[函数](#header-n2)  
 [1 什么是函数](#header-n3)  
 [2、函数的定义](#header-n16)  
 [3. 函数的调用及执行](#header-n36)  
 [4. 传地址](#header-n47)  
 [5. 数组作为参数](#header-n56)  
 [6. 递归函数](#header-n65)

### 函数

#### 1 什么是函数

* 函数是能够完成特定功能的代码集合。
* 函数的作用：
  + 减少程序的代码量，减少代码冗余
  + 提升代码的复用性
  + 改善程序代码的逻辑结构

#### 2、函数的定义

* 函数定义的语法
* 返回值类型 函数名 (形式参数列表)  
  {  
   函数体  
  }
  + 函数名：函数的名称，通过函数名来调用函数，函数名应该尽量有意义，根据函数的功能来定义函数名，需要满足合法标识符的定义要去。
* 返回值类型：当函数执行完成后后，需要返回的数据的类型，当函数给出返回值类型后，必须在函数体中返回相应类型的数据。 - 当函数不需要返回数据时，返回值类型应该写void - 当返回值类型为void时，函数体可以不写return语句
  + 形式参数列表：在调用函数时，需要传递给函数的数据，表示函数正常执行所需要的数据。
    - void fun() : 表示接受任意多的实参，只不过函数体执行时不会去使用这些实参。
    - void fun(void) : 表示不接受任何参数，如果在函数调用语句中传递实参了，那么编译器会报错。
  + 函数体：函数执行的具体逻辑代码。

#### 3. 函数的调用及执行

* 形式参数：函数定义语句中的参数列表
* 实际参数：函数调用时所传递给形式参数的值
* 主调函数：函数调用语句所在函数
* 被调函数：被函数调用语句调用的函数

int myAdd (int a, int b)  
{  
 int sum = 0;  
 sum = a + b;  
 return sum;  
}  
  
int main(void)  
{  
 int num = myAdd(200, 300);  
 printf("%d\n", num);  
 printf("%d\n", myAdd(100, 200));  
  
 return 0;  
}

#### 4. 传地址

* 当期望在函数调用时，在被调函数中修改主调函数中的变量值时，可以通过传递主调函数变量的地址来实现。
* #include <stdio.h>  
    
  void swap1(int a, int b)   
  {  
   int temp = a;  
   a = b;  
   b = temp;  
  }  
    
  void swap2(int \* a, int \* b)  
  {  
   int temp = \*a;  
   \*a = \*b;  
   \*b = temp;  
  }  
    
  int main(void)  
  {  
   int num1 = 100, num2 = 200;  
   swap1(num1, num2);  
   printf("num1 = %d num2 = %d\n", num1, num2);  
   swap2(&num1, &num2);  
   printf("num1 = %d num2 = %d\n", num1, num2);  
    
   return 0;  
  }
* 在实际开发中，可以通过传地址来达到返回多个返回值的效果。
* 无论是传值还是传地址，本质上都是实参往形参的值赋值过程。

#### 5. 数组作为参数

* C语言中，数组无法直接赋值
* C语言中，如果需要在函数调用时传递数组，那么通常传递数组首元素地址和数组长度，来实现传递数组的效果
* 在函数定义时，形参列表如果需要传递数组，可以将数组首元素地址定义成类似于数组的形式，例如 int p[]，但是这只是一种表现形式，本质上还是 int \* p。

// 实现函数printArr， 功能是打印数组每一个元素  
void printArr (int \* p, int len)   
{  
 int i;  
 for (i=0; i<len; i++)   
 printf("%d\n", \*(p+i));  
}  
  
int main(void)  
{  
 int arr[5] = {11, 22, 33, 44, 55};  
 printArr(arr, 5);  
   
 return 0;  
}

#### 6. 递归函数

* 递归是一种函数使用的技巧，所谓递归就是函数调用过程中直接或间接的调用函数自身。

#include <stdio.h>  
  
// 循环求阶乘  
int fun1(int n)   
{  
 int i, rtn;  
 rtn = 1;  
 for (i=1; i<=n; i++)  
 rtn \*= i;  
 return rtn;  
}  
  
// 递归函数实现步骤  
// 1、逻辑上确定递归公式 n! = n \* (n-1)!  
// 2、确定极小问题 1! = 1  
// 3、编码判断极小问题  
// 4、通过递归公式进行递归  
int fun2(int n)   
{  
 if (n == 1)  
 return 1;  
 return n \* fun2(n-1);  
}  
  
int main(void)  
{  
 printf("%d\n", fun1(10));  
 printf("%d\n", fun2(10));  
  
 return 0;  
}

#include <stdio.h>  
  
// 斐氏数列 求斐波那契数  
// 1 1 2 3 5 8 13 21 34 55 89 ....  
// 1、 f(n) = f(n-1) + f(n-2)  
// 2、 f(1) = f(2) = 1  
int fib(int n)  
{  
 if (n==1 || n==2)  
 return 1;  
 return fib(n-1) + fib(n-2);  
}  
  
int main(void)  
{  
 printf("%d\n", fib(10));  
  
 return 0;  
}