C语言

第10讲：指针的复杂形式

赵岩

哈尔滨工业大学软件学院

August 20, 2011

目录

目录

目录

地址与指针复习

- 指针p指向（保存）一个地址
- p+1的含义？
  - 指针p有数据类型的信息。
  - p+1按照数据类型的长度来移动指针。
- *(p)按数据类型信息从当前地址取出内容

指针与一维数组

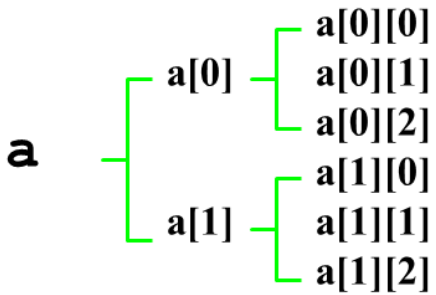
```
1  int a[3]; /* a 类
2  a+1; /*数组第一个
3  *(a+1) /*数组第一
4  int *p = a; /*i
```

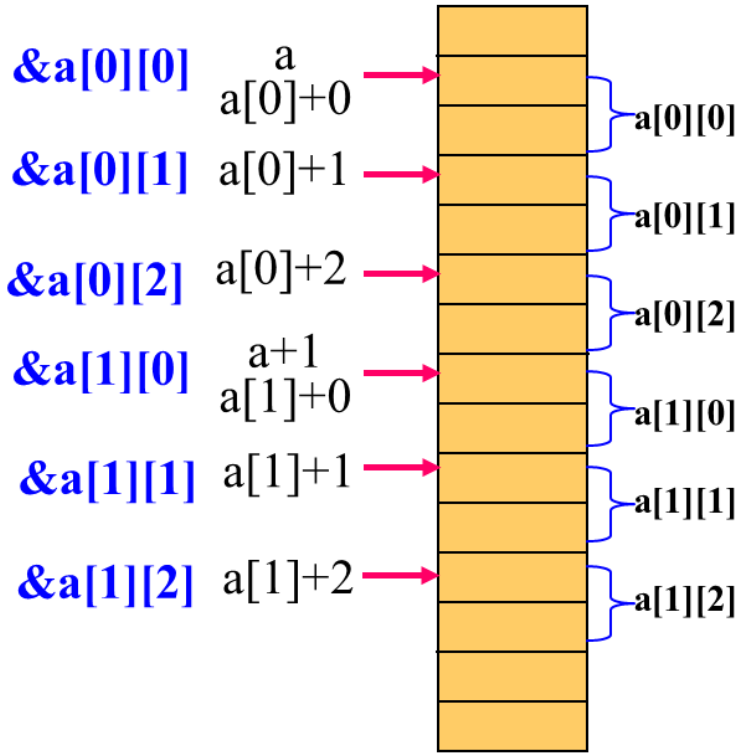
```
1  foo a[3]; /* a 类型为foo的一个地
2  a+1; /*数组第一个元素的地址*/
3  *(a+1) /*数组第一个元素的内容*/
4  foo *p = a; /*foo类型的地址用
```

二维数组的内存布局

- C语言将二维数组看作一维数组，其每个数组元素又是一个一维数组
- 按行**顺序**存放所有元素

```
int a[2][3];
```





指针与二维数组

- int a[3], b[2][3];
- b 类型为（int[3]）的一个地址
  - int[3]是**3个元素整型数组数据类型**
  - 类似于上面的foo，等价于int a[3];
- b+1 数组第一个元素的地址
- *(b+1) 数组第一个元素的内容
- *(b+1)+1 等价于a+1
- *(*(b+1)+1) 等价于*(a+1)
- int (*p)[3] = b 类型为（in[3]）的一个指针指向了一个类型为（int[3]）的一个地址。

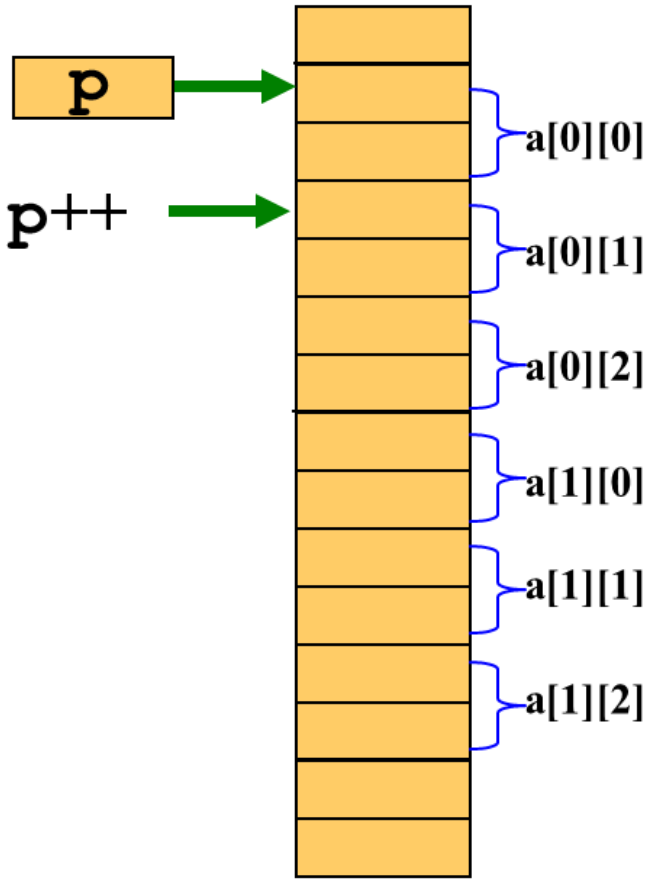
不同的表示形式

- int b[2][3]; 则b
  - 代表二维数组的首地址，第0行的地址
  - int[3]数据类型地址
- b+i
  - 代表第i行的地址
  - int[3]数据类型地址
- *(b+i) 即b[i]
  - 代表第i行第0列的地址
  - int[3]数据类型的值，等价于int a[3]定义中的a;
- *(b+i)+j 即b[i]+j
  - 代表第i行第j列的地址
  - int类型的地址，等价于a+j;
- *(*(b+i)+j) 即b[i][j]
  - 代表第i行第j列的元素
  - int类型的值，等价于b[i][j]

二维数组的列指针表示

- 逐个元素查找元素所在位置
- 相对于数组起始地址的偏移量
  - i*n+j

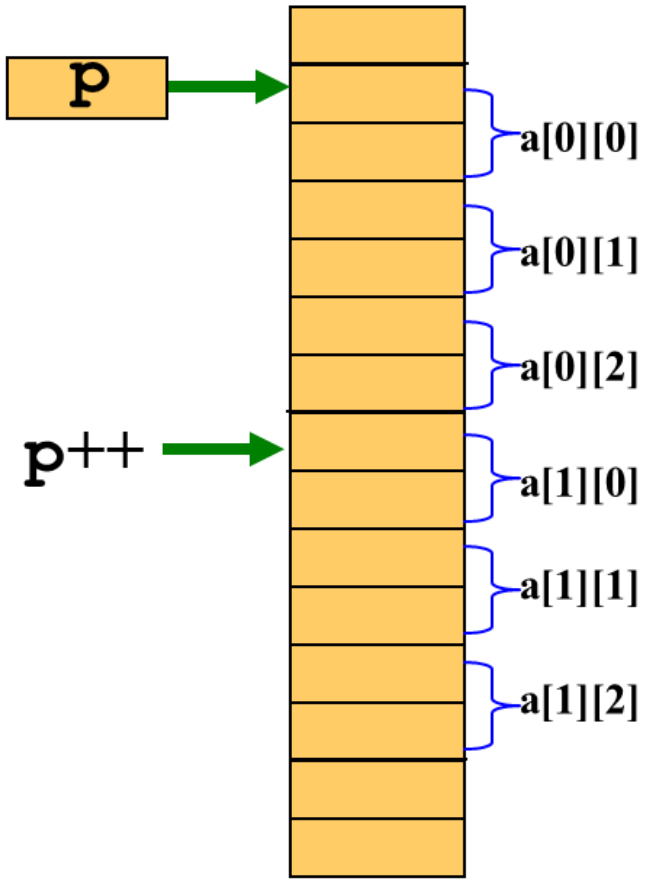
```
1  int a[2][3];
2  int *p;
3  p = *a;
4  for (i=0; i<m; i++)
5    for (j=0; j<n; j++)
6      printf("%d",*(p+i*n+j));
```



二维数组的行指针

- int (*p)[3];
- p = a;//用行地址初始化
- 先逐行查找元素所在行
- 再在行内逐列查找元素所在位置

```
1  for (i=0; i<m; i++)
2    for (j=0; j<n; j++)
3      printf("%d",*(*(p+i)+j));
```



二维数组作为函数实参

```
1  int a[2][3];
2  GetMax(int *p, int m,int n,int *iRow,int *iCol)
3  {
4    *(p+1*n+2) /*第1行，第2列*/
5  }
6  GetMax(*a,2,3,&row,&col); /*调用*/
7
8  GetMax(int array[][3],int m,int n
9              ,int *iRow,int *iCol);{
10   *(*(array+1)+2) /*第1行，第2列*/
11   array[1][2] /*第1行，第2列*/
12 }
13 GetMax(a,2,3,&row,&col); /*调用*/
```

指针数组定义

- 元素均为指针类型的数组，称为指针数组
- 定义形式为：

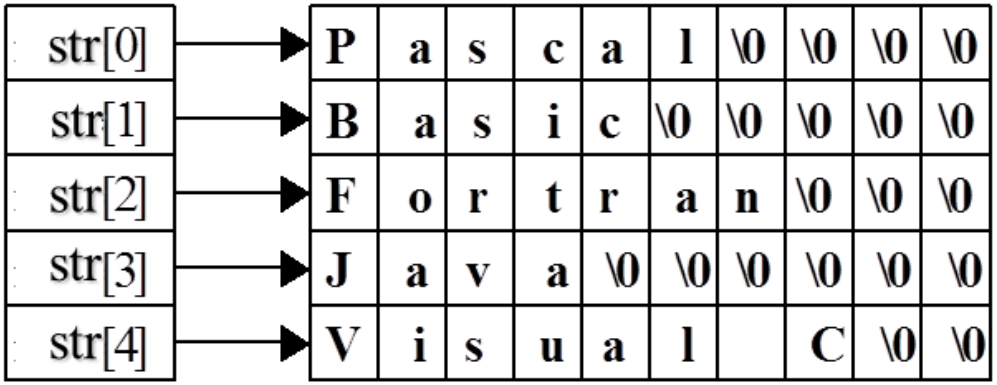
  类型关键字　*数组名　[数组长度];

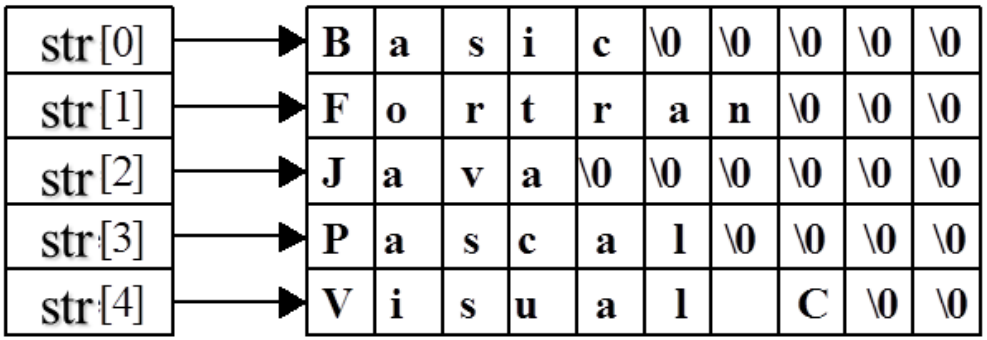
```
1    char *pStr[5];
2    int *pInt[3];
3    float *pFloat[];
```

字符串排序-二维数组实现

```
1  char temp[10];
2  char str[][10] = {"Pascal", "Basic",
3                    "Fortran", "Java", "Visual C"};
4  if(strcmp(str[j],str[i])<0)
5  {
6      strcpy(temp ,str[i]);
7      strcpy(str[i],str[j]);
8      strcpy(str[j] ,str[i]);
9  }
```

二维数组实现图示

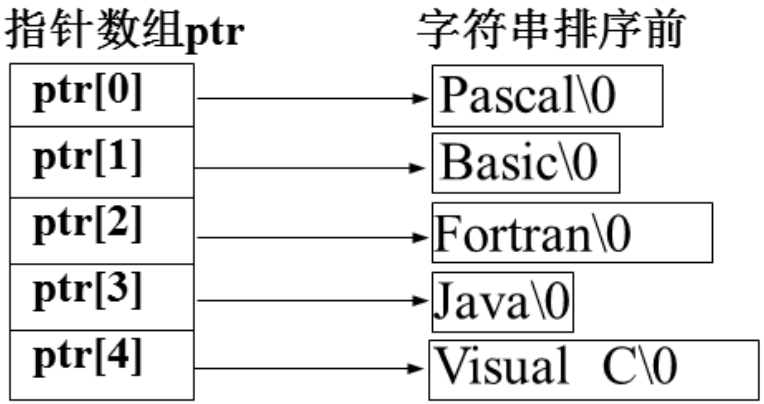

str[0]
str[1]
str[2]
str[3]
str[4]
P a s c a l \0 \0 \0 \0
B a s i c \0 \0 \0 \0 \0
F o r t r a n \0 \0 \0
J a v a \0 \0 \0 \0 \0 \0
V i s u a l C \0 \0

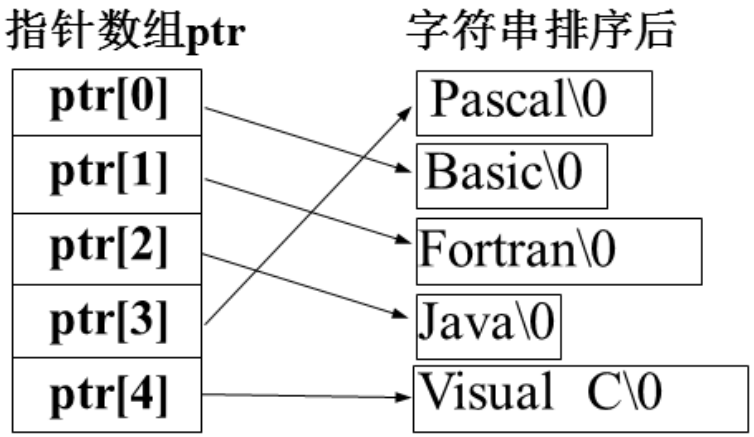

str[0]
str[1]
str[2]
str[3]
str[4]
B a s i c \0 \0 \0 \0 \0
F o r t r a n \0 \0 \0
J a v a \0 \0 \0 \0 \0 \0
P a s c a l \0 \0 \0 \0
V i s u a l C \0 \0

字符串排序-指针数组实现

```
1  char *temp = NULL;
2  char *ptr[N] = {"Pascal", "Basic",
3                  "Fortran", "Java", "Visual C"};
4  if(strcmp(ptr[j],ptr[i])<0)
5  {
6      temp = ptr[i];
7      ptr[i] = ptr[j];
8      ptr[j] = ptr[i];
9  }
```

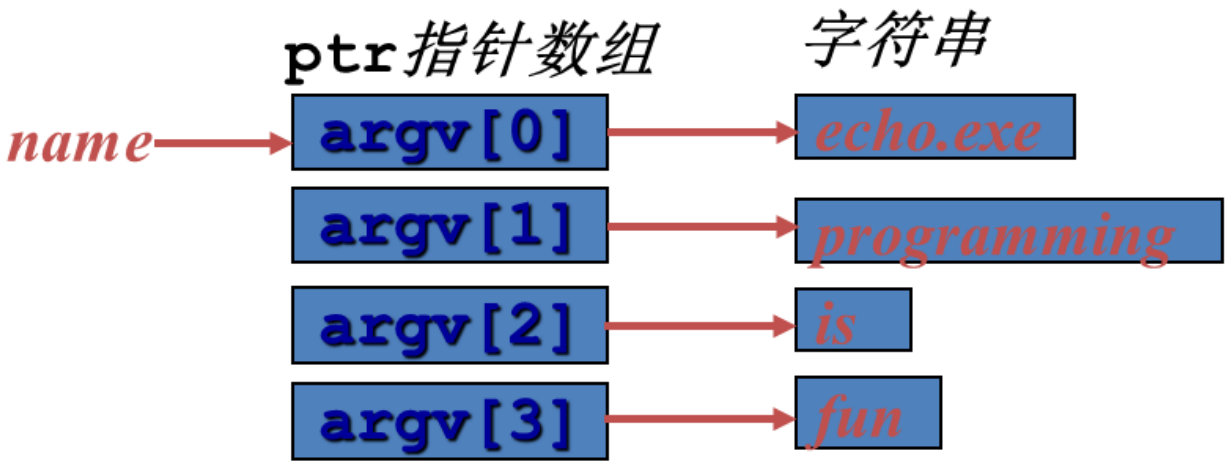
指针数组实现图示


指针数组ptr
字符串排序前
ptr[0]
ptr[1]
ptr[2]
ptr[3]
ptr[4]
Pascal\0
Basic\0
Fortran\0
Java\0
Visual C\0


指针数组ptr
字符串排序后
ptr[0]
ptr[1]
ptr[2]
ptr[3]
ptr[4]
Pascal\0
Basic\0
Fortran\0
Java\0
Visual C\0

命令行参数

- 通过命令行参数，使用户可以根据需要来决定我们的程序干什么、怎么干
- main(int argc, char* argv[])
  - 当你把main函数写成这样时
  - argc的值为程序执行时参数的数目（包括命令本身）
  - argv[i]为指向第i个参数的字符指针
  - 这两个内设形参用于接收命令行参数



命令行参数

```
1  main(int argc, char *argv[])
2  {
3      int i;
4      printf("The program name
5                  is:%s\n", argv[0]);
6      if (argc > 1){
7          printf("The other arguments
8                      are following:\n");
9          for (i = 1; i<argc; i++){
10             printf("%s\n", argv[i]);
11         }
12     }
13 }
```

内存分配函数-1

- #include <stdlib.h>
- #include <alloc.h>
- void*类型的指针可以转换成任意类型的变量
- void* malloc(unsigned int size);
  - 向系统申请大小为size的内存块，把首地址返回。
  - 申请不成功，返回NULL
  - 不初始化

内存分配函数-2

- #include <stdlib.h>
- #include <alloc.h>
- void* calloc(unsigned int num, unsigned int size);
  - 向系统申请num个size大小的内存块，把首地址返回。
  - 如果申请不成功，返回NULL
  - 初始化成0
- void free(void* p);
  - 释放由malloc()和calloc()申请的内存块。

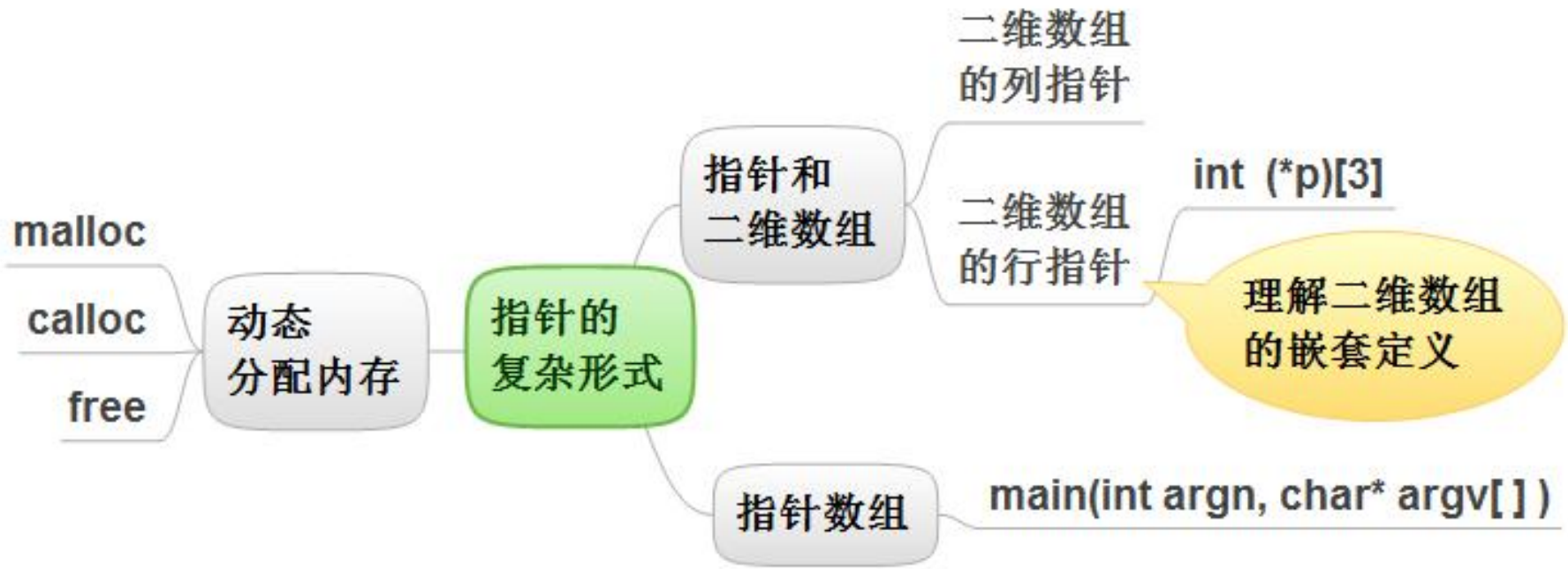
动态一维数组

```
1    int   *p = NULL;
2       printf("Please enter array size:");
3       scanf("%d", &n);
4       p = (int *) malloc(n * sizeof (int));
5       p[i]      /*像一维数组一样使用*/
6       .....
7       free(p);
```

动态二维数组

```
1       printf("Please enter array size m,n:");
2       scanf("%d,%d", &m, &n);
3       p = (int *) calloc(m * n, sizeof (int));
4       p[i*n+j]); /*二维数组列指针使用*/
5       .....
6       free(p);
```

总结


malloc
calloc
free
动态
分配内存
指针的
复杂形式
指针和
二维数组
二维数组
的列指针
二维数组
的行指针
int (*p)[3]
理解二维数组
的嵌套定义
指针数组
main(int argn, char* argv[ ])

谢谢大家，欢迎提问！