北京交通大学
BEIJING JIAOTONG UNIVERSITY

结构体指针

结构体指针

- 在C语言中几乎可以创建指向任何类型的指针，包括用户自定义的类型。

```
typedef struct
{
   char name[21];
   char city[21];
   char college[3];
} Rec;

typedef Rec *RecPointer;


Rec          x;
Rec         *y;
RecPointer r;
```

typedef = type define
C语言的关键字，作用是为一种数据类型定义一个新名字。

Rec是一种数据类型，结构体类型

RecPointer是一种数据类型，结构体指针类型

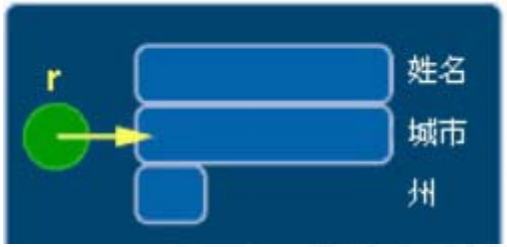
x是结构体变量；
y和**r**都是结构体指针变量；

结构体指针

- r是一个指向结构体的指针。
  - 请注意，因为r是一个指针，所以像其他指针一样占用4个字节的内存。而malloc语句会从堆上分配45字节的内存。
- *r是一个结构体，像任何其他Rec类型的结构体一样。

```
RecPointer r;

r=(RecPointer)malloc(sizeof(Rec));
```



结构体指针

- r是一个指向结构体的指针。
  - 请注意，因为r是一个指针，所以像其他指针一样占用4个字节的内存。而malloc语句会从堆上分配45字节的内存。
- *r是一个结构体，像任何其他Rec类型的结构体一样。
  - r->这种写法和(*r).是完全等效的

```
RecPointer r;

r=(RecPointer)malloc(sizeof(Rec));

strcpy((*r).name, "Leigh");
strcpy((*r).city, "Raleigh");
strcpy((*r).state, "NC");
printf("%sn", (*r).city);
free(r);
```

= strcpy (r->name, “Leigh”);

方便起见

“.”操作符的优先级高于“*”操作符，*r两边必须加括号。

包含指针的结构体

- 结构体成员可以是指针，甚至可以是本结构体类型的指针。

```
1  typedef  struct  tagNode
2  {
3
4  char*  pItem;
5
6  pNode*  pNext;
7
8  }pNode;
```

3)、规范做法：

```
1  struct  tagNode
2  {
3
4  char*  pItem;
5
6  struct  tagNode*  pNext;
7  };
8
9  typedef  struct  tagNode*  pNode;
```

```
1  typedef  struct  tagNode
2  {
3
4
5  char*  pItem;
6
7
8  struct  tagNode*  pNext;
9
10 }*pNode;
```

```
1  typedef  struct  tagNode*  pNode;
2
3
4  struct  tagNode
5  {
6
7  char*  pItem;
8
9  pNode  pNext;//这边不用pNode* ，pNode 已经表示了struc
10 };
```

typedef和#define

- typedef是C语言的关键字；#define是宏定义

```
1  typedef  char*  pStr1;
2
3  #define  pStr2  char*
4
5  pStr1  s1,s2;
6
7  pStr2  s3,s4;
```

在上述的变量定义中，s1、s2、s3都被定义为char *，而s4则定义成了char，不是我们所预期的指针变量。

- typedef是给数据类型起别名；#define是简单的字符串替换。