第六章 函数

一、函数的概述

二、函数的定义

三、函数的使用

四、函数传参

一、函数的概述

1.1 引言

一个较大的程序一般应分为若干个程序块，每一个模块用来实现一个特定的功能。所有的高级语言中都有子程序这个概念，用子程序实现模块的功能。在C语言中，子程序的作用是由一个主函数和若干个函数构成。由主函数调用其他函数，其他函数也可以互相调用。同一个函数可以被一个或多个函数调用任意多次。

1.2 使用函数目的

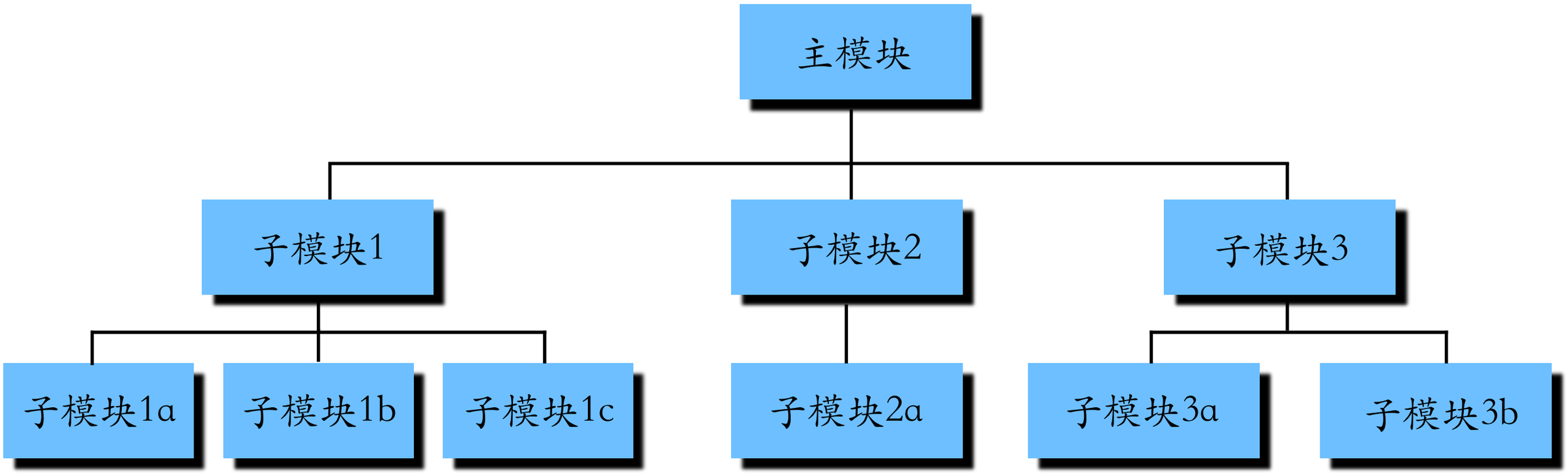
⑴ 程序“复用”，避免在程序中使用重复代码。

⑵ 自顶向下、逐步细化，将复杂问题分解为相对简单的子问题，这些问题用子程序实现，从而提高主程序结构的清晰性和易读性。

⑶ 使程序的调试和维护变得更加容易。

1.3 模块化

C语言中模块化程序设计由许多函数组合而成。可以将一个大的模块分成很多个小的模块使它们协同工作。



非模块化示例：全部写在主函数中。

```c
#include <stdio.h>
/**
 *  要求：用户从键盘输入两个整数,计算两个数的和并输出
 */
int main(int argc, const char * argv[]) {
    int a,b,sum;
    printf("请从键盘输入两个整数：\n");
    scanf("%d%d",&a,&b);
    sum = a + b;
    printf("%d与%d的和是%d\n",a,b,sum);
    return 0;
}
```

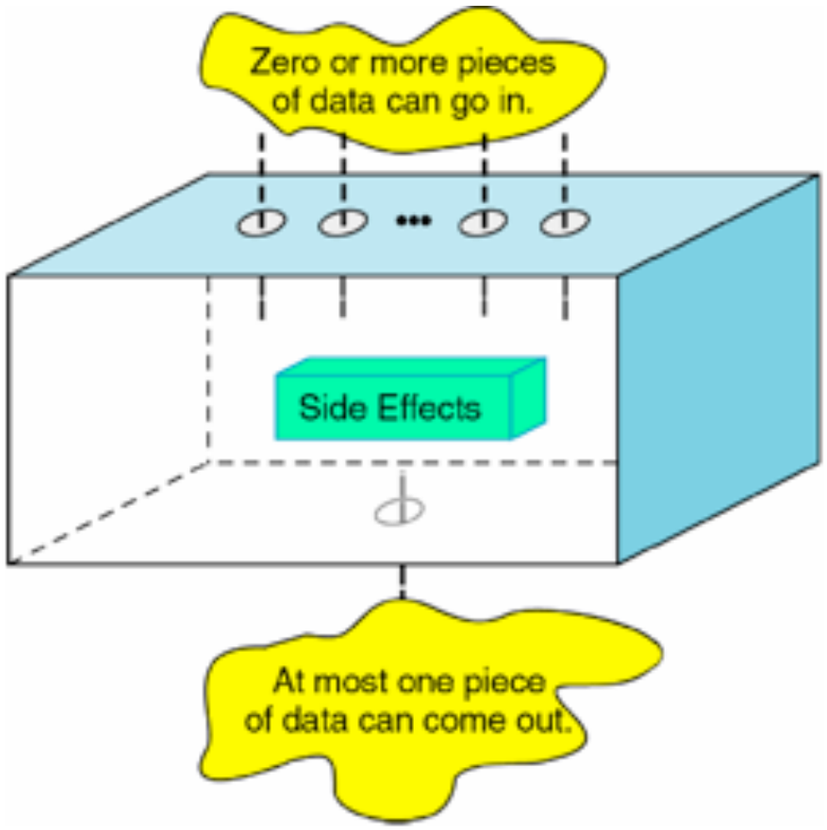
上面的程序代码中是将所有的功能实现都放在主函数中，这显然不利于我们维护代码或实现代码复用或模块化代码。

模块化示例：每一个功能都是一个单独模块，而不是将所有的代码都写在主函数中。

```c
#include <stdio.h>

int add(int x,int y)//一个函数即一个模块
{
    int sum = x+y;
    return sum;
}
int sub(int x,int y)//又一模块，实现另一个功能
{
    int s = x - y;
    return s;
}
int main(int argc, const char * argv[]) {
    int a,b,sum;
    printf("请从键盘输入两个整数：\n");
    scanf("%d%d",&a,&b);
    sum = add(a, b);
    sum = add(a+1, b+2);//add函数可被多次使用
    printf("%d与%d的和是%d\n",a,b,sum);
    return 0;
}
```

1.4 什么是函数

函数相当于一个加工厂，有多个数据片段进入，经过具体的函数功能，最终求得结果。如给定两个整数，写一个add函数，计算求得两个数的和。



编程世界中的函数是这样的:

- 函数的输入: 0个或多个数据；
- 函数的执行: 对输入数据进行加工;
- 函数的输出: 至多有一个数据;

二、函数的定义

2.1 函数的定义

函数的定义包括两部分：**函数头** 和 **函数体**

函数头部：说明函数名和类型特征。包括**函数返回值类型，函数名，参数列表**。参数列表声明参数的个数和各参数的类型。

函数体：函数体是用花括号括起来的若干语句，他们完成了一个函数的具体功能。

函数定义的一般形式：

函数头 head

```
return-type function-name(argument declarations)
{
      declarations and statements
}
```

函数体 body

三、函数的使用

函数使用的三步骤：

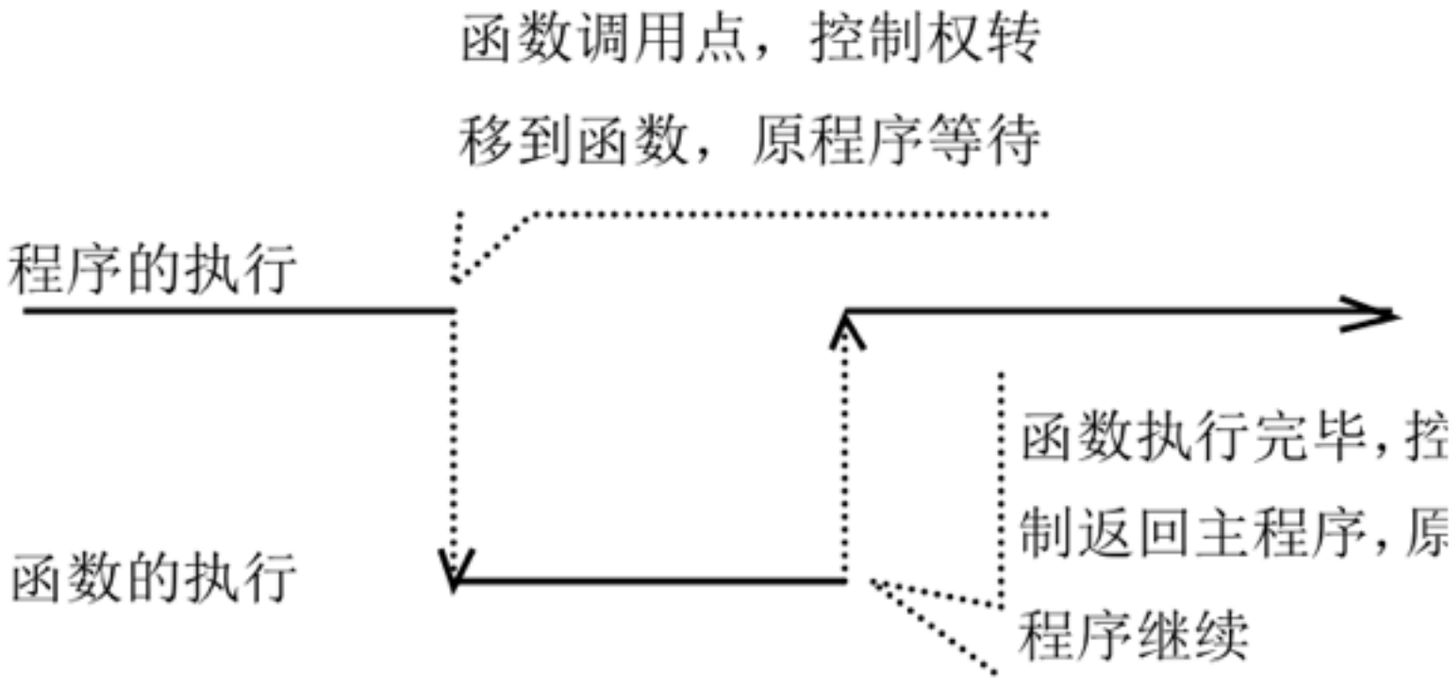
(1) 函数原型的声明，只写函数头，不写函数体，后面有分号。

(2) 函数的调用，在其它地方使用该函数称为函数调用。

(3) 函数的实现，函数的具体实现功能，包括函数头和函数体。

```c
#include <stdio.h>
int add(int x,int y); //第一步：函数声明
int main(int argc, const char * argv[]) {
    int a,b,sum;
    printf("请从键盘输入两个整数：\n");
    scanf("%d%d",&a,&b);
    sum = add(a, b);    //第二步：函数调用
    printf("%d与%d的和是%d\n",a,b,sum);
    return 0;
}
int add(int x,int y) {    //第三步：函数实现
    int sum = x+y;
    return sum;
}
```

函数执行过程：



- 函数调用时，实参值传递给形参变量；
- 暂时离开调用者，去执行函数体；
- 函数执行完毕后，回到调用的位置，继续执行调用者的语句；

3.1 函数的参数

形式参数：简称形参，指的是函数声明时的参数。

实际参数：简称实参，指的是函数调用时实际传递给函数的参数。

```c
#include <stdio.h>

int add(int x,int y);

int main(int argc, const char * argv[]) {
    int a,b,sum;
    printf("请从键盘输入两个整数：\n");
    scanf("%d%d",&a,&b);
    sum = add(a, b);
    printf("%d与%d的和是%d\n",a,b,sum);
    return 0;
}

int add(int x,int y) {
    int sum = x+y;
    return sum;
}
```

a、b即为实参

x、y即为形参

课堂练习：

```
#include <stdio.h>
#include <math.h> //数学库头文件

double hypot (double x , double y);//声明一个函数原型。
int main(int argc, const char * argv[])
{
    double side1, side2;
     printf("Enter 2 sides of Right triangle\n") ;
     scanf("%lf%lf", &side1, &side2);
     printf("The hypotenuse is: %f\n", hypot(side1,side2));//函数调
用，调用hypot。
     //实参：函数调用时，实际传递给函数的参数。
     //将打印hypot函数的返回值
     return 0;
}

double hypot(double x , double y) //形参：函数声明或定义时的函数参数。
{
     return (sqrt( x*x + y*y));
}
```

3.2 函数的返回值

无返回值示例：

```
#include <stdio.h>

void add()//无参数，无返回值
{
    int a,b,sum;
    printf("请从键盘输入两个整数：\n");
    scanf("%d%d",&a,&b);
    sum = a + b ;
    printf("%d与%d的和是%d\n",a,b,sum);
}

int main(int argc, const char * argv[]) {
    add();//调用
    return 0;
}
```

如果要求在主函数中打印两个数的和而不是在add函数中打印，应该如何修改?

返回值示例：在主函数中使用其它函数的返回值。

```c
#include <stdio.h>

int add()
{
    int a,b,sum;
    printf("请从键盘输入两个整数: \n");
    scanf("%d%d",&a,&b);
    sum = a + b ;
    return sum;//将运算后的结果返回
}

int main(int argc, const char * argv[]) {
    int s = add();//调用add函数,并将add函数的返回值赋值给s
    printf("结果为:%d\n",s);
    return 0;
}
```

注意：函数原型声明的返回值类型应当和return的值的类型保持一致。

四、函数传参

在程序设计中，给一个函数传递参数有三种方式:

(1) Pass by value 传值

(2) Pass by address 传地址（指针）

(3) Pass by reference 传引用

Q: 什么情况下需要给函数传递一个参数?

```
#include <stdio.h>
int add(){
    int a,b,sum;
    printf("请从键盘输入两个整数: \n");
    scanf("%d%d",&a,&b);
    sum = a + b ;
    return sum;
}//如果要求用户在主函数中输入两个数,在add函数中计算?
int main(int argc, const char * argv[]) {
    int s = add();
    printf("结果为:%d\n",s);
    return 0;
}
```

4.1 传值

示例代码：

```
#include <stdio.h>
int add(int x,int y);
int main(int argc, const char * argv[]) {
    int a,b,sum;
    printf("请从键盘输入两个整数：\n");
    scanf("%d%d",&a,&b);
    sum = add(a, b);
    printf("%d与%d的和是%d\n",a,b,sum);
    return 0;
}
int add(int x,int y) {
    int sum = x+y;
    return sum;
}
```

传值特点：

（1）函数调用时，将实参值的一份拷贝，赋值给形参

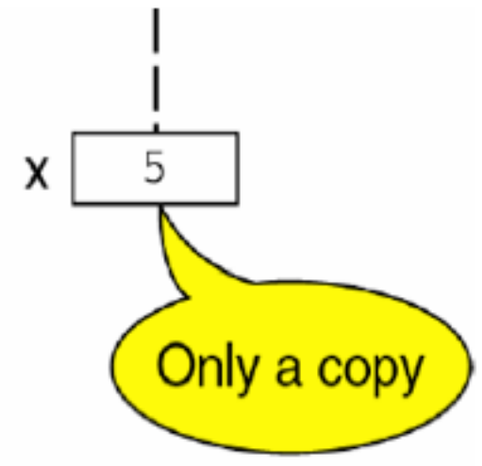
（2）单向传递，由实参传递给形参，即形参的改变不会影响实参

课堂练习：

```
/* Prototype Declarations */
void fun (int num1);

int main (void)
{
/* Local Definitions */
   int  a = 5;

/* Statements */
   fun (a)
   printf("%d\n", a);

   return 0;
}  /* main */
```

prints 5


```
void fun (int x)
{
/* Statements */
   x = x + 3;
   return;
}  /* fun*/
```



The End

渥瑞达·教学部