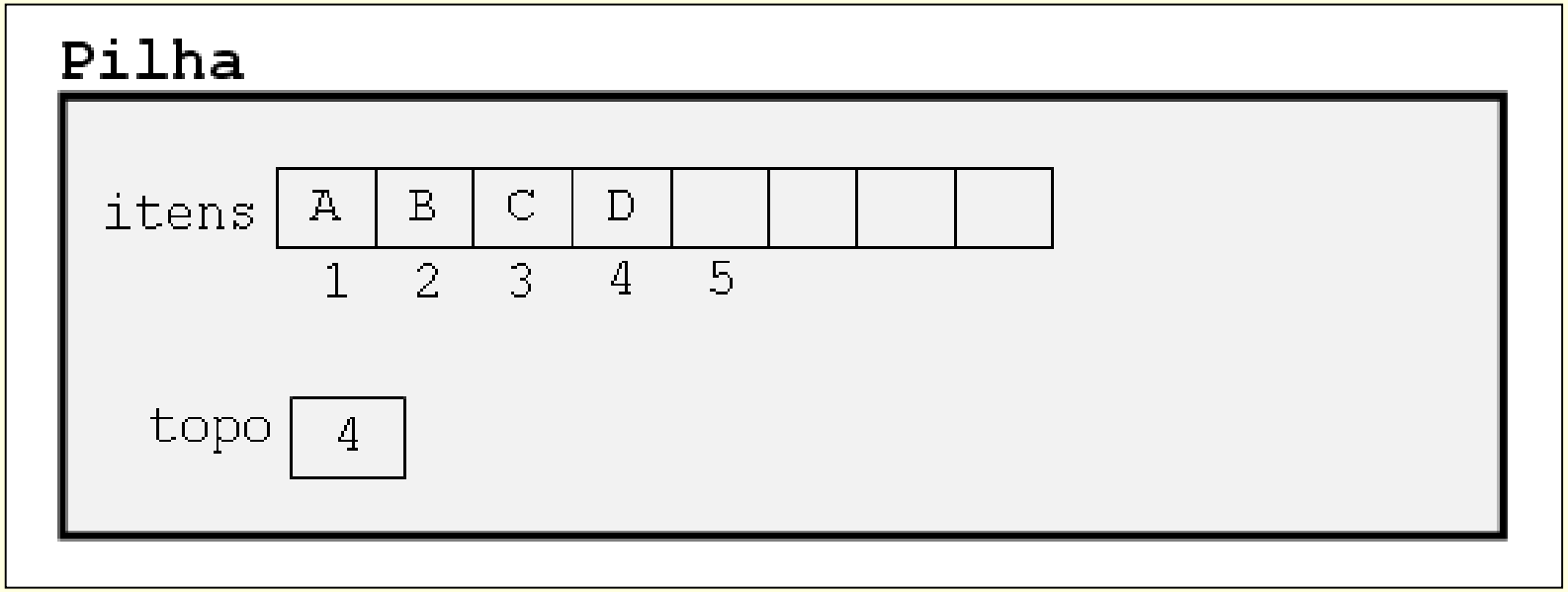
# Pilha

SCC0202 – Algoritmos e Estruturas de Dados I

Baseado nos slides do Prof. Thiago A. S. Pardo

# Implementação da pilha

- Sequencial e estática



# Implementação da pilha

- Declaração em C

```
#define TamPilha 100

typedef int elem;

typedef struct {
            int topo;
            elem itens[TamPilha];
} Pilha;

Pilha P;
```

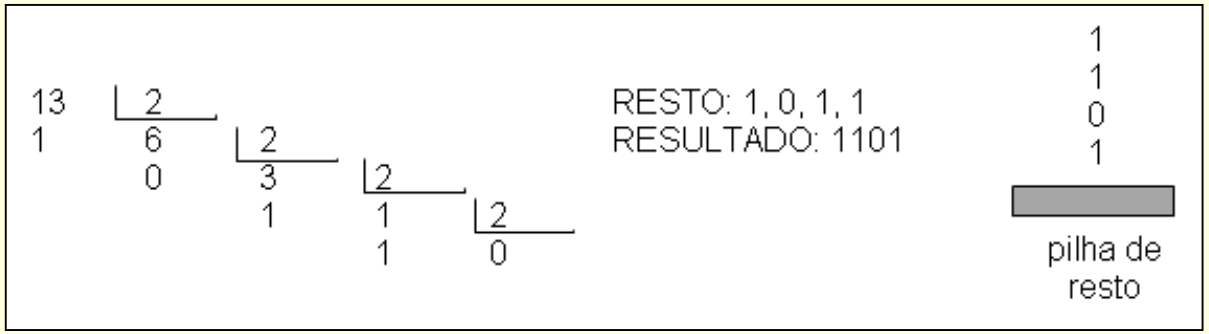
# Funções do TAD pilha (aula passada)

```
void Create(Pilha *P) {
     P->topo=-1;
     return;
}

void Empty(Pilha *P) {
     P->topo=-1;
     return;
}

int IsEmpty(Pilha *P) {
    if (P->topo==-1)
       return 1;
    else return 0;
}

int IsFull(Pilha *P) {
    if (P->topo==TamPilha-1)
       return 1;
    else return 0;
}
```

```
void Push(Pilha *P, elem *X, int *erro) {
     if (!IsFull(P)) {
          *erro=0;
          P->topo++;
          P->itens[P->topo]=*X;
     }
     else *erro=1;
     return;
}

void Pop(Pilha *P, elem *X, int *erro) {
     if (!IsEmpty(P)) {
          *erro=0;
          *X=P->itens[P->topo];
          P->topo--;
     }
     else *erro=1;
     return;
}
```

# Exercício: simular algoritmo abaixo com as funções reais da pilha

- Conversão decimal-binário




```
Variáveis
        P: pilha
        N: inteiro {número a ser
                    convertido}
        X: inteiro {resto da divisão}

Início do algoritmo
        leia N
        create(P)
        ...
        ...
        repita
                X=resto(N,2)
                push(P,X)
                N=quociente(N,2)
        até que (N=0)
        escreva “o resultado é ”
        enquanto (IsEmpty(P)=falso) faça
                pop(P,X)
                escreva X
Fim
```

# Exercício

- Continuando o TAD pilha
  - Implementar função Top

# Exercício

- Adicionar uma rotina ao TAD para **imprimir** os elementos de uma pilha

# Exercício

- Adicionar uma rotina ao TAD para verificar se duas pilhas são **iguais**

# Exercício

- Adicionar uma rotina ao TAD para **inverter** a posição dos elementos de uma pilha

# Notação posfixa: exercício

- Avaliação de expressões aritméticas
  - Às vezes, na aritmética tradicional, faz-se necessário usar parênteses para dar o significado correto à expressão
    - A*B-C/D → (A*B)-(C/D)
  - Notação polonesa (prefixa): operadores aparecem antes dos operandos e dispensam parênteses
    - -*AB/CD
  - Notação polonesa reversa (posfixa): operadores aparecem depois dos operandos
    - AB*CD/-

# Notação posfixa: exercício

- Interpretação da notação posfixa usando pilha
  - Empilha operandos até encontrar um operador
  - Retira os operandos, calcula e empilha o resultado
  - Até que se chegue ao final da expressão

# Notação posfixa: exercício

- AB*CD/-

| | | | | | |
|---|---|---|---|---|---|
| A | | | | | |
| A | B | | | | |
| A | B | * | | | |
| A*B | | | | | |
| A*B | C | | | | |
| A*B | C | D | | | |
| A*B | C | D | / | | |
| A*B | C/D | | | | |
| A*B | C/D | - | | | |
| A*B-C/D | | | | | |

# Exercício

- Implemente um programa que **calcule o valor de uma expressão posfixa** utilizando o TAD pilha

# **Editor de texto**: exercício

- Considere que um editor de texto representa os caracteres digitados como uma pilha, sendo que o último caractere lido fica no topo

- Alguns comandos apagam caracteres
  - # representa *backspace* (apaga só o último caractere digitado)
  - @ indica “apagar tudo”

- Faça um programa que execute essas ações usando o TAD pilha

# Pilha

- TAD bem implementado