C语言程序设计

入门篇

——带你认识C语言

学习目标

- 了解计算机语言的概念及其发展史
- 了解常见的高级语言
- 掌握指令、程序、源程序、目标程序的概念
- 理解C语言的基本特点和上机步骤

一、计算机语言的概念

自然语言：人与人之间进行交流的语言

计算机语言：是人和计算机进行信息交流的工具，人们可以使用计算机语言来命令计算机进行各种操作处理。

二、计算机语言的发展

机器语言： 由0和1二进制代码构成。

例：“加”命令用二进制10110111表示。

汇编语言： 用一组易记的符号代表机器指令。

例：“加”命令用add表示。

高级语言： 接近于人们习惯使用的自然语言和数学语言。又称为类自然语言。

例：“加”命令+表示。

三、常见的高级语言及用途

- **C语言：** 编写系统软件,如编写UNIX, Windows,Linux等操作系统.
- **FORTRAN：** 用于数值计算,如微分方程数值解,比如气候模式，海洋模式，模拟核爆炸试验
- **VB：** 开发应用软件
- **JAVA：** 网络环境语言,编手机上的程序或游戏.
- **C++、VC++、Dephi、FoxPro**

01110001 11010011 01110001 11010011 01110001 11010011 01110001
11010011 01110001 11010011 01110001 11010011 01110001 11010011
01110001 11010011 01110001 11010011 01110001 11010011 01110001
01110001 11010011 01110001 11010011 01110001 11010011 01110001
01110001 11010011 01110001 11010011 01110001 11010011 10110111
01010011 01110001 11010011 01110001 11010011 01110001 11010011
11110000 01110001 11010011 01110001 11010011 01110001 11010011
00110110 01110001 11010011 01110001 11010011 01110001 11010011
01110001 11010011 01110001 11010011 01110001 11010011 11110000
01101010 01110001 11010011 01110001 11010011 01110001 11010011
10101010 01110001 11010011 01110001 11010011 01110001 11010011
01110001 11010011 01110001 11010011 01110001 11010011 10101011
01110001 11010011 01110001 11010011 01110001 11010011 10101010
10110111 01110001 11010011 01110001 11010011 01110001 11010011
11011101 01110001 11010011 01110001 11010011 01110001 11010011
01010011 01110001 11010011 01110001 11010011 01110001 11010011
01110001 11010011 01110001 11010011 01110001 11010011 10101010
01110001 11010011 01110001 11010011 01110001 11010011 10111101
01110001 11010011 01110001 11010011 01110001 11010011 01110001
01110001 11010011 01110001 11010011 01110001 11010011 11011111

加法器源程序

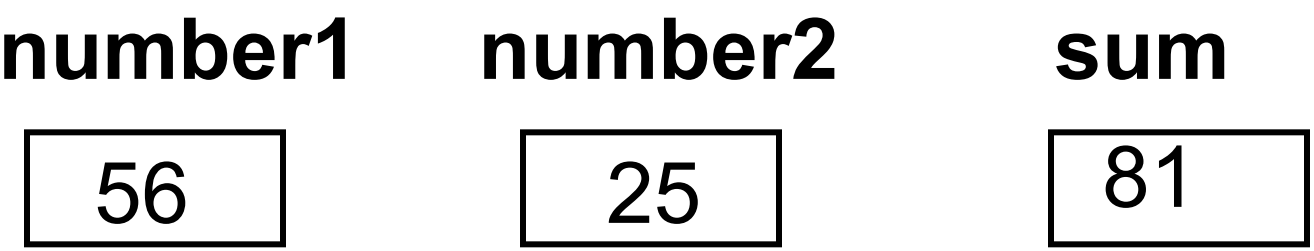
```
 main( )
{int a,b,sum;
 scanf(“%d%d”,&a,&b);
 sum=a+b;
 printf(“sum=%d”,sum);
 }
```

六、走近C语言、认识C语言

由加法器可执行程序引起的几个问题？

1、编写程序需要遵循一定的格式吗？是什么样的格式呢？

2、输入的数据放在什么地方？



3、如何使从键盘输入的数据到指定的地址中呢？

4、如何将计算好的结果数据输出来？

加法器源程序代码

```
main( )                                  /*主函数main( )*/
{int  number1,number2,sum;    /*定义3个变量*/
  printf("请输入加数:");
  scanf(“%d”,&number1);   /*向number1中输入数据*/
  printf("请输入被加数:");
  scanf("%d",&number2);
  sum=number1+number2;
  printf("\n%d+%d=%d",number1,number2,sum); }
```

思考： 加法器的局限性在于，它只能对整数进行加法运算，是否可以编写这样的程序，可以对任意两个整数进行加和减运算呢？

分析： 该计算器实现两个功能：加法功能和减法功能，因此，分别编写两个函数实现这两个功能：加法函数sum( )，减法函数sub( )。

加法函数sum( )

```
sum (int  x,int y)
{int  sum;
 scanf(“%d”,&x);
 scanf(“%d”,&y);
 sum=x+y;
 printf("%d+%d=%d\n",x,y,sum);
 }
```

减法函数sub( )

```
sub(int x,int y)
{int sub;
 scanf(“%d”,&x);
 scanf(“%d”,&y);
 sub=x-y;
 printf("%d-%d=%d\n",x,y,sub);
 }
```

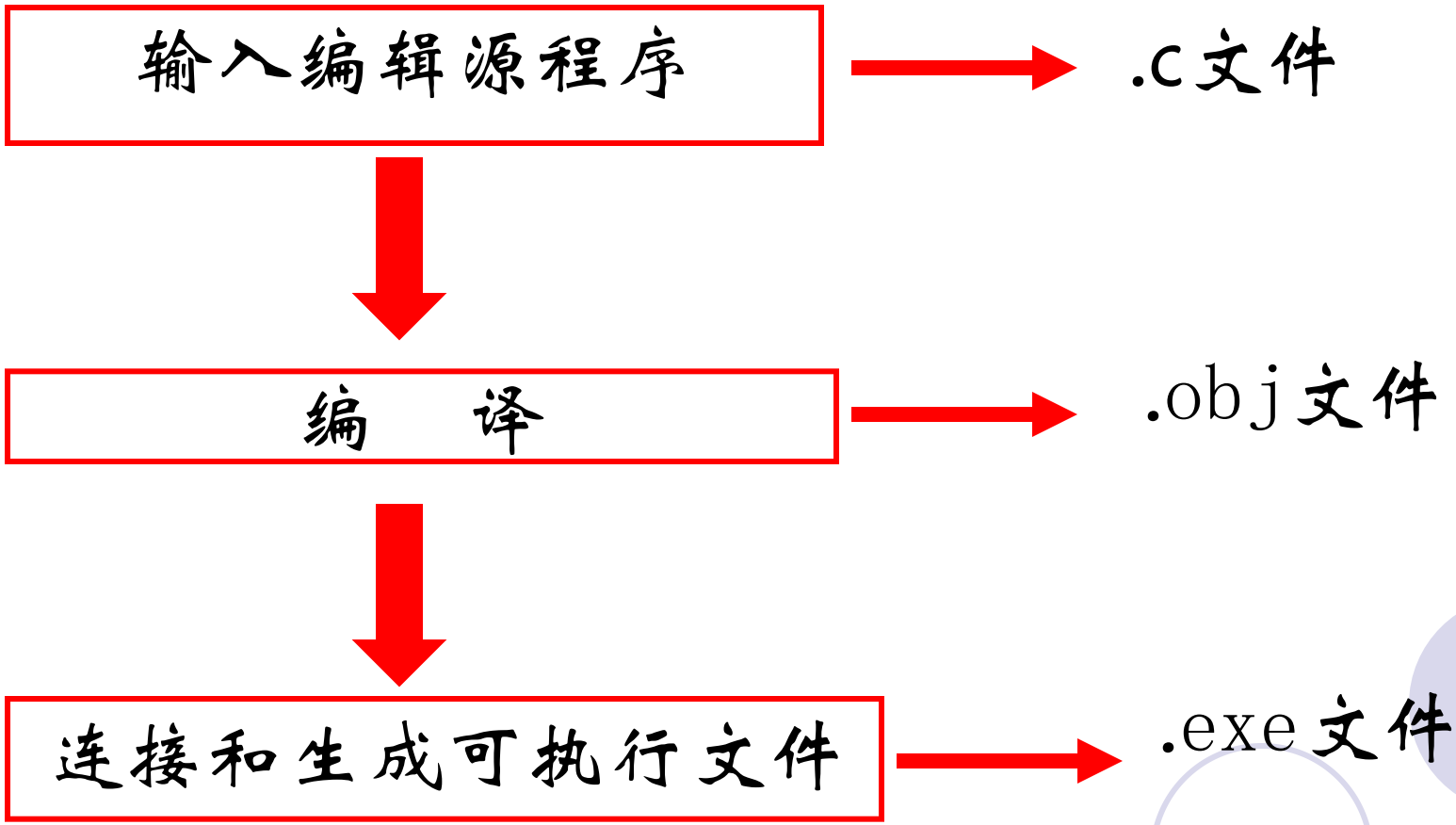
主函数

```
main( )
{int  number1,number2;
 int i;
 printf(“请输入您的选择  1：+    2：-”);
 scanf("%d",&i);
 if(i==1) sum(number1,number2);
 if(i==2) sub(number1,number2);
 }
```

七、总结—关于C语言程序结构的介绍

- C语言程序是由函数构成的，**有且只有一个主函数main。**
- 一个函数由两部分组成：**函数的首部和函数体。函数体有声明部分和执行部分组成。**
- 函数的执行总是先**从主函数main**开始执行。main函数的位置可以任意。
- 每条语句后都有一个分号“；”作为间隔，分号不可少。一行内可以写多个语句，一个语句也可以写在多行上。

八、C语言的上机步骤



小结

- 计算机语言是人和计算机之间交流的语言
- C语言是常见的高级语言，它具有低级语言和高级语言的特点，**不但可以用来编写应用软件，也可以用来编写系统软件.**
- 用高级语言编写出来的程序称为**源程序**，经过编译以后生成**目标程序**，链接后形成可执行程序.
- 就象自然语言要遵循一定的语法规则一样，使用C语言编程时也要遵循一定的语法规则.

编写一个程序，输入一个整数，求该整数的立方。

```
main( )
{ int  number1,sum;
  scanf(“%d”,&number1);
  sum=number1*number1*number1;
  printf(“%d*%d*%d=%d”,number1,number1,number1,sum);
}
```

第二章 数据类型

程序＝数据结构＋算法

在程序中定义所要处理数据的类型和组织形式

对解题方法和解题步骤的描述

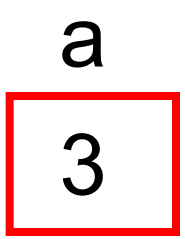
C中最常见的三种基本类型：

- 整 型
- 实 型
- 字符型

2.1 常量和变量

int a; → 定义了一个整型变量，名为a

a=3; → 将常数3赋给变量a



变量： 在程序运行过程中，其值可以改变的量。每个变量在计算机中对应相应长度的存储空间。

常量： 在程序运行过程中，其值不变的量。

一、变量

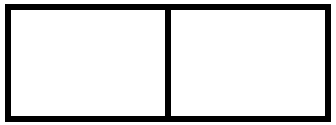
每个变量在使用之前**必须确定其类型**，并为之取一个**合适的名字**。

例：

```
int  age;
```

age 

```
float  score;
```

score 

```
char sex;
sex='a';
```

sex 

定义变量类型的一般形式为：

类型标志符 变量名；

标志符的命名规则：

(1) 标识符只能由**字母、数字、下划线**组成，且**第一个字符不能为数字。**

(2) C语言**区分大小写。**

(3) 标识符的名字不能和C语言中的**关键字**和**特定字**相同。

(4) 标识符的长度最好不要超过8个字符。

(5) 标识符的命名最好做到**见名知意**。如age用来表示年龄,number表示学号。

二、常量

常见的常量如18，65.5，3.14159，‘m’，‘ding’

1、整型常量（整常数）

1）、十进制形式表示：如25，65，23。

2）、八进制形式表示：以数字0开头，如025

3）、十六进制形式表示：以数字0x开头，如0x25，0x65，0x1A。

2、实型常量（实数、浮点数）

十进制小数形式：10.25，-1.2314

指数形式表示：

如23500可以用指数形式表示为：2.35e4

0.00014 可以用指数形式表示为：1.4e-4

3、字符型常量

1）、普通字符常量

常见的字符常量有：’a’, ‘A’, ‘F’, ‘*’, ‘■’, ‘▼‘

2）、转义字符常量

如’\n’表示换行的意思

‘\r’表示使光标回到本行的开头

‘\b’表示使光标回到前一列

‘\t’表示使光标移到下一个Tab的位置

‘\a’表示响起报警声

‘\\’表示一个\

‘\’’表示单撇号’

‘\”’表示双撇号”

‘\ddd’表示一到三位八进制数所代表的字符

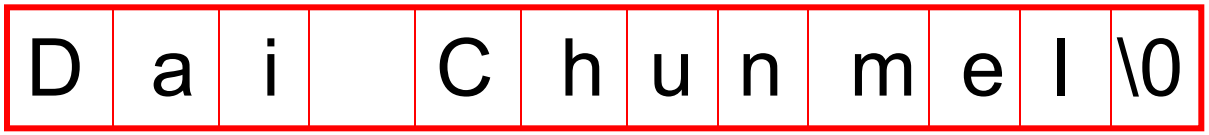
‘\xdd’表示一到二位十六进制所代表的字符

4、字符串常量

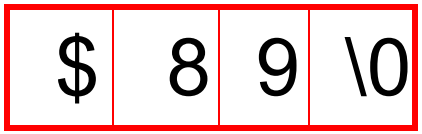
字符串常量就是由多个字符组成的字符序列，用双撇号括起来。

例：

“Dai Chunmei”



“$89 ”



回顾

1、C语言中三种最基本的数据类型包括：（ ）

A、整型、实型、逻辑型　　B、整型、实型、字符型

C、整型、字符型、逻辑型　D、整型、实型、逻辑型、字符型

2、在C语言中，下面哪一个不是整型常量（ ）

A、123　　B、123L　　C、0x123　　D、U123

3、在C语言中，下面哪一个不是字符型常量（ ）

A、‘a’　　B、‘\81’　　C、‘\0x41’　　D、“a”　　E、‘\t’

4、下列选项哪个选项不符合C语言中变量名的命名规则（ ）

A、age_1　　B、1_age　　C、_1age　　D、age*1

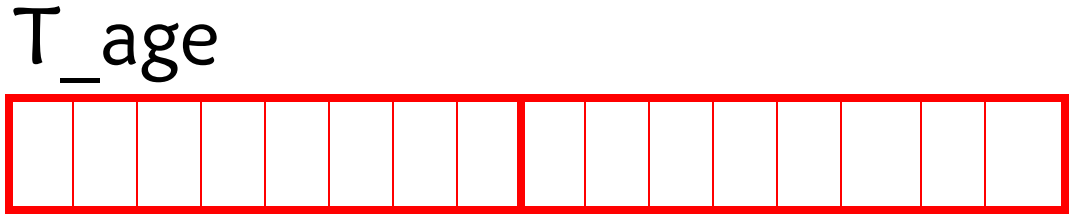
2.2 整型变量

思考题：现在需要编写一个教师的工资管理系统，其中涉及到一个变量，用来存放教师的工龄。该如何定义。

```
int  T_age;
```

类型标志符

变量名

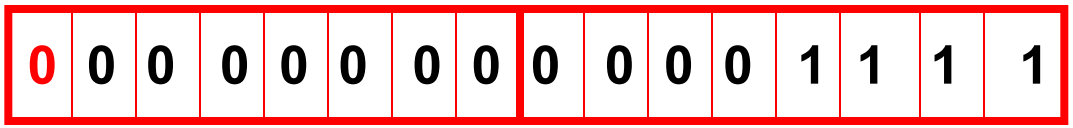


int型变量占2个字节的存储空间

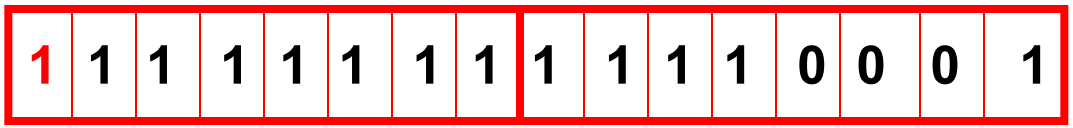
一、整型数据在内存中的存储形式

补码表示，最高位表示符号位，**正数用0表示，负数用1表示。**

假设使T_age=15，则15在计算机中的存储形式为：



假设使T_age=-15，则15在计算机中的存储形式为：



所以，int型数据存储范围为：　-2^{15} —— $2^{15}-1$

-32768　　32767

返回　　下一页

二、 整型变量的分类

- 整型
  - short int（短整型）
    - [signed] short [int]
    - unsigned short [int]
  - int（基本整型）
    - [signed] int
    - unsigned [int]
  - long int（长整型）
    - [signed] long [int]
    - unsigned long [int]

第三章 运算符和表达式

当变量被定义为某一种类型后，即被分配相应的存储空间，此后并不能放置一旁而不用，还需要对其进行**加工**。

何谓加工？加工就是指运算，C中最常见的运算有**加、减、乘、除**等。

运算符就是用来表示运算的符号，如“+”、“-”、“*”、“/”。

参加运算的数据称为**运算量**，也就是运算对象。由运算符把运算对象连接起来的式子称为**表达式**，如“sum＝a+b；”就是一个表达式。

C语言中常见的运算符

3.1 算术运算符和表达式

一、常见的算术运算符

+　　-　　*　　/　　%

当两个整数相除时，**商的结果取整**。如5/3的结果为1，-5/3的结果为-1。

当两个除数中有一个为实数时，**结果为实数**，如5/2.0=2.5。

实现两个整数的**相除取余**，结果的符号和被除数的符号相同。参加取余运算的两个数必须是整数。

二、自增、自减运算符

++　　--

1)自增运算符++: 使变量的值增1

例: i++; 表示使用完i之后,使i的值增1

++i; 表示先使i的值增1,然后再使用i

2)自减运算符--: 使变量的值减1

例: i--; 表示使用完i之后,使i的值减1

--i; 表示先使i的值减1,然后再使用i

例1:

```
int a,b,c;
a=5;
b=++a;
c=a++;
```

结果: a=7;
b=6;
c=6;

例2:

```
int a,b,c;
a=5;
b=- -a;
c=b- -;
```

结果: a=4;
b=3;
c=4;

三、算术运算符的优先级和结合性

思考题： 表达式- -x-a*b/c++的结果值是多少？（假设x的值为3，a的值为5，b的值为2，c的值为3）

运算符的优先级： 运算符的优先次序

运算符的结合性： 当优先级相同时，是自左至右或是自右至左计算

++、--、+（取正）、-（取负）　　自右至左

↓

*、/、%　　自左至右

↓

+、-　　自左至右

- -x-a*b/c++ → （- -x）-a*b/c++ → -1

3.2 赋值运算符和表达式

1、普通的赋值运算符

number= x*y;

赋值运算符

注意：

当赋值运算符右边的常量或表达式的类型和左边变量的类型不一致时，先将右边表达式的类型转换为和左边相同的类型，然后再赋值。

例：

```
float x;
X=3;
```

2、复合的赋值运算符

x=x+y; ⟶ x+=y;

复合的赋值加运算符

常见的复合的赋值运算运算符

(1) +=
(2) -=
(3) *=
(4) /=
(5) %=

例如: x的值为6，

则 x+=4 的结果是什么？

3、赋值运算符的优先级和结合性

算术运算符

↓

赋值运算符　　结合性为自右而左

例：已知a=5，b=8，n未知。求下列各表达式的值。

1、n=b+8；

2、n+=a*=2；

3、n=a=b；

3.3 关系运算符和表达式

关系运算又称比较运算，就是对两个运算量进行比较，判断其比较的结果是否符合给定的条件。如果符合，则结果为“真”，用1表示；如果不符合，则结果为“假”，用0表示。

1、常见的关系运算符

<（小于）、>（大于）、<=（小于或等于）、>=（大于或等于）、==（等于）、!=（不等于）

2、关系运算符的优先级和结合性

算术运算符　　自左至右

↓

关系运算符　　自左至右

↓

赋值运算符　　自右至左

例：number1=25；判断下列表达式的值。

（1）number1>=25;

(2) number1==25;

（3）number1%5==0

（4）number1!=24;

(5) number1/3<23;

（6）n=number1>4;

(7) number1>24>2;

(8) number1+1==26;

实训

已知i=3，j=4，a=5，b=6，m为未知。求下列各式的值。

（1）++i+j---a

（2）a+b-j*a/i

（3）m=-j++

3.4 逻辑运算符和表达式

思考题:如何表示整数x既可以被5整除又可以被7整除.

(x%5==0) && (x%7==0)

↓

逻辑运算符”逻辑与”

逻辑与(&&):一般形式:
a && b;
只有当a和b都为真时,结果才为真.

逻辑或(||): 一般形式 a||b, a和b中任一为真,结果为真.

逻辑非(!): 一般形式 !a, a为真,!a就为假; a为假,!a为真.

例:判断下列各逻辑表达式的值

假设a=5, b=6, c=3, d=10;

(1) (a>b) && (c>d); **0**

(2) (a<b) ||(c>d) **1**

(3) !a **0**

(4) !a && b **0**

(5) 3||1 **1**

逻辑运算符的优先级和结合性

高 ！（逻辑非） 自右至左

算术运算符

关系运算符

&&和||

低 赋值运算符 自左至右

X=!a+2 || x>y+2&&a==3, 已知a=3,x=5,y=4, 求x的值。

实训

1、若a=5，b=4，x=0，判断下面两个表达式的结果及x的值：

(1) 0&&x=a>b

(2) a>b || ++x

2、要判别某一年是否闰年。闰年的条件是符合下面两个条件的其中一个就可以了：

(1)能被4整除，但不能被100整除；

(2)能被4整除，又能被400整除；

(year%4==0&&year%100!=0) ||(year%400==0)

3.5 位运算符和表达式

位运算:就是对二进制位进行的运算.

C中常用的位运算符:

- &(按位与)
- |(按位或)
- ^(按位异或)
- ~（取反）
- >>（右移）
- <<（左移）

“按位与”运算符‘&’

“按位与”又称“按位乘”，即二进位相乘。

“按位与”的规则为：

0&0=0,　0&1=0,　1&0=0,　1&1=1

例：求-1和5按位与的结果.

-1的补码: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

5的补码: 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1

按位与的作用：使一个存储单元清零

3.6 逗号运算符和表达式

，

逗号运算符
作用在于将若干个
表达式连接起来。

逗号表达式的一般形式为：

表达式1，表达式2，表达式3，……，表达式n

整个逗号表达式的值就是最后的表达式n的值

实训

已知a=3，求下面逗号表达式的值。

（1）++a，a+5，a+6

（2）a=2*3，b=a-2，b-a，a+=2

3.7 sizeof运算符和强制类型转换运算符

一、求字节数运算符sizeof( )

功能： 求数据在存储器中所占的字节数

例：

```
sizeof( long double );
sizeof( x+y );
sizeof( 123456 );
```

二、强制类型转换运算符

思考： 假设现在需要编写一个程序，判断一个实数x的整数部分是否为偶数， 要求x从键盘输入。

```
main( )
{ float x;
  scanf("%f",&x);
  if( _____ ) printf("Yes!");
  else printf("No!");
```

强制类型转换运算符的一般使用形式：

（类型名）（表达式）；

希望转换的类型　　希望被转换的对象

已知x=2.5， y=3.7，

求（int）x+y 和 (int)(x+y)的值

实训

1、求下列算术表达式的值

（1）a+b/2%3*(int)c/d*3

设a=3.5, b=11, c=5.7, d=2

（2）(int)x/y+(float)y%x

设x=8.3,y=3

（3）！（x+y）+z&&x*z-1

x的值为2,y的值为3,z的值为1

思考：求任一4位八进制所对应的十进制数

```
main( )
{int x1,x2,x3,x4,o_number,d_number;
 scanf( "%d", &o_number );
x1=o_number/1000;
x2=o_number%1000/100;
x3=o_number%100/10;
x4=o_number%10;
d_number =x1*8*8*8+x2*8*8+x3*8+x4;
printf ("The decimal of %d is
    %d\n“ ,o_number,d_number); }
```

第四章 输入/输出函数

char a;

0	1	0	0	0	0	0	1

输入

scanf (“ %c ”,&a);

a=getch( );

gets( );

输出

printf (“ %c ”, a);

putchar (a);

puts( );

#include “stdio.h ” 或 #include<stdio.h>

4.1 输出函数putchar( )和printf( )

一、 字符输出函数putchar( )

1、功能

输出一个字符

2、一般形式

putchar(c);

可以是 字符变量
字符常量
整型常量
整型变量
转义字符

putchar (c);
putchar (‘c’);
putchar (99);
putchar (c);
putchar (‘\n’);

一、 格式输出函数printf( )

1、功能

它可以将任意多个数据按各自指定的格式输出来。

例：

```
int number=24;
float score=95.5;
char sex='m';
printf(" number=%d, score=%f, sex=%c ",
        number, score, sex);
```

输出结果: number=24, score=95.000000, sex=m

二、一般形式

printf（“格式控制”，输出表列）；

三、格式说明符

1、%d：用来输出十进制int型数据

%md：m是一个常数，用来限定输出数据所占的宽度。

%ld：以长整型形式输出数据。

2、%u格式符　输出十进制unsigned型数据。

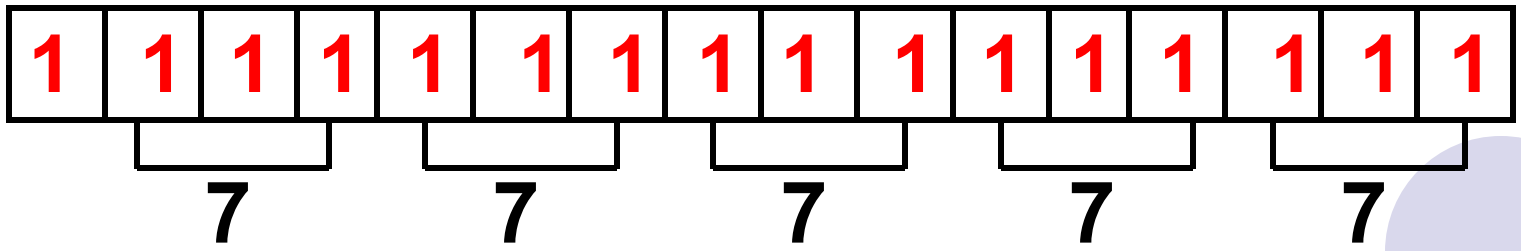
3、%o格式符　以八进制无符号形式输出整数，输出时将符号位一起转化为八进制数值。

4、%x格式符　以十六进制无符号形式输出整数，输出时将符号位一起转化为十六进制数值。

实训

```
main( )
{int a=-1;
 printf("%d,%o,%8o",a,a,a);}
```

a在计算机中的存储形式



输出结果：-1，177777， 177777

5、%c格式符　以字符形式输出数据

例1
```
char ch;
ch='a';
printf(" %d ", ch);
```

字符型数据可以以字符形式输出，也可以以整型形式输出。

例2
```
int  a;
a=65;
printf(" %c ", a);
```

整型数据（0—255）也可以以字符形式输出。

6、%f格式符　以实数形式输出数据

%f：输出带6位小数的实数。对于单精度实数，有效位数为7位

%m.nf：指定输出数据占m列，其中有n位小数

%.nf: 表示带n位小数,对列宽无限制。

实训

```
main( )
{float x=12345.1111001;
printf("%f\n",x);
 printf("%10.2f,%-10.1f,%.2f",x,x,x);}
```

输出结果：12345.111328
12345.11
12345.1
12345.11

7、%s格式符　用来输出一个字符串

printf(“%s”,”china”);

8、%e格式符　以指数形式输出实数

printf(“%e”,123.456);

输出为：1.23456e+02。

4.2 输入函数getchar()和scanf()

一、字符输入函数getchar()

1、功能：从键盘输入一个字符

2、一般形式：

字符变量=getchar();

思考题:从键盘输入任一字符，如果该字符是小写字符，将其转换为大写字符输出，否则不变输出。

实训1

```
#include "stdio.h"
main( )
{char ch;
 ch=__getchar( )_____;
if (ch<='z'&&ch>='a') ch=ch-32;
 _putchar( )_____;}
```

二、 格式输入函数scanf( )

思考1：如何实现向整型变量a中输入数据？

scanf(“ %d ”,&a);

思考2：如何实现向实型变量b中输入数据？

scanf(“ %f ”,&b);

思考3：如何实现向整型变量a和实型变量b中输入数据？

scanf(“ %d %f ”,&a, &b);

1、scanf( )函数的一般形式

scanf(“格式控制”,地址表列);

2、scanf( )函数使用示例

```
int a, b, c;
scanf( “%d %d %d ”, &a, &b, &c);
scanf( “%d ,%d ,%d ”, &a, &b, &c);
scanf( “a=%d ,b=%d,c=%d ”, &a, &b, &c);
```

第五章 算法

程序＝数据结构＋算法

数据的类型
和组织形式

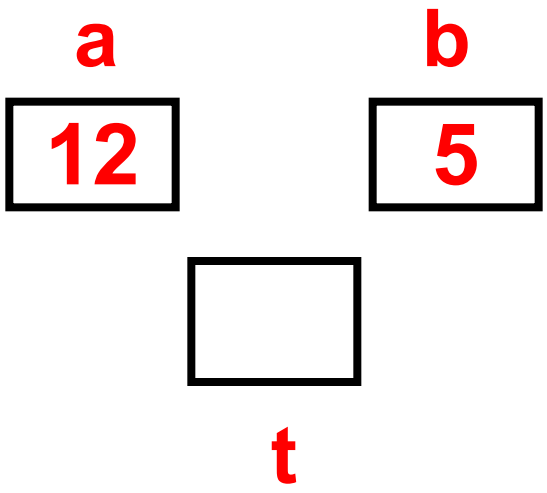
解决问题的
方法和步骤

描述算法的方法：自然语言、流程图

5.1 算法的概念和使用举例

一、三个典型实例

例1： 输入两个数a和b，要求实现将a和b中的数据进行交换。



算法描述

1、定义三个变量a、b、t

2、输入a和b的值

3、t=a；a=b；b=t；

4、输出a和b

```
main( )
{float a, b ,t;
 scanf (“%f %f ”,&a ,&b);
 t=a;
 a=b;
 b=t;
 printf ( “%f, %f ”.a. b);}
```

思考题：从键盘输入两个数，将这两个数按照从小到大的顺序输出来。比如输入5和1，则输出结果为1，5。

例2：从键盘输入三个整数，将这三个整数按照从大到小的顺序输出来。

比如：如果输入34、12、89；则输出为89、34、12

1、定义四个变量a、b、c、t；

2、输入a、b、c的值

3、如果b>a，则{t=b；b=a；a=t；}

4、如果c>a，则{t=c；c=a；a=t；}

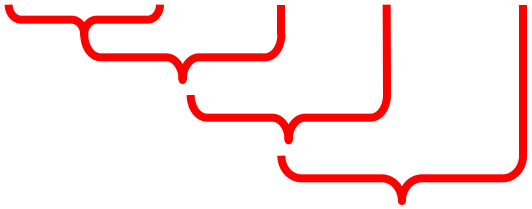
5、如果c>b，则{t=c；c=b；b=t；}

6、输出a、b、c的值

例3：编写一个程序，求1×3×5×……×99

假设用S表示乘积结果,用i表示被乘数

1×3×5×7×9×……×99



S=1

i=1

1、定义2个变量S和i

2、S=1；i=1；

3、S=S*i；i=i+2；

4、如果i<=99;返回3继续
否则,转到5

5、输出s

二、衡量算法好坏的两个因素

- 执行算法所占用的空间资源和时间资源
- 算法是否容易理解、调试和测试

三、算法的特点

- **有穷性**

  应该在有限的步骤内完成

- **确定性**

  明确而可以执行

- **必须有输出**

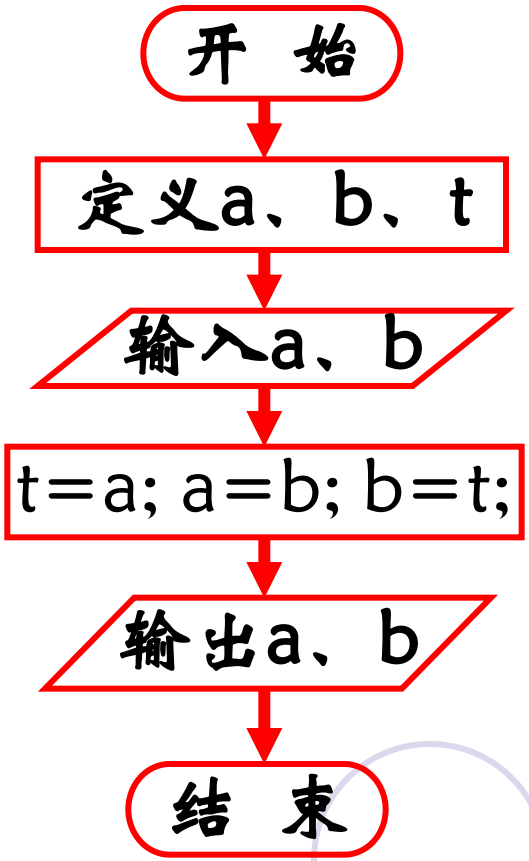
- **有效性**

  每一步都应该有效而可以执行

5.2 算法的流程图表示

一、算法的传统流程图表示

例1： 输入两个数a和b，要求实现将a和b中的数据进行交换。用传统流程图表示。



例2：从键盘输入三个整数，将这三个整数按照从大到小的顺序输出来。

1、定义四个变量a、b、c、t；

2、输入a、b、c的值

3、如果b>a，则{t=b；b=a；a=t；}

4、如果c>a，则{t=c；c=a；a=t；}

5、如果c>b，则{t=c；c=b；b=t；}

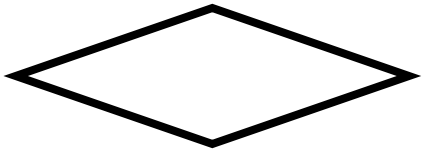
6、输出a、b、c的值

常用的流程图符号

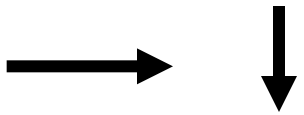
开始/结束框

处理框

输入/输出框



条件判断框



流程线

例3：编写一个程序，求1×3×5×……×99

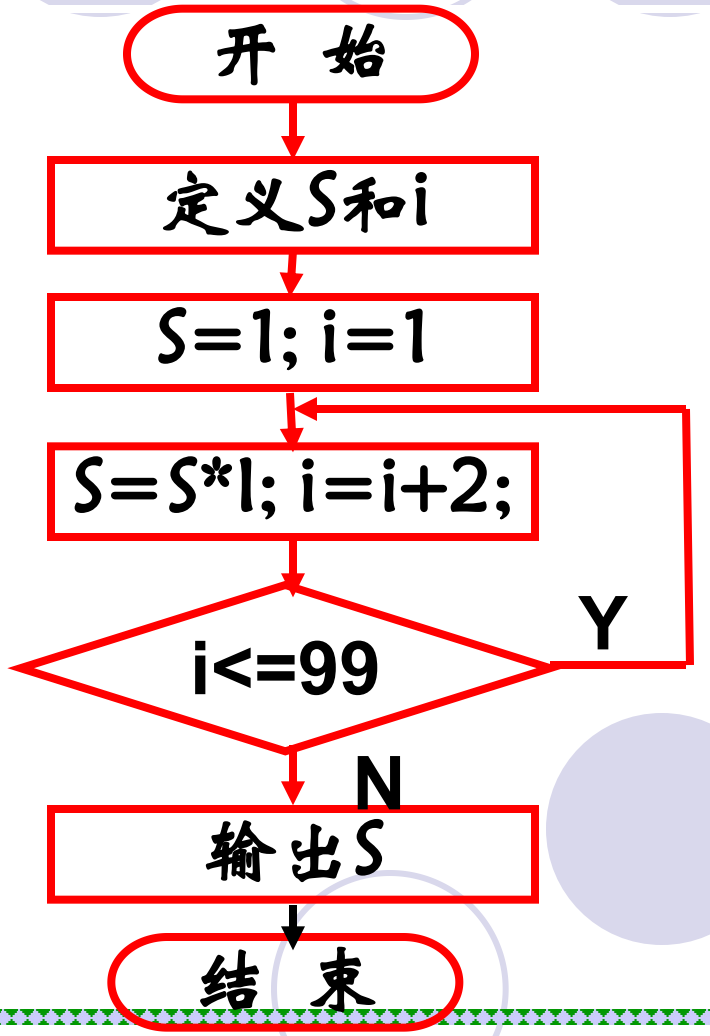
1、定义2个变量S和i

2、S=1；i=1；

3、S=S*i；i=i+2；

4、如果i<=99;返回3继续
否则,转到5

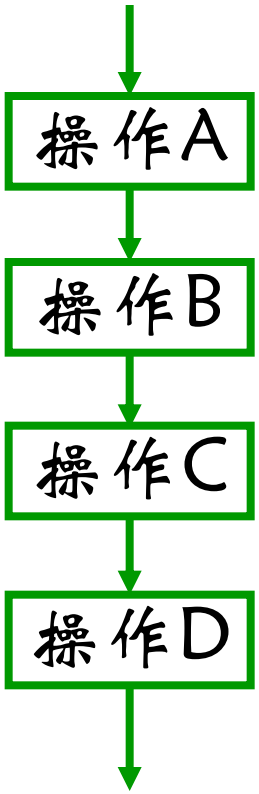
5、输出s



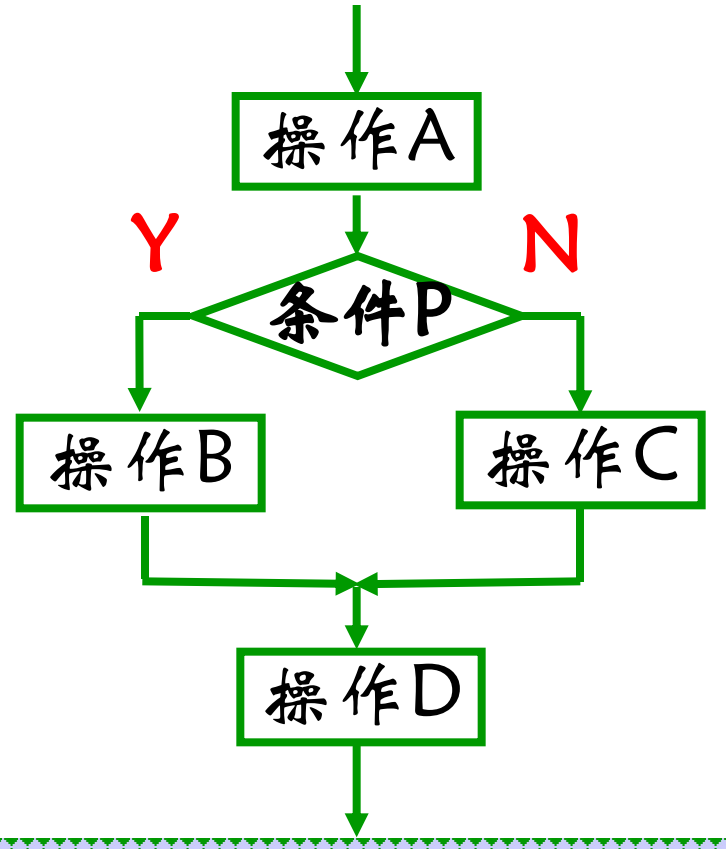
返 回
下一页

传统流程图的三种基本结构

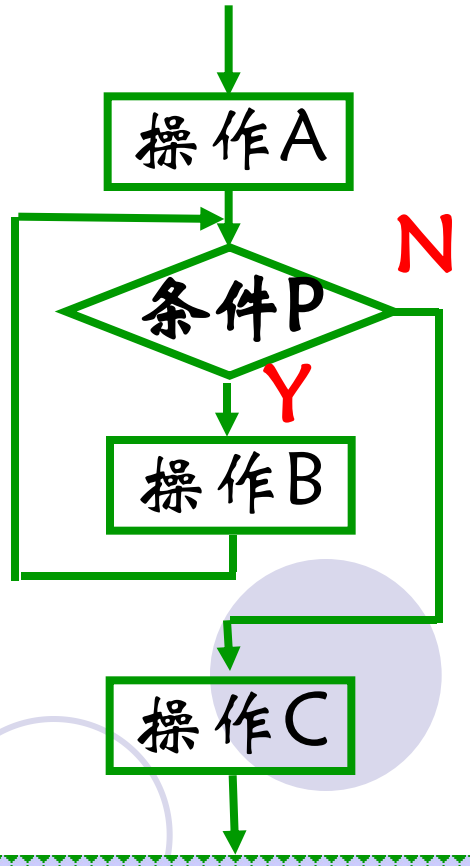
顺序结构



选择结构



循环结构



实训(请用传统流程图表示下列两个题目的算法)

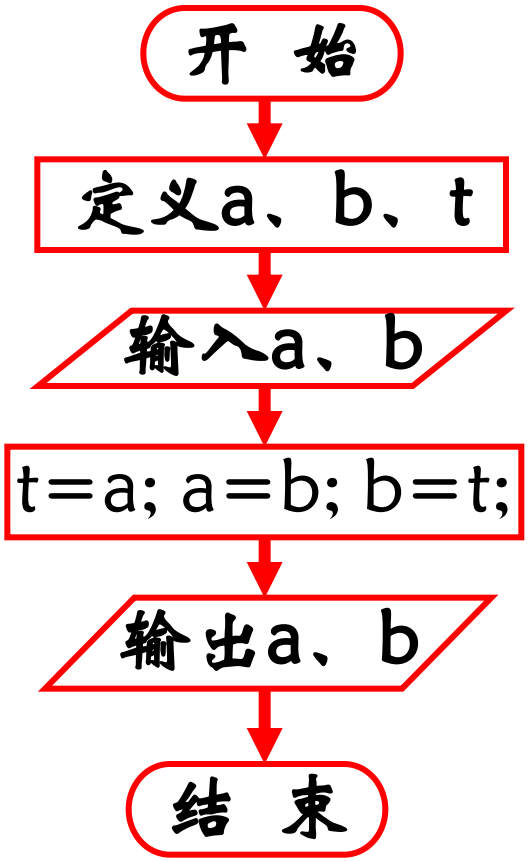
1、当从键盘输入一个字符时，实现判断该字符是大写字符还是小写字符。如果是大写字符，输出“UPWORD”，并将其转换为小写字符输出；如果是小写字符，则输出“DOWNWORD”，并将其转换为大写字符输出。

2、整数累乘：如果求

1×2×3×4×5×……×100的结果。

二、算法的N-S流程图表示

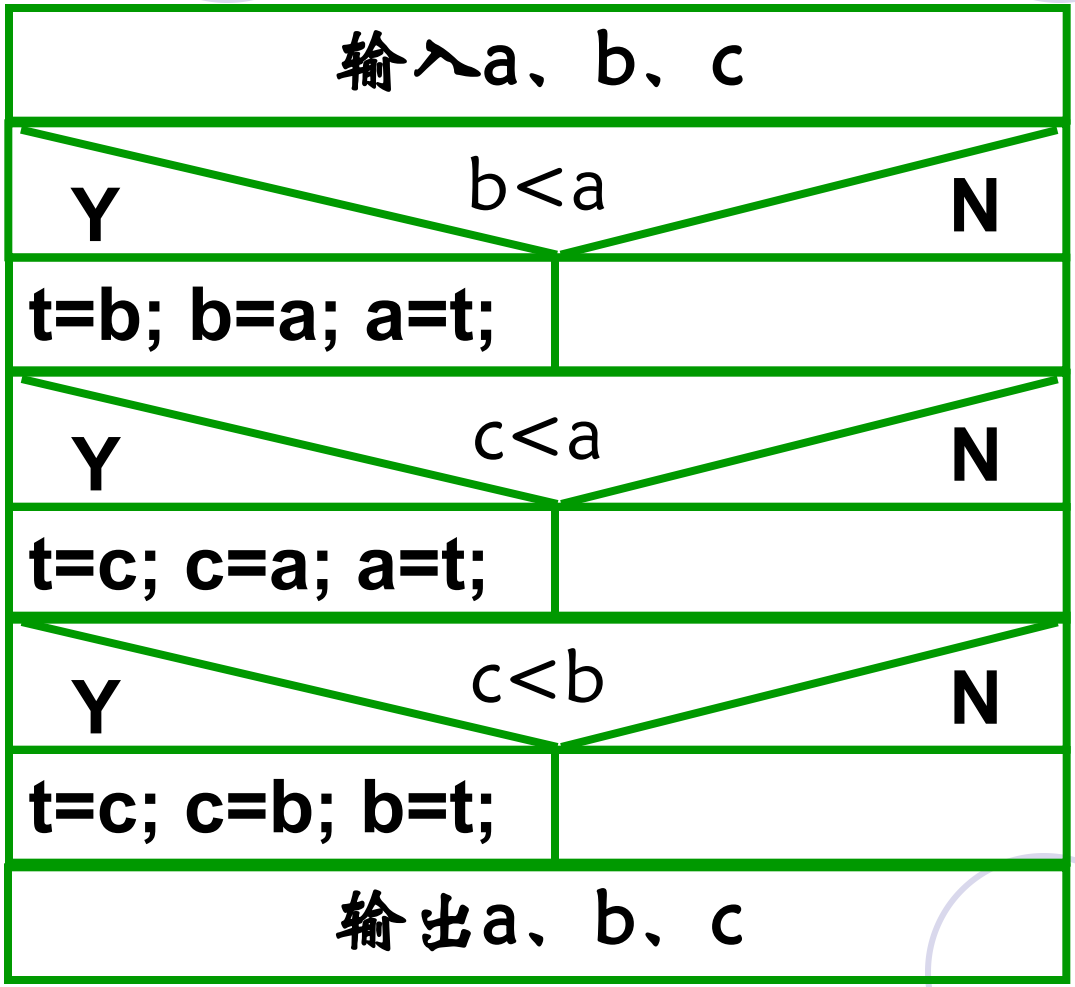
传统流程图



N-S流程图

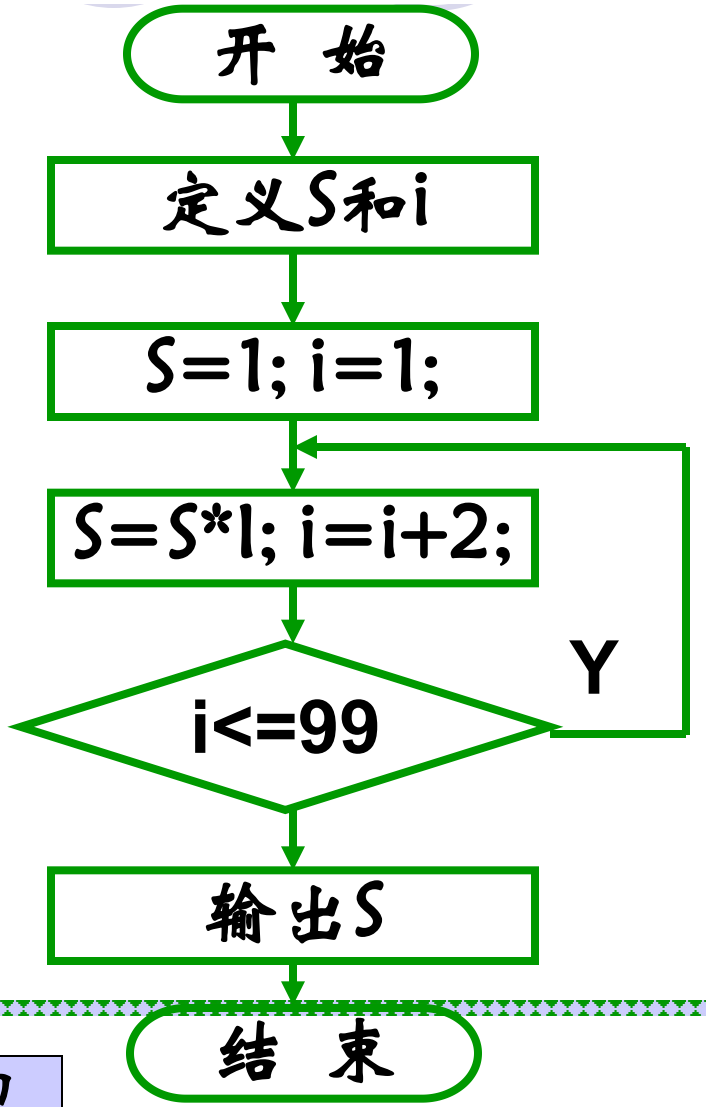
定义a、b、t
输入a、b
t=a; a=b; b=t;
输出a、b

从键盘输入三个整数，将这三个整数按照从大到小的顺序输出来。

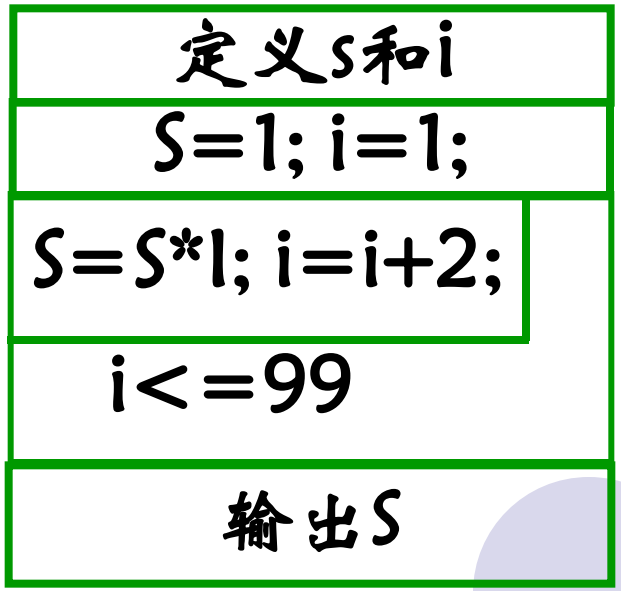


例3：编写一个程序，求1×3×5×……×99

传统流程图



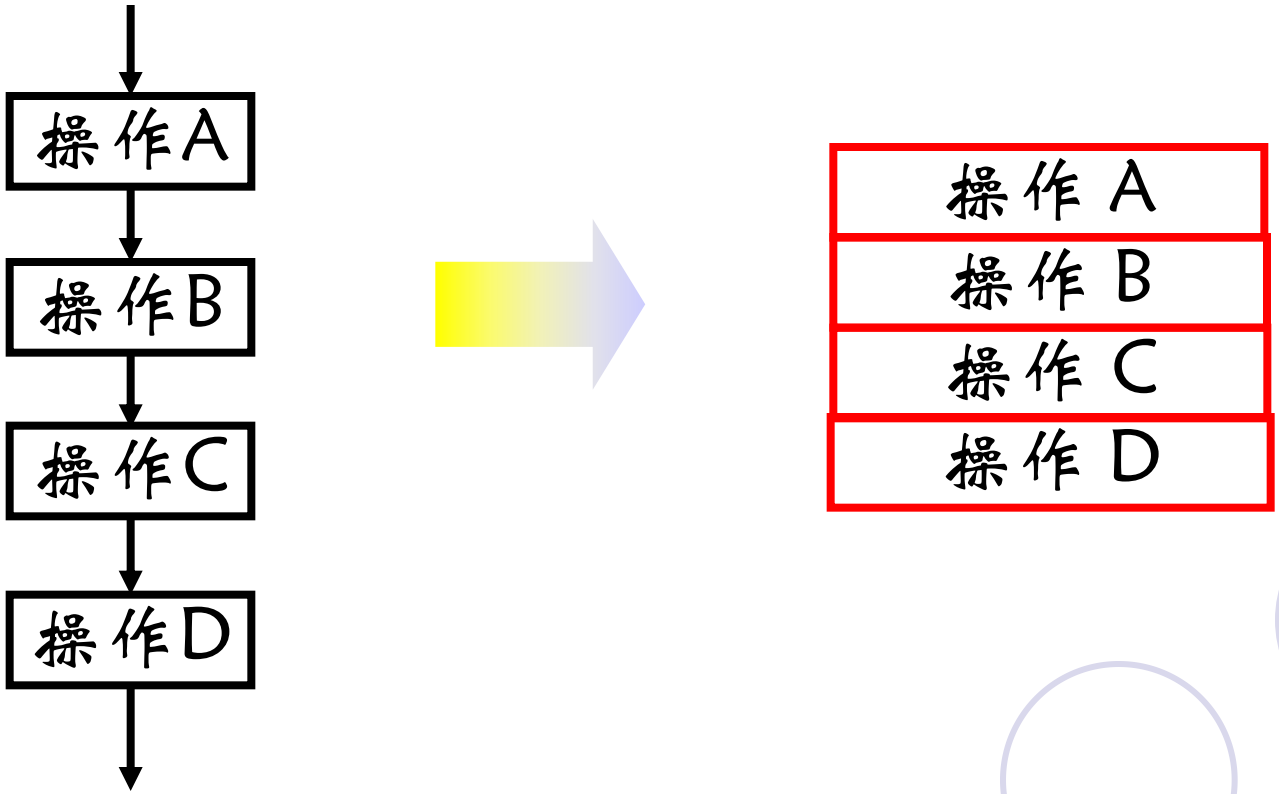
N-S流程图



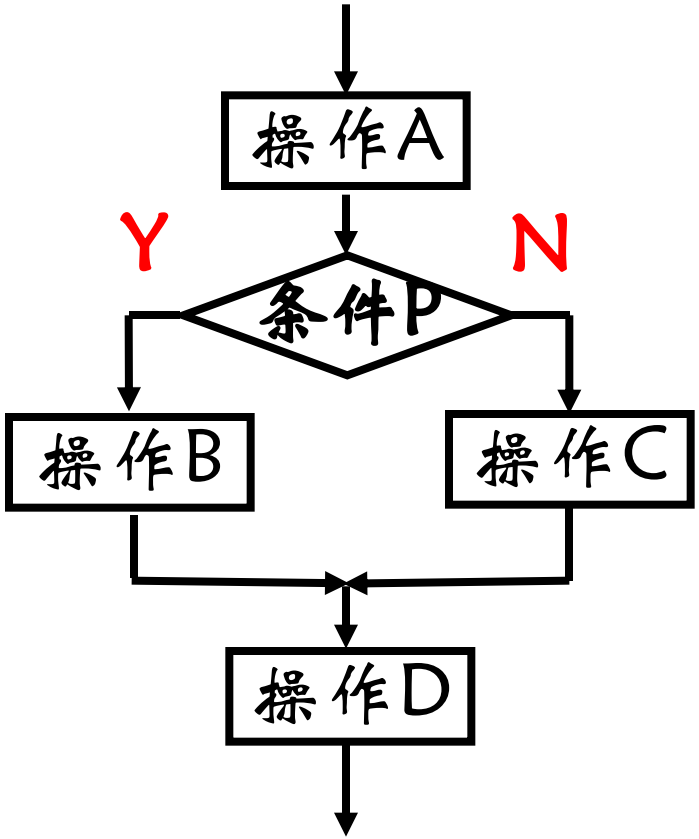
返 回
下一页

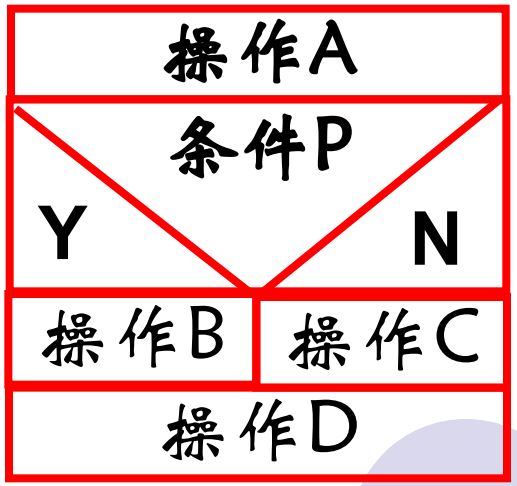
总结:传统流程图的三种基本结构

顺序结构

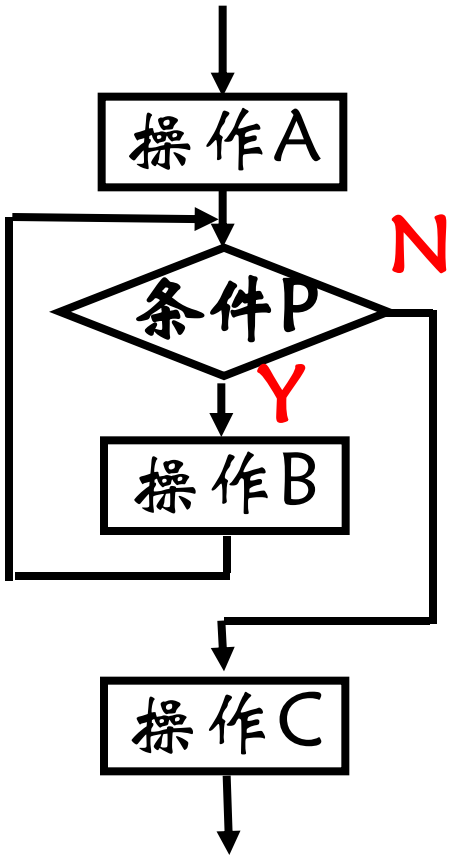


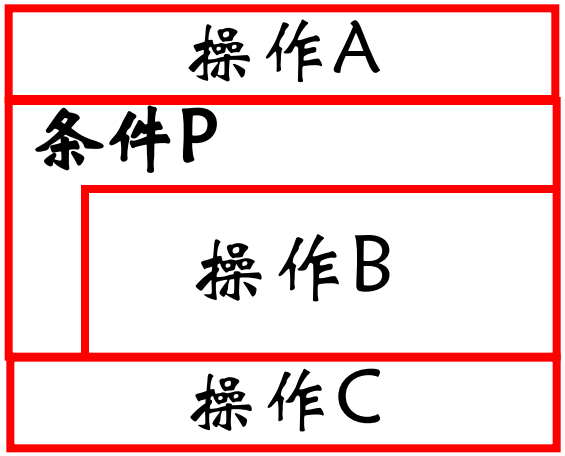
选择结构





循环结构





实训（请用N-S流程图表示下列两个题目的算法）

1、奇偶数识别：要求从键盘输入一个整数，判别其是奇数或偶数，如果是奇数，输出“It is an odd number!”；如果是偶数，输出“It is an even number!”。

2、整数累乘：如果求
1×2×3×4×5×……×100的结果。

综合实训

编写一个程序，要求从键盘输入一个数,能够输出该数的所有公约数。

第六章 结构化程序设计

一、顺序结构程序设计

实例导入：编写一个程序，实现这样的功能:要求从键盘输入任一八进制数字，可以实现将该数字转换为十进制数字，比如当输入八进制4520时，输出4520所对应的十进制数2384。

解析：

$(4520)_8 = 4*8*8*8 + 5*8*8 + 2*8 + 0$

千位数4: 4520/1000

百位数5: 4520%1000/100

十位数2: 4520%100/10

个位数0: 4520%10

求一个四位八进制数o_number所对应的十进制d_number，则需先求出o_number的千位、百位、十位、个位数，假如分别用a、b、c、d表示。

```
a= o_number/1000;
b= o_number%1000/100;
c= o_number%100/10;
d= o_number%10;

d_number=a*8*8*8+b*8*8+c*8+d;
```

算法描述

1、定义变量　int o_number1, d_number2, a, b, c, d;

2、输入o_number1L　scanf (“%d”, &number);

3、求出a、b、c、d：
```
a= o_number/1000;
b= o_number%1000/100;
c= o_number%100/10;
d= o_number%10;
```

4、求出d_number2：
```
d_number=a*8*8*8+b*8*8+c*8+d;
```

5、输出d_number2：　printf (“ %d”, d_number);

完整的程序

```
main( )
{int o_number ,d_number ,a ,b ,c , d;
 printf(“ please input a octal  number:\n”);
 scanf( "%d", &o_number);
 a=o_number/1000;
 b=o_number%1000/100;
 c=o_number%100/10;
 d=o_number%10;
 d_number=a*8*8*8+b*8*8+c*8+d;
 printf( “O: %d, D: %d", o_number, d_number);
 getch( );                          }
```

实训

编写一个体重测量仪：要求从键盘输入身高和体重后，能够计算出体重指数。

体重指数＝体重(kg)/(身高)^2;

算法描述：

1、定义3个变量：　float index, weight, height;

2、输入体重：　scanf(“%f”, &weight);

3、输入身高：　scanf(“%f”, &height);

4、计算指数：　index=weight/(height*height);

5、输出指数：　printf(“%f”, index);

完整程序

```
main( )
{float weight, height, index;
 printf(“ \n please input your weight:");
scanf ("%f", &weight);
 printf(“ \n please input your height:");
scanf ("%f", &height);
index=weight/(height*height);
printf ("The index of your weight is: %f", index);
 getch( );     }
```

总结：

1、顺序结构是最简单的一种结构，它按照操作执行的先后顺序来编写程序。

2、编程应该遵循：先根据功能要求设计出算法（用自然语言描述或用流程图描述），然后再根据算法编写程序。

3、程序应该结构清晰、界面美观、容易操作

二、选择结构程序设计

实训:体重测量仪改进版1

编写一个体重测量仪: 要求从键盘输入身高和体重后,能够计算出体重指数。

体重指数=体重(kg)/(身高)^2;

要求实现这样的判断:

如果体重指数<25, 属于正常;

否则属于肥胖.

如果index<25, 输出 “You are right!”

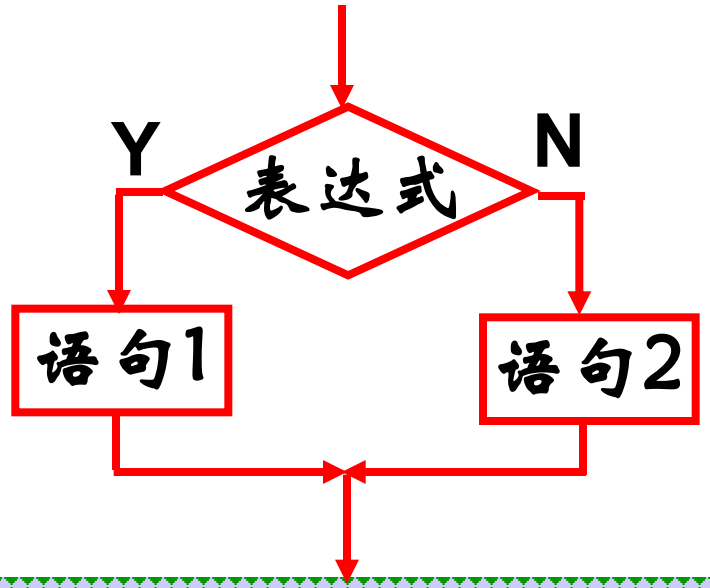
否则 输出 “You are fat!”

```
main( )
{float weight, height, index;
 printf(“ \n please input your weight:");
scanf ("%f", &weight);
 printf(“ \n please input your height:");
scanf ("%f", &height);
index=weight/(height*height);
printf ("The index of your weight is: %f", index);
if( index<25)  printf(“You are right!”);
else  printf (“You are fat!”);  }
```

一、if语句的使用

1、if语句的标准形式

if(表达式) 语句1
else 语句2



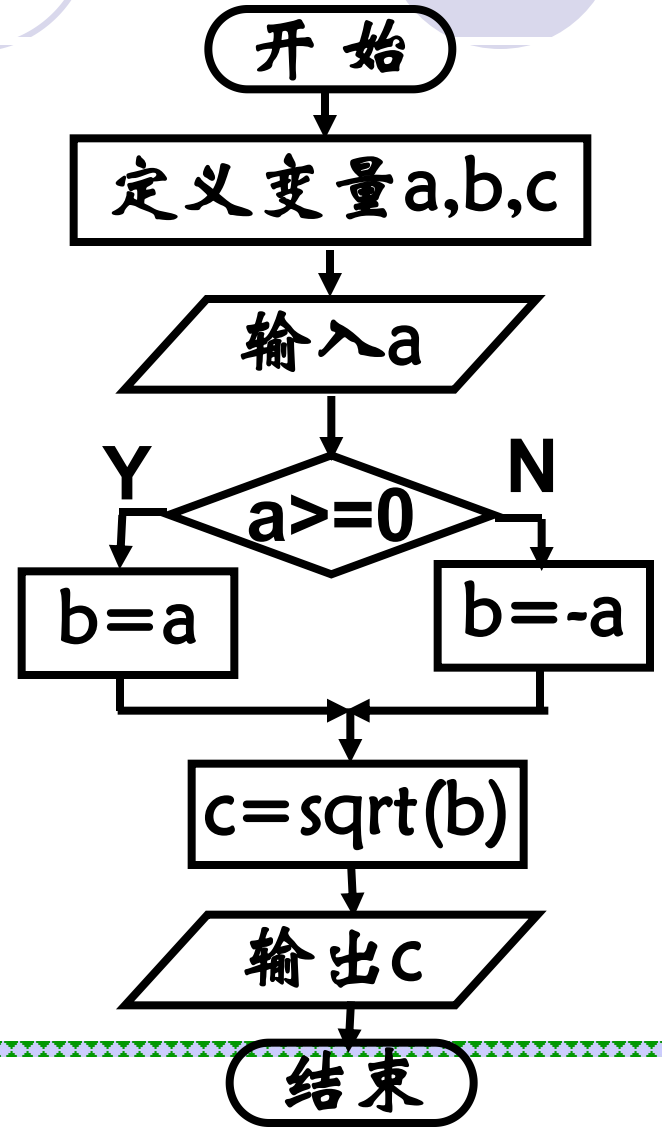
例： 如果工资salary大于1600元,则多余部分按15%征税,否则按5%征税.

```
If( salary>1600)
revenue=(salary-1600)*0.15;
else  revenue=(salary-1600)*0.05;
```

实训

从键盘输入一个数，求出该数绝对值所对应的开方

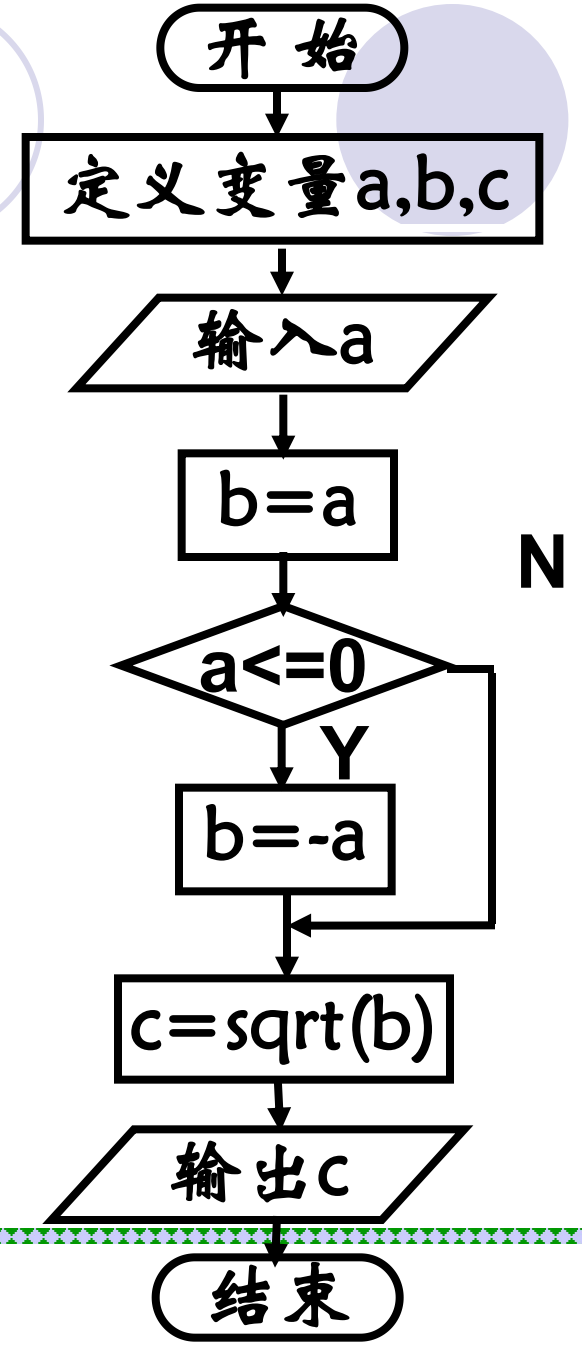
算法1



程序1

```
#include"math.h"
main( )
{float a,b,c;
 printf ("\n please input a number:\n");
 scanf("%f",&a);
 if(a>=0) b=a;
 else b=-a;
 c=sqrt(b);
 printf("The sqrt of %f is %f\n",a,c);
 getch( );}
```

算法2

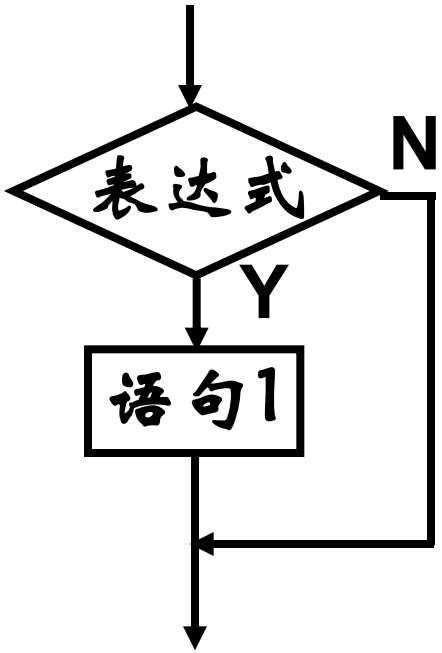


程序2

```
#include"math.h"
main()
{float a,b,c;
 printf("\n please input a number:\n");
 scanf ("%f", &a);
 b=a;
 if(a<=0) b=-a;
 c=sqrt(b);
 printf(“ The sqrt of %f is %f\n",a,c);
 getch( );}
```

2、if语句最简单的形式

if(表达式) 语句



思考：当输入45和32时，下面这个程序的输出结果是什么？

```
main( )
{int a,b,max;
scanf("%d%d",&a,&b);
max=a;
if(b>max) max=b;
printf("max=%d",max);}
```

体重测量仪改进版2

编写一个体重测量仪：要求从键盘输入身高和体重后，能够计算出体重指数。

体重指数＝体重(kg)/(身高)^2;

要求实现这样的判断：

偏瘦：体重指数<18

正常：18<＝体重指数<25;

微胖：25<＝体重指数<30;

较胖：30<＝体重指数<35;

肥胖:35<＝体重指数<40;

很胖：体重指数>＝40;

如果体重指数小于18，偏瘦

否则如果体重指数小于25，标准

否则如果体重指数小于30，微胖

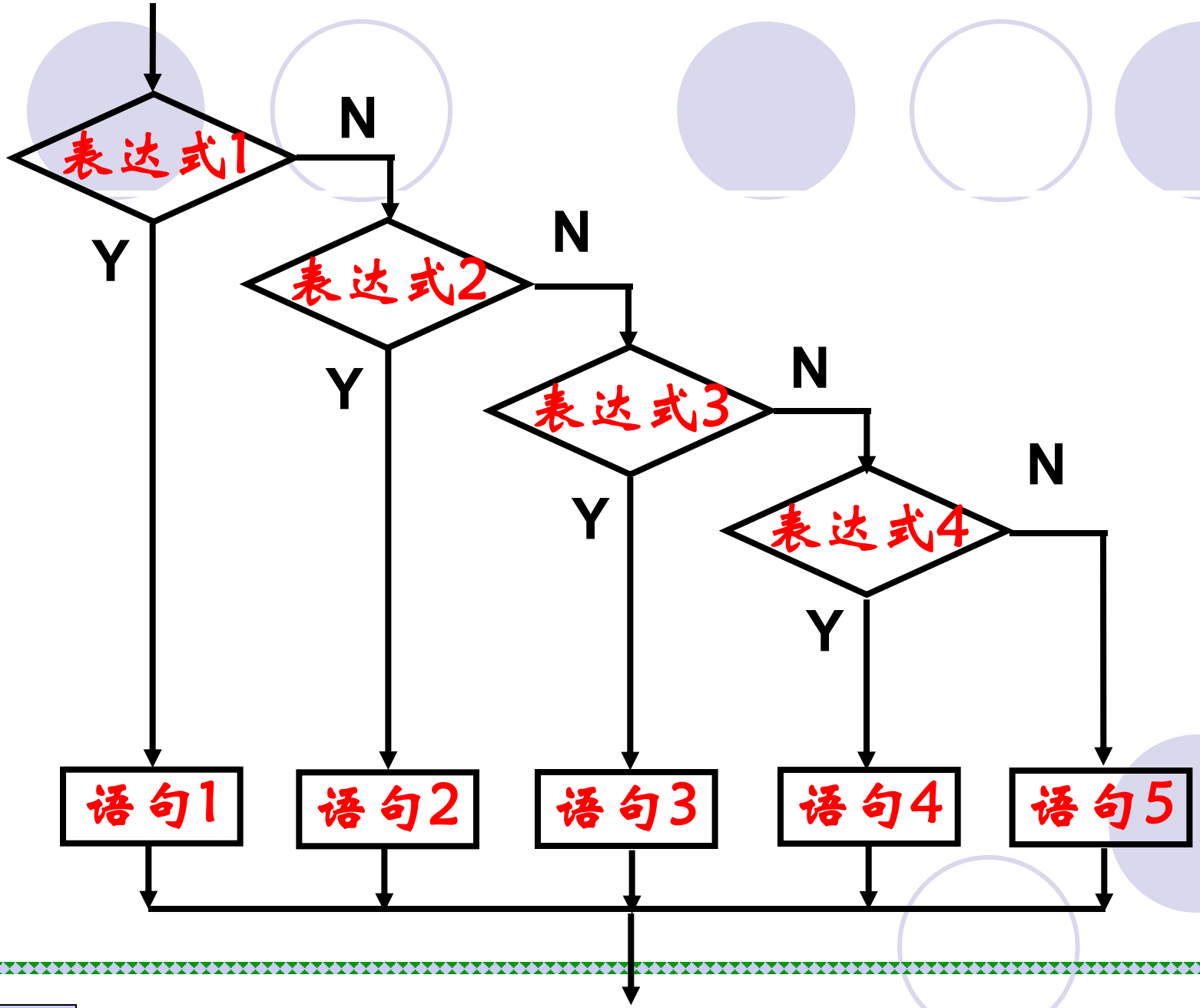
否则如果体重指数小于35，较胖

否则如果体重指数小于40,肥胖

否则很胖

3、if语句的第三种形式

```
if(表达式1) 语句1
    else if(表达式2) 语句2
        else if（表达式3）语句3
            else if（表达式4）语句4
                else 表达式5
```


表达式1
N
Y
表达式2
N
Y
表达式3
N
Y
表达式4
N
Y
语句1
语句2
语句3
语句4
语句5

实训

从键盘输入一个学生的分数，要求实现这样的判断功能：

如果分数大于100，输出”Input error!”

如果分数介于100到90之间，输出”Very Good!”

如果分数介于80到90只，输出”Good!”

如果分数介于70到80之间，输出”Middle”

如果分数介于60到70之间，输出”Pass”

如果分数小于60，输出”No Pass!”

算法1描述

1、定义一个变量score；
2、输入score的值；
3、如果score>100，输出“Input error！”
4、否则如果score>=90，输出“Very Good!”
5、否则如果score>=80，输出“Good!”
6、否则如果score>=70，输出“Middle!”
7、否则如果score>=60，输出“Pass!”
8、否则输出“No Pass!”
9、否则输出“Input error!

```
main( )
{float score;
 printf( "please input a number:\n");
 scanf ("%f", &score);
 if (score>100) printf ("Input error!");
 else if (score>=90)  printf ("very good!");
     else if (score>=80)  printf ("good!");
         else if (score>=70)  printf ("middle!");
             else if (score>=60)  printf ("pass!");
```

```
else if (score>=0) printf (“No pass!”);
        else printf (“Input error!”);}
```

总结 if语句的三种基本形式

1、最基本的形式

```
if(表达式) 语句1
else 语句2
```

2、最简单的形式

```
if(表达式) 语句
```

3、层次形式

```
if(表达式1) 语句1
  else if(表达式2) 语句2
    else if（表达式3）语句3
      else if（表达式4）语句4
        else 表达式5
```

实训

假如从键盘输入67，下面这个程序的结果是什么，如果输入89，90，101呢？

```
main( )
{float score;
 scanf ("%f",&score);
 if(score>=60)
   if(score>=90) printf ("A");
   else printf ("B");
 else
   if(score>=0) printf ("C");
   else printf ("Input error!");}
```

二、if语句的嵌套

一般使用形式

```
if(表达式1)
   if(表达式2) 语句1
   else 语句2
else
   if(表达式3)语句3
   else 语句4
```

思考题

假如从键盘输入5，下面程序的输出结果是什么？

```
main( )
{int x ,y;
 scanf ("%d", &x);
 if (x>=0)
   if (x>10) y=x*x-5;
   else if (x>5) y=x*x*x+2;
        else y=x;
 else
   if (x<-10) y=x*x+5;
   else y=x*x+11;
 printf ("%d", y);}→
```

实训

使用if语句来编写如下的程序：

某单位的工资是这样计算的：

工资＝基本工资+工龄工资；

工龄为1年：　　工龄工资＝基本工资*0.1；

工龄为2—3年：　工龄工资＝基本工资*0.25；

工龄为4—5年：　工龄工资＝基本工资*0.5；

工龄为6—8年：　工龄工资＝基本工资*0.8；

工龄大于8年的：工龄工资＝基本工资*工龄*0.1；

基本工资为1500元。

要求从键盘输入教师的工龄，可以根据工龄计算该教师的基本工资。

算法描述：

1、定义两个实型变量：工资salary、工龄工资a_salary; 定义一个整型变量:工龄age;

2、输入工龄age;

3、如果age==1; a_salary=1500*0.1;

4、如果2<=age<=3; a_salary=1500*0.25;

5、如果4<=age<=5; a_salary=1500*0. 5;

6、如果6<=age<=8; a_salary=1500*0.8;

7、如果8<=age; a_salary=1500*age*0.1;

8、salary=1500+a_salary;　　9、输出salary;

```
main()
{float salary, a_salary;
 int age;
 scanf ("%d", &age);
 if (age==1) a_salary=1500*0.1;
 else if (age>=2&&age<=3) a_salary=1500*0.25;
      else if (age>=4&&age<=5)
                        a_salary=1500*0.5;
           else if (age>=6&&age<=8)
                        a_salary=1500*0.8;
                else if (age>8)
                        a_salary=1500*age*0.1;
 salary=1500+a_salary;
 printf("%.1f",salary);}↵
```

```
main()
{float salary ,a_salary;
 int age;
 scanf ("%d“ ,&age);
 switch (age)
 {case 1: a_salary=1500*0.1;   break;
  case 2: a_salary=1500*0.25; break;
  case 3: a_salary=1500*0.25; break;
  case 4: a_salary=1500*0.5;   break;
  case 5: a_salary=1500*0.5;   break;
  case 6: a_salary=1500*0.8;   break;
  case 7: a_salary=1500*0.8;   break;
  case 8: a_salary=1500*0.8;   break;
  default:a_salary=1500*age*0.1;}
 salary=1500+a_salary;
  printf("salary:%.1f",salary);;}
```

复习

判断一个学生成绩的“优、良、中、差”：

假如从键盘输入的字母是‘A’，输出评语“You are Excellent！”；

如果输入‘B’，输出评语“You are well！”；

如果输入‘C’，输出评语“You are passing！”；

如果输入‘D’，输出评语“You are not passing！”。

如果输入的不是这四个字母，则出现提示“Input error！”

请编写一个程序，实现该判断功能。

算法描述

1、定义一个字符型变量score；

2、输入字符score；

3、如果score==‘A’，输出“Excellent！”

4、如果score==‘B’，输出“well！”

5、如果score==‘C’，输出“pass！”

6、如果score==‘D’，输出“Not pass！”

7、如果以上都不是，则输出“Input error！”

If语句实现

```
main( )
{ char score;
  scanf( “%c”, &score);
  if(score==‘A’)  printf(“excellent!”);
    else if(score==‘B)   printf(“well!”);
        else if(score==‘C’)  printf(“pass!”);
            else if(score==‘D’)  printf(“No pass”);
                else printf(“Input error!”);}
```

switch语句实现

```
main( )
{ char score;
  scanf( “%c”, &score);
  switch(score)
  { case ‘A’: printf(“Excellent!”); break;
    case ‘B’: printf(“Well!”);      break;
    case ‘C’: printf(“Pass!”);      break;
    case ‘D’: printf(“No pass!”);   break;
    default: printf(“Input error!”);}}
```

三、switch语句的使用

```
switch(表达式)
 { case 常量表达式: 语句1
   case 常量表达式: 语句2
   ……
   case 常量表达式: 语句n
   default: 语句 n+1 }
```

四、条件运算符的使用

程序段1

```
if( a>b)  max=a;
else  max=b;
```

程序段2

```
if( x>=0)  y=x;
else  y=-X;
```

```
max=(a>b) ? a: b
```

```
y=(X>=0) ?x : - x;
```

一般使用形式： 表达式1？表达式2：表达式3

优先级仅仅比赋值运算符和逗号运算符高，自右至左的结合性.

综合实训

某企业发放的年终奖金根据职工该年的积分计算。积分等于或低于0分的，奖金为0；积分在1到19分之间的，奖金为积分数乘以100；积分在20到29之间的，奖金为积分数乘以150；积分在30到39之间的，奖金为积分数乘以200；积分在40到49分之间的，奖金为积分数乘以250；积分在50分以上的，奖金都为积分数乘以300。编写一程序，从键盘输入积分数，可以求出该职工的年终奖。

分别用if和switch语句来实现

三、循环结构程序设计

所谓**循环**，就是对一段程序重复执行多次

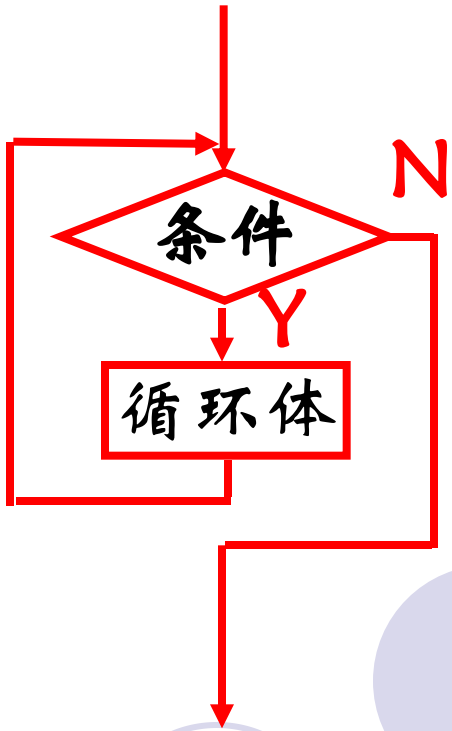
循环体语句：	程序中需要被重复执行的部分
循环的初始条件：	i=0;
循环的执行条件：	i<50;

实现循环的三种方法：
- while语句
- do_ while语句
- for语句

while语句

while语句的一般形式:

```
while（循环的执行条件）
{
循环体语句
  }
```

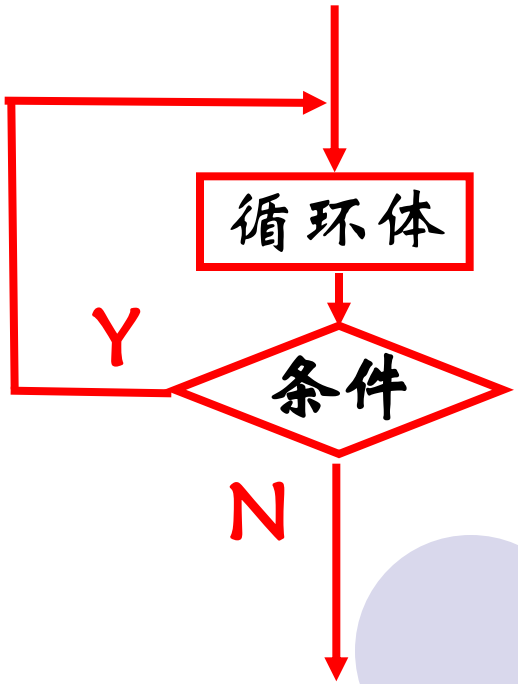


```
main( )
{ float E_score, P_score, M_score, F_score;
   int   i=0;
 while (i<50)
{printf("please input the score of exam:");
 scanf ("%f", &E_score);
 printf ("\n please input the score of performance:");
 scanf ("%f", &P_score);
 printf ("\n please input the score of :");
 scanf ("%f", &M_score);
 F_score=E_score*0.4+P_score*0.3+M_score*0.3;
 printf("%.1f",F_score);
  i=i+1;      }        }
```

do _ while语句

do_while语句的一般形式是：

```
do
{
  循环体语句
 }while(循环的执行条件);
```



```
#include ” math.h ”
 main( )
{ int  number, i;
 scanf (“%d” ,&number);
  i=2;
  do
{If (number % i==0)  printf(“%6d”,i);
  i=i+1; } while (i< sqrt (number));
 }
```

for语句

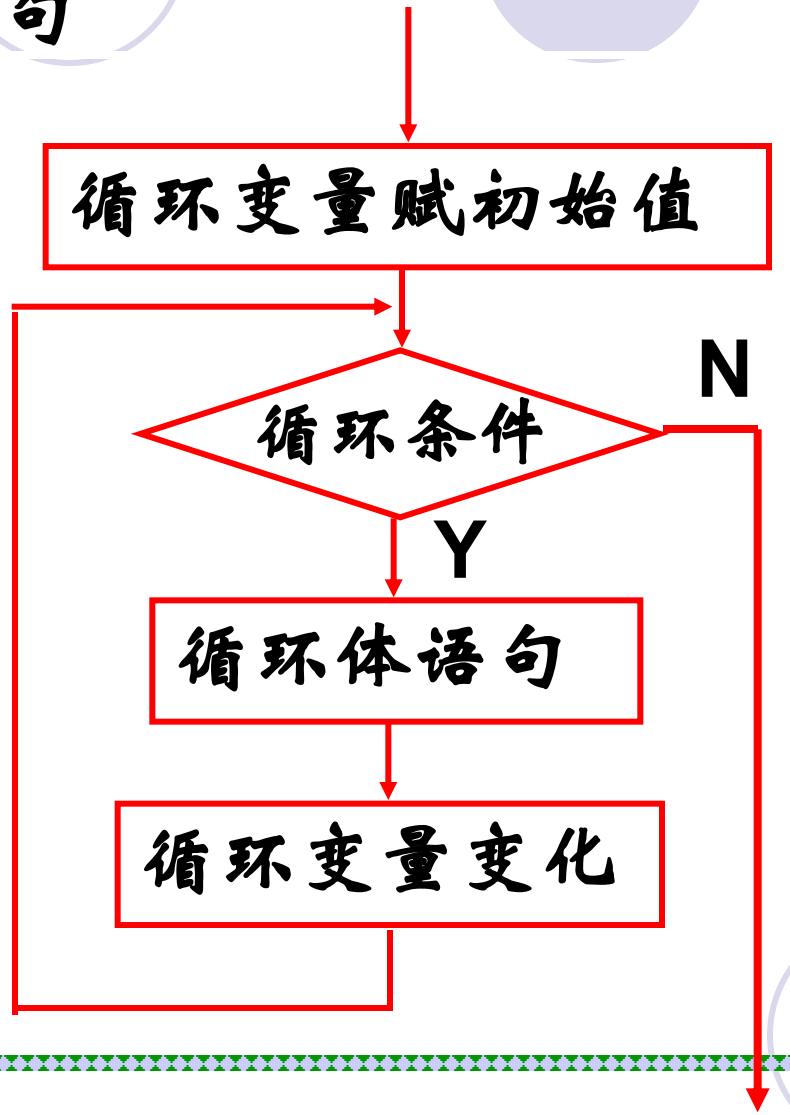
for语句的一般形式是:

for(循环变量初始值;循环的条件;循环变量变化)

循环体语句

```
#include ” math.h ”
main( )
{ int  number, i;
scanf (“%d” ,&number);
for (i=2;i<sqrt (number) ;i++)
if( number %i==0 ) printf(“%6d”,i);}
```

for(循环变量初始值;循环的条件;循环变量变化)
循环体语句



综合实训

编写一个程序，输出1到200间所有可以被18整除的数

1、如果1%18==0，输出1；

2、如果2%18==0，输出2；

3、如果3%18==0，输出3；

4、如果4%18==0，输出4；

5、如果5%18==0，输出5；

……

200、如果200%18==0，输出200；

```
main( )
{ int i;
 for (i=1;  i<=200;  i++)
 If(i%18==0)  printf (“%d”, i); }
```

变形1:

```
main( )
{ int i;
  i=1;
 for(   ;                    )
      i<=200;          i++
 If(i%18==0)  printf (“%d”, i); }
```

结论1:表达式1可以省略,但是分号不能省.

变形2:

```
main( )
{ int i;
  for ( i=1;        ;      i++ )
{ If(i%18==0)  printf (“%d”, i);
  If (i>200) break; }}
```

结论2：表达式2也可以省略，如果省略，分号不能省,而且需要在循环体中加使循环能够结束的语句

变形3：

```
main( )
{ int i;
  for ( i=1;  i<=200;        )
{If(i%18==0)  printf (“%d”, i);
    i++; }

  }
```

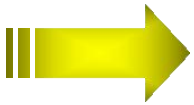
结论3:表达式3也可以省略,如果省略,分号不能省,应该在循环体中加入使循环趋于结束的语句

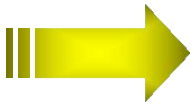
变形4：

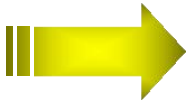
```
main( )
{ int i;
  i=1;
  for( ; ; ; )
{ If(i%18==0)  printf (“%d”, i);
  i++;
  If (i>200) break; }
  }
```

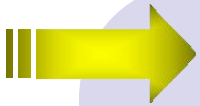
结论4：三个表达式都可以省略,但是分号不能省.

循环结构程序设计综合实训

实例1：兔子繁殖问题 

实例2：素数的判断 

实例3：百钱百鸡 

实例4：趣味思考：C语言绘图 

兔子繁殖问题

题目： 一个饲养场引进一对刚出生的新品种兔子，这对兔子从出生的第三个月开始，每月新生一对兔子，新生兔子从第三个月开始也如此繁殖。如果所有的兔子都不死，那么一年末，三年末，该饲养场共有多少对兔子？

解析

Fibonacci数列

1　1　2　3　5　8　13　21

1、f1=1; f2=1;

2、printf("%d%d",f1,f2);

3、f1=f1+f2;　f2=f1+f2;

4、printf("%d%d",f1,f2);

循环体

5、f1=f1+f2;　f2=f1+f2;

迭代：不断以新值代替旧值的操作

```
main( )
{ int i;
  long f1,f2;
  f1=1;  f2=1;
   i=1;                          /*循环控制变量初始化*/
  while (i<=18)                  /*循环条件*/
{ printf( “%ld,%ld” ,f1,f2);     }
  f1=f1+f2;   f2=f1+f2;          /*循环体*/
   i=i+1;   }
}
```

do……while实现

```
main( )
{int i;
 long f1,f2;
 f1=1; f2=1;
 i=1;                                  /*循环控制变量初始化*/
 do
 {printf("%ld,%ld",f1,f2);
  f1=f1+f2;   f2=f1+f2;
  i=i+1;}                                    /*循环体*/
  while (i<=18); }          /*循环条件*/
```

for实现

```
main( )
{int i;
 long f1,f2;
 f1=1; f2=1;
 for (i=1;i<=18;i++)
 {printf("%ld,%ld,",f1,f2);
  f1=f1+f2; f2=f1+f2;}}→
```

/*循环体*/

小结:三种循环的比较

- 3种循环都可以用来处理同一个问题
- While和do……while循环，只在while后面指定循环条件,在循环体中应包含使循环趋于结束的语句(如i++或i=i+1等).而for在表达式3中包含了使循环趋于结束的语句.
- 在while和do……while循环中,循环变量的初始化应在while和do……while之前完成,而在for循环中,表达式1实现该功能.

返回

返回

素数的判断

素数是指除了1和它本身之外没有其它因子的自然数。

编一程序，要求能够实现判断任一整数是否为素数。

程序

```
main(   )
{ long number;   int i;
  scanf (“%ld”, &number);
  i=2;
  while(i<number)
  If(number%i==0){printf (“No!”); continue; }
  else i++;
  If (i>=number)  printf (“Yes!”); }
```

小结:break和continue的比较

1、break的作用是跳出循环体，转而执行循环体下面的语句。

2、break一般和if语句配合使用，而且只能用在switch和循环结构中。

3、continue的作用是结束本次循环，跳过循环体中下面尚未执行的语句，接着进行下一次是否执行循环的判定。

4、continue一般和if语句配合使用

思考:素数的判断改进版

输出1到1000间所有的素数。

```
main()
{ long number;
  int i=2;
  for(number=1;number<=1000;number++)
  { while(i<number)
    if(number%i==0) break;
    else i++;
    if(i>=number)
   printf("%5d",number);   } }
```

百钱百鸡问题求解编程

我国古代数学家张丘建在《算经》中出了一道题“鸡翁一，值钱五；鸡母一，值钱三；鸡雏三，值钱一。百钱买百鸡，问鸡翁、鸡母、鸡雏各几何？”。

解析：

用x表示鸡翁，y表示鸡母，z表示鸡雏。

$$x + y + z = 100$$

$$5x + 3y + z/3 = 100$$

```
x=100;
    y=100;
        z=0: if(x+y+z==100&&15*x+9*y+z==300)
                输出x  y  z
        z=1:  if(x+y+z==100&&15*x+9*y+z==300)
                输出x  y  z
        z=2:
        .....
        z=100;
```

穷举：对所有可能的情况进行逐一检验，从而找到符合条件的解.

程序

```
main( )
{int x,y,z;
for(x=0;x<=100;x++)
   for(y=0;y<=100;y++)
       for(z=0;z<=100;z++)
 if(x+y+z==100&&15*x+9*y+z==300)
 printf("公鸡:%d,母鸡:%d,小鸡:%d\n",x,y,z);}
```

小结:关于循环嵌套问题

- 循环的嵌套:指的是在一个循环体内还包含一个完成的循环结构.
- 三种循环结构可以自身嵌套,也可以相互嵌套

返回

返回

第七章 模块化程序设计

模 块： 一个具有独立功能的程序段

模块化程序设计：

按适当的原则把一个情况复杂、规模较大的程序系统划分为一个个较小的、功能相关而又相对独立的模块，这样的程序设计方法称为**模块化程序设计**。

模块化程序设计的优点

- 复杂系统化大为小，化繁为简
- 便于维护
- 提高系统设计效率(便于多人并行开发)

在C语言中模块是由函数来实现的。

```
main( )
{long s;
 int a ,b;
 scanf ("%d%d", &a, &b);
 s=power (a,b);
 printf ("%ld", s);}          /*调用函数power*/

long power (x, y)             /*函数power*/
int x, y;
{long sum=1;
 int i;
 for (i=0;i<y; i++)
 sum= sum*x;
 return (sum);}
```

一、关于函数的几个概念

函数： 能够完成一定功能的程序段

函数的调用： 函数的使用

主调函数： 调用其他函数的函数

被调函数： 被其他函数调用的函数

程序的执行总是从main函数开始，和函数的位置无关.

二、函数的分类

1、从用户使用的角度来划分
- 用户自定义函数
- 系统函数

clrscr( )　sqrt( )
sin( )　cos( )
putpixel( )

2、从函数的形式来划分
- 无参函数
- 有参函数

三、函数的定义

1、无参函数的定义

常见的系统无参函数有： getch( );
clrscr( ); exit( ) ;

无参函数定义的一般形式为：

类型标志符 函数名（）
{声明部分；
执行部分；}

实训　定义一个函数line，能够用来绘制一条直线。

```
void line( )
{int i, driver=VGA;
 int mode=VGAHI;
 initgraph (&driver, &mode,"");
 for (i=0;i<=100;i++)
    putpixel(i,100,1);
}
```

2、有参函数的定义

常见的系统有参函数有：sin( ), cos( ). Sqrt( ), setcolor( ); putpixel( )

有参函数定义的一般形式为：

类型标志符 函数名（形式参数表列）

{声明部分；

执行部分；}

实训 定义一个函数line，能够用来绘制一条从指定端点出发的定长直线。。

```
#include"stdio.h"
#include"graphics.h"
void line(x,y,l,z)
int x,y,l,z;
{ int i, driver=VGA;
  int mode=VGAHI;
  initgraph(&driver,&mode,"");
  for(i=x;i<=l;i++)
  putpixel(i,y,z);
```

综合实训　编写一个程序，能够实现当从键盘输入x和y时，求出x^y的结果。

解析：$x^y=\underbrace{x*x*x*x*x*x*x\ldots\ldots*x}_{y个x}$

1、定义变量s、x、y、i；

2、输入x和y；

3、变量初始化：s=1；i=0；

4、当i<y时　s=s*x;i=i+1;

5、输出s;

```
main( )
{ long s, x, y; int  i;
  scanf("%ld%ld",&x,&y);
  s=1;i=0;
  while(i<y)
  { s=s*x; i=i+1;}
    printf("%ld",s); }
```

```
power (long x, long y)
{ long s; int i;

 s=1;i=0;
 while (i<y)
 {s=s*x; i=i+1;}
 printf( "%ld“ ,s);}→
main( )
{power(234, 456);

power (a+2,b*2);}
```

超链接
超链接
返 回

四、函数的参数

1、形式参数(形参)

被调函数中的参数

2、实际参数（实参）

主调函数传送给被调函数的实际参数

小结:形式参数和实际参数之间的关系

- 形参没有固定的值,在实际运算时,形参接收来自实参的值进行运算处理.
- 实参和形参在进行值传递时是一一对应关系，所以其个数应该相同，对应的类型也应该相同
- 形式参数和实际参数之间的值传递是单向传递
- 只有子函数被调用时,才为形参分配内存单元,用完即释放.
- 实参不但可以是变量，也可以是常量，还可以是表达式,但是一定要有确定的值.

五、函数的返回值

1、return的作用

将被调函数中得到的结果带回到主调函数中.

2、return的使用

return 表达式;

3、返回值的类型

和函数类型一致;如果不一致,以函数类型为准

如果函数没有类型，则用void表示无类型

六、函数的调用

1、无参函数的调用

一般形式：函数名（）；

例　clrscr( );　getch( );

2、有参函数的调用

一般形式：函数名（实参表列）；

例　sum=power(34,4)+power(3,98);

七、函数的声明

1、使用输入输出函数getchar( )　putchar( )时

#include"stdio.h"

2、使用数学函数sin( ) cos( ) sqrt( )等时

#include"math.h"

3、使用绘图函数putpixel( ) arc( ) circle( )等时

#include"graphics.h"

4、那么自定义函数该如何声明呢?

小结：

1、所谓声明，就是把函数的类型、函数的名字、函数形参的类型及个数、顺序通知编译系统，便于在调用时进行检查。又称函数原型

2、函数的声明也可以不写形参的名字，只写出类型即可

3、函数原型的一般形式：

函数类型 函数名（参数类型1，参数类型2...，参数类型n）

八、函数使用实训

实例1：编写程序可以实现求x的正弦值、余弦值、正切值、余切值。

要求界面如下：

****

1：sin（）　　2：cos（）

3：tan（）　　4：ctan（）

****

请选择您要进行的计算：

```
printf("**************************\n");
printf("  1:sin()    2:cos()\n");
printf("  3:tan()    4:ctan()\n");
printf("**************************\n");
printf("请选择你要进行何种计算:\n");
scanf(“%d”, &s);
printf(“请输入您要计算的数字:\n");
scanf(“%f”, &x);
switch(s)
{case 1: result=sin(x); break;
 case 2:result=cos(x); break;
 case 3:result=tan(x); break;
 case 4:result=1/tan(x);break;
 default: printf(“Input error!”);}
 printf(“result=%f”,result);}
```

实例2 编写一函数，求n!

解析：

0！ =1

1！ =1

2！ =1*2=1!*2

3！ =1*2*3=2!*3

4！ =1*2*3*4=3!*4

……

n!=(n-1)!*n

$$n! = \begin{cases} 1 & (n=0, 1) \\ n*(n-1)! & (n>1) \end{cases}$$

```
long f(int n)
{long s;
 if(n==0||n==1) s=1;
 else s=f(n-1)*n;
 return(s);}
 main( )
 {int n;
  long y;
  printf("please input a number:\n");
  scanf("%d",&n);
 y=f(n);
 printf("%d!=%ld\n",n,y);}
```

结论：在调用一个函数的过程中，又出现直接或间接调用该函数本身，称为递归调用。

7.2 变量的作用域

作用域: 变量在什么范围内有效

根据变量的**作用域**划分

- 局部变量：在函数内定义，只在本函数范围内有效的变量
- 全局变量：在函数外定义，并不从属于某个函数,可以被文件中的其他函数使用。

一、局部变量

实例1：下面这个程序有错吗？如果没错，当输入12，34，56时，输出是什么；如果有错，为什么呢？

```
main( )
{int a,b,c;
 scanf("%d,%d,%d",&a,&b,&c);
 if(b<a)
 {int t;
  t=b;b=a;a=t;}
 if(c<a)
 {t=c;c=a;a=t;}
 printf("%d",a);}→
```

二、全局变量

实例2：下面这个程序的输出结果是什么？

进入程序

```
int max;
max_min(int a,int b)
{int min;
 max=min=a;
 if(b>max) max=b;
 if(b<min) min=b;
 return(min);}
main( )
{int m,n,minmize;
 m=520;n=888;
 minmize=max_min(m,n);
printf("max=%d,min=%d", max,minmize);}
```

max 888

a 520 b 888

min 520

m 520 n 888 minmize 520

关于全局变量的总结

- 当在函数外定义了一个全局变量后，定义点之后的函数都可以使用该变量.
- 当全局变量的值在一个函数中改变后，另一个函数使用的就是改变后的值.
- 从定义开始到程序结束，全局变量自始至终占用存储空间.
- 全局变量的使用使函数间的联系加强,与程序设计”低耦合”的原则相违背,所以很少使用.

思考

```
int a=5;
main( )
{int a=3,b=4,sum;
 sum=a+b;
 printf("%d",sum);}
```

总结:在局部变量的范围内，局部变量将起作用，外部变量将被屏蔽掉而失去作用。

```
int max,min;
max_min(int n1,int n2,int n3)
{int max;
 if(n2>n1) {max=n2;min=n1;}
 else {max=n1;min=n2;}
 if(n3>max) max=n3;
 if(n3<min) min=n3;
}
main( )
{int num1,num2,num3;
 scanf("%d%d%d",&num1,&num2,&num3);
 max_min(num1,num2,num3);
 printf("max=%d,min=%d",max,min);
}
```

7.3 变量的存储类别

根据变量的**存储类别**划分

- **静态存储方式**：在整个程序运行期间分配**固定**的存储单元，直到整个程序运行结束后才释放变量的存储单元
- **动态存储方式**：在程序运行期间根据需要**动态**的分配存储空间，使用完即释放。

全局变量属于**静态存储方式**，而局部变量一般属于**动态存储方式**。

一、动态存储方式的声明

1、用auto声明动态局部变量

例：auto int a；

将变量a定义为整型局部自动变量

int a；

特点：函数调用结束，该种变量即被释放，属于动态存储方式。

2、用register声明寄存器变量

例：register int a；

将变量a定义为
整型寄存器变量

特点： 加快运算速度，函数运行完即消失、使用有限

二、静态存储方式的声明

1、用static声明静态局部变量

例

```
int incent(int x)
{int z;
 static int y=0;
 y=y+1;
 z=x+y;
 return(z);}
main( )
{int i;
 for(i=0;i<5;i++)
 printf("%d",incent(i));}
```

思考：比较两种程序结果的不同。

静态局部变量的特点：

（1）静态局部变量每次函数调用结束后能够保留调用结束后的值不变，留待下一次调用。

（2）静态局部变量只限于被本函数使用，不能被其他函数使用

（3）静态局部变量属于静态存储类别，在整个程序运行期间都不释放；而动态存储变量属于动态存储类别，函数调用结束即释放。

（4）在定义静态局部变量时，如果不赋初值，系统自动赋值为0或空字符（对字符型变量）；而相应的情况下，系统会给其赋一个不确定的值。

2、用static声明外部变量

static定义的变量叫静态外部变量，限制其只能被本文件所使用，而不能被其他文件所使用。

例

时俊.c

```
static  int number;
        sort(  )
        {.....}
```

刘刚.c

```
static  int number;
        delete(  )
        {.....}
```

3、用extern声明外部变量

张林.c

```
Insert(   )
{ extern a;
  ……}
int a;
query( )
{     }
```

李敏.c

```
extern a;      声明语句
print(   )
{……}
modify( )
{     }
```

小结

- 按照作用范围分类，变量分为局部变量和外部变量。
- 按照存储类别分类，变量分为静态存储方式和动态存储方式。

7.4 函数的作用范围

根据函数的作用范围划分

- 内部函数：只能被本文件所使用
- 外部函数：可以被其他文件使用

一、内部函数

xietianwen.c

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

xubosheng.c

```
#include”xietianwen.c”
main( )
{int a,b,s;
 scanf(“%d%d”,&a,&b);
 s=sum(a,b);
 printf(“%d”,s);
}
```

用**static** 定义的函数只能被本文件中的其他函数所调用，不能被其他文件所使用，称为内部函数。

二、外部函数

xietianwen.c

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

xubosheng.c

```
#include"xietianwen.c"
 main( )
 {int a,b,s;
   scanf("%d%d",&a,&b);
   s=sum(a,b);
   printf("%d",s);
 }
```

用**extern** 定义的函数可以被其他文件所使用，称为外部函数，**extern**可以省略。

```
#define PI  3.1415926
main( )
{float d,s,v,r;
 printf("请输入圆的半径：");
 scanf("%f",&r);
 d=2*      PI      *r;
 s=        PI      *r*r;
 v=3.0/4*      PI      *r*r*r;
 printf("d= 2*3.1415926*r=%f",d);
 printf("s=3.1415926*r*r=%f",s);
 printf("v=3.0/4*3.1415926*r*r*r=%f",v);
}
```

宏定义：用#define定义一个标志符来表示一个字符串或数字。

PI称为**宏名**，展开过程称为**宏替换**。

宏定义的注意事项：

1、宏名，即标志符名一般用大写字母表示，但也可以用小写字母表示。

2、在进行宏替换时,就是用标志符代替字符时,并不做正确性检查,只是做简单的替换,如上例中将3.1415926写成了3.1115926,系统照样替换,不会出现错误提示。

3、宏定义的末尾没有分号。如果有,将作为字符串的一个部分处理。

4、对程序中用双撇号括起来的字符串内的字符,即使与宏名相同,也不进行替换,如例7.12中最后一行输出语句printf内的PI并没有被替换成3.1415926。

5、对于宏定义中的宏名,系统不会为其分配内存空间。

7.5 宏定义

一、不带参数宏定义

一般形式：#define 标志符 字符串

例：#define PR printf
#define N “\n”

```
main( )
{PR(“You are welcomeN”);
 }
```

宏定义的终止：#undef 宏名

#undef PI 表示从该点之后，PI就不代表3.1415926了

二、带参数宏定义

例1：

```
#define S(a,b) a+b
main( )
{int sum;
 sum=S(5,6);
 printf(“%d”,sum);
 }
```

例2：

```
#define MAX(a,b) a>b?a:b
main( )
{int x,y,max;
 x=45;y=56;
 max=MAX(x,y);
 printf("max=%d\n",max);}
```

```
max=MAX(12+13,14+9);
```

第八章 数组

1、数组的概念

具有**相同数据类型**的若干个**有序数据**的集合

2、数组的分类

- 一维数组
- 二维数组
- 字符数组

8.1 一维数组

实例导入

从键盘输入10个学生的成绩，要求能够求出总分和平均分。

解 析：

1、定义一个一维数组score[10]；

2、定义总分sum，平均分average；

3、向一维数组score[10]中输入数据；

4、求出sum和average；

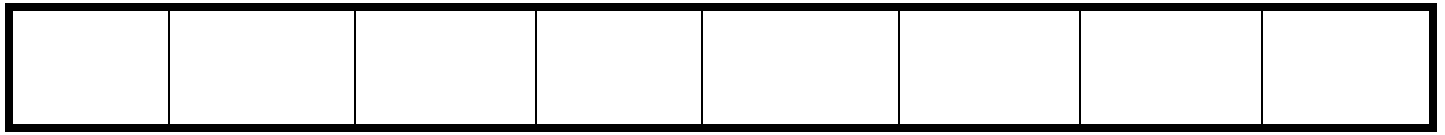
5、输出sum和average.

一、一维数组的定义

一维数组定义的一般形式：

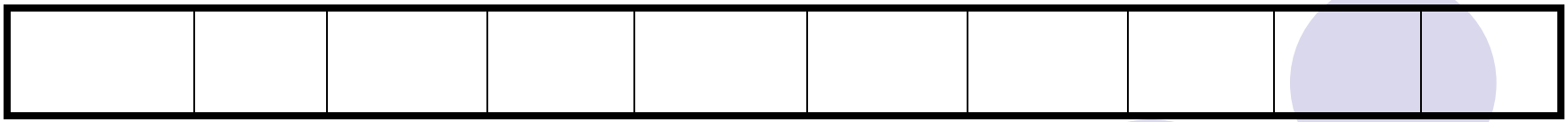
类型说明符 数组名[常量表达式]；

例：　float s[8];



S[0] S[1] S[2] S[3] S[4] S[5] S[6] S[7]

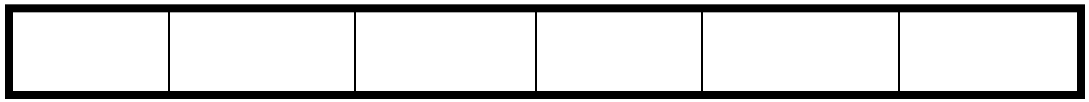
long t[10];



t[0] t[1] t[2] t[3] t[4] t[5] t[6] t[7] t[8] t[9]

二、一维数组的初始化

例：　float s[6];



S[0] S[1] S[2] S[3] S[4] S[5]

如何给数组s赋一组初始值{65.5, 89.2, 90, 89.8, 45.5, 60}

1、依次赋值。s[0]=65.5; s[1]=89.2; s[2]=90;
s[3]=89.8; s[4]=45.5; s[5]=60;

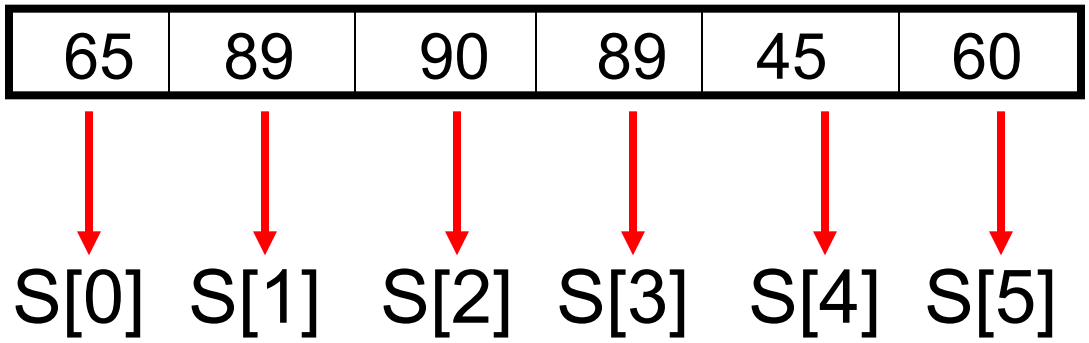
2、全体赋值。s[6]= {65.5, 89.2, 90, 89.8, 45.5, 60}

3、定义时赋值s[ ]= {65.5, 89.2, 90, 89.8, 45.5, 60}

4、省略型赋值s[6]= {65.5, 89.2, 0, 0, 0, 0}
s[6]= {65.5, 89.2}

三、一维数组元素的引用

例： float s[6];



一维数组元素引用的一般形式：

数组名[下标]

四、一维数组实训

实训1：有8个整数{89, 88, 67, 23,45,54,0,0},要求编程求出这8个整数的和、平均值、最大值和最小值。

```
main( )
{float num[8] ,sum, max, min, average;
 num[8]={89, 88, 67, 23,45,54,0,0};
 sum=0;
 for(i=0;i<8;i++)
 sum=sum+num[i];
 average=sum/8;
 max=num[0];
 min=num[0];
 for(i=0;i<8;i++)
 If(num[i]>max)  max=num[i];
 If(num[i]<min) min=num[i];
 printf("sum=%f, average=%.1f, max=%.1f,
 min=%.1f", sum, average, max, min);}}
```

例1：一维数组的输入和输出

定义一个整型数组arr[10]，向其中输入10个数据并依次输出

```
main( )
{ int arr[10]; int i;
 for(i=0;i<10;i++)
 scanf("%d",&arr[i]);
 printf("\n该一维数组为" :);
 for(i=0;i<10;i++)
 printf("%5d",arr[i]);
}
```

循环实现一维数组的输入

循环实现一维数组的输出

例2：一维数组的排序

定义一个整型数组arr[10]，向其中输入10个数据并按照从小到大的顺序依次输出

冒泡排序

```
main( )
{ int arr[10]; int i;
  for(i=0;i<10;i++)
  scanf("%d",&arr[i]);
  for(i=0;i<10;i++)
  for(j=i+1;j<10;j++)
  if(arr[j]<arr[i]) {t=arr[j];arr[j]=arr[i];arr[i]=t;}
  printf("排序后的数组为:\n");
  for(i=0;i<10;i++)
  printf("%5d",arr[i]);
 }
```

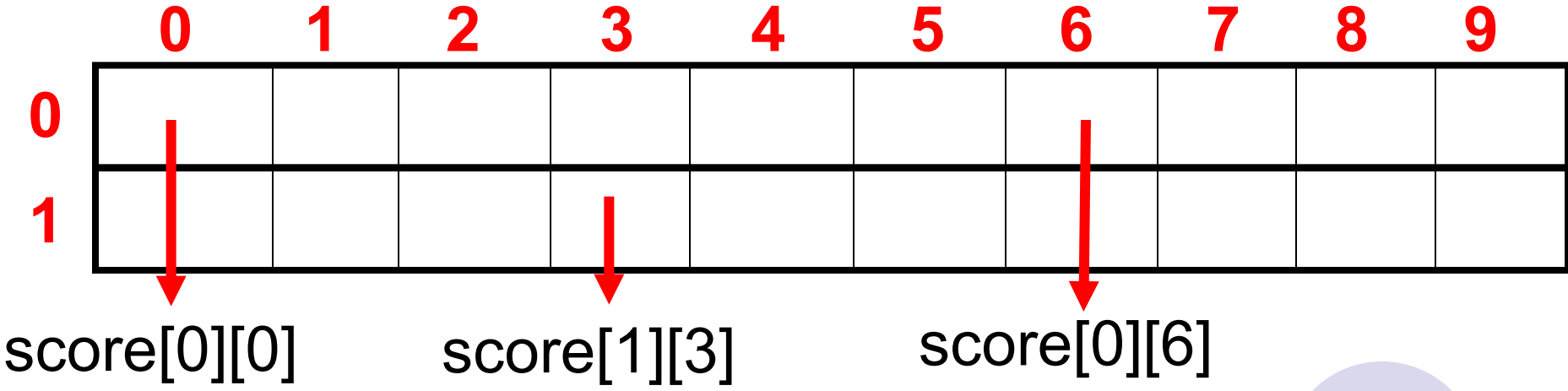
插入排序

```
main( )
{ int arr[10]; int i;
  for(i=0;i<10;i++)
  scanf("%d",&arr[i]);
  for(i=0;i<10;i++)
  for(j=i+1;j<10;j++)
  if(arr[j]<arr[i]) {t=arr[j];arr[j]=arr[i];arr[i]=t;}
  printf("排序后的数组为:\n");
  for(i=0;i<10;i++)
  printf("%5d",arr[i]);
 }
```

8.2 二维数组

一、二维数组的定义

float score[2][10];



二维数组定义的一般形式:

类型说明符　数组名[常量表达式][常量表达式];

二、二维数组元素的初始化

例：现有一个二维数组s[3][4]，如何实现将以下数据赋给s:

1	1	2	0
5	8	0	0
34	55	89	0

1、按行赋值

```
score[  ][4]={{1,1,2,0},{5,8,0,0},{34,55,89,0}};
```

```
score[ ][4]={{1,1,2},{5,8},{34,55,89}};
```

2、全部赋值

score[ ][4]={1,1,2,0,5,8,0,0,34,55,89,0};

score[3][4]={1,1,2,0,5,8,0,0,34,55,89};

3、数组元素的引用

二维数组元素的引用形式为：

数组名[下标][小标]；

二维数组使用实训

实训1: 定义一个5行6列的二维整型数组s,向其中输入数据,并按行输出.

解 析:

1、定义一个二维数组s;

2、向其中输入5行6列30个数据

3、输出这5行6列30个数据

```
main( )
{ int s[5][6];
  int i, j;
  for(i=0;i<5;i++)
     for(j=0;j<6;j++)
  scanf ("%d",&s[i][j]);
  for(i=0;i<5;i++)
  { for(j=0;j<6;j++)
  printf("%4d",s[i][j]);
   printf("\n");  }
   }→
```

printf("请输入第%d行数据:\n",i);

实训2：数据的查找

现有一个二维数组num，其初始值如下

num[3][6]={{3,4,11,12,14,19},{21,20,14,18,30,23},{43,2,23,45,42,1}};

要求从键盘输入一个数据number，在数组num中进行查找,假如该数存在,就输出该数和其所在的行数和列数;假如不存在，就输出”No this number!”

```
main( )
{float num[3][6], number;
int  i, j;
scanf(“%d”, &number);
for(i=0;i<3;i++)
   for(j=0;j<6;j++)
lf(num[i][j]==number)
   {printf ("%d, %d, %d\n ",number, i, j);
     break;}
lf(i>3) printf(“No this number!”);
}
```

实训3

有一个二维数组，编写一程序，求该二维数组两对角线的和，并找出该二维数组中的最大值和最小值及其所在位置。

如该二维数组为：

34　12　15　10
78　9　0　87
80　19　100　21
45　23　35　43

则程序运行结果为：和为 260，最大值为100，位于第2行第2列。最小值为0,位于第1行第2列。

思考1： 编写一个程序,实现这样的功能,从键盘输入一个二维数组S[3][4]后，可以实现将该二维数组中的每个数组元素的值加5。

比如当输入的二维数组如下：

1 2 3 4
5 6 7 8
9 10 11 12

则转换后输出的二维数组为：

6 7 8 9
10 11 12 13
14 15 16 17

算法描述：

1、定义二维数组S[3][4]；

2、向二维数组中输入数值

3、将转换前的二维数组输出来

4、使每个数组元素的值增5

5、将转换后的二维数组输出来

```
main( )
{ int s[3][4];  int i,j;
 for(i=0;i<3;i++)
 for(j=0;j<4;j++)
 scanf("%d",&s[i][j]);
 for(i=0;i<3;i++)
 {for(j=0;j<4;j++)
  printf("%5d",s[i])[j]);
  printf("\n");}
 for(i=0;i<3;i++)
 for(j=0;j<4;j++)
 s[i][j]=s[i][j]+5;
 for(i=0;i<3;i++)
 {for(j=0;j<4;j++)
  printf("%5d",s[i][j]);
  printf("\n");} }
```

```
main( )
{ int s[3][4];  int i,j;
 for(i=0;i<3;i++)
 for(j=0;j<4;j++)
 scanf("%d",&s[i][j]);
 for(i=0;i<3;i++)
 {for(j=0;j<4;j++)
  printf("%5d",s[i])[j]);
  s[i][j]=s[i][j]+5;
  printf("\n");}
 for(i=0;i<3;i++)
 {for(j=0;j<4;j++)
  printf("%5d",s[i][j]);
  printf("\n");}
   }
```

思考2：

编写一个程序，实现这样的功能，从键盘输入一串字符信息后，可以实现将该串字符信息按照一定的方式转换为密文。

比如当输入“Meet”

如果转换密钥为5，则字符串中的每个字符都转换为其后面的第5个字符.

转换后的字符串为:Rjjy

编程实现当输入一段字符后，并输入密钥，则按照密钥将所输入的字符串转化为一段密文并输出.

8.3 字符数组

一、字符数组的定义

char mingwen[500], miwen[500];

字符数组定义的一般形式：

char 数组名 [常量表达式] ；

二、字符数组元素的引用

第i个元素的引用形式：

数组名 [i] ；

三、字符数组的初始化

1、逐个字符赋值

例：char name[20];

name[20]={‘Z’,’h’,’a’,’n’,’g’,’L’,’e’,’i’‘};

2、以字符串方式赋值

例：char name[20];

name[20]={“ZhangLei”};

name[20]=“ZhangLei”;

注意：存储时，在末尾自动填‘\0’作为字符串结束标志.

四、字符数组的输入和输出

例：现在需要向字符数组score[10]中输入内容,并将内容输出来.该如何实现.

1、输入

1）逐个字符输入

```
for(i=0;i<10;i++)
scanf("%c",&score[i]);
```

2）一次性输入

```
scanf("%s",score);
```

2、输出

1）逐个字符输出

```
for(i=0;i<10;i++)
printf("%c",score[i]);
```

2）一次性输出

```
printf("%s",score);
```

思考：从键盘输入一段字符，并能够统计出这段字符的个数。

```
main( )
{char str[200], len, i;
 printf(“请输入一段英文字符：”);
 scanf(“%s”, str);          <⟷>   gets(str);
 len=0;
 i=0;
 while(str[i]!=‘\0’)
 len++;
 printf(“该段字符所包含的字符个数为:%d”,len );
}
```

五、常用的字符串处理函数

1、puts函数：字符串输出函数

一般使用形式：puts（字符数组）；

例：char name[ ]="Liu'an";
puts(name);

2、gets函数：字符串输入函数

一般使用形式：gets（字符数组）；

例：char name[15];
gets(name);

3、strcat函数

一般使用形式：strcat(数组1,数组2)；

```
char surname[15]="Zhang ";
char firstname[  ]="Lin";

strcat(surname, firstname);
```

连接后的结果

Z	h	a	n	g		L	i	n	\0	\0	\0	\0	\0	\0

4、strcpy函数

一般使用形式：strcpy(数组1,数组2)；

```
char surname[15]="Zhang ";
char firstname[  ]="Lin";

strcpy(surname, firstname);
```

连接后的结果

L	i	n	\0	g	\0		\0	\0	\0	\0	\0	\0	\0	\0

5、strcmp函数

一般使用形式：strcmp(数组1,数组2)；

```
char surname[15]="Zhang ";
char firstname[  ]="Lin";

strcmp(surname, firstname);
```

比较的结果大于0

6、strlen函数

一般使用形式：strlen(数组)；

```
char surname[15]=“Zhang ”;
strlen(surname);
```

测量的是字符串所占的实际长度,为5.

7、strlwr函数

一般使用形式：strlwr(数组1)；

用于把字符串中的大写形式转换为小写

8、strupr函数

一般使用形式：strupr(数组1)；

用于把字符串中的小写形式转换为大写

注意：当使用到字符串处理函数时，需要在函数前加
#include”string.h”;
当使用到字符串输入/输出函数时，加
#include”stdio.h”

综合练习

有5个字符串，编写一个程序，可以实现对这5个字符串进行从大到小的排序。字符串由键盘输入。

```
#include"string.h"
main( )
{char s[5][10],s1[10];
 int i,j;
 for(i=0;i<5;i++)
 gets(s[i]);
 for(i=0;i<5;i++)
 for(j=0;j<5;j++)
 if(strcmp(s[i],s[j])<0)
{strcpy(s1,s[i]);strcpy(s[i],s[j]);strcpy(s[j],s1);}
 for(i=0;i<5;i++)
 puts(s[i]);
 }
```

第九章 指针

1、存储单元

长度为8个比特的存储空间

2、地址

每个存储单元的地址编号

注意:变量名、变量值、变量地址的区别。

例：int a=3;

3、指针　就是地址

4、变量的指针　就是变量的地址

5、指针变量

用来存放其他变量地址（指针）的变量

9.1 变量的定义和使用

实例：定义一个指针变量，该指针变量用来存放整型变量a的地址。向变量a中输入数据并将数据输出来。

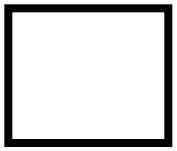
问题1：如何定义一个指针变量？

问题2：如何将整型变量a的地址赋给指针变量p呢？

问题3：如何向指针变量p所指向的变量a中输入和输出数据呢？

一、指针变量的定义

int *p;



指针变量定义的一般形式：

基类型 *指针变量名

例如：float *p; char *p;

二、指针变量的初始化

int *p;　　int a;

int *p=&a;

或　p=&a;

p和a的关系：指针变量p指向变量a;

注意：赋值时,只能将地址值赋给指针变量；而且指针变量和它所指向的变量的类型应该一致.

三、指针变量的使用

输入　方法1:　直接使用变量 a 的方法

scanf(“%d”,&a);

方法 2：间接使用指针变量p的方法

scanf(“%d”,p);

输出　方法1:　直接使用变量 a 的方法

printf(“%d”, a);

方法 2：间接使用指针变量p的方法

printf(“%d”,*p);

```
main( )
{ int a;
  int *p;
  p=&a;
```

int *p=&a; 或

```
int *p;
*p=a;
```

```
scanf (“%d”, p);    或   scanf(“%d”,&a);
printf (“%d”,*p);   或   scanf(“%d”, a);
```

注意：1、&是地址运算符，取变量的地址；*是指针运算符，取地址单元中的值。

2、如果指针变量p指向普通变量a，则*p等价于a；p等价于&a;

实训1　下面程序的结果是什么？

```
main( )
{int a,b;
 int *p,*q;
 p=&a;
 q=&b;
 *p=3;
 *&b=4;
 printf("%d,%d,%d,%d",a,b,*p,*q); }
```

结论：&和*运算符的优先级相同，结合方向为自右而左。

实训2

如果从键盘输入34和54,下面这个程序的输出结果是什么?

```
main( )
{int *p1,*p2,*p;
 int a,b;
 p1=&a;
 p2=&b;
scanf("%d%d",p1,p2);
 if(a<b)
 {p=p1;p1=p2;p2=p;}
 printf("a=%d,b=%d\n",a,b);
 printf("*p1=%d,*p2=%d\n",*p1,*p2); }
```

关于指针的几个思考

int *p1，*p2；

1、*p1+1和p1+1的区别是什么？

2、*p1-*p2和p1-p2的区别是什么？

3、p1>p2和*p1>*p2的区别是什么？

四、指针变量作为函数参数

从一个例子开始：

判断下面程序的结果是什么？

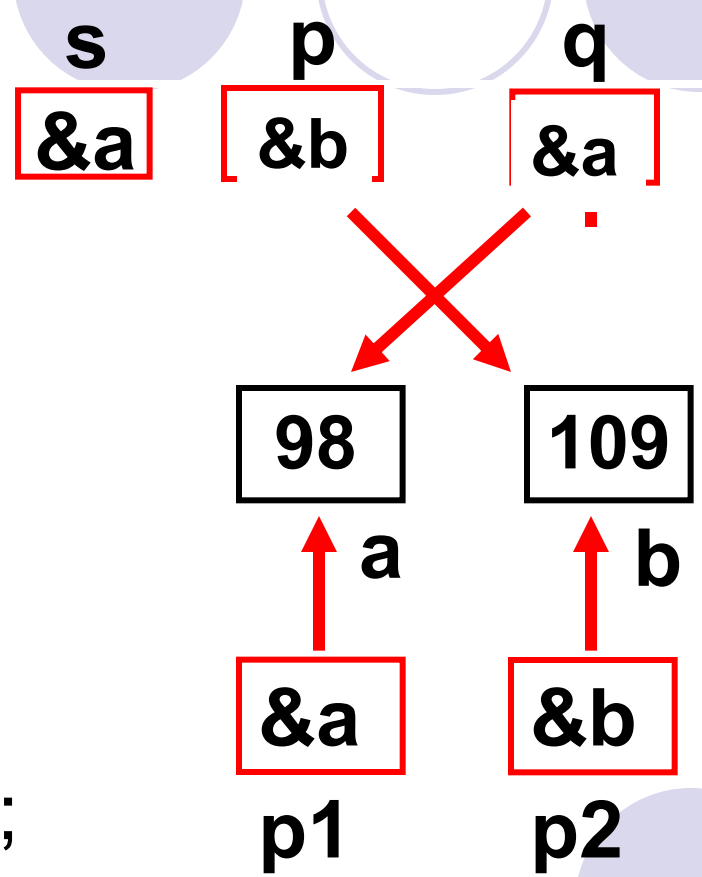
```
swap(int *p,int *q)          /*交换功能函数swap*/
{ int s;
  s=*p;
 *p=*q;
 *q=s;}
 main( )                     /*主函数main*/
 {int *p1,*p2;
  int a,b;
  p1=&a;
  p2=&b;
  scanf("%d%d",p1,p2);
  if(a<b) swap(p1,p2);       /*调用函数swap*/
  printf("%d,%d\n",a,b);
  printf("%d,%d\n",*p1,*p2);}
```

结论:

当实参和形参是指针变量时，在函数的调用中，实参传递给形参的是**地址**，在子函数中，使形参所指向的变量的值发生了变化，函数调用结束后，这些变化了的变量值依然保留了下来，从而在main中使用的就是这些已经改变了的变量值。

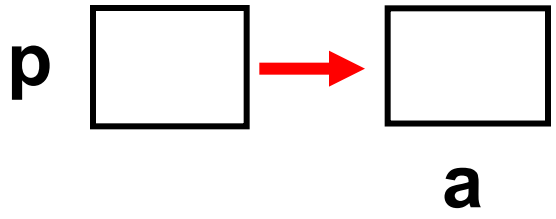
复习　当输入98和109时,输出结果为什么?

```
swap(int *p,int *q)
{int *s;
 s=p;   p=q;   q=s;}
 main( )
 {int *p1,*p2;
  int a,b;
  p1=&a;
  p2=&b;
  scanf("%d%d",p1,p2);
  if(a<b) swap(p1,p2);
  printf("%d,%d\n",a,b);
  printf("%d,%d\n",*p1,*p2);}
```

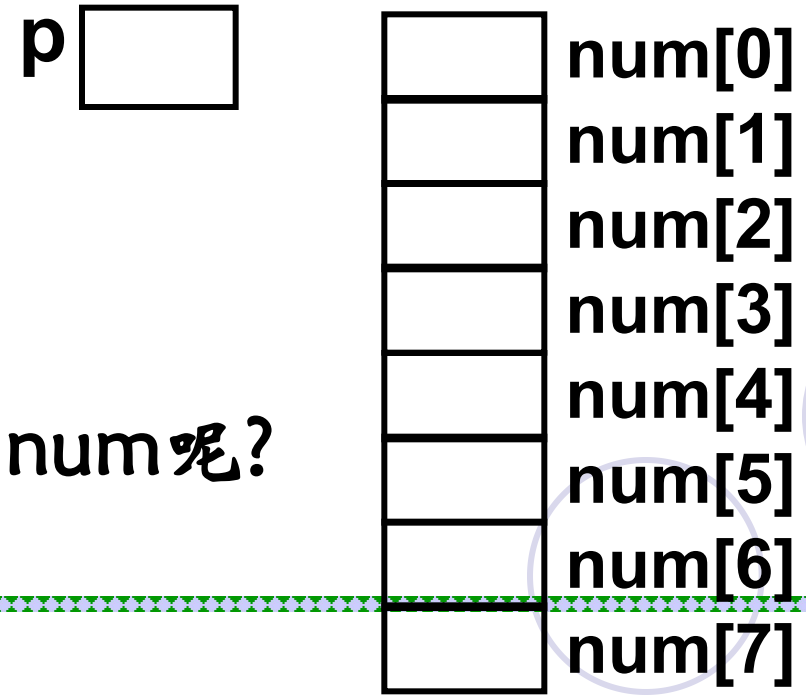


9.2 指针和一维数组

例：int *p;
int a;
p=&a;



int num[8];
int *p;

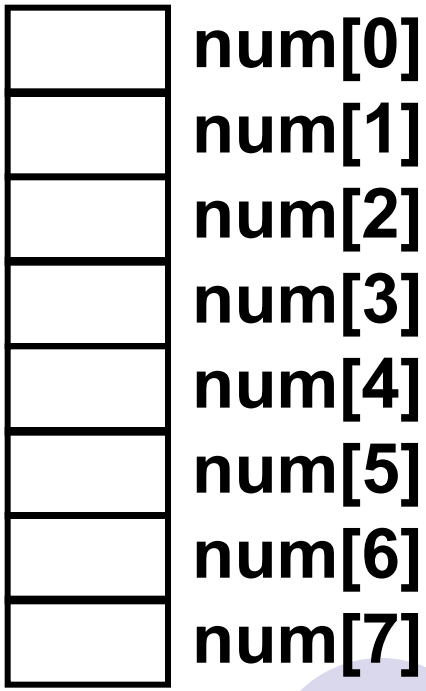


如何使指针p指向数组num呢?

一、指向数组元素的指针

int num[8];

int *p;



1、使指针指向数组元素num[0]

p=&num[0]; p=num;

2、使指针指向数组元素num[1]

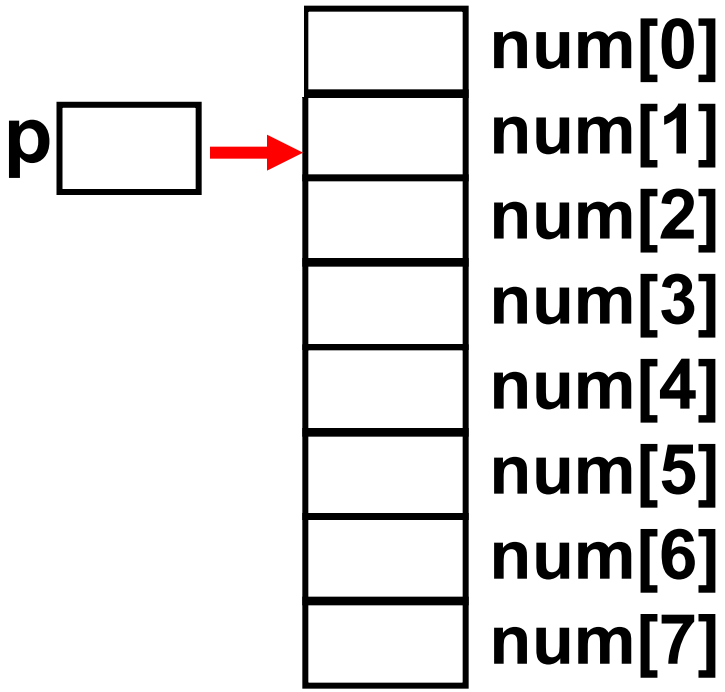
p=&num[1];

3、使指针指向数组元素num[i]

p=&num[i];

返回 下一页

二、通过指针引用数组元素



已知p=num；

1、p+1表示使指针指向num[1]，则*(p+1)表示数组元素num[1]的值

2、p+i表示使指针指向num[i]，则*(p+i)表示数组元素num[i]的值

3、p+i等价于num+i、&num[i]、&p[i];

4、*(p+i)等价于*(num+i)、num[i]、p[i];

返回 下一页

三、数组和指针使用实训

实训1 下面这个程序的功能是使用指针来实现数组的输入和输出。请在程序的空白处填入合适的内容。

```
main( )
{int s[10],*p,i;
  p=s;
 for(i=0;i<10;i++)
 scanf("%d",p++);
  p=s;
 for(i=0;i<10;i++)
 printf("%5d",*p++);
}
```

另一种写法：

```
main( )
{int s[10],*p;
 for(p=s;p<s+10;p++)
 scanf("%d",p);
 for(p=s;p<s+10;p++)
 printf("%5d",*p); }→
```

实训2 下面这个程序的功能是实现将一维数组s[12]升序排列,在在程序空白处填入合适内容.

```
main(  )
{int s[12],num,i,j;
 int *p;
  p=a;
 for(i=0;i<12;i++)
 scanf("%d",p+i);
 for(i=0;i<12;i++)
  for(j=i+1;j<12;j++)
 if(*(p+j)>*(p+i))
 {num=*(p+j);*(p+j)=*(p+i);*(p+i)=num;}
 for(p=a; ;p<=&s[11];  ;p++)
 printf("%5d",*p); }
```

另一种编写方法

```
main( )
{int s[12],num;
 int *p1,*p2;
 for(p1=s;p1<s+12;p1++)
 scanf("%d",p1);
 for(p1=s;p1<s+12;p1++)
  for(p2=p1+1;p2<s+12;p2++)
 if(*p2<*p1)
 {num=*p1;*p1=*p2;*p2=num;}
 for(p1=s;p1<s+12;p1++)
 printf("%5d",*p1);}
```

小结

如果 int s[10]，*p；
p=s；

1、p和s都表示数组元素的首地址。

2、指向数组的指针p可以移动，如p++表示使p下移指向下一个元素。而s不可以移动。

3、因为指针p可以移动，所以要随时注意p的位置，如果需要使p重新指向第一个元素，则p=s

4、*p++：等价于*(p++)表示先得到p指向的变量的值，然后使指针p加1指向下一个元素

5、*（++p）：先使指针p下移，指向下一个元素，然后取其所指向的变量的值

9.3 指针和二维数组

示例： 有一个二维数组s[3][4];

S[3][4]={{12,14,11,1},{43,12,23,45},{34,43,32,12}};

s[0]	12	14	11	1
s[1]	43	12	23	45
s[2]	34	43	32	12

1、s[0]表示第0行首地址

s[1]表示第1行首地址

s[2]表示第2行首地址

2、第i行首地址可以表示为s[i]，或*（s+i）或&s[i][0];

3、二维数组名s表示二维数组第一行首地址。

s[0]	12	14	11	1
s[1]	43	12	23	45
s[2]	34	43	32	12

4、第i行第j列元素的地址可以表示为：s[i]+j；
*（s+i）+j；&s[i][j];

5、第i行第j列元素的值可以表示为：*(s[i]+j)；
(（s+i）+j)；*&s[i][j];

那么如何使指针指向二维数组元素呢?

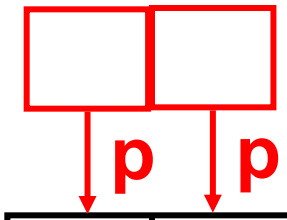
一、指向二维数组元素的指针变量

示例： 有一个二维数组s[3][4];

S[3][4]={{12,14,11,1},{43,12,23,45},{34,43,32,12}};

int *p;

p=s;



s[0]	12	14	11	1
s[1]	43	12	23	45
s[2]	34	43	32	12

1、P++表示使指针移向一下一个数组元素。

2、*p表示p所指向的数组元素中的值。

3、第i行第j列元素的值可以表示为*(p+i*4+j)

实训1 使用指针实现向二维数组中输入输出内容

方法1：

```
main( )
{int array[4][5];
 int *p;
for(p=array[0];p< _____;p++)
 scanf("%d",_____);
for(p=array[0];p<_____;p++)
 printf("%d",_____);
 }→
```

方法 2：

```
main(    )
{int array[4][5];
 int i,*p;
 p=array[0];
 for(i=0;i<20;i++,p++)
 scanf("%d",p);
 for(i=0;i<20;i++,p++)
 scanf("%d",*p);
 }→
```

二、指向二维数组元素的行指针变量

行指针变量:用来指向某一行元素,即一个一维数组的指针变量.

s

p ⟶

12	14	11	1
43	12	23	45
34	43	32	12

```
int (*p)[4];

p=s;
```

1、行指针变量只能指向某一行，而不能具体指向某一个元素。

2、p+i表示第i行元素的首地址。

3、*(p+i)+j等价于*(s+i)+j,表示第i行第j列元素的地址.

实训3

以下程序可分别求出方阵a中两个对角线上元素之和，请在空白处填入合适的语句来完善程序。

```
main()
{int a[6][6],i,j,k,p1,p2;
 for(i=0;i<6;i++)
  for(j=0;j<6;j++)
 scanf("%d",*(a+i)+j);
 k=6;
 p1=0;p2=0;
 for(i=0;i<6;i++)
 {p1=____ +(*(*(a+i)+i));
  p2=_____ +(*(*(a+i)+k));}
 printf("p1=%4d,p2=%4d\n",p1,p2);}
```

9.4 指针和字符串

一、字符串的地址

例：char name[ ]=“Sang DongLin”

存储方式：

name	S	a	n	g	D	o	n	g	L	i	n	\0

1、数组名name表示数组的起始地址

2、name+i表示第i个元素地址

3、*(name+i)表示第 i个元素的值。

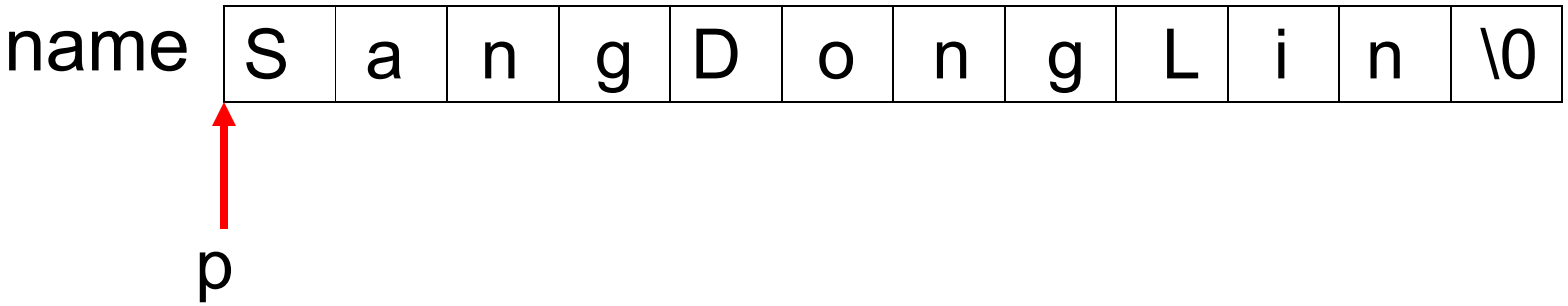
4、name++不成立

二、指向字符串的指针变量

字符指针变量的定义： char *p;

使字符指针变量指向字符数组：p=name;

指向关系：



定义和赋值的另一种形式：

Char *p=“SangDongLin”

实例：观察下面程序的功能。

```
main( )
{char *p="As young man,we should have
great idea!";
 int n;
 n=0;
 for(;*p!='\0';p++)
 if(*p<='z'&&*p>='a')  n++;
 printf("%d",n);
 }
```

9.5 几种特殊类型的指针变量

一、指针数组

一维指针数组的定义形式为：

类型 *数组名[数组长度]；

例： int *pointer[8];

实训：观察下面程序的作用是什么?当输入ding,wang,zhou,huang,chen,zou时,输出是什么?

```
main( )
{char *name[6],*temp; int i,j;
printf("\nPlease input the names:\n");
for(i=0;i<6;i++)
scanf("%s",name[i]);
for(i=0;i<6;i++)
for(j=i+1;j<6;j++)
if(strcmp(name[i],name[j])>0)
{temp=name[i];name[i]=name[j];name[j]=temp;}
for(i=0;i<6;i++)
printf("%s\n",name[i]);}
```

二、指向函数的指针

指向函数的指针变量定义的一般形式为:

数据类型 (*指针变量名)( );

例：long (*p)( );

表示指针p指向返回长整型数值的函数

```
long f(int n)
{long s;
 if(n==0||n==1) s=1;
 else s=f(n-1)*n;
 return(s);}
main( )
{int n;
 long (*p)( );
 p=f;
 long y;
 printf("please input a number:\n");
 scanf("%d",&n);
 y=p(n);
 printf("%d!=%ld\n",n,y);}
```

当输入4时,该程序的输出结果是什么?

三、指向指针的指针

指向指针的指针变量定义的一般形式为:

数据类型 **指针变量名;

例:

下面程序的输出结果是什么?

```
main( )
{int a,*p,**q;
 a=1024;
 p=&a;
 q=&p;
 printf("%d,%d,%d",a,*p,**q);}
```

练习1:以下程序的运行结果是

```
sub(int x,int y,int *z)
{*z=y-x;}
main( )
{int a,b,c;
 sub(10,5,&a);
 sub(7,a,&b);
 sub(a,b,&c);
 printf("%4d,%4d,%4d",a,b,c);
}
```

练习2:当运行以下程序时,从键盘输入”Happy!”,则程序的运行结果是:

```
main( )
{char str[10], *p=str;
 gets(p);
 printf(“%d\n”,stre(p));
}
stre(char str[ ])
{int num=0;
 while(*(str+num)!=‘\0’) num++;
 return(num);
}
```

6

练习3:以下程序的运行结果是

```
fun(char *w, int n)
{char t,*s1,*s2;
 s1=w;s2=w+n-1;
 while(s1<s2) {t=*s1++;*s1=*s2--;*s2=t;}
}
main( )
{char *p;
 p="1234567";
 fun(p,strlen(p));
 puts(p);
 }
```

1711717

```
main()
{char *p,s[6];
 int n;
 p=s;
 gets(p);
 n=*p-'0';
 while(*(++p)!='\0') n=n*8+*p-'0';
 printf("%d\n",n);
 }
```

习题课

1、以下程序运行时输出结果是什么？

```
main( )
{char *p="student";
 if(p=="student")
printf("yes");
 else
printf("no");
}
```

2、以下程序运行时输出结果是什么？

```
main()
{char *s,*s1="Here";
 s=s1;
 while(*s1) s1++;
 printf("%d\n",s1-s);
}
```

3、以下程序运行时输出结果是什么？

```
main( )
{int i,p[3][3]={1,2,3,4,5,6,7,8,9},*p1[3],(*p2)[3];
 for(i=0;i<3;i++)
 p1[i]=p[i];
 p2=p;
 for(i=1;i<3;i++)
 printf("\n%d,%d",*(*(p1+i)+1)+1,*(*++p2+1)+1);
}
```

4、以下程序运行时输出结果是什么？

```
void f(int a,int *b)
{a++;
 b++;
 (*b)++;}
main()
{int i,x[2]={4,4};
 f(x[0],&x[0]);
 printf("%d,%d",x[0],x[1]);
}
```

5、以下程序运行时输出结果是什么？

```
int c;
void f1(int x,int *sum)
{static int y;
 x++;
 y++;
 c=c+y;
 *sum=(x+y)/c;}
main()
{int a,b=100;
 for(a=0;a<2;a++)
 {f1(a,&b);
  printf("%d%d%d\n",a,b,c); }
```

6、有如下程序，假定所用编译系统用两个字节存储一个int型操作数，已知输出结果第一行是FFD2，则第二行应当是什么？

```
main()
{int a[10]={1,2,3,4,5,6,7,8,9,0},*p;
 p=&a[5];
 printf("%X\n",p);
 printf("%X\n",p-1);
}
```

7、以下程序运行时输出结果是什么？

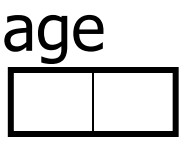
```
void fun(int *p1,int *p2);
main( )
{int i,a[6]={1,2,3,4,5,6};
 fun(a,a+5);
 for(i=0;i<5;i++)
 printf("%2d",a[i]);
 }
void fun(int *p1,int *p2)
{int t;
 if(p1<p2)
 {t=*p1,*p1=*p2;*p2=t;
  fun(p1+=2,p2-=2);
} }
```

8、以下程序运行时输出结果是什么？

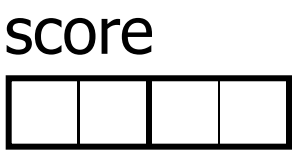
```
void swap(int *a,int *b)
{int c;
 c=*a;*a=*b;*b=c;}
main( )
{int i,j,a[3][3]={1,2,3,4,5,6,7,8,9};
 for(i=0;i<2;i++)
 for(j=0;j<2-i;j++)
 if(i==j) swap(&a[i][j],&a[i+2][j+2]);
 else swap(&a[i][j],&a[i+1][j+1]);
 for(i=0;i<3;i++)
 {for(j=0;j<3;j++)
  printf("%d",a[i][j]);
  printf("\n");  }}
```

C语言中常见的数据类型

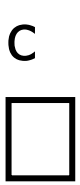
1、整型　　例：int age;



2、实型　　例：float score;

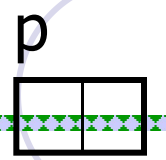


3、字符型　　例：char c;



4、数组类型　　例：int age[10];

5、指针类型　　例：int *p;



返 回　　下一页

第十章 结构体和共用体

- 结 构 体 类 型
- 共 用 体 类 型
- 枚 举 类 型

一、结构体类型的定义

结构体类型定义的一般形式：

```
struct   结构体名
{ 类型名1  成员名1;
  类型名2  成员名2;
  类型名3  成员名3;
           ⋮  ⋮
  类型名n  成员名n;};
```

定义一个通讯地址表结构体：

```
struct address
{  char   name[20];
   char   department[30];
   char   address[30];
   long   box;
   long   phone;
   char   email[30];};
```

通讯地址表结构：

Name	department	address	box	phone	email

定义一个成绩表结构体，该结构中包含班级、学号、姓名、语文成绩、数学成绩、英语成绩。

```
struct score
{char grade[20];
 long number;
 char name[20];
 float chinese;
 float math;
 float english; } ;
```

总结：

根据实际的需要，每个结构体类型所包含的成员类型和名字也不一样，在使用中，可以根据自己的需要进行结构体类型的定义。

思考：定义好结构体类型后，它是否占用内存空间？

类型和变量两个概念的区别所在

成绩表结构：

grade	number	name	chinese	math	english

二、结构体类型变量的定义

（1）先定义结构体类型再定义结构体变量

```
 struct student
{long int num;
  char name[20];
  int age;
  char sex;
  float score;};
```

struct student student1，student2；

（2）在定义类型的同时定义变量

```
struct student
{ long int num;
  char name[20];
  int age;
  char sex;
  float score;} student1,student2;
```

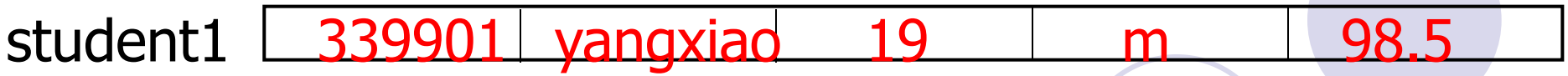
三、结构体类型变量的初始化

（1）可以在定义的同时进行初始化

```
struct  student  student1 ={339901,”yangxiao”,
                            20,’m’,89};
```

（2）在定义完后进行初始化

```
struct  student  student1;
student1={339901,”yangxiao”,20,’m’,89};
```



（3）可以将一个结构体变量作为一个整体赋给另外一个具有相同类型的结构体变量。

```
struct student student1,student2;
student1={3399011,”zhangsong”,20,’m’,89};
student2=student1;
```

注意：赋值时要注意成员项的类型。

关于结构体的几点问题

(1) 结构体类型与结构体变量的区别

类型只是一种模型，并不占内存空间。这种模型只是说明该种结构的数据在内存中占多大的存储空间，可以存放什么样的数据。而变量在定义为某种类型之后，系统根据该种类型结构为该变量分配一定的存储空间。

(2) 结构体类型中所包含的各成员项和普通变量的区别

没有区别，地位和作用相同。只是一个是单独存在，一个是结构体中的一个成员。结构体变量就象普通变量一样可以参加各种运算。

（3）在C语言中，结构类型是固定的，这种说法对吗？

不对。和整型、实型、字符型不同的是，结构体类型是用户根据自己的需要设计定义的。系统中没有预先定义好的固定的结构体类型，用户如果需要使用结构体类型的数据，都必须先在程序中先定义好相应的结构体类型。如成绩表结构、通讯表结构都是用户根据自己的需要定义的。

四、结构体变量的引用

思考：观察下面程序，看其运行结果是什么？

```
struct student
{long num;
 char name[20];
 float math;
 float chinese;
 float english;};

 main()
{struct student student1;
scanf("%ld%s",&student1.num, student1.name);
printf("%ld\t%s\t",student1.num,student1.name);
 }
```

五、结构体数组的定义和使用

1、定义　　例：

```
struct goods
{long num;
 char name[20];
 float price;
 int number;
 float sumprice;}good[5];
```

	num	name	price	number	sumprice
good[0]					
good[1]					
good[2]					
good[3]					
good[4]					

2、初始化

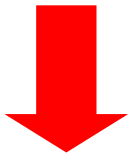
```
good[5]={{1101,“shoes”,89,2},{1102,“bag”,
53.5,8},{1103,“pen”,11,10},{1104,“knife”,1,12},
{1105,“Eraser”,1.5,6}};
```

3、输入

```
   for(i=0;i<5;i++)
{  scanf("%ld",&goods[ i ].num);
   scanf("%s",goods[ i ].name);
   scanf("%f",&goods[ i ].price);
   scanf("%d",&goods[ i ].number);
good[ i ].sumprice=good[ i ].price*good[ i ].number;
 }
```

4、输出

```
for(i=0;i<5;i++)
{printf("%ld\t",goods[ i ].num);
 printf("%s\t",goods[ i ].name);
 printf("%f\t", goods[ i ].price);
 printf("%d\t",goods[ i ].number);
 printf("%f\n",goods[ i ].sumprice);
}
```



```
for(i=0;i<5;i++)
printf("%ld\t%s\t%.1f\t%d\t%.1f\n",
good[i].num,good[i].name,good[i].price,
good[i].number,good[i].sumprice);}
```

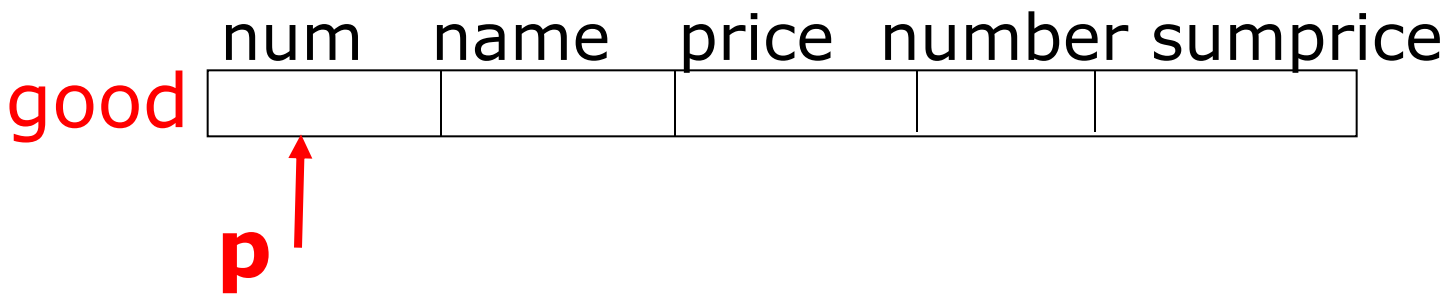
六、指向结构体变量的指针

```
struct goods
{long num;
 char name[20];
 float price;
 int number;
 float sumprice;} good;
```

思考：定义何种类型的指针，才能指向结构体变量good?

1、指向结构体类型变量指针的定义和使用

```
struct goods *p;
p=&good;
```



2、指向结构体类型变量指针引用

(*p).num ⟶ p->num

(*p).name ⟶ p->name

(*p).price ⟶ p->price

七、指向结构体数组的指针

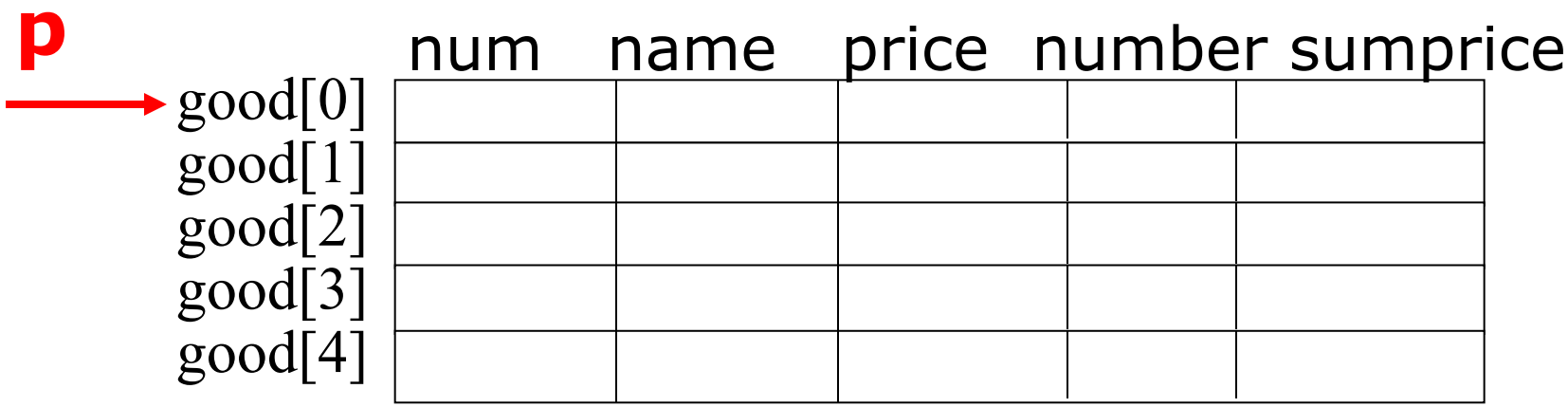
```
struct goods
{long num;
 char name[20];
 float price;
 int number;
 float sumprice;}good[5];
```

struct goods *p;

思考：如何使指针p指向结构体数组good，如果指针p指向了结构体数组good，则p++表示的意思是什么？又如何通过p来引用数组里的元素？

1、使指针指向结构体数组

p=good;



（1）、p++表示使指针下移

（2）、p->num表示引用p所指向的数组元素中的成员项num

2、关于结构体数组的两点注意事项

1）、结构体类型的指针只能指向结构体变量，而不能指向结构体变量中的某个成员项，但是可以通过指向结构体变量的指针来引用成员项。

2）、在指向结构体数组的指针中，(++p)->num表示先使指针p下移指向下一个数组元素，然后得到其所指向的数组元素的成员项num；而(p++)->num表示先得到p现在所指向的数组元素中的成员项num，然后使指针p下移。

本节内容小结

1、结构体类型的概念

2、结构体类型的定义

3、结构体变量的定义及其和结构体类型的区别所在

4、结构体变量的初始化

5、结构体变量的引用

结构体变量名.成员名

10.2 共用体

思考题

现在有三个变量，int型变量i，float型变量j，char型变量t，整型变量a。在任何时刻，只需要使用这4个变量中的某一个。

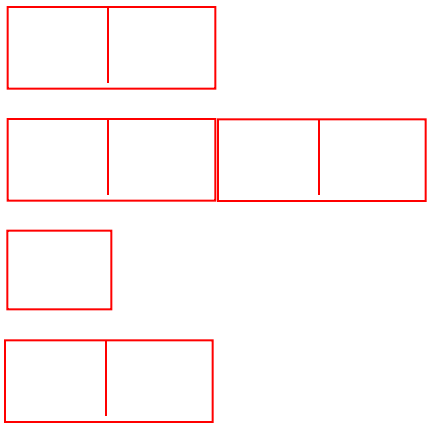
传统的定义:

int i;

float j;

char t;

int a;

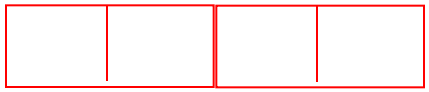


是否可以有更实用更省空间的做法呢?

一、共用体类型的定义

使用共用体定义的方法

```
union s
{int i;
 float j;
 char t;
 int x;};
```



共用体定义的一般形式：

```
Union 共用体名
{类型1  成员1;
 类型2  成员2;
 ……};
```

二、共用体变量的定义

union s x;

表示将变量x定义为共用体s类型

三、共用体变量成员的引用

```
union s
{int i;
 float j;
 char t;
 int g;}x;
```

如x.j或者x.t

四、共用体使用实例

```
union s
{int i;
 float j;
 char t;
 int g;};
main()
{union s x;
 clrscr();
 x.i=8;
 x.j=2.5;
 x.t='c';
 x.g=5;
 printf("%d",x.i);}
```

该程序的结果是什么？

10.3 枚举类型

定义：将对象可能存在的值一一列举出来。

例1：
```
enum week
{sunday,
 monday,
 tuesday,
 wednesday,
 thursday,
 friday,
 saturday,}week2;
```

枚举类型变量中的值，自动从0开始，依次递增。

例**2**：enum week
{sunday,
monday,
tuesday=2,
wednesday,
thursday,
friday,
saturday,}week2;

当枚举中的某个成员赋值后，其后的成员按依次加1的规则确定其值。

10.4 用户自定义类型

例：

Typedef char* STR;

表示用STRING 代表了char*这种数据类型。

char *p;

STR p;

例：

```
type  struct  student
{char  name[20];
  int number;
  float score1;
  float score2;
  float score3;}STU;
```

表示用STU表示结构体数据类型struct student；

```
struct student  student1;
```

```
STU  student1;
```

例：

```
typedef  int INTEGER;
typedef  char  CHARACTER;
```

练习1：

1、当说明一个结构体变量时，系统分配给它的内存是（　　）

A、各成员所需内存量的总和

B、结构中第一个成员所需内存量

C、成员中占内存量最大者所需的容量

D、结构中最后一个成员所需内存量

练习2：

若有以下说明语句：

```
struct student
{int age;
 int num;}std,*p;
p=&std;
```

则以下对结构体变量std中成员age的引用不正确的是（　　）

A、std.age　　B、p->age

C、(*p).age　　D、*p.age

练习3：

若有以下说明语句，则下面表达式中值为1002的是（　　）

```
struct student
{int age;
 int num;};
struct student stu[3]=
{{20，1001},{19，1002},{21，1003}};
struct student *p;
p=stu;
```

A、(++p)->num　　B、(p++)->age

C、(*p).num　　D、(*++p).age

练习4：

C语言共用体变量在程序运行期间（　　）

A、所有成员一直驻留在内存中

B、只有一个成员驻留在内存中

C、部分成员驻留在内存中

D、没有成员驻留在内存中

练习5：

以下程序的运行结果是：

```
 union pw
{int i;
 char ch[2];}a;
main()
{a.ch[0]=13;
 a.ch[1]=0;
 printf("%d\n",a.i);}
```

练习6：

以下程序用以输出结构体变量bt所占内存单元的字节数，请在程序空白处填入合适内容。

```
  struct ps
{double i;
 char arr[20];};
main()
{struct ps bt;
 printf("bt size:%d\n",_____);}
```

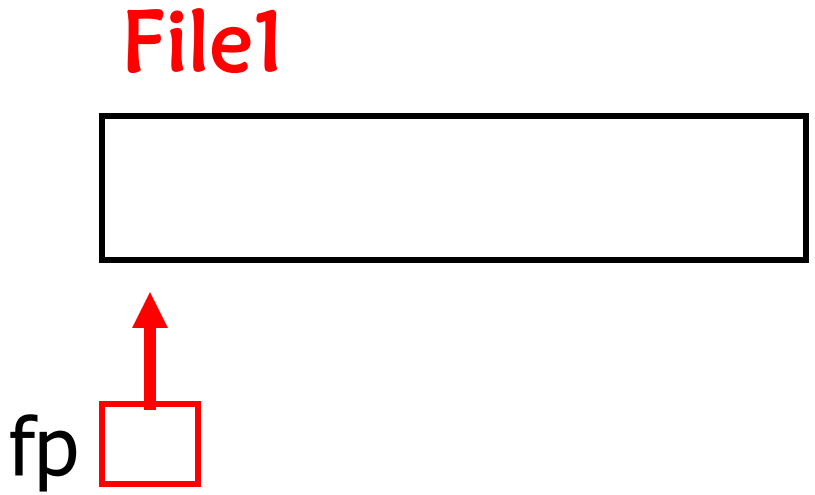
第十一章 文件

学习目标

- 理解C语言中文件的概念
- 掌握新建文件、打开文件的函数使用
- 掌握文件输入输出函数的使用

一、文件的概念

1、文件：存储在外部介质上数据的集合。



2、文件类型指针：指向某一个文件

定义形式：FILE *fp;

二、文件的打开和关闭

说明： 对文件进行读写之前应该“打开”该文件，在使用结束之后应关闭该文件。

1、文件的打开（fopen函数）

Fopen函数的调用方式：

```
FILE *fp;
fp=fopen(文件名，使用文件方式）；
```

使用文件方式：

- rb:该形式打开，只能输出，不能输入
- wb:新建一个二进制文件，向其输入
- rb+:打开一个二进制文件，可读又可写
- wb+:新建一个二进制文件，可读又可写

思考：如果文件打不开，怎么办？

常用的文件打开形式：

```
If((fp=fopen(“file1”,”r”))==NULL)
{printf(“cannot open this file\n”);
  exit(0);}
```

2、文件的关闭（fclose函数）

关闭就是指使文件指针变量不指向该文件。

Fclose函数的调用方式：

```
fclose(fp);
```

三、文件的读写

1、fwrite函数

功 能：从内存写到文件所在的硬盘中。

一般形式：fwrite(buffer,size,count,fp);

Buffer:要输出数据的地址

Size:　要读写的字节数

Count:要读写多少个size字节的数据项

Fp: 文件型指针

2、fread函数

功 能：从硬盘文件中将数据读到内存中。

一般形式：fread(buffer,size,count,fp);

Buffer:要读入数据所在的地址

四、文件输入/输出函数的使用

1、输入函数fscanf

fscanf函数的作用是从指定的文件中读出数据输入到指定的变量中

一般形式为:

fscanf(文件指针，格式字符串，输入表列);

例：如果指针fp所指向的文件中有如下字符：3，4.5，现在需要将这两个值输入到整型变量i和实型变量j中

fscanf(fp,”%d%f”,i,j);

2、输出函数fprintf

fprintf实现将变量输出到指定的文件上

一般形式为:

fprintf(文件指针，格式字符串，输出表列);

例:需要将整型变量i和实型变量j的值分别按%d和%f的格式输出到fp所指向的文件中

```
fprintf(fp,“%d%f”,i,j);
```