

第15讲：深入理解指针(5)

目录

1. 回调函数是什么？
2. qsort使用举例
3. qsort函数的模拟实现

正文开始

1. 回调函数是什么？

回调函数就是一个通过函数指针调用的函数。

如果你把函数的指针（地址）作为参数传递给另一个函数，当这个指针被用来调用其所指向的函数时，被调用的函数就是回调函数。回调函数不是由该函数的实现方直接调用，而是在特定的事件或条件发生时由另外的一方调用的，用于对该事件或条件进行响应。

第13讲中我们写的计算机的实现的代码中，红色框中的代码是重复出现的，其中虽然执行计算的逻辑是区别的，但是输入输出操作是冗余的，有没有办法，简化一些呢？

因为红色框中的代码，只有调用函数的逻辑是有差异的，我们可以把调用的函数的地址以参数的形式传递过去，使用函数指针接收，函数指针指向什么函数就调用什么函数，这里其实使用的就是回调函数的功能。

```
1 //使用回调函数改造前
2 #include <stdio.h>
3 int add(int a, int b)
4 {
5     return a + b;
6 }
7 int sub(int a, int b)
8 {
9     return a - b;
10 }
11 int mul(int a, int b)
12 {
13     return a * b;
```

```
1 //使用回调函数改造后
2 #include <stdio.h>
3 int add(int a, int b)
4 {
5     return a + b;
6 }
7 int sub(int a, int b)
8 {
9     return a - b;
10 }
11 int mul(int a, int b)
12 {
13     return a * b;
```

```

14 }
15 int div(int a, int b)
16 {
17     return a / b;
18 }
19 int main()
20 {
21     int x, y;
22     int input = 1;
23     int ret = 0;
24     do
25     {
26
27         printf("*****
28         *\n");
29
30         printf(" 1:add
31         2:sub  \n");
32         printf(" 3:mul
33         4:div  \n");
34
35         printf("*****
36         *\n");
37
38         printf("请选择: ");
39         scanf("%d", &input);
40         switch (input)
41         {
42             case 1:
43                 printf("输入操作
44                 数: ");
45                 scanf("%d %d", &x,
46                 &y);
47                 ret = add(x, y);
48                 printf("ret =
49                 %d\n", ret);
50                 break;
51             case 2:
52                 printf("输入操作
53                 数: ");
54                 scanf("%d %d", &x,
55                 &y);
56                 ret = sub(x, y);
57                 printf("ret =
58                 %d\n", ret);
59                 break;
60             case 3:
61                 printf("输入操作
62                 数: ");

```

```

14 }
15 int div(int a, int b)
16 {
17     return a / b;
18 }
19 void calc(int(*pf)(int, int))
20 {
21     int ret = 0;
22     int x, y;
23     printf("输入操作数: ");
24     scanf("%d %d", &x, &y);
25     ret = pf(x, y);
26     printf("ret = %d\n", ret);
27 }
28 int main()
29 {
30     int input = 1;
31     do
32     {
33
34         printf("*****
35         *\n");
36
37         printf(" 1:add
38         2:sub  \n");
39         printf(" 3:mul
40         4:div  \n");
41
42         printf("*****
43         *\n");
44
45         printf("请选择: ");
46         scanf("%d", &input);
47         switch (input)
48         {
49             case 1:
50                 calc(add);
51                 break;
52             case 2:
53                 calc(sub);
54                 break;
55             case 3:
56                 calc(mul);
57                 break;
58             case 4:
59                 calc(div);
60                 break;
61             case 0:
62                 printf("退出程序\n");

```

```

48         scanf("%d %d", &x,
    &y);
49         ret = mul(x, y);
50         printf("ret =
    %d\n", ret);
51         break;
52     case 4:
53         printf("输入操作
    数: ");
54         scanf("%d %d", &x,
    &y);
55         ret = div(x, y);
56         printf("ret =
    %d\n", ret);
57         break;
58     case 0:
59         printf("退出程序\n");
60         break;
61     default:
62         printf("选择错误\n");
63         break;
64     }
65     } while (input);
66
67     return 0;
68 }

```

```

55         break;
56     default:
57         printf("选择错误\n");
58         break;
59     }
60     } while (input);
61
62     return 0;
63 }

```

2. qsort 使用举例

2.1 使用qsort函数排序整型数据

```

1  #include <stdio.h>
2
3  //qsort函数的使用者得实现一个比较函数
4  int int_cmp(const void * p1, const void * p2)
5  {
6      return (*(int *)p1 - *(int *) p2);
7  }
8
9  int main()
10 {
11     int arr[] = { 1, 3, 5, 7, 9, 2, 4, 6, 8, 0 };
12     int i = 0;

```

```

13
14     qsort(arr, sizeof(arr) / sizeof(arr[0]), sizeof (int), int_cmp);
15     for (i = 0; i < sizeof(arr) / sizeof(arr[0]); i++)
16     {
17         printf( "%d ", arr[i]);
18     }
19     printf("\n");
20     return 0;
21 }

```

2.2 使用qsort排序结构数据

这里需要补充介绍结构指针和结构体成员访问操作符 `->` ；

```

1  struct Stu //学生
2  {
3      char name[20]; //名字
4      int age; //年龄
5  };
6
7  //假设按照年龄来比较
8  int cmp_stu_by_age(const void* e1, const void* e2)
9  {
10     return ((struct Stu*)e1)->age - ((struct Stu*)e2)->age;
11 }
12
13 //strcmp - 是库函数，是专门用来比较两个字符串的大小的
14 //假设按照名字来比较
15 int cmp_stu_by_name(const void* e1, const void* e2)
16 {
17     return strcmp(((struct Stu*)e1)->name, ((struct Stu*)e2)->name);
18 }
19
20 //按照年龄来排序
21 void test2()
22 {
23     struct Stu s[] = { {"zhangsang", 20}, {"lisi", 30}, {"wangwu", 15} };
24     int sz = sizeof(s) / sizeof(s[0]);
25     qsort(s, sz, sizeof(s[0]), cmp_stu_by_age);
26 }
27
28 //按照名字来排序
29 void test3()
30 {
31     struct Stu s[] = { {"zhangsang", 20}, {"lisi", 30}, {"wangwu", 15} };
32     int sz = sizeof(s) / sizeof(s[0]);

```

```

33     qsort(s, sz, sizeof(s[0]), cmp_stu_by_name);
34 }
35
36 int main()
37 {
38     test2();
39     test3();
40     return 0;
41 }

```

3. qsort函数的模拟实现

使用回调函数，模拟实现qsort（采用冒泡的方式）。

注意：这里第一次使用 `void*` 的指针，讲解 `void*` 的作用。

```

1  #include <stdio.h>
2
3  int int_cmp(const void * p1, const void * p2)
4  {
5      return (*(int *)p1 - *(int *) p2);
6  }
7
8  void _swap(void *p1, void * p2, int size)
9  {
10     int i = 0;
11     for (i = 0; i < size; i++)
12     {
13         char tmp = *((char *)p1 + i);
14         *((char *)p1 + i) = *((char *) p2 + i);
15         *((char *)p2 + i) = tmp;
16     }
17 }
18
19 void bubble(void *base, int count , int size, int(*cmp )(void *, void *))
20 {
21     int i = 0;
22     int j = 0;
23     for (i = 0; i < count - 1; i++)
24     {
25         for (j = 0; j < count - i - 1; j++)
26         {
27             if (cmp ((char *) base + j*size , (char *)base + (j + 1)*size) >
28                 0)
29                 {

```

```
29         _swap((char *)base + j*size, (char *)base + (j + 1)*size,
30             size);
31     }
32 }
33 }
34 int main()
35 {
36     int arr[] = { 1, 3, 5, 7, 9, 2, 4, 6, 8, 0 };
37     int i = 0;
38     bubble(arr, sizeof(arr) / sizeof(arr[0]), sizeof(int), int_cmp);
39     for (i = 0; i < sizeof(arr) / sizeof(arr[0]); i++)
40     {
41         printf( "%d ", arr[i]);
42     }
43     printf("\n");
44     return 0;
45 }
```

完