题的，这样的使用方法是指在内ram分配一个指针变量(“\*”号后的data关键字的作用)，而且这个指针本身指向

xdata区(“\*”前xdata关键字的作用)。编译后的汇编代码如下。

MOV 0x08,#tmp(0x00);0x08和0x09是在内ram区分配的pstr指针变量地址空间

MOV 0x09,#tmp(0x00)

这种情况应该是在这里所有介绍各种情况中效率最高的访问外ram的方法了，请大家记住他。

第3种情况:

uchar xdata \* xdata pstr;pstr=tmp;

这中情况也是对的，但效率不如第2种情况。编译后的汇编代码如下。

MOV DPTR, #0x000A;0x000A,0x000B是在外ram区分配的pstr指针变量地址空间

MOV A, #tmp(0x00)MOV @DPTR, A INC DPTR

MOV A, #tmp(0x00)MOVX @DPTR, A

这种方式一般用在内ram资源相对紧张而且对效率要求不高的项目中。

第4种情况:

uchar data \* xdata pstr;pstr=tmp;

如果详细看了第1种情况的读者发现这种写法和第1种很相似，是的，同第1 种情况一样这样也是有bug的，但是这次是把pstr分

配到了外ram区了。编译后的汇编代码如下。

MOV DPTR, #0x000A;0x000A是在外ram区分配的pstr指针变量的地址空间

MOV A, #tmp(0x00)MOVX @DPTR, A

第5种情况:

uchar \* data pstr;pstr=tmp;

大家注意到“\*”前的关键字声明没有了，是的这样会发生什么事呢？下面这么写呢！对了用齐豫的一首老歌名来说就是 “请跟我

来”，请跟我来看看编译后的汇编代码，有人问这不是在讲C51吗？ 为什么还要给我们看汇编代码。C51要想用好就要尽可能提升C51

编译后的效率，看看编译后的汇编会帮助大家尽快成为生产高效C51代码的高手的。还是看代码吧！

MOV 0x08, #0X01;0x08－0x0A是在内ram区分配的pstr指针变量的地址空间

MOV 0x09, #tmp(0x00)MOV 0x0A, #tmp(0x00)

注意：这是新介绍给大家的，大家会疑问为什么在前面的几种情况的pstr指针变量都用2 byte空间而到这里就用3 byte空间了

呢？这是KeilC的一个系统内部处理，在KeilC中一个指针变量最多占用 3 byte空间，对于没有声明指针指向存储空间类型的指针，系统编译代码时都强制加载一个字节的指针类型分辩值。具体的对应关系可以参考KeilC的help中C51 Users Guide。

第6种情况:

uchar \* pstr;pstr=tmp;

这是最直接最简单的指针变量声明，但他的效率也最低。还是那句话，大家一起说好吗！编译后的汇编代码如下。

MOV DPTR, #0x000A;0x000A－0x000C是在外ram区分配的pstr指针变量地址空间

MOV A, #0x01 MOV @DPTR, A INC DPTR

MOV DPTR, #0x000A MOV A, #tmp(0x00)MOV @DPTR, A INC DPTR

MOV A, #tmp(0x00)MOVX @DPTR, A

这种情况很类似第5种和第3种情况的组合，既把pstr分配在外ram空间了又增加了指针类型的分辨值。

小结一下：大家看到了以上的6种情况，其中效率最高的是第2种情况，既可以正确访问ram区又节约了代码，效率最差的是第 6

种，但不是说大家只使用第2种方式就可以了，还要因情况而定，一般说来应用51系列的系统架构的内部ram资源都很紧张，最好大家

在定义函数内部或程序段内部的局部变量使用内ram，而尽量不要把全局变量声明为内ram区中。所以对于全局指针变量我建议使用第种情况，而对于局部的指针变量使用第2种方式。

C51是很灵活的，也很好理解和使用，但要成为笑傲江湖的一代高手还是要多想多练，没有实际项目的锻炼是不容易提高的。希

望这篇文章对大家一点用处

**第二篇：C语言函数指针变量**

C语言函数指针变量

在C语言中，一个函数总是占用一段连续的内存区，而函数名就是该函数所占内存区的首地址。我们可以把函数的这个首地址（或称入口地址）赋予一个指针变量，使该指针变量指向该函数。然后通过指针变量就可以找到并调用这个函数。我们把这种指向函数的指针变量称为函数指针变量。

函数指针变量定义的一般形式为：

类型说明符(\*指针变量名)();其中类型说明符表示现金网96y.org被指函数的返回值的类型。(\* 指针变量名)表示\*后面的变量是定义的指针变量。最后的空括号表示指针变量所指的是一个函数。例如：

int(\*pf)();

表示pf是一个指向函数入口的指针变量，该函数的返回值(函数值)是整型。

本例用来说明用指针形式实现对函数调用的方法。

int max(int a,int b){

if(ab)return a;

else return b;

}

main(){

int max(int a,int b);

int(\*pmax)();

int x,y,z;

pmax=max;

printf(input two numbers:n);

scanf(%d%d,x,y);

z=(\*pmax)(x,y);

printf(maxmum=%d,z);

}

从上述程序可以看出用，函数指针变量形式调用函数的步骤如下：

先定义函数指针变量，如后一程序中第9行 int(\*pmax)();定义 pmax为函数指针变量。

把被调函数的入口地址（函数名）赋予该函数指针变量，如程序中第11行 pmax=max;

用函数指针变量形式调用函数，如程序第14行 z=(\*pmax)(x,y);

调用函数的一般形式为：(\*指针变量名)(实参表)

使用函数指针变量还应注意以下两点：

函数指针变量不能进行算术运算，这是与数组指针变量不同的。数组指针变量加减一个整数可使指针移动指向后面或前面的数组元素，而博彩娱乐棋牌q07i5.org函数指针的移动是毫无意义的。

函数调用中(\*指针变量名)的两边的括号不可少，其中的\*不应该理解为求值运算，在此处它只是一种表示符号。

**第三篇：C语言结构体指针变量作函数参数**

C语言结构体指针变量作函数参数

在ANSI C标准中允许用结构变量作函数参数进行整体传送。但是这种传送要将全部成员逐个传送，特别是成员为数组时将会使传送的时间和空间开销很大，严重地降低了程序的效率。因此最好的办法就是使用指针，即用指针变量作函数参数进行传送。这时现金网96y.org由实参传向形参的只是地址，从而减少了时间和空间的开销。

计算一组学生的平均成绩和不及格人数。用结构指针变量作函数参数编程。

struct stu{

int num;

char \*name;

char sex;

float score;

}boy[5]={

{101,Li ping,M,45},{102,Zhang ping,M,62.5},{103,He fang,F,92.5},{104,Cheng ling,F,87},{105,Wang ming,M,58}

};

main(){

struct stu \*ps;

void ave(struct stu \*ps);

ps=boy;

ave(ps);

}

void ave(struct stu \*ps){

int c=0,i;

float ave,s=0;

for(i=0;i5;i++,ps++){

s+=ps-score;

if(ps-score60)c+=1;

}

printf(s=%fn,s);

ave=s/5;

printf(average=%fncount=%dn,ave,c);

}

本程序中定义了函数ave，其形参为结构指针变量ps。boy被定义为外部结构数组，因此在整个源程序中有效。在main函数中定义说明了结构指针变量ps，并把boy的首地址赋予它，使ps指向boy数组。然后博彩娱乐棋牌q07i5.org以ps作实参调用函数ave。在函数ave中完成计算平均成绩和统计不及格人数的工作并输出结果。

由于本程序全部采用指针变量作运算和处理，故速度更快，程序效率更高。

**第四篇：C语言字符串指针变量与字符数组的区别**

C语言字符串指针变量与字符数组的区别

用字符数组和字符指针变量都可实现字符串的存储和运算。但是两者是有区别的。在使用时应注意以下几个问题：

1)字符串指针变量本身是一个变量，用于存放字符串的首地址。而字符串本身是存放在以该首地址为首的一块连续的内存空间中并以作为串的结束。字符数组现金网96y.org是由于若干个数组元素组成的，它可用来存放整个字符串。

2)对字符串指针方式

char \*ps=C Language;

可以写为：

char \*ps;

ps=C Language;

而对数组方式：

static char st[]={C Language};

不能写为：

char st[20];

st={C Language};

而只能对字符数组的各元素逐个赋值。

从以上几点可以看出字符串指针变量与字符数组在使用时的区别，同时也可看出使用指针变量更加方便。

前面说过，当一个指针变量在未取得确定地址前使用是危险的，容易引起错误。但是博彩娱乐棋牌q07i5.org对指针变量直接赋值是可以的。因为C系统对指针变量赋值时要给以确定的地址。

因此，char \*ps=C Langage;

或者

char \*ps;

ps=C Language;

都是合法的。

**第五篇：Oracle变量的定义、赋值及使用**

一、常用标量类型 1.VARCHAR2(N)2.CHAR(N)3.NUMBER(P,S)4.DATE 5.TIMESTAMP 该数据类型是9i时增加的，它也用于定义日期和时间数据。给它赋值的方法与给DATE变量赋值的方法完全相同。但当显示时它不仅会显示日期，还会显示时间和上下午标记。

6.LONG和LONG RAW 7.BOOLEAN 8.BINARY\_INTEGER 9.BINARY\_FLOAT 和 BINARY DOUBLE 这两个类型是ORACLE10新增加的 用的时候不是很多

二、使用标量变量

在PL/SQL中为变量赋值不同于其他编程语言，必须要在等号之前加上冒号（:=）。eg: DECLARE x NUMBER;y NUMBER;

本文档由站牛网zhann.net收集整理，更多优质范文文档请移步zhann.net站内查找