



# 第10讲 指针与引用 (Part II)

---

周水庚

2017-11-30



# 提要

---

- 指针形参
- 数组形参
- 字符指针形参
- 函数形参
- 返回指针值的函数



# 提要

---

- 指针形参
- 数组形参
- 字符指针形参
- 函数形参
- 返回指针值的函数



# 函数形参

- 函数设置形参的**目的**是让函数被调用时，能从调用处获得数据或指针信息
- **C语言关于函数形参遵守以下规定**
  - 函数调用时，形参从实参获得初值
  - 函数形参可看作函数内部的局部变量，函数体内对形参的修改不会影响实参本身
- 函数形参的作用由形参的类型确定
  - **基本类型、指针类型、数组类型**
  - 当函数的形参是某种指针类型或数组类型时，形参从实参处得到某环境变量的地址，函数体就能通过指针形参间接引用环境变量，并能改变环境变量的值



# 指针类型形参

---

- 指针形参从实参处得指针值。它使函数得到了调用环境中某变量的地址，函数可用这个地址间接访问函数之外的变量，或引用其值，或修改其值
- 指针类型形参为函数改变调用环境中的数据对象提供了手段

# 指针类型形参示意程序-1

- 对于简单类型形参，实参向形参传值，函数不能改变实参变量值的示意程序

```
#include <stdio.h>
void swap(int u, int v)
{ int t = u; u = v; v = t;
  printf("In swap: u = %d, v = %d\n", u, v);
}
void main()
{ int x = 1, y = 2;
  swap(x, y);
  printf("In main: x = %d, y = %d\n", x, y);
}
```

输出结果:

```
In swap: u=2, v=1
In swap: x=1, y=2
```

# 指针类型形参示意程序-2

```
#include <stdio.h>
void swap-1(int u, int v); void swap-2(int *pu, int *pv);
void main()
{ int a = 1, b = 2; printf("Befor swap. a = %d\tb = %d\n", a, b);
  swap-1(a, b); /* 以变量的值为实参 */
  printf("After swap: a = %d\tb = %d\n", a, b);
  swap-2(&a, &b); /* 以变量的指针为实参 */
  printf("After swap: a = %d\tb = %d\n", a, b);
}
```

a=1 b=2

a=1 b=2

a=2 b=1

```
void swap-1(int u, int v)
{ int t;
  t = u; u = v; v = t;
}
```

```
void swap-2(int *pu, int *pv)
{ int t;
  t = *pu; *pu = *pv; *pv = t;
}
```



# 指针类型形参(续)

---

- 如希望函数能按需要任意引用指定的变量，需要在三个方面协调一致
  - 函数应设置指针类型的形参
  - 函数体必须通过指针形参间接访问变量，或引用其值，或修改其值
  - 调用函数时，要以欲改变值的变量的指针为实参调用函数



```

#include<stdio.h>
void f1(int x, int y){int t=x;x=y;y=t;}
void f2(int *x, int *y){int t=*x; *x=*y; *y=t;}
void f3(int **x, int **y){int *t=*x; *x=*y;*y=t;}
Void main()
{  int x=1, y=2;
   int *xpt=&x,*ypt=&y;
   printf(First: x=%d y=%d\n\n,x,y);

   int x=1, y=2;f1(x,y);
   printf(After call f1(): x=%d y=%d\n\n,x,y);

   int x=1, y=2;f2(&x,&y);
   printf(After call f2(): x=%d y=%d\n\n,x,y);

   int x=1, y=2;
   printf(Before call f3(): x=%d y=%d\n\n,x,y);
   printf(Before call f3(): *xpt=%d *ypt=%d\n\n,*xpt,*ypt);

   f3(&xpt,&ypt);
   printf(After call f3(): x=%d y=%d\n\n,x,y);
   printf(After call f3(): *xpt=%d *ypt=%d\n\n,*xpt,*ypt);
}

```

1 2

1 2

2 1

1 2

1 2

1 2

2 1



# 提要

---

- 指针形参
- **数组形参**
- 字符指针形参
- 函数形参
- 返回指针值的函数

# 数组类型形参

- 使函数能处理不同的成组变量，应使用**数组类型形参**
- 对应数组形参的实参是数组某元素的地址，最通常的情况是数组首元素的地址。由于数组名能代表数组首元素的地址，所以常用数组名实参对应数组形参
  - 例如，下面定义的函数sum()用于求数组的前n个数之和

```
int sum(int a[], int n)
{ int i, s;
  for(s = 0, i = 0; i < n; i++)  s += a[i];
  return s;
}
```
  - 函数sum有两个形参，一个形参是数组类型的，用于对应可变的实在数组；另一个形参是整型的，用于指定求和数组的元素个数

# 数组类型形参(续)

- 前面定义的函数**sum()**用于求数组的前**n**个数之和

- 利用函数**sum()**，如有以下变量定义

```
int x[] = {1, 2, 3, 4, 5};
```

```
int i, j;
```

- 则语句 `i = sum(x, 5); j = sum(&x[2], 3);`  
`printf("i = %d\nj = %d\n", i, j);`

- 将输出 `i = 15`  
`j = 12`

- 函数调用**sum(x,5)**将数组**x**的地址(**&x[0]**)传送给形参**a**；  
函数调用**sum(&x[2], 3)**将数组**x**中的**x[2]**的地址(**&x[2]**)  
传送给形参**a**，而**x[2]**的地址就是数组元素段**x[2]**、**x[3]**、  
**x[4]**的开始地址

# 数组类型形参(续)

- 与数组类型形参对应的实在数组有多少个元素是不确定的，可能会对应一个大数组，也可能对应一个小数组，甚至会对应数组中的某一段。所以在数组形参说明中，形参数组不必指定数组元素的个数。  
*为了正确指明某次函数调用实际参与计算的元素个数，应另引入一个整型形参来指定*
- 因传递给数组形参的实参是数组段的开始地址，函数内对数组形参的访问就是对实参所指数组的访问。  
函数也可以改变实参所指数组元素的值

# 数组类型形参示例

- 函数 `initArray()` 是给数组元素赋指定值的

```
void initArray(int x[], int n, int val)
{ int i;
  for(i = 0; i < n; i++) x[i] = val;
}
```

- 如另有数组定义 `int a[10], b[100];`

- 语句 `initArray(a, 10, 1);`

```
initArray(b, 50, 2);
```

```
initArray(&b[50], 50, 4);
```

- 分别给数组 `a` 的所有元素置值 `1`，为数组 `b` 的前 `50` 个元素置值 `2`，后 `50` 个元素置值 `4`

# 数组类型形参(续)

- 数组形参也可以是多维数组。当数组形参是多维时，除数组形参的第一维大小不必指定外，其它维的大小必须明确指定

- 如下面的函数**sumAToB()**，用于将一个**n**行**10**列的二维数组各行的**10**个元素之和存于另一个数组中

```
void sumAToB(int a[][10], int b[], int n)
{ int i, j;
  for(i = 0; i < n; i++)
    for(b[i] = 0, j = 0; j < 10; j++)
      b[i] += a[i][j];
}
```

- 在函数**sumAToB()**的定义中，对形参**a**的说明如写成  
**int a[][];** 是错误的



## 数组类型形参(续)

- 因二维数组的元素只是按行存放，如在数组形参中不说明它的列数，就无法确定数组元素 $a[i][j]$ 的实际地址
- 在实际使用数组形参编写函数时，注意程序中实参所指数组大小不能小于函数实际要用的数组的大小。如实参数组太小，因系统不检查超界情况，可能会使用实际数组后面别的变量的空间，这会发生意想不到的错误





# 数组类型形参函数示例-1

---

- 为数组输入值的函数

```
void readArray(int a[], int n)
{ int i;
  for(i = 0; i < n; i++) {
    printf("Enter a[%d] ", i);
    scanf("%d", &a[i]);
  }
}
```



## 数组类型形参函数示例-2

- 求数组中最大元素值的函数

```
int max(int a[], int n)
{ int i, m;
  for(m = 0, i = 1; i < n; i++)
    if (a[m] < a[i]) m = i;
  return a[m];
}
```

# 数组类型形参即指针形参

- 数组形参对应的实参也可以是数组某元素的地址，即数组某元素的指针，所以数组形参也是一种**指针形参**

- 所以任何数组形参说明

**类型 形参名[]**

- 都可改写成

**类型 \*形参名**

- 如前面的函数sum()的定义可改写成如下形式

```
int sum(int *a, int n)
{ int i, s;
  for(s = i = 0; i < n; i++) s += a[i];
  return s;
}
```

# 数组类型形参-指针形参示例

- 函数形参也是函数的局部变量，指针形参就是函数的指针变量，函数**sum()**的定义又可改写成如下形式

```
int sum(int *a, int n)
{ int s = 0;
  for(; n--;)
    s += *a++;
  return s;
}
```

# 数组类型形参-指针形参示例(续)

- 为使函数对数据进行操作更具一般性，可为函数设置数据的首元素指针和元素个数等形参，而函数体用指向数据元素的指针对数组作处理
  - 如函数**sum()**也可改写成以下形式

```
int sum(int *a, int n)
{ int *ap, s;
  for(s = 0, ap = a; ap < a+n; ap++)
    s += *ap;
  return s;
}
```

# 数组类型形参程序示例

- 设某个班有若干位学生，共学5门功课。学生成绩单的学号和各门功课成绩存放在二维数组s[][]中，s的每一行存储一位学生的有关信息，其中每行的第一列存放学生的学号
- 程序设计分析
  - 设计函数 `search(int (*p)[6], int m, int no)` 用于寻找指定学号的学生，并输出该生的各门功课成绩
  - 其中，形参p是指向数组的指针，它所指数组有6个int型元素；形参m是学生数；形参no是待寻找学生的学号
  - 主函数按以下格式调用 `search(s, m, n)`;
  - 其中s为成绩单首行数组的指针；m为成绩单行数，即学生人数；n为某学生的学号

# 数组类型形参程序示例(续)

- 学生成绩单  
程序(续)-P1

```
#include <stdio.h>
#define MAXN 3
int s[MAXN][6] = {{ 5, 70, 80, 96, 70, 90},
                  { 7, 40, 80, 50, 60, 80},
                  { 8, 50, 70, 40, 50, 75}};

void search(int (*)[6], int, int);
void main()
{ search(s, MAXN, 7);
  search(s, MAXN, 5);
  search(s, MAXN, 8);
  search(s, MAXN, 2);
}
```

# 数组类型形参程序示例(续)

## ■ 学生成绩单 程序(续)-P2

```
void search(int (*p)[6], int m, int no)
{ int (*ap)[6], *pp;
  for(ap = p; ap < p+m; ap++)
    if (**ap == no) {
      printf("%d 号学生是 :\n", no);
      for(pp = *ap+1; pp <= *ap+5; pp++)
        printf("%4d\t", *pp);
      printf("\n");
      return;
    }
  printf("没有 %d 号的学生.\n", no);
}
```



# 数组类型形参程序示例(续)

## ■ 学生成绩单程序(续)

- 内循环的`ap`是找到的那位学生的信息数组指针，`*ap`是该数组的第一个元素的指针，`*ap+1`是第二个元素的指针
- 利用二维数组元素按行连续存放的规则，如有必要，二维数组也可看作一维数组。如下面的代码段能累计全班学生的全部成绩之和，并存于变量`total`中

```
{ int *pp = *p, i;  
  for(total = 0, i = 0; i < MAXN*6; i++; pp++)  
    if (i%6) /* 跳过学号 */  
      total += *pp;  
}
```

- 工作变量`pp`是一个指向`int`型的指针，每次循环增加1个单位，遍历整个二维数组 `s`



# 提要

---

- 指针形参
- 数组形参
- **字符指针形参**
- 函数形参
- 返回指针值的函数



# 字符指针形参

---

- **字符指针形参**，是说它指向字符串的首字符，对应的实参或是字符数组某个元素的指针，或是字符串的首字符指针
- 字符串就是元素为字符的一维数组。所以字符指针形参与指向数组元素指针形参有相同的使用方法。但因字符串的特殊性，在编写字符串处理函数时还会有许多技巧

# 字符指针形参示例-1

## ■ 字符串拷贝函数 `strcpy()`

- 实现该函数功能的代码前面讨论过，这里把它改写成函数，是将一个已知字符串内容复制到另一字符数组中
- 拷贝函数设有两个形参 `from`，`to`。`from`为已知字符串的首字符指针，`to`为存储复制的字符串首字符指针

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

- 由于字符串结束符 '`\0`'的ASCII码值为0，因此上述测试当前拷贝字符不是字符串结束符的代码 "`!= '\0'`"可以省略。函数可简写为

```
void strcpy (char *to, char *from)
{ while (*to++ = *from++);
}
```

# 字符指针形参示例-2

## ■ 两字符串比较函数 `strcmp()`

- 比较两字符串大小的函数`strcmp()`有两个形参`s`和`t`；分别为两个待比较字符串的首字符指针。如`s`所指的字符串小于`t`所指的字符串，函数返回值小于零；如`s`所指字符串大于`t`所指字符串，函数返回值大于零；如两个字符串相同，则函数返回零

```
int strcmp(char *s, char *t)
{ while (*s == *t) { /* 对应字符相等循环 */
    if (*s == '\0') return 0;
    s++; t++;
}
return *s - *t; /* 返回比较结果 */
}
```

# 字符指针形参示例-3

## ■ 输入字符行函数 `getnch()`

- 函数 `getnch()` 输入字符行，输入的字符存于由指针形参指定的字符数组中。为避免输入过多字符，函数 `getnch()` 另设一个 `int` 型形参，它给出最多能存储的字符数

```
int getnch(char *s, int lim)
{ int c; char *ps = s;
  while (ps < s+lim-1&& c = getchar()) {
    *ps++ = c;
    if (c == '\n') break; /* 输入换行符结束*/
  }
  *ps = '\0';
  return ps - s; /* 返回实际输入存储的字符个数*/
}
```



# 提要

---

- 指针形参
- 数组形参
- 字符指针形参
- **函数形参**
  - **函数指针和函数指针变量、利用函数指针调用函数**
  - **函数指针形参、函数指针数组**
- 返回指针值的函数

# 函数指针和函数指针变量

- 在C语言中，函数不可设置函数形参，但函数可以设置函数指针形参，函数要调用的函数也可以由实参指定
- 为了实现函数通过实参调用其它的函数，C语言引入函数指针类型、函数指针和函数指针变量等概念
- 函数指针如同一般数据一样能复制、存贮，函数指针变量能作为复杂数据结构的成分，也能作为函数的形参等
- 函数的目标码有入口地址，这个入口地址就认作指向该函数的指针值，并用函数名来标记函数的入口地址
- 函数指针变量是一种特殊的指针变量，它能存储函数的指针值，使它指向某个函数。如同数据指针变量可间接访问它所指变量一样，程序也可利用函数指针变量间接调用它所指的函数



# 函数指针和函数指针变量(续)

- 为区别能指向不同特性函数的函数指针变量，用函数的返回值类型和函数的形参类型等来区分不同的函数指针，并以函数指针类型来标识
- 定义指向函数的指针变量的一般形式为  
函数返回值类型 (\* 指针变量名)(形参类型表):
  - 例如， `int (*fp)(int);`
  - 定义 `fp` 是一个能指向函数的指针变量，它能指向的函数的返回值必须是 `int` 型的，并有一个 `int` 类型的形参
  - **注意**， `*fp` 两侧的括弧是必需的，表示 `fp` 先与 `*` 结合，说明它是一个指针变量。然后与后随的 `()` 结合，表示指针变量 `fp` 是指向函数的。最前面的 `int` 表示所指向的函数应是返回 `int` 型值的函数
  - 而代码 `"int *f(int)"`，因 `()` 优先级高于 `*`，就变成是说明一个函数 `f()`，该函数的返回值是指向 `int` 型量的指针。

# 函数指针和函数指针变量(续)

- 指向函数的指针变量表示它能存放函数的入口地址，它能指的函数是函数指针变量定义时所规定的
- 在程序执行时，某个符合要求的函数入口地址都能赋给函数指针变量，使函数指针变量指向该函数，可以用该指针变量间接调用它所指函数
- 可根据需要向函数指针变量赋不同的函数入口地址，使它指向的函数可以按需要改变

# 利用函数指针调用函数

- 定义了指向函数的指针变量，就可向它赋某函数的入口地址
- **C语言约定，单独的函数名本身就是函数的入口地址(不能在函数名之后加上一对圆括号，否则变成函数调用)**
  - 如语句“ **fp = min;** ”使函数指针变量**fp**指向函数**min()**
- 当一个指向函数的指针变量确实指向某个函数时，就可用它调用它所指向的函数。**一般形式的函数调用是**  
**函数名(实参表)**
- **改用指向函数的指针变量调用该函数，就要改写成**  
**(\*函数指针变量名) (实参表)**



# 利用函数指针调用函数示例

- 求两个整数中小的值的函数**min()**，它返回**int**型值

- 如有

```
int (*fp)(int,int), x=5, y=8, z, min(int,int);
```

- 则语句

```
fp = min;
```

- 使指针变量**fp**指向函数**min()**

- 用**fp**间接调用函数**min()**，写成

```
z = (*fp)(x, y);
```

# 利用函数指针调用函数示例

## ■ 用函数指针变量调用函数的示意程序

```
#include <stdio.h>
void main()
{ int (*fp)(int, int), x, y, z;
  int min(int, int), max(int, int);
  printf("Enter x, y:");scanf("%d%d", &x, &y);
  fp = min; /*让fp指向函数min()*/
  z = (*fp)(x, y); /*调用fp所指函数*/
  printf("MIN(%d, %d) = %d\n", x, y, z);
  fp = max; /*现在更改fp, 使它指向函数max()*/
  z = (*fp)(x, y); /*调用fp所指函数*/
  printf("MAX(%d, %d) = %d\n", x, y, z);
}
```

```
int min(int a, int b)
{ return a < b ? a : b;
}
int max(int a, int b)
{
  return a > b ? a : b;
}
```



# 函数指针形参

---

- **为什么要为函数设函数指针形参？**
  - 设有一个函数**fun()**，它有一个函数指针形参**fp**和若干其他形参
  - 函数**fun()**利用函数指针形参**fp**和其他形参，根据给定的算法计算结果
  - 调用函数**fun()**时，除提供其它实参外，还得为它提供与形参**fp**相对应的实在函数指针
  - 这样，**用不同的实在函数指针调用函数**fun()**，就能获得不同的结果**



# 函数指针形参示例

---

- 对给定实数数表，求它的最大值、最小值和平均值
- 程序设计分析
  - 程序有三个函数`max()`、`min()`和`ave()`，另设一个函数`afun()`
  - 主函数调用函数`afun()`，并提供数组、数组元素个数和求值函数指针作为实参
  - 由函数`afun()`根据主函数提供的函数指针实参调用实际函数

# 函数指针形参示例(续)

- 给定实数数表，求最大值、最小值和平均值(续)

```
#include <stdio.h>
#define N sizeof a / sizeof a[0]
double max(double a[], int n)
{ int i; double r;
  for(r = a[0], i = 1; i < n; i++)
    if (r < a[i]) r = a[i];
  return r;
}
double min(double a[], int n)
{ int i; double r;
  for(r = a[0], i = 1; i < n; i++)
    if (r > a[i]) r = a[i];
  return r;
}
```

```
double ave (double a[], int n)
{ int i; double r;
  for (r = 0.0, i = 0; i < n; i++) r += a[i];
  return r/n;
}
double afun(double a[], int n,
            double (*f)(double *, int))
{
  return (*f)(a, n);
}
```





# 函数指针形参示例(续)

- 给定实数数表，求最大值、最小值和平均值(续)

```
void main()
{ double a[]={1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0, 9.0};
  double result;
  printf("\n\nThe results are : ");
  result = afun(a, N, max);
  printf("\t%s = %4.2f", "MAX", result);
  result = afun(a, N, min);
  printf("\t%s = %4.2f", "MIN", result);
  result = afun(a, N, ave);
  printf("\t%s = %4.2f", "AVERAGE", result);
  printf("\n\n\n");
}
```

# 函数指针数组

- 利用函数指针能被存储的性质，可将若干函数指针存于一数组中

- 如以下代码

```
double (*fpt[])(double*, int)={ max, min , ave}; /* 函数指针数组 */
```

- 定义了函数指针数组fpt[]，并用函数名对它初始化,将函数max()、min()和ave()的函数指针填写在函数指针数组fpt[]中，使数组fpt[]成为函数的入口表
- **例子：**设计一个通用的菜单处理函数，除其它有关菜单位置、大小、颜色等信息外，另设两个数组形参。一个数组元素为指向菜单项字符串的指针，另一个数组元素为指向对应处理函数的指针



# 函数指针数组示例

## ■ 通用的菜单处理函数(续)

```
#include <stdio.h>
#define N sizeof a / sizeof a[0]
void main ()
{ double a[]={1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0, 9.0};
  double result;
  double(*fpt[])(double *,int)={ max, min , ave};
  char *title[]={"最大值", "最小值", "平均值"};
  char *menuName[] ={"求最大值", "求最小值", "求平均值", ""};
  int ans, k;
```



# 函数指针数组示例(续)

## ■ 通用的菜单处理函数(续)

```
while (1) {  
    printf("请选择以下菜单命令。 \n");  
    for(k = 0; menuName[k][0] != '\0' ; k++)  
        printf("\t%d: %s\n", k+1, menuName[k]);  
    printf("\t其它选择结束程序运行。 \n");  
    scanf("%d", &ans);  
    if (ans < 1 || ans > k) break;  
    printf("\n\n结果: ");  
    result = (*fpt[ans-1])(a, N);  
    printf("\t%s=%4.2f\n\n", title[ans-1], result);  
}  
}
```



# 提要

---

- 指针形参
- 数组形参
- 字符数组形参
- 函数指针
- **返回指针值的函数**
  - **返回数据对象指针的函数**
  - **返回函数指针的函数**

# 返回数据对象指针的函数

- 函数可以返回整型值、字符值、实型值等，也可以返回指向某种数据对象的指针值。返回指针值的函数与以前介绍的函数在概念上是完全一致的
- **定义(或说明)返回指针值函数的一般形式为**  
**类型说明符 \* 函数名(形参表);**
  - 例如，函数说明：**int \*f(int, int);**
  - 函数**f()**返回指向**int**型数据的指针，有两个整型形参
  - **注意：**在函数名两侧分别为\*运算符和()运算符，而()的优先级高于\*，函数名先与()结合。函数名()是函数的说明形式。在函数名之前的\*，表示函数返回指针类型的值

# 返回数据对象指针的函数示例-1

- 编制在给定的字符串中找特定字符的第一次出现。若找到，返回指向字符串中该字符的指针；否则，返回**NULL**值
- 程序设计分析
  - 设函数为**sear\_ch()**，该函数有两个形参，指向字符串首字符的指针和待寻找的字符
  - 函数**sear\_ch()** 的定义

```
char *sear_ch(char *s, char c)
{ while (*s && *s != c)
    s++;
  return *s ? s : NULL;
}
```

## 返回数据对象指针的函数示例-2

- 改写前面例子（学生成绩单）的函数 `search()`，使新的 `search()` 函数具有单一的寻找功能，不再包含输出等功能。新的函数 `search()` 的功能是从给定的成绩单中找指定学号的成绩表。新的函数 `search()` 的形参与前面例子中的函数 `search()` 的形参相同，但新的函数 `search()` 返回找到的那位学生的成绩表(不包括学号)的指针

- 函数 `search()` 定义如下

```
int *search (int (*p)[6], int m, int no)
{ int (*ap)[6];
  for(ap = p; ap < p+m; ap++)
    if(**ap == no) return *ap+1;
  return NULL;
}
```



# 返回函数指针的函数

- 返回函数指针值函数定义或说明的一般形式为  
类型说明符 **(\*函数名(形参表))(形参类型表):**
  - 按前面的例子，菜单函数接受用户选择，返回相应处理函数的指针：**double (\*menu(char \*\*))(double\*, int)**
  - 在**menu()**函数说明中，
    - 先是**menu(char \*\*)**，**menu**同圆括号结合，说明**menu**是函数；
    - 随后是**(\*menu(char \*\*))**，同**\***结合，函数返回指针；
    - 接着是**(\*menu(char \*\*))(double\*, int)**
    - 同一对圆括号结合，表示是指向函数的指针，所指函数有**double\***和**int**类型两个形参；
    - 最后是**double (\*menu(char \*\*))(double\*,int)**，说明指向的函数的返回值是**double**类型

# 返回函数指针的函数示例程序

```
#include <stdio.h>
#define N sizeof a / sizeof a[0]
double max(double *, int), min(double *, int), ave(double *, int);
double (*fpt[])(double *, int) = {max, min, ave, NULL}; /*函数指针数组*/
char *title[] = {"最大值", "最小值", "平均值", ""};
char *menuName[] = {"求最大值", "求最小值", "求平均值", ""};
double a[] = {1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0};
double (*menu(char **titptr))( double *, int) /*函数返回函数的指针*/
{ int ans, k; printf("请选择以下菜单命令。 \n");
  for(k=0; menuName[k][0] != '\0'; k++) printf("\t%d:%s\n", k+1, menuName[k]);
  printf("\t其它选择结束程序运行。 \n"); scanf("%d", &ans);
  if (ans < 1 || ans > 3) return NULL;
  *titptr = title[ans-1]; return fpt[ans-1]; /* 返回函数的指针 */
}
```



# 返回函数指针的函数示例程序(续)

```
void main ()
{ double (*fp)( double *, int); char *titstr; double result;
  while (1) {
    if ((fp = menu(&titstr)) == NULL) break;
    result = (*fp)(a, N);
    printf("\n结果: %s = %4.2f\n\n", titstr, result);
  }
}

double max(double a[], int n)
{ int i; double r;
  for(r = a[0], i = 1; i < n; i++) if (r < a[i]) r = a[i];
  return r;
}
```



# 返回函数指针的函数示例程序(续)

```
double min(double a[], int n)
{ int i; double r;
  for(r = a[0], i = 1; i < n; i++) if (r > a[i]) r = a[i];
  return r;
}
```

```
double ave(double a[], int n)
{ int i; double r;
  for (r = 0.0, i = 0; i < n; i++) r += a[i];
  return r/n;
}
```



# 第11讲 小结

---

- 指针形参
- 数组形参
- 字符指针形参
- 函数指针
  - 函数指针和函数指针变量、利用函数指针调用函数
  - 函数指针形参、函数指针数组
- 返回指针值的函数
  - 返回数据对象指针的函数
  - 返回函数指针的函数



# 第六章作业

---

- 习题六

- 第10、 11、 12、 16、 17、 19题