LABORATÓRIO DE ALGORITMOS E ESTRUTURAS DE DADOS I

Prof. Caio César de Freitas Dantas

Pilhas

Pilhas em Alocação Sequencial e Estática (Vetor).

- 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

Pilhas em Alocação Sequencial e Estática (Vetor).

- 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.

Pilhas

Implementação TAD Pilha com Vetores.

Definindo a Pilha.

```c
#define MaxTam 100
typedef struct tipoitem TipoItem;
typedef struct tipopilha TipoPilha;

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

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

Pilhas

Implementação TAD Pilha com Vetores.

Criando a Pilha.

```
Pilha* cria (void){
 TipoPilha* p = (TipoPilha*) malloc(sizeof(TipoPilha));
 p->Topo = 0; /* inicializa com zero elementos */
 return p;
}
```

Pilhas

Implementação TAD Pilha com Vetores.

Inserindo elemento a Pilha.

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

Pilhas

Implementação TAD Pilha com Vetores.

Removendo elemento a Pilha.

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

Pilhas

Implementação TAD Pilha com Vetores.

Tamanho da Pilha.

```
int Tamanho (TipoPilha* p) {
    return (p->Topo);
}
```

Pilhas

Implementação TAD Pilha com Vetores.
A função que verifica se a pilha está vazia pode ser dada por:

```
int vazia (Pilha* p){
    return (p->Topo == 0);
}
```

A função para liberar a memória alocada pela pilha pode ser:

```
void libera (Pilha* p){
    free(p);
}
```

Pilhas

Implementação TAD Pilha com Vetores.

Exibir Pilha.

```
void imprimir (){
    int i;
    if (p->Topo == 0){
        printf ("Pilha vazia\n");
        return;
    }
    else{
        printf ("\n Elementos na pilha: \n");
        for (i = p->top; i >= 0; i--){
            printf ("%d\n",  p->Item[i].valor);
        }
    }
}
```

Pilhas

Implementação TAD Pilha com Listas (nós, células).

- Há uma célula cabeça no topo para facilitar a implementação das operações empilha e desempilha quando a pilha estiver vazia.
- Para desempilhar o item xn basta desligar a célula cabeça da lista e a célula que contém xn passa a ser a célula cabeça.
- Para empilhar um novo item, basta fazer a operação contrária, criando uma nova célula cabeça e colocando o novo item na antiga.

Pilhas

Implementação TAD Pilha com Listas (nós, células).

- Cada célula de uma pilha contém um item da pilha e um ponteiro para outra célula.
- O registro TipoPilha contém um ponteiro para o topo da pilha (célula cabeça) e um ponteiro para o fundo da pilha.

Pilhas

Implementação TAD Pilha com Listas

- Definição da pilha

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

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

typedef struct celula_pilha {
TipoItem Item;
struct celula_pilha* Prox;
} Celula;

struct tipopilha {
Celula* Fundo, Topo;
int Tamanho;
};
```

Pilhas

Implementação TAD Pilha com Listas

- Inicialização da pilha

```
TipoPilha* InicializaPilha(){
TipoPilha* p = (TipoPilha*)malloc(sizeof(TipoPilha));
p->Fundo = p->Topo;
p->Topo->Prox = NULL;
P->Tamanho = 0;
return p;
}
```

Pilhas

Implementação TAD Pilha com Listas

- Inserindo elemento na pilha

```
void Empilha (TipoItem* x, TipoPilha *p){
Celula* Aux;
Aux = (Celula*) malloc(sizeof(Celula));
p->Topo->Item = *x;
Aux->Prox = p->Topo;
p->Topo = Aux;
p->Tamanho++;
}
```

Pilhas

Implementação TAD Pilha com Listas

- Removendo elemento da pilha

```
void Desempilha (TipoPilha *p, TipoItem *Item){
Celula* q;
if (Vazia (Pilha)) {
printf ("Erro: lista vazia \n");
return;
}
q = p->Topo;
p->Topo = q->Prox;
*Item = q->Prox->Item;
free (q);
p->Tamanho--;
}
```

Pilhas

Implementação TAD Pilha com Listas

- Exibir pilha

```
void imprime (TipoPilha* p){
 Celula* q;
 for (q=p->Topo; q!=NULL; q=q->Prox)
 printf("%f\n",q->Item);
}
```

Pilhas

Implementação TAD Pilha com Listas

- Exibir pilha

```
void Imprime (TipoPilha* pilha){
Celula* Aux;
Aux = pilha->Topo->Prox;
printf ("Imprime Pilha Encadeada: \n");
while (Aux != NULL){
printf ("%d\n", Aux->Item.valor);
Aux = Aux->Prox;
}
}
```

FIM!