Pilhas e Filas

Conceitos e Implementação em linguagem C

Maria Adriana Vidigal de Lima

Março - 2009

Organização dos dados

Em determinadas aplicações, as **pilhas** e **filas** representam estruturas de dados importantes, nas quais seus elementos são organizados em função de um critério que regulamenta a entrada e a saída dos elementos.

Para uma **pilha**, tem-se o critério LIFO:

LIFO: Last In, First Out - O último elemento a entrar deve ser o primeiro a ser retirado.

Para uma **fila**, tem-se o critério LIFO:

FIFO: First In, First Out - O primeiro elemento a entrar deve ser o primeiro a ser retirado.

Organização dos dados

As operações básicas a serem implementadas para uma estrutura de **pilha** são:

- Iniciar a pilha
- Verificar se a pilha está vazia
- Retornar o elemento que está no topo da pilha
- Inserir um elemento na pilha (no topo)
- Retirar um elemento da pilha (do topo)

Organização dos dados

As operações básicas a serem implementadas para uma estrutura de **fila** são:

- Iniciar a fila
- Verificar se a fila está vazia
- Inserir um elemento na pilha (ao final)
- Retirar um elemento da pilha (do início)

Definição da Estrutura de Pilha

A estrutura de dados a ser declarada para o tipo **Pilha** pode ser definida por meio de ponteiros, como no exemplo abaixo na linguagem C:

```
struct No {
  int Valor;
  struct No *prox; }
typeDef struct No TipoNo;

struct Pilha {
  TipoNo *topo;
  int tamanho; }
typedef struct Pilha TipoPilha;
```

Definição da função *IniciaPilha*

A função define uma pilha vazia, sendo o campo **topo** associado ao ponteiro nulo, e o tamanho da pilha iniciado com o valor 0.

```
void IniciaPilha(TipoPilha *pilha)
{
  pilha->topo = NULL;
  pilha->tamanho = 0;
}
```

Definição da função *Vazia*

A função recebe o parâmetro (ponteiro) pilha, que dá acesso à mesma, e retorna verdadeiro caso não haja elementos na pilha. Neste caso o topo da pilha é o ponteiro nulo.

```
int Vazia(TipoPilha *pilha)
{
  return (pilha->topo == NULL);
}
```

Definição da função *Empilha*

A função recebe como parâmetros o valor a ser empilhado, e um ponteiro para a pilha. Uma nova estrutura *TipoNo* deve ser alocada para que o novo valor seja armazenado e encadeado.

```
void Empilha(int x, TipoPilha *pilha)
{
  TipoNo *aux;

  aux = (TipoNo *) malloc(sizeof(TipoNo));
  aux->valor = x;
  aux->prox = pilha->topo;
  pilha->topo = aux;
  pilha->tamanho++;
}
```

Definição da função *Desempilha*

A função recebe como parâmetro um ponteiro para a pilha e remove o valor que estava no topo da pilha. Este valor é retornado à unidade chamadora.

```
int Desempilha(TipoPilha *pilha){
  TipoNo *q;  int v;
  if (Vazia(pilha)) {
    printf("Lista vazia\n");   return 0;
  }
  q = pilha->topo;
  pilha->topo = q->prox;
  v = q->valor;
  free(q);
  pilha->tamanho--;
  return v;
}
```

Definição da função *Main*

```
int main() {
  int i, numero, max=5;
  TipoPilha *pilha;

  pilha = (TipoPilha *) malloc (sizeof(TipoPilha));
  IniciaPilha(pilha);

  for (i=0;i<max;i++) {
      printf("Leitura do valor (%d) :",i); scanf("%d",&numero);
      Empilha(numero, pilha);
      printf("Empilhou: %d \n", numero);
  }

  for(i=0;i<max;i++) {
      numero = Desempilha (pilha);
      printf ("Desempilhou: %d \n", numero);
  }}
```