Instituto Tecnológico de Aeronáutica
Divisão de Ciência da Computação (IEC)
Lab 02 – CES 11 – Algoritmos e Estruturas de Dados - 2021

Introdução

Existem duas notações principais para inserir informações em calculadoras portáteis. Na maioria das vezes, as calculadoras HP implementam uma notação polonesa reversa (RPN), ou notação "post-fix ", enquanto as calculadoras da TI (Texas Instruments) implementam o notação “in-fix” tradicional. A principal diferença entre os dois estilos é a ordem em que as operações matemáticas são inseridas e executadas.

RPN é um método baseado em pilha de processamento de entrada. Os números são lidos e processados na ordem inversa em que são inseridos, enquanto as próprias operações são lidas e processados sequencialmente.

Por outro lado, a notação “in-fix” é o método tradicional que a maioria dos livros didáticos modernos adota. É basicamente um método de processamento baseado em símbolos para o processamento de expressões matemáticas. Portanto, faz uso de um conjunto de regras de precedência que definem a ordem em quais valores são processados pelo encapsulamento de operadores e símbolos.

RPN tem uma vantagem clara sobre a notação "in-fix" porque RPN oferece uma implementação muito simples, e naturalmente evita ambiguidades e não requer parênteses. Contudo, a menos que o usuário seja bem treinado na arte de RPN, ele não é automaticamente intuitivo em seu uso.

No RPN, os números na parte inferior da pilha são processados primeiro e, em seguida, "são empilhados” à medida que os valores são reduzidos em direção ao topo. Claramente, o formato mais intuitivo do ponto de vista acadêmico é a notação “in-fix”, porém requer lidar com questões de precedência de operadores.

Esses são alguns exemplos de pares de expressões no estilo “in-fix” e RPN lado a lado (note que ~ é equivalente a expressão –(x)):

4 * - (4 + 10 / 2,5 - 8) 4 4 10 2,5 / + 8 - ~ *
17,2 + 21 * 3 / (-7) + 21 17,2 21 3 * 7 ~ / + 21 +
((- (2 + 3 * 8)) ^ 4) * 7 2 3 8 * + ~ 4 ^ 7 *

No laboratório 2, você deve escrever um programa que leia, analise e resolva equações no estilo RPN. Você pode supor que a entrada virá diretamente de uma entrada padrão do terminal (teclado ou dispositivo relacionado) e que a saída deve ser direcionada para a saída padrão para esse terminal (monitor ou dispositivo). No caso, o Run Codes executará os procedimentos dessa forma.

Suas expressões RPN devem ser capazes de ler os seguintes operadores.

\+ Operador de adição. Adiciona dois operandos.

\- Operador de subtração. Subtrai dois operandos juntos.

* Operador de multiplicação. Multiplica dois operandos.

/ Operador de divisão. Divide dois operandos, na ordem em que são colocados na pilha.

c Operador de limpeza da pilha.

Cada linha de entrada será inserida na forma relativa de:

{números} {operadores}

Cada número e operador pode ser considerado separado por alguma forma de espaço em branco para fazer analisar a entrada mais facilmente. Uma sessão de amostra está abaixo, e os use cases do run.codes são os mesmos. O final da string da equação deve finalizar com =. No final de cada cálculo, a calculadora é considerada apagada e a pilha deve ser esvaziada. A aritmética de ponto flutuante deve ser assumida e os programas devem permitir números não inteiros expressões como entrada válida.

3 4 5 + 1 - * =

Resultado: -24.000000

IMPORTANTE: O programa requer que você faça uso do *template* de tipo abstrato de pilha abaixo.

```c
/*
 * Abstract Data Type Declarations -- STACK
 * author: dloubach
 *
 */

#ifndef ADT_STACK_H
#define ADT_STACK_H

#include <stdbool.h>

/* ADT stack type declaration */
typedef struct node_st* stack_t;

/* ADT stack operators */

/* inicializa uma pilha vazia */
void adt_initStack(stack_t*);

/* adiciona um elemento no topo da pilha*/
void adt_pushStack(stack_t*, double);

/* remove um elemento do topo da pilha */
void adt_popStack(stack_t*);

/* retorna o elemento do topo da pilha sem remove-lo */
double adt_topStack(stack_t);

/* retorna <true> quando a pilha encontra-se vazia */
bool adt_isEmptyStack(stack_t);

/* esvazia a pilha */
void adt_emptyStack(stack_t*);

#endif
```