# CSD STDC02 DAY04

Top


1. 局部变量和全局变量
2. 指针的基本用法
3. 指针的优势
4. 指针的优势(续 1)
5. 指针的算术运算


## 1 局部变量和全局变量

### 1.1 问题

分别定义一个局部变量和全局变量，测试生命期限和作用域。

### 1.2 步骤

实现此案例需要按照如下步骤进行。

**步骤一：局部变量和全局变量**

代码如下所示：

```
1.  #include <stdio.h>
2.
3.  /*auto*/ int x = 1000;
4.
5.  void f1(int a)
6.  {
7.      a = 10;
8.      auto int x = 20;
9.      static int y = 30;
10.     y++;
11.     x++;
12.     printf("in f1(),a=%d,x=%d,y=%d\n",a,x,y);
13. }
14.
15. void f2()
16. {
```

```
17.     printf("in f2(),x=%d\n",x);
18.     x = 2000;
19. }
20.
21. void f3()
22. {
23.     printf("in f3(),x=%d\n",x);
24. }
25.
26. int main()
27. {
28.     int x = 30,b=100;
29.     f1(b);
30.     f1(b);
31.     printf("in main(),b=%d,x=%d\n",b,x);
32.     f2();
33.     f3();
34.
35.     return 0;
36. }
```

上述代码中，以下代码：

```
1.  /*auto*/ int x = 1000;
```

定义了一个全局变量 x，该变量在所有函数中共享。
上述代码中，以下代码：

```
1.  void f1(int a)
2.  {
3.      a = 10;
4.      auto int x = 20;
5.      static int y = 30;
6.      y++;
```

```
7.      x++;
8.      printf("in f1(),a=%d,x=%d,y=%d\n",a,x,y);
9.  }
```

定义了一个函数 f1，该函数没有返回值，有一个参数是一个整型变量 a。该函数中以下语句：

```
1.      a = 10;
```

将参数 a 赋值为 10，参数 a 是局部变量，只能用在 f1 函数中。该函数中以下语句：

```
1.      auto int x = 20;
```

定义了另一个局部变量 x，也只能用在 f1 函数中。前面的 auto 关键字可有可无。该函数中以下语句：

```
1.      static int y = 30;
```

定义了一个 static 局部变量 y，也只能用在 f1 函数中，但函数返回后该变量不会被释放，直到主函数退出时，它才会被释放。该函数中以下语句：

```
1.      y++;
2.      x++;
```

对局部变量 x 和 y 进行自增操作。

***注意：在函数 f1 中，定义了局部变量 x，而在该函数的上面还定义全局变量 x，这样在函数 f1 中，有两个 x，但 C 语言规定，在这种情况下，局部变量 x 将隐藏全局变量 x，即全局变量 x 此时不起作用。***

该函数中以下语句：

```
1.      printf("in f1(),a=%d,x=%d,y=%d\n",a,x,y);
```

使用函数 printf 输出局部变量 a 和 x 的值，以及静态局部变量 y 的值。

上述代码中，以下代码：

```
1.  void f2()
2.  {
3.      printf("in f2(),x=%d\n",x);
4.      x = 2000;
5.  }
```

定义了一个无参无返回值的函数 f2。该函数中以下语句：

```
1.      x = 2000;
```

对全局变量 x 进行赋值。

上述代码中，以下代码：

```
1.  void f3()
2.  {
3.      printf("in f3(),x=%d\n",x);
4.  }
```

定义了一个无参无返回值的函数 f2。该函数中以下语句：

```
1.      printf("in f3(),x=%d\n",x);
```

对全局变量 x 进行输出。

上述代码中，以下代码：

```
1.  int main()
2.  {
3.      int x = 30,b=100;
```

在主函数中，定义了两个局部变量 x 和 b，并分别初始化为 30 和 100。这两个局部变量只能用在 main 函数中。

上述代码中，以下代码：

```
1.      f1(b);
2.      f1(b);
```

两次调用函数 f1，并用局部变量 b 作为实参。

***注意：两次调用中，函数 f1 中的静态局部变量 y 输出值不一样。因为 static 关键字使该变量在函数 f1 返回后并不释放，所以第二次调用时，其值会在上一次调用返回前的基础上发生变化。***

上述代码中，以下代码：

```
1.      printf("in main(),b=%d,x=%d\n",b,x);
```

使用函数 printf 输出局部变量 b 和局部变量 x 的值。

***注意：在函数 main 中，定义了局部变量 x，而在该函数的上面还定义全局变量 x，这样在函数 main 中，有两个 x，但 C 语言规定，在这种情况下，局部变量 x 将隐藏全局变量 x，即全局变量 x 此时不起作用。***

上述代码中，以下代码：

```
1.      f2();
2.      f3();
```

调用函数 f2 和 f3，从输出的值上可以看出，全局变量可以在所有函数中共享。

### 1.3 完整代码

本案例的完整代码如下所示：

```
1.  #include <stdio.h>
2.
3.  /*auto*/ int x = 1000;
4.
5.  void f1(int a)
6.  {
7.      a = 10;
8.      auto int x = 20;
9.      static int y = 30;
10.     y++;
11.     x++;
12.     printf("in f1(),a=%d,x=%d,y=%d\n",a,x,y);
13. }
14.
15. void f2()
16. {
17.     printf("in f2(),x=%d\n",x);
18.     x = 2000;
19. }
20.
21. void f3()
22. {
```

```
23.     printf("in f3(),x=%d\n",x);
24. }
25.
26. int main()
27. {
28.     int x = 30,b=100;
29.     f1(b);
30.     f1(b);
31.     printf("in main(),b=%d,x=%d\n",b,x);
32.     f2();
33.     f3();
34.
35.     return 0;
36. }
```

# 2 指针的基本用法

## 2.1 问题

定义一个 int 类型的指针，指向一个整型变量，然后分别使用&和*取得地址或者数据。

## 2.2 步骤

实现此案例需要按照如下步骤进行。

**步骤一：指针的基本用法**

代码如下所示：

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.  int a = 10;
6.  int *p;
7.      p = &a;
8.      printf("p = %p\n", p);
```

```
9.      printf("&a = %p\n", &a);
10.
11.     printf("a = %d\n", a);
12.     printf("*p = %d\n", *p);
13.
14. return0;
15. }
```

上述代码中，以下代码：

```
1.  int a = 10;
```

定义了一个整型变量 a，并初始化为 10。

上述代码中，以下代码：

```
1.  int *p;
2.      p = &a;
```

定义了一个整型指针变量 p，并使用取地址运算符&将整型变量 a 的地址赋值给 p。

上述代码中，以下代码：

```
1.      printf("p = %p\n", p);
2.      printf("&a = %p\n", &a);
```

使用函数 printf 输出指针变量 p 的值和整型变量 a 的地址值。从输出结果可以看出，指针变量 p 中存放的就是整型变量 a 的地址值。

上述代码中，以下代码：

```
1.      printf("a = %d\n", a);
```

使用函数 printf 输出整型变量 a 的值。

上述代码中，以下代码：

```
1.      printf("*p = %d\n", *p);
```

使用函数 printf 输出指针变量 p 所指向的地址单元的值。由于指针变量 p 被赋值成了整型变量 a 的地址值，所以，p 就指向 a 的地址，所以输出的就是 a 的值。

### 2.3 完整代码

本案例的完整代码如下所示：

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.  int a = 10;
6.  int *p;
7.      p = &a;
8.      printf("p = %p\n", p);
9.      printf("&a = %p\n", &a);
10.
11.     printf("a = %d\n", a);
12.     printf("*p = %d\n", *p);
13.
14. return0;
15. }
```

# 3 指针的优势

## 3.1 问题

使用指针实现交换变量值的函数 swap()，同时编写不使用指针交换失败的函数做对比。

## 3.2 步骤

实现此案例需要按照如下步骤进行。

**步骤一：指针的优势**

代码如下所示：

```
1.  #include <stdio.h>
2.
3.  void swap(int a, int b)
4.  {
5.      printf("in swap()  a = %d, b = %d\n", a, b);
6.  int tmp = a;
7.      a = b;
```

```
8.      b = tmp;
9.      printf("in swap()  a = %d, b = %d\n", a, b);
10. }
11.
12. void swap1(int *a, int *b)
13. {
14.     printf("in swap1()  *a = %d, *b = %d\n", *a, *b);
15. int tmp = *a;
16.     *a = *b;
17.     *b = tmp;
18.     printf("in swap1()  *a = %d, *b = %d\n", *a, *b);
19. }
20.
21. int main()
22. {
23. int x = 10, y = 20;
24.     printf("in main()  x = %d, y = %d\n", x, y);
25.     swap(x, y);
26.     printf("in main()  x = %d, y = %d\n", x, y);
27.     swap1(&x, &y);
28.     printf("in main()  x = %d, y = %d\n", x, y);
29.
30. return0;
31. }
```

上述代码中，以下代码：

```
1.  void swap(int a, int b)
2.  {
3.      printf("in swap()  a = %d, b = %d\n", a, b);
4.  int tmp = a;
5.      a = b;
6.      b = tmp;
```

```
7.      printf("in swap()  a = %d, b = %d\n", a, b);
8.  }
```

定义了一个函数 swap，该函数有两个参数，为整型变量 a 和 b。在该函数中，以下语句：

```
1.      printf("in swap()  a = %d, b = %d\n", a, b);
```

使用函数 printf 输出形参 a 和 b 从调用函数传入的值。在该函数中，以下语句：

```
1.  int tmp = a;
2.      a = b;
3.      b = tmp;
```

通过第三个变量 tmp，将整型变量 a 和 b 的值进行交换。在该函数中，以下语句：

```
1.      printf("in swap()  a = %d, b = %d\n", a, b);
```

使用函数 printf 输出形参 a 和 b 交换后的值。

上述代码中，以下代码：

```
1.  void swap1(int *a, int *b)
2.  {
3.      printf("in swap1()  *a = %d, *b = %d\n", *a, *b);
4.  int tmp = *a;
5.      *a = *b;
6.      *b = tmp;
7.      printf("in swap1()  *a = %d, *b = %d\n", *a, *b);
8.  }
```

定义了一个函数 swap1，该函数有两个参数，为整型指针变量 a 和 b。在该函数中，以下语句：

```
1.      printf("in swap1()  *a = %d, *b = %d\n", *a, *b);
```

使用函数 printf 输出整型指针变量 a 和 b 指向的地址的值。在该函数中，以下语句：

```
1.  int tmp = *a;
```

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：https://d.book118.com/856132135001010220