Estruturas de Dados
Aula 11: TAD Pilha

10/05/2010

Fontes Bibliográficas

- Livros:
  - Projeto de Algoritmos (Nivio Ziviani): Capítulo 3;
  - Introdução a Estruturas de Dados (Celes, Cerqueira e Rangel): Capítulo 10;
  - Estruturas de Dados e seus Algoritmos (Szwarefiter, et. al): Capítulo 2;
  - Algorithms in C (Sedgewick): Capítulo 3;
- Slides baseados nas transparências disponíveis em: http://www.dcc.ufmg.br/algoritmos/transparencias.php

Pilhas

- É uma lista linear em que todas as inserções, retiradas e, geralmente, todos os acessos são feitos em apenas um extremo da lista.
- Os itens são colocados um sobre o outro. O item inserido mais recentemente está no topo e o inserido menos recentemente no fundo.
- O modelo intuitivo é o de um monte de pratos em uma prateleira, sendo conveniente retirar ou adicionar pratos na parte superior.

Propriedades e Aplicações das Pilhas

- Propriedade: o último item inserido é o primeiro item que pode ser retirado da lista. São chamadas listas **lifo** ("last-in, first-out").
- Existe uma ordem linear para pilhas, do "mais recente para o menos recente".
- É ideal para processamento de estruturas aninhadas de profundidade imprevisível.
- Uma pilha contém uma seqüência de obrigações adiadas. A ordem de remoção garante que as estruturas mais internas serão processadas antes das mais externas.

Propriedades e Aplicações das Pilhas (2)

- Aplicações em estruturas aninhadas:
  - Quando é necessário caminhar em um conjunto de dados e guardar uma lista de coisas a fazer posteriormente.
  - O controle de seqüências de chamadas de subprogramas.
  - A sintaxe de expressões aritméticas.
- As pilhas ocorrem em estruturas de natureza recursiva (como árvores). Elas são utilizadas para implementar a **recursividade**.

TAD Pilha

- Conjunto de operações:
  - FPVazia (Pilha). Faz a pilha ficar vazia.
  - Vazia (Pilha). Retorna *true* se a pilha estiver vazia; caso contrário, retorna *false*.
  - Empilha (x, Pilha). Insere o item x no topo da pilha. (operação *push*)
  - Desempilha (Pilha, x). Retorna o item x no topo da pilha, retirando-o da pilha. (operação *pop*)
  - Tamanho (Pilha). Esta função retorna o número de itens da pilha.

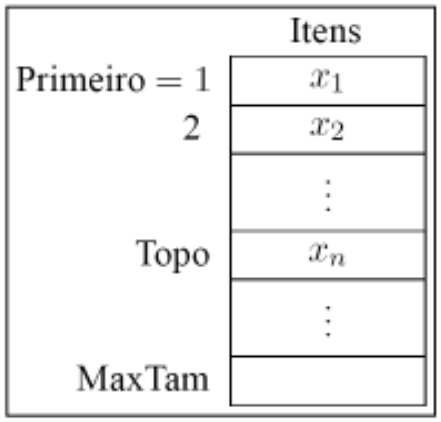
Implementação do TAD Pilha

- Existem várias opções de estruturas de dados que podem ser usadas para representar pilhas.
- As duas representações mais utilizadas são as implementações por meio de *vetores* e de *estruturas encadeadas*.

Pilhas em Alocação Seqüencial e Estática

- Os itens da pilha são armazenados em posições contíguas de memória.
- Como as inserções e as retiradas ocorrem no topo da pilha, um cursor chamado Topo é utilizado para controlar a posição do item no topo da pilha.

Pilhas em Alocação Seqüencial e Estática (2)



Estrutura de Pilhas com Alocação Seqüencial e Estática

- Os itens são armazenados em um **vetor** de tamanho suficiente para conter a pilha.
- O outro campo do mesmo registro contém o índice do item no topo da pilha.
- A constante MaxTam define o tamanho máximo permitido para a pilha.

Estrutura de Pilhas com Alocação Seqüencial e Estática (2)

```
typedef struct tipoitem TipoItem;
typedef struct tipopilha TipoPilha;

TipoPilha* InicializaPilha();
void FPVazia(TipoPilha *Pilha);
int Vazia (TipoPilha* Pilha);
void Empilha (TipoItem* x, TipoPilha* Pilha);
void Desempilha (TipoPilha* Pilha, TipoItem*
  Item);
int Tamanho (TipoPilha* Pilha);
TipoItem* InicializaTipoItem (int n);
void Imprime (TipoPilha* pilha);
```

Implementação TAD Pilha com Vetores

```
#include <stdio.h>
#include <stdlib.h>
#include "Pilha.h"
#define MaxTam  1000

struct tipoitem {
  int valor;
  /* outros componentes */
};

struct tipopilha {
  TipoItem Item[MaxTam];
  int Topo;
};
```

Implementação TAD Pilha com Vetores

```
TipoPilha* InicializaPilha(){
  TipoPilha* pilha
  =(TipoPilha*)malloc(sizeof(TipoPilha));
  return pilha;
}

void FPVazia(TipoPilha* Pilha){
  Pilha->Topo = 0;
}

int Vazia (TipoPilha* Pilha) {
  return (Pilha->Topo == 0);
}
```

Implementação TAD Pilha com Vetores

```
void Empilha (TipoItem* x, TipoPilha*
  Pilha) {
  if (Pilha->Topo == MaxTam)
     printf ("Erro: pilha esta cheia\n");
  else {
    Pilha->Topo++;
    Pilha->Item[Pilha->Topo - 1] = *x;
  }
}
```

Implementação TAD Pilha com Vetores (2)

```
void Desempilha (TipoPilha* Pilha,
  TipoItem* Item) {
  if (Vazia (Pilha))
     printf ("Erro: pilha esta vazia\n");
  else {
     *Item = Pilha->Item[Pilha->Topo-1];
     Pilha->Topo--;
  }
}
int Tamanho (TipoPilha* Pilha) {
  return (Pilha->Topo);
}
```

Implementação TAD Pilha com Vetores (2)

```
TipoItem* InicializaTipoItem (int n)
{
   TipoItem* item = (TipoItem*)malloc(sizeof(TipoItem));
   item->valor = n;
   return item;
}

/*Imprime os itens da pilha */
void Imprime (TipoPilha* pilha)
{
   int Aux;
   printf ("Imprime Pilha Estatica: \n");

   for (Aux = 0; Aux < pilha->Topo; Aux++)
   {
       printf ("%d\n", pilha->Item[Aux].valor);
   }
}
```