Estruturas

Pilha estática

Entenda o mecanismo de implementação de pilhas com a estrutura de um vetor

---

A pilha é uma estrutura de dados que segue a filosofia **LIFO** (Last in, first out). Sua concepção prevê um elemento especial, chamado de topo, que representa o local em que tanto as inserções quanto as remoções de elementos serão realizados. Neste ponto, é imprescindível enfatizar que, em pilhas, remoções e inserções não podem ocorrer em qualquer outra posição que não seja o topo.

A pilha pode ser implementada de forma estática (usando vetores) ou de forma dinâmica, com alocação de memória ajustável conforme a necessidade.

🔋 Declarando a pilha

O código a seguir apresenta a declaração das estruturas necessárias para a criação (e manipulação) de uma pilha implementada de forma estática. Para começar, temos uma estrutura chamada `REGISTRO`, que pode armazenar diversas variáveis de interesse. Para fins de exemplo, o código apresenta apenas uma variável, `chave`.

A pilha em si é definida pela estrutura `PILHA`, que contém `registros`, um vetor de itens do tipo `REGISTRO`. O topo, neste caso, é representado por uma variável do tipo inteiro, que representa a posição da pilha utilizada como topo.

```
#define TAM 3

typedef struct{
    int chave;
} REGISTRO;

typedef struct{
    REGISTRO itens[TAM];
```

```
    int topo;
} PILHAESTATICA;
```

🛠️ Funções de manipulação da pilha

Como destacado anteriormente, uma estrutura de dados do tipo pilha envolve algumas operações essenciais.

Inicializar a pilha

O procedimento de inicialização da pilha consiste, basicamente, em atribuir o valor `0` à variável `topo` da pilha. Esse processo de atribuição é necessário porque, ao instanciarmos uma pilha, o valor dessa variável será desconhecido (lembre-se que em C as variáveis não inicializadas possuem lixo de memória como valor).

O processo de inicialização garante que a pilha funcione de forma adequada, já que sua variável `topo` terá o valor necessário para o funcionamento do algoritmo.

```
void inicializarPilha(PILHAESTATICA *p){
    p->topo = 0;
}
```

> Não se esqueça de inicializar a pilha quando for utilizá-la. Muitos erros acontecem porque a pilha não é inicializada antes da operação de inserção, por exemplo.

Inserir (ou empilhar)

A inserção de elementos numa pilha estática assume que a variável `topo` representa a posição do vetor em que a inserção deve ser efetuada. Se a pilha estiver vazia, a inserção será na posição `0`. Se a pilha tiver um elemento, a inserção será na posição `1` do vetor.

Desse modo, deve-se observar se o valor da variável `p->topo` é menor que a quantidade de itens da pilha. Isso acontece porque, se a pilha puder armazenar 3 elementos (`#define TAM 3`), as posições válidas do vetor vão de `0` a `2`.

```
bool empilhar(PILHAESTATICA *p, REGISTRO r){
```

```
    if(p->topo < TAM){
        p->itens[p->topo] = r;
        p->topo++;
        return true;
    }
    return false;
}
```

Remover (ou desempilhar)

A operação de remoção de um elemento da pilha estática é bastante simples: precisamos apenas verificar se a pilha possui ao menos um elemento. Em seguida, decrementamos o valor da variável `topo`.

```
bool desempilhar(PILHAESTATICA *p, REGISTRO *r){
    if(p->topo > 0){
        *r = p->itens[p->topo - 1];
        p->topo--;
        return true;
    }
    return false;
}
```

Imprimir

A impressão de uma pilha estática envolve o uso de um laço de repetição, como `for` ou `while`. A diferença é que, ao contrário do habitual, o laço deve iniciar no topo da pilha e iterar até o primeiro elemento da pilha, representado pelo índice `0` do vetor.

```
void imprimir(PILHAESTATICA *p){
    for(int i = p->topo - 1; i >= 0; i--){
        printf("%d\n", p->itens[i].chave);
    }
}
```

👩‍💻 Testando a pilha

Para testar a pilha, podemos implementar o `main` da seguinte forma:

```c
int main(void){
    PILHAESTATICA p;
    inicializarPilhaEstatica(&p);
    empilhar(&p, (REGISTRO){10});
    empilhar(&p, (REGISTRO){20});
    empilhar(&p, (REGISTRO){30});
    empilhar(&p, (REGISTRO){40});
    empilhar(&p, (REGISTRO){50});

    REGISTRO d;

    if(desempilhar(&p, &d)){
        printf("Foi desempilhado: %d\n", d.chave);
    }

    imprimir(&p);
    return 0;
}
```

Arquivos
← Arquivos de texto

Estruturas
Pilha dinâmica →