IS12 - Introduction to Programming

Lecture 1: Introduction

Peter Brusilovsky
http://www2.sis.pitt.edu/~peterb/0012-072/

January 8, 2007

Introduction (outline)

- Introduction to course goals and content
- Web site
  - Books
  - Tools
  - Syllabus
  - Materials
- What is programming?
- Introduction to Karel the Robot
IS12 and BSIS

- Information Science and programming
- So, do we need to learn a programming language?
  - To understand it, you have to do it!
  - No magic in programming!
- BSIS: need 2 programming courses
  - (IS12) ⇒ IS17 ⇒ (IS19)
- DLIM: Intro Programming requirement

What is special for IS12?

- Most courses offer a steep introduction
- Gentle introduction for the beginners
- Prepares for IS17 and IS19
- Languages:
  - Karel the Robot (1/4 of the course)
  - Introduction to C (3/4 of the course)
- Is it the right course for you?
  - You can go directly to IS17
Learning programming

- Do not procrastinate!
  - Get books, check/use Web tools, install and get yourself familiar with programming environments

- Practice, practice, practice!
  - Run all examples, modify it, explore
  - Check yourself on quizzes
  - Solve problems and exercises

- Get help!
  - Ask questions in CourseWeb forums
  - Meet your instructor

Course Tools

- Course Web site
  [http://www2.sis.pitt.edu/~peterb/0012-072/](http://www2.sis.pitt.edu/~peterb/0012-072/)

- The complete list of tools is provided on Tools section of this site

- Blackboard (CourseWeb) system will be used as the main learning support tool

- Karel the Robot environment will be used for programming

- Other tools will be introduced later
Books

- Books for Karel the Robot
  - Pattis
  - Online tutorial
- Books for C
  - Perry: Absolute Beginner’s Guide to C
  - Others
    - Kernighan and Ritchie
    - Deitel and Deitel
  - Multiple free tutorials on the Web. You will be able to access them via Knowledge Sea system

Blackboard (CourseWeb)

- Blackboard system will be used for:
  - Posting announcements (WATCH IT!)
  - Posting course materials and assignments
  - Learning about and communicating with each other
  - Asking questions and getting answers
  - Submitting assignments
  - Posting grades
Read Syllabus Carefully (I)

- **Final Grade**
  
  \[
  \text{Final Grade} = \frac{\text{attendance} + \text{hw_points} + \text{quiz_points} + \text{extra_credit_points} + \text{exam_points}}{\text{max_attendance_points} + \text{max_homework_points} + \text{max_quiz_points} + \text{max_exam_points}} \times 100\%
  \]
  
  - <50% corresponds to F, 50-62.5 is D range, 62.5-75 is C range, 75-87.5 is B range, and 87.5-100 is A range.

- **Homework and Late submissions**
  
  Due date: after lecture - for paper version, 11:59 pm for electronic version.
  
  Within 3 days after due date: 80%, before next class: 50%.

- **Quizzes**
  
  One lowest score will be dropped

Read Syllabus Carefully (II)

- **Attendance**
  
  Each lecture worth 1 point, usually 2 lectures per class, up to max_attendance_points=20

- **Extra credit**
  
  - Be active in forums, answer questions, report errors and problems
  
  - Take part in extra credit studies

- **Integrity**
Communication

■ To you
  – Watch closely the CourseWeb site for announcements.
  – Check your Pitt mail (xyz@pitt.edu) connected to CourseWeb regularly - most important and urgent information will be distributed by e-mail

■ From you
  – If a question is not personal (an answer could be useful for others) - *ask via forum*
  – If it is a personal question - ask me via e-mail

■ Office Hours

CourseWeb Assignment (HW0)

■ Due Friday 1/12/2007
■ Try visible features, ask questions, answer questions
■ Home page (*picture!*) (2pts)
■ Complete a Pre-test - results are not counted towards your grade (1pt)
■ Search the Web, find a programming course that uses Karel or a similar language, post URL and a message to the test forum (1pt)
What are computers (robots)?

- “idiot servants” that can do simple operations incredibly fast if you tell them *every step* to do
- like little children in their need for specific and detailed *instruction*
- computers are not “brains” & are not “smart” - they only as good as the *program* they are running

How to give commands?

- **Dialog mode:**
  - Give a command
  - Observe results
  - Give another command
  - Observe results … …
- **Programming:**
  - Give a set of commands in advance
  - Observe final results

*Adapted from J. Wyatt’s slides*
Programs and programming

- **What is a program?**
  - A set of *instructions* given to a computer to work with *objects* (*world, data*) in order to accomplish a specific task

- **What is programming?**
  - The art to control these “robots”, “servants”, “little children” by writing sets of instructions in advance
  - The art and craft of writing *programs*

Karel the Robot

- Invented by Richard Pattis in 1981
- A Gentle Introduction into Programming
- Used in top universities and colleges
- Learning to program by learning to control Robot Karel acting in its World
  - Learn basic principles of programming
  - Learn main programming constructs (same in Karel, C, Java, Pascal, Basic, etc)
Karel: Simple Programs (Outline)

- The Karel programming environment
- Creating worlds
- Writing programs
- Karel built-in commands
- Karel program syntax
- Programming errors
- Edit-Compile-Run-Test loop
- Defining new commands for Karel
- Naming Karel commands

Karel’s world

- Horizontal Streets
- Vertical Avenues
- Corners (Intersections), origin
- Beepers situated at corners, beeper bag
- Walls separating corners
- Robot Karel
  - may stand in any corner
  - can face North, South, West or East
  - can have beepers in his beeper bag
Creating a New World

- Use tab *World* in the environment
- Push *New* to create a new empty world
- Move *cursor* and use world editing tools
  - To place walls
  - To place beepers
  - To position Karel
- Save the world to a file for the future re-use (use .kw extension)

Get out of Jail Problem

- **Task:** Get Karel out of Jail!

Start, or initial situation:

Target, or final situation:
Commanding Karel

- What *commands* we will give Karel? => What commands would it understand?
- Instructions/commands we’ll use to solve this problem:
  - move
  - turnleft
- Let’s get Karel out of jail using these commands

Our First Program

- Go to “Program” tab
  - Create new program
  - Write/Edit program
  - Compile program
- Go to “Execute”
  - *Initialize* execution
  - *Run* program
- The robot will execute the current program in the current world
Get out of Jail Program

beginning-of-program
beginning-of-execution
move;
move;
turnleft;
move;
turnoff;
end-of-execution
end-of-program

IS12 - Introduction to Programming
Lecture 2: Simple Programs

Peter Brusilovsky
http://www2.sis.pitt.edu/~peterb/0012-072/

January 8, 2007
Karel Program Syntax

- Karel programs have the following structure
  
  beginning-of-program
  
  beginning-of-execution
  
  <commands>
  
  turnoff
  
  end-of-execution
  
  end-of-program

- Where <commands> is a sequence of Karel commands separated by semicolons ;
- Note that it is a bit different from C language: in C a semicolon ends a command
- "One command in each line", as well as indentation, is a good style, not a syntax rule!

The Full Set of Karel Commands

- move - move one corner in the current direction
- turnleft - turn left, change direction
- pickbeeper - pick 1 beeper from the current corner, put into the beeper bag
- putbeeper - place 1 beeper from the beeper bag on the current corner
- turnoff - turns itself off
Lexical and Syntactic Errors

- Exact spelling and strict rules of syntax:
  ```
  beginning-of-program
  beginning-of-execution
  move;
  move;
  turnleft
  move;
  turnoff
  end-of-execution
  end-of-program
  ```

- No execution for programs with lexical or syntax errors.

Semantic Errors

- Where is the error?
  ```
  beginning-of-program
  beginning-of-execution
  move;
  move;
  turnoff;
  move;
  turnleft
  end-of-execution
  end-of-program
  ```

- Semantic error: Possible misunderstanding how to use a command or a construction
Execution errors

- Situation: Karel at (1,1), facing North
  beginning-of-program
  beginning-of-execution
    turnleft;
    move;
    move;
    turnoff
  end-of-execution
  end-of-program
- Execution causes error shutoff

Foolproof Karel: Error shutoff

- Can your errors hurt Karel?
- move - shutoff if facing a wall
- pickbeeper - shutoff if no beepers on the corner
- putbeeper - shutoff if no beepers in the beeper bag
- turnleft and turnoff - always possible
Intent Errors (bugs)

- If there are no syntax errors, does it mean that the program is correct?

```
beginning-of-program
beginning-of-execution
move;
move;
move;
turnleft;
turnoff
end-of-execution
end-of-program
```

Intent Errors (bugs)

- It depends on what it suppose to do. What's the task?

```
beginning-of-program
beginning-of-execution
move;
move;
move;
turnleft;
turnoff
end-of-execution
end-of-program
```

- Let's define the task: ready to go north
- And another task: knight's move
The edit-compile-run loop

1. Edit program
2. Compile program
3. If there are errors, fix and go back to 1
   • you have got syntax error
   • Think how to fix it and go back to 1
4. Run it
5. If it produce wrong results
   • watch or simulate execution
   • find the source of the error (debug)
   • think how to fix it and go back to 1

The iterative nature of programming

The “programming in small” loop
Problem: Move beeper

- Move a beeper from 1:4 to 3:5

Example: Move beeper

beginning-of-program
  beginning-of-execution
    move;
    move;
    pickbeeper;
    move;
    turnleft;
    move;
    move;
    putbeeper;
    move;
    turnoff
  end-of-execution
end-of-program
Defining New Instructions

- How to extend Karel's set of instructions?

  ```define-new-instruction <name> as <instruction>;```

- Example:
  ```define-new-instruction go as move;```

- To be placed between beginning-of-program and beginning-of-execution

Why? Case 1: Square Dance

```
beginning-of-program
beginning-of-execution
    move;
    turnleft;
    move;
    turnleft;
    move;
    turnleft;
    move;
    turnleft;
    turnoff;
end-of-execution
end-of-program
```

```
beginning-of-program
beginning-of-execution
    move;
    turnleft;
    turnleft;
    move;
    turnleft;
    turnleft;
    move;
    turnleft;
    turnleft;
    move;
    turnleft;
    turnleft;
    turnleft;
    move;
    turnleft;
    turnleft;
    move;
    turnleft;
    turnleft;
    turnleft;
    move;
    turnleft;
    turnleft;
    turnoff;
end-of-execution
end-of-program
```
Block

- A syntactically correct way to make a sequence of instruction looking as one instruction. A block can be used whenever single instruction can be used

begin
    <instruction>;
    <instruction>;
    ...
    <instruction>
end

New Instruction with the Block

- Blocks can be used to define new instructions from several elementary ones

define-new-instruction <name> as begin
    <instruction>;
    <instruction>;
    ...
    <instruction>
end;
Solution 1: The Missing turnright

- Now we can define turnright

```plaintext
define-new-instruction turnright as begin
    turnleft;
    turnleft;
    turnleft;
end;
```

Square Dancing Clockwise

The place for defining new instructions is between beginning-of-program and beginning-of-execution

```plaintext
beginning-of-program
    define-new-instruction turnright as begin
        turnleft;
        turnleft;
        turnleft;
    end;
beginning-of-execution
    move;
    turnright;
    move;
    turnright;
end-of-execution
end-of-program
```

• Another design with defined instruction “step”
The Flow of Execution: The Glossary Model

- When Karel encounters the new name in the process of program execution, it looks for its "definition" in the glossary of commands.
- If the definition of the new command is found, Karel executes the body of the command definition.
- After that, Karel returns to the next instruction.

Name does not matter (for execution)

- Names are just names. What the new command will do is defined by its body, not by its name.

```plaintext
define-new-instruction turnright as begin
    move;
    move;
    move;
    move
end;
```
Name does matter (for understanding)

- From syntactic prospect, name could be any combination of letters, numbers and hyphens that starts with a letter
- From the understanding prospect, the name should express the function of the new command

```c
define-new-instruction i543 as begin
  turnleft;
  turnleft;
  turnleft
end;
```

Before Next Meeting

- Explore Web site, read syllabus. Decide if this course is for you.
- Get / check the books
- Install / try Karel Environment
- Reading assignment:
  - Pattis: Chapter 1, Chapter 2, Chapter 3 (3.1-3.7)
  - Tutorial on Karel Environment
- Follow Chapter 2 by writing and running code. Check yourself by doing exercises from Chapter 2
- Homework-1 (5 points) due 1/22/07