
年轻人
不要
老熬夜
早睡早起

School of Economics ang Management, Beihang University

Introduction to Computer Programming

Jichang Zhao

jichang@buaa.edu.cn

Pointers And Arrays

Objectives

- Array Names as Pointers
- Processing Strings Using Pointers
- Creating Strings Using Pointers
- 注意：已经初步具备了解决实际问题的能力
  - 有意识地用C解决其他课程中的计算或仿真问题

Array names as pointers

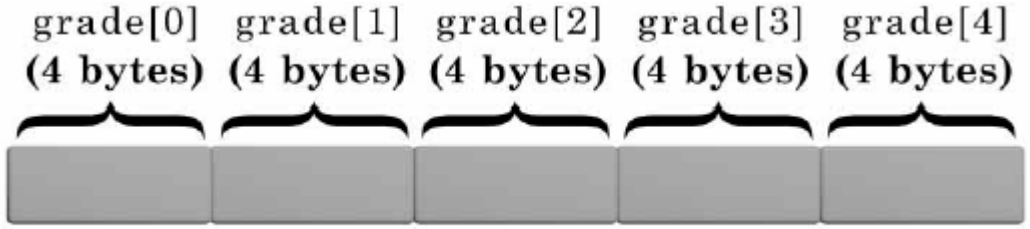


Figure 11.1 The grade array in storage

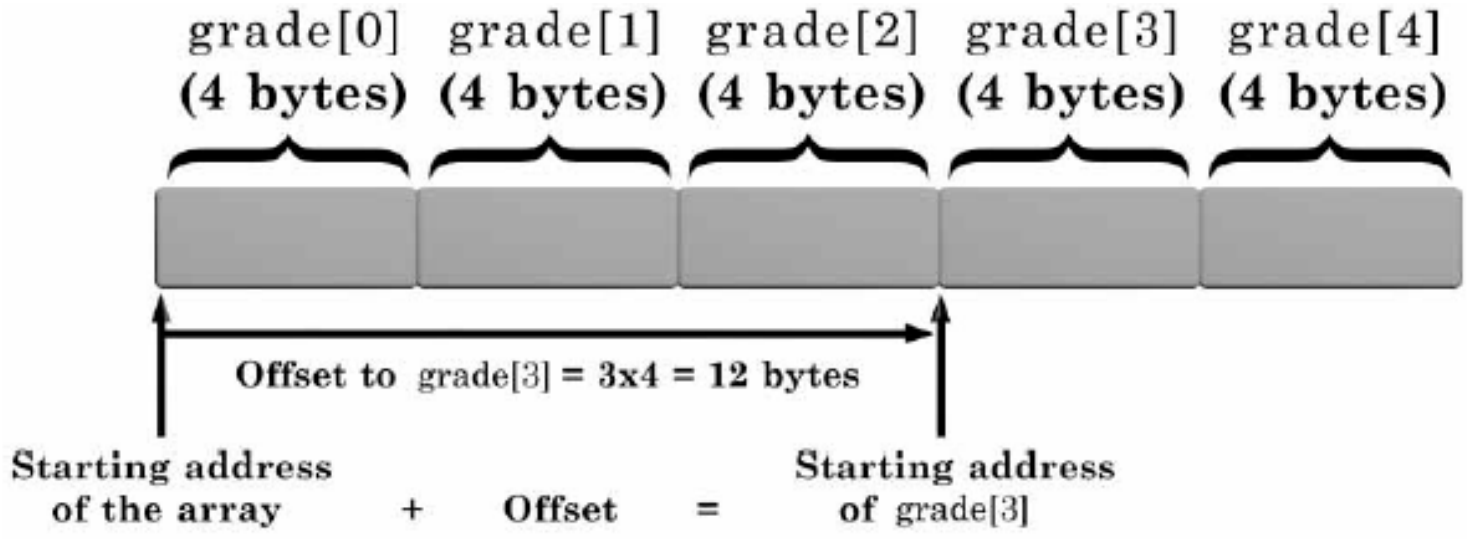


Figure 11.2 Using a subscript to obtain an address

Array Names as Pointers

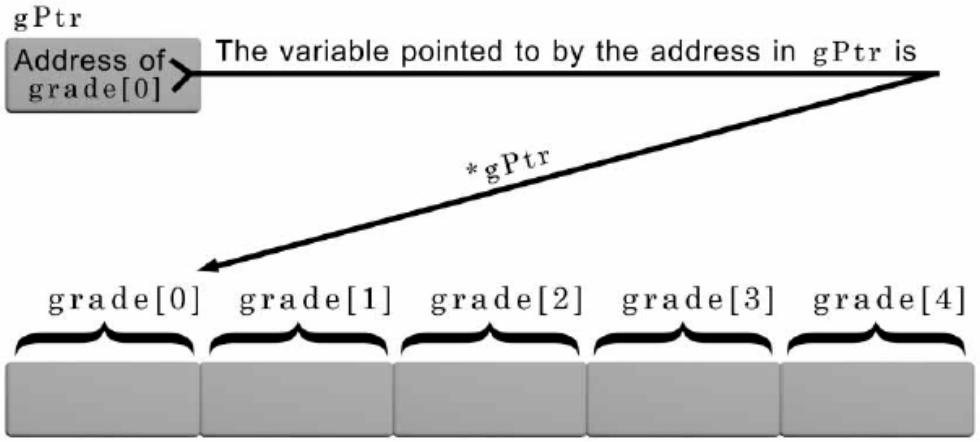


Figure 11.3 The variable pointed to by `*gPtr` is `grade[0]`

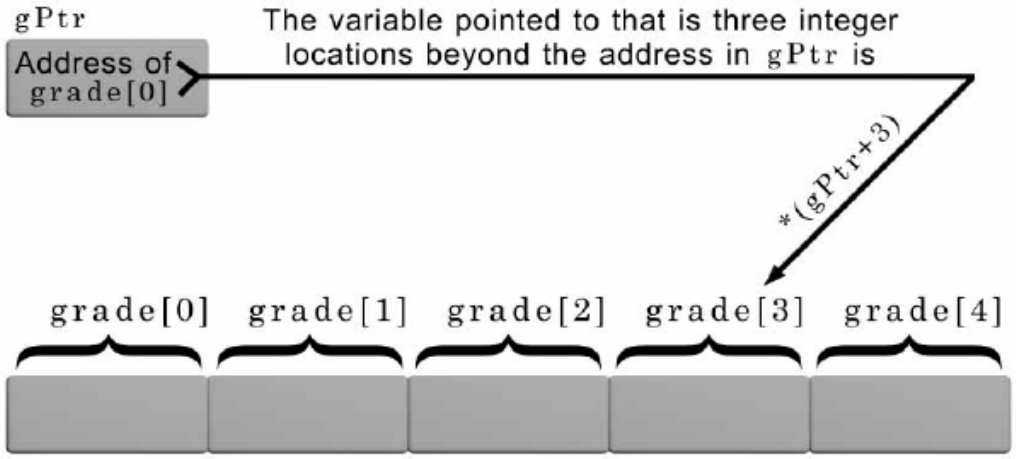
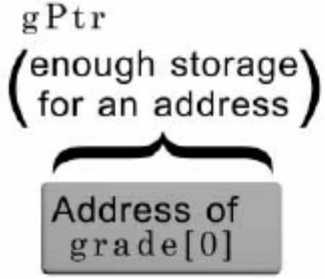


Figure 11.4 An offset of three from the address in `gPtr`

Array Names as Pointers

Table 11.1 Array Elements May be Accessed in Two Ways

Array Element	Subscript Notation	Pointer Notation
Element 0	`grade[0]`	`*gPtr`
Element 1	`grade[1]`	`*(gPtr + 1)`
Element 2	`grade[2]`	`*(gPtr + 2)`
Element 3	`grade[3]`	`*(gPtr + 3)`
Element 4	`grade[4]`	`*(gPtr + 4)`



grade[i] 和 i[grade]是不是一样？

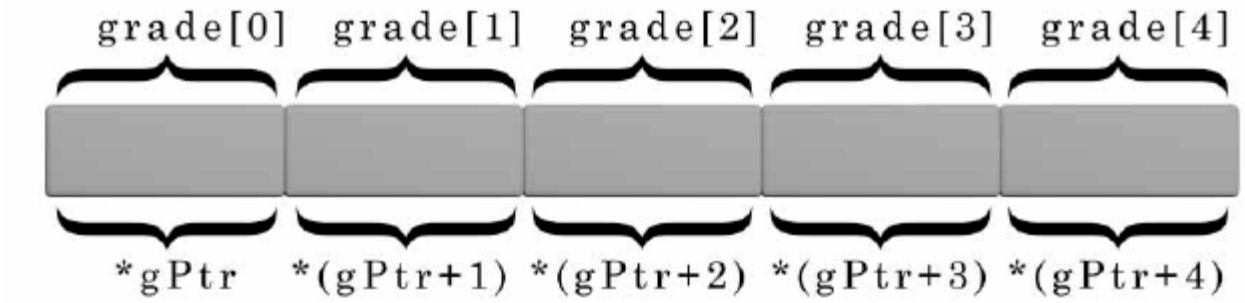


Figure 11.5 The relationship between array elements and pointers

Array Names as Pointers (continued)

- In most respects an array name and a pointer can be used interchangeably（尤其对函数形参）
- An array name is a pointer constant
  - `grade = &grade[2];` **is invalid**
- ***A pointer access can always be replaced using subscript notation***
  - `numPtr[i]` is valid even if `numPtr` is a pointer variable
- **Pointers are more efficient than using subscripts for array processing because the internal conversion from subscripts to addresses is avoided**
- **DEMO(psub.c)**

Processing Strings Using Pointers

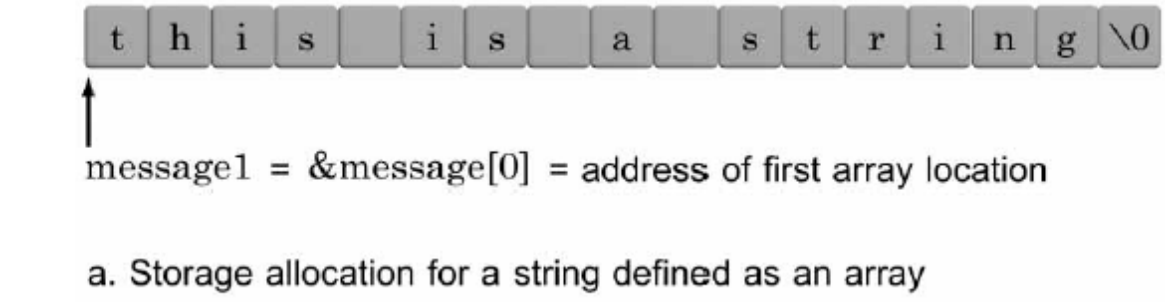
```
void strcopy(char string1[], char string2[])
{
  int i = 0;
  while (string1[i] = string2[i])
    i++;
}

void strcopy(char *string1, char *string2)
{
  while (*string1++ = *string2++)
  ;
  }
  return;
}
```

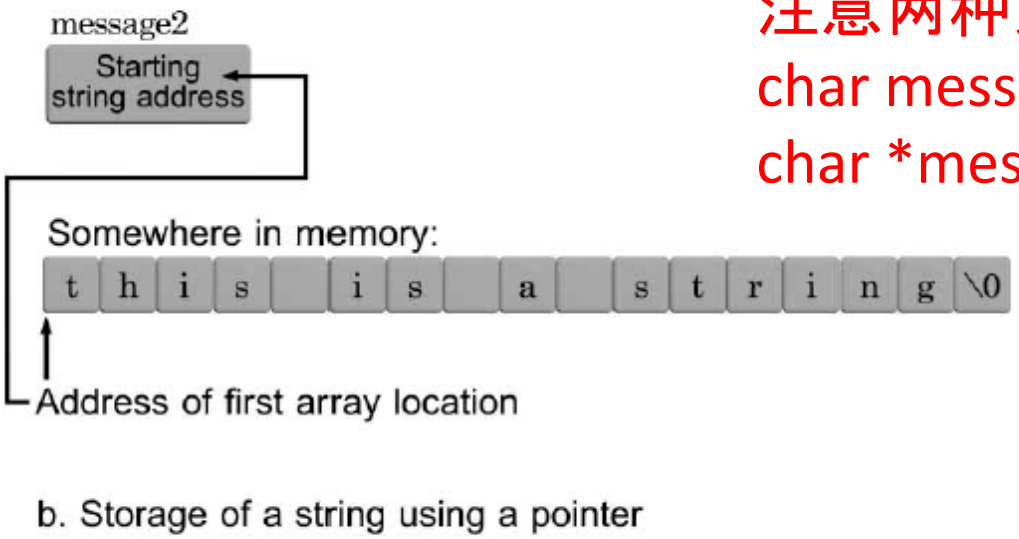
Creating Strings Using Pointers

- The definition of a string automatically involves a pointer (***a pointer constant***)
  - `char message1[81];` reserves storage for 81 characters and automatically creates a pointer constant, `message1`, that contains the address of `message1[0]`
- It is also possible to create a string using a pointer
  - `char *message2;`
  - Now, assignment statements, such as `message2 = "this is a string";`, can be made
- ***Strings cannot be copied using an assignment operator 【浅copy和深copy的区别】***

Creating Strings Using Pointers



a. Storage allocation for a string defined as an array



b. Storage of a string using a pointer

Figure 11.12 String storage allocation

注意两种方式的差异

char message1 [81]=“”;

char *message2=“”;

Allocating Space for a String

- The following declaration is valid:
  - `char *message = "abcdef";`
- But, this is not:
  - `char *message; /* declaration for a pointer */`
  - `strcpy(message,"abcdef"); /* INVALID copy */`
  - ***The `strcpy` is invalid here because the declaration of the pointer only reserves sufficient space for one value—an address***

补充：对指针的操作

- 在一个不指向任何数组元素的指针上执行算术运算会导致未定义的行为
  - `int a[N],*p;`
  - `for(p=&a[0];p<&a[N];p++)`
  - `for(p=a;p<a+N;p++)`
- 只有两个指针指向同一个数组时，把它们相减才有意义
- 指针可以指向复合常量
  - `int *p=(int[]){3,4,5,6,7};`
- 对于形参，声明为指针或数组是一样的；但对于变量而言，声明为数组和指针在空间分配上有差异
  - `int a[10], *p;`
  - `*p=0;//WRONG`

补充：指针与二维数组

- 遍历二维数组
  - `int a[M][N],*p;`
  - `for(p=&a[0][0];p<=&a[M-1][N-1];p++)`
  - **&a[0]（或a[0]）跟&a[1]（或a[1]）差多少？**
- 处理二维数组的一行
  - `int a[M][N],*p;`
  - `i=0;`
  - `for(p=a[i];p<a[i]+N;p++)//&a[i]`也可以
- 处理二数组的一列
  - `int a[M][N], (*p)[N];`
  - `int i;`
  - `for(i=0;i<N;i++)`
    - `for(p=&a[0];p<&a[M];p++)`
      - `(*p)[i]=0;`
  - **`int *p[N];`指针数组**
  - **`int (*p)[N];`指针，指向一个长度为N的整形数组 (arrayp.c)**

Pointer Arrays

- Example:

```
char *seasons[4];
seasons[0] = "Winter";
seasons[1] = "Spring";
seasons[2] = "Summer";
seasons[3] = "Fall";
```

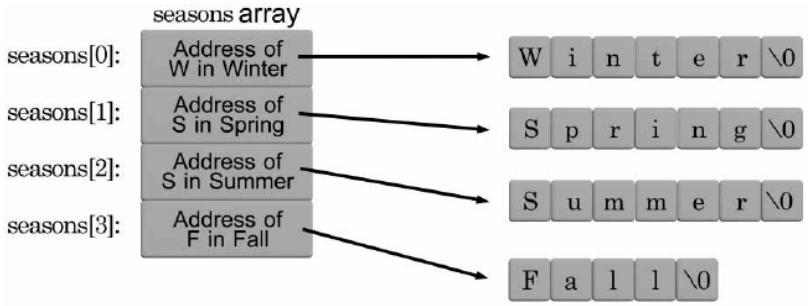


Figure 11.14 The addresses contained in the `seasons[]` pointers

- Or:

```
char *seasons[4] = {"Winter",
                    "Spring",
                    "Summer",
                    "Fall"};
```

这时能够修改字符串数组中的内容吗？(parray.c)

补充：动态内存管理

- 堆上的内存管理
  - 动态进行分配（程序运行时进行分配）
  - 程序员自主管理（申请或释放）
  - stdlib.h
- 关键函数
  - calloc: Allocate and zero-initialize array
    - `void* calloc (size_t num, size_t size);`
  - **malloc: Allocate memory block**
    - `void* malloc (size_t size);`
  - realloc: Reallocate memory block
    - `void* realloc (void* ptr, size_t size);`
  - **free: Deallocate memory block**
    - `void free (void* ptr);`
  - **DEMO(mem.c, rpwd.c)**

model of segregation

- Background
  - Thomas Schelling
- Model
  - void randomly_set_city(void);
  - void print_city(void);
  - double same_neighbors_ratio(int,int);
  - void move(int,int);
  - void evolve(double);
- 思考
  - 稳定后的形态如何度量？
  - 如何在实际中验证这种稳态的意义？

本周作业

- 在提供的smodel.c基础上进行各种修改
  - 城市的规模
  - 初始化条件（富人的比例，穷人的比例，空白的比例）
  - 邻居的定义
  - 迁移的阈值
  - 呈现的方式
  - 跟北京房价的空间分布能不能相互印证？
- 提交方式
  - 动态演化过程发朋友圈（我们是有代码证的人）
    - 真的程序员，敢于在朋友圈晒代码结果
    - 只会晒自拍的都是文科生！！！
  - 我会通过间接寻址的方式来检查提交的结果
    - “六度分隔”了解一下