Atividade B2-1 – Manipulação de Pilha – Implementado HP12C

Enunciado:

Instruções

Nesta aula #09 vimos que uma a Pilha é uma lista com a restrição que inserções e remoções são executadas exclusivamente em uma posição, referenciada como fim ou topo.

Vimos também que a estrutura de dados de Pilha é muito utilizada na vida real (Prática). o exemplo disto é a programação da calculadora financeira HP12c. Esta calculadora utiliza a *Notação Polonesa inversa* (**RPN**-*Reverse Polish Notation*) e entrada no sistema de **pilha de memória**: X, Y, Z e T.

A partir deste cenário, pede-se:

- Implementar um programa na linguagem C
- Este programa deve simular a calculadora HP12c
- Utilizar a estrutura de **pilha** para guardar e processar as entradas (fórmulas matemáticas) nas memórias X; Y; Z e T

Regras de negócio:

1. Solicitar as entradas no formato RPN
2. A fórmula deverá ter "n" operando e "n" operadores
3. Operadores permitidos: *; /; + e -
4. o RPN recebido deve ser armazenado em Array
5. Ao final deve ser informado o resultado da operação aritmética e a fórmula algébrica (Convertido do RPN)

Entrega no GitHub: Lembre-se do nome da pasta: **Atividade-B2-1**

Link para o repositório: https://github.com/CamilleGFAlmeida/Fatec-AMS-ED2024-1-1681432412033-Camille/tree/main/Atividade-B2-1

Código Fonte:

```
/*------------------------------------------------------------------------*/
/*   FATEC - São Caetano do Sul           Estrutura de Dados  */
/*                                                                        */
/*                          Camille Guillen                               */
/*          Objetivo: Manipulação de Pilhas - HP12C                       */
/*                                                                        */
/*                                          Data:21/04/2024               */
/*------------------------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>
#include <stdbool.h>

#define STACK_SIZE 100

// Definição da estrutura da pilha
typedef struct {
   double items[STACK_SIZE];
   int top;
} Stack;

// Funções auxiliares para manipulação da pilha
void push(Stack *s, double value) {
   if (s->top == STACK_SIZE - 1) {
      printf("A Pilha está cheia!\n");
      exit(EXIT_FAILURE);
   }
   s->items[++(s->top)] = value;
```

```
}

double pop(Stack *s) {
  if (s->top == -1) {
    printf("A Pilha está vazia!\n");
    exit(EXIT_FAILURE);
  }
  return s->items[(s->top)--];
}

// Função para calcular o resultado da expressão RPN
double calculateRPN(char *rpn) {
  Stack s;
  s.top = -1;

  while (*rpn != '\0') {
    if (isdigit(*rpn) || (*rpn == '-' && isdigit(*(rpn + 1)))) {
      push(&s, atof(rpn));
      while (isdigit(*rpn) || *rpn == '.') rpn++;
    } else if (*rpn == '+' || *rpn == '-' || *rpn == '*' || *rpn == '/') {
      double op2 = pop(&s);
      double op1 = pop(&s);
      switch (*rpn) {
        case '+':
          push(&s, op1 + op2);
          break;
        case '-':
          push(&s, op1 - op2);
          break;
        case '*':
          push(&s, op1 * op2);
```

```
                break;
            case '/':
                push(&s, op1 / op2);
                break;
        }
        rpn++;
    } else {
        rpn++;
    }
    while (*rpn == ' ') rpn++;  // Ignorar espaços em branco
  }

  if (s.top != 0) {
    printf("Erro: Expressão RPN inválida \n\nDigite ! para inserir outra expressão: \n");
    return NAN; // Retorna um valor NaN (Not a Number)
  }

  return pop(&s);
}

int main() {
  char rpn[STACK_SIZE];
  bool expressionValid = false;

  do {
    printf("\nDigite a expressão RPN: ");
    fgets(rpn, STACK_SIZE, stdin);

    // Calcula o resultado da expressão RPN
    double result = calculateRPN(rpn);
    if (!isnan(result)) {
```

```
            printf("Resultado: %.2f\n", result);

            // Verifica se a expressão RPN é válida
            printf("\nDeseja calcular outra expressão RPN? (s/n): ");
            char choice;
            scanf(" %c", &choice);
            if (choice != 's' && choice != 'S')
                expressionValid = true;
            else
                while (getchar() != '\n'); // Limpa o buffer do teclado
        } else {
            while (getchar() != '\n'); // Limpa o buffer do teclado
        }

    } while (!expressionValid);

    printf("\nPrograma encerrado.\n");

    return 0;
}
```