UNISINOS - **UNIVERSIDADE DO VALE DO RIO DOS SINOS**
CIÊNCIAS EXATAS E TECNOLÓGICAS – Curso: Informática / Ciência da Computação

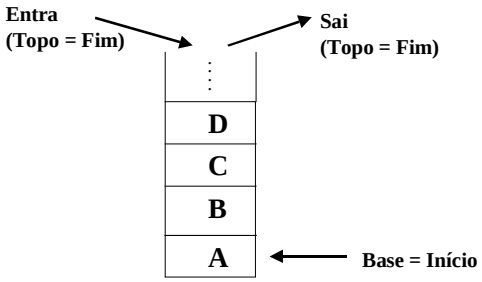
Programação II

Disciplina: Linguagem de Programação PASCAL **E-mail:** ***osorio@exatas.unisinos.br***
Professor responsável: ***Fernando Santos Osório*** **Web:**
Semestre: 2004/2 ***http://www.inf.unisinos.br/~osorio/prog2.html***
Horário: 63 **Xerox : *Pasta 54* – LAB. II (Xerox do "Alemão")**

Pilhas – LIFO = "Last In, First Out"

Conforme indicado anteriormente, as estruturas de dados podem ser organizadas de formas diferentes e especiais, de acordo com **a maneira com que os dados são inseridos e retirados**. Um tipo particular de estruturas de dados, é a **PILHA**, onde a inserção ocorre sempre no topo (final do vetor) e a retirada dos dados também ocorre no topo (final do vetor). Este tipo de estrutura de dados pode "simular" uma pilha de papéis (do mundo real), sendo muito usada em diversos tipos de aplicações computacionais, onde é essencial para a implementação de sub-rotinas (passagem de parâmetros e retorno). Nas pilhas o último dado a entrar é sempre o primeiro a sair (LIFO).

Manipulação de Pilha => PUSH = Insere no topo da pilha.
<= POP = Retira do topo da pilha.



Vamos definir a seguir um conjunto de **rotinas genéricas** para manipulação de pilhas, que simplificarão o seu uso e sua adaptação para uso em outras aplicações.

Definição dos Dados:

```
Const
    PILHA_TAM = 10;

Type
    TPilha_Dado = Integer;
    TPilha = Record
                 Dado: array [1..PILHA_TAM] of TPilha_Dado;
                 Base,Topo: Integer;
             End;
```

Definição das Rotinas:

Pilha_Inicializa,
Pilha_Insere,
Pilha_Retira,
Pilha_Exibe,
Pilha_Esvazia, ...

```
Procedure Pilha_Inicializa (Var P: TPilha);
Begin
   P.Base := 1;
   P.Topo := 0;
End;

Function   Pilha_Insere (Var P: TPilha; Dado: TPilha_Dado): Boolean;   {PUSH}
Begin
    IF  P.Topo = PILHA_TAM            { Testa overflow – Pilha cheia }
    THEN Pilha_Insere := False
    ELSE  Begin
              P.Topo := P.Topo + 1;
              P.Dado [P.Topo] := Dado;
              Pilha_Insere := True;
          End;
End;

Function   Pilha_Retira (Var P: TPilha; Var Dado: TPilha_Dado): Boolean;   {POP}
Begin
    IF  P.Topo < P.Base             { Testa Underflow – Pilha Vazia }
    THEN  Pilha_Retira := False
    ELSE   Begin
               Dado := P.Dado [P.Topo];
               P.Topo := P.Topo – 1;
               Pilha_Retira := True;
           End;
End;

Procedure Pilha_Exibe (P: TPilha);
Var
     aux: Integer;
Begin
    Writeln('Base:');
    FOR aux:=P.Base TO P.Topo
    DO Writeln (P.Dado[aux]);
    Writeln('Topo:');
    Write ('Tecle ENTER para continuar...');
    Readln;
End;
```

```
Procedure Pilha_Esvazia (Var P: TPilha);
Var
   Dado: TPilha_Dado;
Begin
   WHILE  Pilha_Retira (P, Dado)
   DO ;
End;
```

Rotinas complementares...

```
Function   Pilha_Vazia (P: TPilha): Boolean;              { Testa para ver se a pilha está vazia }
Function   Pilha_Quantidade (P: TPilha): Integer;         { Indica quantos dados tem na pilha }
Procedure  Pilha_Inverte (Var P: TPilha);                  { Inverte a ordem dos dados da pilha  }
Procedure  Pilha_Grava (Var ArqTxt: Text; P: TPilha); {Salva em disco os dados}
Function   Pilha_Le (Var ArqTxt: Text; Var P:TPilha); {Lê do disco os dados}
```

Exemplo de uso das rotinas...

```
Program Testa_Pilha;

>>> Incluir rotinas de pilha, {$I pilha.pas }, ou  utilizar uma Unit pilha.tpu: "Uses  pilha;"  <<<

Var
   P1,P2:TPilha;
    Dado:TPilha_Dado;
Begin
    Pilha_inicializa(P1);
    Pilha_inicializa(P2);
    IF not ( Pilha_insere(P1,10) )  THEN Writeln('Erro: Estouro de pilha!');
    IF not ( Pilha_insere(P1,20) )  THEN Writeln('Erro: Estouro de pilha!');
    IF not ( Pilha_insere(P1,30) )  THEN Writeln('Erro: Estouro de pilha!');
    writeln('Pilha P1:');
    Pilha_exibe(P1);
    Pilha_retira(P1,Dado);
    Pilha_insere(P2,Dado);
    Pilha_retira(P1,Dado);
    Pilha_insere(P2,Dado);
    Pilha_retira(P1,Dado);
    Pilha_insere(P2,Dado);
    writeln('Pilha P2:');
    Pilha_exibe(P2);
End.
```

EXERCÍCIOS:

1) Faça um programa que gere 20 dados aleatórios com valores entre 1 e 6, armazenando estes dados em 2 pilhas (10 valores em cada pilha). Após criadas as 2 pilhas de dados, mostre na tela o seu conteúdo e em seguida inicie o jogo: leia os dados da pilha, retirando um dado do topo de cada pilha, e verificando se o usuário conseguiu obter um "duplo seis". Se nas 10 jogadas o usuário não obteve nenhum "duplo seis", você deve exibir na tela a mensagem "Perdeu!", e caso uma das 10 jogadas contenha um "duplo seis", você deve exibir na tela a mensagem "Ganhou!".
   Obs: para gerar um valor pseudo-aleatório entre 1 e 6 use o comando *Valor := rand(6)+1;*

2) Faça um programa que gere uma pilha de 10 cartas de um baralho, ou seja, gere uma seqüência de cartas dispostas em uma pilha onde cada carta deve possuir um valor (1 a 10, J, Q, K ou As) e um naipe ("Ouro", "Paus", "Espadas" ou "Copas"). Assumimos que as cartas podem ser retiradas de múltiplos baralhos, logo, não é preciso verificar se existem cartas duplicadas. Uma vez criado o baralho, fazer um programa que fique interagindo com o usuário, solicitando se ele deseja mais uma carta e exibindo a carta reirada do baralho. Veja abaixo o exemplo de como deve ser a tela de execução do programa:

```
>> Baralho Eletrônico <<

Gerando pilha de cartas... OK
Você deseja mais uma carta (S/N)? S
Carta: 10 de Ouro
Você deseja mais uma carta (S/N)? S
Carta: K de Copas
Você deseja mais uma carta (S/N)? S
Carta: 3 de Espadas
Você deseja mais uma carta (S/N)? S

Fim da execução.
```

   Obs: Para simplificar gere valores entre 1 e 14, rand(14)+1, e atribua o resultado para uma string que deve conter valores de 1 a 10 ou J=11, Q=12, K=13 e As=14. Faça o mesmo para a geração do naipe, Ouro=1, Paus=2, Espadas=3, Copas=4

3) Faça um programa que simule uma calculadora científica HP de pilha, onde o usuário deve ter disponíveis apenas as seguintes operações, acessíveis através de um menu de opções:
   1- **Empilha valor** = Coloca o valor real digitado no topo da pilha
   2- **Desempilha valor** = Retira o dado do topo da pilha, exibindo seu valor
   3- **Exibe valor** = Exibe o valor contido no topo da pilha, sem no entanto retirar este dado definitivamente da pilha
   4- **Somar** = Soma os 2 valores do topo da pilha (desempilha) e guarda (empilha) o resultado
   5- **Subtrair** = Subtrai os 2 valores do topo da pilha (desempilha) e guarda (empilha) o resultado. O primeiro valor desempilhado deve ser subtraído do segundo valor.
   6- **Multiplica** = Multiplica os 2 valores do topo da pilha (desempilha) e guarda (empilha) o resultado
   7- **Divide** = Divide os 2 valores do topo da pilha (desempilha) e guarda (empilha) o resultado. O segundo valor desempilhado deve ser dividido pelo primeiro valor.
   8- **Fim**