Project 2 - Reverse Polish Notation (Documentation)

By Mitchell Mackoul

Tested System Info:

OS: **Microsoft Windows 10 Home**: Version *10.0.19043* (x64 based)
IDE: Repl.it (Primary Testing)
Commit: 6cba118ac49a1b88332f312a8f67186f7f3c1643

Description:

The program is a calculator that reads the user's input through the command line, pushes the inputs into a Stack via space-separated arguments, solves the arithmetic, and returns the solution to the user.

How to Run:

The program needs to be executed in the command line, calling the **main** file. Following that are space separated arguments that are passed in in Reverse Polish Notation (RPN) (e.g. Normal Notation: "1 + 1 = 2"; RPN: "1 1 + 2 =").

Classes:

Node:

- Data Members:
    - Operand (type **double)**, which holds one of the numbers to be operated on.
    - Operator (type **char**), which holds the operator to be performed on the previous two Operands in the stack.
    - next* (pointer of type Node), which points down to the next Node within the Stack.
- Functions:
    - Node(double _Operand)
        - A constructor used when pushing Operands onto the stack.
        - next is set to nullptr, since the new Node is to be at the top of the Stack.
        - Sets the Operator equal to 'F'. This is used in comparisons for checking the data types held within the Node.
    - Node(char _Operator)
        - A constructor used when pushing Operators onto the stack.
        - next is also set to nullptr, since the new Node is, again, to be at the top of the Stack.

    - ~Node()
        - A destructor that is called whenever a Node object is deleted.
    - void setOperand(double operandArg)
        - Allows the program to change the Operand of a specific Node.
    - void setOperator(char operatorArg)
        - Allows the program to change the Operator of a specific Node.
    - double getOperand()
        - Returns the Operand of a specific Node.
    - char getOperator()
        - Returns the Operator of a specific Node.

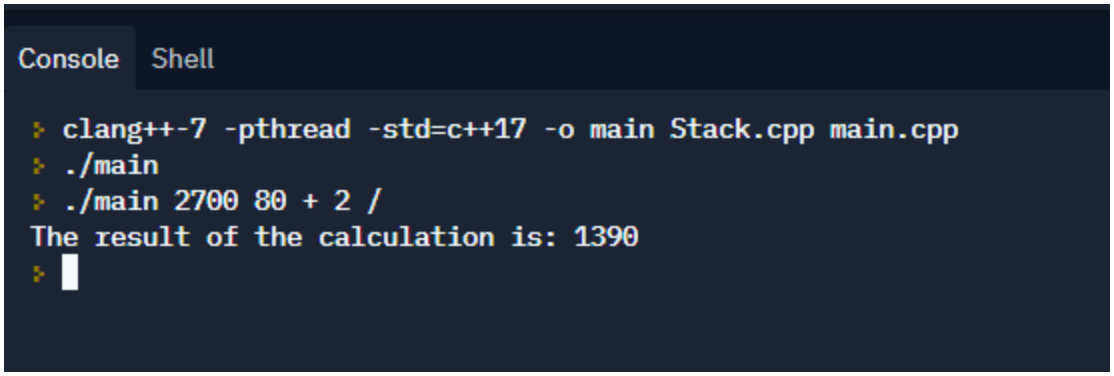
Stack:

- Inherits the functions and data members of the "Node" class.
- Data Members:
    - Uint64_t size
        - Stores the current size of the Stack.
    - size_t maxCapacity
        - Stores the maximum allowed size of the Stack to prevent *Stack Overflow*.
    - Node *topOfStack
        - A Pointer that points to the Node at the top of the Stack.
- Functions:
    - Stack()
        - A default constructor that goes unused.
    - ~Stack()
        - A destructor that deletes all of the Nodes when the object has completed its tasks.
    - void Clear()
        - A function used to clear the entire Stack.
    - bool isEmpty()
        - A function that checks if the Stack is empty.
    - void pushOperator(char)
        - A function that creates a new Node with a Char argument and pushes it to the top of the Stack.
            - The Char argument is always an Operator passed in from the Command Line.
    - void pushOperand(double)
        - A function that creates a new Node with a Double argument pushes it to the Stack.

            - The Double argument is always an Operand passed in from the command line.
    - char getOperator()
        - Returns the Operator of a specific Node.
    - uint64_t getSize()
        - Returns the current size of the Stack. It is compared to maximumCapacity to prevent Stack Overflow.
    - void pop()
        - Removes a Node from the top of the Stack, and changes the topOfStack pointer to the Node below the previous top.

    - void top()
        - Prints the Node at the top of the Stack. Used for testing the pushOperand(), pushOperator(), and pop() functions during debugging.
    - double evaluate()
        - A recursive function.
            - Begins by setting the first Operand, *Operand1*, equal to the top of the Stack, then popping from the top of the Stack.
            - From there, the second Operand, *Operand2*, is set equal to the top of the Stack, then the Stack is once again popped.
            - Next, the Operator, *Operator*, is set equal to the top of the Stack, then the Stack is once again popped.
            - The function will then use an if statement to determine which arithmetic function (‘+’, ‘-’, ‘*’, or ‘/’) to use.
            - The result is set equal to a double Solution.
            - If the Stack is not empty, Solution will be pushed onto the top of the Stack, and evaluate() will be called again, with Operand1 being set to Solution, Operand2 being set to the next Operand, and the Operator being set to the operator afterward.
        - void setCapacity(size_t capacity)
            - A function to set the maximum size of the Stack. If the size is equal to maxCapacity, it will be impossible for the user to add more Nodes.
        - Size_t getCapacity()
            - A function that returns the maximum capacity of the Stack.

Runtime Examples




```
./main 1000 2000 + 3000 + 2 /
The result of the calculation is: 3000
3000

./main 2 2 * 2 * 2 *
The result of the calculation is: ./main 2 2 \* 2 \* 2 \*
The result of the calculation is: 16
16
```