中国人民大学
RENMIN UNIVERSITY OF CHINA

第9讲 指针

余力

buaayuli@ruc.edu.cn

中国人民大学
RENMIN UNIVERSITY OF CHINA

C程序设计

第9讲 指针

余力

buaayuli@ruc.edu.cn

内容

1 指针基本概念

2 指针与数组

3 指针与字符串

4 指针与函数

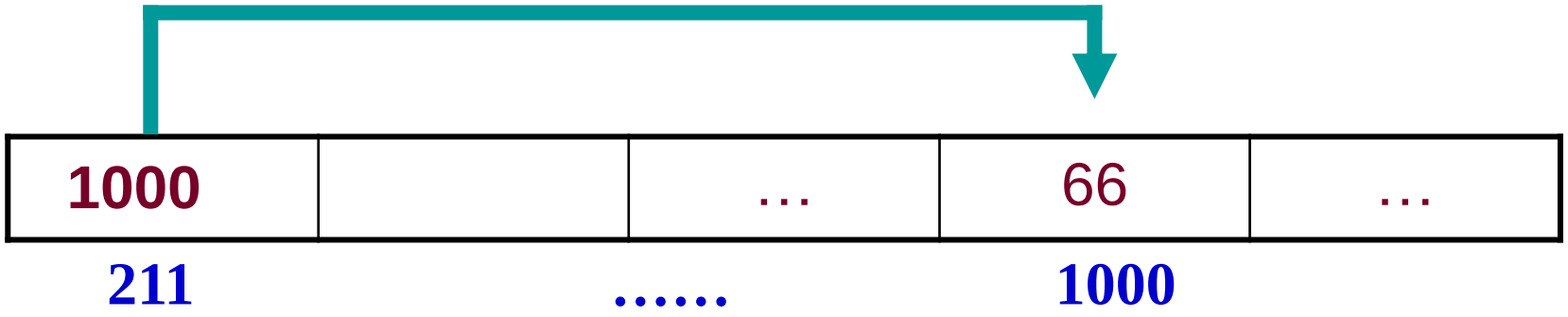
5 指针与结构体

中国人民大学
RENMIN UNIVERSITY OF CHINA

1. 指针的基本概念

什么是指针

- 指针是一类变量
  - 学过的变量：整型int、字符型char，等等
  - 指针也是一类变量，本质并无不同
- 指针的取值是内存的地址



指针的定义与赋值

- 将一个内存地址装入指针变量

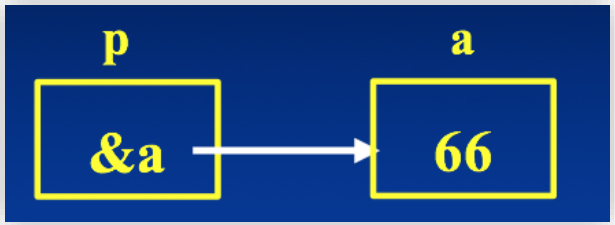
取址运算符&

- 例如：

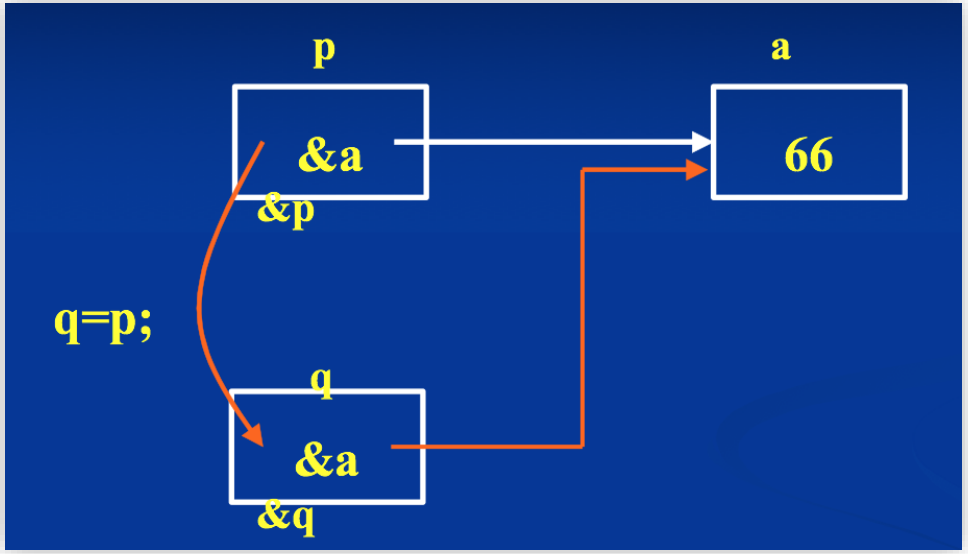
```
int a=66;                    // 定义整型变量 a, 赋初值66
// 定义p,q指针变量，赋初值为0
int *p=NULL, * q=NULL;
p = &a;                      //将变量 a 的地址赋给 p
q = p;                       // 将 p 的值赋给 q
int * p;
```

画图理解指针

- p = &a;　　//将变量 a 的地址赋给 p　(*P)++　a++



- q = p;

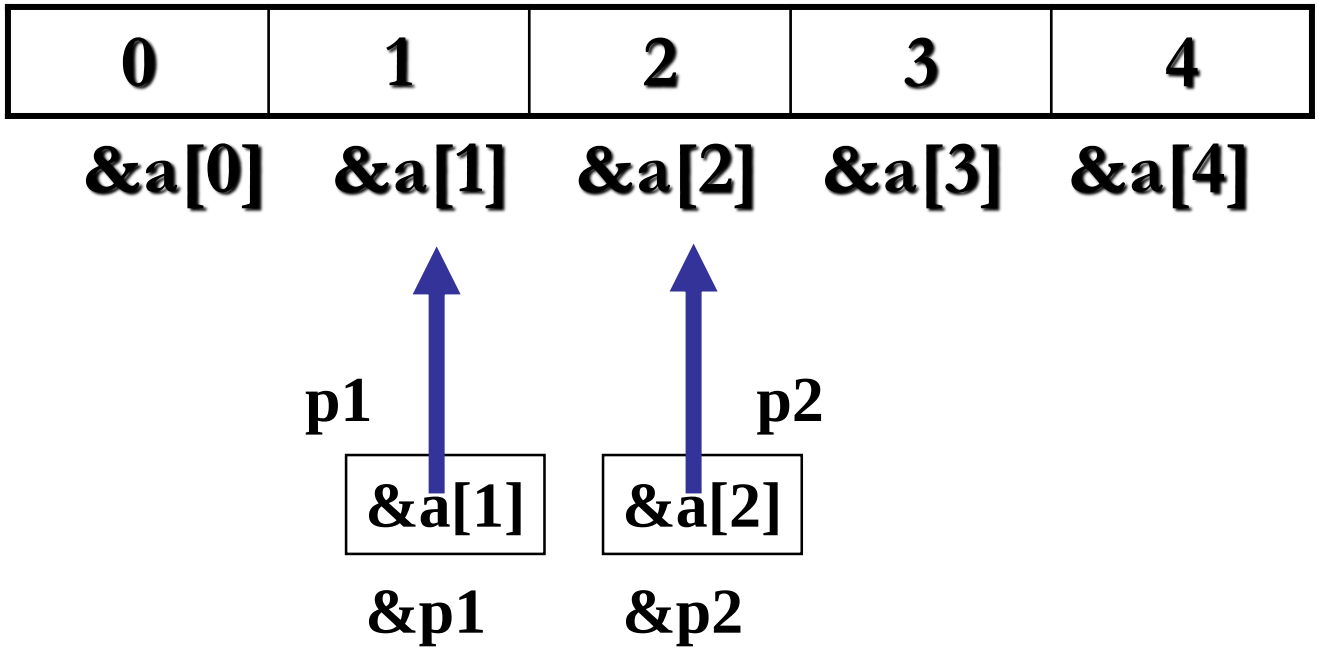


例子

```
#include <stdio.h>
int main() {
   int a[5]={0,1,2,3,4};   //定义数组，赋初值
   int *p1=NULL,*p2=NULL;  //定义指针变量
   p1=&a[1];   //赋值给指针变量，让p1指向a[1]
   p2=&a[2];   //赋值给指针变量，让p2指向a[2]
   printf("%d,%d\n", *p1, *p2);
   printf("%d,%d\n", p1, p2);
   //输出a[1]和a[2]
   return;
}
```

+Chp09_指针赋值.cpp

指针的赋值



用指针实现swap函数

```
#include <stdio.h>
void swap(int * u, int * v);
int main(void) {
  int x = 5, y = 10;
  printf(“Old: x = %d, y=%d.\n", x, y);
  swap(&x, &y);
  printf("Now x = %d, y = %d.\n", x, y);
  return 0;
}

void swap (int * u, int * v) {
  int temp;
  temp = *u;   *u = *v;   *v = temp;
}
```

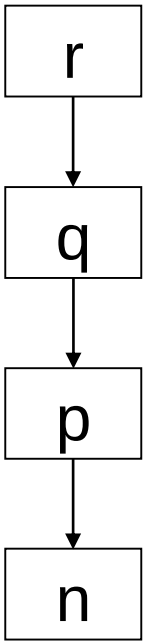
```
void swap (int  u, int  v) {
  int temp;
  temp = u;
  u = v;
  v = temp;
Printf()
}

void swap (int * u, int * v) {
int* temp;
temp = u;
u = v;
v = temp;
}
```

行吗?

+Chp09_sawp交换.cpp

多级指针

- 可以定义指向指针的指针
  - 二级指针，例子：int **q
  - 三级指针，例子：int ***r




```
int  n = 1025;
int *p = &n;
int **q = &p;
int ***r = &q;
printf("*p = %d\n", *p);
printf("*q = %d\n", *q);
printf("**q = %d\n", **q);
printf("**r = %d\n", **r);
printf("***r = %d\n", ***r);
```

中国人民大学
RENMIN UNIVERSITY OF CHINA

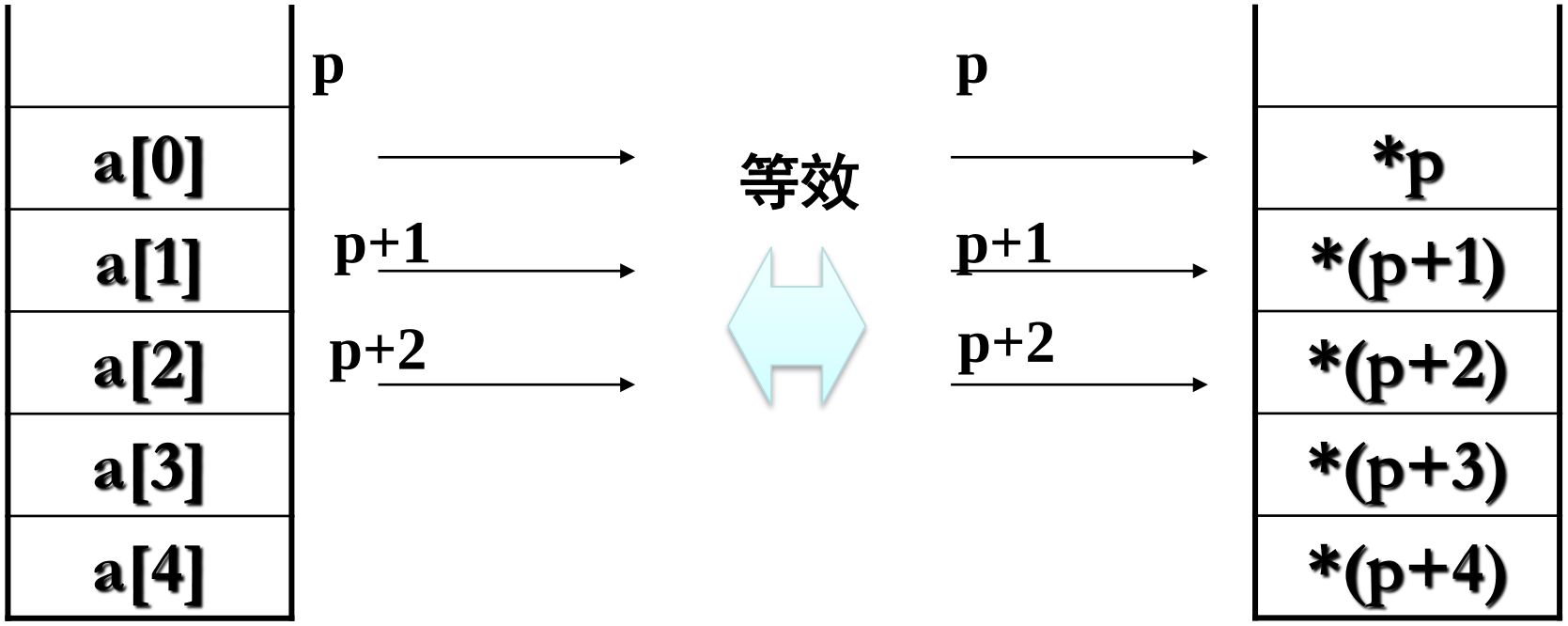
2. 指针与数组

指针 vs. 数组下标

```
#include <stdio.h>
int main() {
    int a[5]={1,3,5,7,9};
    int *p; //定义指针变量
    int i; //定义整型变量
    p=a;//赋值给指针变量，让p指向a数组 p=&a[0]
    for(i=0; i<5; i++) {
        printf("a[%d]=%d\n", i, *p); //输出a数组元素的值
        p++;      //指针变量加1
    }
    return 0;
}
```

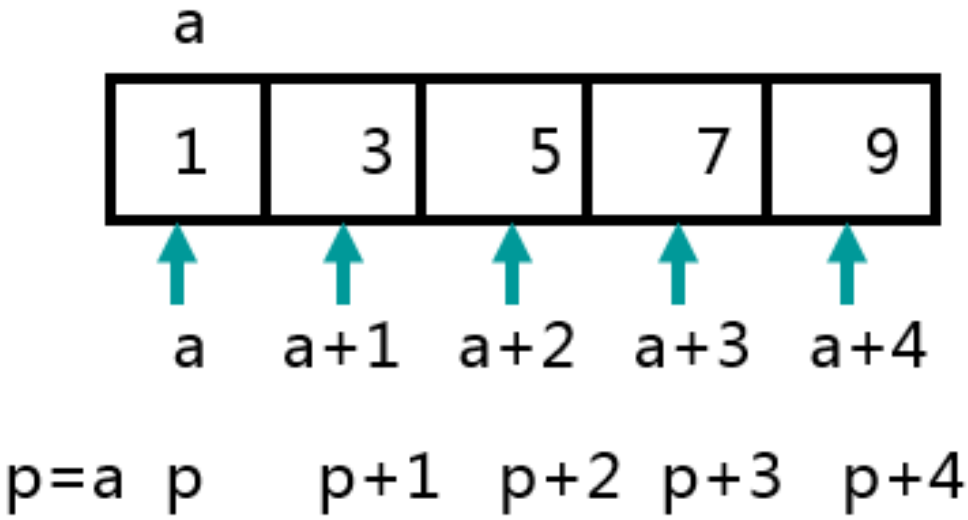
+Chp09_指针数组.cpp

指针与数组 “双轨制”

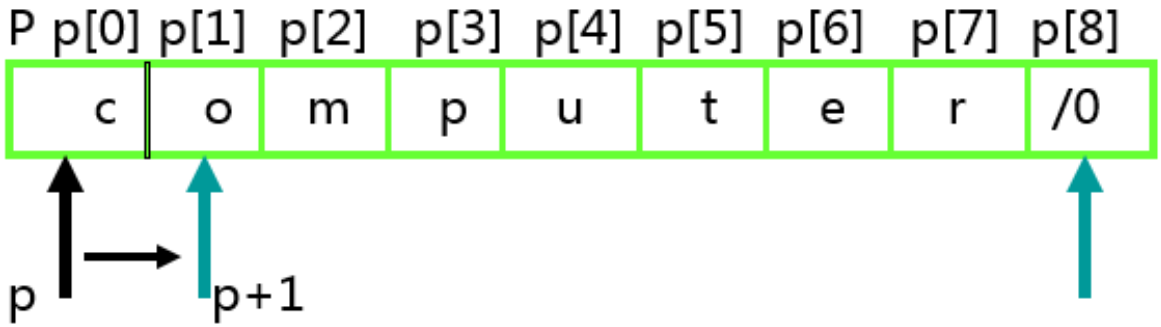


```
#include <stdio.h>
int main()
{int a[5]={1,3,5,7,9};
  int *p, i=0;
  for(p=a; p<a+5;p++)
  {  printf("a[%d]=%d\n", i, *p);
      i++;
  }
  return 0;
}
```

指针风格



```
#include <stdio.h>
int main()
{int i;
char *p;
p="computer";
printf("%s\n", p);
for (i=0; i<8; i++)
      printf("%c", p[i]);
cout<<endl;
while(*p)
{ printf("%c", *p);
   p++; }
cout<<endl;
return 0;
}
```

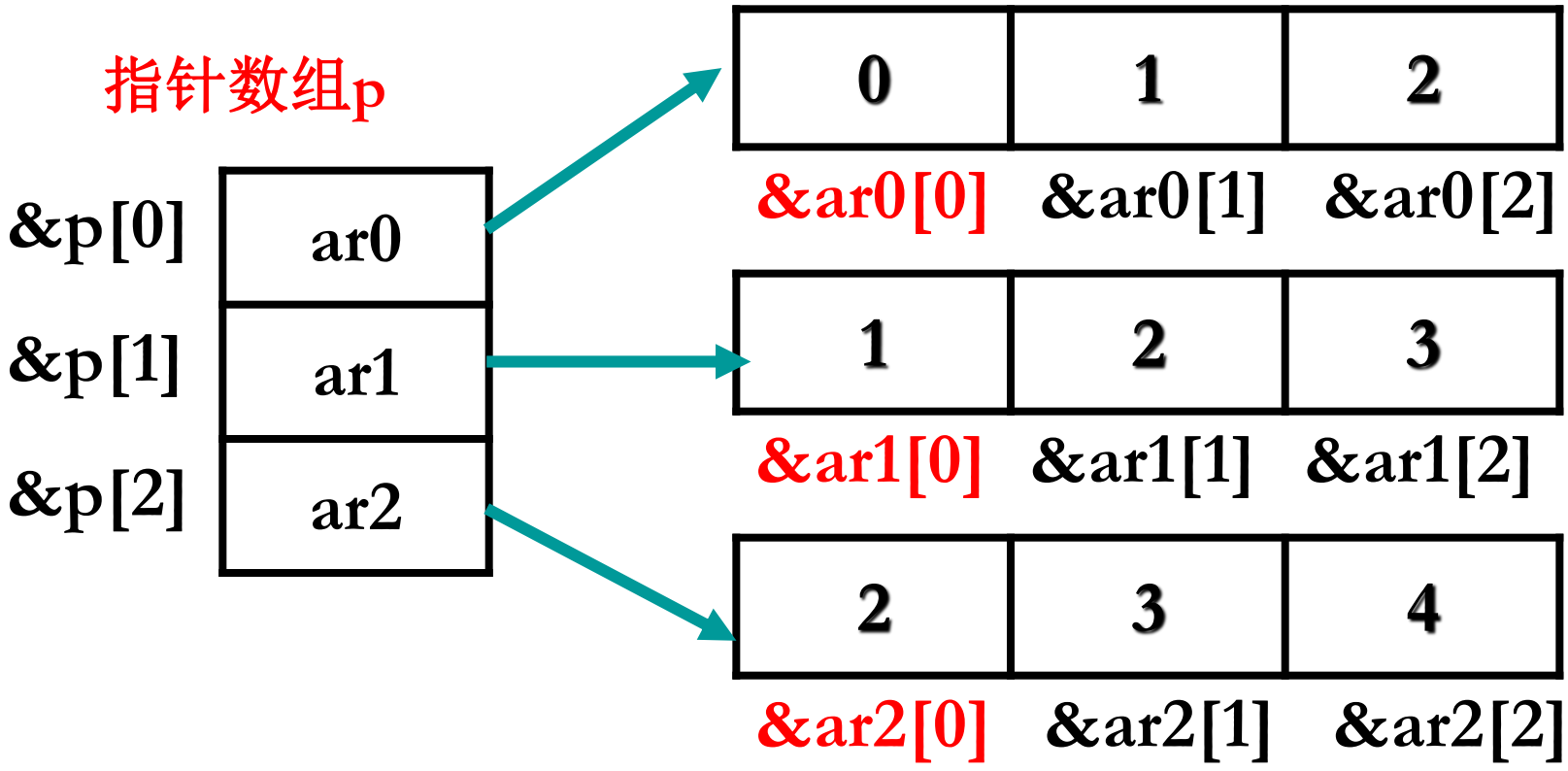


*p==c, *(p+1) ==o, *(p+2) ==m

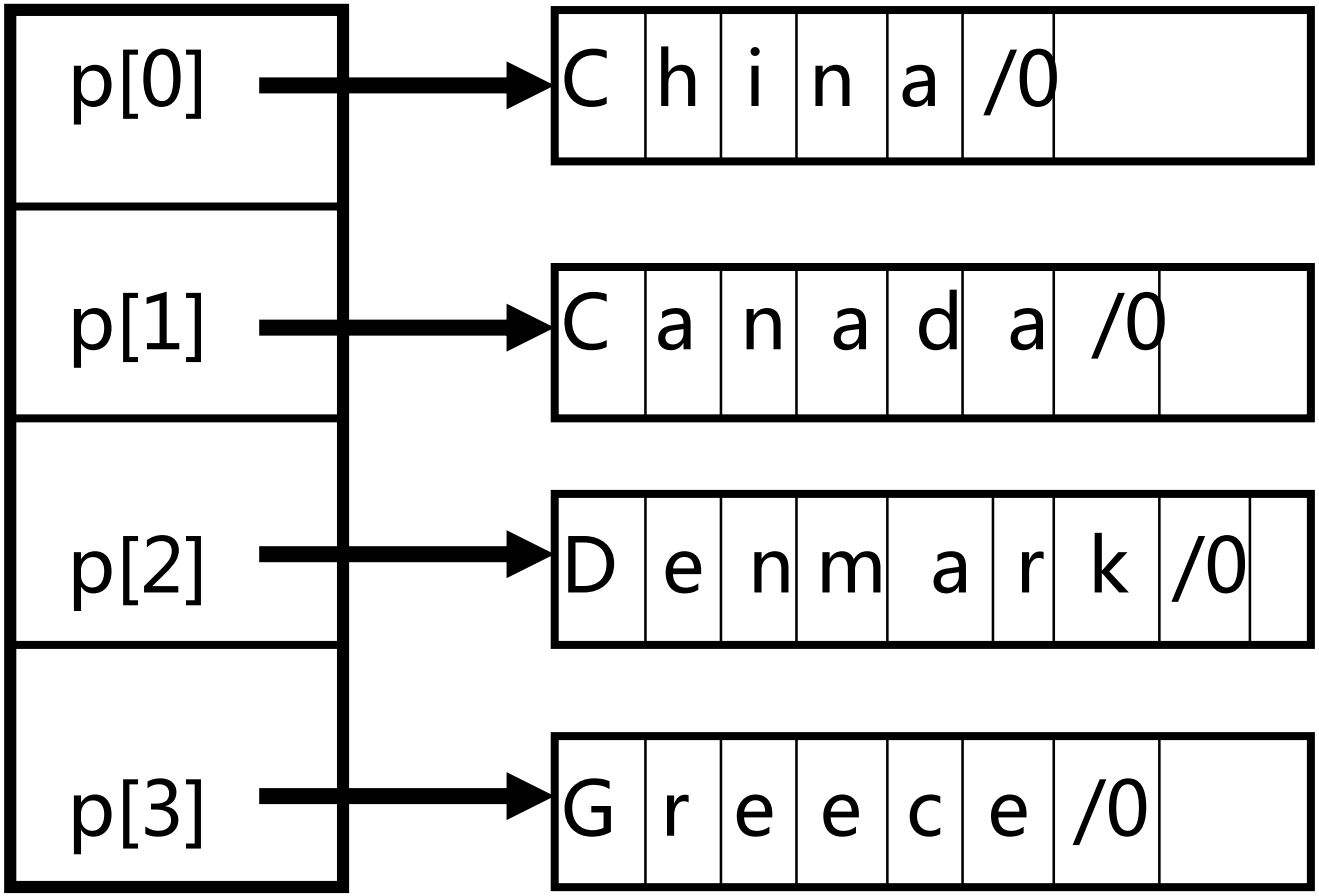
指针数组

- 概念
  - 指针数组是指针的数组
  - 数组单元中存放的是地址，这些地址指向同一种数据类型的变量。
- 指针数组的定义和初始化
  - int **ar0**[ ]={0,1,2};
  - int ar1[ ]={1,2,3};
  - int ar2[ ]={2,3,4};
  - **int* p[3]**= {ar0, ar1, ar2}　定义了数组P，p[0] p[1] p[2]
  - P[0]= ar0　P[0]= &ar0[0]
  - **int (*p)[3]；定义了一个指针变量p，指向一个一维数组**

说明


指针数组p
&p[0]
&p[1]
&p[2]
ar0
ar1
ar2
0
1
2
&ar0[0]
&ar0[1]
&ar0[2]
1
2
3
&ar1[0]
&ar1[1]
&ar1[2]
2
3
4
&ar2[0]
&ar2[1]
&ar2[2]

```
char*p[ ]={ “China",“Canada",
            “Denmark",“Greece“ };
```



中国人民大学
RENMIN UNIVERSITY OF CHINA

3. 指针与字符串

字符指针

- 使用字符指针

char* pets = ” nice cat.";

  - 使用字符数组与字符指针的区别

```
#include <stdio.h>

int main() {
  char p[] = "china";
  p[4] = 'e';
  printf("=%s\n", p);

  char *p1 = "china";
  p1[4] = 'e';
  printf("p1=%s\n", p1);
  return 0;
}
```

字符数组，允许改变！

常量字符串，不允许改变！

字符指针数组

- 字符指针数组

```
char *Names[] = {
   "Tom",
   "Mike",
   "John",
    "Meixi" };
```

  - 以下取值分别是什么？
    - * Names[2], Names[3][0], Names[1]
  - 是否可使用二维数组定义？

```
char Names[10][100];
```

中国人民大学
RENMIN UNIVERSITY OF CHINA

4. 指针与函数

指针作为函数的参数

- 函数的传地址调用（Call by Reference）

```
void Increment (int * p) {
  *p = (*p)+1;
 }

int main () {
  int a = 10;
  Increment (&a);
  printf ("a = %d\n", a);
  return 0;
  }
```

+Chp09_指针参数.cpp

用指针实现swap函数

```
#include <stdio.h>
void interchange(int * u, int * v);
int main(void) {
  int x = 5, y = 10;
  printf( “Originally x = %d and y = %d.\n” , x, y);
  swap(&x, &y);
  printf("Now x = %d and y = %d.\n", x, y);
  return 0;
}

void swap (int * u, int * v) {
  int temp;
  temp = *u;
  *u = *v;
  *v = temp;
}
```

```
int* temp;
temp = u;
u = v;
v = temp;
```

上面行不行?

```
#include <stdio.h>
#include <stdlib.h>
// 传地址（指针），修改指针所间接访问的内容
void swap1(int * x, int * y)
{
    int temp;
    temp = *x;
    *x = *y;
    *y = temp;
    printf("子函数：*x=%d  *y=%d\n", *x, *y);
}
```

交换成功

```
// 传值，修改的子函数空间的变量值
void swap2(int x, int y)
{    int temp;
     temp = x;  x = y;   y = temp;
     printf("子函数：x=%d  y=%d\n", x, y);
}
```

✗ 交换失败

```
// 传地址（指针），修改指针本身
void swap3(int *x, int *y)
{    int *p;
     p = x;  x = y;   y = p;
     printf("子函数：*x=%d  *y=%d\n", *x, *y);
}
```

✗ 交换失败

```
// C++的引用调用方式
void swap4(int &x, int &y)
{
    int temp;
    temp = x;
    x = y;
    y = temp;
    printf("子函数：x=%d  y=%d\n", x, y);
}
swap(a,b);
```

引用变量

不是所有的C都认

了解一下！

交换成功

```
#include <stdio.h>
float * max(float *p, float *q)
{    float *r;
     if (*p > *q) r=p;
     else r=q;
     return r;
}
int main()
{    float x, y, *s;
     scanf("%f",  x);
     printf("x=%f\n", x);
     scanf("%f",  y);
     printf("x=%f\n", y);
     s = max(&x, &y);
     printf("两数中的大数为%f\n", *s);
     return 0;
}
```

函数的返回值

是指针/指针作

为函数的参数

+Chp09_函数返回指针.cpp

例子：查找学生

例 有a个学生，每个学生有b门课程的成绩。要求在用户输入学生序号以后，能输出该学生的全部成绩。用指针函数实现。

解题思路：

- 定义二维数组score存放成绩
- 定义输出某学生全部成绩的函数search，它是返回指针的函数，形参是行指针和整型
- 主函数将score和要找的学号k传递给形参
- 函数的返回值是&score[k][0](k号学生的序号为0的课程地址)
- 在主函数中输出该生的全部成绩

pointer是一个指向一维数组的指针变量

```
#include <stdio.h>
int main()
{ float score[ ][4]={{60,70,80,90}, {56,89,67,88},{34,78,90,66}};
  float  *search(float (*pointer)[4], int n);
  float  *p;  int i, k;
  scanf("%d",&k);
  printf("The scores of No.%d are:\n",k);
  p=search(&score[0], k);
  for(i=0;i<4;i++)
     printf("%5.2f\t",*(p+i));
  printf("\n");
  return 0;
}
```

返回k号学生课程首地址

```
float *search(float (*pointer)[4], int n)
{ float *pt;
   pt=*(pointer+n);
   return(pt);
}
```

+Chp09_查成绩.cpp

找不出及格学生

```
    ……
float  *search(float (*pointer)[4]);
float  *p;  int i,j;
for(i=0;i<3;i++)
{  p=search(score+i);
   if(p==*(score+i))
      { printf("No.%d score:",i);
        for(j=0;j<4;j++)
            printf("%5.2f ",*(p+j));
        printf("\n");
      }
}
   ……
```

```
float *search( float (*pointer)[4] )
{  float *pt=NULL;
   for(int i=0; i<4; i++)
      if(*( *pointer+i )<60)
          pt=*pointer; }
   return(pt);
}
```

再看Qsort函数

- void **qsort**(void* base, size_t n, size_t size, int (*cmp)(const void*, const void*))
  - base 为要排序的数组
  - n为要排序数组的长度
  - size为数组元素的大小（以字节为单位）
  - cmp为判断大小函数的指针
    - 如果a>b，函数返回1；a<b，函数返回-1；a=b，函数返回0

```
#include <stdlib.h>
```

```
int compare (const void * a, const void * b)
{
   if ( *(MyType*)a <  *(MyType*)b ) return -1;
   if ( *(MyType*)a == *(MyType*)b ) return 0;
   if ( *(MyType*)a >  *(MyType*)b ) return 1;
}
```

return *(MyType*)a-*(MyType*)b ;

```
#include <stdio.h>//预编译命令
#include <time.h>
#include <stdlib.h>
int compare(const void *a, const void *b);
int main(){
    int ary[20], i;
    srand((unsigned)time(NULL));       //随机产生20个数
    for (i=0; i<20; i++) ary[i]=rand();
    for (i=0; i<20; i++)  printf("%6d ", ary[i]);
    printf("\n\n");
    qsort(ary, 20, sizeof(ary[0]), compare );          //排序
    for (i=0; i<20; i++)  printf("%6d ", ary[i]);
    printf("\n");
    return 0;}
```

+Chp09_Qsort.cpp

整形数组排序

```
int compare(const void *_a, const void *_b)
{    int * a = (int *)_a;
     int * b = (int *)_b;
     return  *a>*b ? 1 : -1; //return  *a>*b ;
}
```

整型数组排序

如果a>b, 返回1，则是从小到大排序

```
int compare (const void *a, const void *b)
{
    return (*(int*)a - *(int*)b);
}
```

字符中数组排序

```
int compare(const void *a, const void *b)
{    char *p = (char *)a;
     char *q = (char *)b;
     return strcmp(p, q);
}
```

```
char *arr[5] = { "i", "love", "c", "programming", "language" };
qsort(arr, sizeof(arr) / sizeof(arr[0]), sizeof(char *), compare);
```

字符数组排序

```
int compare(const void *a, const void *b) {
    char *p = *(char**)a;
    char *q = *(char**)b;
    return strcmp(p, q);
}
```

中国人民大学
RENMIN UNIVERSITY OF CHINA

5. 指针与结构体

结构数组

- 定义名为student的结构体，包含属性
  - name（字符串）、sex（字符）、birthday（长整型）、height和weight（浮点型）

```
#include "stdio.h"
#include "string.h"
#define N 4

struct student  //名为student的结构类型
{
    char  name[20];                 //姓名
    char sex;                       //性别
    unsigned  long birthday;        //生日
    float height;                   //身高
    float weight;                   //体重
};
```

结构数组排序

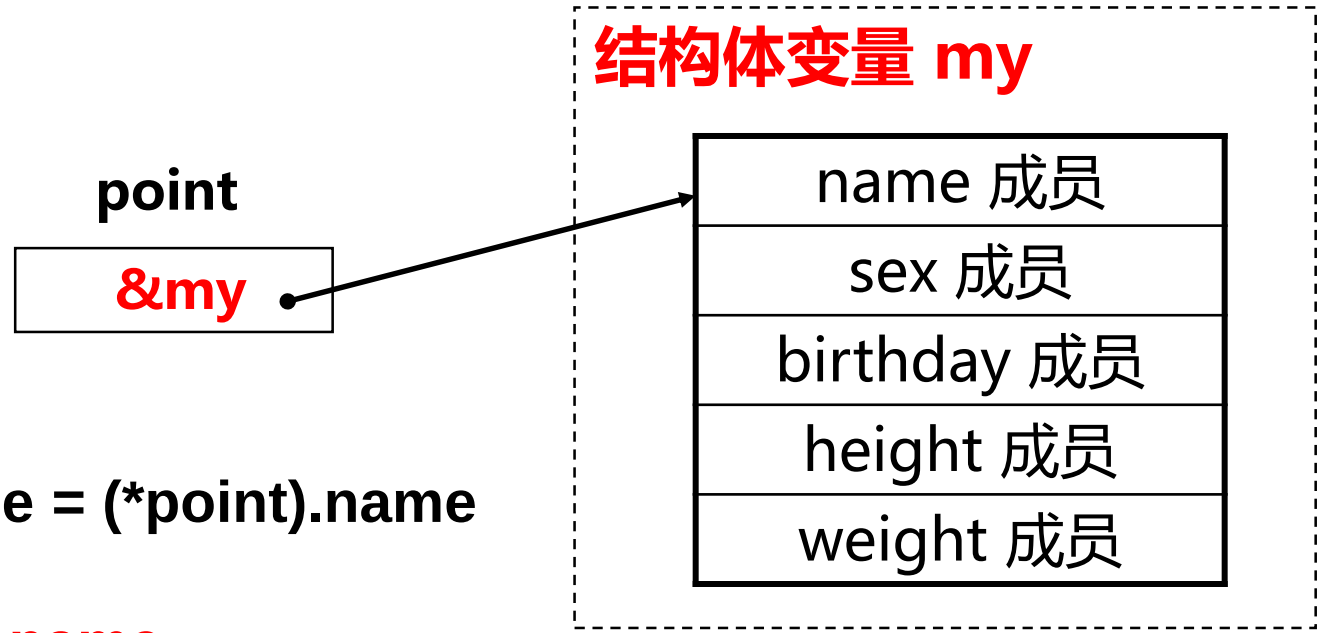
```
struct student STU[N] = {
        {"Li li", 'F', 19840318, 1.88, 65.0 },
        {"Mi mi", 'M', 19830918, 1.75, 58.0 },
        {"He lei", 'M', 19841209, 1.83, 67.1 },
        {"Yu Yu", 'F', 19871109, 1.69, 69.3 }
};
```

+Chp09_结构体数组排序.cpp

```
for ( i = 0; i < N-1; i ++ )
  for ( j = 0; j < N – i-1 ; j++)
    if (STU[j].height > STU[j+1]. height) {
      struct student q = STU[j];
          STU[j] = STU[j+1];
          STU[j+1]= q;   }
```

（结构体）指针存储（结构体）变量

- struct student ***point** ：定义一个指向结构 student 的指针 point
- **point** = &my ：将结构变量my的地址赋给指针 point，指向 my 结构第一个成员的地址。



point->name = (*point).name

my.name

指针访问结构体变量

```
int main() {
        struct student my;
        struct student *point; // 定义student指针
        point = &my; // 将指针point 指向结构my

        strcpy( point->name, "Wang Xiaorong");
        point->sex = 'F';          // 输入学生数据
        point->birthday = 19840923;
        point->height = 1.62f;
        point->weight = 51.5f;

        printf("%s\n", my.name);
        printf("%c\n", my.sex);
        printf("%d\n", my.birthday);
        printf("%f\n", my.height);
        printf("%f\n", my.weight);
        return 0;
}
```

+Chp09_指针访问结构体.cpp

结构体指针数组

```
int main() {
    struct student *p[N];
    for (int i = 0; i < N ; i++)
        p[i] = &STU[i];

    for (int i = 0; i < N - 1; i++)

        for (int j = 0; j < N - i - 1; j++)
            if ( p[j]->height < p[j + 1]->height ) {
                struct student *q;
                q = p[j];  p[j] = p[j + 1];   p[j + 1] = q;     }

    for (int i = 0; i < N; i++)
        printf("%.2f  ", p[i]->height);

    return 0;
}
```

+Chp09_结构体指针排序.cpp

结构体数据排序

```
typedef struct
{
    char name[30];   // 学生姓名
    int Chinese;     // 语文成绩
    int Math;        // 数学成绩
    int English;     // 英语成绩
}st;
```

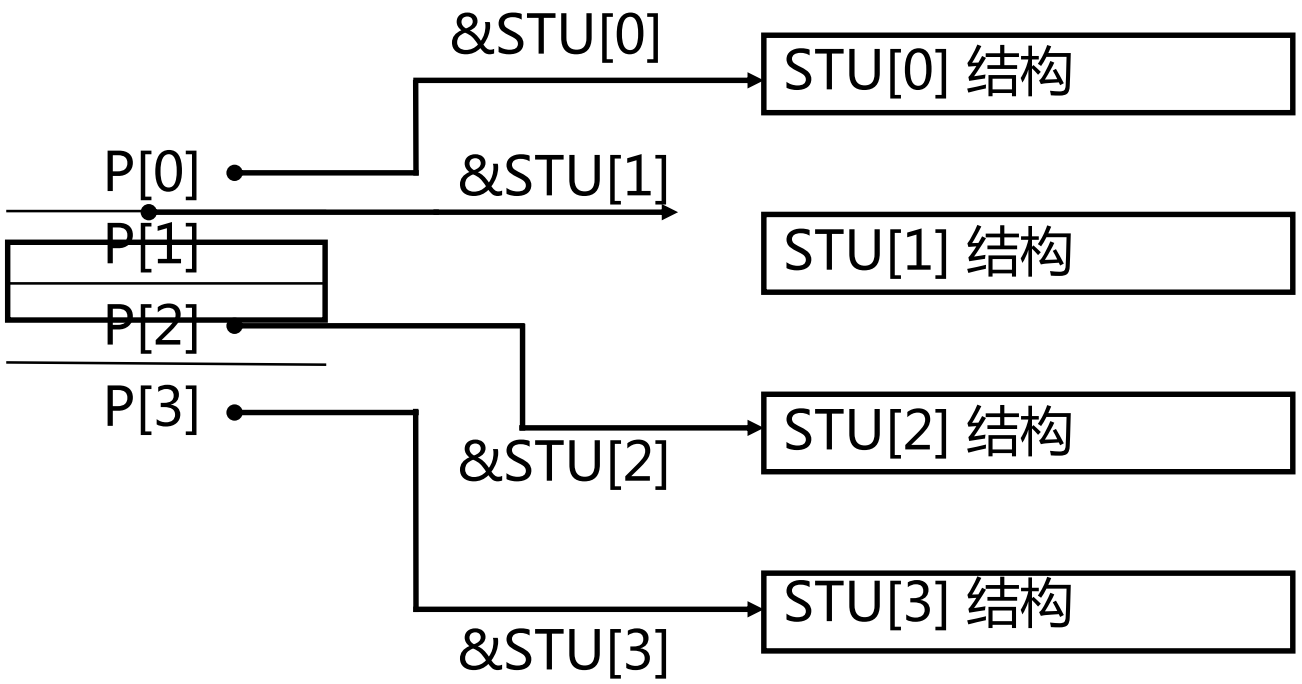
```
st students[7] = {
        {"周",97,68,45},
        {"吴",100,32,88},
        {"郑",78,88,78},
        {"王",87,90,89},
        {"赵",87,77,66},
        {"钱",59,68,98},
        {"孙",62,73,89}
    };
```

qsort(students, 7, sizeof(st), cmp)

```
int cmp(const void* a, const void* b)
{
    st* pa = (st*)a;
    st* pb = (st*)b;
    int num1 = pa->Chinese + pa->English + pa->Math;
    int num2 = pb->Chinese + pb->English + pb->Math;

    return  (int)num1 - num2;      // 从小到大，
    //return (int)num2 - num1;   //  从大到小
}
```

结构体指针数组

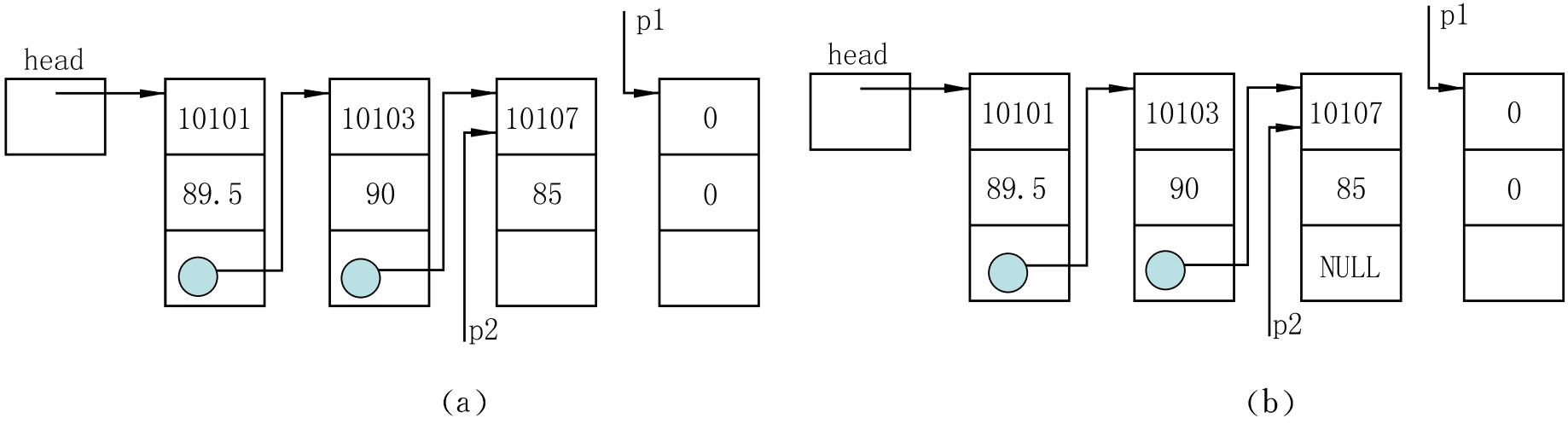
- p[ ]是指向 student 结构的指针数组，经初始化赋入的是 STU 数组中元素所在的地址



动态结构数组

```
int main() {
    struct student *ptr;
    int len;
    cout << "Input array length: ";
    cin >> len;

    ptr = (struct student *) malloc( len*sizeof(struct student));

    for (int i = 0; i < len; i++) {

        // Initialize ptr[i]

    }

    free(ptr);

    return 0;
}
```

链表




```
#include <stdio.h>
#include <malloc.h>
#define NULL 0 //令NULL代表0 ，表示空地址
#define LEN sizeof(struct student)
struct student
  {   long num;
      float score;
      struct student *next;  };
int n;
```

```
struct student *creat()
 {   struct student *head;      struct student *p1,*p2;      n=0;
     p1=p2=( struct student*) malloc(LEN);
     scanf("%ld,%f",&p1->num,&p1->score);
     head=NULL;
     while(p1->num!=0)
        {   n=n+1;
            if(n==1)  head=p1;
            else p2->next=p1;
            p2= p1 ;
            p1=(struct student*)malloc(LEN);
            scanf("%ld,%f",&p1->num,&p1->score);    }
     p2->next=NULL;   return(head);
 }
```

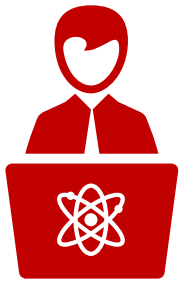
```
void print(struct student *head)
 { struct student *p;
   printf("\nNow,These %d records are:\n",n);
   p=head;
   if(head!=NULL)
      do {  printf("%ld %5.1f\n", p->num, p->score);
            p=p->next;  }
      while(p!=NULL);
}

int main()
{  struct student *p;
   p=creat();    print(p);
  return 0;  }
```

+Chp09_链表.cpp

数组与链表对比

- **一、容量**
  - 1.数组的元素是固定的、连续的，在定义时分配存储单元有可能浪费空间或者下标越界需要重新定义数组，空间效率差；
  - 2.链表的结点个数可按需要增减，可存在任何地方、不要求连续，在程序执行时动态向程序申请存储单元。
- **二、位置**
  - 1.数组中的元素顺序由元素在数组中的位置下标确定；
  - 2.链表的结点顺序保存在节点中
- **三、增删查改速度**
  - 1.数组查询快；但数组插入删除效率低；
  - 2.链表插入删除快；但查询效率低。

中国人民大学
RENMIN UNIVERSITY OF CHINA



谢谢大家！