中国古典中的“博弈”

田忌

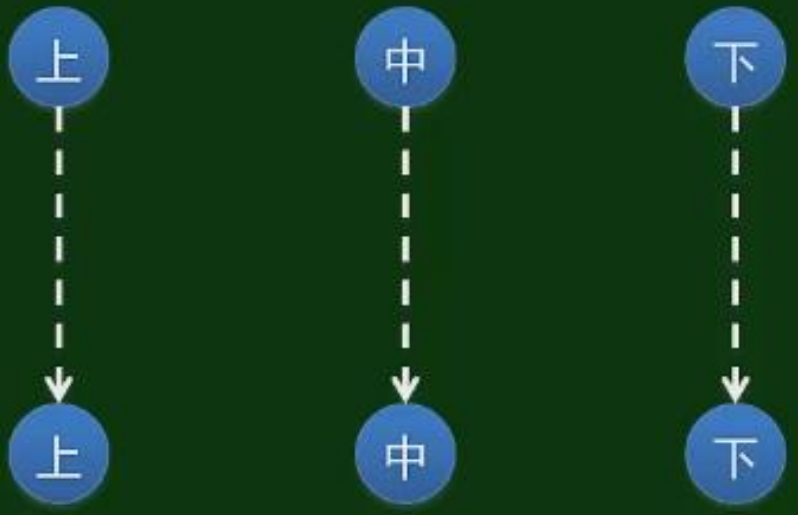
上 中 下

上 中 下

齐威王

中国古典中的“博弈”

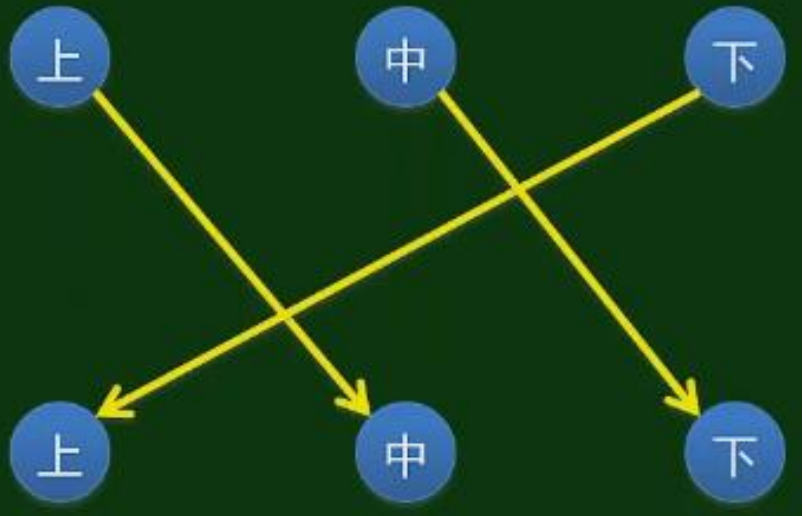
田忌



齐威王

中国古典中的“博弈”

田忌



齐威王

田忌赛马博弈中的稳定策略组合？

齐威王

田忌	上中下	上下中	中上下	中下上	下上中	下中上
上中下	-3，3	-1，1	-1，1	1，-1	-1，1	-1，1
上下中	-1，1	-3，3	1，-1	-1，1	-1，1	-1，1
中上下	-1，1	-1，1	-3，3	-1，1	-1，1	1，-1
中下上	-1，1	-1，1	-1，1	-3，3	1，-1	-1，1
下上中	1，-1	-1，1	-1，1	-1，1	-3，3	-1，1
下中上	-1，1	1，-1	-1，1	-1，1	-1，1	-3，3

（上中下，上中下）是稳定的吗？（中下上，下上中）是稳定的吗？

第2章 选择控制结构及应用

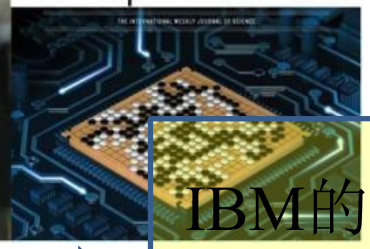
主要内容

- 选择结构的基本运行符
- if-else选择结构
- switch选择结构
- 应用实例

难点

- 多条件选择结构

第2章 选择控制



人工智能

IBM的Deep Blue *vs* 国际象棋冠军卡斯帕罗夫

谷歌(Google)的AlphaGo *vs* 围棋九段李世石

人工智能算法

是--1　　否--0

语言支撑之一—选择结构

第2章 选择控制结构及应用

主要内容

- 选择结构的基本运行符
- if-else选择结构
- switch选择结构
- 应用实例

===选择结构的基本运行符===

➢ 关系运算符及表达式

操作数个数

结合性

优先级别

序号	优先级别	关系运算符	关系运算符含义	操作数个数	结合性
		>	大于		
3	6			2	自左向右
4				2	自左向右
5	7			2	自左向右
6				2	自左向右

表达式1 *关系运算符* 表达式2

值 运算次序

例：int a=3, b=2, c=8, d=2;
float x=1.5f;
a+b>c-d; x>3/2;
'a'+b>c;
a!=(c==d);

表达式的值是由最后（优先级最低）一次运算结果决定的

1 or 0

非0即真

0为假

===选择结构的基本运行符===

➢ 逻辑运算符及表达式

表达式 逻辑运算符 表达式 或 逻辑运算符 表达式

序号					结合性
1	11				右
2	12				自左向右

单目

“短路”原则：

“&&”表达式，两端任意一端值为0，则另一端将不被执行！

“||”表达式，两端任意一端值为1，则另一端将不被执行！

思考：当 i=1(序是怎样的？i、j、k 及表达式的值各是多少呢？若将表达式改为 i==1&&j==3||k=k+1 呢？

非 0	非 0	1	1	0
非 0	0	0	1	0
0	非 0	0	1	1
0	0	0	0	1

“短路”原则大大提升了C语言的运算效率，

第2章 选择控制结构及应用

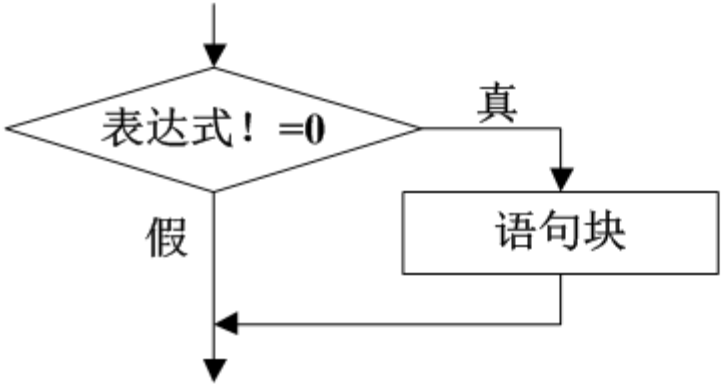
主要内容

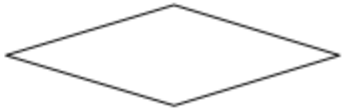
- 选择结构的基本运行符
- **if-else选择结构**
- switch选择结构
- 应用实例

===if-else选择结构===

- if结构

```
if (表达式)
{
    语句块;
}
```





条件判定框

===if-else选择结构===

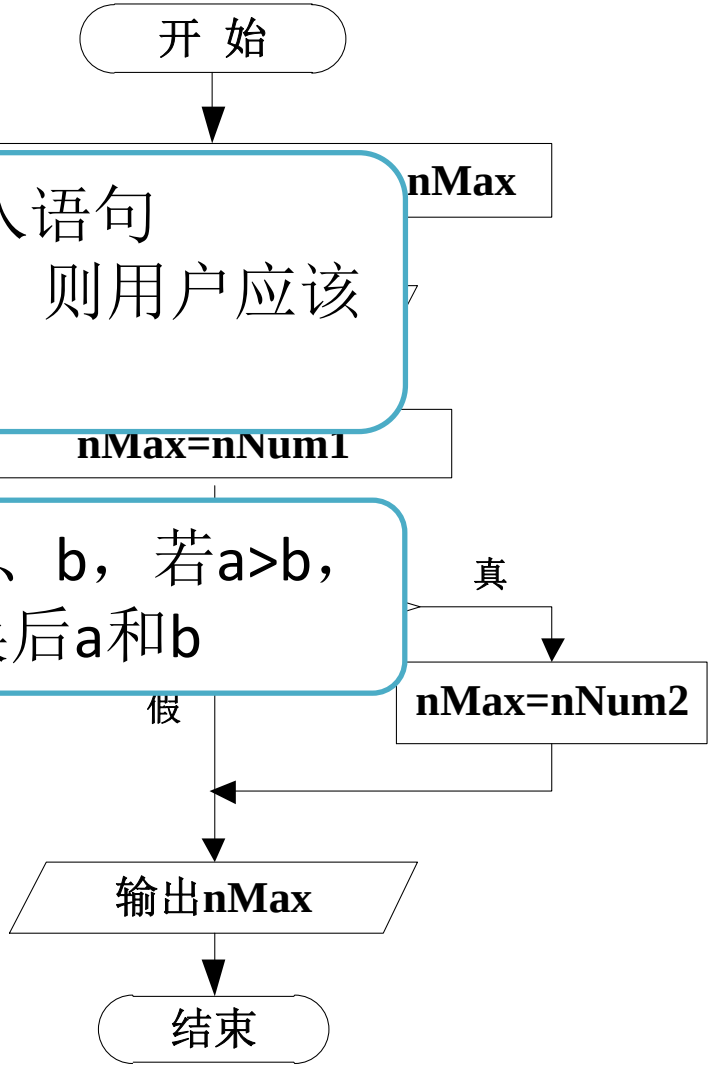
- if结构

[例2-1] 编程实现：任意输入两个整

思考：如果有这样一条输入语句 **scanf("a=%d,b=%d",&a,&b)**，则用户应该怎样输入呢？

练习：输入两个整型变量a、b，若a>b，则交换a和b的值，输出交换后a和b

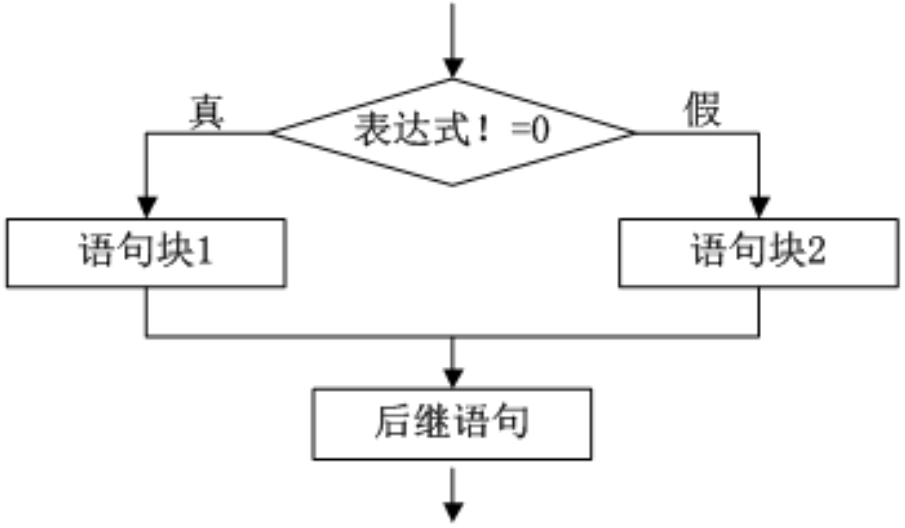
```
#include <s
#include <s
int main(vo
{
    int nN
    int nM
    printf(
    scanf("%d%d",&nNum1,&nNum2);
    nMax= nNum1;
    if (nMax < nNum2)
        nMax= nNum2;
    printf(" Max=%d\n",nMax);
    return 1;
}
```



===if-else选择结构===

➢ if-else形式

```
if (表达式)
{
    语句块1;
}
else
{
    语句块2;
}
```

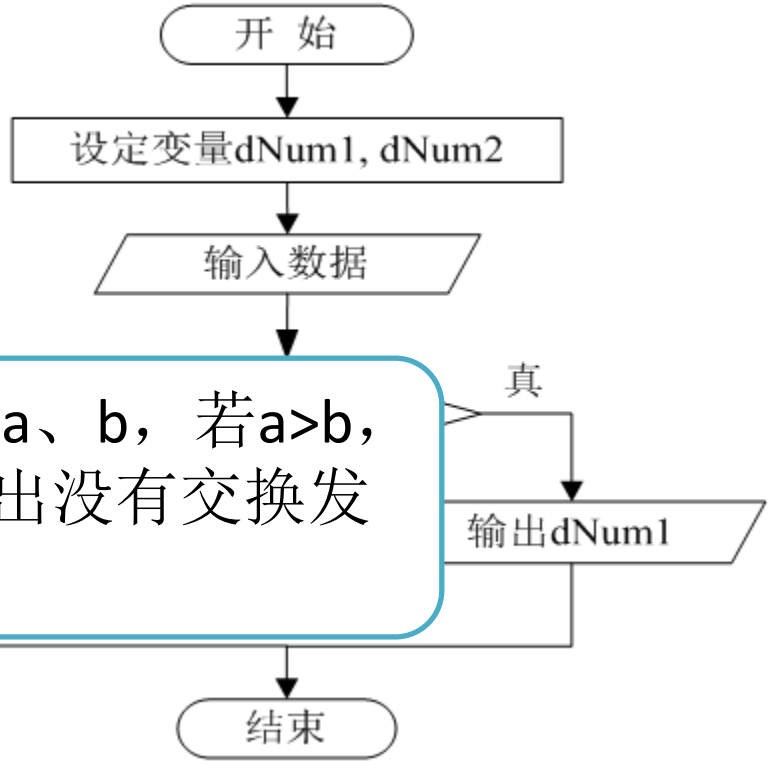


===if-else选择结构===

➢ if-else形式

[例2-2]输入两个实数，请用if-else语句输出其中的大数。

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    double dNu
    printf("\n i
    scanf("%lf
    if(dNum1>d
        printf("\n Max=%5.2lf\n",dNum1);
    else
        printf("\n Max=%5.2lf\n",dNum2);
    return 1;
}
```

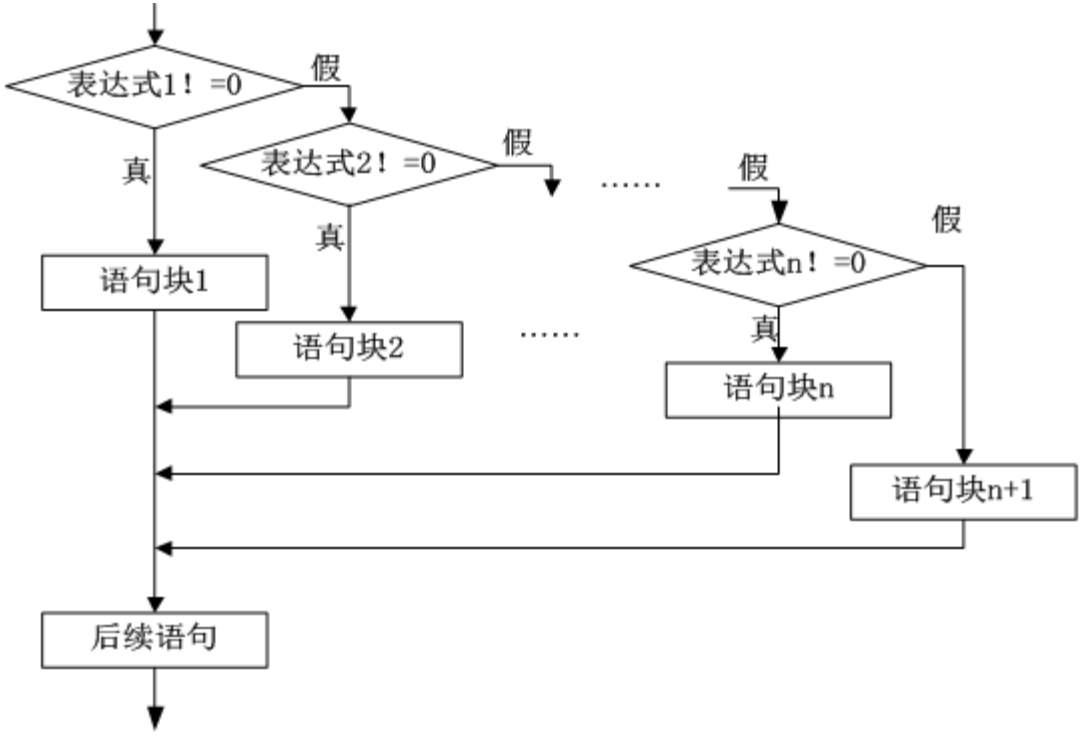


练习：输入两个整型变量a、b，若a>b，则交换a和b的值，否则输出没有交换发生，输出交换后a和b

===if-else选择结构===

➢ if-else-if 形式

```
if (表达式1)
{语句块1；}
else if (表达式2)
{语句块2；}
……
else if (表达式n)
{语句块n；}
else
{语句块n+1；}
```



===if-else选择结构

```
#include <stdio.h>
int main(void)
{
   char cLevel=0x20;                    /*赋初值，空格的ASCII值，以十六进制表示*/
    float fScore;
    printf("请输入成绩： ");
    scanf("%f",&fScore);                /*输入考试分数*/
    if(fScore>100.0||fScore<0.0)
    {
        printf("\n输入成绩错误！ ");
        return 0;                              /*成绩输入错误，则结束程序*/
    }else if (fScore > 89.99)           /*分数fScore>=90*/
        cLevel='A';                                 /*单条语句不用加“{}”*/
    else if (fScore > 79.99)            /*分数90>fScore>=80*/
        cLevel='B';
    else if (fScore > 69.99)            /*分数80>fScore>=70*/
        cLevel='C';
    else if (fScore > 59.99)            /*分数70>fScore>=60*/
        cLevel='D';
    else                                                /*分数fScore<60*/
        cLevel='E';
    printf("该成绩的等级为：%c\n",cLevel);
    return 1;
}
```

===if-else选择结构===

➢ if语句的嵌套

```
if (表达式)
{
    if (表达式)
    {
        语句;
    }
}
```

```
if (表达式)
{   if (表达式)
    { 语句; }
 }else
{   if (表达式)
    { 语句; }
}
```

[例2-4] 将“例2-3”改写为if语句的嵌套形式。判断输入成绩的合法性，再进行分档。如何设计呢？

===if-else选择结构===

➢ if语句的嵌套

思考：可以用右边的程序来实现左边的公式吗？试画出流程图来分析。

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

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

===if-else选择结构===

➢ if结构中使用表达式的拓展

C语言允许所有合法的表达

if(a+b){.....}
if(a%b) {.....}
if(a) {.....}

思考：你知道这些

```
int main()
{   int a,b=0,c=0,d=0;
    scanf("%d", &a);
    if(a=1) {
       b=1;
       c=2;
    } else
        d=3;
    printf("%d,%d,%d,%d\ x0A ",a,b,c,d);
    return 1;
}
```

```
int main()
{   int a=0,b=0;
    if(a=0)
        b=1;
    else b=3;
        printf("%d \ x0A ",a,b);
    return 1;
}
```

===if-else选择结构===

➢ 条件运算符、条件表达式及条件语句

表达式1? 表达式2：表达式3

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

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

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

```
y=x>0?1:(x<0?-1:0)
```

第2章 选择控制结构及应用

主要内容

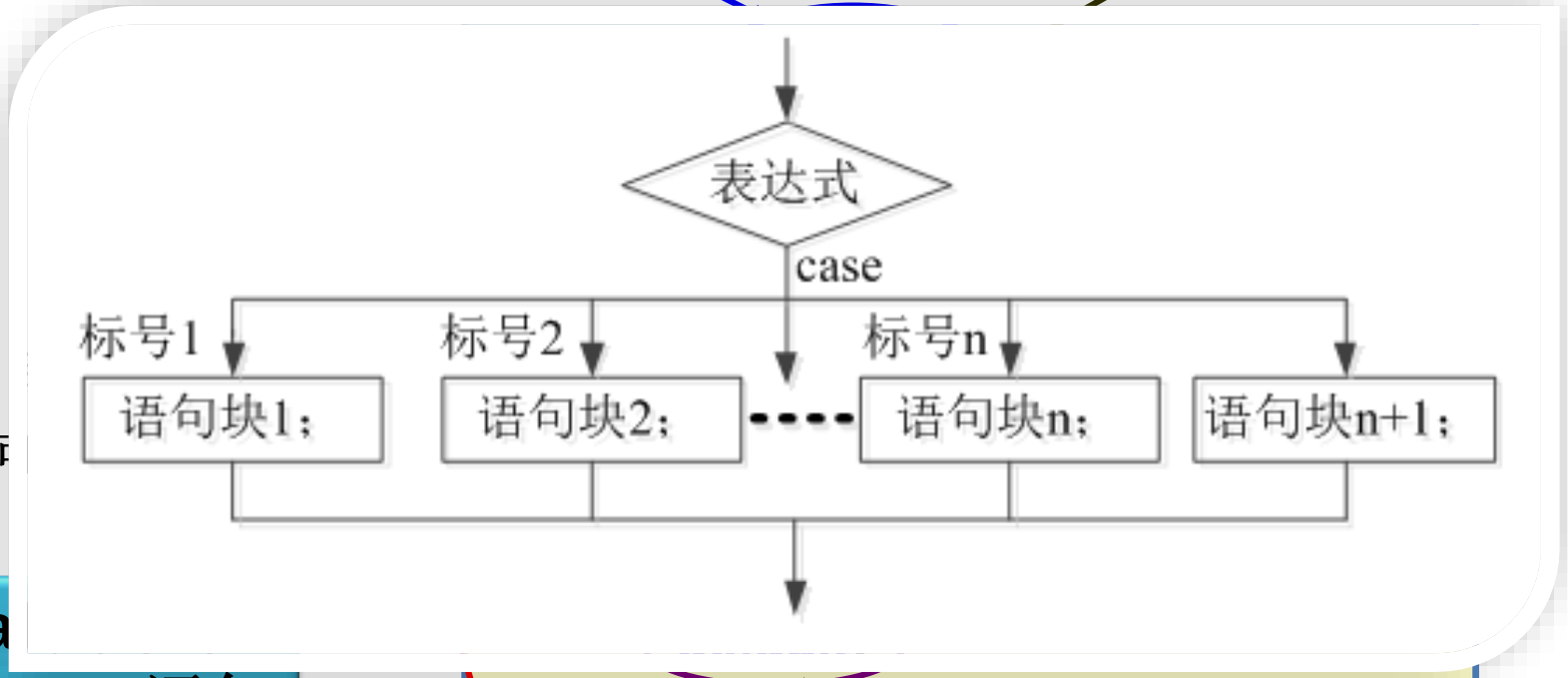
- 选择结构的基本运行符
- if-else选择结构
- **switch选择结构**
- 应用实例

===switch选择结构===

switch功能： 根据表达式的值来选择所要执行的语句
switch作用： 处理多分支选择问题

值为整数类型

有空格



brea
断switch语句

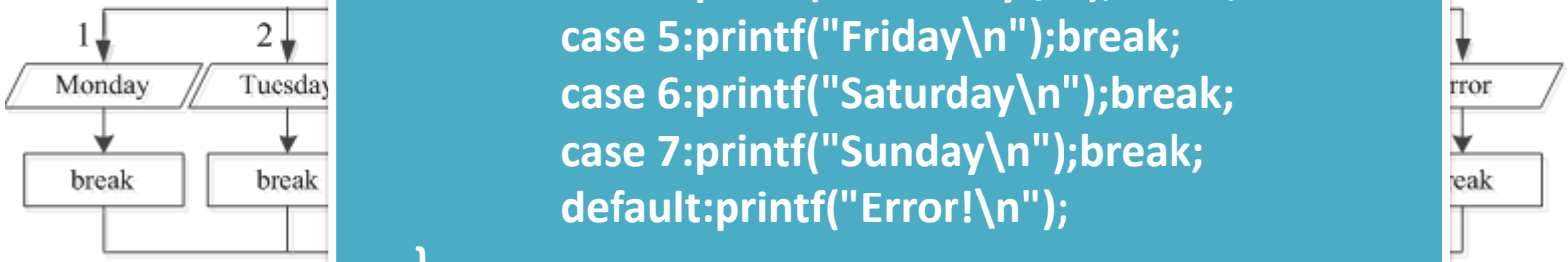
语句组n+1；
}

===switch选择结构

[例2-5] 输入 ...换成星期的英语... ...riday”

➢ 程序设计...

```
int main(void)
{
    int nDay;
    printf("input integer number: ");
    scanf("%d",&nDay);
    switch (nDay)    /*判定表达式*/
    {
            case 1:
                printf("Monday\n");
                break;
            case 2:printf("Tuesday\n");break;
            case 3:printf("Wednesday\n");break;
            case 4:printf("Thursday\n");break;
            case 5:printf("Friday\n");break;
            case 6:printf("Saturday\n");break;
            case 7:printf("Sunday\n");break;
            default:printf("Error!\n");
    }
    return 1;
}
```



===switch选择结构===

➢ if-else-if语句与switch语句

属性 语句	阅读与书写	运行效率	内存空间消耗	适用范围	可嵌套性	可替换性
if-else-if	较难	低	低	广	可以	替换switch容易
switch	容易	高	高	窄	可以	替换if-else-if难

(1) 阅读与书写

(2)运行效率

(3)内存空间的消耗

(4)适用范围

(5)可嵌套性

(6)可替换性

第2章 选择控制结构及应用

主要内容

- 选择结构的基本运行符
- if-else选择结构
- switch选择结构
- 应用实例

===应用实例===

[例2-6

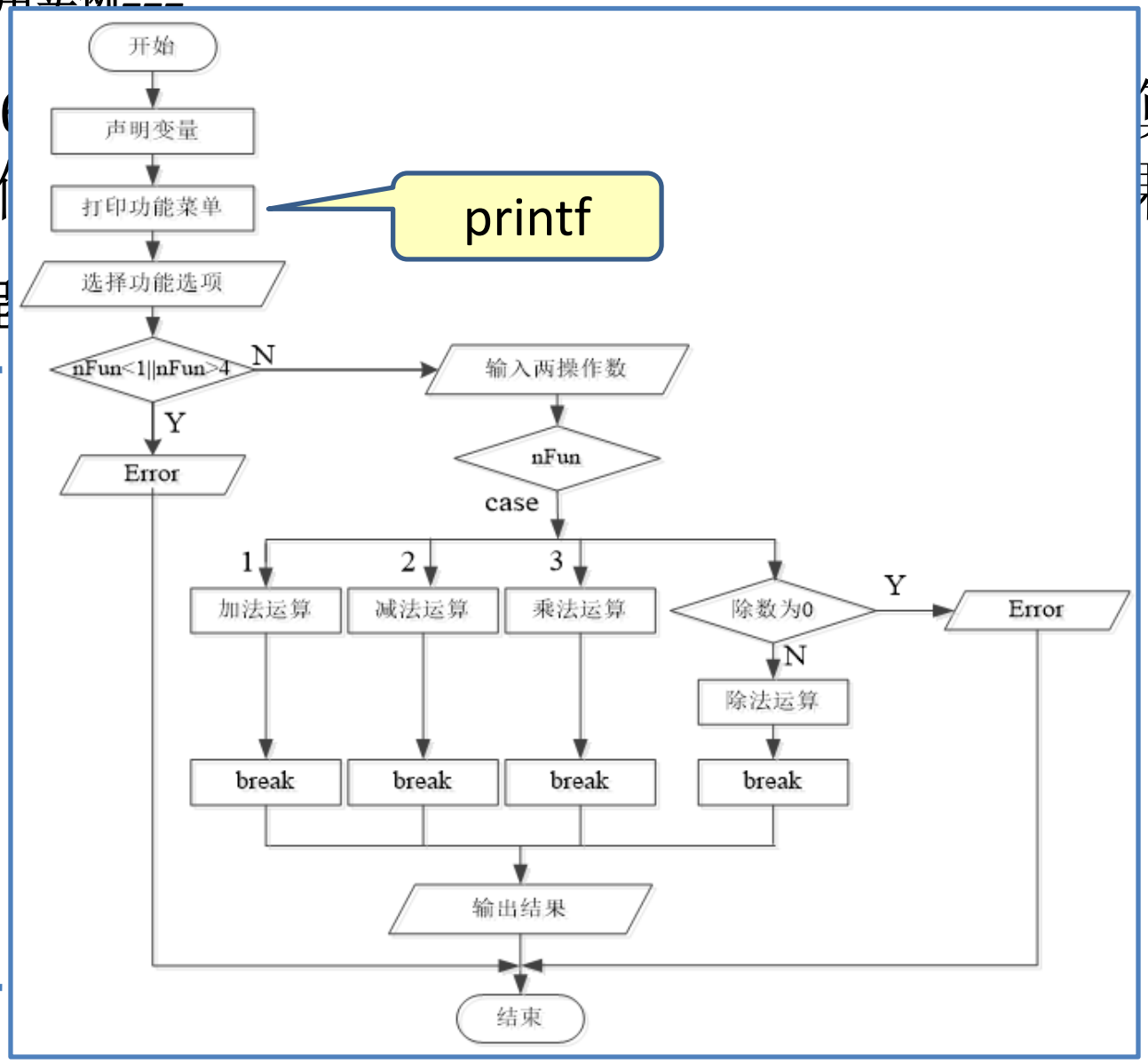
算功能，

对

果。

printf

➢ 程



if-else

switch

===应用实例===

[例2-6] 编写计算器程序。

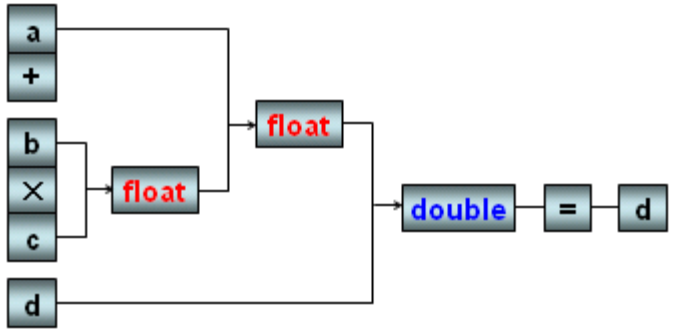
➢ 程序实现

```
#include <stdio.h>
void main(void)
{ double l,r,t;
   int f;
   printf("  ------------------------\n");
   printf("    plus--------1\n");
   printf("    substract--------2\n");
   printf("    multiplication--------3\n");
   printf("    divsion--------4\n");
   printf("  ------------------------\n");
   printf("  input for choice:  ");
   scanf("%d",&f);
if(nFun<1||nFun>4)
    printf("error！ \n ");
 else{
    printf("  input two operators：  ");

    scanf("%lf%lf",&l,&r);
    switch(f){
        case 1: t=l+r; break;
        case 2: t=l-r; break;
        case 3:  t=l*r; break;
        default: if(0.0!=r)

                    r=l/r;
                 else
                    printf("error！ Divisor is 0\n ");
                 break;
      }
       printf("  result：  %8.3lf\n", t);
   }
}
```

===应用实例===

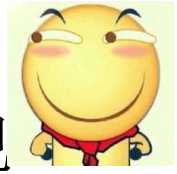
[例2-7] 学生成绩管理程 输入某个学生的某科成绩 中，按优、良、中、及格和不及格的

```
char a= 'a';  int b=12;  float c=11.23;
double d=123.12345;  double e;
e=a +b*c +d;
```



➢ 问题分析

用switch实现



类型转换

成绩是实数，switch只能整形判等

数据类型转换机制：隐式与强制
隐式：由低精度到高精度（自动）
强制：均可（手动）

强制类型转换要丢失精度

*低精度变量＝（**低精度数据类型**）高精*

===应用实例===

[例2-7] 学生成绩管理程序。要求：按百分制任意输入某个学生的某科成绩，将其存储到变量fScore中，按优、良、中、及格和不及格的等级输出。

- 问题分析

  用switch实现

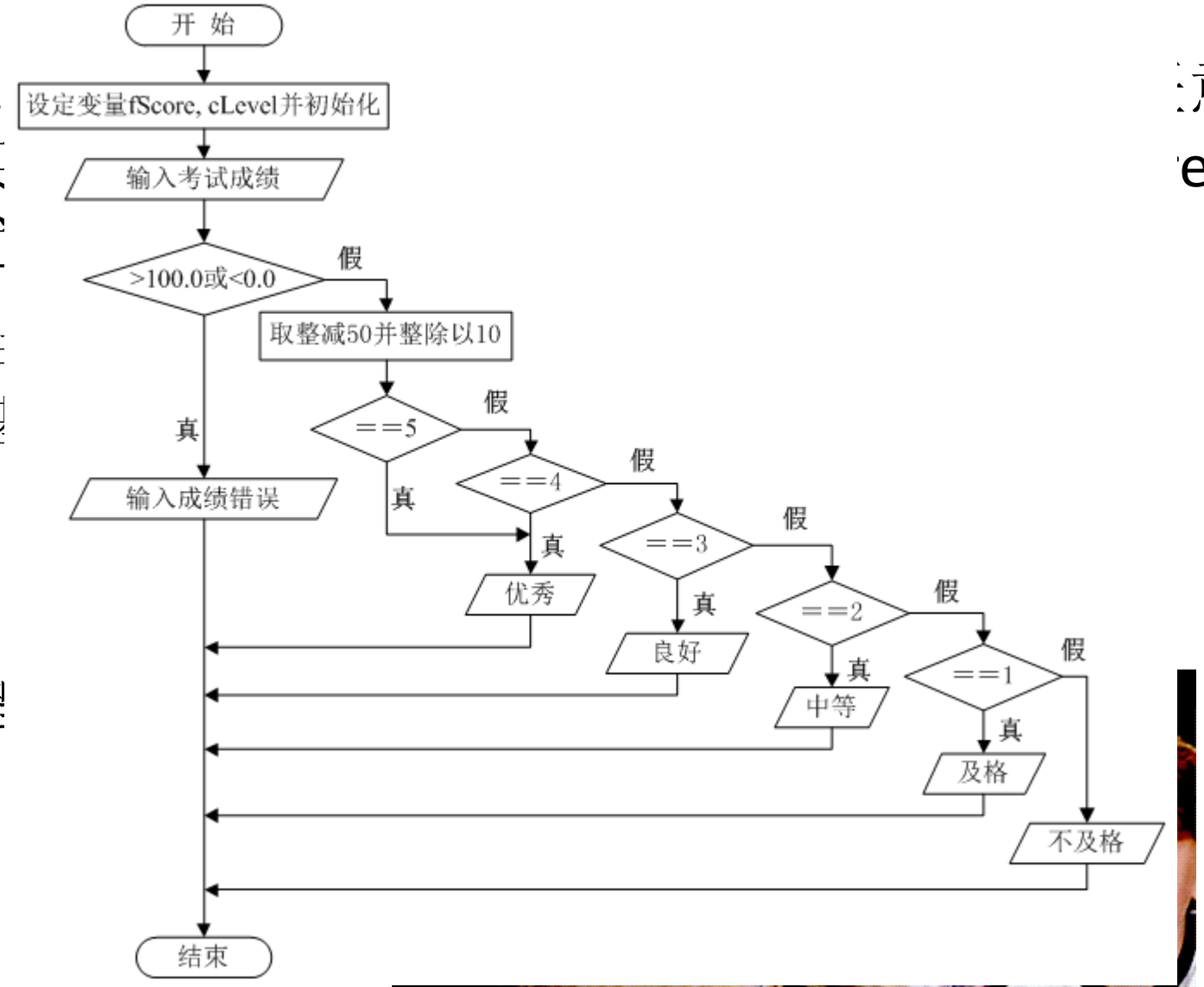
热身：输入一个浮点数，分别取出它的小数部分与整数部分，然后输出它们

```
#include <stdio.h>
int main(){
    float a;
    int b;
    scanf("%f",&a);
    b=(int)a;
    printf("%f; %d",b,a-b);
}
```

===应用实例===

[例2 意输入某 e中，按优

-
- 程



===应用实例===

➢ 程序设计描述

```
/* 考试成绩管理
purpose: 输入考试成绩，判断成绩等级
author : Xiaodong Zhang
created: 2017/01/31 15:58:22  */
#include <stdio.h>
#include <stdlib.h>
void main(void)
{
    float fScore;
    int nLevel;
    printf("请输入成绩： ");
    scanf("%f",&fScore);
   if(fScore>100.0||fScore<0.0)
   {
     printf("\n输入成绩错误！ ");
   }else {
```

```
nLevel=(int)(fScore-50.0)/10;
   switch(nLevel)
  {
     case 5:
     case 4:printf("优秀\n");break;
     case 3:printf("良好\n");break;
     case 2:printf("中等\n");break;
     case 1:printf("及格\n");break;
     default:printf("不及格\n");
   }
  }
}
```

本章小结

知识层面

- 关系运算符与表达式、逻辑运算符与表达式
- if语句的三种语法形式
- 三目运算符
- switch选择结构--break语句

方法层面

- 多分支选择结构的组织方式与设计方法--if-else-if和switch
- 对switch结构中有无break语句的使用与设计技巧
- 类型转换