高级语言程序设计

张伯雷
bolei.zhang@njupt.edu.cn
bolei-zhang.github.io
计算机学院，软件教学中心

高级语言程序设计

第04章 程序流程控制

C语言程序设计

1. 初识计算机、程序与C语言

2. 初识C源程序及其数据类型

- C语言源程序的组成结构
  - C程序的6种符号：关键字、标识符、运算符、分隔符、其他符号、数据
- 基本数据类型常量
- 基本数据类型变量

3. 运算符与表达式

- 运算符、表达式的基本概念（表达式的值）
- 优先级和结合性
- 常用运算符（逻辑短路、自增自减）

习题

- 1. 设a,b,c,d均为0，执行(m=a!=b)&&(n=c==d)后，m, n的值为多少？

- 2. 若有int x=1, y=1, 表达式(!x || y--)的值是多少？

内容提要

- **语句与程序流程**
- **顺序结构**
- **选择结构**
- 循环结构
- break与continue
- 应用举例

语句与程序流程

- 语句
  - 组成程序的基本元素
  - 以“;”作为结尾
- 例

```
i++;
c = a;
```

语句的分类

- (1)表达式语句：表达式 + “;”

```
x = a>b?a:b;
c = a = b;
```

- (2)函数调用语句：函数名(参数表) + “;”

```
scanf( "%d", &a );
printf( "%c", ch );
```

语句的分类

- (3)控制语句：控制各语句执行的顺序及次数
  - 条件判断语句（if、switch）
  - 循环控制语句（while、do~while、for）
  - 中转语句（break、continue、return）

语句的分类

- (4)复合语句：以一对大括号括起的0条或多条语句
  - 在逻辑上相当于一条语句

```
{
        temp = x;
        x = y;
        y = temp;
}
```

  - 作用：在程序的某些地方，语法上只允许出现一条语句，而程序员可能需要多条语句来完成程序功能，这时就可用复合语句。

语句的分类

- (5)空语句：一个分号构成的语句
  - 表示什么事情也不需要做

    ;
  - 作用：在程序某些地方，语法上要求必须有语句出现，而程序员可能没有代码要写，或者留待以后扩充，就可以写一条空语句。

程序流程及其表示

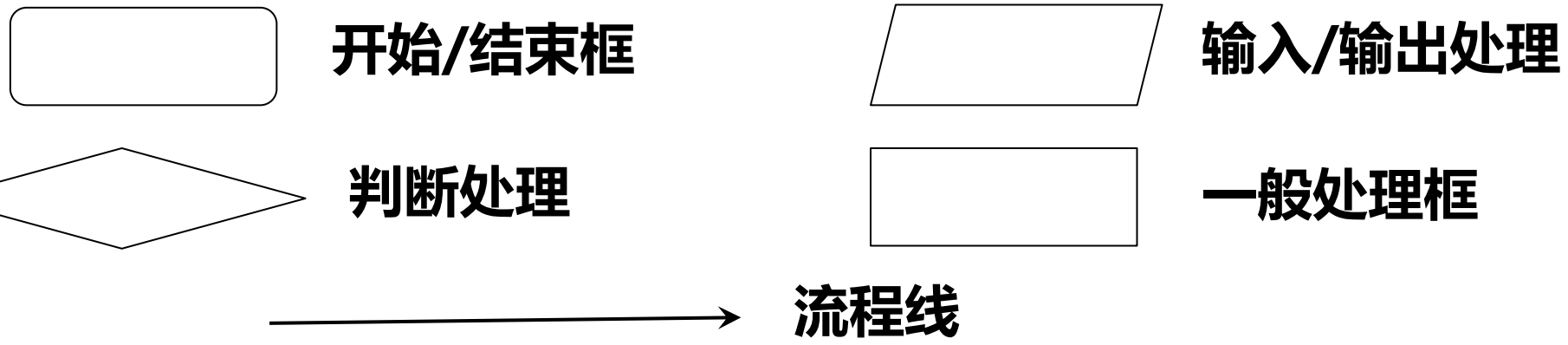
- 程序流程：代码中各语句的执行次序。
- C语言有三种基本结构：
  - 顺序结构：顺序执行每一条语句
  - 选择结构：有选择的执行部分语句
  - 循环结构：重复执行某些语句

程序流程及其表示

- 算法：为解决某个问题而采取的有限操作步骤
- 描述程序流程或者算法的方法
  - 自然语言描述
  - 传统流程图
  - NS流程图
  - 伪代码

程序流程及其表示

- 流程图由一系列图标符号组成，不同图标代表了不同的操作或流程方向。

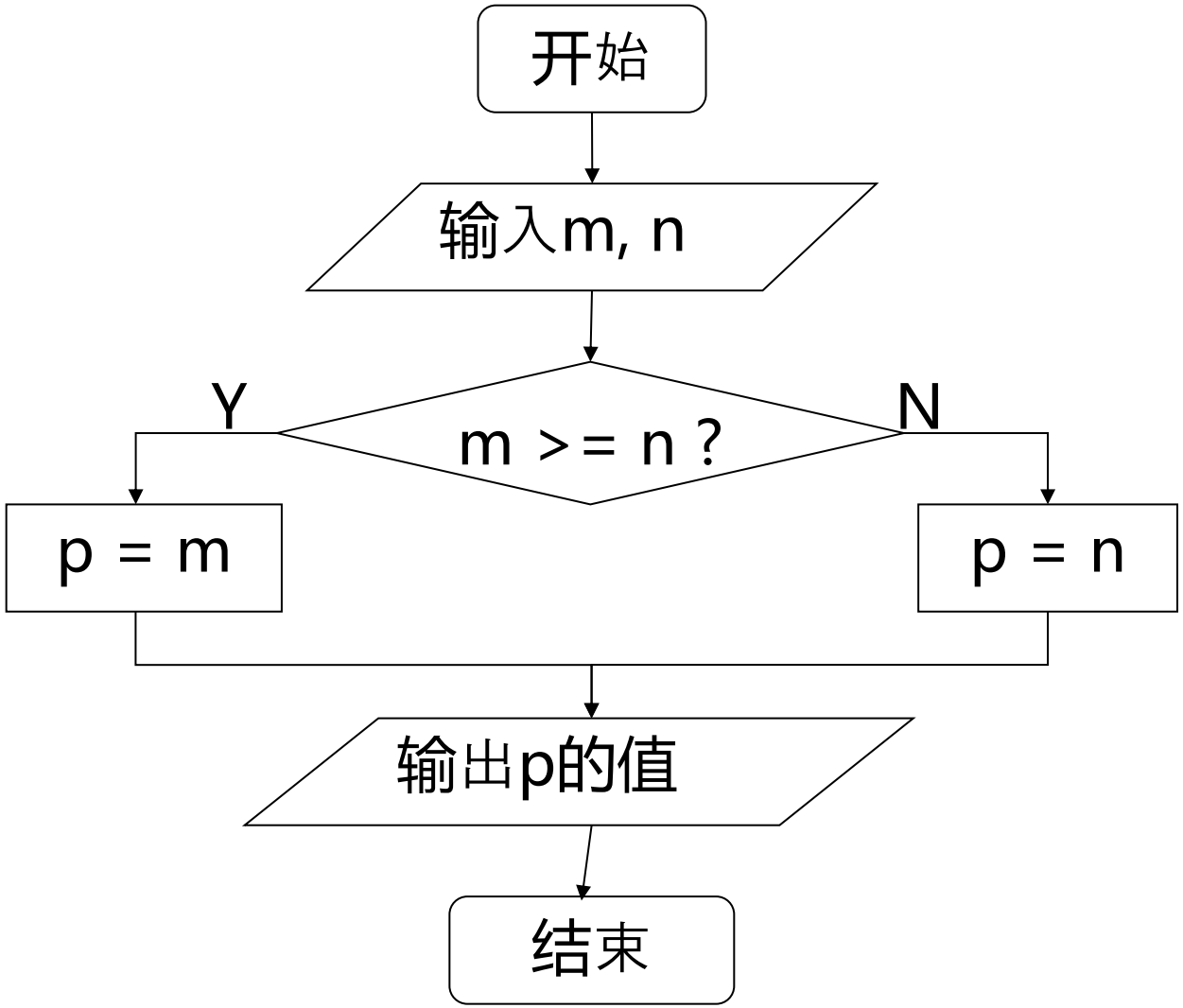


- 程序的执行总是从开始框开始，顺着流程线的方向，经过一系列处理，如输入输出（平行四边形框）、选择（菱形框）、赋值（矩形框）等，最终到达结束框。

程序流程及其表示

- 流程图示例：

  从键盘读入两个数m、n，将其中的较大数赋值给p，并输出。



内容提要

- 语句与程序流程
- 顺序结构
- 选择结构
- 循环结构
- break与continue
- 应用举例

顺序结构

- 基本思想：从前往后依次执行每一条语句，每一条语句只执行一次。
- 课本例4.1：求三角形面积。
  - 从键盘上输入三角形三条边的边长，求该三角形的面积并输出至屏幕。
  - 设三条边分别为a、b、c，可以根据如下数学公式来计算面积，其中$p = (a + b + c) / 2$。

$$\sqrt{p(p-a)(p-b)(p-c)}$$

顺序结构

```
#include <stdio.h>
#include <math.h>                          /* 包含了函数sqrt原型 */
int main( )
{
        double edge1, edge2, edge3, p, area;
        printf ( "Enter three edges of a triangle: " );
        /* 提示用户输入 */
        scanf ( "%lf%lf%lf", &edge1, &edge2, &edge3 );
        /* 从键盘读入三条边长*/
        p = ( edge1 + edge2 + edge3 ) / 2;
        area = sqrt( p * ( p - edge1 ) * ( p - edge2 ) * ( p - edge3 ) );
        /* 使用数学公式求面积*/
        printf ( "area = %lf\n", area );
        /* 输出面积 */
        return 0;
}
```

- 说明
  - 根据公式求解三角型面积时，需要使用求平方根函数sqrt，其原型在math.h头文件中，因此需 “#include <math.h>”。
  - 运行该程序时，将首先从main函数的第一行开始，依次运行每一行的语句，每条语句也仅运行一遍，直至遇到return语句为止。
- 思考题
  - 如果用户输入的三条边不能构成一个三角形，程序的运行结果如何？

内容提要

- 语句与程序流程
- 顺序结构
- 选择结构
- 循环结构
- break与continue
- 应用举例

选择结构

- 选择结构：程序中部分代码的执行受“预设条件”的控制。当“预设条件”符合时才执行这些语句。
- 相关的控制语句
  - if
  - if~else
  - switch

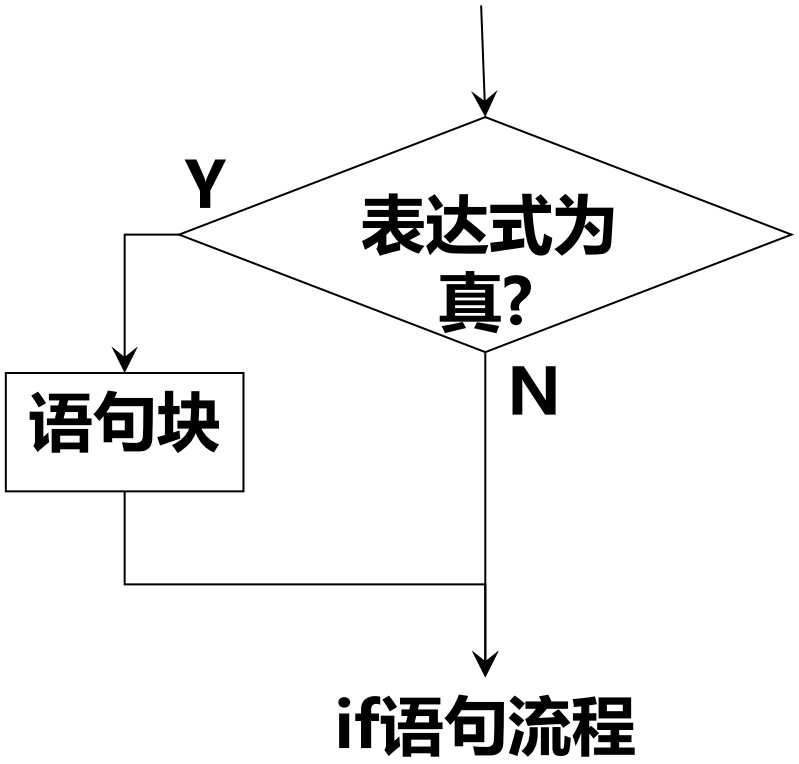
if语句

- 语法

  if (表达式)

  语句块

- 含义
  - 如果表达式为真，则执行语句块，否则不执行。

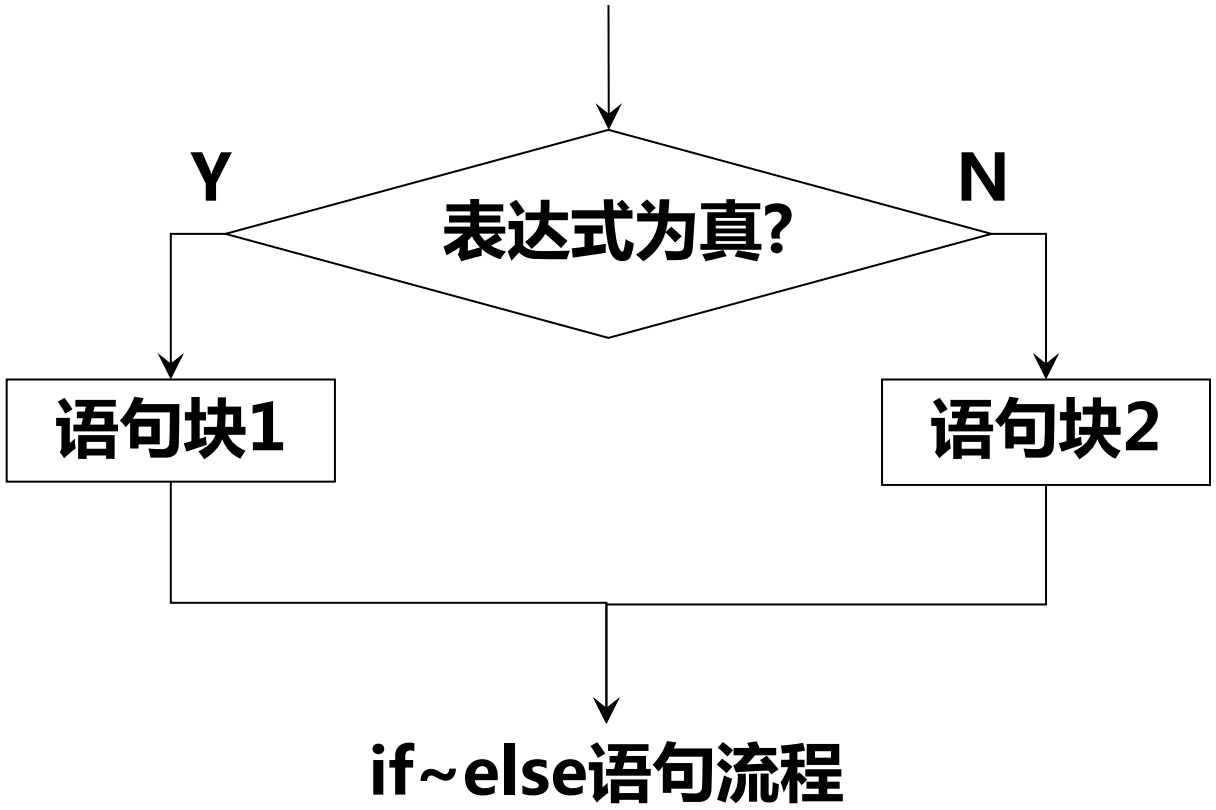


if~else语句

- 语法

```
if (表达式)
        语句块1
else
        语句块2
```

- 含义
  - 如果表达式为真，则执行语句块1，否则执行语句块2。



if~else语句流程

if和if~else的说明

- 表达式可以任何合法的C语言表达式，如加减、赋值、逻辑运算、关系运算等。
- 语句块可以是一条语句，也可以是多条语句组成的复合语句，此时需使用大括号。
- 当语句块是单条语句时，虽然语法上可以不使用大括号，但是为了代码的清晰性和层次性起见，推荐使用大括号。

条件语句的使用

- 例4.2：年龄比较。
  - 从键盘读入两个人的年龄，比较并输出年长者的年龄。
- 基本思想：先从键盘读入两个年龄age1和age2，使用if语句比较它们的大小后，将较大的那个数输出即可。

例4.2

```
#include <stdio.h>
int main( )
{
        int age1, age2;
        printf( "Enter age of two persons : ");
        scanf( "%d%d", &age1, &age2);
        if ( age1 >= age2 )                    /* age1较大 */
        {
                printf( "the older age is %d\n", age1 );
        }
        else                                   /* age2较大 */
        {
                printf( "the older age is %d\n", age2 );
        }
        return 0;
}
```

条件语句的使用

- 例4.3：求三角形面积的改进。
  - 对例4.1求三角形面积的代码进行改进，使得程序能够对不合理输入进行一定的判别。
- 基本思想：当读取三条边以后，首先进行判别（三者均为正数，且任意两边之和大于第三边），然后再根据公式进行求解。

例4.3主要代码

```
double edge1, edge2, edge3, p, area;
……                                                    /* 从键盘读入三条边 */
if ( edge1 > 0 && edge2 > 0 && edge3 > 0 && edge1 + edge2 >      edge3 &&
   edge1 + edge3 > edge2 && edge2 + edge3 > edge1 )
          /* 用户输入判断 */
{
          p = ( edge1 + edge2 + edge3 ) / 2;
          area = sqrt( p * ( p-edge1 ) * ( p-edge2 ) * ( p-edge3 ) );
          printf(   “area = %f\n” , area );
}
else                                                  /* 不合法时需提示用户 */
{
          printf(   “Error input!\n”   );
}
```

条件语句的使用

- 例4.4：直角三角形判别。
  - 用户从键盘输入三个数，判断这三个数能否构成一个三角形，如果可以，进一步判断它们能否构成一个直角三角形。
- 基本思想：程序在读入三个数以后，借鉴上一例，首先判断它们是否能构成一个三角形。如果可以，继续使用if语句，比较它们是否满足勾股定理，以判断它们是否能构成一个直角三角形。

例4.4

- 例4.4代码见课本。
- 说明
  - 本例是一个典型的if嵌套的例子，最多时有三层if语句。
  - 从本例中也可以看出，无论是if子句还是else子句，都可以进行if嵌套。

例4.4

- 说明
  - 本题中，判断直角三角形需使用勾股定理，即两条边的平方和等于第三条边的平方。但在编程实现时，代码没有使用“==”进行计算。
  - 这是因为edge1、edge2、edge3均为double型数据，而double型数据是不精确的，一般不使用“==”来进行相等的判断。如果要判断两个实型数据是否相等，合理的做法是，计算它们的差，如果该差值的绝对值小于一个较小的数（本例中是1E-2），就可近似认为它们相等。

关于if嵌套的进一步说明

- if及if~else语句最多可实现两种情形的判别，而使用if嵌套可判别两种以上的情形。
- 当需要对多种情况进行判别处理时，为结构清晰起见，可使用如下的排版方式：

```
if   (表达式1)
     语句块1
else  if  (表达式2)
     语句块2
else  if  (表达式3)
     ……
else
     语句块n
```

- 在if嵌套中会出现多个if和else。如果没有使用大括号明确，则存在else究竟与哪个if配对的问题。在这种情况下，C语言规定：else总是与它前面最近的、且没有配对的if相匹配。
- 为避免出现误解，推荐每个语句块都使用大括号，哪怕只有一条语句。

switch语句

- switch语句与if嵌套语句的功能类似，主要用于多种情况的判断处理。
- 语法：

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

switch语句的说明

- switch后面的表达式可以为整型、字符型或者枚举型，但不允许是实型。
- case后面必须为常量，且类型应与switch中表达式的类型相同。
- switch语句的执行过程是：首先计算switch后面表达式的值，然后与各case分支的常量进行匹配，与哪个常量相等，就从该分支的语句序列开始执行，直至遇到break或者switch语句块的右大括号。

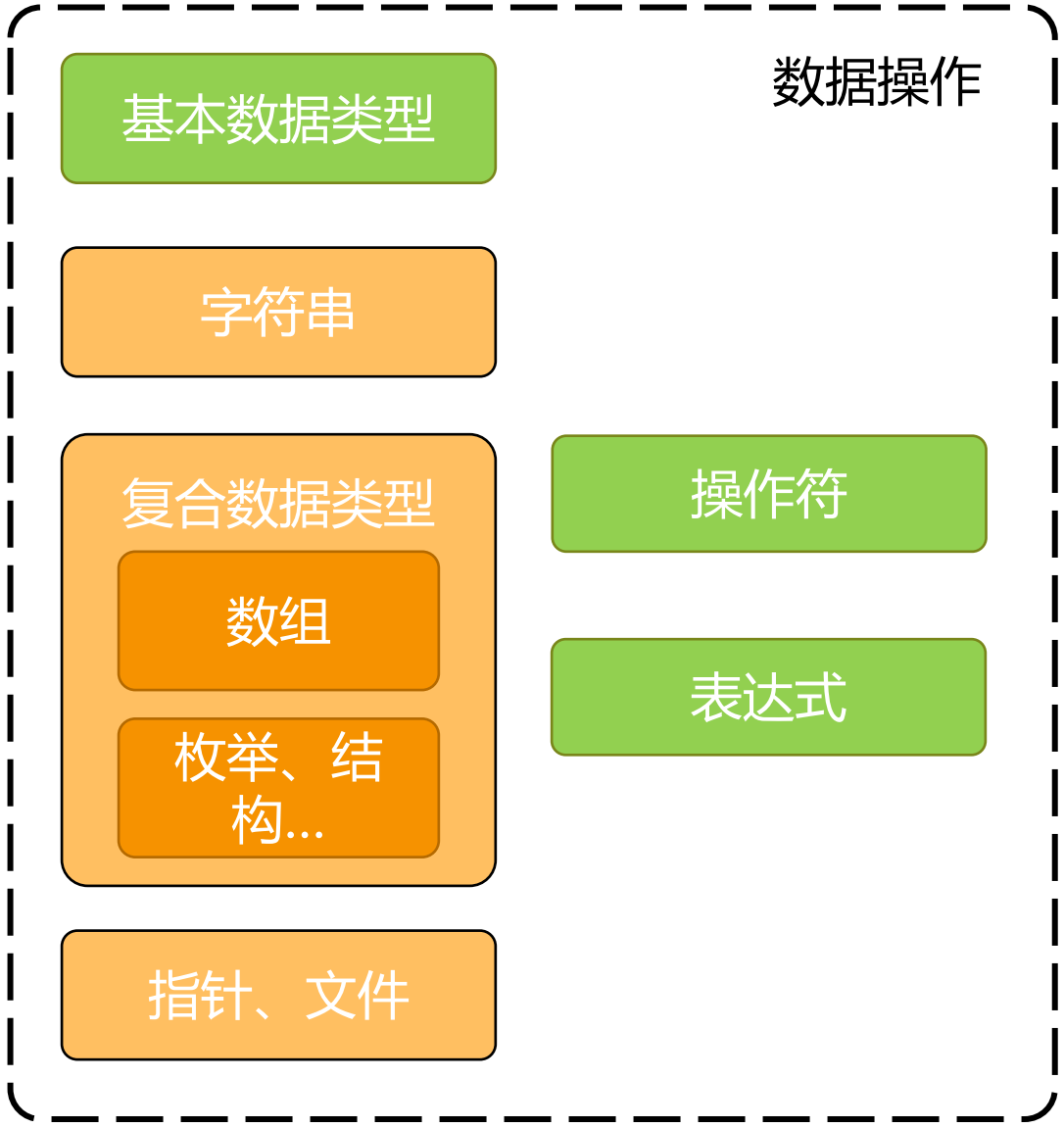
switch语句的说明

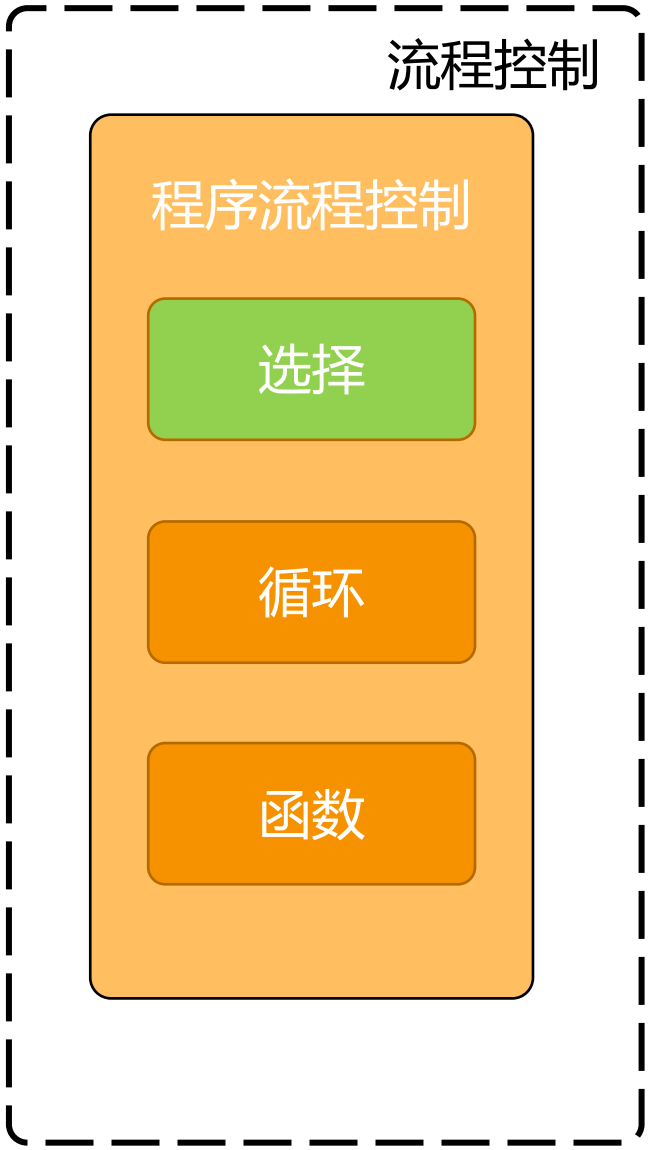
- default分支主要用于处理switch表达式与所有case常量都不匹配的情况。它在语法上可以省略，但推荐使用。

switch语句的使用

- 例4.5 月份天数计算。
  - 从键盘输入年份和月份，计算该月份的天数并输出。
- 分析：从键盘读入年份year和月份month后，根据month的值，使用switch语句，可以计算出这个月份的天数。其中，1月、3-12月的天数确定的，2月份则要根据year判断是否是闰年。
- 代码见课本。

C语言


数据操作
基本数据类型
字符串
复合数据类型
数组
枚举、结构...
指针、文件
操作符
表达式


流程控制
程序流程控制
选择
循环
函数

多文件工程

高级语言程序设计

张伯雷
bolei.zhang@njupt.edu.cn
bolei-zhang.github.io
计算机学院，软件教学中心

高级语言程序设计

第04章 程序流程控制

1. **初识计算机、程序与C语言**
2. **初识C源程序及其数据类型**
3. **运算符与表达式**
   - 运算符、表达式的基本概念（表达式的值）
   - 优先级和结合性
   - 常用运算符（逻辑短路、自增自减）
4. **程序流程控制**
   - 语句与程序流程
   - 顺序结构
   - 选择结构

习题

- 1.

```c
int x = 1, y = 1, z = 1;
switch(x){
    case 1: switch(y){
        case 1: printf("!!"); break;
        case 2: printf("##"); break;
        case 3: printf("@@"); break;
    }
    case 0: switch(z){
        case 0: printf("$$");
        case 1: printf("^^");
        case 2: printf("&&");
    }
    default: printf("**");
}
```

- 2.

```c
int a = 2, b = 3;
if (a == 2){
    printf("%d\n", a);
} else if (b == 3){
    printf("%d\n", b);
}

if (a == 2){
    printf("%d\n", a);
}
if (b == 3){
    printf("%d\n", b);
}
```

内容提要

- 语句与程序流程
- 顺序结构
- 选择结构
- 循环结构
- break与continue
- 应用举例

循环结构

- 循环结构：程序中的某些语句和代码，在“预设条件”的控制下可以执行多次。
- 相关的控制语句
  - while
  - do~while
  - for

while语句

- 语法

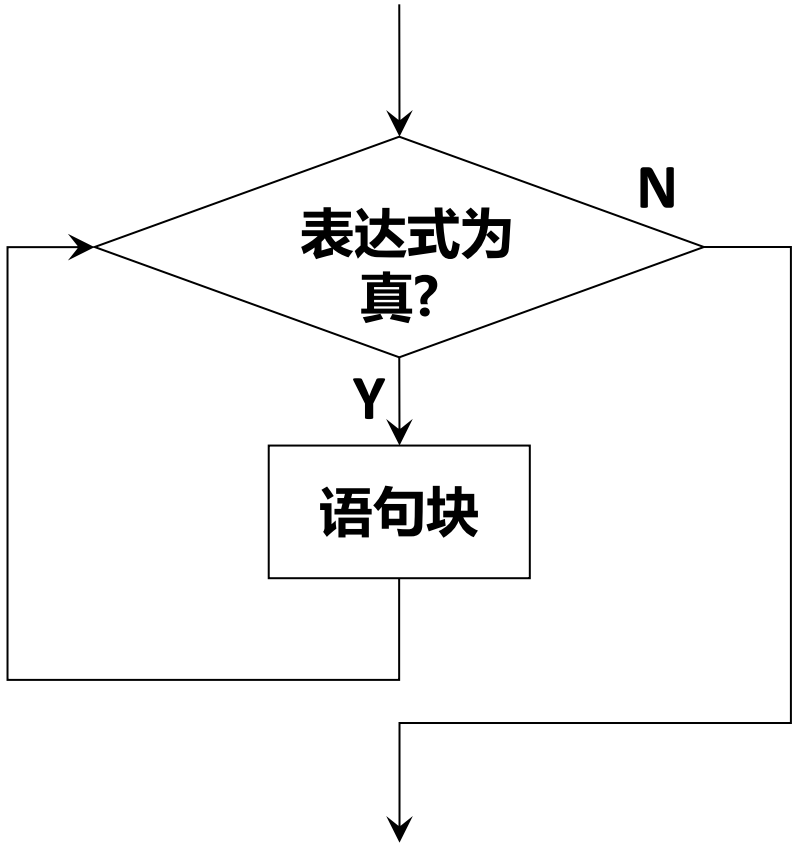
```
while（表达式）
{
  语句块
}
```

- 表达式可以是任何合法的C语言表达式，其计算结果用于判断语句块是否该被执行。
- 语句块是需要重复执行语句的集合，它也被称为循环体。

while语句的执行流程

(1)计算表达式的值。若为真，则转步骤(2)；否则退出循环，执行while的后续语句。

(2)执行语句块，并返回步骤(1)。



while语句的使用

- 例4.6 求累加和。
  - 从键盘读入int型正整数n，计算 $\sum_{i=1}^{n} i$ 的值并输出

- 分析： $\sum_{i=1}^{n} i = 1 + 2 + \cdots n$

  加法操作重复执行。因此可设两个变量sum和i，sum初值为0，i初值为1。然后把i加到sum上，重复n次，每次i的值加1。这样就可以实现从1到n的累加。

例4.6主要代码

```
int i = 1;
int sum = 0;
while ( i <= n )
{
        sum += i;
        i++;
}
```

while语句说明

- 本例中变量i承载了控制循环次数的作用，对于这类变量，我们称之为循环控制变量。
- 通常情况下，循环体内都会有语句对循环控制变量进行修改，以控制循环结束的时机。
- 累加和后面要学的阶乘是一重循环最典型的应用，初学者应好好体会。

do~while语句

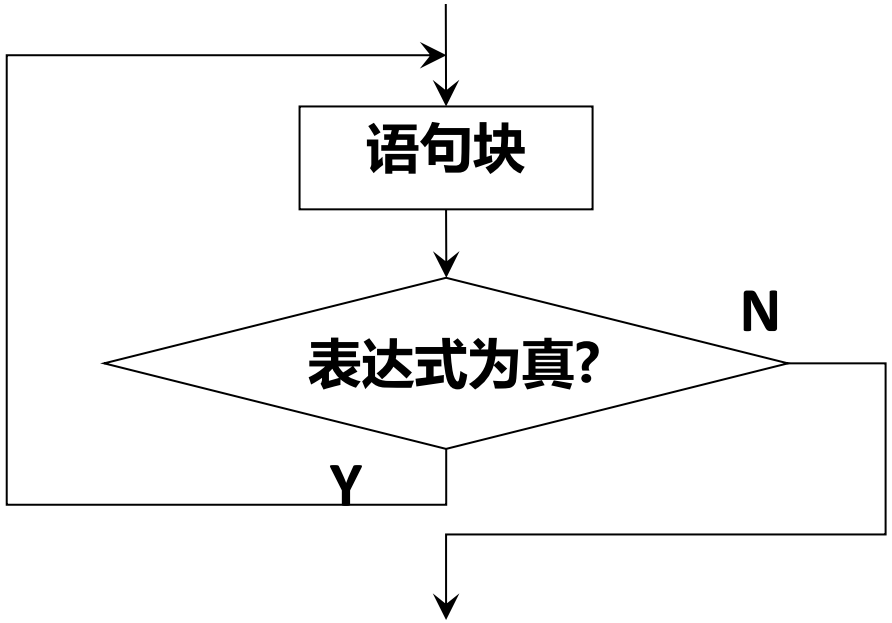
- do~while语句又称直到型循环语句
- 语法

```
do
{
   语句块
} while ( 表达式 );
```

do~while语句执行流程

(1)执行语句块，即循环体。

(2)计算表达式的值。若为真，则转(1)；
否则退出循环，执行下一条语句。



do~while的使用

- 例4.7 求阶乘。
  - 从键盘读入int型正整数n，计算n！并输出。

- 分析：n！= 1 * 2 * 3 * … * n，

  乘法操作重复执行。与上一题类似，可设两个变量fac和i，fac初值为1，i初值为1。然后将i乘以fac，重复n次，每次i的值加1。这样就可以实现从1到n的累乘。

例4.7主要代码

```
int n, i;
double fac;
……                                          /* 输入n的值 */
i = 1;
fac = 1;
do
{
        fac *= i;
        i++;
} while( i <= n );
```

do~while语句说明

- 阶乘变量fac定义成double类型，这时因为阶乘运算很容易超出int的范围。选用double类型可支持大一些的数。即便如此，VC环境下170以上的阶乘也超出了double的范围。

for语句

- for语句是C语言中最常用、功能也最强大的循环控制语句
- 语法

```
for ( 表达式1 ; 表达式2 ; 表达式3 )
{
    语句块
}
```

- for语句有三个表达式，表达式2是控制循环的条件。

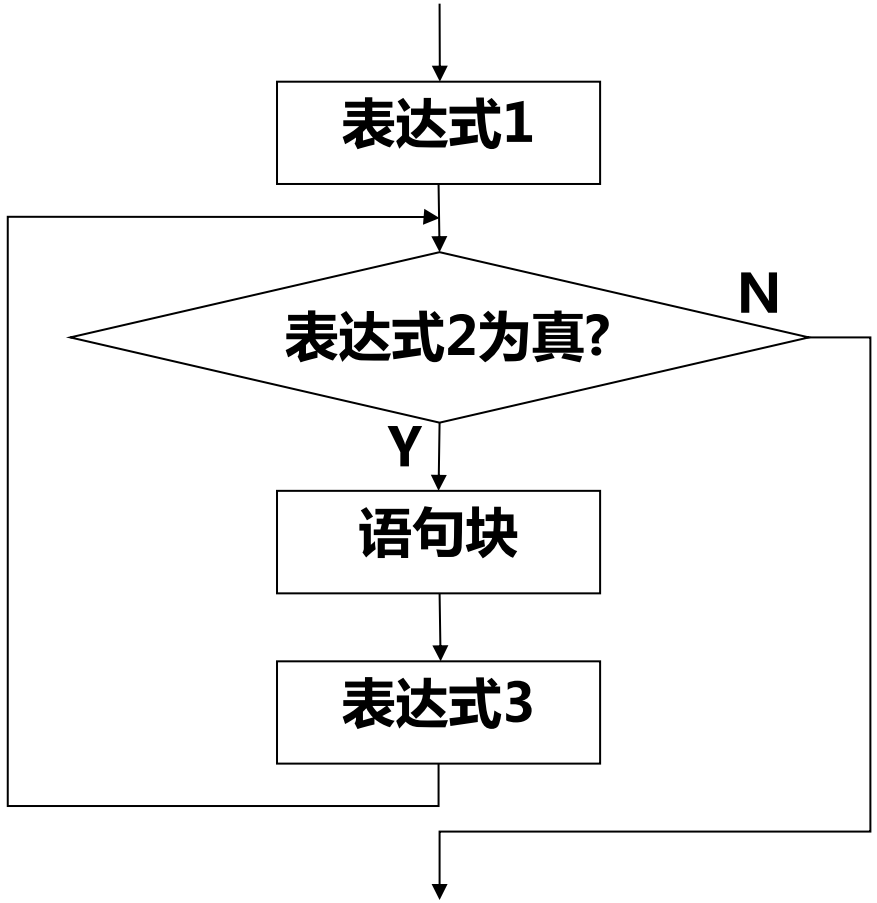
for语句执行流程

(1)计算表达式1。

(2)计算表达式2。若为真，则转(3)；否则退出循环。

(3)执行语句块，即循环体。

(4)计算表达式3，转(2)。



for语句说明

- 表达式1、表达式2、表达式3均可以省略。
- 当表达式2省略时，默认其计算结果为真。

for语句的使用

- 例4.8 数列求和。
  - 已知一个数列如下，求该数列前1000项的和。

$$s=\left\{1,-\frac{1}{3},\frac{1}{5},\ldots,\frac{(-1)^{i+1}}{2i-1}\ldots\right\}$$

- i = 1, 2, ...

- 分析：欲求该数列的和，可设两个变量sum和item，sum初值为0，item初值为数列的第一项1。然后把item加到sum上，重复1000次，每次循环时，对item的值进行修改。这样就可以实现数列元素的累加。

例4.8主要代码

```
int i, sign;
double item, sum;
sum = 0;                                    /* 初值置为0 */
sign = 1;
for ( i = 1 ; i <= 1000 ; i++ )
{
  item = sign / ( 2.0 * i - 1 );      /* 计算每一次的累加项item */
  sum += item;                        /* 将累加项item加到总和sum上 */
  sign = -sign;                       /* 计算下一个累加项的符号sign */
}
```

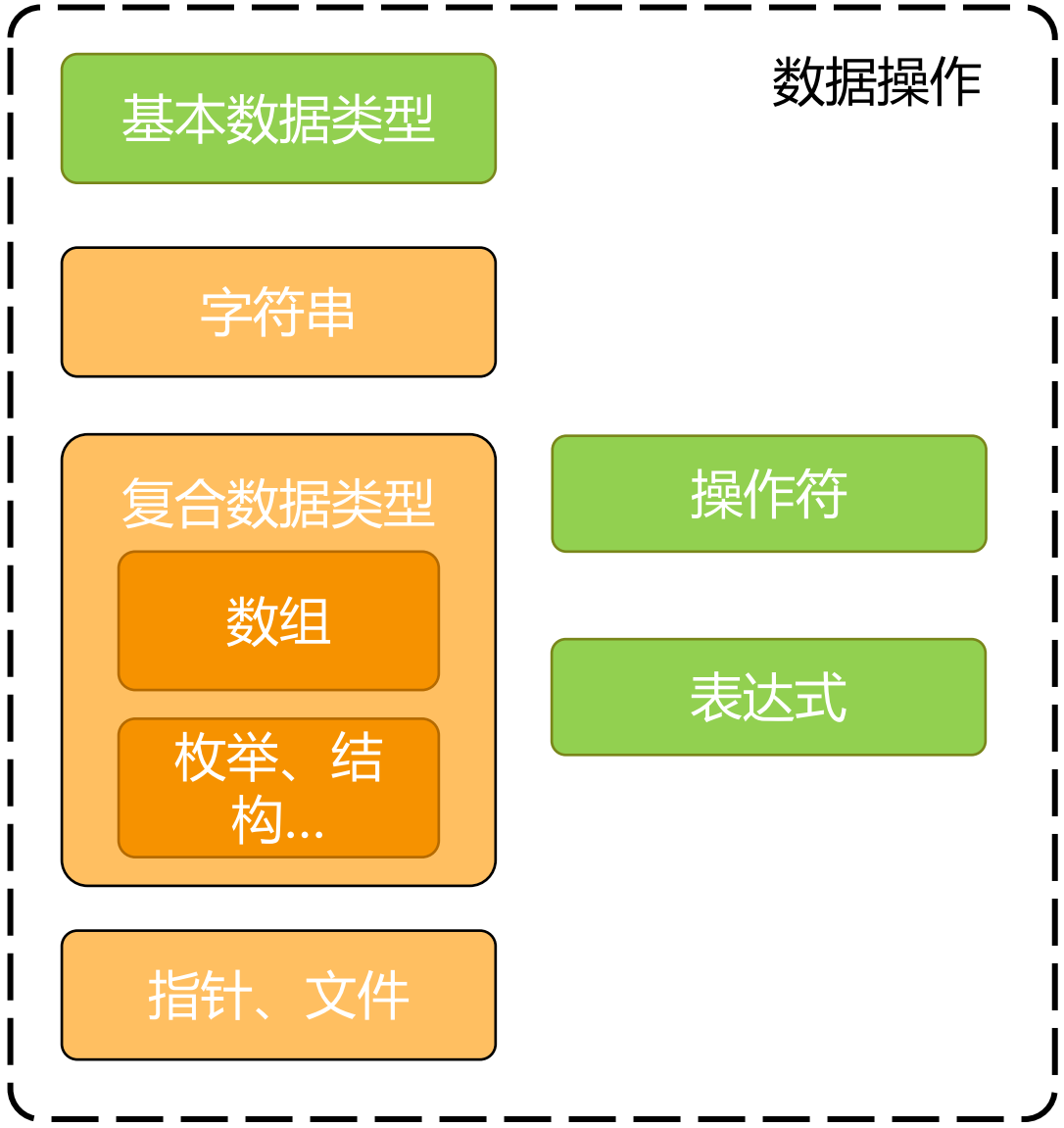
例4.8说明

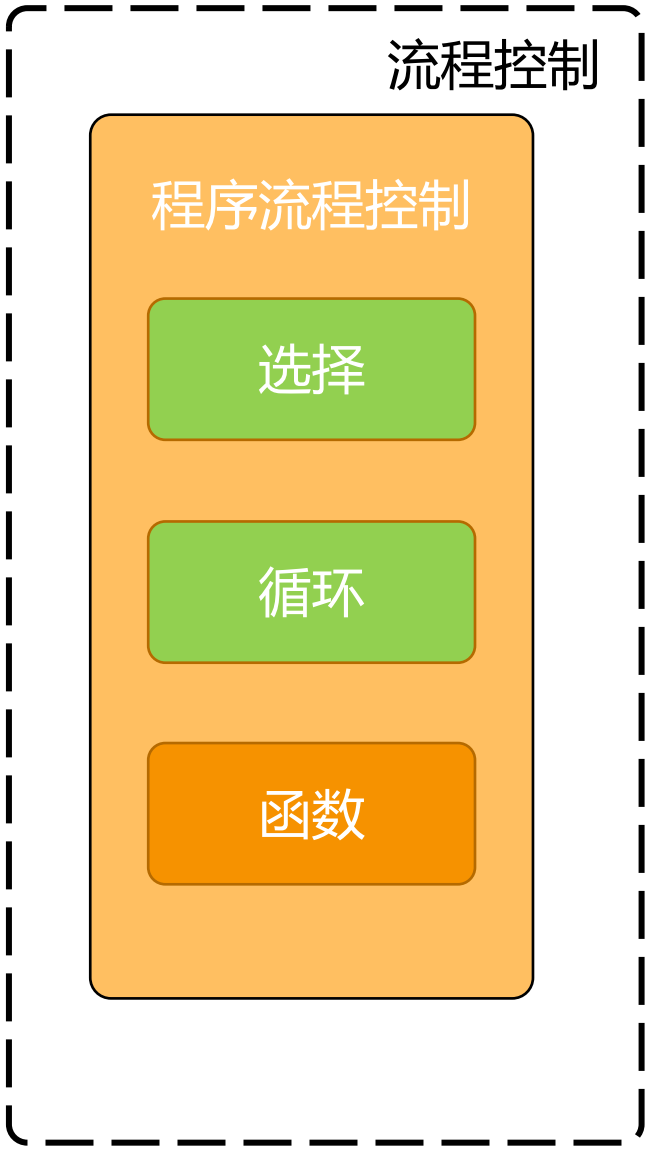
- 本在求解累加项item时，使用了一定的编程技巧。
- item也可以直接根据通项公式求解：

```
item = pow( -1, i + 1 ) / ( 2.0 * i - 1 );
```

  - 其中，pow是幂函数，使用时需包含头文件math.h。
  - 该方法比较直观，但函数调用及求幂计算的系统开销相对较大，不如例题中的高效。

C语言

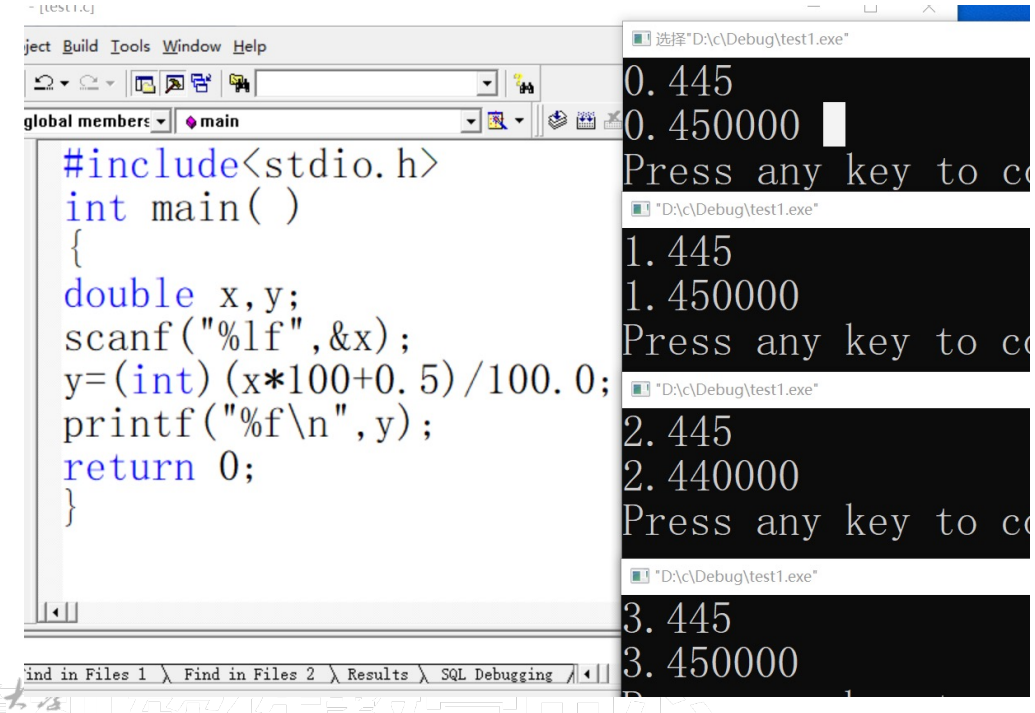

数据操作
基本数据类型
字符串
复合数据类型
数组
枚举、结构...
指针、文件
操作符
表达式


流程控制
程序流程控制
选择
循环
函数

多文件工程

两个例子

```c
#include<stdio.h>
int main( )
{
    int c = 1, d = 2;
    printf("%d\n%d\n%d\n", ++d, c+=d, d);
    printf("%d\n", d);
    return 0;
}
```




```c
#include<stdio.h>
int main( )
{
    double x,y;
    scanf("%lf", &x);
    double z = x*100 + 0.5;
    printf("%.15lf\n", z);
```

习题

- 1. 求z的值

```
int x, y=0, z=0;
for (x=1; x<=5; x++){
    y = y+x;
    z = z+y;
}
```

- 2.输出斐波那契数列的前10项

  1,1,2,3,5,8,13,21,…

- 3. 输出100以内的所有质数