IS12 - Introduction to Programming

Lecture 3: Program Design

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

Overview

- Why else do we need new commands
  - Case 2: Up the Stairs
  - Case 3: Sweep the Stairs
- Program design
  - Top-down and design tree approaches
- Exercises in modifying a well-designed program
Why? Case 2: Up the Stairs

- Move Karel up the stairs

Start:

Target:

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

end-of-execution
end-of-program
Solution 2: Up the Stairs

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

Square Dance Again

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

end;
```

What is the point?
Why? Case 3: Sweep the Stairs

- Move Karel up the stairs picking beepers

  Start:

  Target:

Solution 3: Sweep the Stairs

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

```plaintext
beginning-of-execution
climb-stair;
pickbeeper;
climb-stair;
pickbeeper;
climb-stair;
pickbeeper;
turnoff;
end-of-execution
end-of-program
```

What is the point?
Why do we need new instructions?

- Defining clearly missing commands
  - turnright
- Automating repeating fragments
  - climb-stairs
- Creating useful new instructions that can be re-used in several contexts
  - climb-stairs

Program Design

- Overall goals:
  - our programs must be easy to read and understand
  - our programs must be easy to debug
  - our programs must be easy to modify to solve variations of the original task
- The approach:
  - Programming as problem solving
Polya describes problem solving as a process with four activities:
- definition of the problem
- planning the solution
- implementing the plan
- analyzing the solution

Implementation is just one of four! Planning is the key.

Case 1: The Harvest Task

- Karel has to pick up a field of beepers
- We will use a top-down approach known as stepwise refinement
- Decompose problem into sub-problems
- Write the top-level program using names of new instructions
- Define them later
First Trial with Harvesting a Row

```
beginning-of-execution
move;
harvest-1-row;
return-to-start;
harvest-1-row;
return-to-start;
move-north-1-block;
harvest-1-row;
return-to-start;
harvest-1-row;
return-to-start;
harvest-1-row;
return-to-start;
move-north-1-block;
harvest-1-row;
return-to-start;
harvest-1-row;
return-to-start;
move-north-1-block;
harvest-1-row;
return-to-start;
harvest-1-row;
return-to-start;
move-north-1-block;
harvest-1-row;
return-to-start;
turnoff;
end-of-execution
```

Second Trial: Harvesting 2 Rows

```
Main program:
beginning-of-execution
move;
harvest-2-rows;
position-for-next;
harvest-2-rows;
position-for-next;
harvest-2-rows;
move;
turnoff;
end-of-execution
```

```
Possible implementation of harvest-2-rows
define-new-instruction
harvest-2-rows as
begin
harvest-1-row-moving-east;
go-north-to-next-row;
harvest-1-row-moving-west;
end;
```
Further Refinement: Step 2

harvest-2-rows:

define-new-instruction
harvest-2-rows as
begin
harvest-1-row;
go-to-next-row;
harvest-1-row;
end;

position-for-next:

define-new-instruction
position-for-next as
begin
    turnright;
    move;
    turnright;
end;

Further Refinement: Step 3

harvest-1-row:

define-new-instruction
harvest-1-row as
begin
    pickbeeper; move;
pickbeeper; move;
pickbeeper; move;
pickbeeper; move;
pickbeeper; move;
pickbeeper;
end;

go-to-next-row:

define-new-instruction
go-to-next-row as
begin
    turnleft;
    move;
    turnleft;
end;
Solution for Harvest Problem

beginning-of-program
  define-new-instruction turnright as begin
    turnleft;
    turnleft;
    turnleft;
  end;
  define-new-instruction go-to-next-row as begin
    turnleft;
    move;
    turnleft;
  end;
  define-new-instruction position-for-next as begin
    turnright;
    move;
    turnright;
  end;
  define-new-instruction harvest-1-row as begin
    pickbeeper; move;
    pickbeeper; move;
    pickbeeper; move;
    pickbeeper; move;
    pickbeeper;
  end;
  define-new-instruction harvest-2-rows as begin
    harvest-1-row;
    go-to-next-row;
    harvest-1-row;
  end;
  beginning-of-execution
    move;
    harvest-2-rows;
    position-for-next;
    harvest-2-rows;
    position-for-next;
    harvest-2-rows;
    move;
    turnoff;
end-of-execution
end-of-program

Stepwise refinement tree for Harvest

Harvest problem
  └ harvest-2-rows
    └ harvest-1-row
    └ go-to-next-row
  └ position-for-next
  └ turnright
Stepwise Refinement vs. Design Tree Approaches

- **Stepwise refinement**
  - Breadth first approach
  - Design program down to code
  - Debug components
  - Debug whole

- **Design tree**
  - Depth first approach
  - Design top level program
  - Get the first slice down to code
  - Debug the slice ...

Why do we need new instructions?

- **Make the program readable and understandable**
  - Compare with section 3.9.3 of Pattis
  - Chunking and naming!

- **Make the programs easy to debug**
  - Planning vs. implementation errors

- **Make the programs easy to modify to solve variations of the original task**
  - Modified Harvest problems
Modification 1: Longer Rows

- Where the changes are localized?

Modification 2: More Rows

- Where the changes are localized?
Modification 3: Now what?

- Can we solve this problem by modifying the original harvest program?
- Complete exercise 3.11-5 at home

Before next class (Karel part):

- Reading assignment: Pattis, Chapter 3
- Run Classroom Examples
- Check yourself by doing exercises 1, 2, and 9 from Section 3.11. Practice top-down design approach.
- Attempt to solve exercise 5 with minimal changes to the harvesting program
- Homework 2 (due 1/29/07)
  - Solve the specified problem using at least two new instructions. Use top-down design!