

第8章 善于利用指针

8.1 指针是什么

8.2 指针变量

8.3 通过指针引用数组

8.4 通过指针引用字符串

8.5 指向函数的指针

8.6 返回指针值的函数

8.7 指针数组和多重指针

8.8 动态内存管理

8.9 有关指针的小结

8.5 指向函数的指针

8.5.1 函数指针及其声明

8.5.2 用函数指针变量调用函数

8.5.3 怎样定义和使用指向函数的指针变量

8.5.4 用指向函数的指针作函数参数

8.5.1 函数指针及其声明

- 函数指针是指向函数的指针变量。因此“函数指针”本身首先应是指针变量，只不过该指针变量指向函数。这正如用指针变量可指向整型变量、字符型、数组一样，这里是指向函数。
- **C**在编译时，每一个函数都有一个入口地址，该入口地址就是函数指针所指向的地址。有了指向函数的指针变量后，就可以使用该指针变量调用函数，就如同用指针变量可引用其他类型变量一样，在这些概念上是大体一致的。
- 函数指针有两个用途
 - ◆ 调用函数
 - ◆ 做函数的参数。

8.5.1 函数指针及其声明

- 可以定义一个指向函数的指针变量，用来存放某一函数的起始地址，这就意味着此指针变量指向该函数。
- 函数指针的声明方法为：
 - ◆ 返回值类型 (***指针变量名**) (**[形参列表]**);
 - ◆ 返回值类型说明函数的返回类型，为某一数据类型说明符，(***指针变量名**) 中的括号不能省，括号改变了运算符的优先级。若省略整体则成为一个函数说明，说明了一个返回的数据类型是指针的函数，后面的**[形参列表]**表示指针变量指向的函数所带的参数列表。
 - 例如: **int (*p)(int, int);**
 - 定义**p**是指向函数的指针变量，它可以指向类型为整型且有两个整型参数的函数。
 - **p**的类型用**int (*)(int, int)**表示
 - 这里**p++**、**p--**、**p+i**无意义

8.5.2 用函数指针调用函数

例**8.16** 用函数求整数**a**和**b**中的大者(函数调用使用函数指针来实现)。

```
int max(int x, int y){  
    return x>y?x:y;  
}
```

(1)通过函数名来调用

```
int main(){  
    ...  
    c = max(a, b);  
    ...  
}
```

(2)通过函数指针来调用

```
int (*p)(int, int); //声明函数指针  
p = max; //函数名就是函数的入口地址  
c = p(a, b); //c = (*p)(a, b);亦可
```

例**8.17** 输入两个整数，然后让用户选择**1**或**2**，选**1**时调用**max**函数，输出二者中的大数，选**2**时调用**min**函数，输出二者中的小数。

- 解题思路：定义两个函数**max**和**min**，分别用来求大数和小数。在主函数中根据用户输入的数字**1**或**2**，使指针变量指向**max**函数或**min**函数。

```
inline int max(int x, int y);
inline int min(int x, int y);
int main( ){
    int a, b, c, select, (*p)(int, int); .....
    if(select == 1) p = max;
    else if(select == 2) p = min;
    c = p(a, b); // c = (*p)(a, b);
    ...
}
int max(int x, int y){return x>y?x:y;}
int min(int x, int y){return x<y?x:y;}
```

只看此行看不出调用哪函数

8.5.4 用指向函数的指针作函数参数

- 指向函数的指针变量的一个重要用途是把函数的地址作为参数传递到被调用函数
- 指向函数的指针可以作为函数参数，把函数的入口地址传递给形参，这样就能夠在被调用的函数中使用实参函数
- 函数指针(指向函数的指针变量)使得函数名可以作为函数调用的实参

例**8.18** 通过指向函数的指针变量，求两个整数的大者、小者、和、差、积等。

- 解题思路
 - ◆ 先定义实现上述基本功能的几个函数
 - ◆ 再定义一个参数为函数指针的函数，实现对指定函数的调用

8.5.4 用指向函数的指针作函数参数

```
int max(int x, int y){return x>y?x:y;}
int min(int x, int y){return x>y?x:y;}
int add(int x, int y){return x>y?x:y;}
int minus(int x, int y){return x>y?x:y;}
int multiply(int x, int y){return x>y?x:y;}
int process(int x, int y, int (*p)(int, int)){
    return p(x, y);
}
int main(){
    int a, b; .....
    c = process(a, b, max);
    c = process(a, b, add);
    .....
}
```


8.6 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。其概念与以前类似，只是返回的值的类型是指针类型而已。
- 定义返回指针值的函数的一般形式为：

返回值类型 *函数名(参数表列){

.....

return pointer;

}

- ◆ **返回值类型**说明函数返回的指针所引用的类型，为某一数据类型说明符
- ◆ 在函数体的**return**语句中，要返回一个指针类型的值，一般该值不能为局部变量的地址，往往使用参数中的某个指针变量作为返回值。

例**8.19**有**a**个学生，每个学生有**4**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。用返回指针值的函数实现。

➤ 分析：

◆ 定义一个二维数组来存储成绩 **float score[a][4]**

◆ 定义一个函数查找指定学生序号的成绩，因为一个学生包含**4**门课程的成绩（相当于一个长度为**4**的一维数组），所以该函数返回一个指针值，一维数组的首地址。

◆ 函数的形参

● 学生的成绩

● 学生的序号

```
float *find(float (*p)[4], int i){
```

```
    return *(p+i);
```

```
}
```

```

#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 i);
  float *p; int i,k;
  scanf("%d",&k);
  printf("The scores of No.%d are:\n",k);
  p=find(score,k);      返回k号学生课程首地址
  for(i=0;i<4;i++)
    printf("%5.1f",*(p+i));
  printf("\n");
  return 0;
}

```

修改上题为：找出第一个有不及格课程的学生，函数如何修改呢？

注意：不排除没有不及格课程的学生

8.7 指针数组和多重指针

8.7.1 指针数组

8.7.2 指向指针数据的指针

8.7.3 指针数组作main函数的形参

8.7.1 指针数组

- 数组是具有同类型元素数据的有序集合，若其元素均为指针类型数据，则称为指针数组，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，都用来表示一个地址。
- 定义一维指针数组的一般形式为
数据类型说明符 *数组名[非负整型表达式];
int *p[4];
 - ◆ 指针数组比较适合用来指向若干个字符串，使字符串处理更加方便灵活
 - ◆ 可以分别定义一些字符串，然后用指针数组中的元素分别指向各字符串
 - ◆ 由于各字符串长度一般是不相等的，所以比用二维数组节省内存单元

8.7.1 指针数组

例**8.20** 使用指针数组将若干字符串按字母顺序(由小到大)输出。

- 解题思路：定义一个指针数组，用各待排序字符串对它进行初始化，然后用冒泡法、选择法或插入法进行排序。

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

此处用**strcpy**函数可以吗？

- 本函数不移动字符串，而是改变指针数组各元素所指向的字符串。

8.7.2 指向指针数据的指针

- 指针是一种派生数据类型，指针变量存储的是它所指向的变量的地址。在程序运行期间，会为所有的变量分配存储空间，同样指针变量也占用固定的内存。这样就出现了指向指针变量的指针，称为多重指针。
- 二重指针
 - ◆ 数据类型说明符 ****** 指针变量名;
 - ◆ 如: **int **p;**
 - ◆ **p**是指针变量，用来指向**int***类型的变量
 - ◆ **int **pp;**
 - ◆ **int *p, a;**
 - ◆ **pp = &p;**
 - ◆ **p = &a;**

➤ **const** 约束的指针

◆ **const int *p;** //表示***p**是只读的，亦即***p**的值不能改变，即不能通过指针修改它所指向对象的值。

- **int a = 5, b;**

- **const int *p = &a;**

- ***p = 8;** //错误

- **p = &b;** //正确

◆ **int * const p = &a;** //表示指针变量**p**是只读的，一般在定义时应为其初始化

- **int * const p;** //编译器未必报错，但**p**一直指向同一地址单元，其值无法修改

- **p = &a;** //错误

◆ **const int **pp;**

◆ **int a, *p = &a;**

◆ **pp = &p;** //错误，违反约束

8.7.3 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 **int main(void)**

- 表示**main**函数没有参数，调用**main**函数时不必给出实参。
- 这是一般程序常采用的形式。
- 实际上，在某些情况下，**main**函数可以有参数，如：

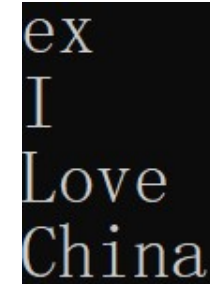
int main(int argc, char *argv[])

- ◆ 其中，**argc**和**argv**就是**main**函数的形参，它们是程序的“命令行参数”。
- **argv**是**char**指针数组，数组中每一个元素(其值为指针)指向命令行中的一个字符串。

8.7.3 指针数组作main函数的形参

- 一个程序，通常由**main**函数和其他函数组成，可以包含一个或多个文件。
- 对这些文件进行编译和连接，可以得到可执行文件(后缀为**.exe**，**非Windows操作系统不同**)。用户执行这个可执行文件，操作系统就调用**main**函数，然后由**main**函数调用其他函数，从而完成程序的功能。
- **main**函数的参数是怎么传递过来的呢？
 - ◆ 在程序中，**main**函数不会被其他函数调用，显然参数的值不可能在程序中得到。
 - ◆ **main**函数是操作系统调用的，实参只能由操作系统给出。通过命令行窗口运行程序来实现。

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    for(int i=0; i<argc; i++)
        printf("%s\n", *(argv++));
    return 0;
}
```



ex
I
Love
China

这里**argc**，表示通过命令行所输入的字符串的总个数(以空格分隔)

argv是字符型指针数组，用来指向命令行中的参数。
。 **argv[0]** 指向可执行程序文件名，
argv[1],...依次指向文件名后依次出现的参数。

若上例可执行文件名为**ex**，则在命令行窗口输入：

ex I Love China<回车>