FATEC IPIRANGA

PASTOR ENÉAS TOGNINI

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

PROGRAMAÇÃO ESTRUTURADA E MODULAR

PROFESSOR CARLOS HENRIQUE VERISSIMO PEREIRA

MILENA MITIE AOKI

SÃO PAULO, SP
2024

SUMÁRIO

ATIVIDADE N1-6 - CALCULADORA HP12C

PROGRAMAÇÃO EM LINGUAGEM C

```
/*-------------------------------------------------------*
* Disciplina: Programaçao Estruturada e Modular          *
*          Prof. Carlos Veríssimo                        *            *                                                  *
*-------------------------------------------------------*
* Objetivo do Programa: Atividade N_6: Calculadora HP12c *
* Data - 20/09/2024                                      *
* Autor: Milena Mitie                                    *
*-------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define TAM_PILHA 4

// Estrutura de dados da pilha com 4 elementos
typedef struct {
    int dados[TAM_PILHA];
    int topo;
} Pilha;

// Funções da pilha
void inicializar_pilha(Pilha *p);
int pilha_vazia(Pilha *p);
int pilha_cheia(Pilha *p);
void push(Pilha *p, int valor);
int pop(Pilha *p);

// Função para realizar operações aritméticas
int operar(int op1, int op2, char operador);

// Função para processar a notação RPN
void processarRPN(char entrada[]);

int main() {
    char entrada[100];
    int continuar = 1;

    printf("Bem-vindo à Calculadora Fatec-HP12c!\n");

    while (continuar) {
        printf("Digite a fórmula RPN (ex: 5 8 + 3 *): ");
```

```
        fgets(entrada, 100, stdin);

        processarRPN(entrada);

        printf("Deseja fazer outra operação? (1 - Sim, 0 - Não): ");
        scanf("%d", &continuar);
        getchar(); // Limpa o buffer de entrada
    }

    printf("Obrigado por usar nossa Calculadora Fatec-HP12c!\n");
    return 0;
}

void inicializar_pilha(Pilha *p) {
    p->topo = -1;
}

int pilha_vazia(Pilha *p) {
    return p->topo == -1;
}

int pilha_cheia(Pilha *p) {
    return p->topo == TAM_PILHA - 1;
}

void push(Pilha *p, int valor) {
    if (pilha_cheia(p)) {
        printf("Erro: Pilha cheia!\n");
        return;
    }
    p->topo++;
    p->dados[p->topo] = valor;
}

int pop(Pilha *p) {
    if (pilha_vazia(p)) {
        printf("Erro: Pilha vazia!\n");
        exit(1);
    }
    int valor = p->dados[p->topo];
    p->topo--;
    return valor;
}

int operar(int op1, int op2, char operador) {
    switch (operador) {
        case '+': return op1 + op2;
        case '-': return op1 - op2;
```

```
        case '*': return op1 * op2;
        case '/': return op1 / op2;
        default:
            printf("Operador inválido!\n");
            exit(1);
    }
}

void processarRPN(char entrada[]) {
    Pilha p;
    inicializar_pilha(&p);

    char *token = strtok(entrada, " ");
    while (token != NULL) {
        if (isdigit(token[0])) {
            push(&p, atoi(token));
        } else {
            int op2 = pop(&p);
            int op1 = pop(&p);
            int resultado = operar(op1, op2, token[0]);
            push(&p, resultado);
        }
        token = strtok(NULL, " ");
    }
    printf("Resultado: %d\n", pop(&p));
}
```

DIAGRAMA DE BLOCOS DA SOLUÇÃO