第四章
程序设计基础

4.1 程序设计的概念

一、什么是程序

程序：计算任务的处理对象和处理规则的描述。

处理对象：指数据和信息。

处理规则：指处理动作和步骤。

程序：能实现特定功能的一组指令序列的集合。

指令：机器指令、汇编语言命令、高级语言语句
自然语言描述的运算或操作行为。

- 程序是程序设计或软件中最基本的概念。
- 程序是软件的本体，又是软件的研究对象。
- 程序的质量决定软件的质量。

4.1 程序设计的概念

二、程序设计

程序设计又称编程，是指设计、编制和调试程序的方法与过程，或研究、开发上述方法与过程中所涉及的理论、原则及技术的学科。

程序设计：计算机解决问题的全过程。

程序设计的基本过程：

①分析问题，明确任务；

②建立数学模型，选择合适的解决方案；

③确定数据结构和算法；

④编写程序；

⑤调试程序；

⑥整理文档，交付使用。

4.2 程序设计语言

程序设计语言：用于书写计算机程序的语言。

程序设计语言＝计算机语言

计算机语言：计算机所能够识别的语言。

文章	汉语、英语	字、词、语法结构	写文章
程序	计算机语言	数据表示、表达式 语句结构	编程序

4.2 程序设计语言

计算机语言的分类

应用范围	通用语言、专用语言 系统程序设计语言、科学计算语言 事务处理语言、实时控制语言
程序设计方法	结构化语言 模块化语言 面向对象语言
与硬件的联系程度	机器语言、汇编语言 } 依赖于计算机硬件(低级语言) 高级语言→与计算机硬件基本无关

4.2 程序设计语言

一、机器语言

机器语言：指计算机能够直接识别的基本指令的集合，它是以二进制代码表示的机器指令集合。

机器指令＝操作码＋操作数

从本质上说，计算机只能识别0和1两个数字，因此计算机能够直接识别的指令是由一连串的0和1组合起来的二进制编码。

4.2 程序设计语言

[例题]用Intel 8086指令系统来编写机器语言程序，要求完成 7+8=?

指令序号	机器语言程序	指令功能
1	10110000 00000111	把一加数 7 送到累加器 AL 中
2	00000100 00001000	把累加器 AL 中的内容与另一数相加,结果仍存放在 AL 中
3	11110100	停止操作

优点：占用内存少，执行速度快；

缺点：面向机器语言，通用性差，不易阅读和记忆，编程工作量大，难以维护。

4.2 程序设计语言

二、汇编语言

汇编语言：用符号代替机器指令所产生的语言。

例如，8086汇编语言对7+8=?编程

序号	汇编语言程序	语句功能
1	MOV AL,7	把加数 7 送累加器 AL 中
2	ADD AL,8	把累加器 AL 中的内容与另一数相加，结果存入 AL,即完成 7+8 运算
3	HLT	停止操作

优点：比机器语言程序容易阅读和修改

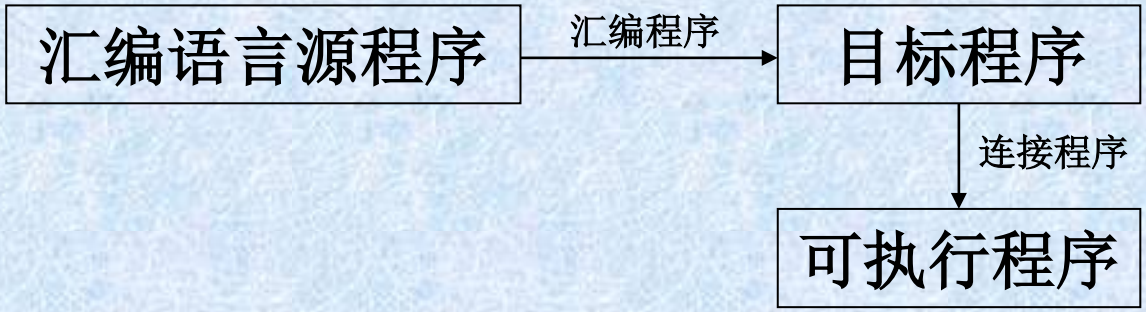
缺点：面向机器语言，通用性差。

4.2 程序设计语言

源 程 序：用汇编语言编写的程序。

汇编程序：负责翻译的软件。其主要作用是将源程序转换成用二进制代码表示的目标程序。

连接程序：将目标程序与库文件或其他目标程序连接起来形成可执行程序。



4.2 程序设计语言

三、高级语言

高级语言：用接近自然语言和数学语言的语法、符号描述基本操作的程序设计语言。

BASIC、FORTRAN、Pascal、COBOL、Delphi

Python、Java、C/C++

[例如]7+8=?，用C语言编程：

```
main()
{
   int a1;
   a1=7+8;
}
```

```
#include <iostream>
using namespace std;
int main()
{
    int a1;
    a1=7+8;
    cout<<"7+8="<<a1<<endl;
    return 0;
}
```

4.2 程序设计语言

优点：符合人类习惯，简单易学
通用性强，便于维护

源程序：用高级语言编写的程序。

翻译程序：将源程序翻译成用二进制代码表示的目标程序。

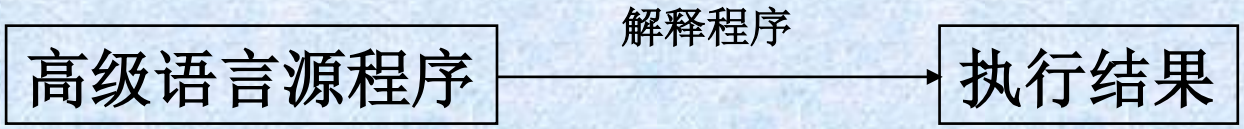
翻译程序的工作方式：

①解释方式：边解释边执行

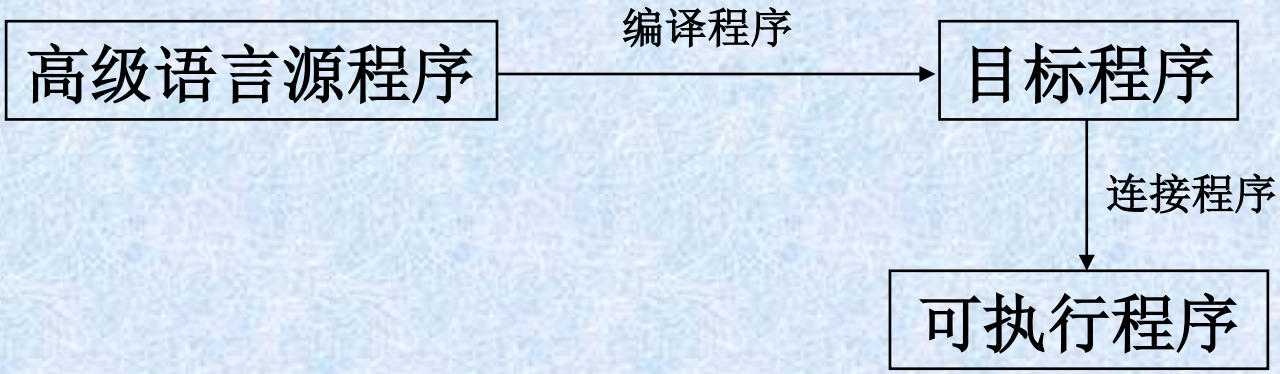
②编译方式：写程序、编译程序
运行程序

4.2 程序设计语言

解释方式



编译方式



4.2 程序设计语言

四、第四代语言

程序设计语言分代：

第一代语言/1GL：机器语言；

第二代语言/2GL：汇编语言；

第三代语言/3GL：面向过程的高级语言；

第四代语言/4GL：面向问题的、非过程化的语言。

第四代语言是快速开发应用软件的各种高生产率的软件工具统称。用户只需告诉系统“做什么”，而无须说明“怎么做”，系统将自动调用相应的过程，达到要实现的目标。

4.2 程序设计语言

第四代语言的特点：

①非过程化；

②支持面向对象技术；

③图形化、可视化。

第四代语言的分类：

①查询语言和报表生成器：SQL（数据库查询语言）、Power Builder、Delphi等。

②面向对象编程语言和网络语言：C++、HTML。

③可视化编程语言：VB、VC++、Java。

④软件开发环境：CASE（计算机辅助软件工程）、Raptor（算法原型工具）。

4.3 Raptor编程基础

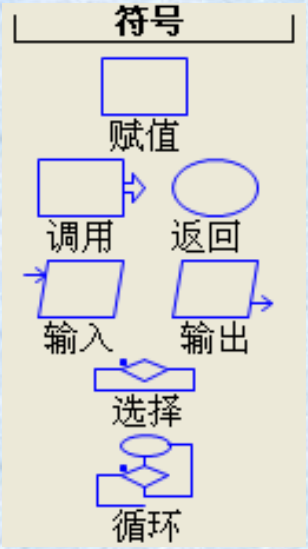
一、什么是Raptor

Raptor（the Rapid Algorithmic Prototyping Tool for Ordered Reasoning，用于有序推理的快速算法原型工具）是一种基于流程图的可视化编程环境，用它可以解决具体的编程问题。

- 流程图是一系列相互连接的图形符号的集合，每个符号代表要执行的特定类型的指令。
- 符号之间的连接决定了指令的执行顺序。
- 与其他程序设计语言相比，Raptor语法很少。
- Raptor是可视化的，可以让使用者看到程序语句的控制流程。

4.3 Raptor编程基础

二、Raptor的符号



赋值语句：应用各类运算来更改变量的值。

调用语句：调用子程序、自定义函数或Raptor过程。

返回语句：开始和结束的符号。

输入语句：输入数据，并将数据赋值给一个变量。

输出语句：显示变量的值或保存到文件中。

选择语句：经过条件判断后选择两种路径之一继续执行。

循环语句：允许重复执行一个或多个语句，直到某些条件为真值。

4.3 Raptor编程基础

三、变量和常量

1.变量

- 变量代表内存中具有特定属性的一个存储单元，它用来存放数据，即存放变量的值。
- 一个变量应该有一个名字，以便被引用。
- 变量名必须以字母开头，可以包含字母、数字和下划线，但不能有空格或其他特殊字符。
- Raptor变量只需在第一次使用时声明即可。
- 变量只能是数值类型或字符串类型。数值变量可以是整数，也可以是浮点数。

4.3 Raptor编程基础

2.常量

常量：在程序运行过程中，其值不能被改变的量。

Raptor没有为用户提供定义常量的功能，而只是在系统内部定义了若干符号表示常用的数值型常量。

当用户需要时，可使用代表这些常量的符号。

Pi（圆周率）：定义为3.1416

e（自然对数的底数）：定义为2.7183

True/Yes（布尔值真）：定义为1

False/No（布尔值假）：定义为0

4.3 Raptor编程基础

四、输入语句

输入语句：能使程序获取用户的输入数据。

在Raptor中，运行输入语句时，将出现提示信息，用户输入的值将存储到指定变量中。

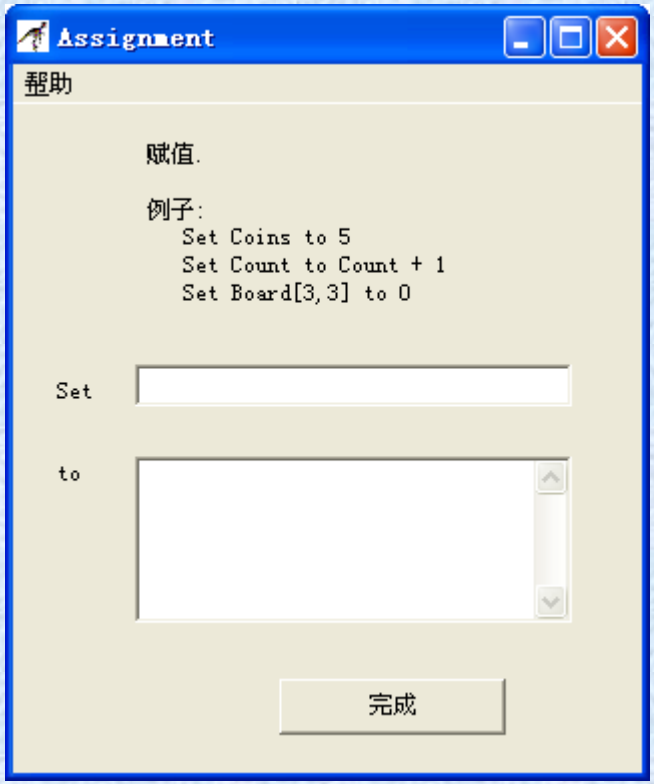
操作过程：把输入符号拖拽到流程图区域后，双击它将弹出[输入]对话框。

4.3 Raptor编程基础

五、赋值语句

赋值语句：用于变量的初始化或计算工作，然后将结果值存储到变量中。

操作过程：把赋值符号拖拽到流程图区域后，双击它将弹出Assignment对话框。



4.3 Raptor编程基础

赋值语句格式：

变量←表达式

赋值语句的执行过程是：

①计算赋值语句右边的表达式。

②将表达式计算结果赋予变量。

赋值语句的表达式可以是简单或复杂的公式，经过运算后得到一个值。

表达式可以是常量、变量与运算符和函数的组合。这里就涉及到Raptor内置函数、运算符以及运算优先级别。

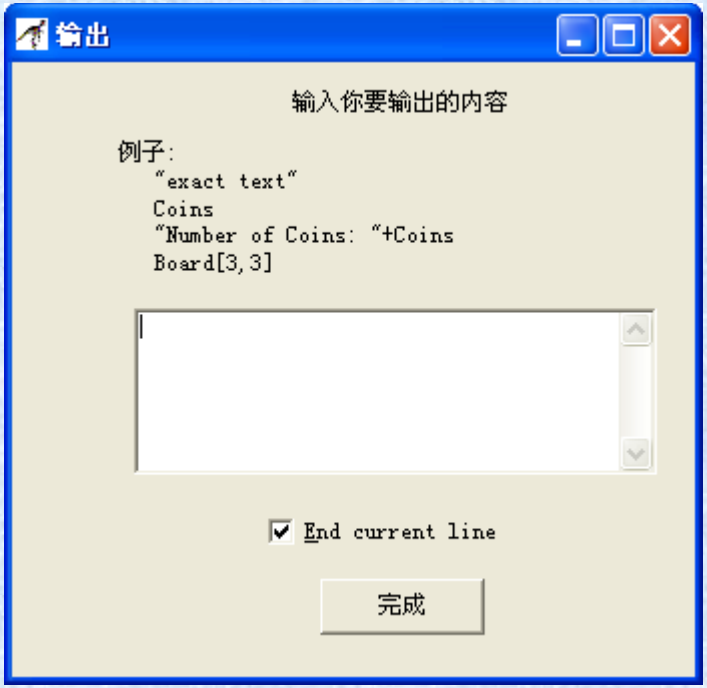
4.3 Raptor编程基础

六、输出语句

输出语句：将计算结果呈现给用户。

在Raptor中，运行输出语句时，将计算结果数据输出到[主控台]窗口。

操作过程：把输出符号拖拽到流程图区域后，双击它将弹出[输出]对话框。



4.3 Raptor编程基础

输出语句的输出方式：

①输出内容为纯文本，用引号括起来。

PUT "Goodbye, my friend."

②输出内容为初始数据、对计算的描述和计算结果。

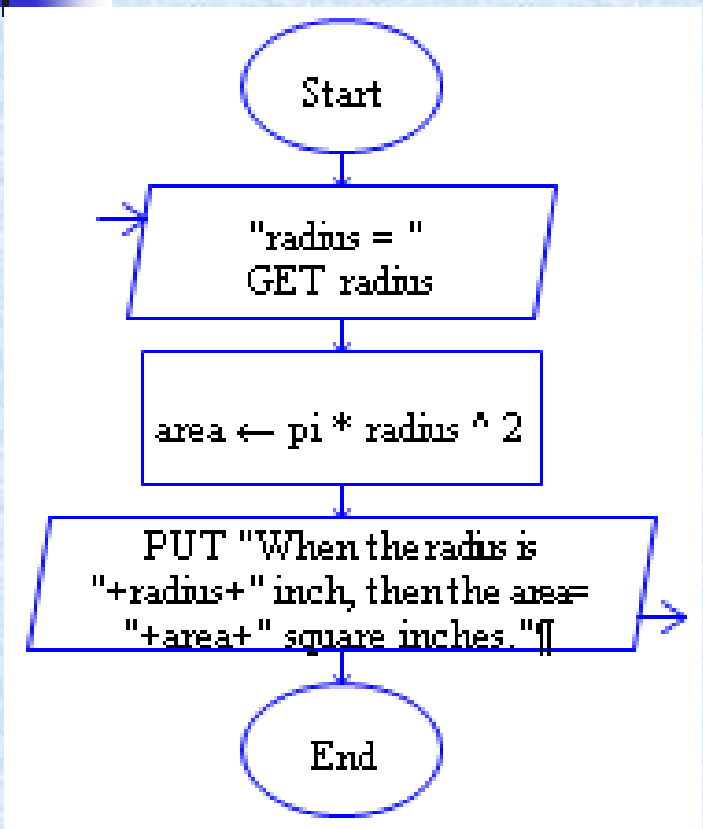
PUT "When the radius is "+radius+" inch, then the area= "+area+" square inches. "

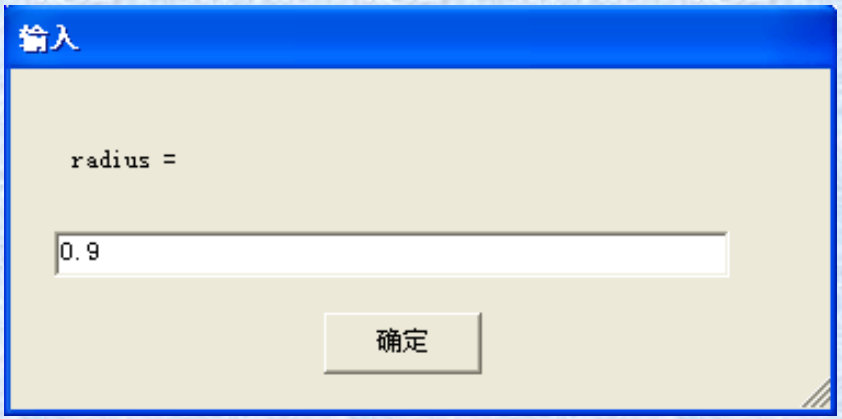
[例4-1] 已知圆的半径，求解圆的面积。

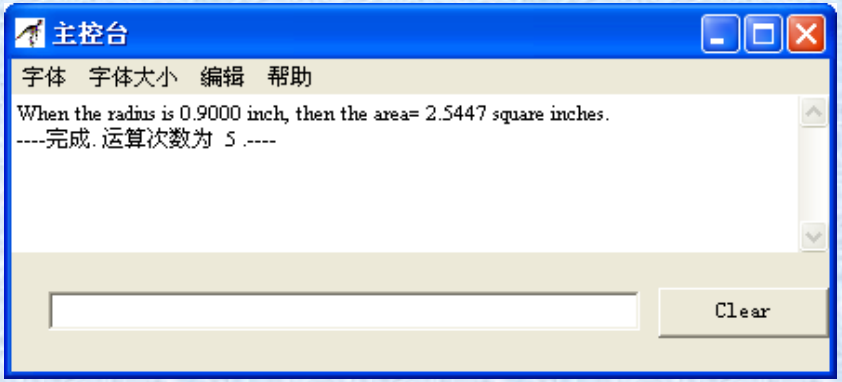
[解] 圆的面积公式：area=pi*radius^2

利用Raptor编制求解圆面积流程图。

4.3 Raptor编程基础



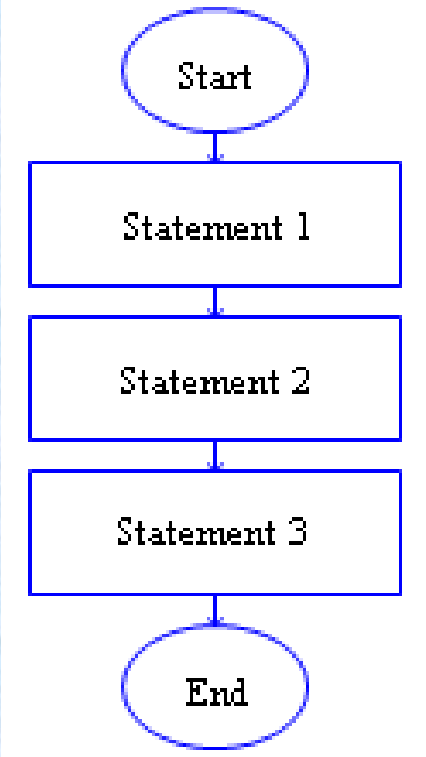




4.4 Raptor控制结构

一、顺序结构

顺序结构：将每条语句按顺序排列，程序执行时从Start语句顺序执行到End语句，箭头连接着语句并指示程序的执行方向。



4.4 Raptor控制结构

[例4-2] 计算存款利息。

有10000元，想存一年。有2种方法可选：

①活期存款，年利率为r1；

②一年定期存款，年利率为r2。

请分别计算出一年后按2种方法所得到的本息和。

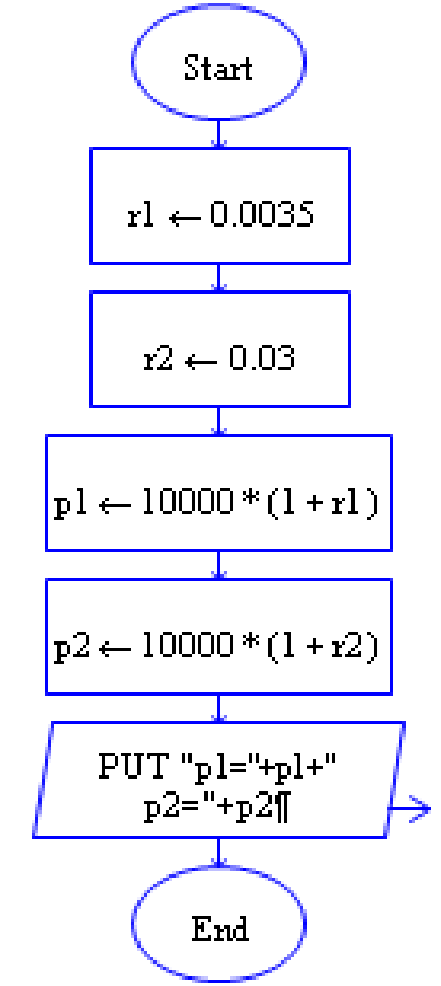
[解] 首先确定计算本息和的公式。

活期存款一年后本息和为：p1＝10000×(1+r1)；

定期存款一年后本息和为：p2＝10000×(1+r2)。

在Raptor中编制相应的流程图。

4.4 Raptor控制结构



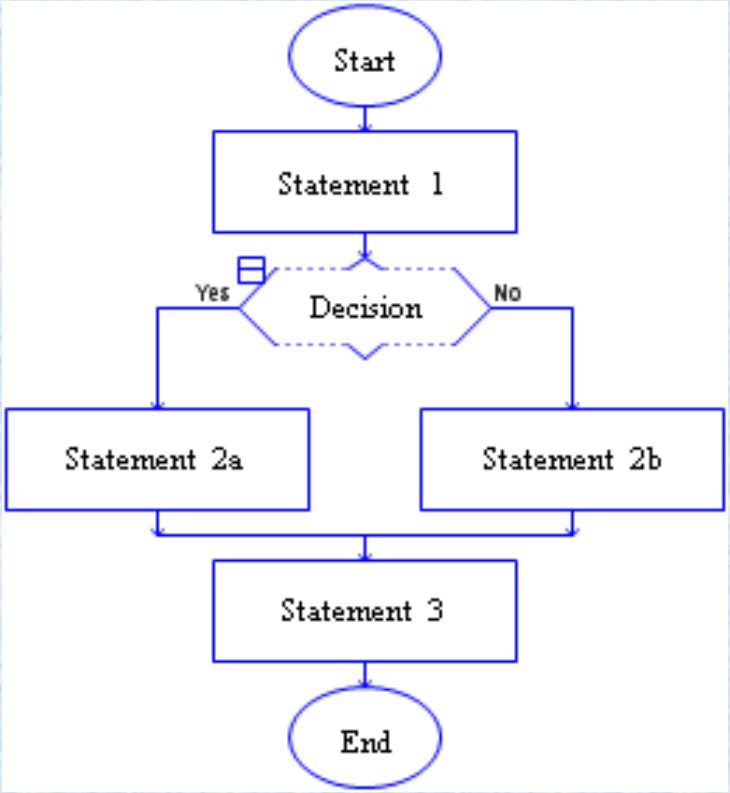
运行结果：

```
p1=10035   p2=10300
----完成. 运算次数为 7 .----
```

4.4 Raptor控制结构

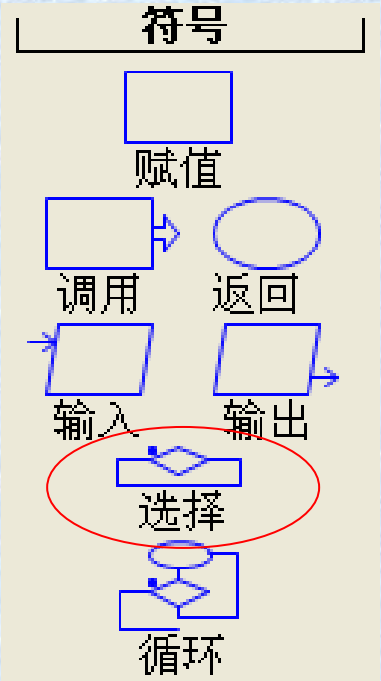
二、选择结构

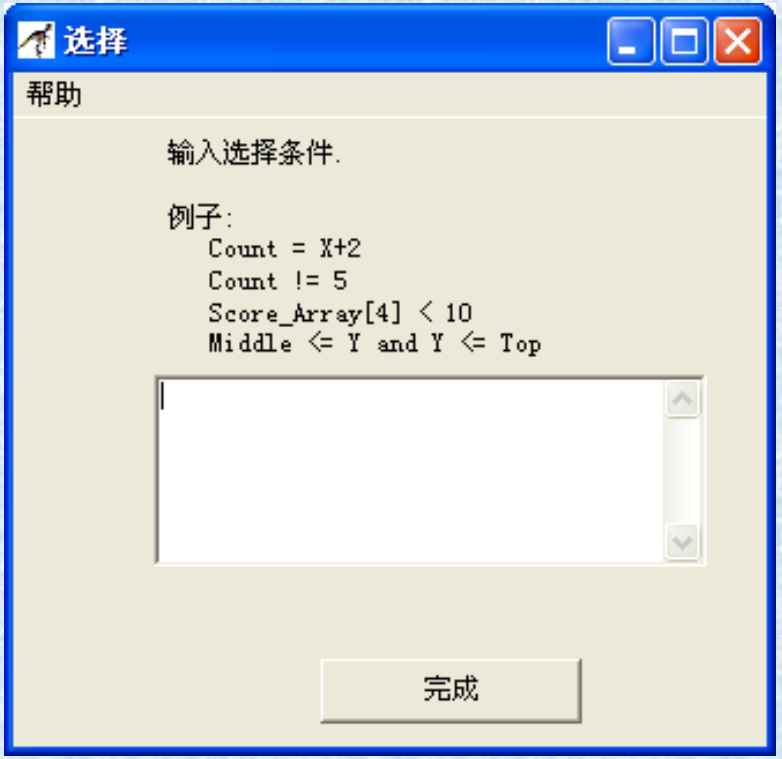
选择结构：判断某个条件是否满足来决定程序的执行方向，即从给定的两种操作选择其一。



4.4 Raptor控制结构

- Raptor选择语句是用菱形的选择符号表示，用Yes/No表示对问题的判断结果以及决策后程序语句的执行方向。
- 当把选择符号拖曳到编辑区域并双击菱形符号时，会弹出[输入选择条件]对话框。

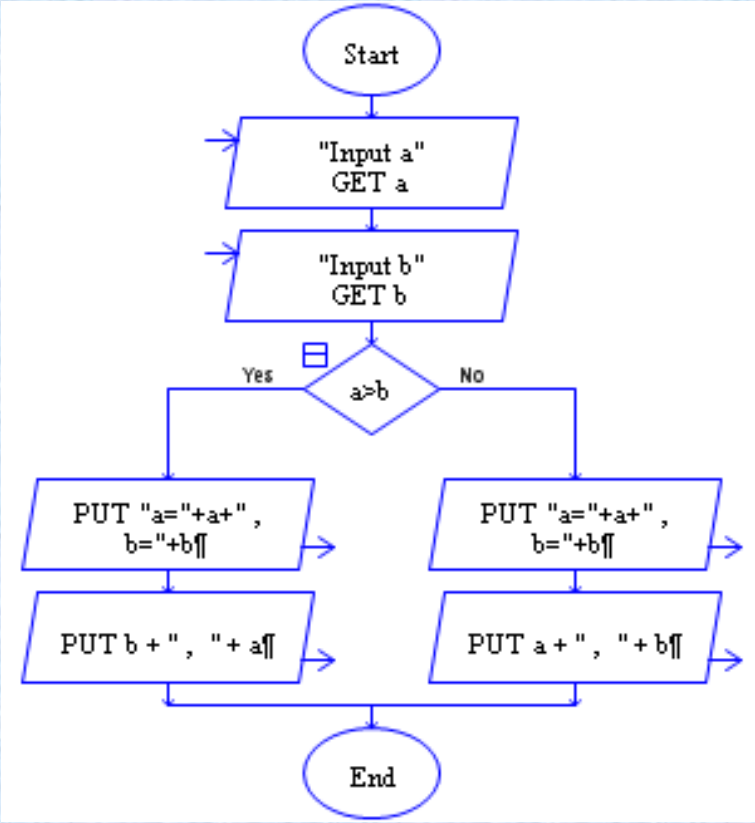




4.4 Raptor控制结构

[例4-3] 输入2个数a、b，要求按照从小到大的顺序显示结果。

[解] 因为该题比较简单，可以直接画出其流程图。



运行结果：

```
a=7,b=3
3,7
----完成.运算次数为 7.----
a=12,b=20
12,20
----完成.运算次数为 7.----
```

4.4 Raptor控制结构

[例4-4] 求$ax^2+bx+c=0$方程的根，a、b、c由键盘输入。

[解] 阅读教材相关内容，思考下列问题：假如将选择条件 disc<0 改变为 disc>=0，那么，流程图将如何变化？

（学生思考并回答，停留3分钟）

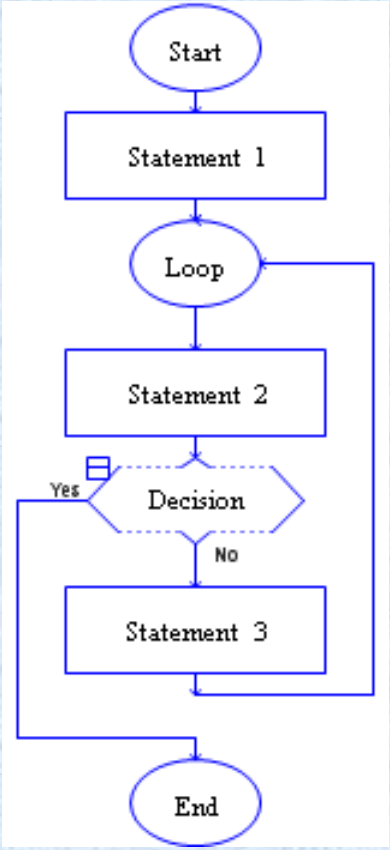
[答案] Yes和No互换，

或选择结构两条分支路径的所有语句互换。

4.4 Raptor控制结构

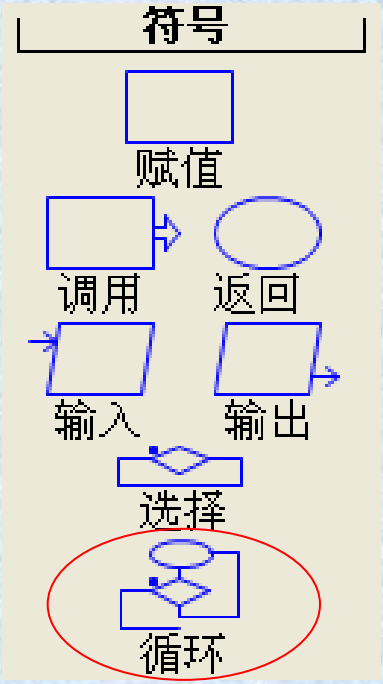
三、循环结构

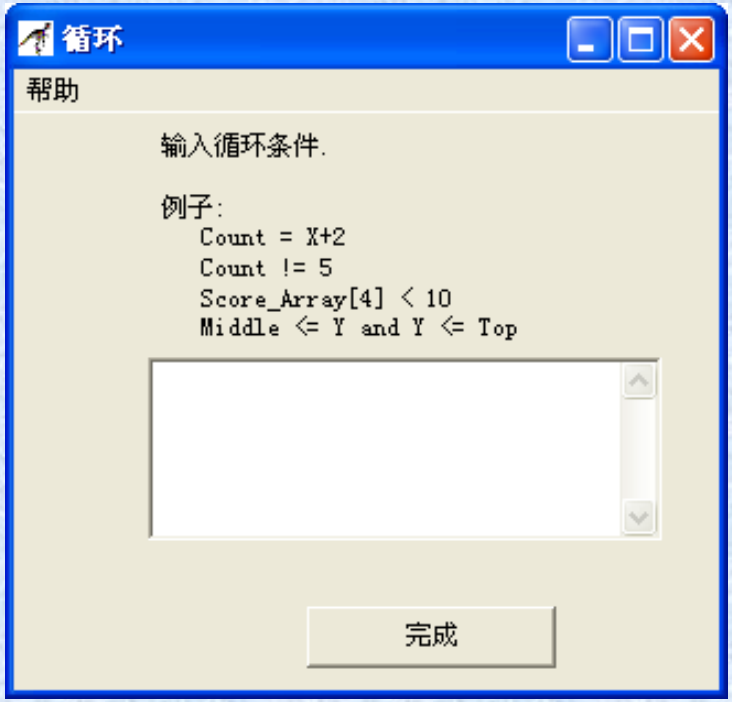
循环结构：反复执行某一段程序，直到某种条件满足时才结束执行该段程序的一种结构。



4.4 Raptor控制结构

- Raptor循环语句是用椭圆和菱形符号组合在一起用来表示循环过程，循环次数由菱形符号中的表达式控制。
- 当把循环符号拖曳到编辑区域并双击菱形符号时，会弹出[输入循环条件]对话框，输入判断条件是决定循环是否继续还是退出的关键因素。

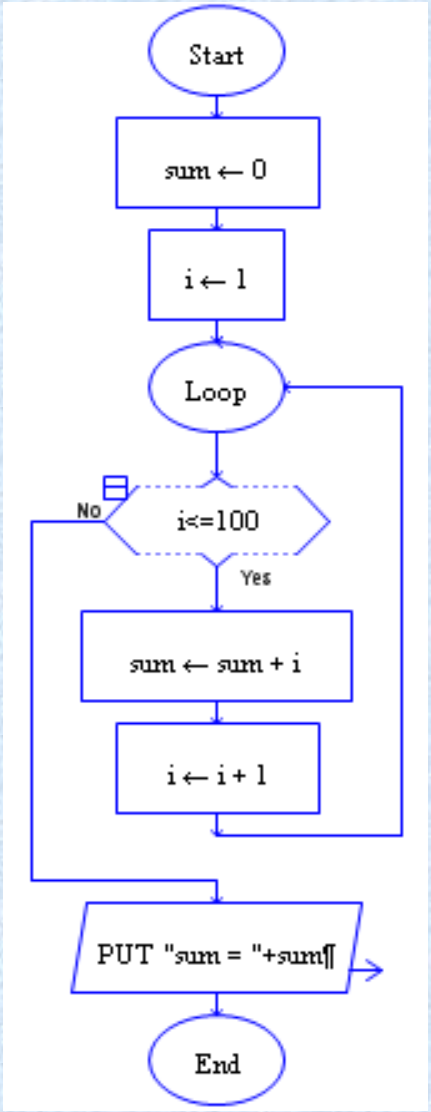




4.4 Raptor控制结构

[例4-5] 求1＋2＋3＋…＋100。

[解] 累加问题，要进行100-1次加法运算，可用循环结构来实现。首先要找出每次累加时的规律，后一个数是前一个数加 1 。因此不需要每次利用输入语句从键盘录入数据，只须在加完上一个数i后，使i加 1 就可得到下一个数。



运行结果：

```
sum = 5050
----完成.运算次数为 407 .----
```

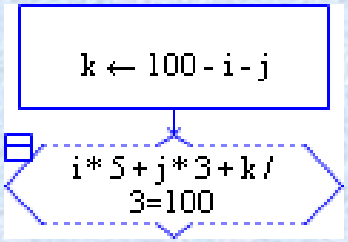
4.4 Raptor控制结构

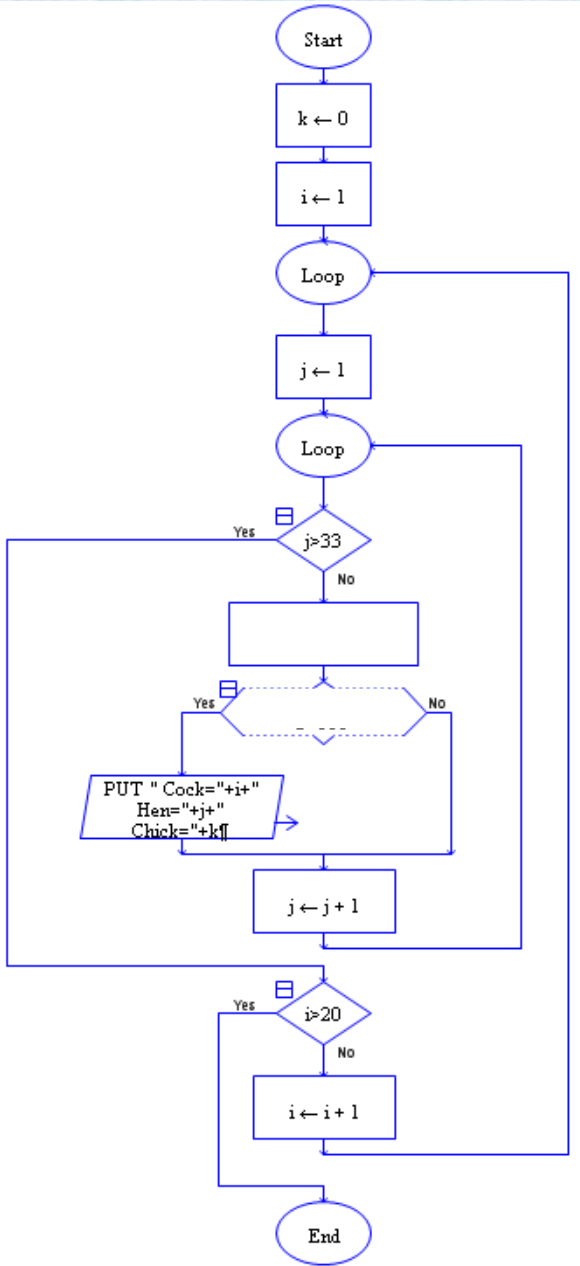
[例4-6]百钱买百鸡问题。在例3-4的基础上对流程图进行修改，使它适应Raptor编程环境。

[解] 阅读教材相关内容，思考并填充流程图中 2 处空格。

（学生思考并填空，停留3分钟）

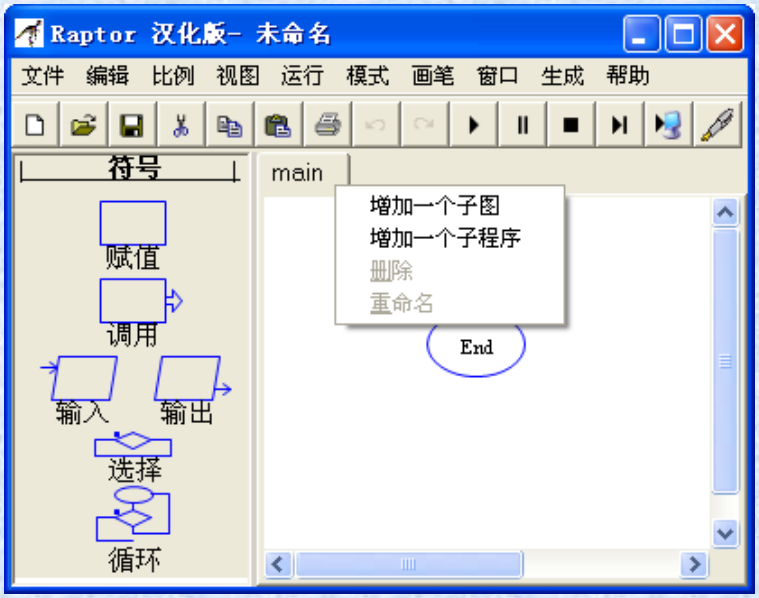
[答案]





4.5 调用语句与子图

- 调用语句是在主程序中调用子模块或子程序、自定义函数或Raptor过程，Raptor将子程序称为子图。
- 子图可以将Raptor程序分解成逻辑块，由主程序来调用它们，这样可简化程序设计的工作。
- 打开Raptor，主窗口左上角有main。若要创建子图，右击main，然后选择[增加一个子图]命令即可。



4.5 调用语句与子图

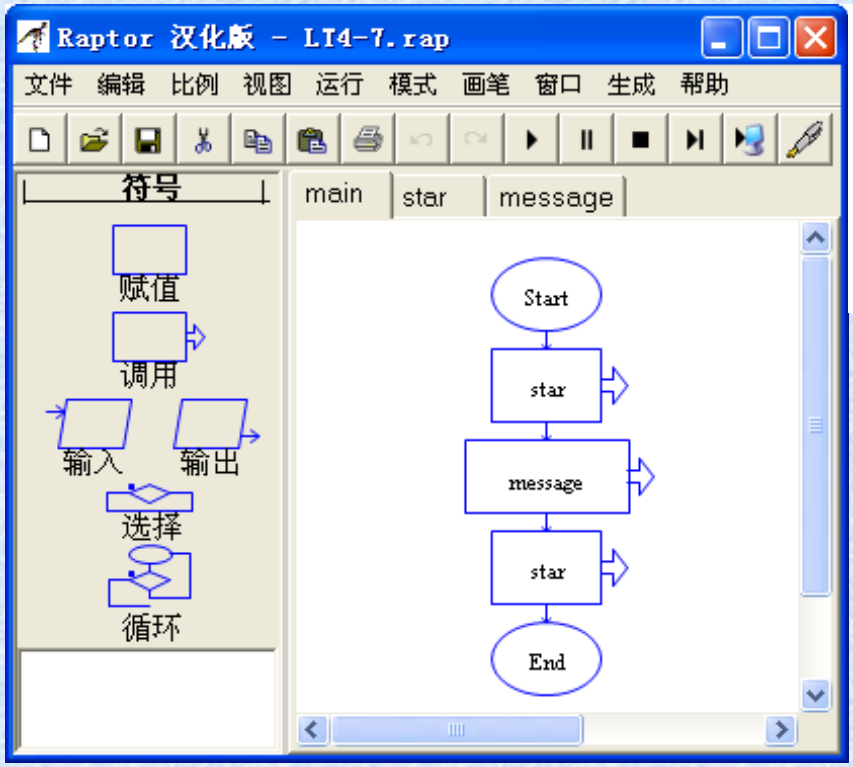
- 如果需要调用子图，只需把调用语句插入到相应的位置，输入调用子图名称即可。
- 子图可以被主程序调用，也可以被其他子图调用，还可以被子图本身调用。
- 运行程序时，若遇到调用语句，程序控制权转移到子图。当子图执行完后，程序控制权自动返回到调用语句的下一条语句继续执行。

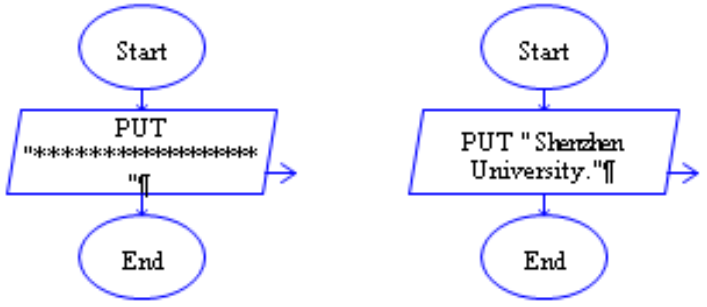
[例4-7] 要输出以下的结果，用子图实现。

```
**********************
 Shenzhen University.
**********************
```

4.5 调用语句与子图

[解]用一个子图star来实现输出一行“*”号功能，再用另一个子图message来输出中间文字信息，最后主程序分别调用这两个子图。





运行结果：

```
**********
 Shenzhen University.
**********
----完成.运算次数为 14 .----
```