

3

结构化程序设计

3.1 引言

3.2 基本数据类型

3.3 表达式与运算规则

3.4 控制语句与基本语法

3.5 数组与批量数据处理

3.6 结构体与复杂信息处理

3.7 结构化与计算思维实践

3.8 小结与能力要求

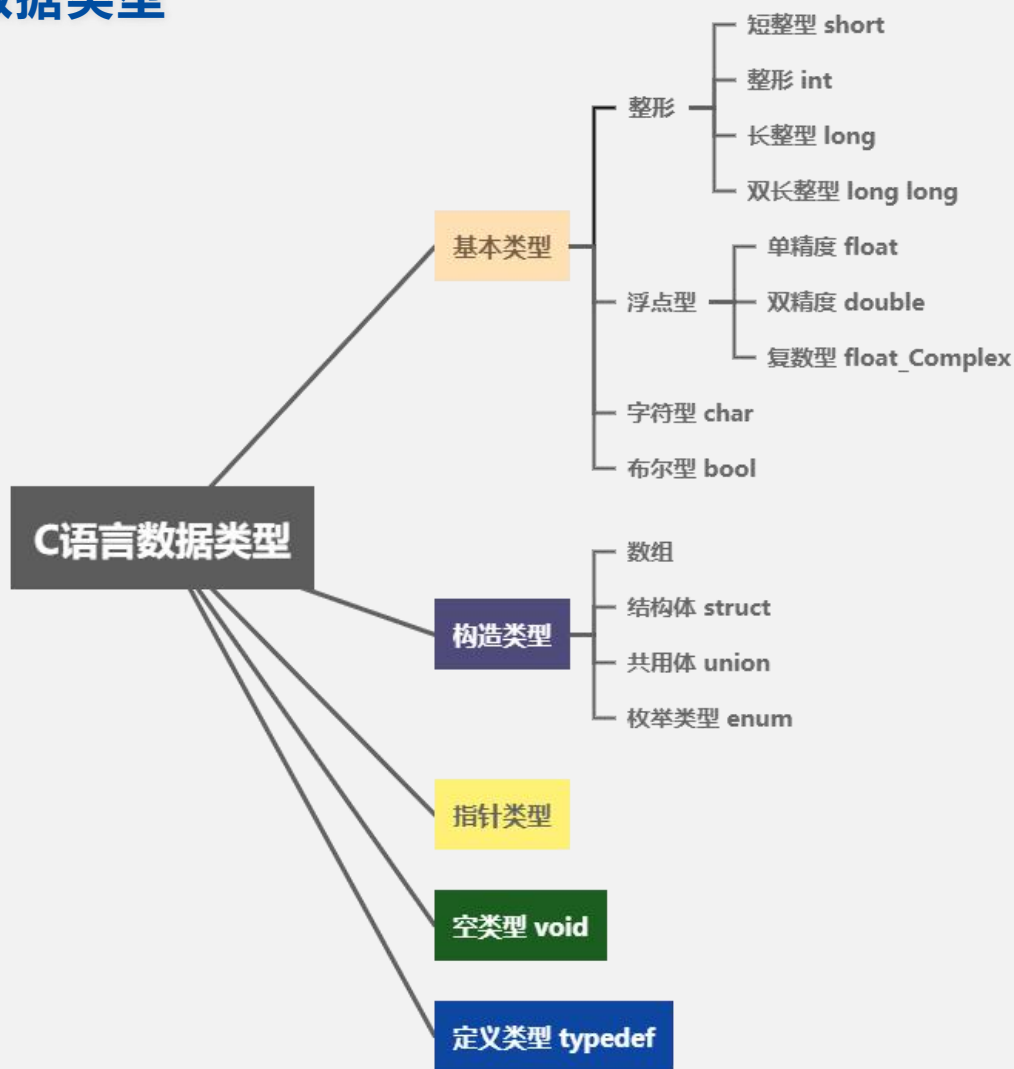


3.1 引言

- 结构化是一种被广泛使用并且与语言无关的程序设计思想与方法。
- 在面向对象方法兴起之后，人们在进行软件开发时很少再提起结构化程序设计方法。但事实上，无论使用哪种程序设计技术、哪种程序设计语言，无论软件规模大小，无论问题简单还是复杂，在实现具体的程序功能时，都需要用“顺序、选择、循环”等控制结构组织程序语句，这就是狭义的“结构化”思想。广义上，解决问题的思路都被称为“算法”。



3.2 基本数据类型





类型	符号	关键字	所占位数	数的表示范围
整型	有	(signed) int	32	-2147483648 - 2147483647
		(signed) short	16	-32768 - 32767
		(signed) long	32	-2147483648 - 2147483647
	无	unsigned int	32	0 - 4294967295
		unsigned short	16	0 - 65535
		unsigned long	32	0 - 4294967295
实型	有	float	32	4.4e-384.4e38
	有	double	64	1.7e~308 - 1.7e~308
字符型	有	char	8	-128 - 127
	无	unsigned char	8	0 - 255

说明: 数据类型所占字节数可能随系统环境不同而不同



3.2.1 整型数据

例 3.1 `#include<stdio.h>`
`int main()`
`{`
`short a=32767;`
`printf("short 类型的长度是%d\n", (int) sizeof (short));`
`a=a+3;`
`printf("a 的值是%d",a);`
`return 0;`
`}`

sizeof : 求对象长度运算符

- 操作对象可以是任何数据对象或数据类型，在编译时计算对象或类型占用的存储空间字节数。

(TYPE) 操作数 : 强制类型转换运算符

- 经强制类型转换运算符运算后，返回一个具有 TYPE 类型的数值。其中TYPE 与操作数都只能是基本数据类型，且该运算并不改变操作数本身，而是得到一个新的值。



3.2.1 整型数据

例 3.2 `#include<stdio.h>`

```
int main()
{
    int score, i;
    for(i=0;i<10;i++)
    {
        printf("请输入第%d个成绩:", i+1);
        scanf("%d",&score);
        if(score>=60) printf("恭喜你, 通过考试! ");
        else printf("别灰心, 明年再来! ");
    }
    return 0;
}
```



C语言提供了以下三种浮点数据类型：

- ## 浮点常量表示形式:

- 进制数形式: 0.123 12.37 1.489
- 指数形式 (e或E之前必须有数字 指数必须为整数) : 1.23e4 123E-2
e-5 1.2E-4.5



3.2.2 浮点型数据

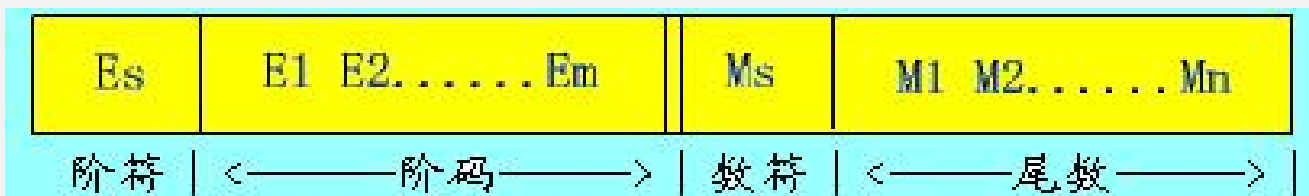
11.0101: 0.110101×2^{10} 1.10101×2^1 1101.01×2^{-10} $0.00110101 \times 2^{100}$

- 为了提高数据精度以及便于浮点数间的比较计算，在计算机中规定浮点数的尾数用纯小数表示，此外，将尾数最高位为1的浮点数称为**规格化数**，即 $N=0.110101 \times 2^{10}$ 为规格化形式，此时精度最高。

一个机器浮点数由阶码和尾数及其符号位组成

尾数：用定点小数表示，给出有效数字的位数决定了浮点数的表示精度；

阶码：用整数形式表示，指明小数点在数据中的位置，决定了浮点数的表示范围。





3.2.2 浮点型数据

浮点数的误差：

- 浮点数的小数部分在从十进制转换为二进制时产生的计算误差；
- 有限的有效数字位数导致的数据表示误差。

例 3.3 `#include <stdio.h>`

```
int main()
```

```
{
```

```
    float x1=0.1;
```

```
    double x2=0.1;
```

```
    if( x1 == x2) printf (" x1 与 x2 相等 ");
```

```
    else printf (" x1 与 x2 不相等 ");
```

```
    return 0;
```

```
}
```

```
if(fabs(x1-x2)<1e-6)
```



3.2.3 字符型数据

字符常量：用单引号括起来的单个普通字符或转义字符.

如： 'a' 'A' '?' '\n' '\101'

字符常量的值：该字符的ASCII码值

'A' - 65	'a' - 97
'0' - 48	'\n' - 10



3.2.3 字符型数据

转义字符：反斜线后面跟一个字符或一个代码值表示

转义字符	ASCII码值	功能含义
\a	7	响铃(beep)
\b	8	退格符，打印位置向左移一列
\f	12	走纸换页，打印位置移到下一页的开头
\t	9	横向制表符，打印位置移到下一个横向制表位
\n	10	换行符，打印位置移到下一行的开头
\r	13	回车符，打印位置移到当前行的开头
\v	11	纵向制表符，打印位置移到下一个纵向制表位
\\	92	反斜杠字符\
\?	63	问号字符?
\'	39	单引号字符'
\"	34	双引号字符"
\o \oo \ooo		1~3 位八进制数，ASCII 码等于该值的字符
\xh \xhh		1~2 位十六进制数，ASCII 码等于该值的字符



3.2.3 字符型数据

例 3.4 #include<stdio.h>

```
int main()
{
    char c,i;
    for(i=0;i<5;i++)
    {
        c=getchar();
        putchar(c-'a'+'A');
    }
    return 0;
}
```

getchar()

功能：从键盘读一字符

返回值：正常 返回读取的代码值；

出错 返回EOF(-1)

putchar(c)

参数：c为字符常量、变量或表达式

功能：把字符c输出到显示器上

返回值：正常 为显示的代码值

出错 为EOF(-1)



3.2.3 字符型数据

例 3.5 `#include<stdio.h>`
`int main()`
`{`
`char c1,c2,c3,c4;`
`scanf("%c%c",&c1,&c2);`
`getchar();`
`c3=getchar(); c4=getchar();`
`printf("%c %c %c %c\n",c1,c2,c3,c4);`
`printf("%c,%d\n",c1,c1);`
`printf("%d,%d\n",sizeof(c1),sizeof("1234"));`
`return 0;`
`}`



3.2.4 幻数与宏定义

- C 语言的设计者把直接书写在程序中的常数称为“幻数”。
- 处理这种幻数的一种方法是赋予它们有意义的名字。

例 3.6 `#include<stdio.h>`

```
int main()
{
    int num;
    float average;
    printf("请输入成绩为优秀的学生数量:");
    scanf("%d",&num);
    average=num/80.0;
    printf("优秀学生占比为%%%.2f\n",100*average);
    return 0;
}
```



3.2.4 幻数与宏定义

- C 语言的设计者把直接书写在程序中的常数称为“幻数”。
- 处理这种幻数的一种方法是赋予它们有意义的名字。

例 3.6 `#include<stdio.h>`

```
#define STU_NUM 80.0
```

```
int main()
```

```
{
```

```
    int num;
```

```
    float average;
```

```
    printf("请输入成绩为优秀的学生数量:");
```

```
    scanf("%d",&num);
```

```
    average=num/STU_NUM;
```

```
    printf("优秀学生占比为%%%.2f\n",100*average);
```

```
    return 0;
```

```
}
```




3.2.4 幻数与宏定义

“**#define**” 指令 (宏定义): 用符号名 (在宏定义中称为宏名) 定义一个特定的字符串

#define 宏名 替换的文本字符串

```
#define YES 1
#define NO 0
#define PI 3.1415926
#define OUT printf("Hello,World");
```

例 `#define PI 3.14159`
`printf("2*PI=%f\n",PI*2);`
宏展开: `printf("2*PI=%f\n",3.14159*2);`

例 `#define WIDTH 80`
`#define LENGTH WIDTH+40`
`var=LENGTH*2;`
宏展开: `var= 80+40 *2;`



3.2.4 幻数与宏定义

- C 语言的设计者把直接书写在程序中的常数称为“幻数”。
- 处理这种幻数的一种方法是赋予它们有意义的名字。

例 3.7 `#include<stdio.h>`
`#define N a==2) printf("a=%d\n",a);`
`#define b 2;`
`int main()`
`{`
 `int a=b`
 `if(N`
`}`



3.3 表达式与运算规则

C 运算符

算术运算符: + - * / % ++ --
关系运算符: < <= == > >= !=
逻辑运算符: ! && ||
位运算符: << >> ~ | ^ &
赋值运算符: = 及其扩展
条件运算符: ? :
逗号运算符: ,
指针运算符: * &
求字节数: sizeof
强制类型转换: (类型)
分量运算符: . ->
下标运算符: []
其它: () -



3.3 表达式与运算规则

- **操作对象的类型：**部分运算符对操作数的类型有特定要求，比如求余和自增自减不能作用于浮点型的对象，否则语法检查会报错。二元与三元运算符则要求操作数类型相同，否则会进行隐式类型转换；
- **操作对象的存储特性：**赋值表达式 $E1=E2$ 中的 $E1$ 被称为左值 (lvalue)。左值是一种特殊的表达式，是对某些具有存储空间的对象引用。变量是最简单的左值。左值的名称起源于赋值运算符的左操作数，但并非都处于运算符的左边。取地址运算符的右操作数（如 $\&a$ 中的 a ）、自增自减运算符的操作数（如 $++i$ 和 $i++$ 中的 i ）都要求是左值，否则会引起编译错误。左值被操作后通常不再是左值（唯一的例外是指针操作符 $*$ ），比如 $(i++)++$ 之所以编译错误，是因为 $i++$ 是一个表达式，其结果不再是一个左值，不能再进行 $++$ 运算。
- **优先级：**当表达式含有多个运算符时，需要注意是否能根据优先级得到正确的运算次序，必要时可以使用圆括号提升优先级。
- **结合性：**当表达式含有多个优先级相同的运算符时，需要注意是否能根据结合性得到正确的运算次序，必要时可以使用圆括号改变结合关系。
- **副作用：**自增自减以及赋值等运算符，在对表达式求值的同时，会修改操作对象的值。某些情况下副作用发挥作用的时机与编译器有关，可能引起意料之外的问题。



算术表达式

- 基本算术运算符： $+$ $-$ $*$ $/$ $\%$

结合方向：从左向右

优先级： $-$ ----- $*$ $/$ $\%$ ----- $+$ $-$

(2)

(3)

(4)

说明：

'-'为单目运算符时,右结合性

两整数相除, 结果为整数

%要求两侧均为整型数据

例	$5/2$	$=$	2
	$-5/2.0$	$=$	-2.5



算术表达式

自增、自减运算符++ --

- 作用：使变量值加1或减1
- 种类：
 - 前置 ++i, --i (先执行i+1或i-1, 再使用i值)
 - 后置 i++,i-- (先使用i值,再执行i+1或i-1)

例	j=3; k=++j;	//k=4,j=4
	j=3; k=j++;	//k=3,j=4
	j=3; printf("%d",++j);	//4
	j=3; printf("%d",j++);	//3
	a=3;b=5;c=(++a)*b;	//c=20,a=4
	a=3;b=5;c=(a++)*b;	//c=15,a=4



算术表达式

说明:

- ++ -- 不能用于常量和表达式,如5++, (a+b)++
- ++ --结合方向: 自右向左
- 优先级: - ++ -- ----->* / % ----->+ -
(2) (3) (4)

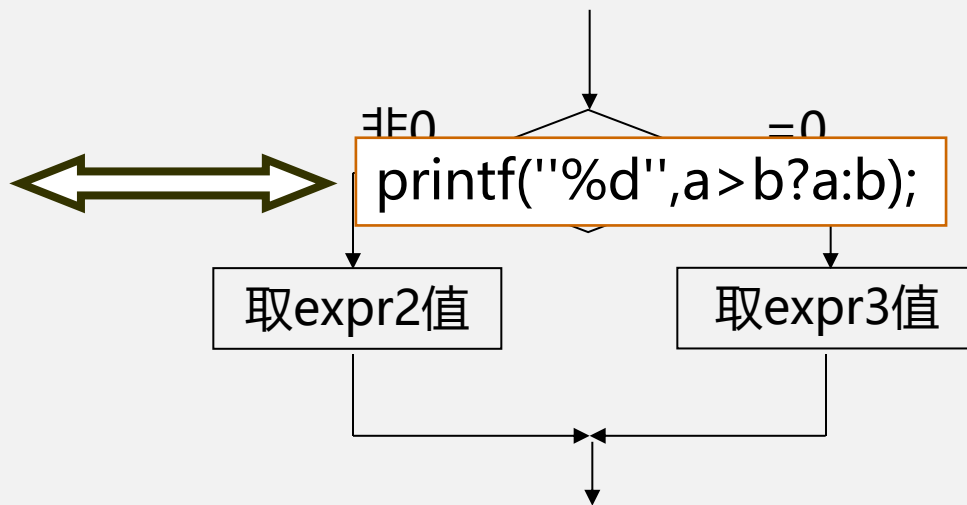
```
#include<stdio.h>
main()
{
    int i=3,j;
    printf("j=%d\n",j=i+++i+++i++);
    printf("i=%d\n",i);
}
```



条件表达式

- 一般形式: $\text{expr1} ? \text{expr2} : \text{expr3}$
- 执行过程
- 功能: 相当于条件语句, 但不能取代一般if语句

```
例 if (a > b)
    printf("%d", a);
else
    printf("%d", b);
```





条件表达式

- 一般形式: `expr1 ? expr2 : expr3`
- 执行过程
- 功能: 相当于条件语句, 但不能取代一般if语句

例 求 $a+|b|$
`printf("a+|b|=%d\n", b>0?a+b:a-b);`

例 `(a==b)?'Y':'N'`
`(x%2==1)?1:0`
`(x>=0)?x:-x`
`(c>='a' && c<='z')?c-'a'+'A':c`



条件表达式

- 一般形式: `expr1 ? expr2 : expr3`
- 执行过程
- 功能: 相当于条件语句, 但不能取代一般if语句
- 条件运算符可嵌套 如 `x>0?1:(x<0?-1:0)`
- 优先级: 13
- 结合方向: 自右向左
如 `a>b?a:c>d?c:d ⇔ a>b?a:(c>d?c:d)`
- `expr1`、`expr2`、`expr3`类型可不同, 表达式值取较高的类型

`x?'a':'b'` `//x=0,表达式值为'b' ; x≠0,表达式值为'a'`

`x>y?1:1.5` `//x>y,值为1.0 ; x<y,值为1.5`



逗号表达式

- 把多个独立的表达式并列放置在一个表达式中

temp=a, a=b, b=temp;

for(sum=1.0, i=1; i<=n; ++i) sum*=i;

a=3*5,a*4	//a=15,表达式值60
a=3*5,a*4,a+5	//a=15,表达式值20
x=(a=3,6*3)	//赋值表达式, 表达式值18, x=18
x=a=3,6*a	//逗号表达式,表达式值18,x=3
a=1;b=2;c=3;	
printf("%d,%d,%d" ,a,b,c);	//1,2,3
printf("%d,%d,%d" ,(a,b,c),b,c);	//3,2,3

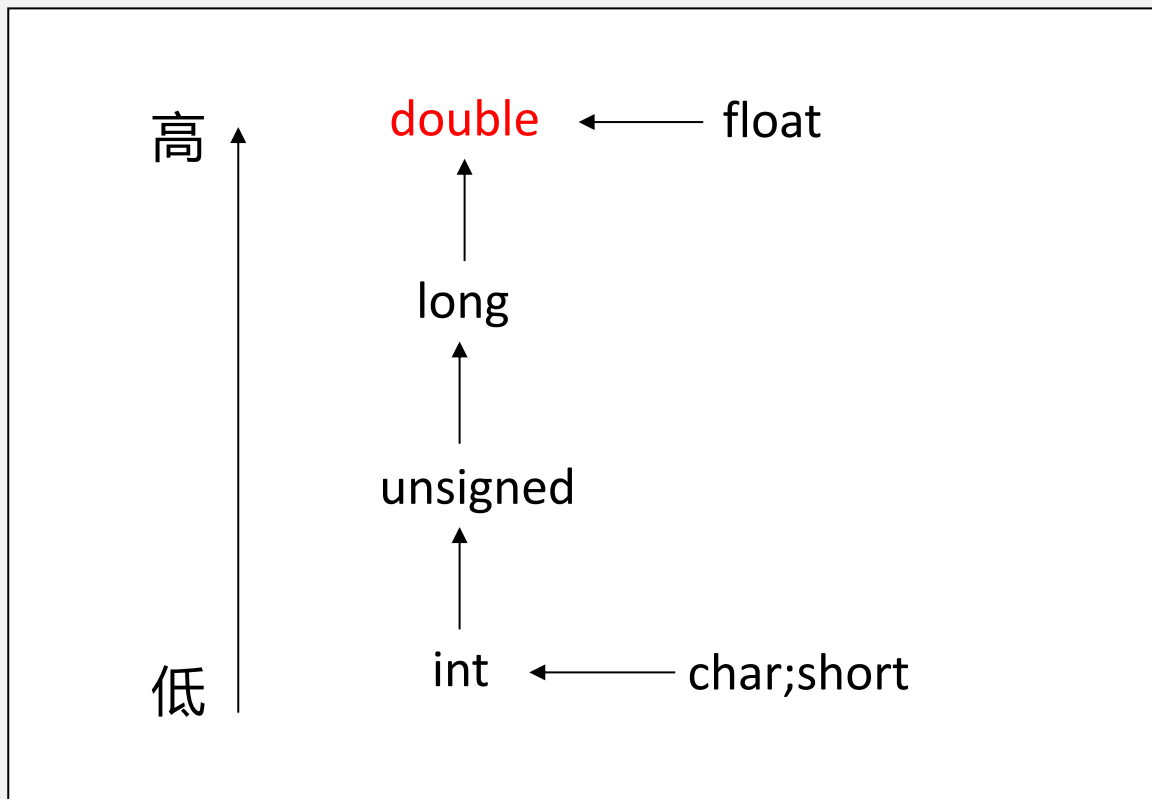


隐式类型转换

- **运算转换**：不同类型数据混合运算时
 - **赋值转换**：把一个值赋给与其类型不同的变量时
 - **输出转换**：输出时转换成指定的输出格式
 - **函数调用转换**：实参与形参类型不一致时转换
-
- 运算转换规则:不同类型数据运算时先**自动**转换成同一类型



隐式类型转换



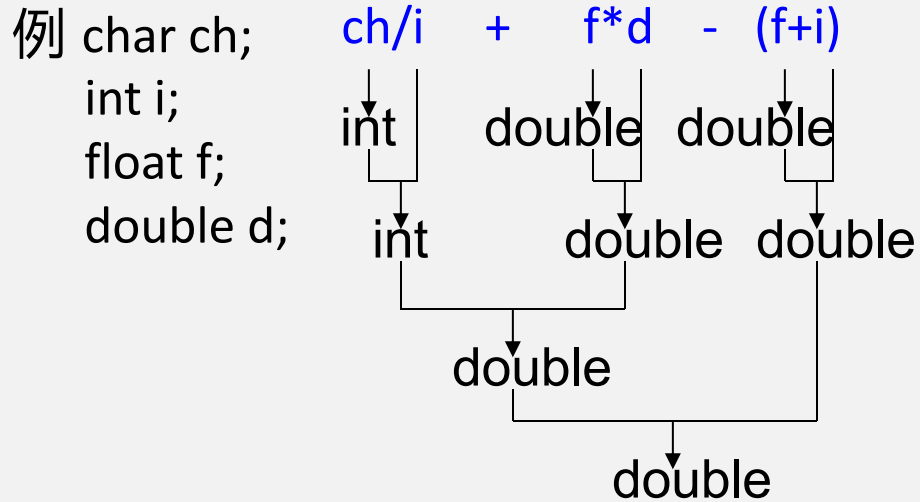
说明:

← 必定的转换

↑ 运算对象类型不同时转换



隐式类型转换





隐式类型转换

例 3.8 `#include<stdio.h>`
`int main()`
`{`
 `unsigned uint=0;`
 `int i=-1;`
 `if(i<uint) printf("-1 当然小于 0");`
 `else printf("-1 并不小于 0");`
 `return 0;`
`}`



3.4 控制语句与基本算法

结构化程序设计

基本思想：任何程序都可以用三种基本结构表示，限制使用无条件转移语句（goto）。

结构化程序：由三种基本结构反复嵌套构成的程序。

优点：结构清晰，易读，提高程序设计质量和效率。

三种基本结构

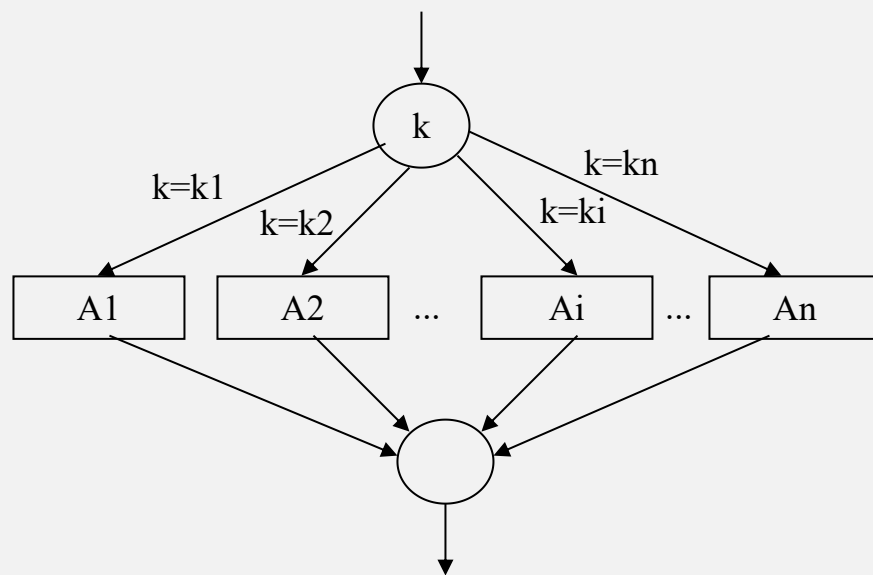
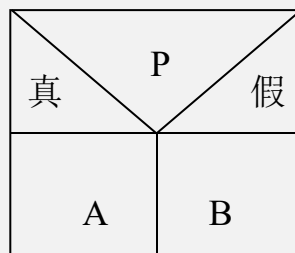
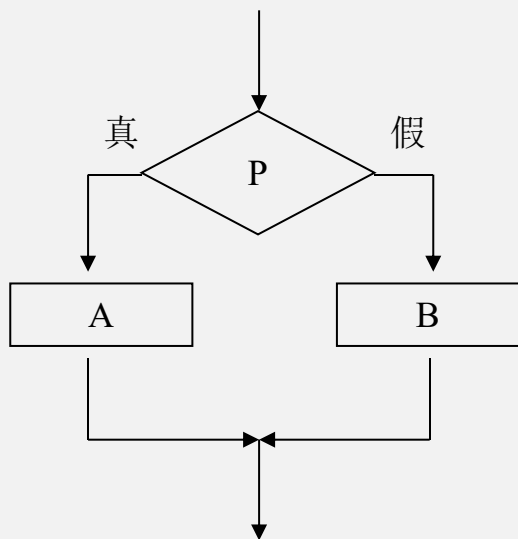
顺序结构 选择结构 循环结构



3.4 控制语句与基本算法

分支语句（选择结构）

- 又称为分支结构，根据给定的条件是否满足，来决定从给定的两组或多组操作中选择之一去执行。
- 通常使用两种语句来实现：**if语句（条件语句）**
switch语句（开关语句）





基本if语句

简单if语句

格式: `if` (表达式)

语句1/复合语句1

选择if语句

格式: `if` (表达式)

语句1/复合语句1

`else`

语句2/复合语句2



例 3.9 用if~else语句，编写程序实现比较输入的两个数a、b的大小。且将较大者赋给变量x，较小者赋给变量y。

```
#include<stdio.h>
void main()
{
    int a,b,x,y;
    printf("input a:"); scanf("%d",&a);
    printf("input b:"); scanf("%d",&b);
    if(a>b)
    {
        x=a;
        y=b;
    }
    else
    {
        x=b;
        y=a;
    }
    printf("a=%d,b=%d\n",a,b);
    printf("x=%d,y=%d\n",x,y);
}
```

变量定义和变量输入

判断a、b的大小关系并根据情况选择不同赋值



多重if语句（多条件分支语句）

语句格式	if (表达式1)	语句1/复合语句1
	else if (表达式2)	语句2/复合语句2
	else if (表达式3)	语句3/复合语句3
	
	else	语句n/复合语句n



注意else和if的配对关系，一个else必须与一个if配对，且只能与一个if配对。



例 3.10 编写成绩评定等级的C程序，根据输入学生的成绩来划分优、良、及格和不及格4个等级

```
#include<stdio.h>
void main()
{
    int score;
    printf("score=");
    scanf("%d",&score);
    if(score>84)
        printf("优秀\n");
    else if(score>74)
        printf("良好\n");
    else if(score>59)
        printf("及格\n");
    else
        printf("不及格\n");
}
```

定义并输入一个成绩

根据成绩选择不同等级

分数	等级
100~85	优秀
84~75	良好
74~60	及格
59~0	不及格



if语句的嵌套

```
if (表达式1)
    if (表达式2)
        语句1/复合语句1
    else
        语句2/复合语句2
```

内嵌if

```
if (表达式1)
    if (表达式2)
        语句1/复合语句1
else
    复合语句2
```

内嵌if

```
if (表达式1)
    语句1/复合语句1
else
    if (表达式2)
        语句2/复合语句2
    else
        语句3/复合语句3
```

内嵌if

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

内嵌if

内嵌if



例 3.11 编程实现输入三个数a, b, c, 输出其中最大者

```
#include<stdio.h>
void main()
{
    int a,b,c;
    printf("input a:");
    scanf("%d",&a);
    printf("input b:");
    scanf("%d",&b);
    printf("input c:");
    scanf("%d",&c);

    if(a>b)
        if(a>c) printf("a最大\n");
        else   printf("c最大\n");
    else
        if(b>c) printf("b最大\n");
        else   printf("c最大\n");
}
```



1、相等的情况如何判断?

变量定义和输入

2、题目改成输出三个数的大小关系如何实现?

通过if~else的嵌套实现最大数的判断和输出



例 3.12 编程实现求分段函数

$$y = f(x) = \begin{cases} x & 0 \leq x < 10 \\ x^2 + 1 & 10 \leq x < 20 \\ x^3 + x^2 + 1 & 20 \leq x < 30 \end{cases}$$

```
#include<stdio.h>
void main()
{
    float x,y;
    printf("input x:");
    scanf("%f",&x);
    if(x<20&&x>=0)
    {
        if(x>=10)
            y=x*x+1;
        else
            y=x;
    }
    else
        y=x*x*x+x*x+1;
    printf("x=%f,y=%f\n",x,y);
}
```

通过if~else的嵌套实现分段函数的计算



例 3.13 判断以下程序哪些能够正确实现该分段函数

$$y = f(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases}$$

PROGRAM 1

```
int main( )
{
    int x,y;
    scanf("%d",&x);
    if(x<0)
        y=-1;
    else
        if(x==0) y=0;
        else y=1;
    printf("x=%d,y=%d\n",x,y);
    return 0;
}
```

PROGRAM 2

```
int main( )
{
    int x,y;
    scanf("%d",&x);
    if(x>=0)
        if(x>0) y=1;
        else y=0;
    else y=-1;
    printf("x=%d,y=%d\n",x,y);
    return 0;
}
```



例 3.13 判断以下程序哪些能够正确实现该分段函数

$$y = f(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases}$$

PROGRAM 3

```
int main( )
{
    int x,y;
    scanf("%d",&x);
    y=-1;
    if(x!=0)
        if(x>0 ) y=1;
    else y=0;
    printf("x=%d,y=%d\n",x,y);
    return 0;
}
```

PROGRAM 4

```
int main( )
{
    int x,y;
    scanf("%d",&x);
    y=0;
    if(x>=0)
        if(x>0) y=1;
    else y=-1;
    printf("x=%d,y=%d\n",x,y);
    return 0;
}
```



例 3.14 考虑下面程序输出结果:

```
#include<stdio.h>
int main()
{
    int x=100,a=10,b=20;
    int v1=5,v2=0;
    if(a<b)
        if(b!=15)
            if(!v1)
                x=1;
            else
                if(v2) x=10;
                else x=-1;
    printf ("x=%d\n",x);
    return 0;
}
```

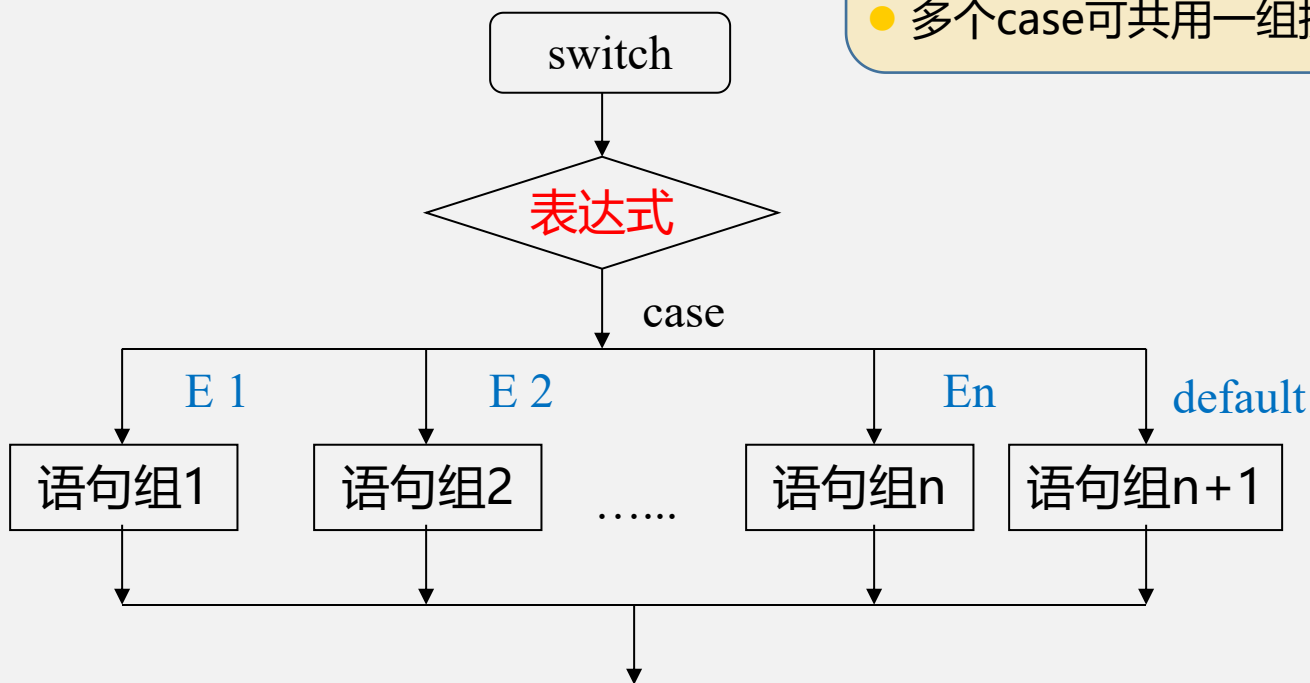


switch语句（开关分支语句）

```
switch(表达式){  
    case 常量表达式 1 : 语句序列 1; break;  
    case 常量表达式 2 : 语句序列 2; break;  
    ... ..  
    case 常量表达式 n : 语句序列 n; break;  
    default : 语句序列n+1;  
}
```

说明:

- E1,E2,...En是**常量表达式**,且值必须**互不相同**
- 语句标号作用, 必须用**break**跳出
- case后可包含多个可执行语句, 且不必加{ }
- switch可**嵌套**
- 多个case可共用一组执行语句





例 3.15 输入分数给出相应等级。

```
#include<stdio.h>
void main()
{
    int score;
    printf("score=");
    scanf("%d",&score);
    switch(score/10){
        case 10:
        case 9:printf("优秀\n");break;
        case 8:printf("良好\n");break;
        case 7:printf("中等\n");break;
        case 6:printf("及格\n");break;
        default:printf("不及格\n");
    }
}
```

?

如果没有break会出现什么情况

根据输入的score的值跳转到不同的case之后输出相应的结果



例 3.16 通过switch实现菜单选择

```
for(;;) //有用的死循环——反复显示菜单
{
    printf("成绩管理系统\n");
    printf("1.录入成绩\n");
    printf("2.修改成绩\n");
    printf("3.查询成绩\n");
    printf("0.退出系统\n");
    printf("请选择 0-3:\n");
    scanf("%d",&num); //输入菜单选项编号
    switch(num) { //也可用字符'0' '1'
        case 1: //录入成绩的语句组
            break;
        case 2: //修改成绩的语句组
            break;
        case 3: //查询成绩的语句组
            break;
        case 0: return 0; //退出程序
    }
}
```



例 3.17 编程实现对若干个乘客计收行李托运费。当输入乘客行李重量小于等于零时结束程序运行。用 w 表示行李重量， d 表示每公斤的托运费率， f 表示托运费。根据重量不同，托运费率核收规则如下：

行李重量 (kg)	托运费率
$w < 20$	免费
$20 \leq w < 60$	0.15
$60 \leq w < 100$	0.20
$100 \leq w < 160$	0.25
$w \geq 160$	0.30

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

```
void main()
```

```
{
```

```
    double f,d,w;
```

```
    int k;
```

```
    printf("输入行李重量:");
```

```
    scanf("%lf",&w);
```

```
    while(w>0){
```

```
        k=(int)(w/20);
```

```
        switch(k){
```

```
            case 0:printf("免费! \n");d=0.0;break;
```

```
            case 1:
```

```
            case 2:d=0.15;break;
```

```
            case 3:
```

```
            case 4:d=0.20;break;
```

```
            case 5:
```

```
            case 6:
```

```
            case 7:d=0.25;break;
```

```
            default :d=0.30;
```

```
        }
```

```
        f=w*d;
```

```
        printf("行李重量: %lf,托运费率: %lf,行李托运费: %lf\n",w,d,f);
```

```
        printf("输入行李重量: ");
```

```
        scanf("%lf",&w);
```

```
    }
```

```
}
```

输入一个行李重量

根据行李重量的值执行循环，只要大于0就继续输入下一个

将行李重量取整

根据行李取整后的值计算托运费率

继续输入下一个乘客的行李重量



循环控制结构

C语言可实现循环的语句：

while 语句

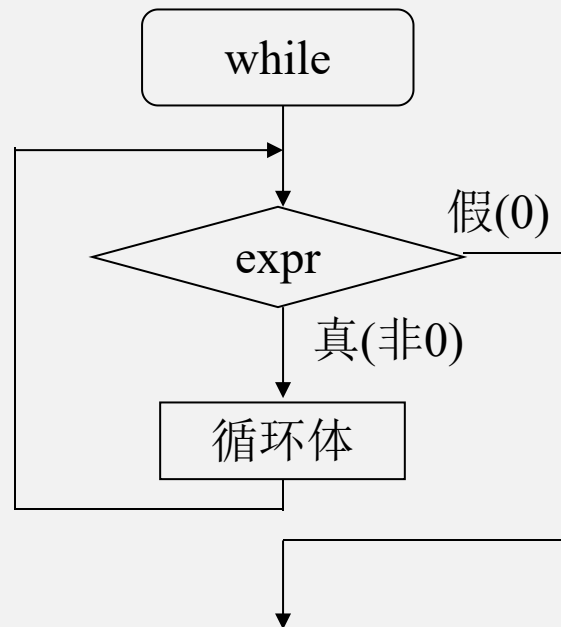
do ~ while 语句

for 语句



while循环语句

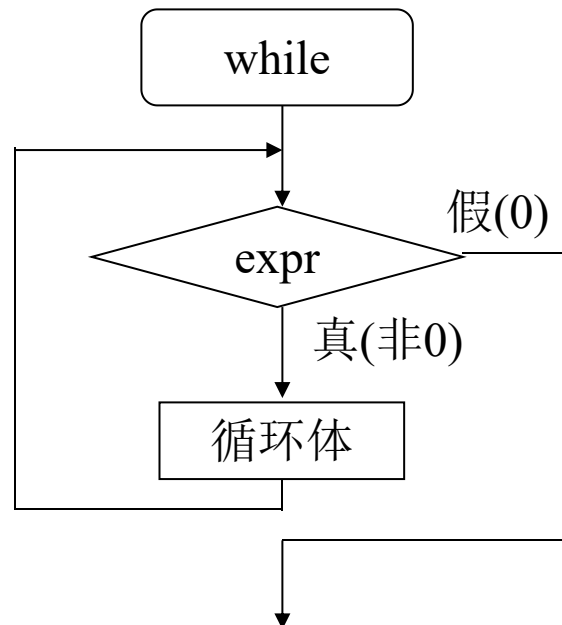
```
while(表达式){  
    循环体语句组;  
}
```





while循环语句

```
while(表达式){  
    循环体语句组;  
}
```



```
#include<stdio.h>  
void main()  
{  
    int i=1;  
    float sum=0;  
    while(i<=100){  
        sum=sum+1.0/i;  
        i++;  
    }  
    printf("sum=%6.2f\n",sum);  
}
```

//i作是循环控制变量，置初值为1
//作为存储累加和的变量一般初始化为0
//当i小于等于100时执行循环体，大于100时结束循环



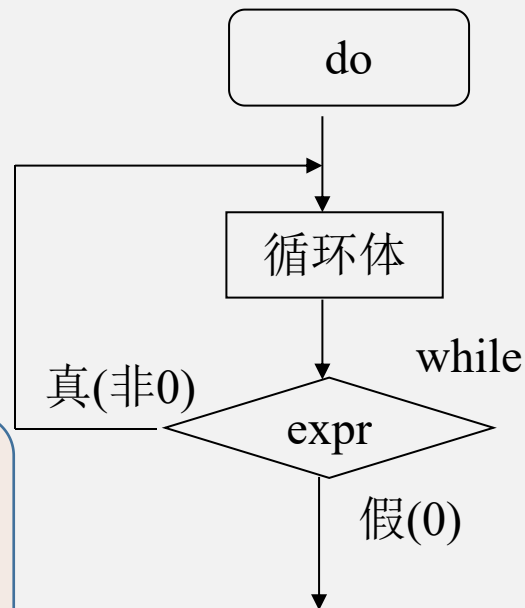
do~while循环语句

```
do{  
    循环体语句组;  
}while(表达式)
```

```
#include <stdio.h>  
main()  
{  
    int i,sum=0;  
    scanf("%d",&i);  
    do  
    {  
        sum+=i;  
        i++;  
    }while(i<=10);  
    printf("%d",sum);  
}
```

while和do~while的区别?

```
#include <stdio.h>  
main()  
{  
    int i,sum=0;  
    scanf("%d",&i);  
    while(i<=10)  
    {  
        sum+=i;  
        i++;  
    }  
    printf("%d",sum);  
}
```





例 3.18 整数按位倒序输出

```
#include <stdio.h>
int main( )
{
    int n;
    scanf("%d",&n);
    do {
        printf("%d",n%10);
        n=n/10;
    } while(n!=0); //末尾的分号必不可少
}
```

输出最低位

将源操作数减少一位



例 3.19 求阶乘

先来看最简单的要求，求一个给定数字的阶乘，比如10！

```
#include<stdio.h>
int main()
{
    int i;
    long fact=1;
    for(i=2;i<=10;i++)
        fact=fact*i;
    printf("10!=%ld\n", fact);
    return 0;
}
```

定义循环增量*i*和存放结果变量fact

for循环求出10!



例 3.19 求阶乘

先来看最简单的要求，求一个给定数字的阶乘，比如10！

提高一些难度，让用户输入一个数字n，求n！

```
#include<stdio.h>
int main()
{
    int i, n;
    long fact=1;
    printf("input n:");
    scanf("%d", &n);
    for(i=2; i<=n; i++)
        fact=fact*i;
    printf("%d!=%ld\n", n, fact);
    return 0;
}
```

用户输入n的值



例 3.19 求阶乘

先来看最简单的要求，求一个给定数字的阶乘，比如10!

提高一些难度，让用户输入一个数字n，求n!

再复杂一些，用户输入0~100之间的数字n，只要在此范围内，就一直求n!

```
#include<stdio.h>
int main()
{
    int i,n;
    long fact=1.0;
    printf("plz input n:");
    scanf("%d",&n);
    while(n>0)
    {
        for(i=2;i<=n;i++)
            fact=fact*i;
        printf("%d!=%ld\n",n,fact);
        printf("plz input n:");
        scanf("%d",&n);
    }
    return 0;
}
```

? 此程序有没有问题



例 3.19 求阶乘

```
#include<stdio.h>
int main()
{
    int i,n;
    double fact=1.0;
    for(i=0;i<=7;i++) printf("\n");
    for(i=0;i<=79;i++) printf("*");
    printf("\n\n");
    for(i=0;i<=25;i++) printf(" ");
    printf("The Factorial Program\n\n\n");
    for(i=0;i<=79;i++) printf("*");
    printf("press enter to continue");
    getchar();system("cls");
    printf("input n(0~100,0 as end):");
    scanf("%d",&n);
    while(n!=0)
    {
        if(n<0||n>100) {
            printf("error number!\n\n");
            printf("input n(0~100,0 as end):");
            scanf("%d",&n);
        }
        else
        {
            for(i=2,fact=1;i<=n;i++)
                fact=fact*i;
            printf("%d!=%le\n\n",n,fact);
            printf("input n(0~100,0 as end):");
            scanf("%d",&n);
        }
    }
    system("cls");
    for(i=0;i<=7;i++) printf("\n");
    for(i=0;i<=79;i++) printf("*");
    printf("\n\n\n");
    for(i=0;i<=30;i++) printf(" ");
    printf("The End\n\n\n");
    for(i=0;i<=79;i++) printf("*");
    return 0;
}
```

界面

判断输入数据合法性

计算结果并进行下一次
数据输入

界面



多层循环

例 3.20 两个数组的操作：寻找相同的元素

```
#include <stdio.h>
int main()
{
    int a[5]={3,5,7,9,11},i,j;
    int b[6]={1,2,3,4,5,6},found=0;
    for(i=0;i<5;i++)
        for(j=0;j<6;j++)
            if(a[i]==b[j]) found=1;
    if(found)
        printf("a 与 b 有相同的元素");
    else
        printf("a 与 b 没有相同的元素");
    return 0;
}
```

通过双重循环依次比较两个数组中的元素



break的应用

例 3.21 两个数组的操作：寻找相同的元素

```
#include <stdio.h>
int main()
{
    int a[5]={3,5,7,9,11},i,j;
    int b[6]={1,2,3,4,5,6},found=0;
    for(i=0;i<5;i++) {
        printf("i=%d\t",i);
        for(j=0;j<6;j++) {
            printf("j=%d\t",j);
            if(a[i]==b[j]) {
                found=1; break;
            }
        }
        printf("\n");
    }
    if(found)
        printf("a 与 b 有相同的元素");
    else
        printf("a 与 b 没有相同的元素");
    return 0;
}
```

内层循环中加入了
对比较结果的判断



continue语句

功能：结束本次循环，跳过循环体中**尚未执行的语句**，进行下一次是否执行循环体的判断。

说明：仅用于循环语句中。

例 3.22 求输入的十个整数中正数的个数及其平均值

```
#include <stdio.h>
int main()
{
    int i,num=0,a;
    float sum=0;
    for(i=0;i<10;i++)
    {
        scanf("%d",&a);
        if(a<=0) continue;
        num++;
        sum+=a;
    }
    printf("%d plus integer's sum :%6.0f\n",num,sum);
    printf("Mean value:%6.2f\n",sum/num);
    return 0;
}
```

continue语句终止本次循环



基本算法：穷举法

例 3.23 百元买百鸡问题：公鸡5元一只，母鸡3元一只，小鸡一元三只，一百元买三种，有哪些组合？

```
for(x=0;x<=20;x++)  
    for(y=0;y<33;y++)  
        for(z=0;z<=100;z++)  
            if((x+y+z==100)&&(5*x+3*y+z/3==100)&&(z%3==0))  
                printf("公鸡数%4d,母鸡数%4d,小鸡数%4d\n",x,y,z);
```

```
for(x=0;x<=20;x++)  
    for(y=0;y<33;y++) {  
        z=100-x-y;  
        if((5*x+3*y+z/3==100)&&(z%3==0))  
            printf("公鸡数: %4d ,母鸡数: %4d ,小鸡数: %4d\n",x,y,z);}
```



基本算法：穷举法

例 3.24 编程实现求3~100之间的所有素数。

- 如何判断一个数num是否是素数：用 $2 \sim \sqrt{num}$ 和num相除，出现余数为0则说明不是素数，反之则是素数；
- 循环运行上一步中的判断程序依次判断3~100是否是素数并输出。

```
#include<stdio.h>
#include<math.h>
int main()
{
    int num,n;
    printf("enter num:");
    scanf("%d",&num);
    for(n=2;n<=sqrt(num);n++)
        if(num%n==0) break;
    if(n>sqrt(num))printf("%d is a prime number\n",num);
    else printf("%d is not a prime number\n",num);
    return 0;
}
```

？ 循环结束后如何判断是否是素数



基本算法：穷举法

例 3.24 编程实现求3~100之间的所有素数。

- 如何判断一个数num是否是素数：用 $2 \sim \sqrt{num}$ 和num相除，出现余数为0则说明不是素数，反之则是素数；
- 循环运行上一步中的判断程序依次判断3~100是否是素数并输出。

```
#include<stdio.h>
#include<math.h>
int main()
{
    int num,n,flag=1;
    printf("input num:");
    scanf("%d",&num);
    for(n=2;n<=sqrt(num);n++)
        if(num%n==0){flag=0;break;}
    if(flag==1)printf("%d is a prime number\n",num);
    else printf("%d is not a prime number\n",num);
    return 0;
}
```

利用变量flag的值



基本算法：穷举法

例 3.24 编程实现求3~100之间的所有素数。

- 如何判断一个数num是否是素数：用 $2 \sim \sqrt{num}$ 和num相除，出现余数为0则说明不是素数，反之则是素数；
- 循环运行上一步中的判断程序依次判断3~100是否是素数并输出。

```
#include<stdio.h>
#include<math.h>
int main()
{
    int num,n;
    for(num=3;num<=100;num++)
    {
        for(n=2;n<=sqrt(num);n++)
            if(num%n==0) break;
        if(n>sqrt(num))printf("%d is a prime number\n",num);
    }
    return 0;
}
```

内层循环的作用



基本算法：穷举法

例 3.25 简单推理问题

有四位同学中的一位做了好事，未留名，表扬信来了之后，校长问这四位是谁做的好事。

- A 说：不是我
- B 说：是 C
- C 说：是 D
- D 说：他胡说

已知三个人说的是真话，一个人说的是假话。现在要根据这些信息，找出做了好事的人。

```
for(k=0; k<4; k=k+1)
{
    theone='A'+k;
    sum=(theone!='A')+(theone=='C')+(theone=='D')+(theone!='D');
    if(sum==3) {
        flag=1;
        printf("This man is %c\n",theone);
        break;
    }
}
if(flag==0) printf("No found!\n");
```

4人依次验证

根据4人发言得出的表达式

根据题意进行判断



基本算法：递推法

例 3.26 斐波那契数列

```
#include <stdio.h>
int main()
{
    int i, f1=1, f2=1;
    for(i=1; i<=20; i++)
    {
        printf("%12d%12d", f1, f2);
        if(i%4==0) printf("\n");
        f1=f1+f2; f2=f2+f1;
    }
    return 0;
}
```

```
#include <stdio.h>
int main()
{
    int i, fib[40];
    for(i=2; i<=39; i++)
        fib[i]=fib[i-1]+fib[i-2];
    for(i=0; i<=39; i++){
        if(i%4==0) putchar('\n');
        printf("%12d", fib[i]);
    }
    return 0;
}
```



基本算法：贪心算法

例 3.27 最少纸币支付问题

假设有七种面值的纸币：1 元、2 元、5 元、10 元、20 元、50 元、100 元。某人手里对应上述面值的纸币分别有 5, 5, 0, 0, 2, 1, 3 张。现在要用这些钱来支付特定的金额，问最少要用多少张纸币。

```
#define N 7
#include <stdio.h>
int main()
{
    int value[N]={1,2,5,10,20,50,100};
    int a[N]={5,5,0,0,2,1,3};
    int money,i,num=0,c=0;
    printf("请输入需要支付的总额:");
    scanf("%d",&money);
    for(i=N-1;i>=0;i--) {
        if(money/value[i]<=a[i]) c=money/value[i];
        else c=a[i];
        money=money-c*value[i]; num+=c;
    }
    if(money>0) num=-1;
    if(num!=-1) printf("%d\n",num);
    else printf("无解!");
    return 0;
}
```

纸币的面值和数量，用数组存放

纸币面值数组从小到大存放，按照贪心算法从大到小来计算和匹配数量。



3.5 数组与批量数据处理

一维数组的定义

- 数组的名称
- 数组的大小（长度/数组元素的个数）
- 数组的基类型（元素的类型）

[] :数组运算符
单目运算符
优先级(1)
左结合
不能用()

定义方式： 数据类型 数组名[整形常量表达式];

例 `int a[6];`

合法标识符

a

表示元素个数
下标从0开始

数组名表示内存首地址
是地址常量

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

编译时分配连续内存
内存字节数 =
数组长度 * sizeof(元素数据类型)



常用的排序算法

❖ 插入排序

❖ 选择排序

❖ 冒泡排序

❖ 快速排序

❖ 堆排序

❖ 归并排序

❖ 基数排序

❖ 希尔排序



常用的排序算法

❖ 插入排序

❖ 选择排序

❖ 冒泡排序

❖ 快速排序

❖ 堆排序

❖ 归并排序

❖ 基数排序

❖ 希尔排序



插入排序

- 插入排序 (**Insertion Sort**)，一般也被称为直接插入排序。对于少量元素的排序，它是一个有效的算法。
- 插入排序是一种最简单的排序方法，它的基本思想是将一个记录插入到已经排好序的有序表中，从而一个新的、记录数增1的有序表。



插入排序

- ① 从第一个数据开始，该数据可以直接作为已经排序的数组元素
- ② 读入新数据，在已经排序的元素序列中从后向前扫描
- ③ 如果该元素大于新数据，将该元素后移一个位置
- ④ 重复步骤③，直到找到已排序的元素小于或者等于新数据的位置
- ⑤ 将新数据放入该位置
- ⑥ 重复步骤②~⑤



插入排序

数组

-1	0	0	0	0
----	---	---	---	---

输入n

5

数组

5	0	0	0	0
---	---	---	---	---

第一轮

数组

5	0	0	0	0
---	---	---	---	---

输入n

8

数组

5	8	0	0	0
---	---	---	---	---

第二轮

数组

5	8	0	0	0
---	---	---	---	---

输入n

3

数组

3	5	8	0	0
---	---	---	---	---

第三轮



插入排序

数组

3

5

8

0

0

输入n

4

数组

3

4

5

8

0

第四轮

数组

3

4

5

8

0

输入n

-6

数组

-6

3

4

5

8

第五轮



插入排序

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

```
#define N 5
```

```
int main()
```

```
{
```

```
    int a[N]={-1},n,i,j,k;
```

```
    for(i=0;i<N;i++) {
```

```
        printf("Input data:\n");  
        scanf("%d", &n);
```

输入待插入的数据

```
        for(j=i;j>0;j--)  
            if(a[j-1]>n) a[j]=a[j-1];  
            else break;
```

将当前数组从后向前与待
插入数据比较大小并移动
位置

```
        a[j]=n;
```

```
        for(j=0;j<=i;j++) printf("%d\t",a[j]);  
        putchar('\n');
```

移动完毕后插入数据

```
    }
```

```
    return 0;
```

```
}
```



插入排序

```
#include<stdio.h>
#define N 5
int main()
{
    int a[N]={-1},n,i,j,k;
    for(i=0;i<N;i++) {
        printf("Input data:\n");
        scanf("%d", &n);
        for(j=i;j>0;j--)
            if(a[j-1]>n) a[j]=a[j-1];
            else break;
        a[j]=n;
        for(j=0;j<=i;j++) printf("%d\t",a[j]);
        putchar('\n');
    }
    return 0;
}
```



针对已有数组
如何插入排序



冒泡排序

- 冒泡排序（**Bubble Sort**），是一种较简单的排序算法。
- 通过无序区中相邻记录关键字间的“**比较**”和位置的“**交换**”，实现关键字较大（或较小）的记录向序列“一端”移动，从而达到记录按关键字递增（或递减）顺序排列的目的。
- 这个算法的名字由来是因为越大（或越小）的元素会经由交换慢慢“浮”到数列的顶端，故名。



冒泡排序

n=8

38	38	38	38	13	13	13	13	13
49	49	49	13	27	27	27	27	27
65	65	65	27	30	30	30		30
76	13	13	30	38	38			38
13	27	27	49	49				49
27	30	30	65					65
30	76	65						76
97	97							97
初始序列	第一趟	第二趟	第三趟	第四趟	第五趟	第六趟	第七趟	排序结果



冒泡排序

排序过程：

- 比较第一个数与第二个数，若为逆序 $a[0] > a[1]$ ，则交换
- 依次类推，比较第二个数与第三个数，直至第 $n-1$ 个数和第 n 个数比较为止 — 第一趟冒泡排序结束，结果最大的数被安置在最后一个元素位置上。
- 对前 $n-1$ 个数进行第二趟冒泡排序，结果使次大的数被安置在第 $n-1$ 个元素位置。
- 重复上述过程，共经过 $n-1$ 趟冒泡排序后，排序结束。



冒泡排序

```
#include<stdio.h>
int main()
{
    int a[11],i,j,t;
    printf("Input 10 Numbers:\n");
    for(i=1;i<=10;i++)
        scanf("%d",&a[i]);
    putchar('\n');
    for(j=1;j<=9;j++)
        for(i=1;i<=10-j;i++)
            if(a[i]>a[i+1])
                {t=a[i];a[i]=a[i+1];a[i+1]=t;}
    printf("output:\n");
    for(i=1;i<=10;i++)
        printf("%5d",a[i]);
    return 0;
}
```

定义数组、输入数组元素
数据

利用双重循环，两两比
较数组相邻元素，9轮过
后排序结束



冒泡排序



几个小问题

❖ 如何排逆序?

```
#include<stdio.h>
int main()
{
    int a[11],i,j,t;
    printf("Input 10 Numbers:\n");
    for(i=1;i<=10;i++)
        scanf("%d",&a[i]);
    putchar('\n');
    for(j=1;j<=9;j++)
        for(i=1;i<=10-j;i++)
            if(a[i]>a[i+1]) → if(a[i]<a[i+1])
                {t=a[i];a[i]=a[i+1];a[i+1]=t;}
    printf("output:\n");
    for(i=1;i<=10;i++)
        printf("%5d",a[i]);
    return 0;
}
```




冒泡排序



几个小问题

❖ 如何排逆序?

❖ 改进

10 9 8 7 6 5 4 3 2 1 0

1 0 2 3 4 5 6 7 8 9 10



冒泡排序

```
#include<stdio.h>
int main()
{
    int a[11],i,j,t;
    int flag;
    printf("Input 10 Numbers:\n");
    for(i=1;i<=10;i++)
        scanf("%d",&a[i]);
    putchar('\n');
    for(j=1;j<=9;j++)
    {
        flag=1;
        for(i=1;i<=10-j;i++){
            if(a[i]>a[i+1]){
                t=a[i];a[i]=a[i+1];a[i+1]=t;
                flag=0;
            }
        }
        if(flag==1) break;
    }
}
```

定义flag变量

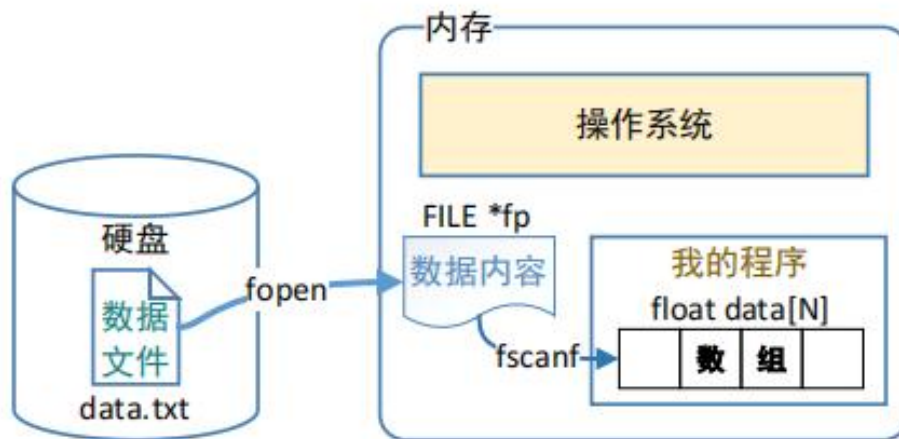
利用flag变量判断是否
有交换从而结束排序



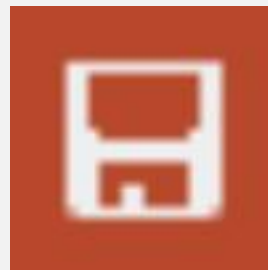
从文件读取数据

```
fscanf( file , "%d" , &a[i] );
```

```
scanf ( "%d" , &a[i] );
```









从文件读取数据

- ❖ C文件操作用库函数实现,包含在stdio.h
- ❖ 文件使用方式:打开文件-->文件读/写-->关闭文件
- ❖ 系统自动打开和关闭三个标准文件:

- 标准输入-----键盘 `stdin`
- 标准输出-----显示器 `stdout`
- 标准出错输出-----显示器 `stderr`

❖ 打开文件fopen

- 函数原型: `FILE *fopen(char *name,char *mode)`
- 功能: 按指定方式打开文件
- 返回值: 正常打开, 为指向文件结构体的指针, 打开失败, 为NULL

使用文件方式

要打开的文件名

```
例 FILE *fp;  
    fp= fopen ("test.dat","r");
```



从文件读取数据

文件使用方式	含义
"r/rb" (只读)	为输入打开一个文本/二进制文件
"w/wb" (只写)	为输出打开或建立一个文本/二进制文件
"a/ab" (追加)	向文本/二进制文件尾追加数据
"r+/rb+" (读写)	为读/写打开一个文本/二进制文件
"w+/wb+" (读写)	为读/写建立一个文本/二进制文件
"a+/ab+" (读写)	为读/写打开或建立一个文本/二进制文件



从文件读取数据

例 文件打开与测试

```
FILE *fp;  
if((fp=fopen("test.c", "w"))==NULL)  
{  
    printf("File open error!\n");  
    exit(0);  
}
```

例 文件打开与测试

```
FILE *fp;  
fp=fopen("test.c", "w");  
if(fp==NULL)  
{  
    printf("File open error!\n");  
    exit(0);  
}
```




从文件读取数据

格式化读写 : fscanf 与 fprintf

❖ 函数原型:

```
int fscanf(FILE *fp,const char *format[,address,...])
```

```
int fprintf(FILE *fp,const char *format[,argument,...])
```

功能: 按格式对文件进行I/O操作

❖ 返回值: 成功,返回I/O的个数;出错或文件尾,返回EOF



从文件读取数据

例 3.28 从键盘按格式输入数据存到磁盘文件中

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

```
int main()
```

```
{
```

```
    char s[80],c[80];
```

```
    int a,b;
```

```
    FILE *fp;
```

```
    if((fp=fopen("test.txt","w"))==NULL)
    { printf("can't open file\n"); exit(0); }
```

```
    fscanf(stdin,"%s%d",s,&a);
```

```
    fprintf(fp,"%s %d",s,a);
```

```
    fclose(fp);
```

```
    if((fp=fopen("test.txt","r"))==NULL)
    { printf("can't open file\n"); exit(0); }
```

```
    fscanf(fp,"%s%d",c,&b);
```

```
    fprintf(stdout,"%s %d",c,b);
```

```
    fclose(fp);
```

```
    return 0;
```

```
}
```

打开文件

从键盘输入
向文件输出

关闭文件

再次打开文件

从文件输入
向显示器输出



多维数组

二维数组的定义

定义方式:

数据类型 数组名[常量表达式][常量表达式];

❖ 数组元素的存放顺序

- 原因:内存是**一维**的
- 二维数组: 按行序优先
- 多维数组: 最右下标变化最快

int c[2][3][4]

行数

列数

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

例 int a[3][4];

float b[2][5];

int c[2][3][4];

int a[3,4];

int a[3][2]

$\begin{pmatrix} a[0][0] & a[0][1] \\ a[1][0] & a[1][1] \\ a[2][0] & a[2][1] \end{pmatrix}$

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

20	c[1][2][0]
21	c[1][2][1]
22	c[1][2][2]
23	c[1][2][3]



例 3.29 将二维数组行列元素互换，存到另一个数组中

```
#include <stdio.h>
main()
{ int a[2][3]={{1,2,3},{4,5,6}};
  int b[3][2],i,j;
  printf("array a:\n");
  for(i=0;i<=1;i++)
  { for(j=0;j<=2;j++)
    { printf("%5d",a[i][j]);
      b[j][i]=a[i][j];
    }
    printf("\n");
  }
}
```

$$a = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad b = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

```
printf("array b:\n");
for(i=0;i<=2;i++)
{ for(j=0;j<=1;j++)
  { printf("%5d",b[i][j]);
    }
  printf("\n");
}
```



例 3.30 计算两个矩阵相乘

矩阵相乘的条件： $A \times B$ **矩阵A的列数要与B的行数相等** $A[i][k] \times B[k][j] = C[i][j]$

记作 $C=AB$,其中矩阵C中的第i行第j列元素可以表示为：

$$(AB)_{ij} = \sum_{k=1}^p a_{ik} b_{kj} = a_{i1} b_{1j} + a_{i2} b_{2j} + \cdots + a_{ip} b_{pj}$$

$$C = AB = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} = \begin{pmatrix} 1 \times 1 + 2 \times 2 + 3 \times 3 & 1 \times 4 + 2 \times 5 + 3 \times 6 \\ 4 \times 1 + 5 \times 2 + 6 \times 3 & 4 \times 4 + 5 \times 5 + 6 \times 6 \end{pmatrix} = \begin{pmatrix} 14 & 32 \\ 32 & 77 \end{pmatrix}$$

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

```
int main()
```

```
{
```

```
    int a[2][3]={1,2,3},{4,5,6}};
```

```
    int b[3][2]={1,4},{2,5},{3,6}};
```

```
    int c[2][2],i,j,k,s;
```

```
    for(i=0;i<2;i++)
```

```
        for(j=0;j<2;j++)
```

```
        {
```

```
            for(k=0;k<3;k++)
```

```
                s=s+a[i][k]*b[k][j];
```

```
            c[i][j]=s;
```

```
        }
```

```
    for(i=0;i<2;i++)
```

```
        for(j=0;j<2;j++)
```

```
            printf("%6d%c",c[i][j],((j+1)%2==0)?'\n':' ');
```

```
}
```

**三层循环计算
矩阵相乘**



例 3.31 读入下表中值到数组，分别求各行、各列及表中所有数之和

12	4	6
8	23	3
15	7	9
2	5	17

感觉很简单?

程序怎么写?

简单粗暴的写法

```
int array[4][3]={12,4,6,8,23,3,15,7,9,2,5,17};
int row_sum[4],column_sum[3],all_sum;
row_sum[0]=12+4+6;
row_sum[1]=8+23+3;
row_sum[2]=15+7+9;
row_sum[4]=2+5+17;
column_sum[0]=12+8+15+2;
column_sum[1]=4+23+7+5;
column_sum[2]=6+3+9+17;
all_sum=12+4+6+8+23+3+15+7+9+2+5+17;
```



例 3.28 读入下表中值到数组，分别求各行、各列及表中所有数之和

12	4	6	
8	23	3	
15	7	9	
2	5	17	

有没有问题?

```
#include<stdio.h>
main()
{
    int array[5][4],i,j;
    for(i=0;i<4;i++)
        for(j=0;j<3;j++)
        {
            printf("%d row,%d column:",i+1,j+1);
            scanf("%d",&array[i][j]);
        }
    for(i=0;i<4;i++)
        for(j=0;j<3;j++)
        {
            array[i][3]+=array[i][j];
            array[4][j]+=array[i][j];
            array[4][3]+=array[i][j];
        }
    for(i=0;i<5;i++)
    {
        for(j=0;j<4;j++)
            printf("%5d\t",array[i][j]);
        putchar('\n');
    }
}
```



例 3.28 读入下表中值到数组，分别求各行、各列及表中所有数之和

12	4	6	
8	23	3	
15	7	9	
2	5	17	

有没有问题?

```
#include<stdio.h>
main()
{
    int array[5][4],i,j;
    for(i=0;i<4;i++)
        for(j=0;j<3;j++)
        {
            printf("%d row,%d column:",i+1,j+1);
            scanf("%d",&array[i][j]);
        }
    for(i=0;i<4;i++)    array[i][3]=0;
    for(j=0;j<4;j++)    array[4][j]=0;
    for(i=0;i<4;i++)
        for(j=0;j<3;j++)
        {
            array[i][3]+=array[i][j];
            array[4][j]+=array[i][j];
            array[4][3]+=array[i][j];
        }
    for(i=0;i<5;i++)
    {
        for(j=0;j<4;j++)
            printf("%5d\t",array[i][j]);
        putchar('\n');
    }
}
```




字符数组

字符串常量

定义：用双引号 “ ” 括起来的字符序列

存储：每个字符串尾自动加一个 '\0' 作为字符串结束标志

例 字符串 “hello” 在内存中

h	e	l	l	o	\0
---	---	---	---	---	----

例 空串 ""

\0

❖ 字符常量与字符串常量不同

例 'a'

a

"a"

a	\0
---	----

例: char ch; ch="A"; ❌

例: char ch; ch='A';



字符数组

字符数组和字符串的关系

- 定义
- 字符数组的初始化
 - 逐个字符赋值
 - 用字符串常量

例 char c[10], ch[3][4];

char ch[5]={‘H’,‘e’,‘l’,‘l’,‘o’};

char ch[5]={‘B’,‘o’,‘y’};

char ch[6]={“Hello”}; char ch[6]=“Hello”;
char ch[]=“Hello”;

有问题!

H	e	l	l	o	\0
ch[0]	ch[1]	ch[2]	ch[3]	ch[4]	ch[5]



字符数组

字符数组和字符串的关系

- 定义
- 字符数组的初始化
 - 逐个字符赋值
 - 用字符串常量
- 用库函数对字符数组赋值

```
char name[16];  
scanf("%s",name); 问题：无法接收输入空格
```

```
getchar() gets()
```



字符数组

例 3.32 用getchar()完成字符串输入

```
#include<stdio.h>
main()
{
    char name[16],c;
    int i=0;
    while((c=getchar())!='\n')
        name[i++]=c;
    printf("%s",name);
}
```



是否能实现
一串字符串
的输入



字符数组

例 3.33 字符串比较

```
#include<stdio.h>
#include<string.h>
#define N 15
int main() {
    char pwd[N]="zhimakaimen";
    char str[N]={0};
    int i,j;
    for(j=0;j<5;j++){
        printf("input password:\n");
        scanf("%s",str);
        for(i=0;str[i]==pwd[i];i++);
        if(i==N) {
            printf("welcome to our world.");
            return 0;
        }
        else printf("warning!incorrect!\n");
    }
    printf("fail to enter.\n");
    return 0;
}
```

通过循环判断
两个数组是否
一致



结构体与复杂信息处理

- 构造数据类型，不同类型的数据需要统一处理
- 例如学生信息，需要学号、姓名、性别、年龄、班级等等，数据类型不一致，使用数组会难以实现
- 结构体是一种构造数据类型，把不同类型的数据组合成一个整体
- 结构体类型的定义

合法标识符，可省略
无名结构体

struct

结构体名

struct是关键字，
不能省略

类型标识符
类型标识符
.....

成员名;
成员名;
.....

成员类型可以是
基本型或构造型

};



结构体与复杂信息处理

```
struct student{  
    long no;  
    char name[16];  
    char sex;  
    int age;  
    float score;  
    char addr[30];  
};
```

- 每个成员必须有自己的数据类型，定义后的需要有分号
- 相同类型、位置连续的结构成员可定义在一起，如

```
struct date{  
    int month , day , year;  
}
```
- 结构体类型定义的位置既可以在函数内部，也可以在函数外部
- 结构体类型的定义描述结构的组织形式，不分配内存



结构体与复杂信息处理

结构体变量的定义

- 先定义结构体类型，再定义结构体变量

```
struct 结构体名
{
    类型标识符 成员名;
    类型标识符 成员名;
    .....
};
struct 结构体名 变量名表列;
```

```
例 struct student{
    long no;
    char name[16];
    char sex;
    int age;
    float score;
    char addr[30];
};
struct student stu1,stu2;
```




结构体与复杂信息处理

结构体变量的定义

- 定义结构体类型同时结构体变量

```
struct  结构体名
{
    类型标识符  成员名;
    类型标识符  成员名;
    .....
} 变量名表列;
```

```
例 struct student{
    long no;
    char name[16];
    char sex;
    int age;
    float score;
    char addr[30];
}stu1,stu2;
```



结构体与复杂信息处理

结构体变量的定义

- 直接定义结构体变量（省略结构体名）

```
struct
{
    类型标识符  成员名;
    类型标识符  成员名;
    .....
} 变量名表列;
```

例

```
struct {
    long no;
    char name[16];
    char sex;
    int age;
    float score;
    char addr[30];
}stu1,stu2;
```



结构体与复杂信息处理

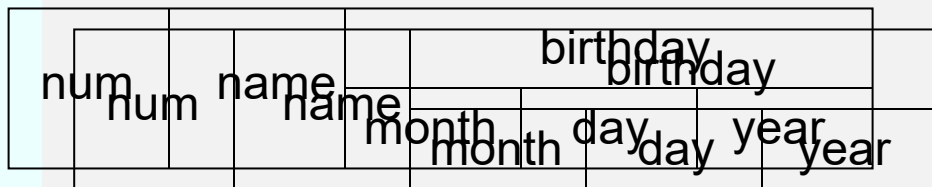
❖ 结构体类型与结构体变量概念不同

- 类型:不分配内存; 变量:分配内存
- 类型:不能赋值、存取、运算; 变量:可以

❖ 结构体可嵌套

❖ 结构体成员名与程序中变量名可相同，不会混淆

```
struct student
{
    int num;
    char name[20];
    struct date
    {
        int month;
        int day;
        int year;
    } birthday;
} stu;
} stu;
```





结构体与复杂信息处理

结构体变量的引用

❖ 结构体变量不能整体引用，只能引用变量成员

引用方式： 结构体变量名 . 成员名

例 struct student

{ long num;

char name[20];

char sex;

int age;

float score;

char addr[30];

} stu1, stu2;

stu1.num=2016003;

stu1.score=85.5;

stu1.score+=stu2.score;
stu1.age++;



结构体与复杂信息处理

结构体变量的引用

❖ 结构体变量不能整体引用，只能引用变量成员

引用方式： 结构体变量名 . 成员名

```
例 struct student
{ long num;
  char name[20];
  char sex;
  int age;
  float score;
  char addr[30];
} stu1,stu2;
```

printf(“%d,%s,%c,%d,%f,%s\n”,stu1);

×

stu1={101,“Wan Lin”,‘M’,19,87.5,“DaLian”};

×



结构体与复杂信息处理

结构体变量的引用

❖ 结构体变量不能整体引用，只能引用变量成员

引用方式： 结构体变量名 . 成员名

```
例 struct student
{ long num;
  char name[20];
  char sex;
  int age;
  float score;
  char addr[30];
} stu1, stu2;
```

if(stu1==stu2)

.....

×



结构体与复杂信息处理

结构体变量的引用

- ❖ 结构体变量不能整体引用，只能引用变量成员

引用方式： 结构体变量名 . 成员名

- ❖ 可以将一个结构体变量赋值给另一个结构体变量
- ❖ 结构体嵌套时逐级引用

```
例 struct student
{ long num;
  char name[20];
  char sex;
  int age;
  float score;
  char addr[30];
} stu1, stu2;
```

stu2=stu1; ✓



结构体与复杂信息处理

结构体变量的引用

- ❖ 结构体变量不能整体引用，只能引用变量成员

引用方式： 结构体变量名 . 成员名

- ❖ 可以将一个结构体变量赋值给另一个结构体变量
- ❖ 结构体嵌套时逐级引用

```
例 struct student
{ long num;
  char name[20];
  struct date
  { int month;
    int day;
    int year;
  } birthday;
} stu1, stu2;
```

stu1.birthday.month=12;

num	name	birthday		
		month	day	year



结构体与复杂信息处理

例 3.34 定义结构体，求其所需内存空间大小

```
#include<stdio.h>
int main()
{
    struct student
    {
        int number;
        char name[6];
        char sex;
        int age;
        char address[20];
    };
    printf("%d\n ",sizeof(struct student));
    return 0;
}
```



结构体与复杂信息处理

结构体变量的初始化和赋值

形式一

```
struct 结构体名
{
    类型标识符 成员名;
    类型标识符 成员名;
    .....
};
struct 结构体名 结构体变量={初始数据};
```

```
例 struct student
{
    long num;
    char name[20];
    char sex;
    int age;
    char addr[30];
};
struct student stu1={2016015, "Wang Lin", 'M', 19, "200 Beijing Road"};
```



结构体与复杂信息处理

结构体变量的初始化和赋值

形式二 struct 结构体名
 {
 类型标识符 成员名;
 类型标识符 成员名;

 } 结构体变量={初始数据};

```
例     struct student
      {     long num;
          char name[20];
          char sex;
          int age;
          char addr[30];
      } stu1={2016015, "Wang Lin", 'M', 19, "200 Beijing Road"};
```



结构体与复杂信息处理

结构体变量的初始化和赋值

形式三

```
struct
{
    类型标识符  成员名;
    类型标识符  成员名;
    .....
} 结构体变量={初始数据};
```

```
例 struct
{
    long num;
    char name[20];
    char sex;
    int age;
    char addr[30];
} stu1={2016015, "Wang Lin", 'M', 19, "200 Beijing Road"};
```



结构体与复杂信息处理

结构体变量的初始化和赋值

```
main()
{
    struct
    {   char name[15];
        char class[12];
        long num;
    } stu;
    scanf("%s",stu.name);
    scanf("%s",stu.class);
    scanf("%ld",&stu.num);
    printf("%s,%s,%ld\n",stu.name,stu.class,stu.num);
}
```

亦可用以下赋值语句：

```
strcpy(stu.name,"wenli");
strcpy(stu.class,"Computer");
stu.num=2016015;
```




结构体与复杂信息处理

结构体变量的初始化和赋值

若有以下定义，则正确的赋值语句为_____。

```
struct complex {  
    float real;  
    float image;  
};  
struct value {  
    int no;  
    struct complex com;  
}val1;
```

- A) com.real=1; B) val1.complex.real=1;
C)  val1.com.real=1; D) val1.real=1;



结构体与复杂信息处理

结构体数组的定义

三种形式

```
struct student
{
    long num;
    char name[20];
    char sex;
    int age;
};
struct student stu[2];
```

```
struct student
{
    long num;
    char name[20];
    char sex;
    int age;
}stu[2];
```

```
struct
{
    long num;
    char name[20];
    char sex;
    int age;
}stu[2];
```



结构体与复杂信息处理

结构体数组的初始化

分行初始化:

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

struct student stu[ ]={{100,"Wang Lin",'M',20},
                        {101,"Li Gang",'M',19},
                        {110,"Liu Yan",'F',19}};
```

全部初始化时长度可省



结构体与复杂信息处理

结构体数组的初始化

直接定义变量并初始化

```
struct student
{
    long num;
    char name[20];
    char sex;
    int age;
}stu[ ]={{.....},{.....},{.....}};
```



结构体与复杂信息处理

结构体数组的初始化

引用方式 结构体数组名[下标].成员名

```
struct student
```

```
{ long num;
```

```
  char name[20];
```

```
  char sex;
```

```
  int age;
```

```
}stu[3];
```

strcpy(stu[0].name,"ZhaoDa");

stu[1].age++;



结构体与复杂信息处理

例 3.35 用结构体编程实现统计候选人选票

```
struct person
{ char name[20];
  int count;
}leader[3]={ "li",0,"zhang",0,"wang",0};
#include<stdio.h>
int main()
{
    int i,j; char leader_name[20];
    for(i=1;i<=10;i++)
    {
        printf("\n输入候选人:");
        scanf("%s",leader_name);
        for(j=0;j<3;j++)
            if(strcmp(leader_name,leader[j].name)==0)
                leader[j].count++;
    }
    for(i=0;i<3;i++)
        printf("\n%5s:%d\n",leader[i].name,leader[i].count);
    return 0;
}
```

name	count
Li	0
Zhang	0
Wang	0



结构体与复杂信息处理

例 3.36 用结构体编程实现排序

```
#include<stdio.h>
int main()
{
    struct person{
        long num;char name[20];float salary;
    };
    int i,j,k;
    struct person ps[5],temp;
    for(i=0;i<5;i++){
        printf("第%d位员工的编号:",i+1);
        scanf("%ld",&ps[i].num);
        printf("第%d位员工的姓名:",i+1);
        scanf("%s",ps[i].name);
        printf("第%d位员工的工资:",i+1);
        scanf("%f",&ps[i].salary);
    }
    for(i=0;i<5;i++){
        for(j=0;j<4-i;j++){
            if(ps[j].salary>ps[j+1].salary)
                {temp=ps[j];ps[j]=ps[j+1];ps[j+1]=temp;}
        }
    }
    printf("    no.    name.    salary.\n");
    for(i=0;i<5;i++)
        printf("%8ld%14s%15.2f\n",ps[i].num,ps[i].name,ps[i].salary);
    return 0;
}
```

定义结构体

用结构体名定义
结构体数组

用循环对结构体数
组完成数据输入

根据工资进行排序



结构化与计算思维实践

简易教务系统中的成绩录入功能实现

问题分解与抽象

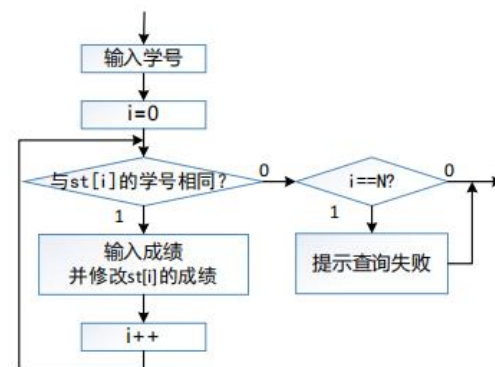
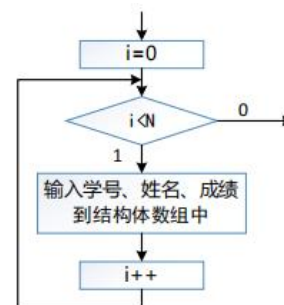
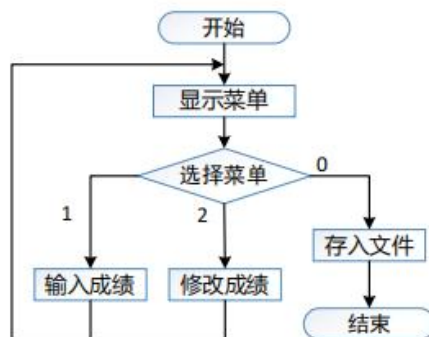
用户需求:

- 键盘输入成绩
- 连续运行
- 存盘功能
- 修改功能

数据和流程:

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

细化流程:





小结

- 根据问题的需求，定义合适的变量、数组或结构体；
- 分析问题并设计处理问题的算法逻辑，绘制流程图或书写伪代码；
- 使用顺序、分支、循环结构编程实现算法逻辑；
- 测试并完善后得到能解决问题的程序。