Program.jan

•••

Jan C. Bierowiec

Design Goals

- Readability, writability, reliability, cost, simplicity, high functionality
  - Readability:
    - Provide detailed documentation
    - Easy to understand for programmers
  - Writability:
    - Easy to write programs
  - Reliable:
    - Strictly related to animations, simulations, and visualizations
  - Cost:
    - Small learning curve
    - Catered to a broad audience
  - High functionality:
    - Memory would be allocated automatically

Targeted Domains

- Scientific & Educational
    - Can be used to learn mathematical programming
    - Can be used for calculations, plots, simulations
    - Used by both students & professionals
- Real-time:
    - Can process data inputted into it quickly
- Computationally intensive:
    - Can make calculations & other additional actions based on coder’s choice

Targeted Users

- Domain experts
  - Scientists interested in simulations/mathematical computation
- Students
  - A good introduction to plotting, animation & simulation
- Professional programmers
  - Programmers interested in developing visualizations for a wide range of topics

Type of Language & Features

- Compiled language
  - Procedural & Object Oriented
- What differentiates your language from others?
  - Syntax
  - Interactive debugger that lets you visualize the code. Use more of the visual cortex, graphs, visualizations, the works.
  - Concurrency & Parallelism

Example Program

```
f(x):
    return x * x.

let i.
let x.

output("Enter a number").
input(x).

output("All of the squares up until", x, "are: ").
for(i = 1. i <= x. i = i + 1.):
    let result = f(x).

    output(result, ", ").
```

Walk-Through of the Backus-Naur Form Notation Part 1

0.0 <program> ::= <statement_list>
<statement_list> ::= <function-declaration> <function-declaration>

1. <function_declaration> ::= f <identifier> (<parameter_list>): <statement_list>
<parameter_list> ::= ( ) | (<identifier>) | (<identifier> , <identifier>)
<function_declaration> ::= f sq (<parameter_list>): <statement_list>
<parameter_list> ::= ( ) | (x) | (<identifier> , <identifier>)

1.1. <statement_list> ::=
<return_statement> ::= return <expression> .
<return_statement> ::= return x * x.

Walk-Through of the Backus-Naur Form Notation Part 2

2. <statement_list> ::=
<declaration> ::= let <identifier> .
<declaration> let i.
<declaration> ::= let <identifier> .
<declaration> let x.

2.1. <statement_list> ::=
<output_statement> ::= output (<expression_list>) .
<output_statement> ::= output ("Enter a number").
<input_statement> ::= input (<expression_list>) .
<input_statement> ::= input (x).
<output_statement> ::= output (<expression_list>) .
<output_statement> ::= output ("All of the squares up until", x, "are: ").

Walk-Through of the Backus-Naur Form Notation Part 3

2.2. <statement_list>

<for_loop> ::= for (<declaration> <condition> . <expression>): { <statement_list> }

<for_loop> ::= for (i = 1. i <= x. i = i + 1.): { <statement_list> }

2.3. <statement_list> ::=

<definition> ::= let <identifier> = <expression> .

<definition> ::= let result = sq(x).

<function_call> ::= <identifier> (<optional_args>)

<function_call> ::= sq (x).

2.4. <output_statement> ::= output (<expression_list>) .

<output_statement> ::= output (result, ", ").

Thank You!