

Curso: ADS

Estrutura de Dados

Aula 5 – Pilhas em C#

Profº Msc. Anderson L. Coan
anderson.coan@fatec.sp.gov.br



Filas e Pilhas

conceito

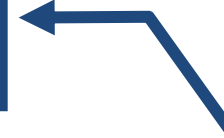
Início da FILA



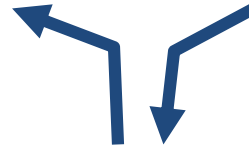
Primeiro que sai
(FIFO)



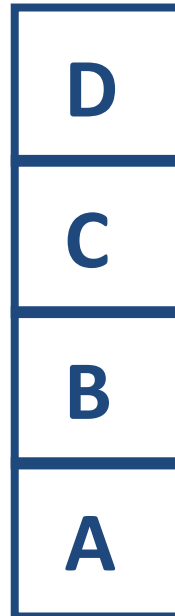
Primeiro
que entra



Último que entra
Primeiro que sai
(LIFO)



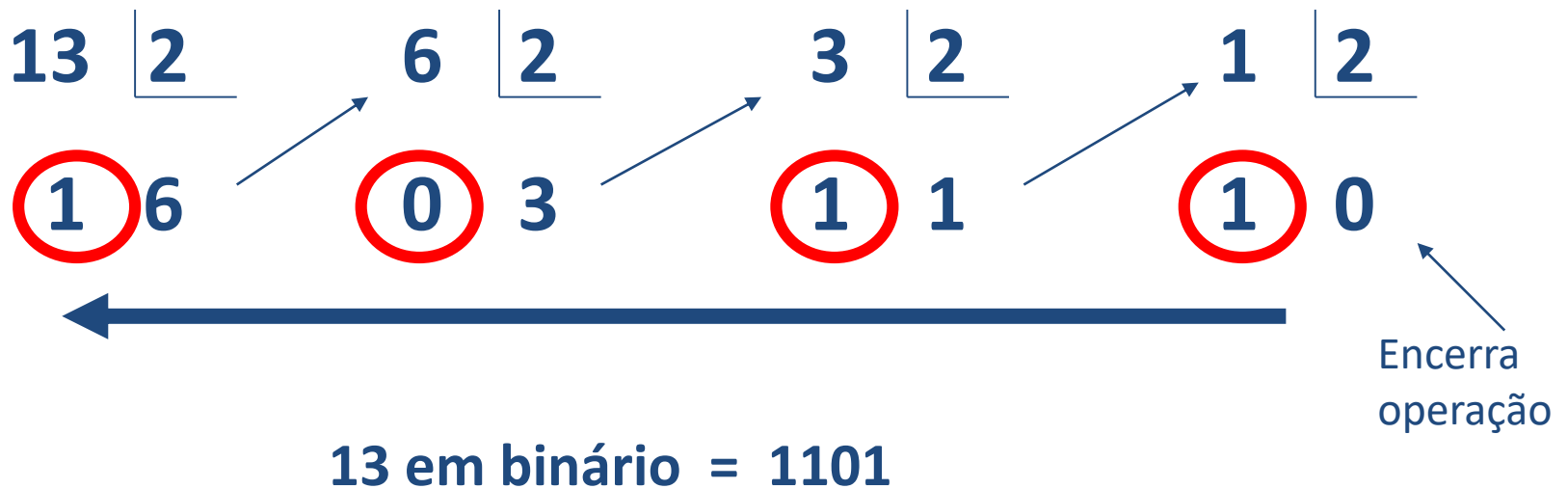
Topo da PILHA



- Uma pilha é uma lista linear em que apenas as operações de acesso, inserção e remoção são possíveis.
- Todas estas operações devem ser realizadas num mesmo extremo denominado topo.
- Devido às características das operações da pilha, o último elemento a ser inserido será o primeiro a ser retirado.
- Estruturas desse tipo são conhecidas como "LIFO" (last in, first out).
- Exemplo:
 - Em uma rua sem saída, tão estreita que apenas um carro passa por vez, o primeiro carro a sair será o último a ter entrado. Observe ainda que não podemos retirar qualquer carro e não podemos inserir um carro de tal forma que ele não seja o último.

Exemplos de uso de PILHAS

Conversão da base decimal para binária



Exemplos de uso de PILHAS

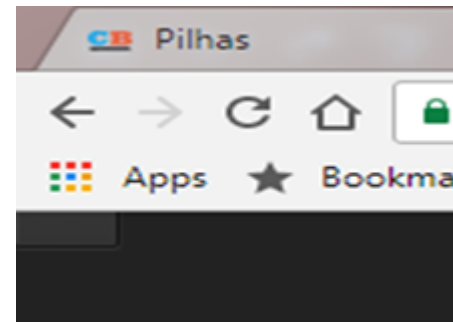
Pilha de pratos em um restaurante (analogia);



Mecanismo de desfazer/refazer dos editores de texto;



Botão Voltar dos Navegadores (Browsers);



Principais operações numa PILHA

- ✓ empilhar (push);
- ✓ desempilhar (pop);
- ✓ mostrar o topo;
- ✓ verificar se a pilha está vazia;
- ✓ verificar se a pilha está cheia.

Funções de manipulação de Pilhas

Comando *stack*

O comando *stack* é utilizado para a criação das Pilhas em C#; sua sintaxe é:

```
Stack<string> minhaPilha = new Stack<string>();
```

onde:

<string> = tipo de elemento que será enfileirado;

minhaPilha = nome da pilha;

new Stack<string>(); = criação da instância;

Funções de manipulação de Pilhas

Comando *Push* (Empilhar)

O comando *Push* é utilizado para adicionar elementos as Pilhas, como parâmetro passamos o elemento que desejamos adicionar.

Comando *Push* (Empilhar) – Criando e exibindo a pilha

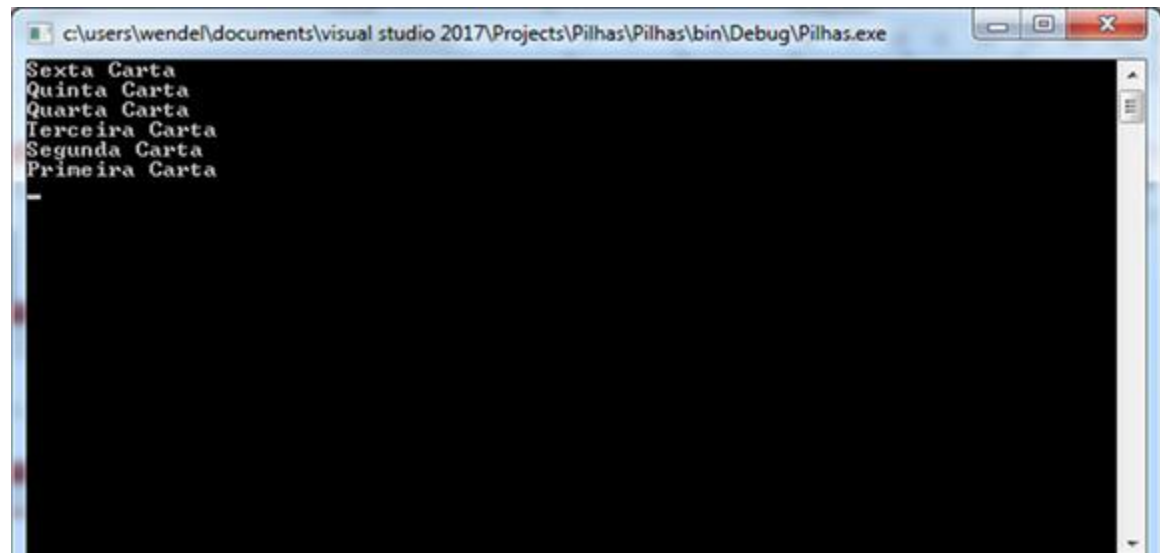
```
5  [using System.Threading.Tasks;
6
7  namespace Pilhas
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             Stack<string> minhaPilha = new Stack<string>();
14             minhaPilha.Push("Primeira Carta");
15             minhaPilha.Push("Segunda Carta");
16             minhaPilha.Push("Terceira Carta");
17             minhaPilha.Push("Quarta Carta");
18             minhaPilha.Push("Quinta Carta");
19             minhaPilha.Push("Sexta Carta");
20
21             foreach (string carta in minhaPilha)
22             {
23                 Console.WriteLine(carta);
24             }
25             Console.ReadLine();
26         }
27     }
28 }
29
30
```


Funções de manipulação de Pilhas

Comando *Foreach*

O *Foreach* é usado neste caso para percorrer a lista de objetos da pilha. Crie um atributo relacionado a coleção de objetos que você quer imprimir. Após isso basta imprimir este atributo, que a sua pilha será listada.

Note que o último elemento a ser adicionado foi o que veio para o topo (primeira posição da Pilha). Isto é o que caracteriza a diferença entre Pilhas e Filas.



```
c:\users\wendel\documents\visual studio 2017\Projects\Pilhas\Pilhas\bin\Debug\Pilhas.exe
Sexta Carta
Quinta Carta
Quarta Carta
Terceira Carta
Segunda Carta
Primeira Carta

```

Funções de manipulação de Pilhas

Comando *Peek*

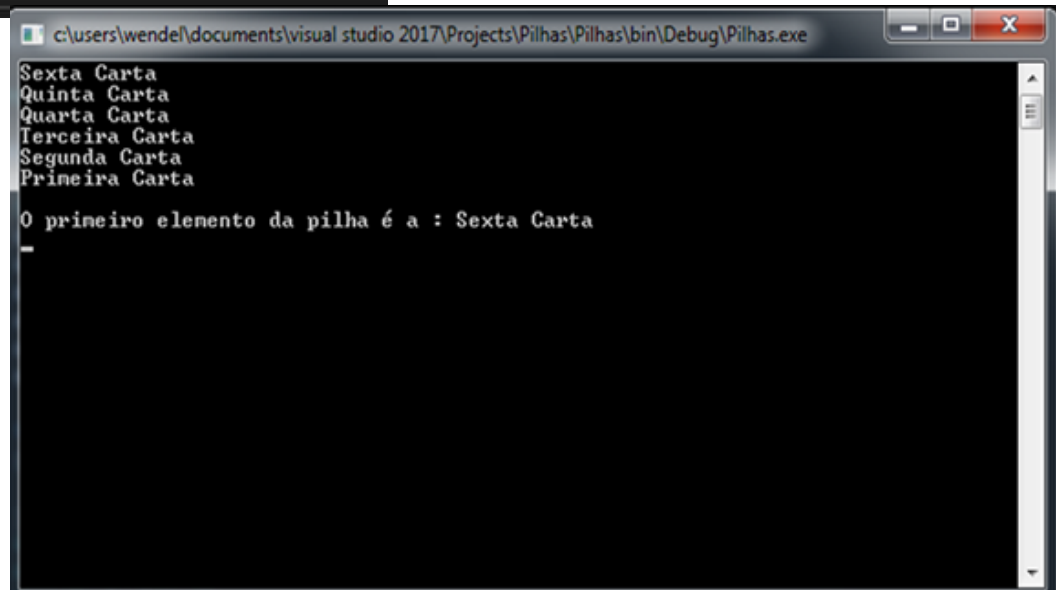
Para visualizar o primeiro elemento da Pilha, usamos o comando Peek (similar à Fila):

```
using System.Threading.Tasks;

namespace Pilhas
{
    class Program
    {
        static void Main(string[] args)
        {
            Stack<string> minhaPilha = new Stack<string>();
            minhaPilha.Push("Primeira Carta");
            minhaPilha.Push("Segunda Carta");
            minhaPilha.Push("Terceira Carta");
            minhaPilha.Push("Quarta Carta");
            minhaPilha.Push("Quinta Carta");
            minhaPilha.Push("Sexta Carta");

            foreach (string carta in minhaPilha)
            {
                Console.WriteLine(carta);
            }
            Console.WriteLine();
            Console.WriteLine("O primeiro elemento da pilha é a : " + minhaPilha.Peek());
            Console.ReadLine();
        }
    }
}
```

Comando Peek (Saída) →



```
c:\users\wende\documents\visual studio 2017\Projects\Pilhas\Pilhas\bin\Debug\Pilhas.exe
Sexta Carta
Quinta Carta
Quarta Carta
Terceira Carta
Segunda Carta
Primeira Carta

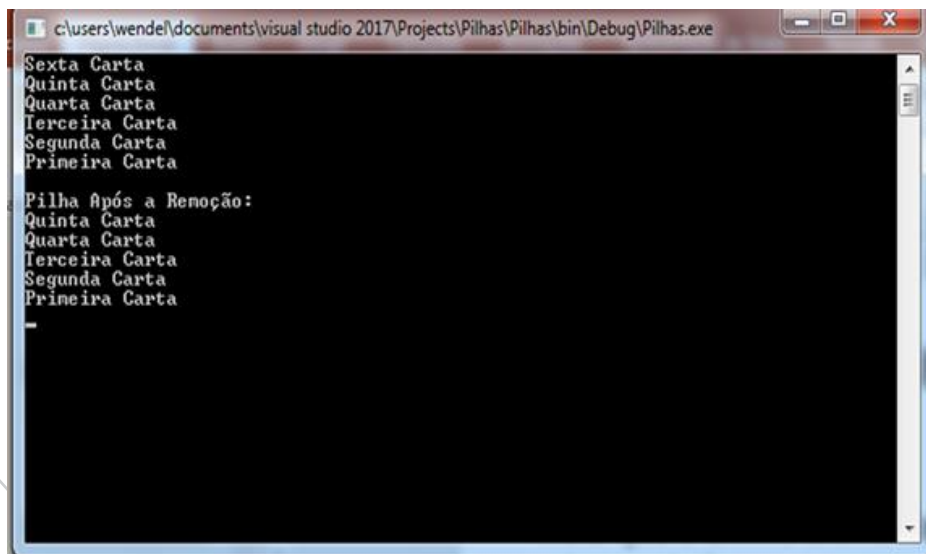
O primeiro elemento da pilha é a : Sexta Carta
_
```

Funções de manipulação de Pilhas

Comando *Pop* (Retira o primeiro elemento do topo)

Para removermos um elemento, utilizamos este comando, que retira um elemento (o do topo). É como se o primeiro prato em um restaurante self-service fosse pego por um cliente. O prato (elemento) a ser retirado é o que está no topo.

```
6 namespace Pilhas
7 {
8     class Program
9     {
10         static void Main(string[] args)
11         {
12             Stack<string> minhaPilha = new Stack<string>();
13             minhaPilha.Push("Primeira Carta");
14             minhaPilha.Push("Segunda Carta");
15             minhaPilha.Push("Terceira Carta");
16             minhaPilha.Push("Quarta Carta");
17             minhaPilha.Push("Quinta Carta");
18             minhaPilha.Push("Sexta Carta");
19
20             foreach (string carta in minhaPilha)
21             {
22                 Console.WriteLine(carta);
23             }
24             minhaPilha.Pop();
25             Console.WriteLine();
26             Console.WriteLine("Pilha Após a Remoção: ");
27             foreach (string carta in minhaPilha)
28             {
29                 Console.WriteLine(carta);
30             }
31             Console.ReadLine();
32         }
33     }
34 }
```



```
c:\users\wendel\documents\visual studio 2017\Projects\Pilhas\Pilhas\bin\Debug\Pilhas.exe
Sexta Carta
Quinta Carta
Quarta Carta
Terceira Carta
Segunda Carta
Primeira Carta

Pilha Após a Remoção:
Quinta Carta
Quarta Carta
Terceira Carta
Segunda Carta
Primeira Carta
```

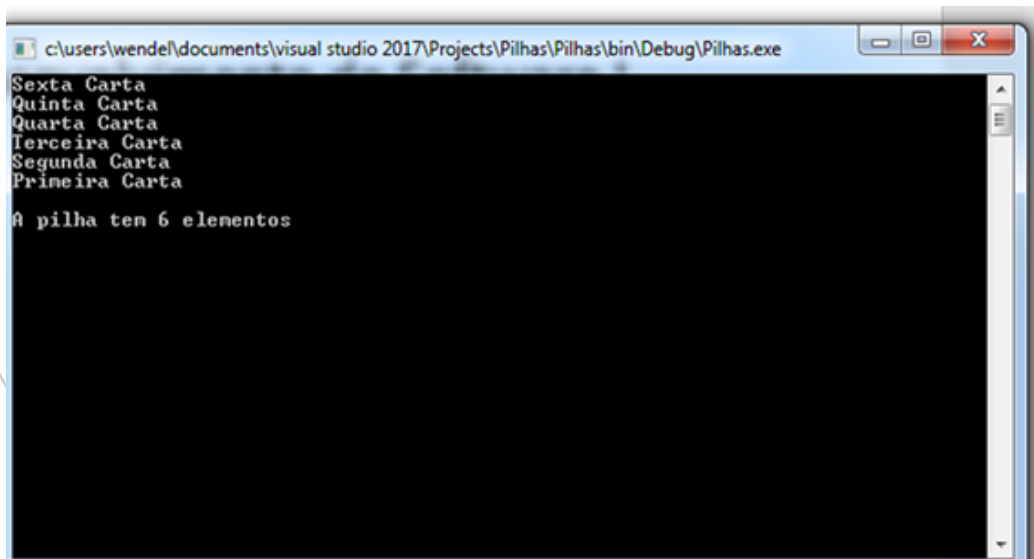
Foi inserido um segundo foreach (um antes da remoção e o outro após) para demonstrar a alteração:

Funções de manipulação de Pilhas

Comando *Count*

O Count (similar às Filas) permite contar quantos elementos existem numa Pilha.

```
13 Stack<string> minhaPilha = new Stack<string>();
14 minhaPilha.Push("Primeira Carta");
15 minhaPilha.Push("Segunda Carta");
16 minhaPilha.Push("Terceira Carta");
17 minhaPilha.Push("Quarta Carta");
18 minhaPilha.Push("Quinta Carta");
19 minhaPilha.Push("Sexta Carta");
20
21 foreach (string carta in minhaPilha)
22 {
23     Console.WriteLine(carta);
24 }
25 int contador = minhaPilha.Count();
26 Console.WriteLine();
27 Console.WriteLine("A pilha tem " + contador + " elementos");
28 Console.ReadLine();
29
30 // minhaPilha.Pop();
31 // Console.WriteLine();
32 // Console.WriteLine("Pilha Após a Remoção: ");
33 // foreach (string carta in minhaPilha)
34 // {
35 //     Console.WriteLine(carta);
36 // }
37 // Console.ReadLine();
38
39 }
40
41 }
```



```
c:\users\wendel\documents\visual studio 2017\Projects\Pilhas\Pilhas\bin\Debug\Pilhas.exe
Sexta Carta
Quinta Carta
Quarta Carta
Terceira Carta
Segunda Carta
Primeira Carta
A pilha tem 6 elementos
```

← Saída do comando

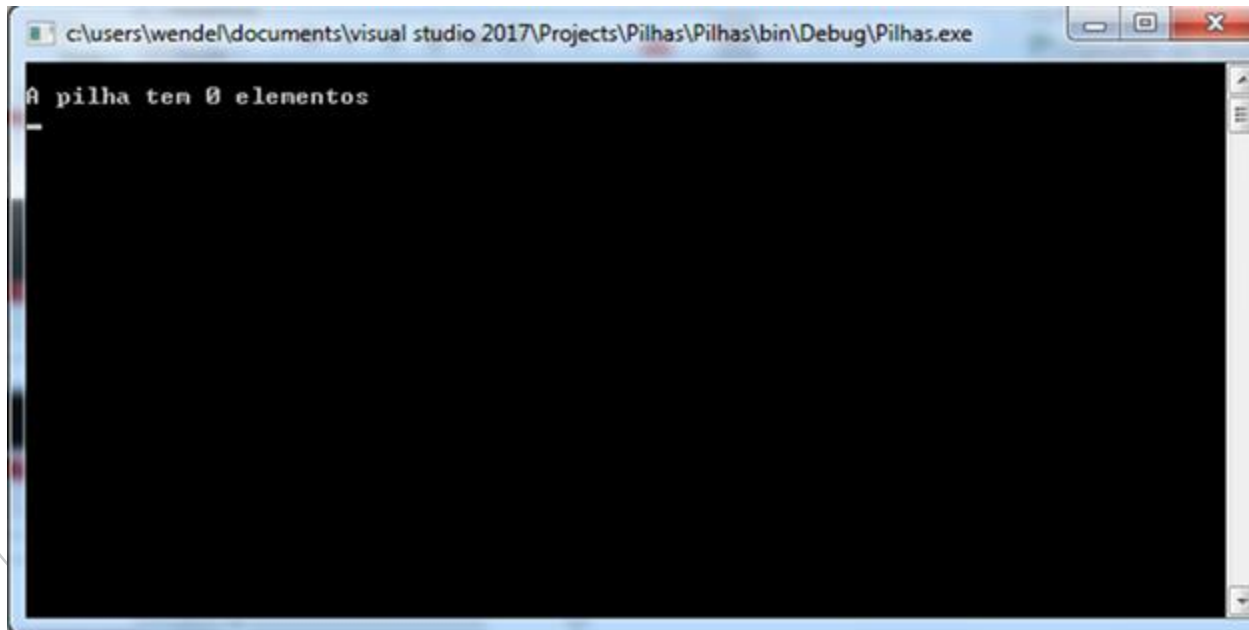
Funções de manipulação de Pilhas

Comando *Clear*

Para limpar nossa Pilha, usamos o Clear (Similar às Filas)

```
class Program
{
    static void Main(string[] args)
    {
        Stack<string> minhaPilha = new Stack<string>();
        minhaPilha.Push("Primeira Carta");
        minhaPilha.Push("Segunda Carta");
        minhaPilha.Push("Terceira Carta");
        minhaPilha.Push("Quarta Carta");
        minhaPilha.Push("Quinta Carta");
        minhaPilha.Push("Sexta Carta");
        minhaPilha.Clear();

        foreach (string carta in minhaPilha)
        {
            Console.WriteLine(carta);
        }
        int contador = minhaPilha.Count();
        Console.WriteLine();
        Console.WriteLine("A pilha tem " + contador + " elementos");
        Console.ReadLine();
    }
}
```

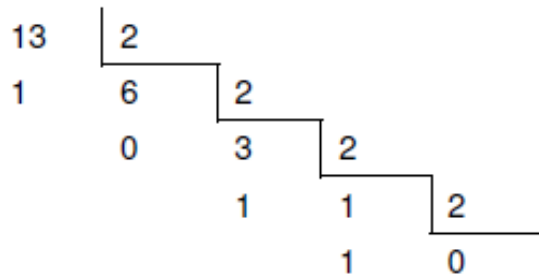


← Saída do comando

Exercícios

1. Seguindo a lógica para conversão de base decimal para binário, implemente em C# a função abaixo que foi escrita originalmente em C.

Dado um número inteiro, positivo em base decimal, convertê-lo para binário.



13 (decimal) = 1101 (binário)

```
# include <stdio.h>
# include <stdlib.h>
# include "pilhas.h"

int main ( ) {
    int n, r ;
    tpPilha p ;
    printf ( "Digite um inteiro positivo: " ) ;
    scanf ( "%d", &n) ;
    init ( &p ) ;
    do {
        r = n % 2 ;
        push ( &p, r ) ;
        n = n / 2 ;
    } while ( n != 0 ) ;
    printf ( "\n\nCorrespondente ao Binario => " ) ;
    while ( isEmpty ( &p ) == 0 ) {
        r = pop ( &p ) ;
        printf ( "%d", r ) ;
    }
    printf ( "\n\n\n" ) ;
    system ( "pause" ) ;
    return 0 ;
}
```

Exercícios

Arquivo “pilhas.h”:

```
#include <stdlib.h>
#include <stdio.h>

#define Max 50

typedef char tpElem;

typedef struct
{
    int topo;
    tpElem valor[50];
}tpPilha;

void init(tpPilha *p);
int isFull(tpPilha *p);
int isEmpty(tpPilha *p);
void push(tpPilha *p, tpElem X);
tpElem pop(tpPilha *p);
tpElem top(tpPilha *p);

void init(tpPilha *p)
{
    p->topo=-1;
}

int isFull (tpPilha *p)
{
    return (p->topo==Max);
}

void push (tpPilha *p, tpElem x)
{
    if (isFull(p)==0)
        p->valor[++p->topo]=x;
    else
    {
        printf("Pilha cheia! \n\n");
        system("pause");
    }
}

int isEmpty(tpPilha *p)
{
    return (p->topo==-1);
}

tpElem pop(tpPilha *p)
{
    if(isEmpty(p)==0)
        return (p->valor[p->topo--]);
    else
    {
        printf("Pilha vazia! \n\n");
        system("pause");
    }
}

tpElem top(tpPilha *p)
{
    if(isEmpty(p)==0)
        return (p->valor[p->topo]);
    else
    {
        printf("Pilha vazia! \n\n");
        system("pause");
    }
}
```

Exercícios

2. Uma pessoa endinheirada tem 7 marcas de veículos de luxo em sua garagem, a saber:

Mercedes-Benz; Jaguar; Rolls-Royce; BMW; Aston Martin; Lamborghini; Ferrari.

Você deve criar um vetor que irá armazenar estes veículos na ordem que foram passados e em seguida apresentar os veículos em ordem alfabética(ordenação).

3. Em uma aplicação Console Application crie uma matriz de 4 linhas e 5 colunas. Preencha-a (solicitando ao usuário os dados) e exiba seus valores.

4. Em um terminal rodoviário existe uma tabela de destinos com os ônibus que chegam e saem do terminal. Esta tabela segue o conceito First-In-First-Out. A tabela pode ser vista abaixo:

Ônibus 1 - Boracéia
Ônibus 2 - Guarujá
Ônibus 3 - Peruíbe
Ônibus 4 - Itanhaém
Ônibus 5 - Maresias

Crie uma aplicação (Console Applica

- exiba esta fila;
- conte quantos ônibus estão no terminal (fila cheia);
- depois que o primeiro ônibus sair do terminal exiba qual é o seguinte.

5. Estudamos em C# o algoritmo de ordenação QuickSort (Dividir para Conquistar). É sabido que existem vários outros algoritmos que permitem a ordenação de um vetor; neste sentido, qual a principal diferença entre o QuickSort e o BubbleSort?

DÚVIDAS



Curso: ADS

Estrutura de Dados

Aula 5 – Pilhas em C#

Profº Msc. Anderson L. Coan
anderson.coan@fatec.sp.gov.br

