

Welcome

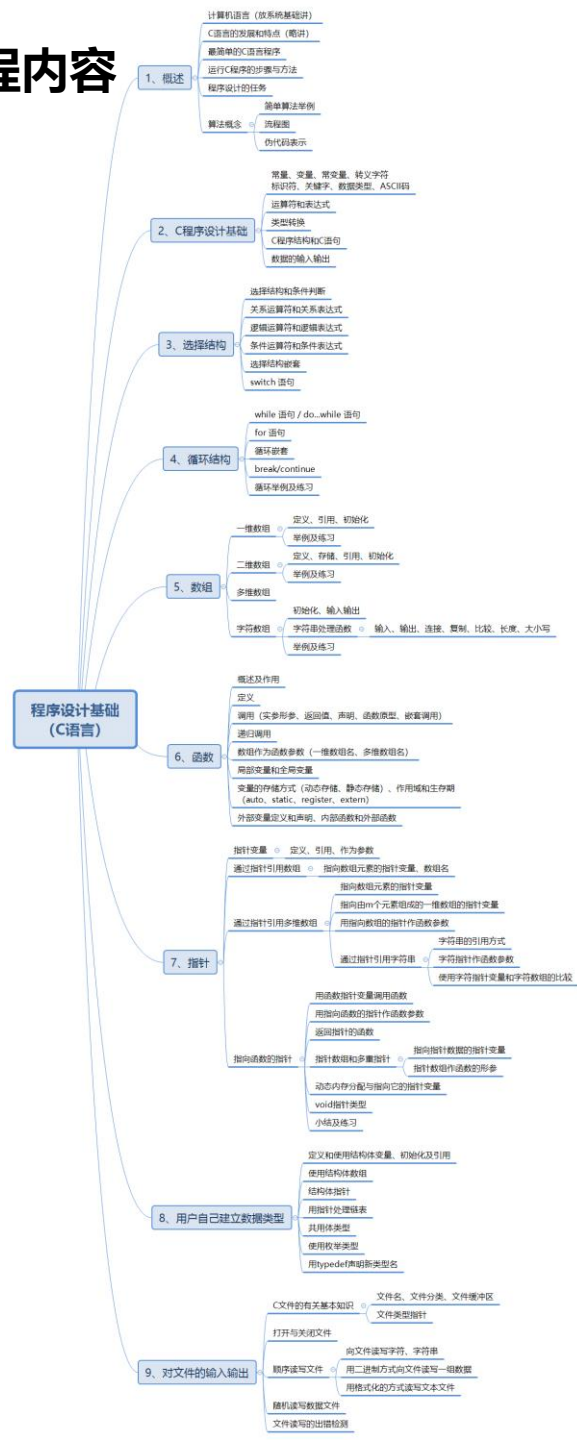
数据科学与大数据技术专业 程序设计基础(C语言)

上海体育学院经济管理学院

Wu Ying



课程内容



第5周 程序设计概述

计算机语言及C语言

最简单的C语言程序

运行C程序的步骤和方法

程序设计的任务

算法概念及简单算法

算法表示

流程图

伪代码

第6周 C程序设计基础

常量、变量及数据类型

常量变量常量变量转义字符

标识符关键字数据类型

运算符和表达式及类型转换

C程序结构和C语句

数据的输入输出

第7周 选择结构

选择结构和条件判断

运算符和表达式

关系运算符和关系表达式

逻辑运算符和逻辑表达式

条件运算符和条件表达式

选择结构嵌套

switch 语句

第8周 循环结构

while语句和do...while语句

for语句

循环嵌套, break, continue

循环练习

第9周 数组 (一)

一维数组定义、引用、初始化

一维数组练习

二维数组定义引用初始化

多维数组及练习

第10周 数组 (二) 及函数 (一)

字符数组初始化及输入输出

字符串处理函数

函数定义与调用

函数返回值、声明、原型、嵌套调用

第11周 函数 (二)

递归调用

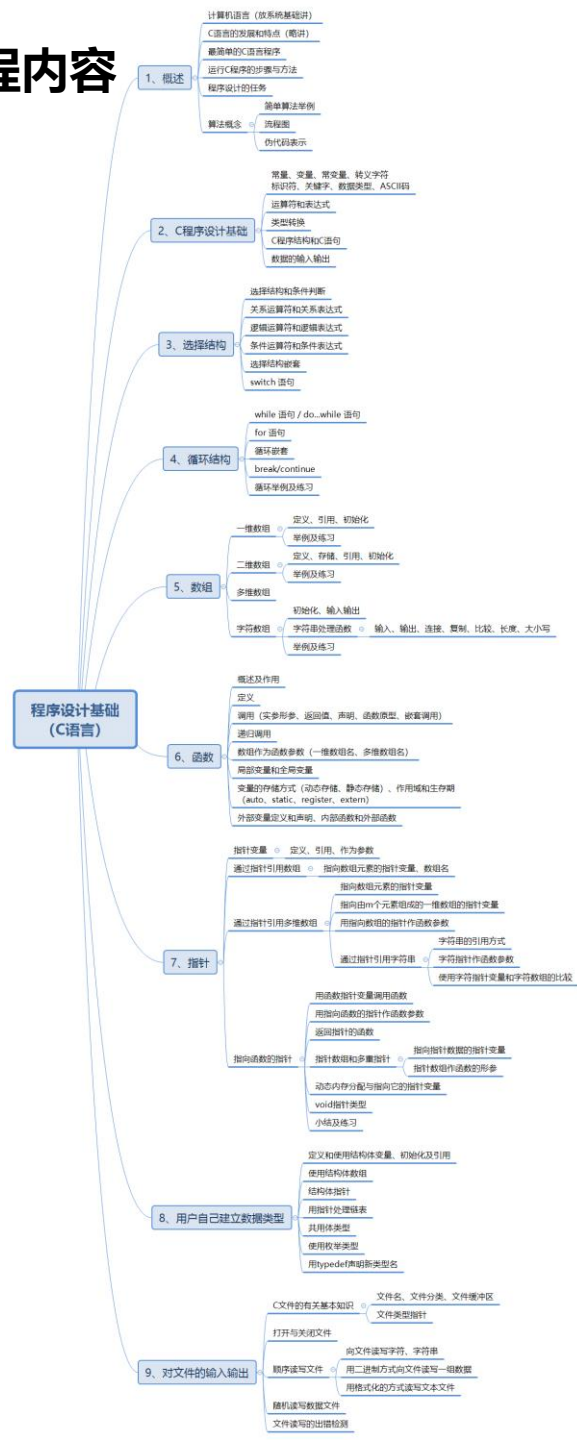
数组作为函数参数

变量作用域和生存期

变量存储方式、内部函数与外部函数



课程内容



第12周 指针 (一)

指针变量

通过指针引用数组

通过指针引用多维数组 (I)

指向数组元素的指针变量

指向一维数组的指针变量

第13周 指针 (二)

通过指针引用多维数组 (II)

用指向数组的指针作为函数参数

通过指针引用字符串

字符串的引用方式

字符指针作函数参数

使用字符指针变量和字符串数组的比较

第14周 指针 (三)

指向函数的指针 (I)

用函数指针变量调用函数

用指向函数的指针作函数参数

返回指向的函数

指向函数的指针 (II)

指针数组和多重指针

指向指针数据的指针变量

指针数组作函数的形参

动态内存分配与指向它的指针变量

第15周 指针 (四) 及结构体 (一)

指向函数的指针 (IV)

void指针类型

小结及练习

结构体

定义和使用结构体变量

使用结构体数组、结构体指针

第16周 结构体 (二) 及文件 (一)

用指针处理链表、共用体类型

枚举类型、使用typedef声明新类型

C文件的基本知识

打开与关闭文件

第17周 文件 (二)

顺序读写文件

随机读写数据文件

总复习练习

总复习练习

作业

⑤ 【P165 7】 输出“魔方阵”。所谓魔方阵是这样的方阵，它的每一行、每一列的对角线之和均相等。例如三阶魔方阵，要求输出 $1 - n^2$ 的自然数构成的魔方阵， n 为奇数。

- 1) 1 放在第一行中间一列; $[0][n/2]$
- 2) 从 2 开始直到 $n \times n$ 为止，各数依次按此规则存放：

$(k - 1) \% N == 0 \rightarrow \text{row}++;$

行-1，如果行为0， $\text{row}=N-1$ ，或者 $\text{row}=\text{row}-1$

$\text{row} = (\text{row} + N) \% N;$

列+1，

$\text{col} = \text{col} \% N$

8	1	6
3	5	7
4	9	2

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

```

2  #define N 5                //定义符号常量，魔方阵的阶
3  #include<stdio.h>
4  int main()
5  {
6      int a[N][N] = { 0 }; //数组元素初始化为0
7      int row, col;
8      int k;                //魔方阵的数
9
10     //建立魔方阵
11     row = 0;    col = (N - 1) / 2;
12     a[row][col] = 1; // 1 放在第一行的中间
13
14     //依次存放 2 至 N*N
15     for (k = 2; k <= N * N; k++)
16     {
17         if (0 == (k - 1) % N)
18         {
19             row++;
20         }
21         else
22         {
23             row--;
24             row = (row + N) % N; // 上一个数的行号减1为0，row=N-1
25
26             col++;
27             col = col % N; // 上一个数的列号增1若为N，取余为0，回到第一列
28         }
29         a[row][col] = k;
30     }
31
32     //输出魔方阵
33     for (row = 0; row < N; row++)
34     {
35         for (col = 0; col < N; col++)
36             printf("%5d", a[row][col]);
37
38         printf("\n");

```

Microsoft Visual Studio 调试控制台

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

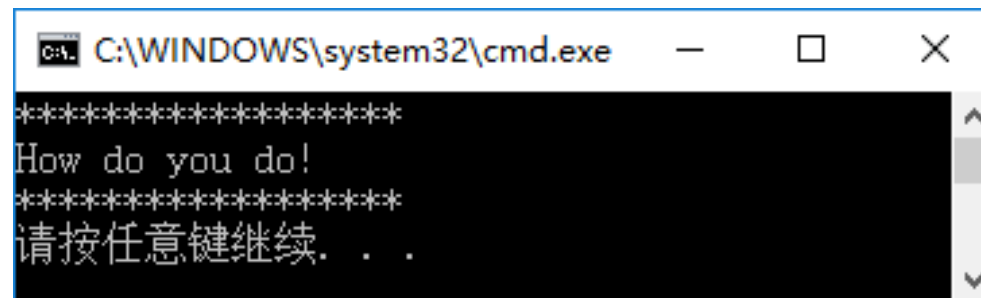
为什么要用函数

【例7.1】想输出以下的结果，用函数调用实现。

```
#include <stdio.h>
int main()
{
    void print_star();           //声明print_star函数
    void print_message();        //声明print_message函数
    print_star();                //调用print_star函数
    print_message();             //调用print_message函数
    print_star();                //调用print_star函数
    return 0;
}

void print_star()                //定义print_star函数
{
    printf("*****\n");          //输出一行*号
}

void print_message()             //定义print_message函数
{
    printf("How do you do!\n");  //输出一行文字信息
}
```

A screenshot of a Windows Command Prompt window. The title bar shows the path 'C:\WINDOWS\system32\cmd.exe'. The window contains the output of the program: a line of asterisks, the text 'How do you do!', another line of asterisks, and the Chinese text '请按任意键继续. . .' (Press any key to continue...).



为什么要用函数

- (1) 较大的程序，可分别放在若干个源文件中。便于分别编写和编译，提高调试效率
- (2) C程序的执行是从main函数开始的，如果在main函数中调用其他函数，在调用后流程返回到main函数，在main函数中结束整个程序的运行。
- (3) 所有函数都是平行的，一个函数并不从属于另一个函数.函数间可以互相调用，但不能调用main函数。main函数是被操作系统调用的。
- (4) 从用户使用的角度看：
 - ① 库函数，由系统提供，不同的C语言编译系统提供的库函数的数量和功能会有一些不同。
 - ② 自定义函数。它是用以解决用户专门需要的函数。
- (5) 从函数的形式看：
 - ① 无参函数。在调用无参函数时，主调函数不向被调用函数传递数据。
 - ② 有参函数。在调用函数时，主调函数在调用被调用函数时，通过参数向被调用函数传递数据。

定义函数

定义函数

在程序中用到的所有函数，必须 **“先定义，后使用”**。

定义函数应包括以下几个内容:

- (1) 函数名
 - (2) 函数类型
 - (3) 函数参数的名字和类型，无参函数不需要
 - (4) 函数体，实现函数功能
-

定义函数的方法

定义无参函数

```
类型名 函数名()  
{  
    函数体  
}
```

或

```
类型名 函数名(void)  
{  
    函数体  
}
```

函数名后面括号内的void表示“空”，即函数没有参数。

定义有参函数

```
类型名 函数名(形式参数表列)  
{  
    函数体  
}
```

```
int max(int x,int y)  
{  
    int z;           //声明部分  
    z=x>y?x:y;       //执行语句部分  
    return(z);  
}
```

空函数

```
类型名 函数名()  
{ }
```

函数体为空，什么也不做。

调用函数

函数调用的形式

函数名(实参表列)

```
print_star(); //调用无参函数  
c=max(a,b); //调用有参函数
```

1. 函数调用单独作为一个语句

```
printf_star();
```

2. 函数表达式

函数带回一个确定的值以参加表达式的运算

```
c=max(a,b);
```

3. 函数返回值作为另一个函数的实参

```
m=max(a,max(b,c));
```

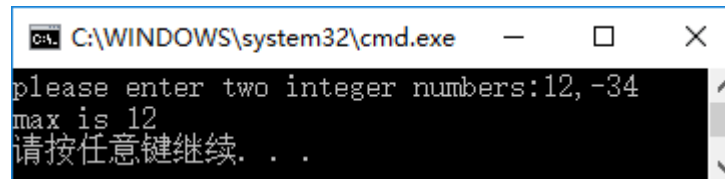
```
printf ("%d", max (a,b));
```

实参和形参间的数据传递

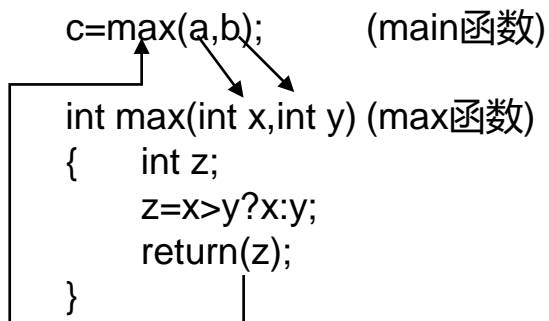
【例7.2】输入两个整数，要求输出其中值较大者。要求用函数来找到大数。

```
#include <stdio.h>
int main()
{
    int max(int x,int y); //对max函数的声明
    int a,b,c;
    printf("please enter two integer numbers:"); //提示输入数据
    scanf("%d%d",&a,&b); //输入两个整数
    c=max(a,b); //调用max函数，有两个实参。大数赋给变量c
    printf("max is %d\n",c); //输出大数c
    return 0; }
```

```
int max(int x,int y) //定义max函数，有两个参数
{
    int z; //定义临时变量z
    z=x>y?x:y; //把x和y中大者赋给z
    return(z); //把z作为max函数的值带回main函数
}
```



```
C:\WINDOWS\system32\cmd.exe
please enter two integer numbers:12,-34
max is 12
请按任意键继续. . .
```



在调用函数过程中发生的实参与形参间的数据传递称为“虚实结合”。

函数调用的过程

- (1) 形参，函数未被调用时，并不占内存中的存储单元。发生函数调用时，函数的形参才被临时分配内存单元。
- (2) 将**实参的值**传递给对应形参。
- (3) 在函数运行期间，由于形参已经有值，利用形参进行运算。
- (4) **通过return语句将函数值带回到主调函数**。应当注意返回值的类型与函数类型一致。**如果函数不需要返回值，则不需要return语句**。这时函数的类型应定义为void类型。
- (5) **函数调用结束，形参单元被释放**。

注意

- **实参向形参的数据传递是“值传递”，单向传递，只能由实参传给形参**，实参和形参在内存中占有不同的存储单元,实参无法得到形参的值。

函数的返回值

(1) **通过函数中的return语句获得。** 一个函数中可以有一个以上的return语句，执行到哪一个return语句，哪一个return语句就起作用。 “return z;” 与 “return(z);” 等价。return后面的值可以是一个表达式。

(2) **函数值的类型。** 函数值的类型在定义函数时指定。

int max (float x, float y)	//函数值为整型
char letter (char c1,char c2)	//函数值为字符型
double min (int x,int y)	//函数值为双精度型

(3) **函数类型决定返回值的类型，return语句中的表达式类型应该和函数类型一致。**

如果不一致，则以函数类型为准。对数值型数据，可以自动进行类型转换

(4) **对于不带回值的函数，应当用定义函数为“void类型”**（或称“空类型”）

此时在函数体中不出现return语句

函数的返回值

【例7.3】 将例7.2稍作改动，将在max函数中定义的变量z改为float型。

函数返回值的类型与指定的函数类型不同，分析其处理方法。

```
#include <stdio.h>
int main()
{
    int max(float x,float y);
    float a,b;
    int c;
    scanf("%f,%f",&a,&b);
    c=max(a,b);
    printf("max is %d\n",c);
    return 0; }
```

```
int max(float x,float y)
{
    float z;      //z为实型变量
    z=x>y?x:y;
    return(z);
}
```

C:\WINDOWS\system32\cmd.exe

```
1.5, 2.6
max is 2
请按任意键继续. . .
```

被调函数的声明和函数原型

- (1) 首先**被调用函数**必须是已定义的函数
- (2) 库函数，在本文件开头用#include指令将库函数所在文件“包含”进来。
- (3) 自定义函数，如果该函数位于**调用它的函数（即主调函数）**的后面（在同一个文件中），则应该在主调函数中对被调用的函数作声明(declaration)。

声明的作用是把函数名、函数参数的个数和参数类型等信息通知编译系统，以便在遇到函数调用时，编译系统能正确识别函数并检查调用是否合法。

被调函数的声明和函数原型

【例7.4】输入两个实数，用一个函数求出它们之和。

```
#include <stdio.h>
int main()
{
    float add(float x, float y);    //函数声明
    float a,b,c;
    printf("Please enter a and b:"); //提示输入
    scanf("%f%f",&a,&b);           //输入两个实数
    c=add(a,b);                     //调用add函数
    printf("sum is %f\n",c);         //输出两数之和
    return 0;
}

float add(float x,float y) //定义add函数
{
    float z;
    z=x+y;
    return(z);              //把变量z的值作为函数值返回
}
```

- ① float add(float x, float y);
- ② float add(float, float); //不写参数名，只写参数类型
- ③ float add(float a, float b); //参数名不用x,y，而用a,b。合法

函数的首行(即函数首部)称为函数原型(function prototype)。



函数名、函数返回类型、形参类型

函数原型与函数声明保持一致

声明中的形参名可以省略，可以与函数定义（原型）中的形参名不一致

被调函数的声明和函数原型

如果已在文件的开头(在所有函数之前), 已对本文件中所调用的函数进行了声明, 则在各函数中不必对其所调用的函数再作声明。

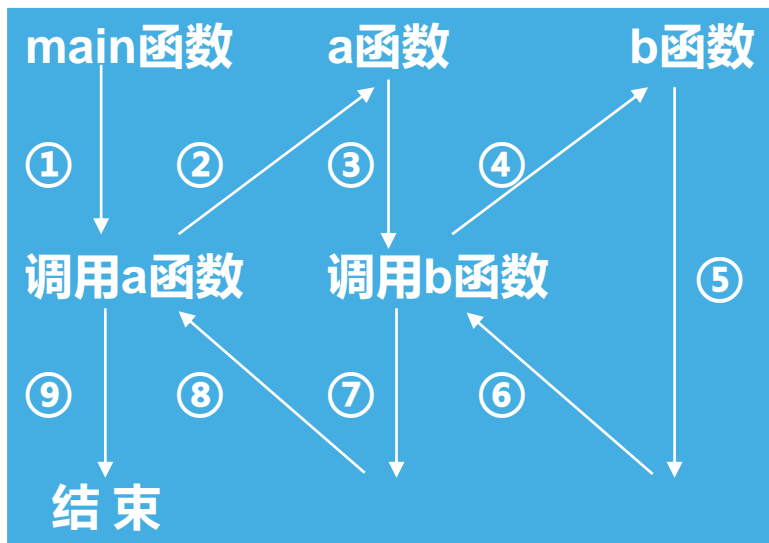
```
char letter(char, char);  
float f(float, float);  
//所有函数之前, 且在函数外部进行函数声明  
int main() //在函数中如果调用 letter, f 函数, 不必再对其声明  
{ ... }  
  
char letter(char c1, char c2)  
{ ... } //定义letter函数  
  
float f(float x, float y)  
{ ... } //定义f函数
```

注意

- 函数的定义是指对函数功能的确立, 确定函数名、函数值类型、形参及其类型, 完成函数体, 是一个完整、独立的单位。
- 函数的声明是把函数的名字、函数类型以及形参的类型、个数以及顺序通知编译系统, 以便在调用该函数时, 系统按此进行对照检查。

函数的嵌套调用

- ① 执行main函数的开头部分;
- ② 遇函数调用语句, 调用函数a, 流程转去a函数;
- ③ 执行a函数的开头部分;
- ④ 遇函数调用语句, 调用函数b, 流程转去函数b;
- ⑤ 执行b函数, 如果再无其他嵌套的函数, 则完成b函数的全部操作;
- ⑥ 返回到a函数中调用b函数的位置;
- ⑦ 继续执行a函数中尚未执行的部分, 直到a函数结束;
- ⑧ 返回main函数中调用a函数的位置;
- ⑨ 继续执行main函数的剩余部分直到结束。



习题 P215

1. 写两个函数，分别求两个整数的最大公约数和最小公倍数，并在main()函数中调用这两个函数，输出结果。两个整数由键盘输入。（greatest common divisor, gcd ; least common multiple, lcm)

```
4   int gcd(int m, int n); //求最大公约数
5   int lcm(int m, int n, int gcd);
6
7   int main(void)
8   {
9       int s, t;
10      int g, m;
11      printf("input two numbers:");
12      scanf("%d%d", &s, &t);
13      g = gcd(s, t);
14      m = lcm(s, t, gcd(s, t));
15      printf("最大公约数为: %d\n最小公倍数为: %d\n", g, m);
16  }
17  int gcd(int m, int n)
18  {
19      int r;
20      r = m % n;
21      while (r != 0)
22      {
23          m = n;
24          n = r;
25          r = m % n;
26      }
27      return n;
28  }
29  int lcm(int m, int n, int gcd)
30  {
31      return m * n / gcd;
32  }
33
```

Microsoft Visual Studio 调试控制台

```
%
input two numbers:24 16
最大公约数为: 8
最小公倍数为: 48
temp.
```