6. Pilhas Múltiplas ("2 Pilhas") - Estruturas de Dados

6.1 Conceito

- **Pilha** é uma estrutura de dados que admite *remoção de elementos e inserção de novos objetos*;
- Sempre que houver uma remoção, o elemento removido é o que está na estrutura a menos tempo;
- **LIFO** ("Last in, First out"), ou seja, o último a entrar será o primeiro a sair;
- Os elementos vão sendo inseridos um a um com o comando PUSH, e retirados com o comando POP, assim são empilhados e desempilhados sempre pelo topo da pilha;
- Pilhas múltiplas com o exemplo de uso sendo uma **lista concorrente**, são estruturas que "disputam" pelo mesmo espaço de armazenamento;
- Em exemplos com pilhas, tem-se dois casos: "2 pilhas" e "N pilhas", a que será abordada nesse tópico será a primeira opção.

6.2 Exemplos de Aplicação

- Softwares aplicativos;
- Editores de texto;
- Editores de Planilhas;
- Função "CTRL+Z", o voltar;
- Lista concorrente;
- Navegação entre páginas Web;
- Entre outros.

6.3 Estrutura da PILHA (caso 2 pilhas)

```
#define MAXPILHA 10

struct TpPilhaM{
    int topo1, topo2;
    char pilha[MAXPILHA];
};
```

Obs.: Nesse exemplo é criado uma pilha de char(caracteres).

6.4 Operações Associadas

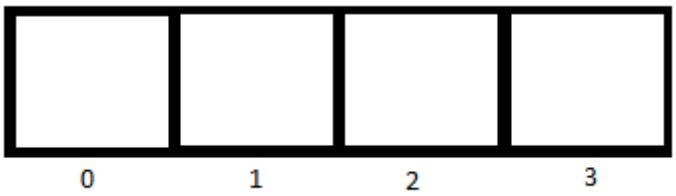
- void inicializar (TpPilhaM &pilha);
- void inserir (TpPilhaM &pm, char elem, int nPilha); (PUSH)
- char retirar (TpPilhaM &pm, int nPilha); (POP)
- char elementoTopo (TpPilhaM pm, int nPilha); (isTop)
- int cheia (int topo1, int topo2); (isFull)
- int vazia (int topo, int nPilha); (isEmpty)
- void exibir (TpPilhaM pm, int nPilha);

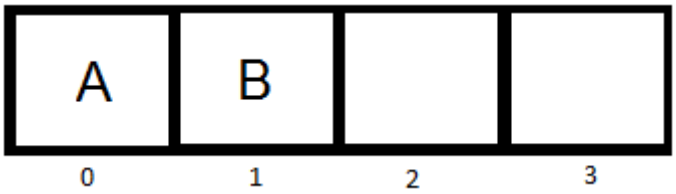
6.5 Implementação em C

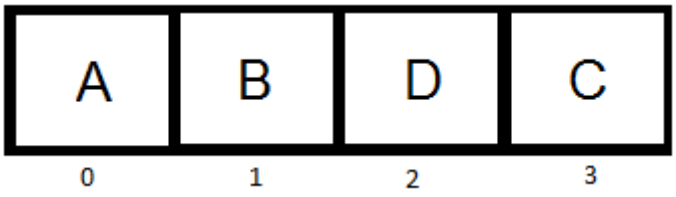
```
#define MAXPILHA 10

struct TpPilhaM{
    int topo1, topo2;
    char pilha[MAXPILHA];
};

void inicializar(TpPilhaM &pilha){
    pilha.topo1 = -1;
    pilha.topo2 = MAXPILHA;
}

void inserir(TpPilhaM &pm, char elem, int nPilha){
    if(nPilha == 1)
        pm.pilha[++pm.topo1] = elem;
    else
        pm.pilha[--pm.topo2] = elem;
}
char retirar(TpPilhaM &pm, int nPilha){
    if(nPilha == 1)
        return pm.pilha[pm.topo1--];
    else
        return pm.pilha[pm.topo2++];
}

char elementoTopo(TpPilhaM pm, int nPilha){
    if(nPilha == 1)
        return pm.pilha[pm.topo1];
    else
        return pm.pilha[pm.topo2++];
}

int cheia(int topo1, int topo2){
    return topo1 == topo-1;
}
int vazia(int topo, int nPilha){
    if(nPilha == 1)
        return topo == -1;
    else
        return topo == MAXPILHA;
}
```

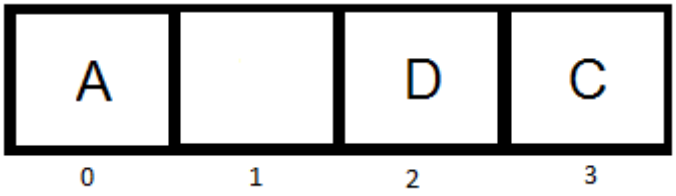
```
void exibir(TpPilhaM pm, int nPilha){
    if(nPilha == 1)
        while(!vazia(pm.topo1, nPilha))
            printf("\n%c", retirar(pm, nPilha))

    else
        while(!vazia(pm.topo2, nPilha))
            printf("\n%c", retirar(pm, nPilha))
}
```

6.6 Funcionamento

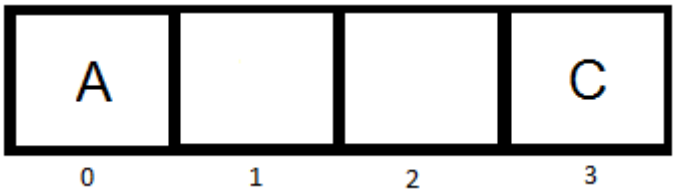
1. Inicializar:
    - Topo 1 = -1;
    - Topo 2 = MAXPILHA;
    - Pilha = 

2. Inserir 'A'/'B' na pilha 1:
    - Topo 1 = 1;
    - Topo 2 = MAXPILHA;
    - Pilha = 

3. Inserir 'C'/'D' na pilha 2:
    - Topo 1 = 1;
    - Topo 2 = 2;
    - Pilha = 

4. Retirar da pilha 1:
    - Topo 1 = 0;
    - Topo 2 = 2;
    - Pilha = 

5. Retirar da pilha2:
    - Topo 1 = 0;
    - Topo 2 = 3;

- Pilha = 

6. Inserir 'E' na pilha 1:
    - Topo 1 = 1;
    - Topo 2 = 3;
    - Pilha =