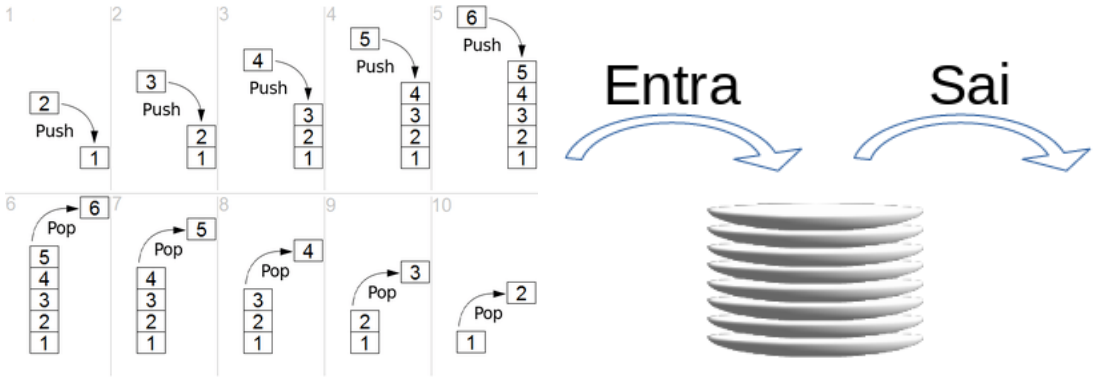
Pilhas Dinâmicas em C

Material de suporte para a vídeo aula da estrutura de dados árvore

1_ O que são Pilhas Dinâmicas?

- Uma pilha (em inglês, "stack") é uma estrutura de dados que segue o princípio de Last In, First Out (LIFO), ou seja, o último elemento inserido é o primeiro a ser removido. Isso significa que você só pode acessar o elemento mais recentemente adicionado à pilha.



2_ Estrutura da Pilha Dinâmica:

- 2.1 NODO

```
typedef struct nodo *ptrNodo;

struct nodo {
   elemento elem;
   ptrNodo prox;
};

typedef struct {
   ptrNodo topo;
   int tamanho;
} pilha;
```

```
typedef pilha tipo_pilha;
```

struct nodo: É uma estrutura que representa um nó da pilha.
elemento elem: Representa o elemento armazenado no nó da pilha.
ptrNodo prox: É um ponteiro para o próximo nó na pilha.
typedef struct nodo *ptrNodo;: Define um novo tipo ptrNodo que é um ponteiro para a estrutura struct nodo. Isso é comum em C para simplificar a declaração de ponteiros para essa estrutura.
tipo_pilha: É o novo tipo criado a partir de pilha. Isso é feito para fornecer um nome mais descritivo para o tipo.

- 2.2 MÉTODO CRIA PILHA:

```
void criarPilha(pilha *p) {
   p->tamanho = 0;
   p->topo = NULL;
}
```

Este método instância a estrutura pilha,
p->tamanho = 0 indica que o tamanho da pilha inicialmente é igual a zero.
p->topo = NULL indica que, inicialmente, o topo da pilha será nulo.

- 2.3 MÉTODO EMPILHAR:

```
int empilharElemento(pilha *p, elemento e) {
   ptrNodo pnodo;
   pnodo = (ptrNodo)malloc(sizeof(struct nodo));

   if (pnodo == NULL) {
      return 0; // Falha na alocação de memória
   } else {
      pnodo->elem = e;
      pnodo->prox = p->topo;
      p->topo = pnodo;
      p->tamanho++;
      return 1; // Sucesso ao empilhar o elemento
   }
}
```

O método de empilhar é responsável por inserir os elementos na pilha. Nesta função, passamos por referência a pilha e o elemento que será adicionado.
ptrNodo pnodo;
pnodo = (ptrNodo)malloc(sizeof(struct nodo)); Alocando memória para um ponteiro para o nodo.

f (pnodo == NULL) {
        return 0; // Falha na alocação de memória
    } tratamento caso haja falha em alocar memória para o nó.

   pnodo->elem = e;
        pnodo->prox = p->topo;
        p->topo = pnodo;
        p->tamanho++;
        return 1; caso a alocação seja bem sucedida, o elemento é passado para o nó penedo criado pnodo->elem = e; e o ponteiro para o próximo passa a apontar para o topo atual da pilha pnodo->prox = p->topo; . Por fim, o topo da pilha aponta para o nó recém adicionado

- 2.4 MÉTODO PILHA VAZIA:

```
int pilhaVazia(pilha p) {
   return (p.topo == NULL);
}
```

Este método verifica se a pilha está vazia, fazendo uma simples verificação do topo da pilha, caso p.topo == NULL, retorna verdadeiro, senão retorna false;

- 2.5 MÉTODO OBTER TOPO(TOP):

```
elemento obterTopo(pilha p) {
   if (!pilhaVazia(p)) {
      return p.topo->elem;
   } else {
      // Valor padrão ou sinal de erro para o tipo int
      return 0;
   }
}
```

Este método verifica inicialmente se a pilha não está vazia, caso não esteja, retorna o elemento armazenado no topo da pilha sem retirá-lo da pilha.

- 2.6 MÉTODO DESEMPILHAR:

```
int desempilharElemento(pilha *p, elemento *e) {
   ptrNodo pnodo;

   if (pilhaVazia(*p)) {
      return 0; // Pilha vazia, operação de desempilhar falhou
```

```
    } else {
        pnodo = p->topo;
        p->topo = p->topo->prox;
        *e = pnodo->elem;
        free(pnodo);
        p->tamanho--;
        return 1; // Sucesso ao desempilhar o elemento
    }
}
```

O método de desempilhar consiste em remover o topo da pilha. Primeiro, é feita uma verificação para saber se a pilha está vazia, caso não esteja, passamos o topo da pilha para um nó `pnodo = p->topo;` , depois o ponteiro para o topo aponta para o próximo do topo `p->topo = p->topo->prox;` . O elemento, passado por referência recebe o elemento que antes estava no topo da pilha `*e = pnodo->elem;` . Por fim, é feita uma liberação de memória `free(pnodo);` e redução do tamanho da pilha `p->tamanho--;`.

EXERCÍCIOS:

Agora com base na vídeo aula e no material de suporte responda os exercícios abaixo:

1. A estrutura de dados Pilha é baseada no princípio do(a):

   (A) último que entra é o primeiro que sai.

   (B) primeiro que entra é o primeiro que sai.

   (C) ordem de entrada não alterar a ordem de saída.

   (D) primeiro que sai é o primeiro que entra.

   (E) último que sai é o último que entra.

2. O elemento de dados **A** encontra-se no topo de uma pilha e o elemento **B** na base quando **C** e **D** são, nessa ordem, inseridos. Em seguida, os dois elementos retirados serão:

   (A) D e C.

   (B) B e C.

   (C) A e D.