A SIMPLE CALCULATOR AND DRAGON CURVES

Lecture notes of the course “*Programming Techniques*”

Lê Hồng Phương[1]

[1]Department of Mathematics, Mechanics and Informatics
Hanoi University of Science, VNUH
<*phuonglh@gmail.com*>

10/2012

Content

1. **Simple calculator**
   - Prefix, infix and postfix expressions
   - Shunting yard algorithm
   - Evaluating postfix expressions

2. **Dragon curves**
   - Dragon curves
   - Drawing dragon curves

3. **Exercises**

Content

Content

Content

1. **Simple calculator**
   - Prefix, infix and postfix expressions
   - Shunting yard algorithm
   - Evaluating postfix expressions

2. **Dragon curves**
   - Dragon curves
   - Drawing dragon curves

3. **Exercises**

Content

Prefix, infix and postfix expressions

- Stacks and queues allow us to design a simple expression evaluator.
- Prefix, infix and postfix notation: operator before, between and after operands, respectively.
- Infix is more natural to write, postfix is easier to evaluate.

Examples:

Infix	**Prefix**	**Postfix**
$A + B$	$+AB$	$AB+$
$A * B - C$	$- * ABC$	$AB * C-$
$(A + B) * (C - D)$	$* + AB - CD$	$AB + CD - *$

Prefix, infix and postfix expressions

- How can we convert an infix expression to its postfix expression?

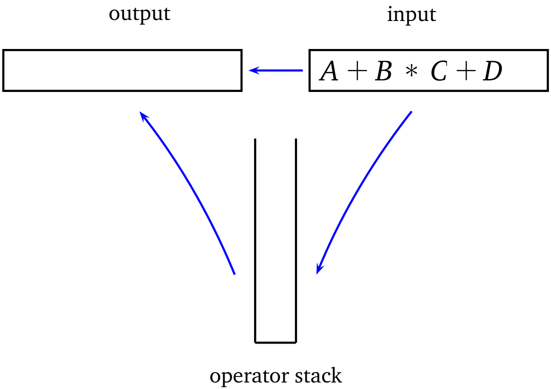
$$\underbrace{(A+B)*(C-D)}_{\text{infix}} \Longrightarrow \underbrace{AB+CD-*}_{\text{postfix}}$$

- The "Shunting yard" algorithm of Dijisktra (1961):
  - uses two queues to store input and output
  - use a separate stack for holding operators
- We shall consider the simplest version of this problem where we have *only binary operators*.

Content

1. **Simple calculator**
   - Prefix, infix and postfix expressions
   - Shunting yard algorithm
   - Evaluating postfix expressions
2. **Dragon curves**
   - Dragon curves
   - Drawing dragon curves
3. **Exercises**

Shunting yard algorithm



Shunting yard algorithm

1. Dequeue token from input;
2. If the token is an operand (number), add it to output queue;
3. If it is an operator then pop operators off stack and add to output queue as long as:
    - top operator on stack has higher precedence, or
    - top operator on stack has same precedence and is *left-associative*

   and push new operator onto stack;
4. Return to step 1 as long as tokens remain in input;
5. Pop remain operators from stack and add to output queue.

Operator associativity

- A property that determines how operators of the same precedence are grouped in the absence of parentheses.
- Operator may be *left-associative* or *right-associative*, meaning that the operators are grouped from the left or from the right.
- Example: consider the expression $a \sim b \sim c$.
  - If the operator $\sim$ is left-associative, the expression would be evaluated as $(a \sim b) \sim c$.
  - If the operator $\sim$ is right-associative, the expression would be evaluated as $a \sim (b \sim c)$.

Normal mathematical usage:

- Addition, subtraction, multiplication and division operators are usually left-associative.
- Exponentiation and assignment operators are typically right-associative.

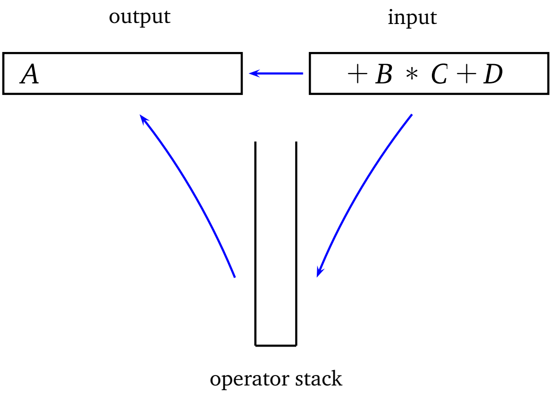
Operator associativity

- A property that determines how operators of the same precedence are grouped in the absence of parentheses.
- Operator may be *left-associative* or *right-associative*, meaning that the operators are grouped from the left or from the right.
- Example: consider the expression $a \sim b \sim c$.
  - If the operator $\sim$ is left-associative, the expression would be evaluated as $(a \sim b) \sim c$.
  - If the operator $\sim$ is right-associative, the expression would be evaluated as $a \sim (b \sim c)$.

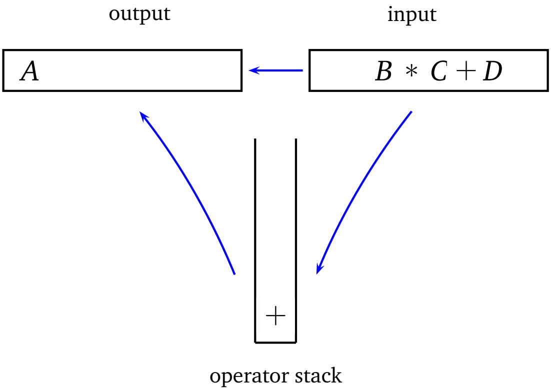
Normal mathematical usage:

- Addition, subtraction, multiplication and division operators are usually left-associative.
- Exponentiation and assignment operators are typically right-associative.

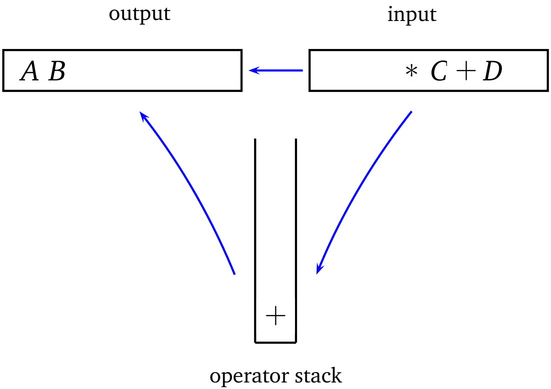
Shunting yard algorithm



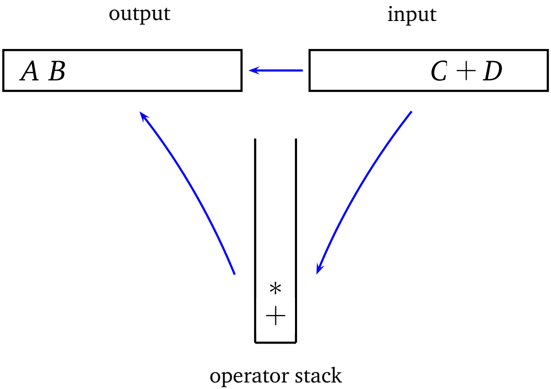
Shunting yard algorithm



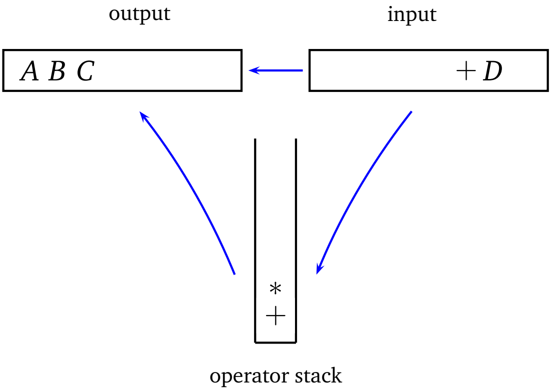
Shunting yard algorithm



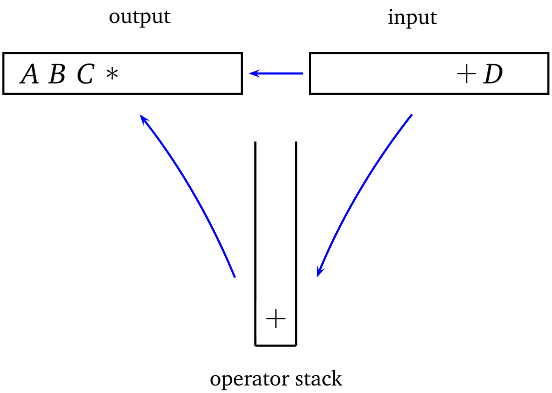
Shunting yard algorithm



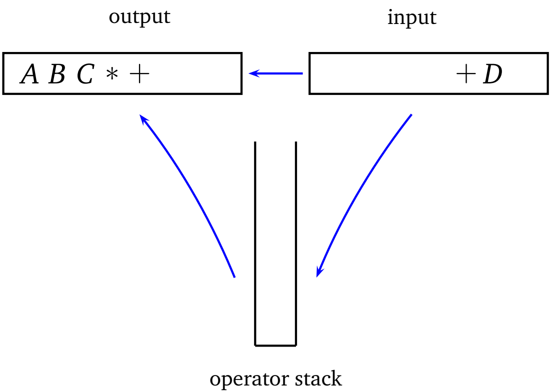
Shunting yard algorithm



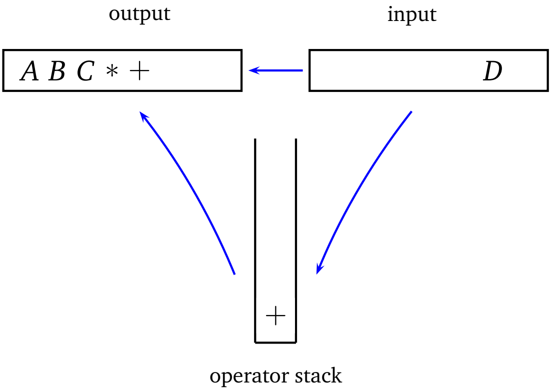
Shunting yard algorithm



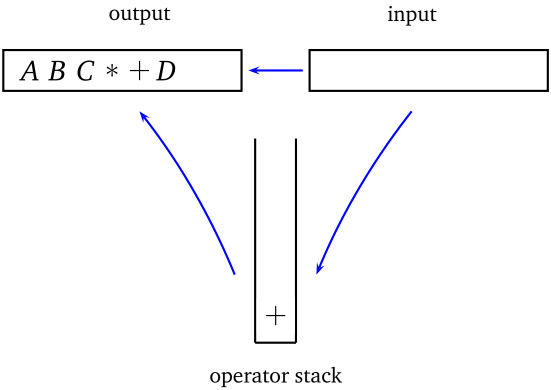
Shunting yard algorithm



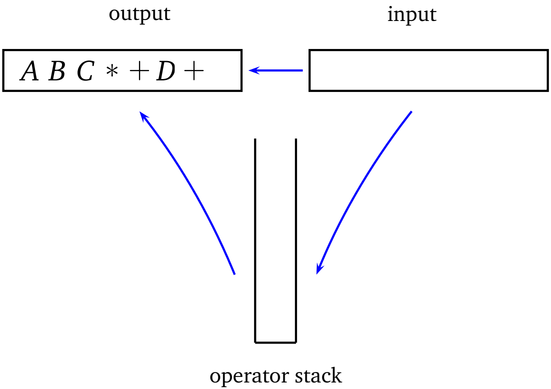
Shunting yard algorithm



Shunting yard algorithm



Shunting yard algorithm



Shunting yard algorithm

- What if infix expression includes parentheses?
- Example:

$$(A + B) * (C - D) \Longrightarrow ?$$

- For short, we use a table to track steps of the algorithm.

Shunting yard algorithm

Token	Output	Stack
$($		$($
A	A	$($
$+$	A	$(+$
B	AB	$(+$
$)$	$AB+$	
$*$	$AB+$	$*$
$($	$AB+$	$*($
C	$AB+C$	$*($
$-$	$AB+C$	$*(-$
D	$AB+CD$	$*(-$
$)$	$AB+CD-$	$*$
end	$AB+CD-*$	

The result is $AB+CD-*$.

Content

Evaluating postfix expressions

Postfix evaluation is easy with a stack. Here is the algorithm:

1. Dequeue a token from the postfix queue;
2. If token is an operand, push it onto stack;
3. If token is an operator:
   - pop operands off stack (2 operands for binary operator);
   - evaluate the expression;
   - push result onto stack;
4. Repeat until postfix queue is empty;
5. Item remaining in stack is the final result.

Evaluating postfix expressions

Example: evaluating the expression $6\,5 + 7\,2 - *$

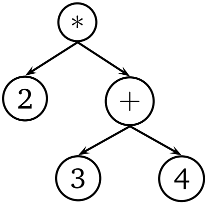
Token	Stack	Evaluation
6	6	
5	$6, 5$	
$+$	11	compute $6 + 5 = 11$
7	$11, 7$	
2	$11, 7, 2$	
$-$	$11, 5$	compute $7 - 2 = 5$
$*$	55	compute $11 * 5 = 55$

The result is 55. Note that the equivalent infix expression is

$$(6 + 5) * (7 - 2).$$

Expression trees

- We can use binary trees to represent expressions: leaf nodes are operands, internal nodes are operators.
- Prefix, infix and postfix expressions are obtained by using pre-order, in-order and post-order traversal on a tree.



- Infix expression: $2 * (3 + 4)$.

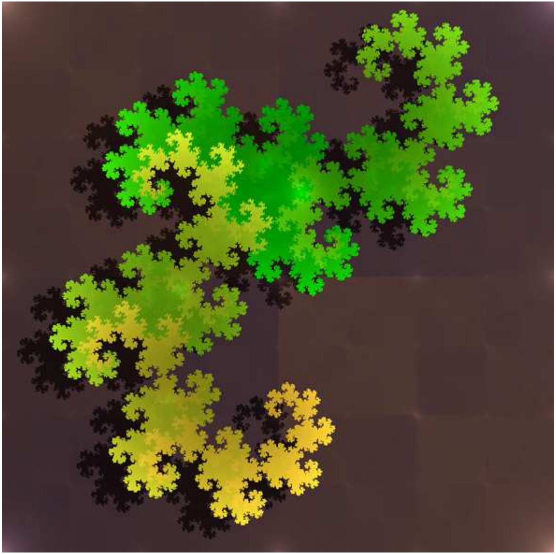
Content

Content

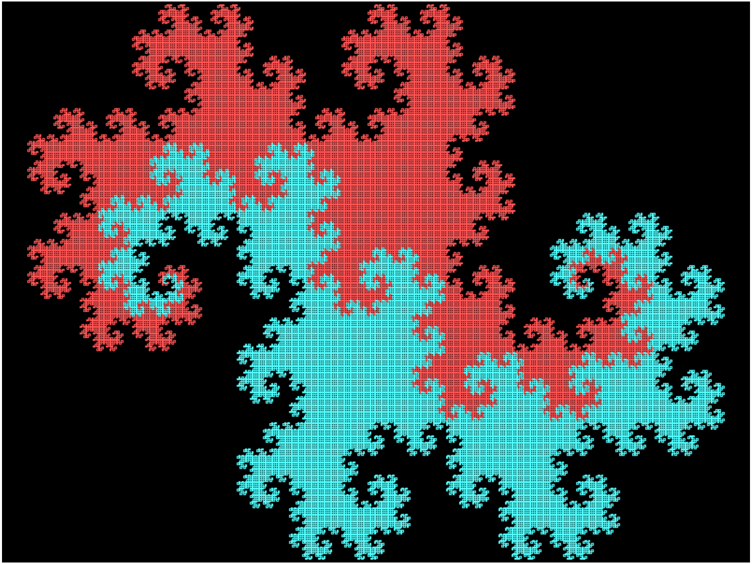
Dragon curves

- A dragon curve is any member of a family of **self-similar fractal curves**, which can be approximated by recursive methods.
- The Jurassic Park dragon was first invented by NASA physicists (John Heighway, Bruce Banks and William Harter).
- It was described in 1967 by Martin Gardner in Mathematical Games column.
- More about this:
  - `http://en.wikipedia.org/wiki/Dragon_curve`
  - `http://mathworld.wolfram.com/DragonCurve.html`

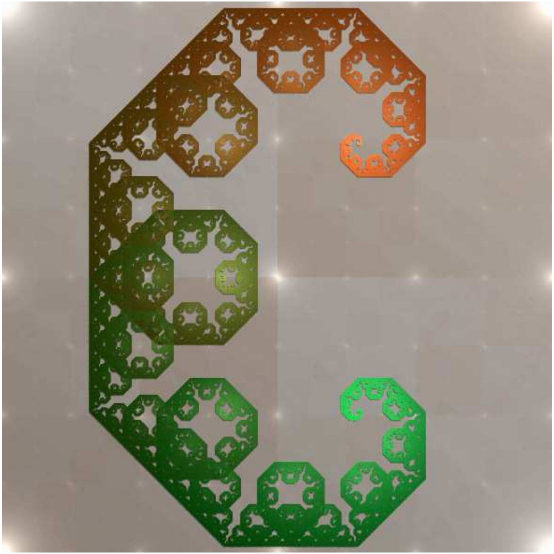
Dragon



Twindragon – Davis–Knuth dragon



Lévy dragon – Lévy C curve



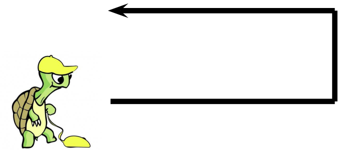
Content

Drawing dragon curves

- How can we draw a dragon programmatically?
- Simple techniques: strait lines and recursive operations
- Simple tools:
  - Postscript (PDF) commands for graphical lines
  - Text file I/O in C

Drawing dragon curves

- Use simple "turtle graphics":
  - `F` (forward): move turtle forward one step (pen down)
  - `L` (left): turn left 90 degrees
  - `R` (right): turn right 90 degrees
- Example: `F L F L F`



Drawing dragon curves

- `dragon(0): F`
- `dragon(1): F L F`
- `dragon(2): F L F L F R F`
- `dragon(3): F L F L F R F L F L F R F R F`
- `dragon(4):`
  - `dragon(3): F L F L F R F L F L F R F R F`
  - `L`
  - `nogard(3): F L F L F R F R F L F R F R F`

Rule?

- The first part of `dragon(n)` is `dragon(n-1)`.
- The second part of `dragon(n)` is the backward of `dragon(n-1)`: reverse string, switch `L` and `R`.

Drawing dragon curves

- `dragon(0): F`
- `dragon(1): F L F`
- `dragon(2): F L F L F R F`
- `dragon(3): F L F L F R F L F L F R F R F`
- `dragon(4):`
  - `dragon(3): F L F L F R F L F L F R F R F`
  - `L`
  - `nogard(3): F L F L F R F R F L F R F R F`

Rule?

- The first part of `dragon(n)` is `dragon(n-1)`.
- The second part of `dragon(n)` is the backward of `dragon(n-1)`: reverse string, switch `L` and `R`.

Simple PostScript commands

```
FILE *f;

void forward() {
  fprintf(f, "5 0 rlineto\n");
}

void left() {
  fprintf(f, "90 rotate\n");
}

void right() {
  fprintf(f, "-90 rotate\n");
}
```

Recursive functions

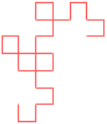
<pre>void dragon(int n) {
  if (n == 0)
    forward();
  else {
    dragon(n-1);
    left();
    nogard(n-1);
  }
}</pre>
	<pre>void nogard(int n) {
	  if (n == 0)
	    forward();
	  else {
	    dragon(n-1);
	    right();
	    nogard(n-1);
	  }
	}</pre>

The main function

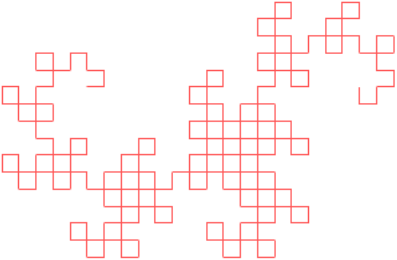
```
int main(int argc, char **argv) {
  f = fopen("dragon.ps", "w");
  fprintf(f, "%%!PS\n");
  fprintf(f,"%% Set the page size to A4\n");
  fprintf(f,"<< /PageSize [595 842] >> setpagedevice\n");
  fprintf(f, "0.1 setlinewidth\n");
  fprintf(f, "400 400 moveto\n");
  dragon(12);
  fprintf(f, "1 0 0 setrgbcolor\n");
  fprintf(f, "stroke\n");
  fprintf(f, "showpage\n");
  fclose(f);
  return 0;
}
```

Examples: $n = 5$, $n = 8$

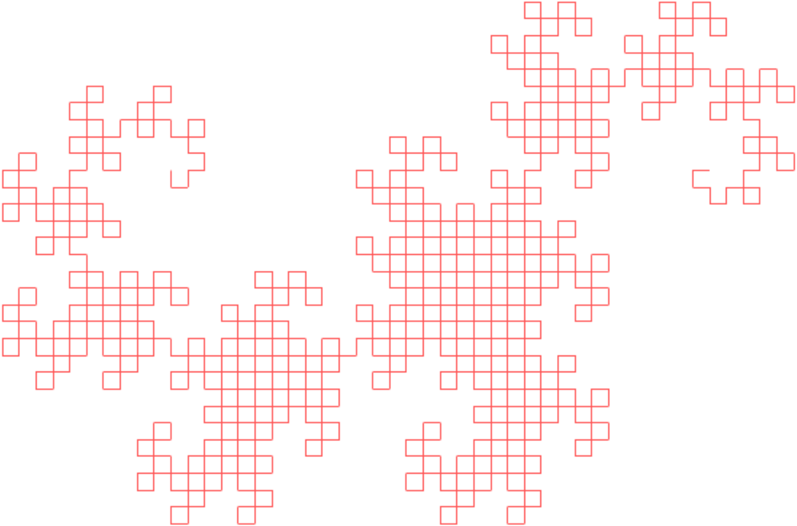
$n = 5$:



$n = 8$:



Examples: $n = 10$



Examples: $n = 12$

Content

Exercises

Exercise 1. Implement the shunting yard algorithm for converting an infix expression to its postfix expression.

Exercise 2. Implement the algorithm for evaluating a postfix expression.

Exercise 3. Implement a program which draws dragon curves.

- The program accepts an argument which is n;
- Try to use different colors for the dragons.