INSTITUTO FEDERAL
Catarinense
Campus Blumenau

Estruturas de Dados

Bacharelado em Ciência da Computação
Prof. Dr. Paulo César Rodacki Gomes

Lista de exercícios-ED-05
Pilhas

Estes exercícios devem ser entregues no Google Classroom. O objetivo desta lista é implementar uma estrutura de dados de pilha baseada em listas para armazenar valores reais. Além disso, vamos usar esta pilha em uma aplicação para implementar uma calculadora pós-fixada.

Questões:

1. **Implementação da pilha.** Inicialmente crie os seguintes arquivos fonte:
    - **principal.c**: função main
    - **pilha.h**: declarações de estruturas, tipo e funções da pilha
    - **pilha.c**: implementação das funções da pilha
    - **lista.h**: declarações de estruturas, tipo e funções da lista
    - **lista.c**: implementação das funções da lista

2. **Estruturas e tipos de dados.** Inicialmente crie os seguintes arquivos fonte:

```
struct noLista {
        float info;
        struct noLista *prox;
};

typedef struct noLista NoLista;

struct pilha {
        NoLista *topo;
};

typedef struct pilha Pilha;
```

3. **Protótipos das funções a serem implementadas.**

- `Pilha *pilha_cria(void);` instancia uma nova struct Pilha, criando uma pilha vazia e inicializando o primeiro elemento (o topo da pilha) como `NULL`.

- `void pilha_push(Pilha *p, float v);` instancia um novo nó da lista e o insere no topo da pilha. Note que o ponteiro para o topo da pilha sempre deve ser atualizado.

- `float pilha_pop(Pilha *p);` retira o elemento do topo da pilha e retorna seu valor. Se a pilha estiver vazia, a função deve exibir uma mensagem de erro e abortar o programa.

- `int pilha_vazia(Pilha *p);` a pilha está vazia se a lista estiver vazia.

- `void pilha_libera(Pilha *p);` libera a pilha desalocando (liberando) todos os nós da lista.

4. **Implementação da calculadora pós fixada.** Implemente na sua função main uma calculadora pós-ficada de números reais, com as 4 operações aritméticas básicas (+, -, * e /). A calculadora funciona da seguinte maneira: cada operando é empilhado na pilha de valores. Quando se encontra um operador, desempilha-se o número apropriado de operandos, realiza-se a operação devida e empilha-se o resultado.

   Nossa calculadora vai ser um struct com dois campos: uma pilha e uma string para formatação dos números da calculadora, da seguinte forma:

```
struct calc {
   charf[21]; /* formato para impressão, por exemplo "%.2f\n" */
   Pilha *p;  /* pilha de operandos */
};

typedef struct calc Calc;
```

   Funções a serem implementadas pela calculadora:

- `Calc calc_cria(char* f);` instancia uma nova calculadora com formatação de números definida pela cadeia f.
- `void calc_operando(Calc* c, float v);` coloca o operando passado como valor no topo da pilha.
- `void calc_operador(Calc* c, char op);` retira dois valores do topo da pilha (considerando que só estamos usando operadores binários), efetua a operação correspondente e coloca o valor do resultado na pilha. As operações que voce deve implementar são +, -, * e /. Se não existirem operandos na pilha, consideramos que seus valores são zero.
- tanto a operação operando quanto a operação operador imprimem, com a utilização do formato especificado na criação da pilha, o novo valor do topo da pilha
- `void calc_libera(Calc *c);` libera a memória usada pela calculadora (incluindo a Pilha).

5. **Implemente um programa cliente que usa a calculadora**. Voce pode criar um laço do tipo do…while para receber operandos e operadores, com o recebimento da opção "q" (de quit) para encerrar a entrada de dados e liberar a calculadora.

   Exemplo: criando a calculadora com string de formatação "%.2f\n",
   se o usuário digitar: `3 5 8 * + 7 / q`, teríamos a seguinte saída:

```
3.00
5.00
8.00
40.00
43.00
7.00
6.14
```