Estrutura de Dados I
Trabalho Prático

Data de entrega: 17/08/2025

1 Objetivo do Trabalho

Este trabalho tem por objetivo a implementação e a aplicação do Tipo Abstrato de Dado (TAD) Pilha para tipos genéricos de dados.

Para alcançar o objetivo, implemente o TAD Pilha para armazenar tipos genéricos de dados nos arquivos Pilha.h e Pilha.c

Após a implementação da biblioteca, implemente, para cada problema a seguir, um programa para atender o que é solicitado. Ou seja, para cada problema, um programa deve ser implementado. Este programa deve incluir a biblioteca Pilha.h e utilizar as funções dessa biblioteca.

OBSERVAÇÕES:

- Caso a TAD Pilha não seja para tipo genérico de dado, a nota do trabalho será **zero**.
- Caso a biblioteca não seja utilizada no programa que resolve o problema, a nota referente ao problema em questão será **zero**.

Problema 1

Em expressões matemáticas, a ordem de resolução segue um padrão específico, conhecido como ordem das operações, para garantir que o resultado seja sempre o mesmo, independentemente de quem está calculando. A ordem das operações devem seguir a seguinte ordem:

1. **Parênteses/Colchetes/Chaves:** Operações dentro de parênteses ( ) são resolvidas primeiro, seguidas por colchetes e, por último, chaves . Dentro de cada um desses símbolos, a ordem de resolução segue os próximos passos.
2. **Expoentes/Radicais:** Potenciação e radiciação são calculadas após a resolução das operações dentro de parênteses, colchetes e chaves.
3. **Multiplicação e Divisão:** São realizadas na ordem em que aparecem, da esquerda para a direita.
4. **Adição e Subtração:** São realizadas na ordem em que aparecem, também da esquerda para a direita.

Implemente um programa que faça validação de expressões matemáticas digitadas pelo usuário. Utilize a implementação do **TAD Pilha para tipo genérico de dado** solicitada no **Objetivo do Trabalho**. As expressões devem conter literais de A a J, operadores (+ (adição), - (subtração), / (divisão),* (multiplicação), ^ (potenciação)) e delimitadores de escopo ("(", ")", "[", "]", "{", "}"). O programa deve prever 2 tipos de validação:

a) Expressões contendo parênteses, colchetes e chaves, apenas verificando se cada delimitador aberto possui seu fechamento realizado corretamente.

b) Expressões contendo parênteses, colchetes e chaves, considerando a precedência entre os delimitadores: os escopos devem ser abertos e fechados corretamente e deve-se verificar se os escopos dos colchetes abrangem os parênteses e os escopos das chaves abrangem os colchetes.

Por exemplo:

- a expressão ((A+D)) *J) não é validada em nenhuma das opções.
- a expressão [({A+D}/B)*J] é validada apenas pela opção (a).
- a expressão {[(A+D)/B]*J} é validada pela opções (a) e (b).

Problema 2

Em notação matemática, uma expressão **infixa** é aquela onde o operador está entre os operandos (ex: $a + b$), enquanto uma expressão **posfixa** (ou notação polonesa reversa) coloca o operador após os operandos (ex: $ab+$).

Expressões Infixas

- A forma mais comum de escrever expressões matemáticas, onde o operador aparece entre os dois operandos.
- Exemplo: $(a + b) * c$
- Requer o uso de parênteses para indicar a ordem das operações, caso contrário, a precedência dos operadores entra em jogo (multiplicação antes da adição, por exemplo).

Expressões Posfixas

- O operador segue os operandos.
- Exemplo: xy+w* (equivalente a (x + y) * w)
- Não necessita de parênteses para definir a ordem das operações, pois a estrutura da expressão já indica qual operação realizar primeiro.

Implemente um programa para manipulação de expressões matemáticas envolvendo variáveis literais de A a J, operadores (+ (adição), - (subtração), / (divisão),* (multiplicação), ^ (potenciação)) e os delimitadores de escopo tipo parênteses ("(", ")"). Utilize a implementação do **TAD Pilha para tipo genérico de dado** solicitada no **Objetivo do Trabalho**.

Para tal, o programa deve ter as seguintes funcionalidades:

1. Entrada de expressões: o usuário deve optar entre entrar com expressões em 2 formatos: a) forma Posfixa, b) forma Infixa com uso de parênteses eventuais (quando necessário, para modificar a precedência da operação). Se o usuário optar pelo formato infixo o programa deverá realizar a conversão da expressão para a forma pós-fixa e imprimir a expressão resultante da conversão.

2. Entrada dos valores das literais: o usuário deve associar os valores a todas as literais de A a J.

3. Avaliação da expressão: o programa deve avaliar a expressão digitada pelo usuário (após a conversão para a forma pós-fixa, se necessário), associando os valores das literais e imprimindo o resultado da expressão.

Problema 3

Dada uma planta de uma casa, e sua tarefa é contar o número de cômodos que ela possui. A planta é representada por uma matriz de $n \times m$ quadrados, e cada quadrado é ou um piso ou uma parede. Você pode se mover para a esquerda, direita, para cima e para baixo através dos quadrados de piso. Utilize a implementação do **TAD Pilha para tipo genérico de dado** solicitada no **Objetivo do Trabalho**.

Entrada

A primeira linha da entrada contém dois inteiros n e m: a altura e a largura do mapa, tal que $1 \leq n, m \leq 1000$. Em seguida, há n linhas com m caracteres descrevendo o mapa. Cada caractere é . (piso) ou # (parede).

Saída

Imprima um único inteiro: o número de cômodos.

Example

Entrada	Saída:
5 8 ######## #..#...# ####.#.# #..#...# ########	3

2 Códigos e Compilação dos Programas

Para cada problema acima, implemente o programa em linguagem C, nomeando os aquivos referentes a cada problema com o nome "programaN.c", onde "N" deve ser trocado pelo número do problema.

Para compilar o programa, implemente o arquivo *makefile* com os alvos:

- programa1 (para compilar o código referente ao programa que resolve o problema 1)
- programa2 (para compilar o código referente ao programa que resolve o problema 2)

- programa3 (para compilar o código referente ao programa que resolve o problema 3)

3 Grupos

O trabalho deverá ser realizado em grupos de, no máximo, 4 alunos. Todos os integrantes do grupo devem **estar matriculados no mesmo horário de laboratório**. Não será aceita como justificativa para a ausência na apresentação do trabalho o fato de haver membros do grupo matriculados em horários diferentes. Nesses casos, a nota do trabalho será **ZERO**.

4 Entrega do trabalho

Todos os arquivos (Pilha.h, Pilha.c, programa1.c, programa2.c, programa3.c e makefile) devem estar em uma única pasta. Esta deverá ser compactada em um único arquivo com o nome ED1_Grupo<*número do grupo*>_[nomeDoAluno1][nomeDoAluno2][nomeDoAluno3].zip, onde <*número do grupo*> deve ser substituído pelo número do grupo definido na lista de grupos..

A avaliação do trabalho será realizado no sistema Linux. Logo, caso o programa implementado não compile no Linux a nota será **zero** (compilação e execução em linha de comando utilizando o programa **make**).

A primeira página do relatório, assim como **cada arquivo de código do programa**, devem conter o número do grupo e os nomes dos componentes do grupo.

Além dos arquivos de código fonte, os grupos deverão escrever um **relatório** com a documentação do trabalho. O trabalho deverá ser entregue até o dia 17/08/2025 às **18:00** hs no sistema ava.ufes.br.

Atenção: O sistema encerrará o envio dos arquivos às **18:00**. Por isso, envie o trabalho **ANTES** desse horário. Os trabalhos enviados fora do sistema AVA receberá nota **ZERO**.

5 Apresentação:

As apresentações dos trabalhos ocorrerão no dia 18/08/25 no laboratório de informática (2º piso do prédio de salas de aula). Os grupos devem preparar *slides* e realizar a apresentação de forma mais didática possível. A qualidade da apresentação será considerada na avaliação do trabalho.

A não apresentação do trabalho resultará em nota **ZERO** para todo o trabalho.