FATEC IPIRANGA – PASTOR ENÉAS TOGNINI

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ISABELLA SANAE KIYATAKE

DISCIPLINA: PROGRAMAÇÂO ESTRUTURADA E MODULAR

PROF. CARLOS VERÍSSIMO

Atividade – N1- Desafio PEM – Bugs da HP12c

SÂO PAULO

2024

Diagnóstico de erros:

1. DIVISÃO POR ZERO:

   - Erro: O programa tenta realizar a operação de divisão sem verificar se o divisor é zero, o que resultaria em erro de execução.

   Código original:
   ```
   if (pilha[0] == 0) {
    printf("Erro: DivisÃ£o por zero nÃ£o
   permitida.\n");
    return;
    }
   ```

   - Correção: Adicionada uma verificação antes de realizar a operação de divisão para evitar divisão por zero:

   Código corrigido:
   ```
   if (pilha[0] == 0) {
       printf("Erro: Divisão por zero não permitida.\n");
       return;
   }
   ```

2. ATUALIZAÇÃO DA PILHA APÓS OPERAÇÃO:

   - Erro: Após a execução de uma operação, a pilha estava sendo atualizada incorretamente, não liberando corretamente os espaços.

   Código original:
   ```
   pilha[0] = resultado;
    for (int i = 1; i < TAMANHO_PILHA; i++) {
    pilha[i] = pilha[i + 1]; // Shift nos valores para
   liberar espaÃ§o
    }
    pilha[TAMANHO_PILHA - 1] = 0; // Limpa o topo da pilha
   Programação Estruturada e Modular
   }
   ```

   - Correção: Ajustado o loop de deslocamento para que ele mova os elementos corretamente e limpe a última posição da pilha:

   Código corrigido:
   ```
   for (int i = 1; i < TAMANHO_PILHA - 1; i++) {
       pilha[i] = pilha[i + 1];
   }
   pilha[TAMANHO_PILHA - 1] = 0; // Limpa o topo da pilha
   ```

3. OPERANDOS INSUFICIENTES:
4.

-Erro: O programa não verifica se há operandos suficientes na pilha antes de tentar realizar uma operação.

Código original:

```
resultado = pilha[1] / pilha[0];
 } else {
 printf("Operador invÃ¡lido!\n");
 return;
 }
```

-Correção: Adicionada uma verificação para garantir que haja operandos suficientes para a operação:

Código corrigido:

```
if (pilha[1] == 0 && operador != '/') {
    printf("Erro: Operandos insuficientes para a operação.\n");
    return;
}
```

5. SOBREPOSIÇÃO DE OPERANDOS NA FILA:

- Erro: Quando um novo valor era empurrado na pilha, não havia tratamento adequado para valores antigos, o que poderia causar sobreposição de dados.

Código original:

```
for (int i = TAMANHO_PILHA - 1; i > 0; i--) {
 pilha[i] = pilha[i - 1];
 }
```

- Correção: O loop de deslocamento ao empurrar valores na pilha foi ajustado para garantir que os elementos sejam movidos corretamente:

Código corrigido:

```
for (int i = TAMANHO_PILHA - 1; i > 0; i--) {
    pilha[i] = pilha[i - 1];
}
```

Programa Fonte Refatorado

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

#define TAMANHO_PILHA 4

// Função para exibir a pilha
void exibirPilha(int pilha[]) {
    printf("Pilha: [T: %d] [Z: %d] [Y: %d] [X: %d]\n", pilha[3],
pilha[2], pilha[1], pilha[0]);
}

// Função para empurrar valores na pilha
void empurrar(int pilha[], int valor) {
    for (int i = TAMANHO_PILHA - 1; i > 0; i--) {
        pilha[i] = pilha[i - 1];
    }
    pilha[0] = valor;
}

// Função para executar a operação entre os dois
primeiros operandos da pilha
void executarOperacao(int pilha[], char operador) {
    int resultado;

    // Verificar se há operandos suficientes
    if (pilha[1] == 0 && operador != '/') {
        printf("Erro: Operandos insuficientes para a
operação.\n");
        return;
    }

    // Operação entre o penúltimo (pilha[1]) e o último
(pilha[0]) valor
    if (operador == '+') {
        resultado = pilha[1] + pilha[0];
    } else if (operador == '-') {
```

```
        resultado = pilha[1] - pilha[0];
    } else if (operador == '*') {
        resultado = pilha[1] * pilha[0];
    } else if (operador == '/') {
        if (pilha[0] == 0) {
            printf("Erro: Divisão por zero não permitida.\n");
            return;
        }
        resultado = pilha[1] / pilha[0];
    } else {
        printf("Operador inválido!\n");
        return;
    }

    // Atualizar a pilha com o resultado da operação
    pilha[0] = resultado;
    for (int i = 1; i < TAMANHO_PILHA - 1; i++) {
        pilha[i] = pilha[i + 1]; // Shift nos valores para liberar
espaço
    }
    pilha[TAMANHO_PILHA - 1] = 0; // Limpa o topo da pilha
}

int main() {
    int pilha[TAMANHO_PILHA] = {0, 0, 0, 0}; // Inicializando
a pilha com zeros
    char entrada[100]; // Para armazenar a entrada do
usuário
    char continuar;

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

    do {
        printf("\nDigite a expressão em formato RPN (ex: 5 1
2 + 4 * + 3) ou 'sair' para encerrar: ");
        fgets(entrada, sizeof(entrada), stdin); // Lê a entrada
do usuário

        // Verificar se o usuário deseja sair
        if (strncmp(entrada, "sair", 4) == 0) {
```

```
            break;
        }

        // Dividir a entrada em tokens (números e
operadores)
        char *token = strtok(entrada, " ");
        while (token != NULL) {
            // Verifica se o token é um número (operando)
            if (isdigit(token[0]) || (token[0] == '-' &&
isdigit(token[1]))) {
                int valor = atoi(token); // Converte o token para
número inteiro
                empurrar(pilha, valor); // Empurra o número para
a pilha
                exibirPilha(pilha); // Exibe o estado da pilha
            }
            // Caso contrário, trata-se de um operador
            else {
                executarOperacao(pilha, token[0]); // Executa a
operação
                exibirPilha(pilha); // Exibe o estado da pilha
            }
            token = strtok(NULL, " "); // Avança para o próximo
token
        }

        printf("\nResultado final: %d\n", pilha[0]); // Exibe o
resultado final

        // Pergunta ao usuário se deseja realizar outra
operação
        printf("\nDeseja realizar outra operação? (s/n): ");
        scanf(" %c", &continuar);
        getchar(); // Limpa o buffer

    } while (continuar == 's' || continuar == 'S');

    // Mensagem de encerramento
    printf("Obrigado por usar nossa Calculadora Fatec-
HP12c!\n");
```

```
    return 0;
}
```