第七章

C指针

7.4通过指针引用字符串

7.4.1 字符串的引用方式

(1) 用字符数组存放一个字符串，通过数组名和下标引用字符串中的一个字符，也可以用数组名和格式声明“%s”输出整个字符串。

例 7.15 定义一个字符数组，对它初始化， 然后输出该字符串

```
#include <stdio.h>
int  main()
{
    char string[]="I love China!";
    printf("%s\n",string);
    printf("%c\n", string[7]);
    return 0;
}
```

```
I love China!
C
```

(2) 用字符指针指向一个字符串常量，通过字符指针变量引用字符串常量

定义一个字符指针，用字符指针指向字符串中的字符。

例7.16 定义字符指针

```
#include <stdio.h>
int  main()
{char * string="I love China!";
 printf("%s\n",string);
 return 0;
}
```

定义字符指针变量string并初始化

I love China!

对字符串中字符的存取，可以用下标法，也可以用指针法

例7．17 将字符串 a 复制为字符串 b，并输出字符串b

对字符串中字符的存取，可以用下标法，也可以用指针法

```
#include <stdio.h>
int main()
{   char  a[]="I am a boy.",b[20];
    int i;
    for(i=0;*(a+i)!='\0';i++)
        *(b+i)=*(a+i);
    *(b+i)='\0';            //在b数组的有效字符之后加'\0'
    printf("string a is :%s\n",a);
    printf("string b is:");
    for(i=0;b[i]!='\0';i++)
        printf("%c",b[i]);
    printf("\n");
}
```

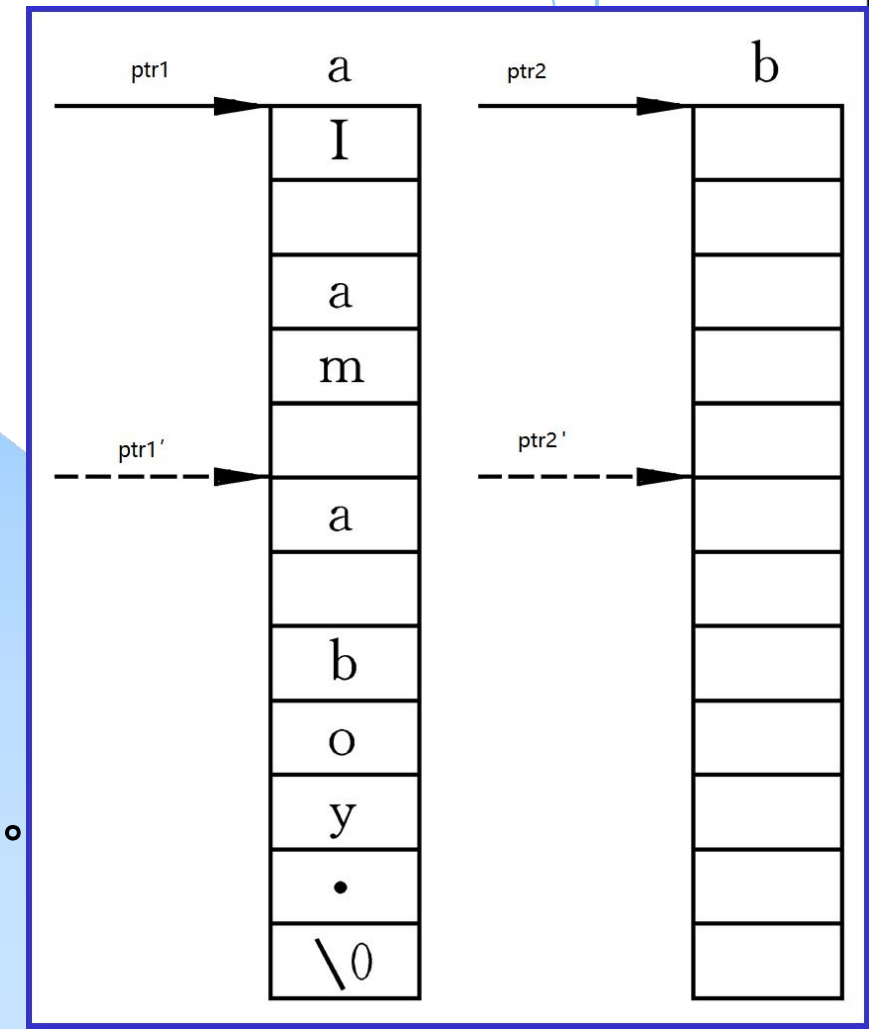
使用指针变量，用其值的改变来指向字符串中不同的字符。

例7.18 用指针变量来处理例7.1 7问题。

思路：定义2个指针变量p1和p2，分别指向字符数组a和b。改变指针变量p1和p2的值，使他们指向数组中的各元素，进行对应元素的复制。

```
#include <stdio.h>
int  main()
{
    char a[] ="I am a teacher.",b[20],*ptr1,*ptr2;
    int i;
    ptr1=a;ptr2=b;
    for(;*ptr1!='\0';ptr1++,ptr2++)
          *ptr2=*ptr1;
    *ptr2='\0’ ;
     printf("string a is:%s\n",a);
     printf("string b is:%s\n",b);
}
```

程序必须保证使ptr1和ptr2同步移动。



7.4.2 字符指针作函数参数

例7.19-1 用函数调用实现字符串的复制

(1) 用字符数组名作参数

```
#include <stdio.h>
int main()
{       void  copy_string(char from[], char to[]);
        char a[]="I am a teacher.";
        char b[]="you are a student.";
        printf("string a=%s\n string b=%s\n",a,b);
        printf("copy string a to string b:\n ");
        copy_string (a,b);
        printf("\nstring a=%s\nstring b=%s\n",a,b);
}
```

```
void copy_string(char from[], char to[])
{   int i=0;
    while(from[i]!='\0')
    {
          to[i]=from[i];
          i++;
    }
    to[i]='\0';
}
```

程序运行结果如下:

string a=I am a teacher.

string b=you are a student.

copy string a to string b:

string a=I am a teacher.

string b=I am a teacher.

（2） 形参用字符指针变量

例7.19-2 用字符指针实现字符串的复制

```
#include <stdio.h>
void  copy_string(char *from, char *to);
int main()
{
        char * a ="I am a teacher.";    //a是char*型指针变量
        char b[]="you are a student.";  //b为字符数组
        char *p=b;                      //使指针变量p指向b数组首元素
        printf("\nstring a=%s\nstring b=%s\n",a,b);
        printf("copy string a to string b:\n ");
        copy_string (a,p);       /调用copy_string函数，实参为指针变量
        printf("\nstring a=%s\nstring b=%s\n",a,b);
}
```

```
void  copy_string(char *from,char *to)
{   for(;*from!='\0';from++,to++)
    *to= *from;
    *to='\0';
}
```

（３） 对 copy_string 函数还可作简化

1、将copy_string函数改写为

```
void  copy_string (char *from,char *to)
{   while((*to=*from)!='\0')
    {to++;from++;}
 }
```

2、copy_string函数的函数体还可改为

```
{
    while((*to++=*from++)!='\0');
}
```

3、copy_string函数的函数体还可写成

```
{
    while(*from!='\0')
      *to++=*from++;
      *to='\0';
 }
```

4、copy_string函数的函数体还可写成

```
{
    while(*from)
      *to++=*from++;
      *to='\0';
 }
```

5、上面的while语句还可进一步简化为:

while(*to++=*from++);

它与下面语句等价:

while((*to++=*from++)!='\0');

将*from赋给*to,如果赋值后的*to值等于'\0',则循环终止('\0'已赋给*to)

6、函数体中while语句也可以改用for语句:

for(;(*to++=*from++)!=0;);

或

for(;*to++=*from++;);

7、也可用指针变量,函数copy_string可写为

```
void  copy_string (char from[ ],char to[ ])
{   char *p1,*p2;
    p1=from;p2=to;
    while((*p2++=*p1++)!='\0');
}
```

7.4.3 对使用字符指针变量和字符数组的讨论

虽然用字符数组和字符指针变量都能实现字符串的存储和运算，但它们二者之间是有区别的，主要有以下几点：

(1) 字符数组由若干个元素组成,每个元素中放一个字符，而字符指针变量中存放的是地址(字符串第1个字符的地址)，决不是将字符串放到字符指针变量中。

(2)赋值方式

对字符数组只能对各个元素赋值,不能用以下办法对字符数组赋值。

```
char  str[14];
str="I love China!";
```

而对字符指针变量,可以采用下面方法赋值:

```
char  *a;
a="I love China!";
```

但注意赋给a的不是字符,而是字符串第一个元素的地址。

(3) 初始化的含义

对字符指针变量赋初值:

char *a="I love China!";等价于

char *a;

a="I love China!";

而对数组的初始化:

char str[14]={"I love China!"};

不能等价于

char str[14];

str[ ]="I love China!";

数组可以在定义时对各元素赋初值，但不能用赋值语句对字符数组中的全部元素进行整体赋值

(4) 如果定义了一个字符数组，在编译时为它分配内存单元，它有确定的地址。

而定义一个字符指针变量时，给指针变量分配内存单元，在其中可以放一个字符变量的地址，该指针变量可以指向一个字符型数据，但如果未对它赋予一个地址值，则它并未具体指向一个确定的字符数据。

如: **char str[10];**

scanf（"% s "， str）； 是可以的。

而用下面的方法,目的是想输入一个字符串， 虽然一般也能运行，但这种方法是危险的：

```
char * a ;
scanf（"% s "， a ）;
```

应当这样:

```
char *a, str[10];
a=str;
scanf("%s",a);
```

(5) 指针变量的值是可以改变的，而字符数组名代表一个固定的值（数组首元素的地址），不能改变。

例7.20　改变指针变量的值

```
#include <stdio.h>
int main()
{
    char *a="I love China!";
    a=a+7;
    printf("%s",a);
}
```

运行结果

China!

需要说明，若定义了一个指针变量，并使它指向一个字符串，就可以用下标形式引用指针变量所指的字符串中的字符。

例7.21用带下标的字符指针变量引用字符串中的字符。

```
 int main()
{
    char *a="I love China!";
    int i;
    printf ("The sixth character is %c\n",a[5]);
    for(i=0;a[i]!='\0';i++)
      printf("%c",a[i]);
}
```

运行结果

```
The sixth character is e
I love China!
```

（6）字符数组中各元素的值可以改变，但是指针变量指向的字符串常量中的内容是不可以再赋值的。

char a[]=“House”;//字符数组a初始化

char *b=“House”; //字符指针变量b指向字符串常量的第一个字符

a[2]=‘r’; //合法，r取代a数组元素a[2]的原值u

b[2]=‘r’; //非法，字符串常量不能改变

(7)引用数组元素

对字符数组元素可以用下标法（用数组名和下标）引用一个数组元素（如：a[5]）的方法：

地址法（如：*（a+5））引用数组元素a[5]。

但是，如果指针变量没有指向数组，则无法用p[5]或者*（p+5）形式引用数组中的元素。

若字符指针变量p指向字符串常量，就可以用指针变量带下标的形式引用所的字符串中的字符。

char *a=“I love China!” 此时a[5] 的值是字母 ‘e’

(8)用指针变量指向一个格式字符串，可以用它来代替printf函数中的格式字符串。

char * format;

format=“a=%d，b=%f\n”；

printf(format,a,b)

相当于

printf(“a=%d，b=%f\n”,a,b);

只要改变指针变量format所指向的字符串，就可以改变输入输出的格式。这种printf函数称为可变格式输出函数。

- 7.5 指向函数的指针

7.5 指向函数的指针

7.5.1 函数指针的定义

指针变量可以指向整型变量、字符串、数组，也可以指向一个函数。

如果在程序中定义了一个函数，在编译时会把函数的源代码转换为可执行代码并分配一段存储空间。这段内存空间有一个起始地址，也称为函数入口地址。每次调用时，都从改地址入口开始执行此段函数代码。

该函数在编译时被分配给一个入口地址，该**入口地址称为函数的指针。**

7.5 指向函数的指针

7.5.1 函数指针的定义

函数名就是函数的指针，它代表函数的起始地址。

指向函数的指针变量：int （*ptr）（int，int）；

即：定义ptr为指向函数的指针变量，它可以指向函数类型为整型且有两个整型参数的函数。

此时函数的类型是：int （*ptr）（int，int）；

7.5.2 用函数指针变量调用函数

调用一个函数，除了可以通过函数名调用之外，还可以通过指向函数的指针变量来调用该函数。

例7.22 用函数求 a 和 b 中的大者。

1）通过函数名调用

1）通过函数名调用函数

```
#include <stdio.h>
int main()
{ int  max(int,int);
  int a,b,c;
  scanf("%d,%d",&a,&b);
  c=max(a,b);
  printf("a=%d,b=%d,max=%d",a,b,c);
}

int  max(int x,int y)
{   int z;
    if(x>y)z=x;
    else      z=y;
    return(z);
}
```

2）通过指针变量调用它所指向的函数

将 **main** 函数改写为

```
#include <stdio.h>
int main()
{  int  max(int,int);     //函数声明
   int (*ptr)(int,int);      //定义指向函数的指针变量p
   int a,b,c;
   ptr=max;                  //使p指向max函数
   scanf("%d,%d",&a,&b);
   c=(*ptr)(a,b);            //通过指针变量调用max函数
   printf("a=%d,b=%d,max=%d",a,b,c);
}
```

7.5.3 定义和使用指向函数的指针变量

一般形式：

类型名 （*指针变量名）（函数参数表列）；

如：int（*ptr） （int，int）

注释：

（1）只能指向其定义时指定的类型的函数；

（2）指针调用函数，必须先使其指向该该函数；

（3）给函数指针变量赋值时，只需给出函数名而不必要给参数；

7.5.3 定义和使用指向函数的指针变量

（4）调用时，只须将（*ptr）代替函数名即可，在（*ptr）之后的（）内根据需要写上实参。

（5）对指向函数的指针变量不能进行算术运算，如ptr+n, ptr++, ptr--等运算。

（6）用函数名调用函数，只能调用所指定的一个函数，而通过指针变量调用函数比较灵活，可以根据不同情况调用不同的函数。

例输入两个整数，然后让用户选择1或2，选择1则调用max函数，选择2则调用min函数，输出二者中的小数。

见例7.23

例7.23

```
#include <stdio.h>
int main()
{  int  max(int,int);     //函数声明
   int  min(int,int);     //函数声明
   int (*ptr)(int,int);     //定义指向函数的指针变量p
   int a,b,c,n;
   printf("please enter a and b:");
   scanf("%d,%d",&a,&b);
   printf("please choose 1 or 2:");
   scanf("%d",&n);
   if (n==1) ptr=max;              //通过指针变量调用max函数
   else if (n==2) ptr=min;     //通过指针变量调用max函数
   c=(*ptr)(a,b);                    //通过指针变量调用p指向的函数
   printf("a=%d,b=%d\n",a,b);
   if (n==1) printf("max=%d\n",c);
   else printf("min=%d\n",c);
   return 0;
}
```

```
int max(int x,int y)
{
  int z;
  if(x>y) z=x;
  else z=y;
  return(z);
}

int min(int x,int y)
{
  int z;
  if(x<y) z=x;
  else z=y;
  return(z);
}
```

7.5.4 用指向函数的指针作函数参数

函数指针变量常见的用途之一是把指针作为参数传递到其他函数。函数的参数可以是变量、指向变量的指针变量、数组名、指向数组的指针变量等。

指向函数的指针也可以作为函数参数，把函数的入口地址传给形参，这样就能够在被调用的函数中使用实参函数。

7.5.4 用指向函数的指针作函数参数

基本原理：有一个函数（假设函数名为fun），它有两个形参（x1和x2），定义x1和x2为指向函数的指针变量。在调用函数fun时，实参为两个函数名f1和f2，给形参传递的是函数f1和f2的地址。这样在函数fun中就可以调用f1和f2函数了。例如：

实参函数名 f1 f2

↓ ↓

```
void fun(int (*x1)(int),int (*x2)(int,int))
{  int a,b,i=3,j=5;
    a=(*x1)(i);      /*调用f1函数，i是实参*/
    b=(*x2)(i,j);     /*调用f2函数，i，j是实参*/
}
```

例7.24 设一个函数fun，在调用它的时候，每次实现不同的功能。输入a和b两个数，第一次调用fun时找出a和b中大者，第二次找出其中小者，第三次求a与b之和。

例7.24

```
#include <stdio.h>
int max(int,int);   /* 函数声明 */
int min(int,int);   /* 函数声明 */
int add(int,int);   /* 函数声明 */
int fun(int x,int y,int (*ptr)(int,int));
int main()
{
    int a,b,n;
    printf("please enter a and b:");
    scanf("%d,%d",&a,&b);
    printf("please choose 1,2,3:");
    scanf("%d",&n);
    if (n==1) fun (a,b,max);
    else if (n==2) fun(a,b,min);
    else if (n==3) fun(a,b,add);
    return 0;
}
```

```
int fun(int x,int y,int (*ptr)(int,int)) /* 定义fun函数*/
{
    int result;
    result=(*ptr)(x,y);
    printf("%d\n",result);
}

int  max(int x,int y)          /* 定义max函数*/
{    int z;
     if(x>y)z=x;
     else z=y;
     printf("max=");
     return(z);
}
```

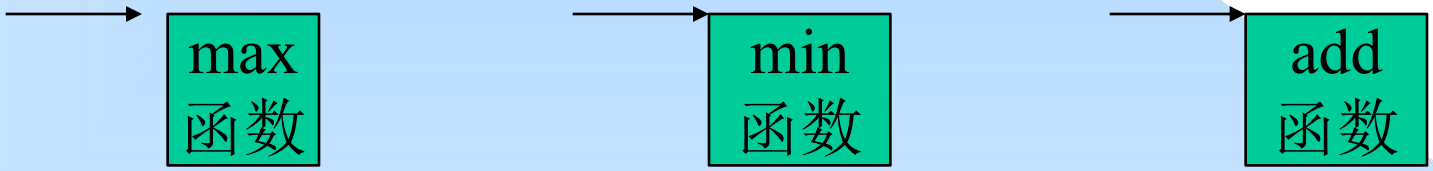
```
int min(int x,int y)          /* 定义min函数 */
{    int z;
     if(x<y) z=x;
     else z=y;
     printf("min=");
     return(z);
}

int add(int x,int y)          /* 定义add函数 */
{    int z;
     z=x+y;
     printf("sum=");
     return(z);
}
```

程序分析

在定义fun函数时，在函数的首部用int（*ptr)(int,int）声明形参p是指向函数的指针，该函数是整型函数，有两个整型形参。max，min和add是三个自定义函数。

当输入1时，调用fun函数，除了将a和b作为实参，将整数a和b传给形参x和y之外，还将函数名max作为实参将其入口地址传给fun函数中的形参ptr（ptr是指向函数的指针变量）。

此时fun函数中的（*ptr）(x,y)相当于max(x,y）。



- 7.6 返回指针值的函数

7.6 返回指针值的函数

一个函数可以返回一个整型值、字符值、实型值等，也可以返回指针型的数据，即地址。

这种返回指针值的函数，一般定义形式为：

类型名 *函数名（参数表列）；

例如：

```
int  *a（int x，int y）
```

例7.25 有a个学生的成绩（每个学生有 b门课程），要求在用户输入学生序号以后，能输出该学生的全部成绩。用指针函数来实现。

解决思路：定义一个二维数组score，用来学生成绩。

定义一个查询学生成绩的函数search，它是一个返回指针的函数，形参是指向一维数组的指针变量和整型变量n，从主函数将数组名score和要找的学生好k传递给形参。函数的返回值是&score[k][0]（即存放序号为k的学生的序号为0的课程的数组元素的地址）。然后在主函数中输出该生的全部成绩。

```
#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 *ptr;
        int i,m;
        printf("Enter the number of  student:");
        scanf("%d",&m);          //输入要找的学生序号
        printf("The scores of No.%d are:\n",m);
        ptr=search(score,m);     //调用search函数，返回score[k][0]的地址
        for(i=0;i<4;i++)
            printf("%5.2f\t",*(ptr+i)); //输出score[k][0]~score[k][3]的值
        printf("\n");
        return 0;
}
```

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

运行情况如下：

enter the number of student：1↙

The scores of No. 1 are：

56.00　　89.00　　67.00　　88.00

例7.26 找出其中有不及格课程的学生及其学生号。

```
#include <stdio.h>
int main()
{       float score[][4]={{60,704,80,90},{56,89,67,88},{34,78,90,66}};
        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");
                 }
         }
        return 0;
}
```

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

• 7.7 指针数组

7.7指针数组和指向指针的指针

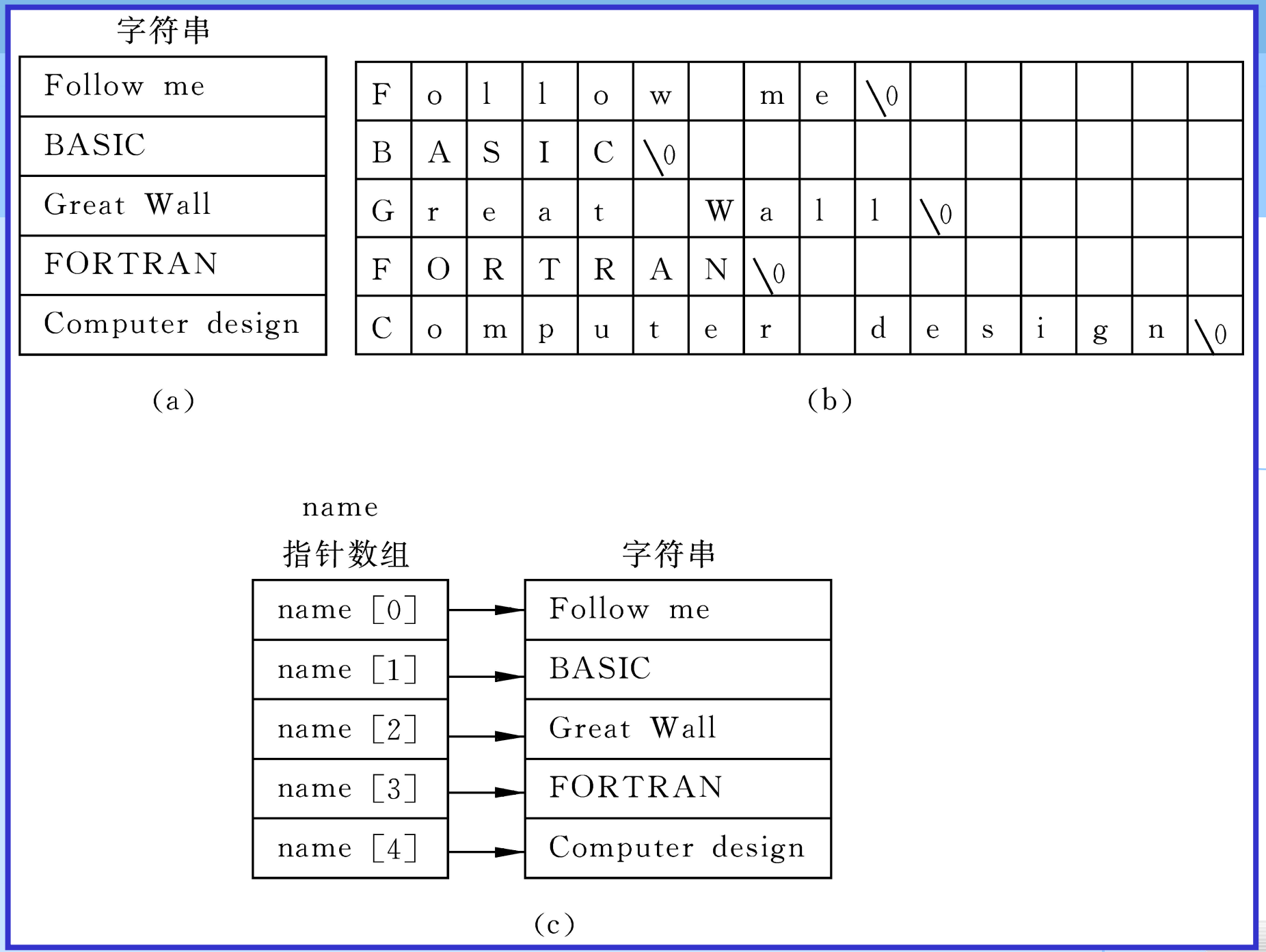
7.7.1指针数组的概念

一个数组，若其元素均为指针类型数据，称为**指针数组**，也就是说，指针数组中的每一个元素都相当于一个指针变量。一维指针数组的定义形式为

类型名 数组名［数组长度］；

例如：

```
int *p[4];
```


字符串
Follow me
BASIC
Great Wall
FORTRAN
Computer design
F o l l o w m e \0
B A S I C \0
G r e a t W a l l \0
F O R T R A N \0
C o m p u t e r d e s i g n \0
(a)
(b)
name
指针数组
字符串
name [0]
name [1]
name [2]
name [3]
name [4]
Follow me
BASIC
Great Wall
FORTRAN
Computer design
(c)

例7.26 将若干字符串按字母顺序（由小到大）输出。

```
#include <stdio.h>
#include <string.h>
void sort(char *name[],int n);
void print(char *name[],int n);
int  main()
{
   char *name[]={"Follow me","BASIC","Great Wall","FORTRAN",
"Computer design"};
    int n=5;
    sort(name,n);
    printf(name,n);
    return 0;
}
```

```
void sort(char *name[],int n)
{
    char *temp;
    int i,j,k;
    for(i=0;i<n-1;i++)
   {if k=i;
      for(j=i+1;j<n;j++)
           if(strcmp(name[k],name[j])>0) k=j;
     if(k!=i)
           temp=name[i];
        name[i]=name[k];
        name[k]=temp;}
}
```

```
void print(char *name[],int n)
{
    int i;
    for(i=0;i<n;i++)
        printf("%s\n",name[i]);
}
```

运行结果为:

BASIC
Computer design
FORTRAN
Follow me
Great Wall

7.7.2 指向指针数据的指针变量

怎样定义一个**指向指针数据的指针变量**呢? 如下:

char< **p;

p的前面有两个*号。*运算符的结合性是从右到左,因此**p相当于*(*p),显然*p是指针变量的定义形式。

如果没有最前面的*,那就是定义了一个指向字符数据的指针变量。现在它前面又有一个*号,表示指针变量p是指向一个字符指针变量的。*p就是指向的指针型数据的指针变量。

例7.28 使用指向指针的指针

编程思路：定义一个指针数组name，并对它进行初始化，使name数组中每一个元素分别指向5个字符串。定义一个指向指针型数据的指针变量p，使p先后指向name数组中各元素，输出这些元素所指向的字符串。

例7.28 使用指向指针的指针

```
#include <stdio.h>
int  main()
{  char *name[]={"Follow me","BASIC","Great Wall", "FORTRAN",
"Computer design"};
    char **p;   //指向指针的指针
    int i;
    for(i=0;i<5;i++)
        {
        p=name+i;
        printf("%s\n",*p);
        }
}
```

例7.29 一个指针数组的元素指向整型数据的简单例子

```
#include <stdio.h>
int main()
{  int a[5]={1,3,5,7,9};
    int *num[5]={&a[0],&a[1],&a[2],&a[3],&a[4]};
    int  **p,i;
    p=num;
        for(i=0;i<5;i++)
        {
         printf("%d    ",**p);
         p++;
        }
}
```

7.7.3 指针数组作main函数的形参

指针数组的一个重要应用是作为main函数的形参。在以往的程序中，main函数的第一行一般写成以下形式：

int **main()** 或者int main(void)

实际上，main函数可以有参数，例如：

int **main** (int argc, char *argv[ ])

其中的argc和argv就是main函数的形参。他们是程序的命令行参数。argc(argument count，即参数个数)，argv(argument vector,即参数向量)，它是一个*char 指针数组数组中的每个元素（其值为指针）指向命令行中的一个字符串的首字符。

7.7.3 指针数组作main函数的形参

注意：如果用带参数的main函数，其第一个形参必须是int型，用来接收参数个数；第二个形参必须是字符指针数组，用来接收从操作系统传来的字符串中首字符的地址。

main函数和其他函数组成一个文件模块，有一个文件名。
是由操作系统调用的。那么，main函数的形参的值从何处得到呢？显然不可能在程序中得到。
实际上实参是和命令一起给出的。也就是在一个命令行中包括命令名和需要传给main函数的参数。命令行的一般形式为**命令名 参数1 参数2……参数n**

如果有一个名为file1的文件,它包含以下的main函数:

```
int  main(int argc,char *argv[ ])
{
    while(argc>1)
    { ++argv;
     printf("%s\n", argv);
       --argc;
     }
}
```

在DOS命令状态下输入的命令行为

file1 China Beijing

则执行以上命令行将会输出以下信息:

China

Beijing

7.8 小结

7.8.1指针的数据类型的小结

变量定义	类型表示	含义
int i ;	int	定义整型变量 i
int * p ;	int *	p 为指向整型数据的指针变量
int a[n];	int []	定义整型数组 a，它有 n 个元素
int *p[n];	int * [4]	定义指针数组 p，它由 n 个指向整型数据的指针元素组成
int (*p)[n];	int (*) [4]	p 为指向含 n 个元素的一维数组的指针变量
int f();	int ()	f 为带回整型函数值的函数
int *p();	int *()	p 为带回一个指针的函数，该指针指向整型数据
int (*p)();	int (*) ()	p 为指向函数的指针，该函数返回一个整型值
int ** p ;	int **	p是一个指针变量，它指向一个指向整型数据的指针变量
void *p	void *	p是一个指针变量，基类型为void（空类型），不指向具体的对象

7.8.2 指针运算小结

(1) 指针变量加（减）一个整数

例如：p++、p--、p+i、p-i、p+=i、p-=i等。

(2) 指针变量赋值

将一个变量地址赋给一个指针变量。如:

p=&a;（将变量a的地址赋给p)

p=array;（将数组array首元素地址赋给p)

p=&array[i]; (将数组array第i个元素的地址赋给p)

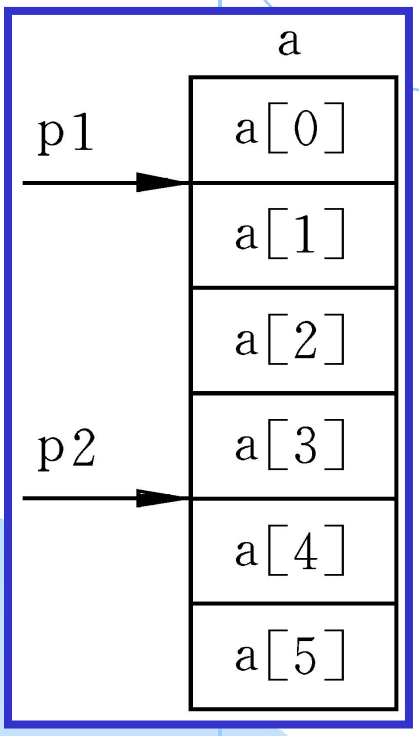
p=max; (max为已定义的函数,将max的入口地址赋给p)

p1=p2; (p1和p2都是指针变量,将p2的值赋给p1)

(3) 指针变量可以有空值，即该指针变量不指向任何变量，可以这样表示：p=NULL;

(4) 两个指针变量可以相减

如果两个指针变量都指向同一个数组中的元素，则两个指针变量值之差是两个指针之间的元素个数。



(5) 两个指针变量比较

若两个指针指向同一个数组的元素，则可以进行比较。指向前面的元素的指针变量“小于”指向后面元素的指针变量。

7.8.3 void指针类型

ANSIC新标准增加了一种“void”指针类型，即可定义一个指针变量，但不指定它是指向哪一种类型数据的。

ANSIC标准规定用动态存储分配函数时返回void指针，它可以用来指向一个抽象的类型的数据，在将它的值赋给另一指针变量时要进行强制类型转换使之适合于被赋值的变量的类型。例如:

```
char< *p1;
void< *p2;
…
p1=(char *)p2;
```

同样可以用(void *)p1将p1的值转换成void *类型。如:

p2=(void *)p1;

也可以将一个函数定义为void *类型,如:

void *fun(char ch1,char ch2)

表示函数fun返回的是一个地址,它指向“空类型”,如需要引用此地址,也需要根据情况对之进行类型转换,如对该函数调用得到的地址要进行以下转换:

p1=(char *)fun(ch1,ch2);

Questions & Answers