Estruturas de Dados II

Professor: Francisco Assis da Silva

Tipo Abstrato de Dados (TAD)

É definido com um modelo matemático pelo par (v,o) onde v é um conjunto de valores e o é um conjunto de operações sobre esses valores.

Ex: Tipo real associado aos seus valores e operações válidas. A característica essencial de um TAD é a separação entre conceito e implementação. A separação da definição do TAD de sua implementação permite que a mudança de implementação não altere o programa que usa o TAD. O TAD é compilado separadamente, e uma mudança força somente a compilação, de novo, do módulo envolvido. Ex: TAD dicionário Inglês-Português (v,o):

onde: v: conjunto de pares de palavras; o: operação: inserção de um novo par, remoção, consulta, alteração.

Exercícios:

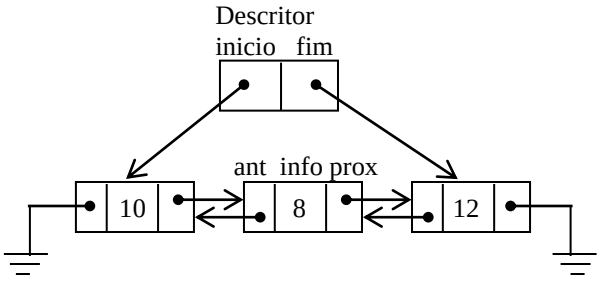
**a) Implementar o TAD Pilha** Init(P): inicializa a pilha P no estado "vazia"; Top(P): acessa o elemento posicionado no topo da pilha P; Push(P,x): insere um novo elemento x no topo da pilha P; Pop(P,x): remove um elemento x do topo da pilha P; IsEmpty(P): verifica se a pilha P está vazia.

**b) Implementar o TAD Fila** Init(F): inicializa a fila F no estado "vazia"; Enqueue(F,x): insere um elemento x no final da fila F; Dequeue(F,x): remove um elemento x do começo da fila F. QisEmpty(F): verifica se a fila F está vazia.

**c) Implementação TAD – String Dinâmica**

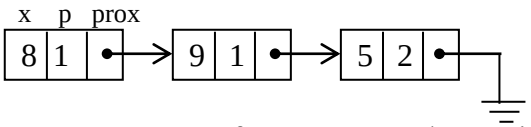
1) Cria uma string vazia, deve ser usada antes de qualquer outra operação;
2) Reinicia uma string existente, através da remoção de todos os elementos contidos nela;
3) Exibe uma string dinâmica;
4) Exibe uma string dinâmica invertida;
5) Função que retorna o tamanho de uma string dinâmica;
6) Insere um determinado caracter **`c`** na string **`str1`**;
7) Copia o conteúdo da string **`str1`** para a string **`str2`**;
8) Concatena a string **`str1`** com a string **`str2`** armazenando o resultado na string **`str3`** utilizando a função copia;
9) Remove da string **`str1`**, a quantidade de caracteres especificado por **`nro`** a partir da posição **`start`**;
10) Insere na string **`str1`** uma substring **`subs`** a partir da posição **`start`**;
11) Função que verifica se uma string **`str1`** é menor que outra **`str2`**, caso verdade a função retorna **True**, senão **False**;
12) Função que verifica se uma string (str1) é igual a outra (str2), caso verdade a função retorna True, senão False;
13) Busca a posição **`local`** na string **`str1`** em que a string **`subs`** se inicia. Se **`local`** = 0 então **`subs`** não está contida em **`str1`**, caso contrário **`local`** é a posição de início da string **`subs`** dentro da string **`str1`**.

**d) Implementar o TAD Lista Duplamente Encadeada**
Estrutura de dados:




→ Init(LD): inicializa a lista duplamente encadeada **LD** no estado "vazia";
→ InserirInicio(LD, x): insere o elemento **x** no início da lista duplamente encadeada **LD**;
→ InserirFinal(LD, x): insere o elemento **x** no final da lista duplamente encadeada **LD**;
→ Busca(LD, x): retorna o endereço do nó da lista **LD** encontrado referente ao elemento **x**, caso contrário, não encontre o elemento **x**, retorna NULL;
→ Exclui(LD, x): exclui um elemento **x** lista duplamente encadeada **LD** (usa a busca para localizar o endereço do nó a ser excluído);
→ isEmpty(LD): verifica se a lista duplamente encadeada **LD** está vazia.

**e) Implementar o TAD Fila com prioridade**
Estrutura de dados:




→ Init(FP): inicializa a fila **FP** no estado "vazia";
→ Enqueue(FP, x, p): insere um elemento **x** na fila **FP** obedecendo a prioridade (1, 2, 3, .., n);
→ Dequeue(FP, x, p): remove um elemento **x** de prioridade **p** do começo da fila **FP**;
→ isEmpty(FP): verifica se a fila **FP** está vazia.