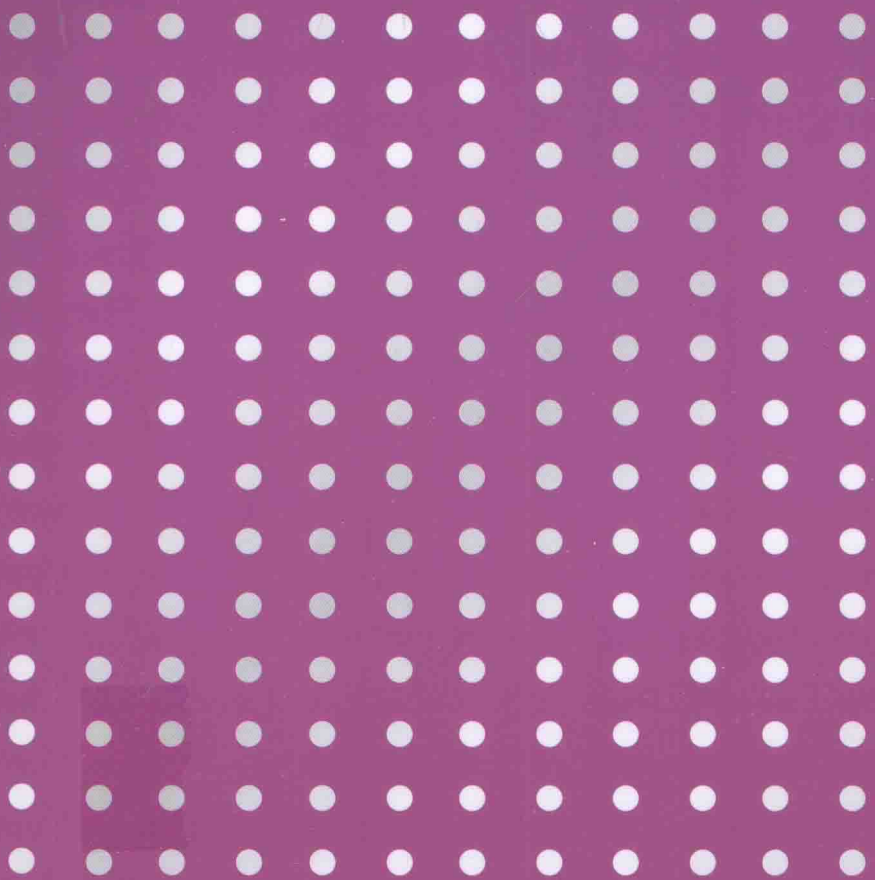
高等院校信息技术规划教材

基于RAPTOR的可视化计算案例教程

程向前 周梦远 编著

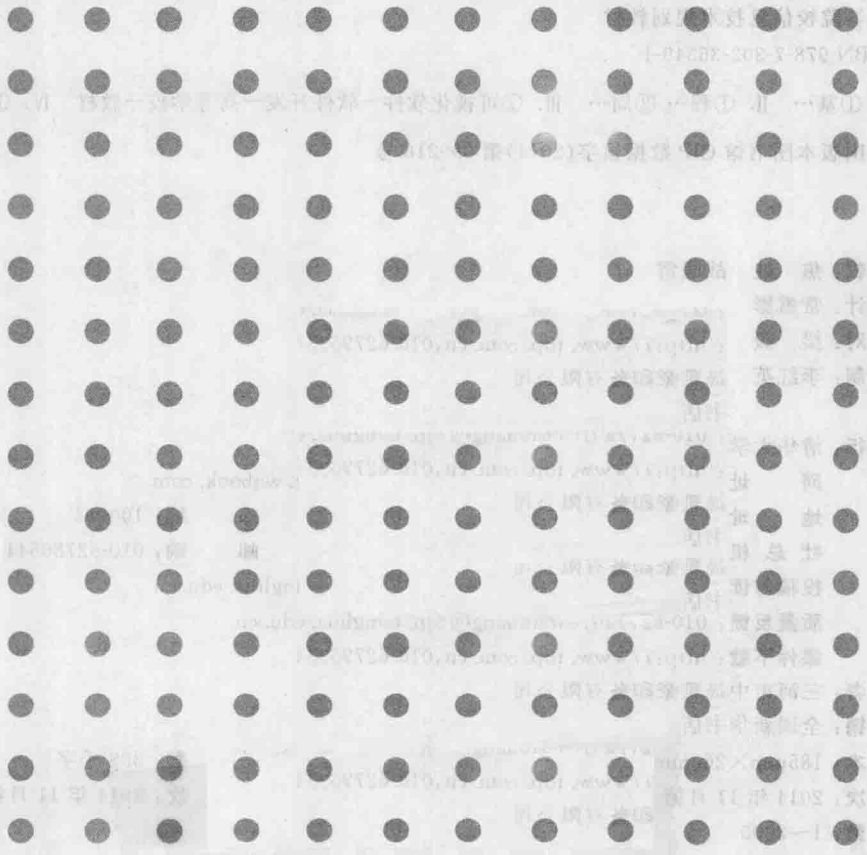


清华大学出版社

高等院校信息技术规划教材

基于RAPTOR的可视化计算案例教程

程向前 周梦远 编著



清华大学出版社
北京

内容简介

全书分为两大部分，第一部分为可视化编程工具 RAPTOR 应用基础；第二部分为问题求解案例，分为“枚举和数论”、“游戏与博弈”、“图论”和“学科应用”4 个部分。书中案例大部分为设计型实验，取材于学生自选并实现的算法作业。本书着眼于以学生为学习主体精神指导下的实践与创新活动，充分体现现代大学生的思想与表达方式的多样性、难能可贵的创新探索和旺盛的求知欲和好奇心。为读者跨入计算机算法的大门开辟了富有趣味、简便快捷的途径。

本书适合作为大学计算机、计算思维导论和计算机科学导论课程的配套实验教材，也可以独立设课，还可以供自学者学习参考。

图书在版编目(CIP)数据

基于 RAPTOR 的可视化计算案例教程/程向前，周梦远编著. —北京：清华大学出版社，2014
高等院校信息技术规划教材
ISBN 978-7-302-36349-1

Ⅰ. ①基… Ⅱ. ①程… ②周… Ⅲ. ①可视化软件—软件开发—高等学校—教材 Ⅳ. ①TP311

中国版本图书馆 CIP 数据核字(2014)第 099216 号

责任编辑：焦　虹　战晓雷
封面设计：常雪影
责任校对：梁　毅
责任印制：李红英

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn,010-62795954

印 装 者：三河市中晟雅豪印务有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：16　　**字　　数**：398 千字
版　　次：2014 年 11 月第 1 版　　**印　　次**：2014 年 11 月第1次印刷
印　　数：1～2000
定　　价：33.00 元

产品编号：054384-01

前言 foreword

算法思维是计算思维的核心之一，在以往的大学计算机和计算机科学导论课程中，作者苦于没有适宜的教学平台来表现计算机科学思维，更谈不上让普通高校学生在接触计算机课程的初期，就有机会来体验和探索计算机科学的丰富乐趣与博大精深。

本书是作者在三年来“计算概论”教学实践的基础上，通过收集整理西安交通大学少年班预科二年级学生的算法作业编写而成的。目的是为计算机科学的初学者提供一个面向问题求解的实践方案。

全书主要包括两大部分，第一部分为可视化编程工具RAPTOR应用基础；第二部分为问题求解案例。书中案例大部分为设计型实验，取材于学生自选并实现的算法作业。

作为一种新型的教学工具软件，RAPTOR迄今对大部分国内读者来说仍然是比较陌生的。所以本书前两章主要介绍RAPTOR编程环境、图形和视窗编程基础和常用算法的实用程序。供初次接触RAPTOR环境的读者参考。需要对RAPTOR可视化编程工具进行深入了解的读者可参考由清华大学出版社出版的《RAPTOR程序设计案例教程》。

本书的问题求解案例是从作者教学活动中积累的上百个案例中选取的。入选案例体现了学生思维的多样性，分为“枚举和数论”、“游戏与博弈”、“图论”和“学科应用”4个部分。本书作为算法实验的入门读物，创新性主要体现在第2章“RAPTOR图形与视窗交互”、第4章“游戏与博弈”和第5章“图论”中，这些内容充分体现了可视化计算手段在提高算法的表达能力和趣味性方面的优势，对深入引导学生的计算思维能力可以起到示范作用。大学生具有求知的好奇心，本书中可视化算法的探索是在教师引导、学生自主选题的基础上完成的，而探索的成果体现在全书尤其是第6章“学科应用”中。

本书案例的编写基本依照了波利亚(George Polya)问题求解的四部法则(理解问题、制定方案、实施方案、回顾与反思)进行。全书案例的电子文件放在清华大学出版社网站上(http://www.tup.com.cn)，

读者可以下载后试运行，进行算法复杂性评估，并改进或者将其用于同类其他问题的求解。

为了让大学新生能够了解算法案例问题的求解过程，本书尽量保持原始案例的编写风格和多样性，这就意味着这些算法案例并不完美，存在很大的改进余地。这是本书在算法案例部分没有给出习题的主要原因，这一部分的习题就是改进和完善本书中的案例，或者将其中的某些实现方法应用到读者自身可以联想到的算法实现，这些可以作为读者自主学习的作业。

本书由程向前负责全书的构架设计与文稿编写，周梦远负责全书案例RAPTOR流程图设计、改进与调试。本书选取了西安交通大学少年103、104、111、112、113、114班的高航(求解哈密顿回路的存在)、李浩成(SCC的求解)、张钱东(24点计算的解空间探索)、陈明博(矩阵算法)、杨金成(稳定婚姻算法等)、滕一铭(中国邮递员问题)、朱贾(求关系网的最小分割)、夏子豪(视窗下的鼠标交互)、何自惟(求解冰激凌车问题)、马腾(古希腊点灯术)、黄喆(图形视窗下输入图并产生邻接矩阵)、姜博馨(人机猜数字游戏)、田凌云(视窗下的键盘交互)、段嘉炜(钟摆的动画效果)、李婧涵(阶梯数求解)、杨泽(素数环)、郭宁(狐狸与鹅的游戏)、于弦(求解绳子问题)、郭力豪(细菌繁殖与随机图)等同学的作业，以及软件103班王嘉伟(用随机法求解居住隔离模型)所提交的算法实现案例，作为全书的基本素材。

上述可视化算法案例的交流视频由少年班同学自行录制和制作，并在优酷网站上发布，有兴趣的读者可以在线观看，以了解算法案例的原作者的设计和实现意图。

西安交通大学教务处拔尖办为少年班“计算概论”课程的教学改革提供了项目支持，作者在此表示衷心感谢。

由于时间仓促，本书在文字和案例上一定存在瑕疵，恳请读者批评指正。

作　者

2014年1月于西安交通大学

目录 Contents

第1章 chapter 1

RAPTOR 计算环境

本章学习目标：

- RAPTOR 计算环境的基本概念
- RAPTOR 环境下算法设计的主要手段

1.1 RAPTOR 的基本概念

RAPTOR(Rapid Algorithmic Prototyping Tool for Ordered Reasoning，用于有序推理的快速算法原型工具)是一种可视化的程序设计环境，专为程序和算法设计的基础课程的教学提供实验环境。

在 RAPTOR 上的程序和算法设计采用了流程图仿真的方式，而流程图几乎是所有计算机程序设计学习中的必经之路，所不同的只是传统流程图是静态的，不可运行，不可在计算机上直接验证设计结果，而这一切在 RAPTOR 中则不成问题，除了可以直接运行设计成功的算法流程图之外，RAPTOR 的设计成果还可以直接转换成 C++、C# 和 Java 等高级程序语言，这就为程序和算法的初学者提供了一个平缓、自然的学习阶梯。

RAPTOR 的特点如下：

(1) 语言简单、紧凑、灵活(6 个基本语句/符号)。使用流程图形式实现程序设计，使得初学者无须花费太多时间，就可以进入问题求解的实质性算法学习阶段。

(2) 具备基本运算功能(18 种运算符)，可以实现大部分基本运算。

(3) 具备基本的数据类型与结构，提供了数值、字符串、字符 3 种数据类型以及一维和二维数组。组合以后，可以实现大部分算法所需要的数据结构，包括堆栈、队列、树和图。

(4) 具有严格的结构化的控制语句。

(5) 语法限制宽松，程序设计自由度大。例如，在一个数组中，可以存在不同的数据类型，使得数据库类的记录实现有了可能。

(6) 可移植性好，程序的设计结果可以直接执行，也可以转换成其他高级语言，如 C++、C#、Java 和 Ada 等。

(7) 程序的设计结果可以编译成可执行文件，直接运行。

(8) 支持图形库应用，可以实现计算问题的图形表达和图形结果输出。

(9) 支持面向过程和面向对象的程序和算法设计。

(10) 具备单步执行、断点设置等重要的调试手段，便于快速发现问题和解决问题。

1.2 RAPTOR 的基本程序环境

RAPTOR 程序是一组连接的符号，表示要执行的一系列动作。符号间的连接箭头确定所有操作的执行顺序。RAPTOR 程序执行时，从开始(Start)符号起步，并按照箭头所指方向执行程序。RAPTOR 程序执行到结束(End)符号时停止。最小的 RAPTOR 程序(什么也不做)如图 1-1 所示。在开始和结束符号之间插入一系列 RAPTOR 符号，就可以创建有意义的 RAPTOR 程序。由于 RAPTOR 符号(赋值、调用、输入、输出、选择和循环)完成的任务与大部分程序设计语言中相应功能的语句或指令相当，因此，本书在不容易引起误解的地方，也使用“语句”或“指令”来描述这些符号的应用。

图 1-1 开始和结束符号

1.2.1 基本符号

RAPTOR 有 6 种基本符号，每个符号代表一个独特的指令类型。基本符号如图 1-2 所示。首先介绍赋值(Assignment)，调用(Call)，输入(Input)和输出(Output)4 种语句，而选择(Selection)和循环(Loop)语句，将在稍后解释。

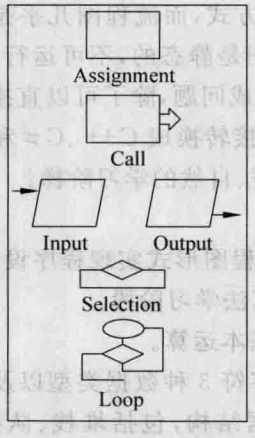


图 1-2 RAPTOR 的 6 种基本符号

典型的计算机程序有 3 个基本组成部分：

(1) 输入(Input)：在 RAPTOR 中，输入指令实际上以变量赋值的方式得到完成计算任务所需要的数据值。大部分初学者使用输入语句从键盘接收用户输入的初始数据，经过加工语句处理后，通过输出语句将结果输出到计算机屏幕上。输入语句也可以通过文件输入数据，以提高程序设计和运行的自动化程度，减少用户与程序的交互步骤，使得程序可以真正解决工程和科学研究问题。在 RAPTOR 图形窗口中，输入语句也可以接收来自鼠标的操作指令。

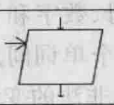
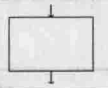
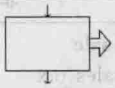
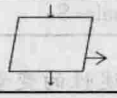
(2) 处理(Process)：通过操作数据值来完成任务。处理语句主要包含各种运算与赋值操作，负责将初始数据加工成运算结果，或者产生运算过程中的各类中间结果。在加工过程中，需要关注各种数据的类型，因为运算处理与数据类型紧密相关。例如，可以将两个数值相乘得到乘积；但是，将两个人名直接相乘，则得不到任何有意义的结果。

(3) 输出(Output)：显示(或保存)加工处理后的结果。大部分初学者会将计算结果直接在屏幕上输出。但输出语句也可以将计算结果输出到文件，保存在文件中的计算结果可以为其他的各种后续处理提供输入数据。例如，对一个科学计算模型进行模拟计算

的数据结果可以保存在数据文件中，为后续的统计和绘图软件提供输入数据。在RAPTOR中也有图形处理功能，一些计算结果可以设计成图形形式输出。

上述3个组件与RAPTOR指令形成直接的关系，如表1-1所示。

表1-1 4种RAPTOR基本指令说明

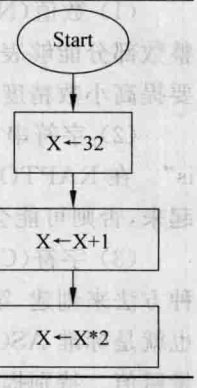
程序的基本组成部分	符号	名称	说明
输入		输入语句	允许用户输入数据，并将数据值赋给一个变量
处理		赋值语句	使用各类运算来更改变量的值
		过程调用	执行一组在命名过程中定义的指令。在某些情况下，过程中的指令将改变一些过程的参数(即变量)
输出		输出语句	显示变量的值(或保存到文件中)

这4个指令的共同点是，它们都对变量进行某种形式的操作。要了解如何设计程序，进而使之成为可以运行的计算机算法，必须明白变量的概念。

1.2.2 变量

变量(variable)表示的是计算机内存中的位置，用于保存数据值。在任何时候，一个变量只能容纳一个值。然而，在程序执行过程中，变量的值可以改变。这就是为什么它们被称为“变量”的原因。例如，某变量名称为X的变量，其值的变化过程如表1-2所示。

表1-2 变量赋值过程

说明	X的值	程序
当程序开始时，没有任何变量存在。RAPTOR变量在某个语句中首次使用时被自动创建	未定义	
第一个赋值语句，X←32，分配数据值32给变量X	32	
第二个赋值语句，X←X+1，检索到当前X的值为32，给它加1，并把结果33赋予变量X	33	
第三个赋值语句，X←X*2，检索到X当前值为33，将它乘以2，并把结果66赋予变量X	66	

在表1-2中，程序在执行过程中，变量X存储过3个不同的值。请注意，在一个程序中语句的顺序是非常重要的，如果重新排列这3个赋值语句，存储在X中的值则会有所不同。

一个变量值的设置(或改变)可以采取以下3种方式之一：

- 通过输入语句赋值。
- 通过赋值语句中的公式运算后赋值。
- 通过调用过程的返回值赋值。

因此，变量数据值的变化导致程序每次执行的结果可以不同。

程序员应给予所有的变量以有意义的和具有描述性的名称。变量名应该与该变量在程序中的作用有关。变量名必须以字母开头，可以包含字母、数字和下划线(但不可以有空格或其他特殊字符)。如果一个变量名中包含多个单词，两个单词间用下划线字符分隔，这样变量名就更具有可读性。表 1-3 给出了一些好的、差的和非法的变量名的例子。

表 1-3 变量名实例

好的变量名	差的变量名		非法的变量名	
tax_rate	a	(没有描述)	4sale	(不能以数字开头)
sales_tax	milesperhour	(需添加下划线)	sales tax	(包含空格)
distance_in_miles	my4to	(没有描述)	sales $	(包含无效字符)

注意：如果给程序中的每个值一个有意义的、具有描述性的变量名，会帮助用户更清楚地思考需要解决的问题，并帮助寻找程序中的错误。在后续的算法设计中可以看到，良好的变量名称会大大改善算法的可读性，有利于用户进行算法的学习和交流；而不好的变量名称，即使是设计者自己，在回顾程序或算法时，也会感到困惑和茫然。

理解变量的方法之一，是将它们看成程序不同部分之间进行信息交流的一种手段。在程序的不同部分使用相同的变量名，实际上使用的是存储在同一内存位置上的数据。也就是把变量看作一个内存区域并在程序的计算过程中参与计算的数据。

RAPTOR 程序开始执行时，没有变量存在。当 RAPTOR 遇到一个新的变量名，它会自动创建一个新的内存位置并将该变量的名称与该位置相关联。在程序执行过程中，该变量将一直存在，直到程序终止。当一个新的变量创建时，其初始值将决定该变量的数据类型。变量自动创建时，RAPTOR 可以在其中保存的数据类型有以下 3 种。

(1) 数值(Number)：例如 12，567，−4，3.1415，0.000371。在 RAPTOR 中，数值的整数部分能够表达的有效位数约为 15 位十进制数；而小数部分的有效位默认为 4 位，需要提高小数精度时，可以使用 set_precision()函数进行设置。

(2) 字符串(String)：例如“Hello，how are you?”，“James Bond”，“The value of x is”。在 RAPTOR 的输入提示、处理语句和输出组合语句中，字符串必须使用双引号引起来，否则可能会被误认为是变量。这是由于变量实际上是由字符串来标识和引用的。

(3) 字符(Character)：例如'A'，'8'，'!'。字符变量的创建方法比较特殊，一般使用两种方法来创建，第一种是用 to_character(number)函数，其中 number 的范围是 0～127，也就是标准 ASCII 代码表的数值范围；第二种方法是从一个字符串中取出一个字符给变量赋值。特别提醒：字符串和字符是两种不同类型的变量，所以，不可以放在一起做比较；但可以放在＋运算符的两侧，把字符串与字符变量连接成字符串。

使用变量时常见的错误有以下两个：

(1) 变量没有找到(Variable not found)。

此错误的常见原因是变量无值或变量名拼写错误(见图 1-3)。

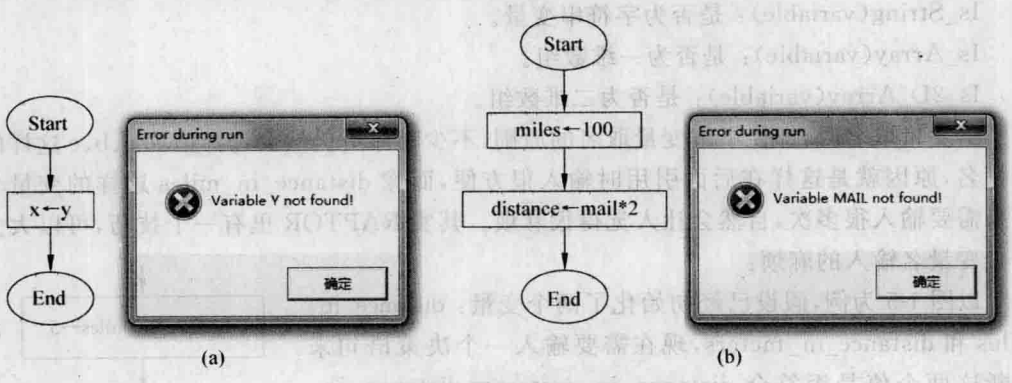


图 1-3 变量没有找到的报错信息

(2)“不能将字符串类型的值与字符类型的值进行比较(can't compare these values),将两个不同类型的数据放在一起比较,会发生这类错误(见图 1-4)。

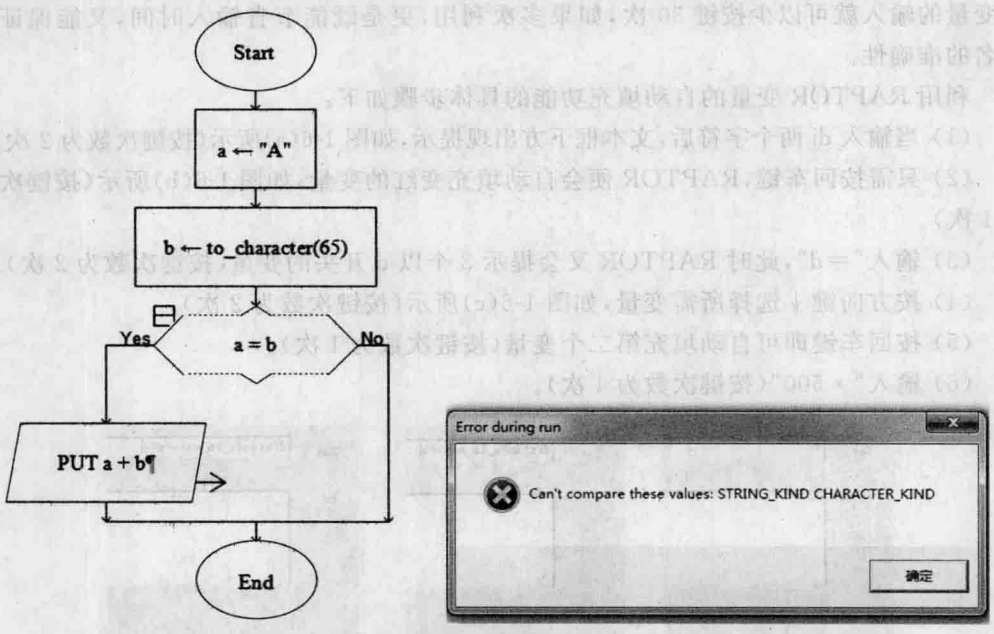


图 1-4 数据类型比较出错的提示

需要认真考虑 RAPTOR 变量名的另一个重要原因是,在程序运行过程中,变量可能会因为赋了不同类型的值而改变其类型。这在程序开发过程中有好处,也有害处。好处是,一个变量名在初始定义的作用完成后,改作他用,可以节省程序运行的空间;害处是,由于没有保持变量名和变量类型使用的一贯性,时间久了,开发者自己可能都搞不清楚程序中变量名所代表的意义和类型。为了避免出现上面提到的问题,RAPTOR 专门设置了若干函数在程序运行过程中测试变量的类型(所有返回值为布尔值)。

Is_Number(variable):是否为数值变量。

Is_Character(variable)：是否为字符变量。

Is_String(variable)：是否为字符串变量。

Is_Array(variable)：是否为一维数组。

Is_2D_Array(variable)：是否为二维数组。

给变量取名时，即使了解变量取名的原则，不少程序员仍然会选择诸如 a、b、c 这样的变量名，原因就是这样在后面引用时输入很方便，而像 distance_in_miles 这样的变量名如果需要输入很多次，自然会让人觉得很麻烦。其实 RAPTOR 里有一个技巧，可以大大减少变量名输入的麻烦：

以图 1-5 为例，假设已经初始化了两个变量：distance_in_miles 和 distance_in_meters，现在需要输入一个决策语句来判断这两个值是否符合 distance_in_meters = distance_in_miles * 500 这样一个关系，那么是否要在文本框里逐个字符地输入相关的变量名（按键数为 41 次）呢？实际上利用 RAPTOR 变量的自动填充功能，只需要按键 11 次即可。两个变量的输入就可以少按键 30 次，如果多次利用，更是既能节省输入时间，又能保证变量名的准确性。

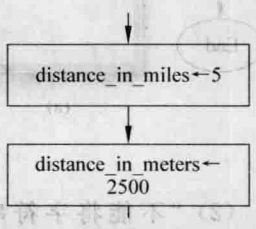


图 1-5　变量取名的小案例

利用 RAPTOR 变量的自动填充功能的具体步骤如下：

（1）当输入 di 两个字符后，文本框下方出现提示，如图 1-6(a)所示（按键次数为 2 次）。

（2）只需按回车键，RAPTOR 便会自动填充变红的变量，如图 1-6(b)所示（按键次数为 1 次）。

（3）输入"=d"，此时 RAPTOR 又会提示 3 个以 d 开头的变量（按键次数为 2 次）。

（4）按方向键↓选择所需变量，如图 1-6(c)所示（按键次数为 2 次）。

（5）按回车键即可自动填充第二个变量（按键次数为 1 次）。

（6）输入" * 500"（按键次数为 4 次）。

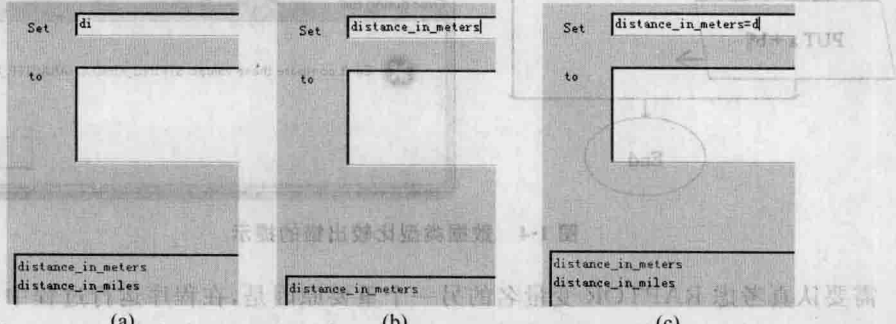


图 1-6　RAPTOR 变量自动填充功能的应用技巧

1.2.3　常量

所谓常量（constant），正好与变量相反，就是在程序运行过程中不改变其值的量。

RAPTOR目前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。当用户需要相应的值时，可使用代表这些常数的符号。

pi(圆周率)定义为3.1416(默认精度为4位小数，用户可用Set_Precision()过程扩展精度表达的范围)。

e(自然对数的底)定义为2.7183(精度设置同上)。

true/yes(布尔值真)定义为1。

false/no(布尔值假)定义为0。

以上6个符号也称为保留字，所谓保留字就是不可以再用作变量或者子图、子程序的名字的符号。

下面分别说明RAPTOR的输入(Input)、赋值(Assignment)、调用(Call)和输出(Output)这4个基本语句。

1.2.4 输入语句

输入语句允许用户在程序执行过程中输入变量的数据值。这里最为重要的是，必须让用户明白当前程序中需要什么类型的数据及其值的大小。因此，在定义一个输入语句时，一定要在提示文本框(Enter Prompt Here)中说明所需要的输入。提示应尽可能明确，如预期值所需要的单位或量纲(如英尺、米或英里)等(见图1-7)。

当定义一个输入语句时，用户必须指定两件事：一是提示文本(如上所述)，二是变量名称，该变量的值将在程序运行时由用户输入。正如可以在Enter Variable Here部分看到的那样(见图1-7)。

由图1-7所示的Enter Input对话框所产生的输入语句在运行时(run-time)将显示一个输入对话框，如图1-8所示。

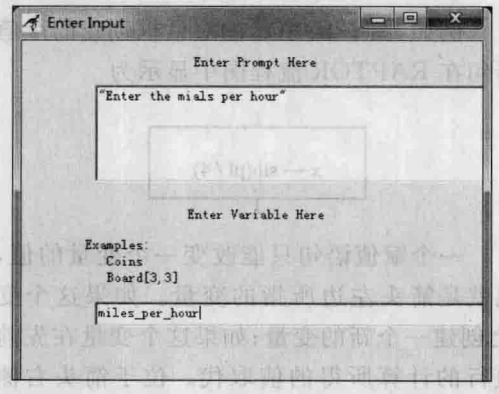


图1-7 输入语句的编辑(Edit)对话框

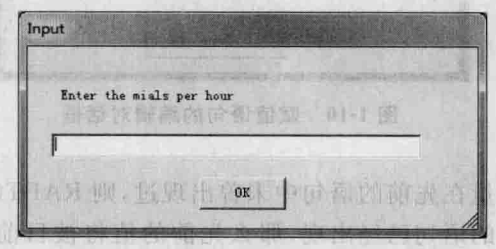


图1-8 输入语句在运行时的对话框

用户输入一个值，并按下回车键(或单击OK按钮)，用户输入的值由输入语句赋给变量。

请仔细思考“语句定义”(definition of a statement)和“语句执行”(execution of a statement)的区别。语句定义所使用的Enter Input对话框与执行程序时使用的对话框是完全不同的。

使用表达式提示(expression prompt)可以将文本与变量进行组合构成输入提示，如"Enter a number between " + low + " and " + high + ": "。当然，在使用上述的提示方法时，必须在程序中给 low 和 high 这两个变量赋值。

此外，已经编辑完毕的输入语句在流程图中的显示形式也会发生变化，如图 1-9 所示。注意，在图 1-9 中显示在输入语句符号中的 GET 字样，无须用户在编辑时输入，而是由系统自动给出的。

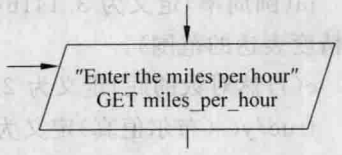


图 1-9 输入语句编辑完成后在流程图中显示的状态

1.2.5 数据处理语句

尽管输入语句可以给程序中的变量赋值，但 RAPTOR 还设计了专门的赋值语句用于对变量赋值。除了给变量赋值之外，赋值语句也是程序设计中数据处理的主要手段，数据处理则是通过表达式完成的。

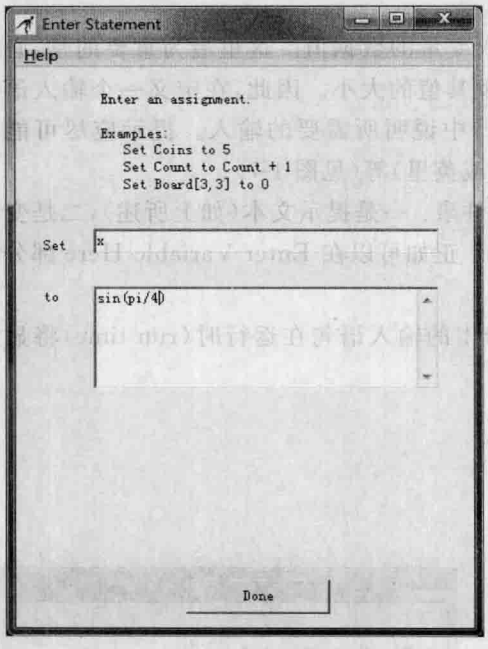


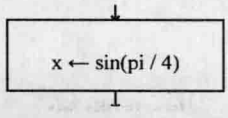
图 1-10 赋值语句的编辑对话框

赋值符号用于执行计算，然后将其结果存储在变量中。赋值语句的定义使用如图 1-10 所示的对话框。需要赋值的变量名须输入到 Set 文本框中，需要执行的计算输入到 to 文本框中。图 1-10 的示例是将变量 X 赋值为 0.7071。

RAPTOR 使用的赋值语句在其符号中的语法为

```
Variable ← Expression(变量←表达式)
```

例如，图 1-10 所示的对话框创建的赋值语句在 RAPTOR 流程图中显示为



一个赋值语句只能改变一个变量的值，也就是箭头左边所指的变量。如果这个变量在先前的语句中未曾出现过，则 RAPTOR 会创建一个新的变量；如果这个变量在先前的语句已经出现，那么先前的值将被目前所执行的计算所得的值取代。位于箭头右侧(即表达式)中的变量的值则不会被赋值语句改变。

一个赋值语句中的表达式(Expression)可以是任何计算单个值的简单公式或复杂公式。表达式是值(无论是常量或变量)和运算符的组合。请仔细研究构建有效表达式的规则。

由于计算机一次只能执行一项操作或运算。所以，当一个表达式进行计算时，并不是像用户输入时那样按从左到右的顺序进行。实际的运算执行顺序是按照预先定义的

“优先顺序”进行的。例如，考虑下面的两个例子：

x←(3+9)/3

x←3+(9/3)

在第一种情况中，变量 x 被赋值为 4；而在第二种情况下，变量 x 被赋的值为 6。从这两个例子中可以看出，使用括号可以任意地、明确地控制值和运算符的分组。

一般的优先顺序如下：

(1) 计算所有函数(function)的值。

(2) 计算括号中的表达式。

(3) 计算乘幂(^,**)。

(4) 从左到右计算乘法和除法。

(5) 从左到右计算加法和减法。

计算机根据运算符或函数对一些数据执行计算。运算符必须放在操作数据之间(如 X/3)，而函数使用括号来表示正在操作的数据(例如求平方根的函数 SQRT(4.7))。在执行时，运算符和函数按照优先顺序执行各自的计算，并返回其结果。表 1-4 概括了 RAPTOR 内置的运算符和函数。

表 1-4 RAPTOR 内置的运算符和函数

基本数学运算	+,−,*,/,^,**,rem,mod,sqrt,log,abs,ceiling,floor
三角函数	sin,cos,tan,cot,arcsin,arccos,arctan,arccot
杂项	random,Length_Of

表 1-5 简要介绍了 RAPTOR 内置的运算符和函数的基本用途和简单应用案例。关于这些运算符和函数更详细的说明可以查阅 RAPTOR 的帮助文档。

表 1-5 运算符和函数说明

运算符和函数	说 明	范 例
+	加	3+4 = 7
−	减	3−4 = −1
−	负号	−3
*	乘	3*4 = 12
/	除	3/4=0.75
^或**	幂运算	3^4 = 3**4 = 81
rem 或 mod	求余数	10 rem 3 = 1;10 mod 4 = 2
sqrt	求平方根	sqrt(4) = 2
log	自然对数(以 e 为底)	log(e) = 1
abs	绝对值	abs(−9) = 9
ceiling	向上取整	ceiling(3.14159) = 4