TAD Pilha (LIFO – Last in, First out)

Uma pilha é uma coleção ordenada de itens na qual os itens podem ser inseridos e retirados a partir da última posição, chamada *topo* da pilha. A Figura 1 representa uma estrutura do tipo pilha.

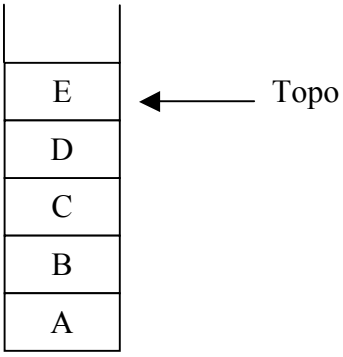


Figura 1: Representação de uma Pilha.

A definição de um TAD Pilha indica que apenas três operações podem ser realizadas: empilhamento, desempilhamento e uma eventual verificação do elemento que está no topo da pilha. A representação anterior ilustra uma pilha cujo conteúdo do topo é o elemento *E*. Neste caso, quando um elemento é inserido, ele é inserido "sobre" *E*. A remoção de um elemento na pilha anterior eliminaria o elemento *E* da pilha.

Implementação de Pilha

Pode-se criar um TAD pilha utilizando a representação em array. A seguir é descrita a implementação de uma pilha. A definição de uma estrutura para pilha necessita de apenas dois membros, um campo chamado *topo* e um campo onde as informações são armazenadas representado por um array chamado *dado*. O campo *topo* é do tipo inteiro e representa o índice do topo na array. Para linguagens onde não há como determinar facilmente o tamanho do array, é recomendável também manter a quantidade máxima de elementos que a pilha pode conter (MAX).

```
Estrutura Pilha{
 int topo = -1;
 int dado[];
 int MAX;
} p;
```

Sendo MAX a capacidade máxima da Pilha (tamanho do array). Inicialmente o índice topo é inicializado com –1. Isto indica que a pilha está vazia. Desta forma pode-se implementar um teste para verificar se a pilha está vazia (teste de underflow) da seguinte forma:

```
Se (topo == -1) então
        retorne pilha vazia
```

```
senão
        retorne pilha não vazia
```

A operação de inserção (empilhamento) pode ser facilmente implementada da seguinte forma:

```
empilha(p, 10) // Empilha o elemento 10 na pilha p
  Se p não está cheia então
        topo = topo +1;
        p.dado[topo] = 10
  senão
        "Informe Pilha Cheia."
```

É importante verificar a realização do teste de "overflow" antes de efetivamente inserir o elemento na pilha. Este teste pode ser implementado da seguinte forma:

```
Se p.topo == tamanho de p.dado – 1 então
        Retorne "pilha cheia"
Senão
        Retorne "Pilha não Cheia"
```

A operação de remoção (desempilhamento) de um elemento também é muito simples:

```
desempilha(p)   // Desempilha o elemento da Pilha p
        Se a p não está vazia então
                Retorne p.dado[topo]
                topo = topo – 1
        Senão
                Informe "Pilha está vazia!"
```

Exemplo 1

Um exemplo de utilização de pilha consiste em avaliar uma expressão aritmética segundo a utilização dos parênteses "()", colchetes "[]" e chaves "{}". O problema consiste em criar uma pilha de caracteres. O programa recebe uma sequência de caracteres que representa uma expressão aritmética genérica, por exemplo: {A*(A+B)}. A expressão é lida caracter a caracter da esquerda para direita. Quando um caracter de abertura é encontrado "(", "[", ou "{" esse caracter é empilhado. Quando um caracter de fechamento é encontrado ")",. "]"ou "}" o elemento do topo da pilha é comparado. Se o elemento do topo da pilha representa a abertura do respectivo fechamento, então o elemento é desempilhado. Quando o final da expressão é obtido, se a pilha está vazia, então a expressão está correta, caso contrário a expressão está incorreta (Figura 2).

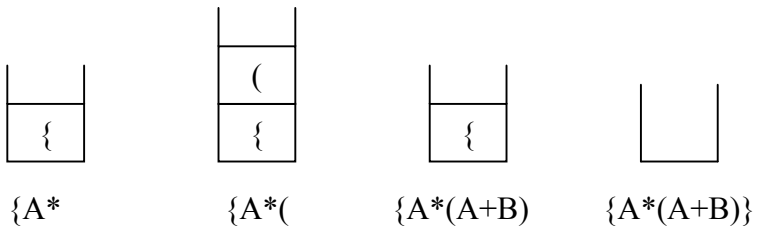
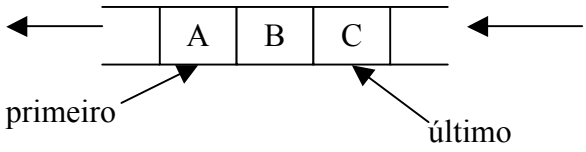


Figura 2: Exemplo de Aplicação da Pilha na avaliação de uma expressão

Fila (FIFO – First in, First out)

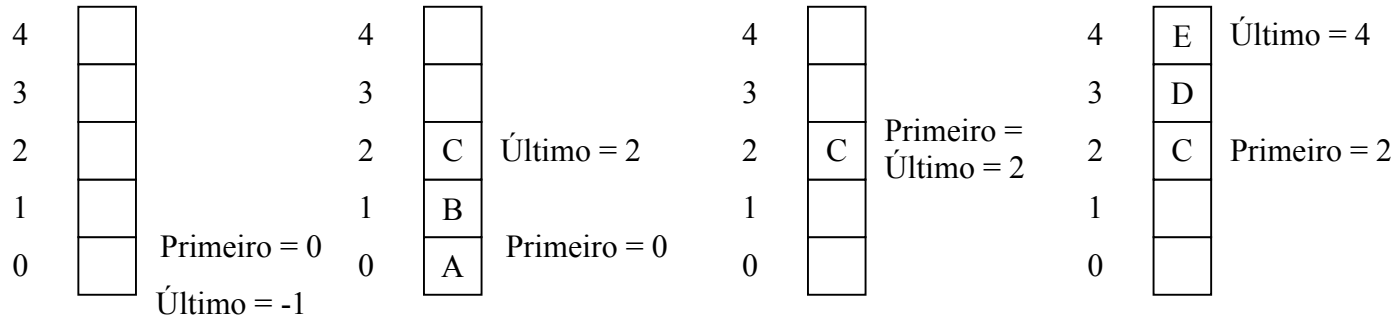
Uma fila é uma coleção ordenada de itens na qual itens são retirados pela posição frontal, chamada *primeiro*, e elementos são inseridos na ultima posição, chamada *último*.



Pode-se definir uma estrutura para representação de uma fila da seguinte forma:

```
Estrutura Fila{
  int dado[];
  int primeiro = 0, ultimo = -1, MAX;
} f;
```

Problemas com a representação de Filas em Array



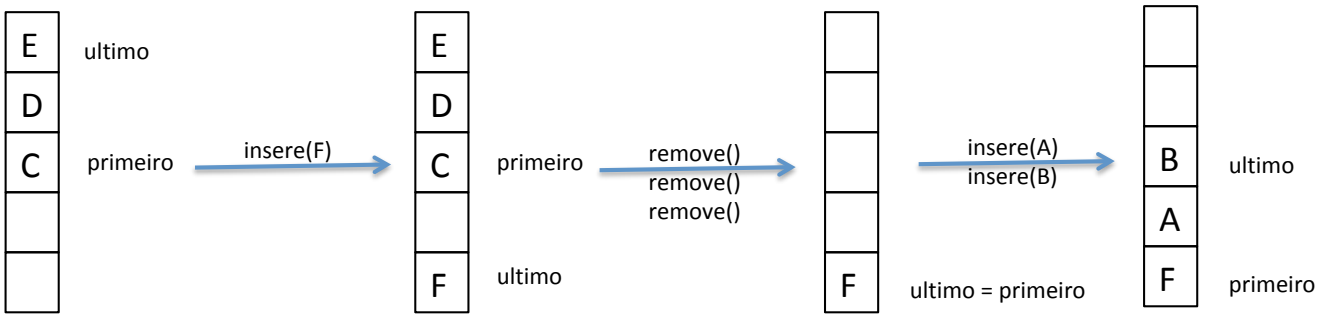
- Na implementação tradicional em array, a fila pode estar cheia mesmo apresentando posições livres (último estado do desenho acima);
- É possível deslocar todos os elementos restantes de uma fila quando o primeiro for removido, mas isso pode ser computacionalmente inviável (fila pode ter milhões de elementos)
- Uma forma mais eficiente de implementar fila pode utilizar uma representação circular, mas as condições de fila cheia e vazia não são mais válidas.

Implementação Circular de Fila

Nessa implementação, os ponteiros primeiro e último se movem circularmente dentre os índices do array, sempre na mesma direção, retornando de MAX - 1 para 0 quando houver espaço disponível.
Para detectar as condições de fila cheia ou vazia, a forma mais simples é manter um contador "tamanho", inicializado com zero (fila vazia) e que pode crescer até MAX – 1 (fila cheia).

```
Estrutura FilaCircular{
    int dado[];
    int primeiro = 0, ultimo = -1, MAX, tamanho = 0;
```

}



Exemplos de operações em Fila Circular em array.

Atividade

1) Implemente a estrutura e as operações de um TAD FilaCircular em array conforme a discussão da seção anterior na linguagem JAVA.

Exercícios

1. Dada uma determinada pilha *p* de números inteiros inicialmente vazia, represente graficamente as operações a seguir:
   a) p.empilha(10)
   b) p.empilha(-2)
   c) p.empilha(16)
   d) p.topo()
   e) p.desempilha()
   f) p.empilha(40)
   g) p.desempilha();
   h) p.desempilha();
   i) p.desempilha();
   j) p.empilha(35);

2. Simule a ação do algoritmo de verificação de "()", "{}" e "[]" em nas expressões aritméticas a seguir e verifique o resultado da avaliação (se a expressão é válida ou não):

a) (A + B})
b) {[A + B] – [(C - D)]
c) (A + B) – {C + D} – [F + G]
d) ((H) * {([J + K])})
e) (((A)))