函数

概念

> 在C语言中一般分为**两种函数**:
>
> 第一种是 **库函数** --> **标准库中提供的函数**
>
> 第二种是 **自定义函数** --> **自己设计和实现的函数**

```
//简单自定义加法求和
#define  _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int sum(int x,int y)                    //自定义sum函数
{
    int k = x + y;                      //将相加的范围值赋值给k，k返回给主函数
    return k;
}

int main() {
    int a = 0, b = 0;
    scanf("%d %d", &a, &b);             //输入
    int s = sum(a, b);                  //调用sum函数，返回的类型要和自定义函数定义的类型要一
致，这里是int，上面也要int。
    printf("%d", s);                    //输出
    return 0;
}
```

形参 与 实参

> 实际参数（实参）就是真实传递**给函数**的参数
>
> 形式参数（形参）只是**形式上**存在，如果不去调用的话，是不会申请内存空间的，只有函数被调用的时候为了存放被调用过来的值，才会申请内存空间。叫做形参的实例化
>
> 形参和实参是不同的内存空间
>
> 对形参的修改不会影响实参

```
//简单自定义加法求和
#define  _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

//函数的定义
int sum(int x,int y)                    //这里的x和y是形参（形式参数）
{
    int k = x + y;
    return k;
}

int main() {
    int a = 0, b = 0;
    scanf("%d %d", &a, &b);
```

```
    int s = sum(a, b);                  //a和b真实传递给sum的参数叫做实参（实际参数）
    printf("%d", s);
    return 0;
}
```

---

return 语句

```
//写一个程序，如果是偶数就返回1 如果是奇数就返回0

#define  _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int even(int x) {
    if (x % 2 == 0) {
        return 1;           //返回1
    }
    else {
        return 0;           //返回0
    }
}
int main() {
    int a = 0;
    scanf("%d",&a);
    int ret = even(a);      //even 偶数
    if (ret == 1)           //根据返回值判断奇偶性
        printf("是偶数");
    else
        printf("是奇数");
    return 0;
}
```

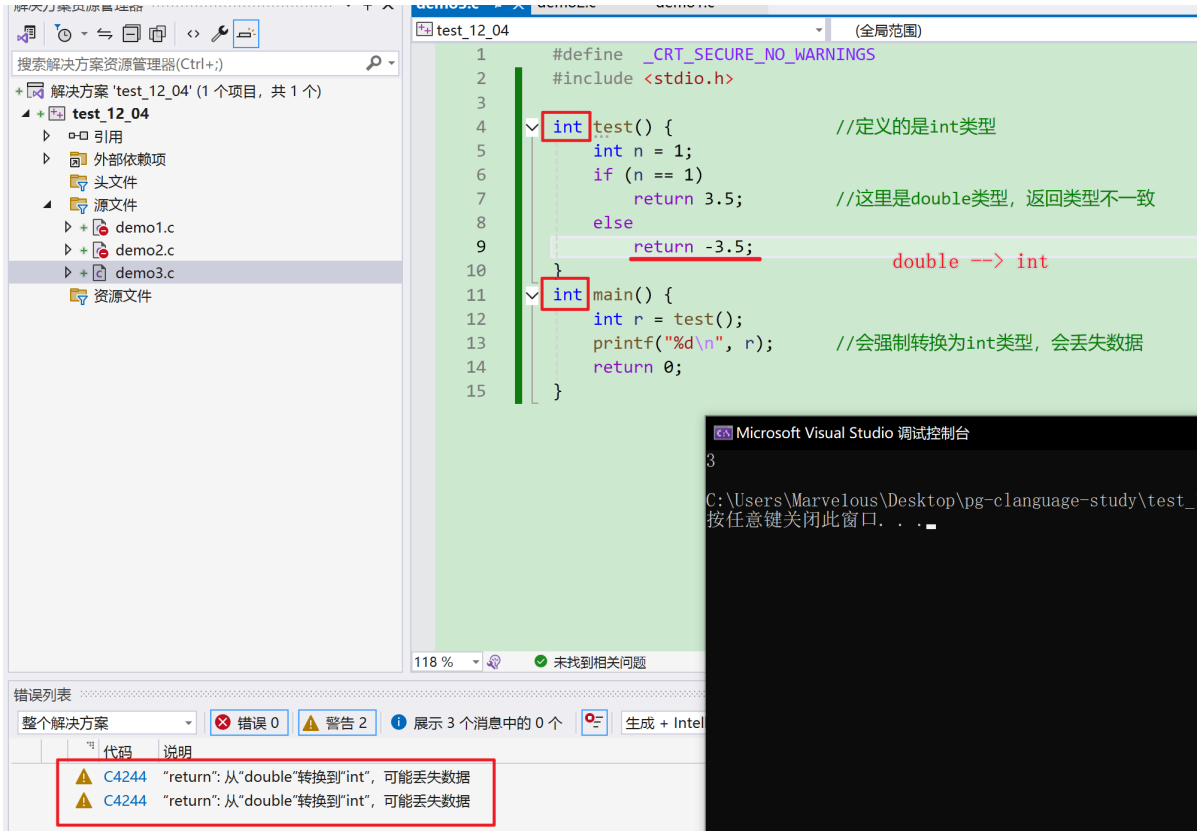
return语句的隐式转换

> return返回的值和函数返回**类型不一致**，系统会自动将返回的值**隐式转换**为函数的返回类型。

```
#define  _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int test() {                //定义的是int类型
    int n = 1;
    if (n == 1)
        return 3.5;         //这里是double类型，返回类型不一致
    else
        return -3.5;
}
int main() {
    int r = test();
    printf("%d\n", r);      //会强制转换为int类型，会丢失数据
    return 0;
}

//输出结果为:
3
```



自定义函数返回随机值的三种情况

> 1. 如果函数中存在if等分支的语句，则要保证**每种情况下都有return返回**，否则会出现编译错误
> 2. 函数的**返回类型如果不写**，编译器会默认函数的**返回类型是int**
> 3. 函数写了返回类型，但是函数中没有使用**return返回值**，那么函数的**返回值是未知的**

```
//第一种情况
#define  _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int test() {
    int n = 0;
    if (n == 1)                 //n != 1,这种情况，没有return返回值，数值随机，存在逻辑问题。
        return 3.5;

}
int main() {
    int r = test();
    printf("%d\n", r);
    return 0;
}

//输出结果：
1                               //如果加上一些其他语句 返回值还会改变
```

```
//第二种情况 和 第三种情况

#define  _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

test() {                        //未定义类型，默认为int
```

```
    printf("test\n");       //没有return，返回默认值
}
int main() {
    int r = test();
    printf("%d\n", r);
    return 0;
}

//输出结果：
5
```

数组做函数参数

一维数组传参

```
//通过自定义函数，分别将数组的内容设置为-1，并且通过自定义函数打印。

#define  _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
void set_arr(int arr[], int len) {
    for (int i = 0; i < len; i++) {
        arr[i] = -1;
    }
}

void
print_arr(int arr[], int len) {
    for (int i = 0; i < len; i++) {
        printf("%d ", arr[i]);
    }
}
int main() {
    int arr[] = { 1,2,3,4,5 };
    int len = sizeof(arr) / sizeof(arr[0]);
    set_arr(arr, len);
    print_arr(arr,len);
    return 0;
}
```

```
//通过自定义函数，打印二维数组
#include <stdio.h>
void printf_a(int a[3][2], int h, int l) {          //可以忽略行，但是不能忽略列 int
a[][2]
    int i = 0;
    for (i = 0; i < 3; i++) {
        int j = 0;
        for ( j = 0; j < 2; j++)
        {
            printf("%d ", a[i][j]);
        }
        printf("\n");
    }
}
```

```c
int main() {
    int a[3][2] = { 1,2,3,4,5,6 };
    printf_a(a,3,2);
    return 0;
}
```

嵌套调用 和 链式访问

嵌套调用

> 输入年数和月份，使用自定义函数判断该月份有多少天

```c
#define  _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int get_runnian_of_year(int year) {

    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))
        return 1;
    else
        return 0;
}
int get_days_of_month(int year, int month) {            //太过冗余，可以优化
    int days = 0;
    int rn;
    switch (month) {
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12:
        days = 31;
    case 4:
    case 6:
    case 9:
    case 11:
        days = 30;
    case 2:
        rn = get_runnian_of_year(year);
        if (rn == 1)
            days = 29;
        else
            days = 28;
    }
    return days;
}
int main() {
    int year = 0;
    int month = 0;
    scanf("%d %d", &year, &month);
    int days = get_days_of_month(year,month);
```

```
    printf("%d年%d月有%d天", year, month, days);
    return 0;
}
```

```
#include <stdio.h>
int get_leapyear(int year) {
    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))
        return 1;
    else
        return 0;
}
int get_days_of_month(int year, int month) {
    int a[] = { 0,31,28,31,30,31,30,31,31,30,31,30,31 };
    //          0 1  2  3  4  5  6  7  8  9  10 11 12
    int days = a[month];
    if (get_leapyear(year) && month == 2) {
        days = days + 1;
        return days;
    }
    else
        return days;
}
int main() {
    int year = 0 , month = 0;
    scanf("%d %d", &year, &month);
    int days = get_days_of_month(year, month);
    printf("%d年%d月有%d天", year, month, days);
    return 0;
}
```

链式访问

> 链式访问就是将一个函数的返回值当作另一个函数的参数

```
#include <stdio.h>

int main()
{
    printf("%d", printf("%d", printf("%d", 43)));
    return 0;
}
```

```
输出结果：
4321
```

链式访问多次调用`printf`函数。

1. 最内层的`printf("%d", 43)`会打印出`43`，并且返回打印字符的数量，也就是`2`（因为`43`有两个字符）。
2. 中间层的`printf("%d", 上一步的返回值)`，这里上一步返回`2`，所以这一层会打印出`2`，并且返回打印字符的数量，也就是`1`。

3. 最外层的`printf("%d", 上一步的返回值)`，这里上一步返回`1`，所以这一层会打印出`1`。

多个文件函数定义

> demo.c

```
#define  _CRT_SECURE_NO_WARNINGS
#include "add.h"                    //调用头文件
#include <stdio.h>
int main() {
    int a = 0, b = 0;
    scanf("%d %d", &a, &b);
    int s = add(a, b);
    printf("%d", s);
    return 0;
}
```

> add.c

```
#define  _CRT_SECURE_NO_WARNINGS
int add(int a, int b) {
    int s = a + b;
    return s;
}
```

> add.h

```
#pragma once

//函数的声明
int add(int a, int b);
```

> 多文件封装定义文件的好处：适用于多人协作，并且之前写的代码直接调用，每次不用再重复写一遍