Listas estáticas

Objetivo da aula

Estudar a estrutura lista como um tipo abstrato de dados com exemplos de aplicação de acordo com a abordagem estática.

Introdução

Listas encadeadas são estruturas de dados que representam uma sequência de elementos. A implementação dessa estrutura pode seguir uma abordagem estática ou dinâmica.

Na abordagem estática, um vetor é utilizado para representar a lista. Na abordagem dinâmica, ponteiros são utilizados para localizar os elementos subsequentes na lista.

Operações principais de lista:

- Inserir elemento na lista (no início, no final)
- Buscar um elemento na lista
- Remover um elemento da lista

Abordagem estática

A abordagem estática utiliza um vetor para a implementação da lista. Utilizaremos um tipo abstrato de dado vetor.

Instruções

1. Crie um novo projeto.
   - Dev C/C++ ou Code::Blocks: File → New… → Project… :
   - Nomeie o projeto como, por exemplo, “Lista_Estatica”.
   - Será criado um arquivo main() contendo a função `int main()`. Deixe-o de lado por enquanto.
2. Criando o cabeçalho: clique com o botão direito sobre o projeto e crie um novo arquivo (New → File). Nomeie-o como `lista` e salve como arquivo tipo header.
3. Copie e cole o código a seguir com as **declarações** das funções:

```
#ifndef LISTA_H_INCLUDED
#define LISTA_H_INCLUDED

//tipo exportado
typedef struct tipo_lista lista;

//funcoes exportadas
lista* criar(int tam);
void liberar(lista* L); //libera a memoria ocupada pela lista
void inicializar(lista* L);
void inserir_inicio(lista* L, int valor);
void mostrar(lista* L);
void inserir_fim(lista* L, int valor);
#endif
```

4. Criando a implementação (**definição**) das funções do cabeçalho:

```
#include "lista.h"
#include <stdio.h>
#include <stdlib.h>

struct tipo_lista{
      int tamanho;
      int index; //indica próxima posicao disponível no final da lista
      int *vetor;
};

lista* criar (int tam){
      lista *temp = (lista*) malloc(sizeof(lista));
      temp -> vetor = (int*) malloc(tam*sizeof(int));
      temp -> tamanho = tam;
      temp -> index = 0;
      return temp;
}

void liberar(lista* L){
      free(L->vetor);
      free(L);
}

void inicializar(lista* L){
      int i;
      for (i=0; i<L->tamanho; i++){
            L->vetor[i] = 0;
      }
}

int lista_cheia(lista* L){
      if (L->index == L->tamanho){
            return 1;
      }
      else{
            return 0;
      }
}

void inserir_inicio(lista* L, int valor){
      int i;
      if(!lista_cheia(L)){
            for(i=L->index; i>0; i--){
                  L->vetor[i] = L->vetor[i-1];
            }
            L->vetor[0] = valor;
            L->index++;
      }
      else{
            printf("Lista cheia!");
      }
}
```

```
void mostrar(lista* L){
	int i;
	for(i=0; i<L->tamanho; i++){
		printf("%d ", L->vetor[i]);
	}
}
```

5. Voltando para o arquivo `main.c` (salve ou renomeie como `lista_estatica_main.c`), vamos criar o programa que utiliza o modelo TAD lista estática criado anteriormente. Copie e cole o código abaixo:

```
#include "lista.h"
#include <stdio.h>
#include <stdlib.h>

int const TAM = 5;
int menu(){
	int opcao;
	printf("\nOpcoes:");
	printf("\n0. Sair");
	printf("\n1. Inserir um numero no inicio da lista");
	printf("\n2. Mostrar a lista");
	printf("\nInforme a opcao: ");
	scanf("%d", &opcao);
	return opcao;
}
int main() {
	int opcao, elemento;
	lista *L;
	L = criar(TAM);
	inicializar(L);
	do{
		opcao = menu();
		switch (opcao){
			case 1:	printf("\nInforme o valor: ");
				scanf("%d", &elemento);
				inserir_inicio (L, elemento);
				break;
			case 2:	printf("\nValores armazenados no vetor:\n");
				mostrar(L);
				break;
		}
	}while(opcao);
	liberar(L);
	return 0;
}
```

6. Compile e execute o programa. Observe a saída para compreender o funcionamento do programa.

- **Importante:** siga a seguinte ordem de compilação:
  - 1º: lista_estatica.c
  - 2º: lista_estatica_main.c

Agora é com você:

7. Implemente o procedimento para inserir um valor no final da lista.
8. Faça uma função que retorne a posição de determinado valor na lista, caso ele exista.
9. Faça um procedimento para inserir em qualquer posição da lista. Deve ser verificado se a posição para inserir é válida.
10. Faça um procedimento para excluir um valor da lista. Dado um valor, este valor deve ser excluído, caso exista na lista.
    - Lembre-se que ao excluir, os elementos devem ser realocados para que permaneçam em posições contíguas.