

• Definição da estrutura do 'No' da Pilha:

```
typedef struct No {  
    int data;  
    struct No* next;  
} No;
```

• Definição da estrutura da Pilha:

```
typedef struct {  
    No* top;  
} Stack;
```

• Função Para criar uma nova Pilha vazia:

```
Stack* createStack() {  
    Stack* stack =  
        (Stack*) malloc(sizeof(Stack));  
    if (stack == NULL) {  
        printf("Erro: Falha na alocação  
            de memória para a Pilha.\n");  
        exit(EXIT_FAILURE);  
    }  
    stack->top = NULL;  
    return stack;  
}
```

• Função Para verificar se a Pilha está vazia:

```
int isEmpty(Stack* stack) {  
    return stack->top == NULL;  
}
```


• Função para empilhar um elemento na pilha:

```
void Push(Stack* stack, int data) {
    Node* newNode =
    (Node*) malloc(sizeof(Node));
    if (newNode == NULL) {
        printf("Erro: Falha na alocação de memória  

        para o nó da pilha.\n");
        exit(EXIT_FAILURE);
    }
    newNode->data = data;
    newNode->next = stack->top;
    stack->top = newNode;
}
```

• Função para desempilhar um elemento da pilha:

```
int Pop(Stack* stack) {
    if (isEmpty(stack)) {
        printf("Erro: A pilha está vazia.\n");
        exit(EXIT_FAILURE);
    }
    Node* temp = stack->top;
    int data = temp->data;
    stack->top = temp->next;
    return data;
}
```

• Função para buscar um elemento na pilha de forma recursiva:

```
int searchRecursive(Node* current, int target) {
    if (current == NULL) {
        return 0; // Elemento não encontrado
    }
}
```



```

IF (current->data == target) {
    Return 1; // Elemento encontrado
}
return searchRecursiva(current->
    next, target);
}

```

• Função para buscar um elemento na Pilha:

```

int search(Stack* stck, int target) {
    return searchRecursiva(stck->top, target);
}

```

• Função para exibir os elementos da Pilha:

```

void display(Stack* stck) {
    IF (isEmpty(stck)) {
        printf("A Pilha está vazia\n");
    }
}

```

```

Node* current = stck->top;
printf("Elementos da Pilha:");
while (current != NULL) {
    printf("%d", current->data);
    current = current->next;
}
printf("\n");
}

```

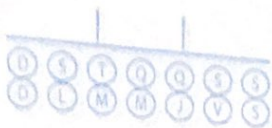
• Função Principal:

```

int main() {
    Stack* stck = createStack();

    Push(stck, 10);
    Push(stck, 20);
    Push(stck, 30);
    Push(stck, 40);
}

```



```
Display(lstok);
```

```
int target = 20;
```

```
if (search(lstok, target)) {
```

```
    printf("O elemento %d está na lista.\n", target);
```

```
} else {
```

```
    printf("O elemento %d não está na lista.\n", target);
```

```
    printf("Desembarcando um elemento: %d.\n",
```

```
    pop(lstok));
```

```
Display(lstok);
```

```
free(lstok); // Libera a memória alocada
```

```
para a lista
```

```
return 0; }
```