List-based Stacks and Queues

CMPT 145

Objectives

After this topic, students are expected to be able to:

1. Employ data abstraction to implement a queue.
2. Employ data abstraction to implement a stack.
3. Describe the list-based implementation of stacks and queues.
4. Explain some of the reasons that wrapping an ADT around a list can improve correctness and adaptability.

Two implementations

```
1  # initialize data structures
2  expr_list = exprn.split()
3  evalu8 = []
4
5  # evaluate the expression
6  for c in expr_list:
7
8    if c == '*':
9      v1 = evalu8.pop()
10     v2 = evalu8.pop()
11     evalu8.append(v1*v2)
12   ...
```

```
1  # initialize data structures
2  expr_list = exprn.split()
3  evalu8 = Stack.create()
4  expr = Queue.create()
5
6  # put items into Queue
7  for c in expr_list:
8    Queue.enqueue(expr, c)
9
10 # evaluate the expression
11 while not Queue.is_empty(expr):
12   c = Queue.dequeue(expr)
13
14   if c == '*':
15     v1 = Stack.pop(evalu8)
16     v2 = Stack.pop(evalu8)
17     Stack.push(evalu8, v1*v2)
18   ...
```

Using ADTs instead of lists

- Clarifies the algorithm
- Prevents errors, e.g., `pop()` vs. `pop(0)`
- Enhances adaptability (in general)
- Makes the script longer

Adapting Python Lists to Implement Queues

- An adaptor is a program that makes ADT A look like another ADT B
- If we have ADT A, we can write operations that make A look like it's a B.
- We'll adapt Python lists to implement a Queue
- Very common programming task!
- Usually not this trivial!

Code walk-through

Adapting Python Lists to Implement Stacks

- We'll adapt Python lists to implement a Stack

Code walk-through

Exercise 1

1. Change the Queue code so that the front and back are reversed.
2. Change the Stack code so that the top is on the opposite end of the list.

In both cases, change the ADT implementation only. Code that uses Queue or Stack will not change!

Exercise 2

For the following scenarios:

- Is it better to use Stack and Queue ADTs;
- Or is it okay to use Python lists without any abstraction?

1. CMPT 145, for educational purposes!
2. Large projects, important projects, paid projects.
3. Your code is not part of a long term solution.
4. Your code is a very short-lived prototype
5. Your code will not run more than a few times
6. Software engineering course assignments and projects