全国高等院校计算机基础教育研究会
“计算机系统能力培养教学研究与改革课题”立项项目
普通高等教育“十三五”规划教材

RAPTOR流程图+算法程序设计教程

冉娟　吴艳　张宁◎主编　张钢◎主审

案例教学 | 融合探究式的案例教学，兼顾基础和应用
内容丰富 | 实例、习题详尽，可读性强，实用性强
效果明显 | 重视实践，突出技能，提高程序设计能力

北京邮电大学出版社
www.buptpress.com

全国高等院校计算机基础教育研究会
“计算机系统能力培养教学研究与改革课题”立项项目
普通高等教育“十三五”规划教材

RAPTOR流程图+算法程序设计教程

冉 娟　吴 艳　张 宁◎主 编
张 钢◎主 审

北京邮电大学出版社
www.buptpress.com

内 容 简 介

《RAPTOR 流程图＋算法程序设计教程》是以培养学生计算思维能力为目标，从解决实际问题的角度出发，由案例引出知识点，强化程序设计求解问题的思路和方法，将真正程序设计中最基本的“思想”和“方法”挖掘出来，让学生充分体会到计算机求解问题的过程。

本书以 RAPTOR 作为程序设计工具，从大量实用性和趣味性实例入手，典型例题一题多解，由浅入深，系统介绍了利用 RAPTOR 进行程序设计的基本思想和方法，努力实现“零基础”入门。

本书适合作为高等学校非计算机专业和计算机专业程序设计入门的教材用书。

图书在版编目(CIP)数据

RAPTOR 流程图＋算法程序设计教程 / 冉娟，吴艳，张宁主编．-- 北京 ：北京邮电大学出版社，2016.8
ISBN 978-7-5635-4874-3

Ⅰ．①R… Ⅱ．①冉… ②吴… ③张… Ⅲ．①程序设计－高等学校－教材 Ⅳ．①TP311.1

中国版本图书馆 CIP 数据核字（2016）第 185034 号

书　　名：RAPTOR 流程图＋算法程序设计教程
著作责任者：冉娟　吴艳　张宁　主编
责 任 编 辑：王丹丹
出 版 发 行：北京邮电大学出版社
社　　址：北京市海淀区西土城路 10 号（邮编：100876）
发 行 部：电话：010-62282185　传真：010-62283578
E-mail：publish@bupt.edu.cn
经　　销：各地新华书店
印　　刷：北京通州皇家印刷厂
开　　本：787 mm×1 092 mm　1/16
印　　张：12.75
字　　数：316 千字
版　　次：2016 年 8 月第 1 版　2016 年 8 月第 1 次印刷

ISBN 978-7-5635-4874-3　　定　价：26.00 元

前　言

近些年各高校都将培养学生“计算思维”能力作为大学计算机基础教育教学改革的核心任务，这不仅是对计算思维能力本身的培养，更在于大学计算机基础教育要突破以往能力培养范畴，提升包括计算思维能力在内的普适性能力，从更高层面为专业服务。以计算思维为切入点，对大学计算机基础教育能力培养的深层启示究竟是什么？通过对这个问题的深入思考和多年教学实践，我们领悟到：计算机基础教育不应该仅仅是传统知识和技能的传授，而应以解决问题为目标，注重学生在学习过程中各种思维方式和行动能力的培养。而思维正是产生于各种实践应用中，随着实践而发展。计算机基础教育的课堂不仅是知识传递和实践锻炼，也是思维的训练。本书以此为出发点，通过简单的可视化程序设计工具 RAPTOR 进行问题描述，并能在计算机上执行这一过程“体会和实践”计算思维，不仅达到计算思维的培养，而且更加强化了学习程序设计的目的是学习计算机分析和解决问题的基本过程和思路。

本教材选取 RAPTOR 作为程序设计的工具，不仅是因为其简单，更主要的是对于不具备程序设计基础的新生而言，能够利用该工具代替静态的流程图和伪代码进行基础算法训练，将真正程序设计中最基本的“思想”和“方法”挖掘出来，让学生充分体会到计算机求解问题的过程，为今后进一步学习诸如高级语言程序设计等课程打下良好基础。

本教材中所有案例是作者从教学活动中积累的案例中选取的，不仅具有趣味性，而且这些案例一题多解，由浅入深，强化知识点、算法、求解问题的基本过程和思路；很多案例后面出现思考题、举一反三，不仅帮助读者消化理解这些案例，而且学会灵活应用。

本教材由冉娟、吴艳和张宁共同编写。其中冉娟负责全书架构设计及统稿，张宁编写第 3、9 章，吴艳编写第 1、2 章，冉娟编写第 1、4、5、6、7、8、10 章。

本书由天津大学张钢教授担任主审，在成稿过程中得到了张钢教授细心指点和帮助，并给予非常多的建议，在此对张钢教授表示衷心的感谢。

同时感谢王瑞航和纪文涛两位同学对部分算法实现案例所做的工作，也衷心的感谢北京邮电大学出版社对本书的出版过程中所给予大力支持和帮助。

由于时间仓促，本书在文字和案例难免有不完善之处，敬请广大读者谅解，并诚挚地欢迎读者提出宝贵建议。

作者

2016 年 6 月

目　录

第 1 章　程序设计与算法

本章学习目标：

通过本章学习，你将能够：

- ☑ 了解为什么要学习程序设计；
- ☑ 了解算法的概念和描述；
- ☑ 了解程序、程序设计以及程序设计语言的概念；
- ☑ 了解什么是 RAPTOR 以及它具有的特点。

1.1　为什么要学习程序设计

初学者看到“程序设计”，一定会想为什么要学习程序设计？程序设计对我们的生活、学习和工作会产生影响吗？

当我们在学校餐厅买饭，伴随着饭卡在刷卡机上轻轻扫过，饭卡上的余额马上减少了；当我们使用全自动洗衣机清洗衣物时，洗衣机会按照洗衣流程自动为我们把衣物清洗的崭新如初；当我们要去一个地方而不知如何乘坐车辆时，只需打开百度地图手机 APP 即可查询到公交车换乘方案。中国科技大学研制出的我国首台交互式机器人“佳佳”，谷歌的人工智能系统 AlphaGo 战胜了世界围棋冠军李世石，这些离不开程序设计。

作为一名优秀的技术工作者，不懂计算机程序设计，就不能真正理解计算机，也无法在自己所从事的工作领域内深入地应用计算机。

目前，虽然计算机应用软件及工具层出不穷，对于高等学校的学生来说，了解计算机科学，使计算机成为一种可以帮助人们思维的工具，显得尤为重要。而程序设计是实践计算思维的重要手段之一，在后续章节的学习，本书以 RAPTOR 为程序设计工具，围绕计算机问题求解开始程序设计学习。本章将简要介绍一些必要的概念和程序设计基础以引领读者进入程序设计世界。

1.2　认识算法

在学习 RAPTOR 程序设计之前，先来了解一下算法的基本概念。当我们遇到问题需要借助计算机去解决时，首先想到的是如何将一个想法或者问题的解决方案放到计算机上去实现，那么如何让计算机解决这些问题呢？这就需要了解和掌握计算机中的程序设计。而程序设计是利用计算机求解问题的一种方式，是程序员为解决特定问题而利用计算机语言编制相关软件的过程，是软件构造活动中的重要组成部分。程序设计的关键是解题的方法与步骤，即算法。

1.2.1 什么是算法

我们在日常生活中要做任何事情都要有一定的步骤。例如，刚刚考上大学的同学们一定都经历了这样的过程：填报高考志愿，交报名费，拿到准考证，参加高考，得到录取通知书，到学校报到注册。这些步骤都是按一定顺序进行的，缺一不可，次序还不能乱。实际上，日常生活中人们有意无意都在按一定有序步骤执行和实施，如烹饪美味佳肴。你会发现生活中描述这些步骤时可能会使用文字或者符号来表示。

再比如，有两瓶相同容量的水，一瓶是茶水，一瓶是矿泉水，现在却错把茶水装入了矿泉水瓶中，矿泉水错装入了茶水瓶中，要求将其互换。要解决这个问题，需要借助一个空瓶子，因此交换的步骤如下。

第一步：将茶水倒入空瓶子；

第二步：将矿泉水倒入茶水瓶；

第三步：将空瓶子中的茶水倒入矿泉水瓶；

第四步：交换结束。

因此，广义地说，为解决一个问题而采取的方法和步骤，就称为“算法”。当然，算法有执行主体，在计算机世界设计算法是让计算机可以执行，因此在此考虑的是计算机算法。

什么是算法？当代著名计算机科学家 D. E. Knuth 在他撰写的《The art of computer programming》一书中写道：“一个算法，就是一个有穷规则的集合，其中的规则规定了一个解决某一特定类型的问题的运算序列。”通俗地说，算法规定了任务执行/问题求解的一系列步骤。算法中的每一步必须是“明确的、可执行的”。下面我们通过例子来体会算法的含义。

【例 1-1】 求 1+2+3+…+10 的累加和。

方法一：

步骤 1，先求 1 与 2 的和，得到结果 3；

步骤 2，将步骤 1 得到的和与 3 相加，得到结果 6；

……

步骤 9，将步骤 8 得到的和与 10 相加，得到结果 55。

方法二：

步骤 1，分别求 1 与 10 的和、2 和 9 的和、3 与 8 的和、4 与 7 的和、5 与 6 的和；

步骤 2，求 5 个 11 的和，得到结果 55。

当然本题还有其他方法，一般来说，希望采用方法简单、运行步骤少的方法。如果要求求解 1 000 以内的累加和呢？显然这种方法较为烦琐，就需要找到一种通用的表示方法。因此，设计算法就是掌握分析问题、解决问题的方法，就是锻炼分析、分解，最终归纳整理出算法的能力。

1.2.2 算法的基本条件

为了能编写程序，必须学会设计算法，但不是任意写出的一些执行步骤就能构成一个有效算法，一个有效算法应该具备以下几个条件。

(1) 输入

在执行算法过程中，从外界获得的信息就是输入，一个算法可以有 0 个、1 个或多个输入。

(2) 输出

一个算法所得到的结果就是该算法的输出，其必须有 1 个或多个输出。

(3) 确定性

每条规则、每个操作步骤都应当是确定的，不允许存在多义性和模棱两可的解释。

(4) 有穷性

算法必须能在执行有限个步骤之后终止。

(5) 有效性

算法的每个操作步骤都应该是可执行的。

一个算法是由有限步骤组成的集合构成的，每一步都需要一条或多条操作。计算机能执行哪些操作，不可避免地对算法中可以包含哪类操作作出限制。因此，在设计一个算法时，必须要考虑它的可行性。

1.2.3　算法的描述工具

为了表示一个算法，可以用任何形式的语言和符号描述，通常有自然语言、伪代码、流程图、N-S 图和计算机语言描述等。

(1) 自然语言描述

用自然语言描述算法，就是用人们日常使用的语言描述或表示算法的方法。自然语言方法容易理解和掌握，但存在着很大的缺陷，就是容易出现二义性，所以一般用来粗略地描述算法思想。

例如，利用欧几里得算法求解两个正整数的最大公约数用自然语言描述如下。

Step 1：输入正整数 m、n；

Step 2：计算 $r=m \bmod n$；

Step 3：若 $r=0$，输出最大公约数 n，算法结束；若 $r\neq 0$，令 $m=n$，$n=r$，转 Step 2 继续。

由此可见，用自然语言描述算法通俗易懂，即使没有学过数学或算法，也能看懂算法的执行。

(2) 流程图描述

流程图是最早出现的用图形表示算法的工具，它利用几何图形的框代表各种不同性质的操作，用流程线指示算法的执行方向。例如图 1-1 是利用欧几里得算法求解两个正整数的最大公约数的流程图描述。

用流程图表示算法，直观形象、易于理解，能较清楚地显示出各个框之间的逻辑关系和执行流程，因此流程图成为程序员们交流的重要手段。

(3) N-S 图描述

1973 年美国学者 I. Nassi 和 B. Shneiderman 提出了一种新的流程图形式。在这种流程图中，完全去掉了带箭头的流程线。全部算法写在一个矩形框内，在该框内还可以包含其他从属于它的框，或者说，由一些基本的框组成一个大的框。这种流程图又称为 N-S 结构

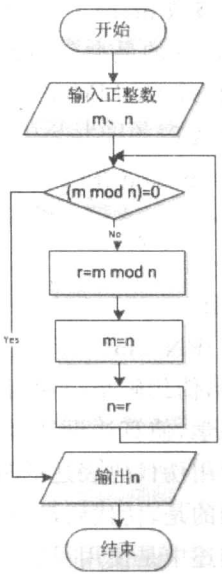

图 1-1　求解两个正整数的最大公约数的流程图

化流程图。例如图 1-2 是利用欧几里得算法求解两个正整数的最大公约数的 N-S 流程图描述。

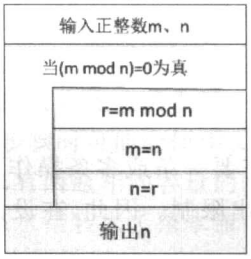

图 1-2　求解两个正整数的最大公约数的流程图

(4) 伪代码描述

算法最终是要用程序设计语言实现并在计算机上执行的。用自然语言描述算法虽然通俗易懂，但存在表达不严谨，容易出现二义性。用流程图描述算法虽然简单、直观，但缺少结构化的约束，不符合结构化程序设计的要求。用 N-S 图描述算法画图和修改都比较麻烦。而现用计算机程序设计语言多达几千种，不同的语言在设计思想、语法功能和适用范围等方面都有很大差异。此外，用程序设计语言表达算法往往需要考虑所用语言的具体细节，分散了算法设计者的注意力。因此，用某种特定的程序设计语言描述算法需要考虑语法细节，有些麻烦，伪代码描述正是在这种情况下产生的。

一般来说，伪代码是一种与程序设计语言相似但更简单易学的用于表达算法的语法。程序表达算法的目的是在计算机上执行，而伪代码表达算法的目的是给人看。伪代码应该易于阅读。简单和结构清晰，它是介于自然语言和程序设计语言之间的。伪代码不拘泥于程序设计语言的具体语法和实现细节。

由于伪代码在语法结构上的随意性，目前并不存在一个通用的伪代码语法标准。往往都是以某种具体的高级程序设计语言为基础，简化后进行伪代码的编写。最常见的这类高级程序设计语言包括 C、Basic、Java 和 ALGOL 等。由此而产生的伪代码往往被称为“类 C 语言”“类 ALGOL 语言”等。

例如，利用欧几里得算法求解两个正整数的最大公约数的伪代码描述。

```
Input:正整数 m、n
Output:m、n 的最大公约数
GREATEST-COMMON-DIVISOR(m、n)
1  REPEAT
2      r←m mod n
3      m←n
4      n←r
5  UNTIL  r=0
6  RETURN  m
```

由此可见，伪代码是一种用类似于程序的文本来表达算法的方式。它比一般的程序设计语言简单易学，使算法设计者可以把注意力集中在设计算法而不是具体程序设计语言的语法细节上。用伪代码表达的算法容易翻译成程序。因此，伪代码往往出现在程序的注释中。需要强调的是，伪代码没有统一的格式标准，只要能够简洁完整地表达算法就可以。伪代码在算法描述中是使用得非常多的一种工具。

(5) 计算机语言描述

计算机是无法识别流程图和伪代码的，只有用计算机语言编写的程序才能被计算机执

行。因此在用流程图或伪代码描述出一个算法后，还要将它转换成计算机语言程序。

例如，利用欧几里得算法求解两个正整数的最大公约数的C语言描述。

```
int MaxCommonFactor(int m,int n)  // MaxCommonFactor()函数，功能是计算两个正整数m、n的最大公约数，默认m>n
{
  int  r;
  do {
       r=m%n;
       m=n;
       n=r;
  } while(r)
  return  m;
  }
```

1.3 程序设计

1.2节讨论了什么是算法和算法的描述，为了让算法在计算机上执行，就需要用计算机语言来表示算法，这就要涉及计算机语言和程序设计。什么是程序？什么是程序设计？如何进行程序设计？这些都是初学者会遇到的问题，也是程序设计的基本问题。

1.3.1 程序

“程序”通常指完成某些事务的一种既定方式和过程，如学生每天开始的程序是起床、刷牙、洗脸、吃饭、上课等。

在计算机领域，程序是为实现特定目标或解决特定问题而用计算机语言编写的命令序列的集合，是人们求解问题的逻辑思维活动的代码化描述。程序表达了人的思想，体现了程序员要求计算机执行的操作。

对于计算机而言，程序是计算机的一组机器指令，它是程序设计的最终结果。程序经过编译和执行才能最终完成程序的功能。对于使用计算机的人而言，程序员用某种高级语言编写的语句序列也是程序。程序通常以文件的形成保存起来，所以源文件、源程序和源代码都是程序。

1.3.2 程序设计

什么是程序设计？对于初学者而言，往往把程序设计简单地理解为只是编写一个程序，这是不全面的。程序设计是指利用计算机解决问题的全过程，它包含多方面的内容，而编写程序只是程序设计的一部分。使用计算机解决实际问题，通常是先要对问题进行分析并建立数学模型，然后考虑数据的组织方式和算法，并用某种程序设计语言编写程序，最后调试程序，使之运行后能产生预期的结果，这一过程称为程序设计。程序设计的基本目标是实现算法和对初始数据进行处理，从而完成问题的求解。

学习程序设计的目的不只是学习一种特定的程序设计语言，而是要结合某种程序设计

语言学习程序设计的思想和方法。

程序设计的基本过程包括分析所求解的问题、抽象数学模型、设计合适的算法、编写程序，以及调试运行直至得到正确结果等几个阶段。具体的设计步骤如下。

(1) 分析问题，明确任务

接受某项任务后，首先需要对任务进行调查和分析，明确要实现的功能。然后详细地分析要处理的原始数据有哪些，从哪里来，是什么性质的数据，要进行怎样的加工处理，处理的结果送到哪里，如打印还是显示。

(2) 建立数学模型，选择合适的解决方案

对要解决的问题进行分析，找出他们的运算和变换规律，然后进行归纳，并用抽象的数学语言描述出来。也就是说，将具体问题抽象为数学模型。

(3) 确定数据结构和算法

方案确定后，要考虑程序中要处理的数据组织形式(即数据结构)，并针对选定的数据结构简略地描述用计算机解决问题的基本过程，再设计相应的算法。然后根据已确定的算法，画出流程图。这样能使程序思路更加清晰，减少编写程序的错误。

(4) 编写程序

编写程序就是将流程图或其他方法描述的算法用程序设计语言表达出来。这一步应注意的是：要选择一种合适的语言来适应实际算法和计算机环境，并要正确地使用语言，准确地描述算法。

(5) 调试程序

将源程序送入计算机，通过运行程序找出程序中存在的错误并修改，直到程序的运行效果达到预期的目标。

(6) 整理文档，交付使用

程序调试通过后，应将解决问题整个过程的有关文档进行整理，编写程序使用说明书。

以上是一个完整的程序设计的基本过程。对于初学者而言，因为要解决的问题比较简单，所以可以将上述前三步合并为一步，即分析问题和设计算法。

程序设计是一种高智力的活动，不同的人对同一件事物的处理可以设计出完全不同的程序。正因为如此，在计算机发展的早期，程序设计被认为是一个与个人经历、思想与技艺相关联的一种技艺与技巧，所以需要探索出各种方法与技巧。

1.3.3 程序设计语言

以上在介绍程序和程序设计中都提到，程序是用某种语言来描述的，程序设计也是要用到某种语言来设计程序，我们不妨说程序设计语言是人与计算机进行交流的工具。如同想与外国人交谈就必须学会外语一样，要想比较深入地掌握计算机，就必须学习有关程序设计语言的知识。各类计算机程序设计语言具有一定的共性，掌握这些就可以触类旁通。

程序设计语言的发展，经历了机器语言、汇编语言和高级语言等几个阶段。其中机器语言是指计算机能够直接识别的基本指令的集合，是最早出现的计算机语言。汇编语言是用一些容易记忆和辨别的有意义的符号代替机器指令所产生的语言，它比机器语言程序容易阅读和修改，但它仍然与机器指令相对应。高级语言是一种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，消除了机器语言的缺点，使得普通用户容易学习

和记忆，因此简单易学。目前，高级语言从 1954 年第一个完全脱离机器硬件的高级语言 FORTRAN 问世，到现在已经有几百种，如 C、C++、Python、Java、HTML 等。但对于初学者而言，过于复杂的高级语言不适用于初学者，那么有没有适合初学者进行思维表达而又无须太多难度的入门语言工具呢？回答是肯定。RAPTOR 就是这样一种简单、易用的程序设计工具，它是用流程图的方式解决问题，我们只需要把精力放在如何表达自己的想法上就可以，而无须了解过多的程序设计技巧和工具本身的技巧，具体的内容将在后续章节进行介绍。

1.4　RAPTOR 简介

1.4.1　什么是 RAPTOR

RAPTOR(Rapid Algorithmic Prototyping Tool for Ordered Reasoning)，是一种基于有序推理的快速算法原型设计工具。它是由各种相互连接的图形符号构成的可执行流程图，为程序设计和算法设计的基础课程的教学提供实验环境。使用 RAPTOR 设计的程序和算法可以直接转换成为 C++、C# 和 Java 等高级程序设计语言，这就为程序和算法的初学者铺就了一条平缓、自然的学习阶梯。

1.4.2　为什么使用 RAPTOR

沙克尔福德和勒布朗(Shackelford and LeBlanc)曾经观察到，在计算机导论课程中使用特定的编程语言容易导致学生“受到干扰并从算法问题求解的核心上分散注意力”。由于教师希望学生在上课时间内能够解决实际的问题，因此，教师往往把授课的重点集中在程序语言的语法上，这就使得学生从学习算法问题求解的核心上转移到学习语言的语法上，失去学习程序设计的真正目的：为了解决各种实际问题，能够将实际问题以抽象化和程序化的形式表示出来。

RAPTOR 专门用于解决非可视化环境的语法困难和缺点，其目标是通过缩短现实世界中行动与程序设计的概念之间的距离来减少学习上的认知负担。使用 RAPTOR 进行程序设计基于以下几个原因：

(1) 由各种相互连接的图形符号构成的可执行流程图，最大程度地减少了程序语言的语法理解；

(2) 操作简单，学生只需要通过拖拽操作就可将不同图形符号放置到所需要的位置上，工具软件就可以自动将这些不同图形符号连接在一起，形成一个完整的流程图；

(3) 简单易懂，由于流程图与自然的思维过程相近，能够比较简单地让学生掌握和理解程序的设计与算法。

RAPTOR 除了具有流程图特色外，还具有其他诸多重要特点，例如，计算操作的原子化和算法的执行步骤统计等，为算法设计、算法优化、算法复杂性分析提供了有力的实验或验证手段。

目前，RAPTOR 作为一种可视化程序设计软件工具，从 2006 年开始已经在卡内基·梅隆大学、美国空军学院等 20 多个国家和地区的高等院校使用，在计算机基础课程教学中取

得良好的效果。现在国内一些高校也陆续开始将 RAPTOR 作为程序设计入门课程，让学生从简单易懂的程序流程图入手，允许流程图在其环境下直接调试和运行算法，并且可以直观看到当前执行符号所在的位置及其所有变量的变化过程。不仅如此，RAPTOR 既避免重量级高级语言（如 C、C++）所带来的烦琐语法规则，又提供了更直观的图形化界面，使得刚刚入学的大学生都能在很短的时间内学会编写简洁程序并能够正常运行，更能激发学生学习程序的热情。请先看一个简单的 RAPTOR 程序。

【例 1-2】 利用 RAPTOR 编写程序，实现输出“Welcome RAPTOR world!”，如图 1-3 所示。

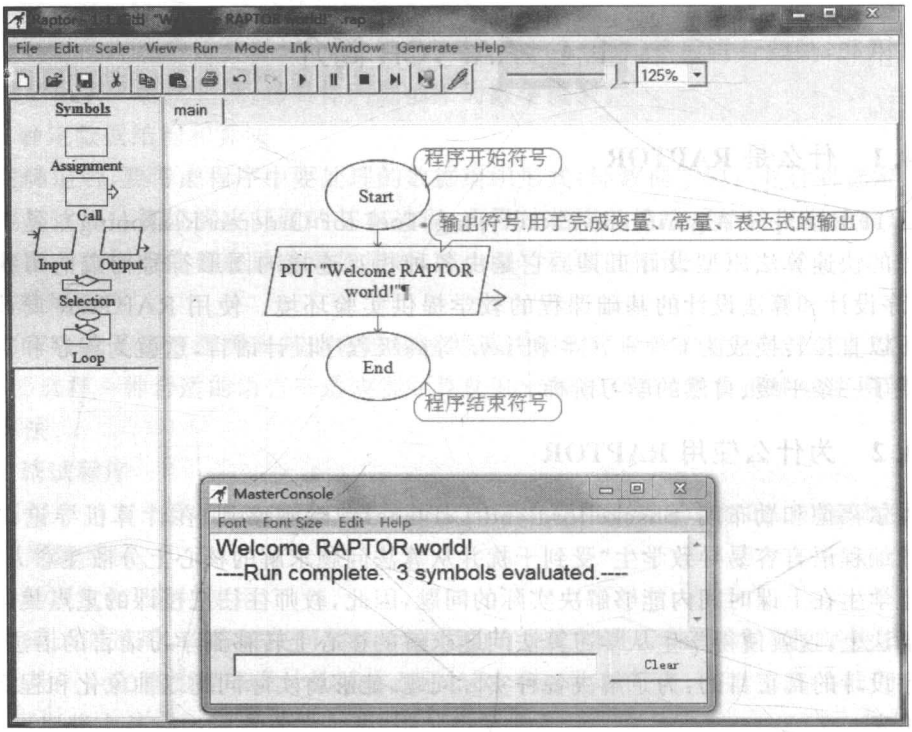

图 1-3　简单 RAPTOR 程序设计及运行结果

相信各位读者通过本例题一定感觉在没有任何程序设计基础的前提下也可以使用 RAPTOR 可视化程序设计来编写程序，程序简单、直观，能够按照自己想法来实现题目的功能，而且整个程序设计过程是在图形界面中完成，避免了可视化程序设计语言的复杂、枯燥的语法。

这对于学生而言，不仅让学生体会了实现算法的可视化，而且体会了学习程序设计的“思想”和“方法”。因此，与其他可视化程序设计语言环境相比，RAPTOR 更能够让学生创造出更有趣的算法。

1.4.3　RAPTOR 的特点

（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算法设计的学习阶段；

（2）具备基本运算功能，有 18 种运算符，可以实现大部分基本运算；

（3）具备基本数据类型与结构，提供了数值、字符串和字符 3 种数据类型以及一维数

组、二维数组等数据组织形式，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；

(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++、C#等；

(7) 程序的设计结果可以直接编译成为可执行文件并运行；

(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。

本章小结

本章内容主要涉及程序设计的一般性概念，包括程序、程序设计、程序设计语言以及算法等。通过对这些问题的介绍，为今后更好学习程序设计打下基础。由于 RAPTOR 是一种基本功能完备而又十分简洁的算法描述性程序设计环境，对于程序设计入门学习极为有利。

另外，本书介绍程序设计概念的目的并不是为了介绍某种特殊的程序设计语言，而是为了描述算法，无论今后读者学习任何一种语言，程序设计的基本概念和算法都是相通的。因此，在本书学习过程中，读者要细心体会对问题求解的过程。

习　题

1. 什么是程序、程序设计和程序设计语言？

2. 什么是算法？算法描述的方法有几种，分别是什么？

3. 用传统流程图方式描述下列题目的算法。

(1) 从键盘输入圆半径，计算圆的面积和周长。

(2) 求三个正整数 a、b、c 的最大值。

(3) 早上起床到准备上课的流程。

(4) 出外旅游的准备工作的流程。

(5) 古堡算式问题：

福尔摩斯到某古堡探险，看到门上写着一个奇怪的算式：ABC * ?=CBA

福尔摩斯对华生说："ABC 应该代表不同的数字，问号也代表某个数字！"华生说："我猜也是！"。但是两人沉默了好久，还是没有算出合适的结果来。请你帮助福尔摩斯和华生找出 ABC 这个数。