

暨南大学本科实验报告专用纸

课程名称 高级语言程序设计实验 成绩评定
实验项目名称 指针验证实验 指导教师 张鑫源
实验项目编号 ⑪ 实验项目类型 实验地点 家中
学生姓名 位雨昕 学号 2019051112
学院 智能科学与工程学院 系 专业 信息安全
实验时间 2020 年 5 月 20 日 上午 ~ 5 月 20 日 下午 温度 °C 湿度

（一）实验目的

1. 进一步了解 Visual Studio 的使用以及 C 语言程序的结构；
2. 掌握 C 语言中的常用函数的使用方法；
3. 了解值传递和引用传递两种参数传递方式两种方式的机制；
4. 锻炼个人的编程操作能力。

（二）实验内容和要求

内容：

设计实验探讨“值传递”和“引用传递”两种参数传递方式的差别，并结合实验结果在实验报告中对两种方式的机制做出说明。

示例：设计函数 f1 和 f2，分别采用值传递和引用传递两种方式实现实参与形参之间的数据传递，f1 和 f2 的功能均是实现形参数值的互换。

要求：

1. 设计合理的输出展示实验结果；
2. f1 和 f2 可以放在独立的源文件中。

（三）主要仪器设备

仪器：计算机

实验环境：Visual Studio Community 2019

（四）源程序

主程序.c:

```
#include<stdio.h>
extern void fun1(int x, int y);
extern void fun2(int *x, int *y);
extern void fun3(int x[1], int y[1]);

int main()
{
    int a = 1;
    int b = 2;
    printf("[值传递]\n");
    fun1(a, b);
    printf("a=%d, b=%d\n", a, b);
    printf("\n");

    printf("[采用指针]\n");
    fun2(&a, &b);
    printf("a=%d, b=%d\n", a, b);
    printf("\n");

    printf("[采用数组]\n");
    int A[1], B[1];
    A[0] = 1;
    B[0] = 2;
    fun3(A, B);
    printf("a=%d, b=%d\n", a, b);
    printf("\n");

    return 0;
}
```

值传递.c:

```
#include<stdio.h>
void fun1(int x, int y)
{

    int temp;
    temp = x;
    x = y;
```

```
y = temp;
printf("x=%d, y=%d\n", x, y);
}
```

引用传递-指针.c:

```
#include<stdio.h>
void fun2(int *x, int *y)
{

    int temp= *x;
    *x = *y;
    *y = temp;
    printf("x=%d, y=%d\n", *x, *y);
}
```

引用传递-数组.c:

```
#include<stdio.h>
void fun3(int x[1], int y[1])
{

    int temp;
    temp = x[0];
    x[0] = y[0];
    y[0] = temp;
    printf("x=%d, y=%d\n", x[0], y[0]);
}
```

（五）实验步骤与调试

实验步骤：

1. 对实验进行基本构思；
2. 启动 Visual Studio，创建新项目。将源程序写在新项目中；
3. 利用“本地 Windows 调试器”进行调试；
4. 进行多次调试，得出理想结果。

实验思路：

1. 新建 4 个源文件，一个用于写主程序，其余 3 个用于定义实现数值对调功能的 3 个函数；
2. 通过值传递实现实参与形参之间的数据传递的函数 fun1 在定义时形参使用(int x, int y)的形式；
3. 通过指针传递实现实参与形参之间的数据传递的函数 fun2 在定义时形参使用(int *x, int *y)的形式；
4. 通过数组传递实现实参与形参之间的数据传递的函数 fun3 在定义时形参使用(int x[1], int y[1])的形式；
5. 在主程序中调用外部函数 fun1、fun2、fun3，并用 extern 进行声明；
6. 在 fun1、fun2、fun3 中添加打印 x、y 值的语句，同时调用函数后增添打印 a、b 值的语句，可以更好地反映参数的传递的情况。

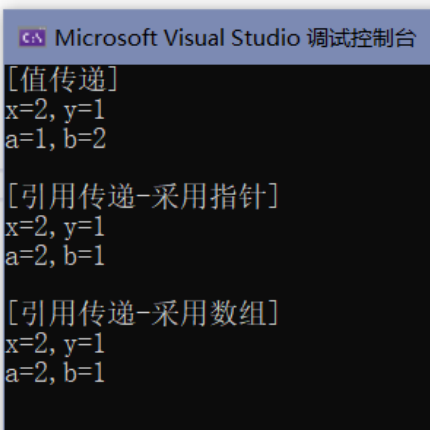
（六）实验结果与分析

实验结果：

```
= 1;
= 2;
["值传递\n"];
a, b);
("a=%d,b=%d\n", a, b);
("\n");

["引用传递-采用指针\n"];
&a, &b);
("a=%d,b=%d\n", a, b);
("\n");

["引用传递-采用数组\n"];
```



```
Microsoft Visual Studio 调试控制台
[值传递]
x=2, y=1
a=1, b=2

[引用传递-采用指针]
x=2, y=1
a=2, b=1

[引用传递-采用数组]
x=2, y=1
a=2, b=1
```

可以看到，通过值传递方式实现实参与形参之间数据传递的 fun1 并没有实现将 a、b 的值对调的功能。而 fun1 中的变量 x、y 的值却发生了对调。

采用指针的 fun2 中 x、y 的值发生了对调，且调用 fun2 也实现了将 a、b 的值对调的功能。

采用数组的 fun3 中 x、y 的值发生了对调，且调用 fun3 也实现了将 a、b 的值对调的功能。

分析：

1. 在调用 fun1 函数时，将实参 a、b 的值传入形参 x、y。在传入 fun1 后其实隐含了

```
int x = a;  
int y = b;
```

这两行语句的操作（进行值传递）。在函数 fun1 内部操作的其实是定义 fun1 时定义的变量 x、y，而不是 main 函数中的变量 a、b。此后 fun1 内部没有任何对 main 函数中变量 a、b 进行任何操作。因此采用值传递的方式并没有实现对调 a、b 的值的功能；

[值传递]
x=2, y=1
a=1, b=2

如图，只有 x、y 实现了对调。

2. 在调用 fun2 函数时，将实参 a、b 的地址传入形参 *x、*y。在传入 fun2 后其实隐含了

```
int *x = &a;  
int *y = &b;
```

这两行语句的操作。这两句定义了指针 x、y，使 x、y

分别指向 a、b 的地址。

```
int temp = *x;  
*x = *y;  
*y = temp;
```

定义变量 temp，用以交换指针 x、y 的指向。此后 x 指向 b，y 指向 a。从而实现 a、b 地址的交换。在函数 fun2 中访问地址，才能真正交换两个数字的值。

[采用指针]
x=2, y=1
a=2, b=1

如图，x、y 及 a、b 均实现了对调。

3. 数组名表示的是数组的首地址，即 A[0]的地址。定义一维数组 A、B，并将未对调时 a、b 的值分别赋给 A[1]、B[1]。将 A、B 传入 fun3 中定义的一维数组 x[1]、y[1]。

```
int temp;  
temp = x[0];  
x[0] = y[0];  
y[0] = temp;
```

定义变量 temp，用以交换 x、y 的地址。

因为数组名表示的是数组的首地址，所以实际操作的是数组 A、B 地

址。访问地址,才能真正交换两个数字的值。通过操作地址实现了 A[1] 与 B[1]值的交换。

```
[采用数组]
x=2, y=1
a=2, b=1
```

如图, x、y 及 A[1]与 B[1]均实现了对调。

4. 综上, 值传递和引用传递两种参数传递方式的差别是: 值传递是将原变量的值传递给另一变量, 重新分配存储空间; 而引用传递是直接访问变量的地址并直接对地址进行操作, 不用分配自己的内存空间。值传递操作的是不同的变量, 而引用传递操作的是同一变量。