7. Pilhas Múltiplas ("N Pilhas") - Estruturas de Dados

7.1 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 segunda opção.

7.2 Exemplos de Aplicação

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

7.3 Estrutura da PILHA (caso N Pilhas)

```
#define MAXPILHA 10

struct TpPilhaM2{
    int bases[MAXPILHA], topos[MAXPILHA];
    char pilhas[MAXPILHA];
};
```

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

7.4 Operações Associadas

- void inicializar (TpPilhaM2 &PM, int quantidade);
- void inserir (TpPilhaM2 &PM, char elem, int nPilha); (PUSH)
- char retirar (TpPilhaM2 &PM, int nPilha); (POP)
- char elementoTopo (TpPilhaM2 PM, int nPilha); (isTop)
- int cheia (TpPilhaM2 PM, int nPilha); (isFull)
- int vazia (TpPilhaM2 PM, int nPilha); (isEmpty)
- void exibir (TpPilhaM2 PM, int nPilha);

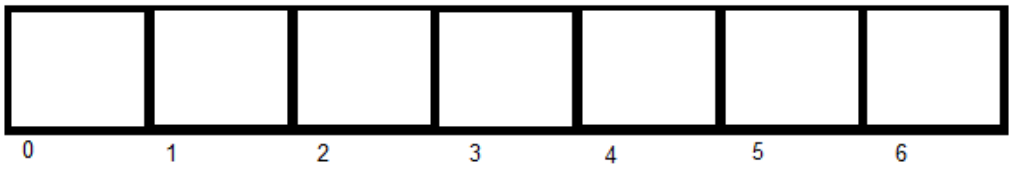
7.5 Implementação em C

```
#define MAXPILHA 10

struct TpPilhaM2{
    int bases[MAXPILHA], topos[MAXPILHA];
    char pilhas[MAXPILHA];
};

void inicializar(TpPilhaM2 &PM, int quantidade){
    int i, quantElem = MAXPILHA/quantidade;

    for(i=0; i<=quantidade; i++){
        PM.bases[i] = quantElem * i;
        PM.topos[i] = PM.bases[i] - 1;
    }
}

void inserir(TpPilhaM2 &PM, char elem, int nPilha){
    PM.pilhas[++PM.topos[nPilha]] = elem;
}
char retirar(TpPilhaM2 &PM, int nPilha){
    return PM.pilhas[PM.topos[nPilha]--];
}

char elementoTopo(TpPilhaM2 PM, int nPilha){
    return PM.pilhas[PM.topos[nPilha]];
}

int cheia(TpPilhaM2 PM, int nPilha){
    return PM.topos[nPilha]+1 == PM.bases[nPilha+1];
}
int vazia(TpPilhaM2 PM, int nPilha){
    return PM.topos[nPilha] < PM.bases[nPilha];
}

void exibir(TpPilhaM2 PM, int nPilha){
    while(!vazia(PM, nPilha))
        printf("\n%c", retirar(PM, nPilha));
}
```

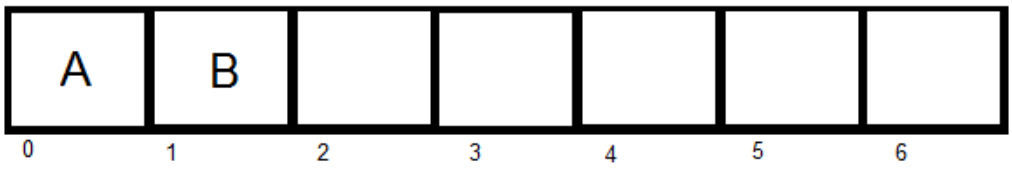
7.6 Funcionamento

- Observação:
  - O MAXPILHA nesse exemplo está valendo 7;
  - Foram-se criadas 3 pilhas (resultando que cada pilha terá 2 espaços, pois 7/3 ≅ 2 (sempre é considerado somente a parte inteira)).

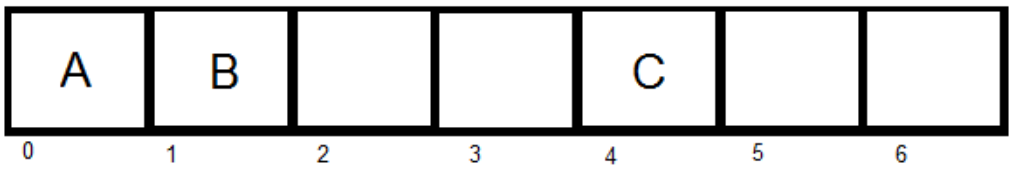
1. Inicializar com 3 pilhas:
   - Bases = 0/2/4 (respectivamente)
   - Topos = -1/1/3 (respectivamente)
   - Pilha =



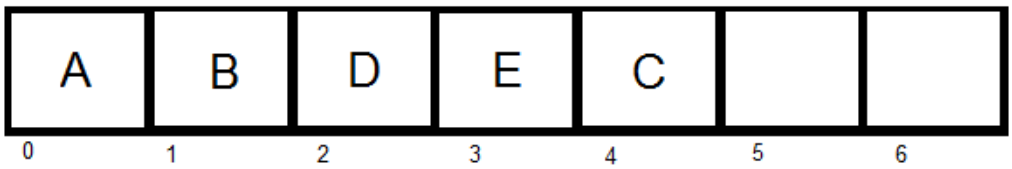
2. Inserir 'A'/'B' na pilha 0:
   - Bases = 0/2/4
   - Topos = 1/1/3 (topo da pilha 0 está valendo 1)
   - Pilha =



3. Inserir 'C' na pilha 2:
   - Bases = 0/2/4
   - Topos = 1/1/4 (topo da pilha 2 está valendo 4)
   - Pilha =

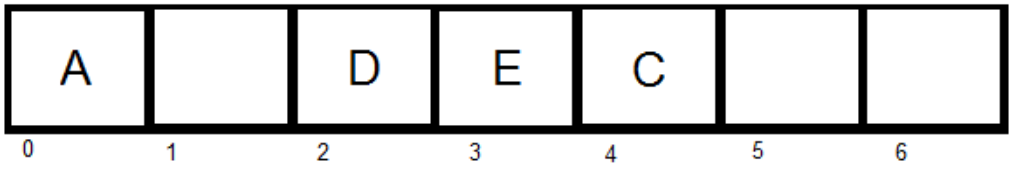


4. Inserir 'D'/'E' na pilha 1:
   - Bases = 0/2/4
   - Topos = 1/3/4
   - Pilha =

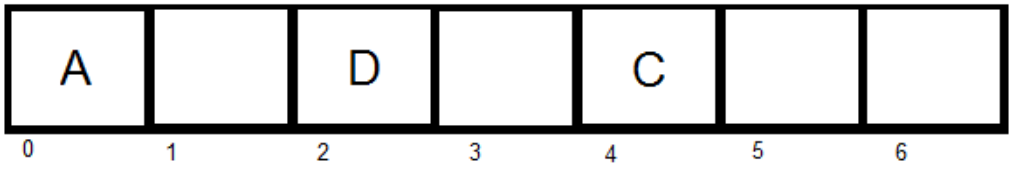


5. Retirar da pilha 0:
   - Bases = 0/2/4
   - Topos = 0/3/4

- Pilha =



6. Retirar da pilha 1:
   - Bases = 0/2/4
   - Topos = 0/2/4
   - Pilha =



7. Inserir na pilha 2:
   - Bases = 0/2/4
   - Topos = 0/2/5
   - Pilha =