The iterate instruction

- How to repeat an action known number of times?

```
iterate <positive-integer> times <instruction>;
```

- Example:
```
iterate 5 times
move;
```

- Note indentation!
iterate instruction with a block

iterate <positive-integer> times begin
  <instruction-1>;
  <instruction-2>;
  ...
  <instruction-k>;
end;
<next-instruction>;

Semantics of execution
– A sequence of instructions from instruction-1 to instruction-k will be executed positive-integer times. After that - next-instruction

Example 1: Square Dance

New way

beginning-of-program
beginning-of-execution
iterate 4 times begin
  move;
  turnleft;
end;
turnoff;
end-of-execution
end-of-program

Old way

beginning-of-program
beginning-of-execution
move;
turnleft;
move;
turnleft;
move;
turnleft;
move;
turnleft;
turnoff;
end-of-execution
end-of-program
Problem 3.10: Nested Loops

Explicit  Implicit

beginning-of-program
beginning-of-execution
iterate 4 times begin
iterate 3 times begin
putbeeper;
move;
end;
turnleft;
end;
turnoff;
end-of-execution
end-of-program

beginning-of-program
beginning-of-execution
iterate 3 times begin
putbeeper;
move;
end;
beginning-of-execution
iterate 4 times begin
plant-4;
turnleft;
end;
turnoff;
end-of-execution
end-of-program

Old way: Cleaner Stairs

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

beginning-of-execution
define-new-instruction pickbeeper-if-present as
if next-to-a-beeper then
pickbeeper;
climb-stair;
pickbeeper-if-present;
climb-stair;
pickbeeper-if-present;
climb-stair;
pickbeeper-if-present;
turnoff;
end-of-execution
end-of-program

define-new-instruction
pickbeeper-if-present as
if next-to-a-beeper then
pickbeeper;
beginning-of-execution
climb-stair;
pickbeeper-if-present;
climb-stair;
pickbeeper-if-present;
climb-stair;
pickbeeper-if-present;
turnoff;
end-of-execution
end-of-program
New Way: Cleaner Stairs 2

Is iterate always good?

beginning-of-program
define-new-instruction
turnright as
   iterate 3 times
      turnleft;

define-new-instruction climb-stair as begin
   turnleft;
   move;
   turnright;
   move;
end;

define-new-instruction
   pickbeeper-if-present as
   if next-to-a-beeper then
      pickbeeper;

beginning-of-execution
   iterate 3 times begin
      climb-stair;
      pickbeeper-if-present;
   end;
   turnoff;
end-of-execution
end-of-program

Old way: Carpet (problem 3.9)

beginning-of-program
define-new-instruction
   laycarpet as begin
      move;
      putbeeper;
      move;
      putbeeper;
      move;
      putbeeper;
      move;
      putbeeper;
      move;
      putbeeper;
      move;
      putbeeper;
      move;
      putbeeper;
      move;
      putbeeper;
      move;
   end;

beginning-of-execution
   laycarpet;
   turnleft;
   laycarpet;
   turnleft;
   laycarpet;
   turnleft;
   laycarpet;
   turnleft;
   laycarpet;
   turnleft;
   laycarpet;
   turnleft;
   laycarpet;
   turnleft;
   laycarpet;
   turnleft;
   laycarpet;
   turnleft;
   laycarpet;
   turnleft;
   laycarpet;
   turnleft;
   laycarpet;
   turnleft;
end-of-execution
end-of-program
New way: Carpet (problem 3.9)

beginning-of-program
  define-new-instruction
  laycarpet as
  iterate 7 times begin
    move;
    putbeeper;
  end;
  beginning-of-execution
  iterate 4 times begin
    laycarpet;
    turnleft;
  end;
  endoff;
end-of-execution
end-of-program

while loop

while <condition> do
  <instruction>;
<next-instruction>;

Semantics of execution
  - While condition is true - instruction is executed over and over.
  - After that - next-instruction
  - What if it is wrong right away?
Flowchart of while

while instruction with a block

while <condition> do begin
  <instruction-1>;
  <instruction-2>;
  ...
  <instruction-k>;
end;
<next-instruction>;

- Semantics of execution
  - While condition is true - instruction-1 ... instruction-k repeated over and over
  - after that - next-instruction
Examples

- **Find beeper**
  define-new-instruction go-to-beeper as
  while not-next-to-a-beeper do
    move;

- **Get all beepers**
  define-new-instruction clear-corner-of-beepers as
  while next-to-a-beeper do
    pickbeeper;

---

Case 1: Long Race to a Beeper

- Move Karel through a row of “hurdles”
- Each pair of Avenues may or may not have a hurdle between them
- The race is arbitrary long
- There is a beeper at the end of the course
Solution: Long Race to a Beeper

Main program:
beginning-of-execution
while not-next-to-a-beeper do
  race-stride;
pickbeeper;
turnoff;
end-of-execution

Main subtask:
define-new-instruction
race-stride as
if front-is-clear then
  move
else
  jump-hurdle;

Solution (2): Long Race to a Beeper

Decomposing jump-hurdle:
define-new-instruction
jump-hurdle as begin
  jump-up;
  move;
  jump-down;
end;
define-new-instruction
jump-up as begin
  turnleft;
  move;
  turnright;
end;
define-new-instruction
jump-down as begin
  turnright;
  move;
  turnleft;
end;
Case 2: Lay Any Carpet

beginning-of-program
define-new-instruction
lay-carpet-side as
  while front-is-clear do begin
    move;
    putbeeper;
  end;
beginning-of-execution
  iterate 4 times begin
    lay-carpet-side;
    turnleft;
  end;
  turnoff;
end-of-execution
end-of-program

Steps of Building a While loop

- What should be true when Karel has to finish the loop?
- Use opposite condition for while test
- “Frame” the while - do what you need before/after to solve the problem
- Do the minimum what is needed to ensure that the loop eventually stops
## Loop Invariant and Changes

- At the beginning of every iteration:
  - What is always the same - some condition that is true when we need to execute the loop body and false when we do not need to do it anymore?
  - What is different for each subsequent iteration that makes the new situation closer to the solution than previous?

## Universal Harvest Program

![Graphs showing initial and final harvest situations.](image-url)
Original Solution for Harvest

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;
// at the beginning of every iteration Karel stands at the beginning of the next double row facing east
while next-to-a-beeper do begin
   harvest-1-row;
   go-to-next-row;
   harvest-1-row;
   position-for-next;
end;
position-for-next;
move;
turnoff;
end-of-execution
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;
// at the beginning of every iteration Karel stands at the beginning of the next double row facing east
while next-to-a-beeper do begin
   harvest-1-row;
   go-to-next-row;
   harvest-1-row;
   position-for-next;
end;
position-for-next;
move;
turnoff;
end-of-execution
end-of-program

While Loops in Harvest

beginning-of-execution
move;
// at the beginning of every iteration Karel stands at the beginning of the next double row facing east
while next-to-a-beeper do begin
   harvest-1-row;
   go-to-next-row;
   harvest-1-row;
   position-for-next;
end;
position-for-next;
move;
turnoff;
end-of-execution

- What is true at the beginning of every iteration?
  - at the beginning of every iteration Karel stands at the beginning of the next double row facing east

- What is different for each subsequent iteration that makes it closer to the solution?

- How we had to “frame” this loop?
While Loops in Harvest

define-new-instruction harvest-1-row as begin
  while next-to-a-beeper
do begin
    pickbeeper;
    move;
  end;
  step-back;
end;

define-new-instruction step-back as begin
  turnleft;
  turnleft;
  move;
  turnleft;
  turnleft;
end;

- What is true at the beginning of every iteration?
- What is different for each subsequent iteration that makes it closer to the solution?
- How we had to "frame" this loop?

Before next lecture:

- Do reading assignment
  - Pattis: Chapter 5
  - Tutorial: lessons 8, 11
- Run classroom examples
- Check yourself by doing any 3 out of exercises 4-13 from Section 5.9
- HW4 is due on 2/12/07