

Pilha dinâmica

Entenda o mecanismo de implementação de pilhas com alocação dinâmica

Como visto na página sobre pilha estática, pilhas são estruturas de dados que adotam a política **LIFO** (Last in, first out). Assim, as operações de remoção e inserção acontecem em um único ponto: o topo da pilha.

Enquanto a implementação estática da pilha ocorre em um vetor, a implementação dinâmica tem como principal vantagem a possibilidade de se alocar mais memória em tempo de execução - e de liberar memória, se for necessário.



Declarando a pilha dinâmica

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>

typedef struct{
    int chave;
} REGISTRO;

typedef struct item{
    struct item *ant;
    REGISTRO r;
} ITEM;

typedef struct{
    ITEM* topo;
} PILHADINAMICA;
```



Funções de manipulação da pilha

Inicializar a pilha

```
void inicializar(PILHADINAMICA *p){  
    p->topo = NULL;  
}
```

A variável `topo` recebe o valor `NULL` por ser tratar de um ponteiro. Já a variável `tam` recebe o valor `0`.

Inserir (ou empilhar)

```
ITEM* criarItem(REGISTRO r){  
    ITEM* novo = (ITEM*)calloc(1, sizeof(ITEM));  
    if(novo){  
        novo->r = r;  
        novo->ant = NULL;  
    }  
    return novo;  
}
```

```
void empilhar(PILHADINAMICA *p, REGISTRO r){  
    ITEM* novo = criarItem(r);  
    if(!novo) return;  
    novo->ant = p->topo;  
    p->topo = novo;  
}
```

Remover (ou desempilhar)

```
ITEM* desempilhar(PILHADINAMICA *p){  
    if(p->topo){  
        ITEM *exc = p->topo;  
        p->topo = p->topo->ant;  
    }
```

```
        return exc;
    }
    return NULL;
}
```

Imprimir

```
void imprimir(PILHADINAMICA *p){
    ITEM *aux = p→topo;
    while(aux){
        printf("%d\n", aux→r.chave);
        aux = aux→ant;
    }
}
```

Liberar a pilha

```
void liberarPilha(PILHADINAMICA *p){
    ITEM *aux = p→topo;
    while(aux){
        ITEM* exc = aux;
        aux = aux→ant;
        free(exc);
    }
    p→topo = NULL;
}
```



Testando a pilha

```
int main(void){
    PILHADINAMICA d;

    inicializar(&d);
    empilhar(&d, (REGISTRO){10});
    empilhar(&d, (REGISTRO){20});
    empilhar(&d, (REGISTRO){30});
}
```

```
ITEM *i = desempilhar(&d);
if(i) {
    printf("Removido: %d\n", i->r.chave);
    free(i);
}

empilhar(&d, (REGISTRO){40});
imprimir(&d);

liberarPilha(&d);
}
```

Estruturas

← Pilha estática

Estruturas

Fila estática →