# Aula 10  - Estrutura de dados 1 – Uniube

Prof. Marcos Lopes

34 9 9878 0925

## Estrutura de dados do tipo pilha (implementação com array/vetor)

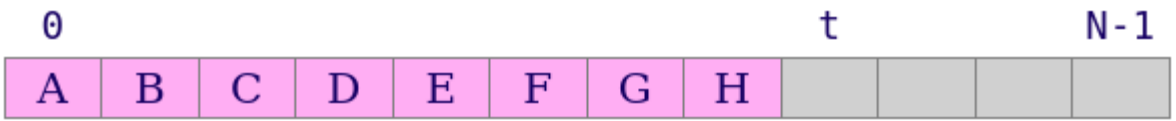
Suponha que nossa pilha está armazenada em um vetor pilha[0..N-1]. (A natureza dos elementos do vetor é irrelevante: eles podem ser inteiros, bytes, ponteiros, etc.)

Digamos que a parte do vetor ocupada pela pilha é pilha[0..t-1] .

O índice t indica a primeira posição vaga do vetor e t-1 é o índice do topo da pilha.

A pilha está vazia se t vale 0 e cheia se t vale N.

No exemplo da figura, os caracteres A, B, … , H foram inseridos na pilha nessa ordem:



Para remover, ou tirar, um elemento da pilha — essa operação é conhecida como desempilhar (= to pop) — faça:

```
x = pilha[--t];
```

Isso equivale ao par de instruções `t -= 1;` `x = pilha[t];`. É claro que você só deve desempilhar se tiver certeza de que a pilha não está vazia.

Para inserir, ou colocar, um objeto y na pilha — a operação é conhecida como empilhar (= to push) — faça:

```
pilha[t++] = y;
```

Isso equivale ao par de instruções `pilha[t] = y;` `t += 1;`. Antes de empilhar, certifique-se de que a pilha não está cheia, para evitar um transbordamento (= overflow).

Para facilitar a leitura do código, é conveniente embalar essas operações em duas pequenas funções. Se os objetos com que estamos lidando são do tipo char, podemos escrever

```
char pop () {
   return pilha[--t];
}

void push (char y) {
   pilha[t++] = y;
}
```

**Estamos supondo aqui que as variáveis pilha e t são globais, ou seja, foram definidas fora do código das funções.**

Para completar o pacote, precisaríamos de mais três funções: uma que crie uma pilha, uma que verifique se a pilha está vazia e uma que verifique se a pilha está cheia. (Veja exercício abaixo.)

Exercícios)

1) Implementar um programa em C que demonstra a implementação da estrutura de dados pilha, usando arrays. Usar o seguinte programa como ‘main’:

```
int main(void) {

    push('A');
    push('B');

    char topo = peek();
    printf("\ntopo peek: %c", topo);

    topo = pop();
    printf("\ntopo pop: %d", topo);

    return EXIT_SUCCESS;
}
```

2) Implementar a solução do balanceamento de expressões (definida na aula anterior) usando essa solução de pilha.