Prefix/Postfix and LISP

Dr. Baldassano

chrisb@princeton.edu

Yu’s Elite Education

Today and next week

- Today: 2 related ACSL topics:
  - Prefix/Postfix notation
  - LISP
- Next week – start with new instructor

Writing a math program

- Say we want to write a program to evaluate this expression:

  3+5*4^2-(3-8)*3

- This is going to be very complicated – we'll need to program in order of operations, do multiple passes…

Prefix notation

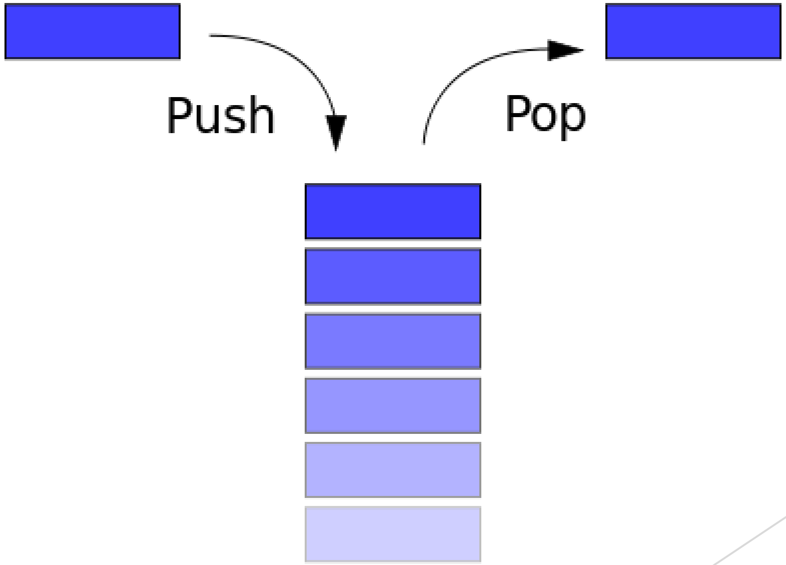
- Instead let’s write expressions with operator first:

```
1 + 1       =>    + 1 1
10 - 5      =>    - 10 5
3 * 2       =>    * 3 2
1 + 3 * 2   =>   + 1 * 3 2
```

- How can we evaluate these?

Stack

- We’re going to use a data structure called a stack
- A stack is a list where items get added and removed at the top



Stack

- Example:
  - Push !
  - Push H
  - Push I
  - Pop
  - Pop
  - Pop

Prefix notation with stacks

- Push each item onto the stack
- Whenever there is an operation and two numbers at the top of the stack, pop them off and push on the result

Examples

Postfix notation

- Can also do the opposite: put the operation after the numbers

```
1 + 1      =>    1 1 +

10 - 5     =>    10 5 -

3 * 2      =>    3 2 *

1 + 3 * 2  =>   1 3 2 * +
```

Converting to pre/postfix examples

ACSL Sample Problems

- Convert to postfix: $$\frac{\left(A-\frac{B}{C}+D\right)^{\frac{1}{2}}}{A+B}$$

- Given A=4, B=14 and C=2, evaluate the following prefix expression:

```
* / - + A B C * A C B
```

LISP

- The idea of prefix operators can be used to build a whole programming language
- LISP = LISt Processing
- Only two kinds of things exist in LISP:
  - Atoms: individual items (numbers, functions, data…)
  - Lists of atoms

Example LISP Programs

```
(MULT 2 3)               6
(ADD 1 2 3)              6
(ADD 3 (MULT 3 4))       15
(SUB 6 (SQUARE 2))       2
(EQ 4 (SQUARE 2))        TRUE
(EQ 10 (DIV 20 4))       FALSE
(POS -4)                 FALSE
(NEG (SUB 10 20))        TRUE
```

LISP list functions

- CAR function: equals first item of list

```
(CAR ‘(10 4 1))    => 10
```

- CDR function: equals all but first item of list

```
(CAR ‘(10 4 1))    => (4 1)
```

- The quote character tells LISP not to try to evaluate a list

CAR/CDR examples

ACSL Sample problems

```
(EXP (MULT 2 (SUB 5 (DIV (ADD 5 3 4) 2)) 3) 3)
```

```
(CDR '((2 (3))(4 (5 6) 7)))
```