

Pilhas

Prof. Denio Duarte

Prof. Geomar Schreiner

Pilha

- Pilhas são estruturas de dados baseadas em lista simples.
 - Abordagem LIFO: last-in first-out, ou seja, o último a entrar é o primeiro a sair
- Operações básicas
 - **Inserir** um novo item no topo da pilha (push)
 - **Remover** um item do topo da pilha (pop)
- É comum também podermos realizar outras operações úteis em uma pilha:
 - **Inicializar** a pilha
 - **Testar** se a pilha está **vazia**
 - **Destruir** a pilha (tipicamente, liberar a memória alocada para a estrutura de dados)

Pilha

- Usos
 - Controle de chamada de funções dentro dos programas, a última função chamada é a primeira a ser destruída
 - Controle de prioridade nas operações matemáticas: (, {, [e operadores

Exemplo de funcionamento

Entrada:

L A * S T I * N ...

Processamento:

- Se é uma letra: insere na pilha
- Se é um * : remove da pilha

Saída:



Pilha



Pilha vazia

Exemplo de funcionamento

Entrada:

A * S T I * N ...

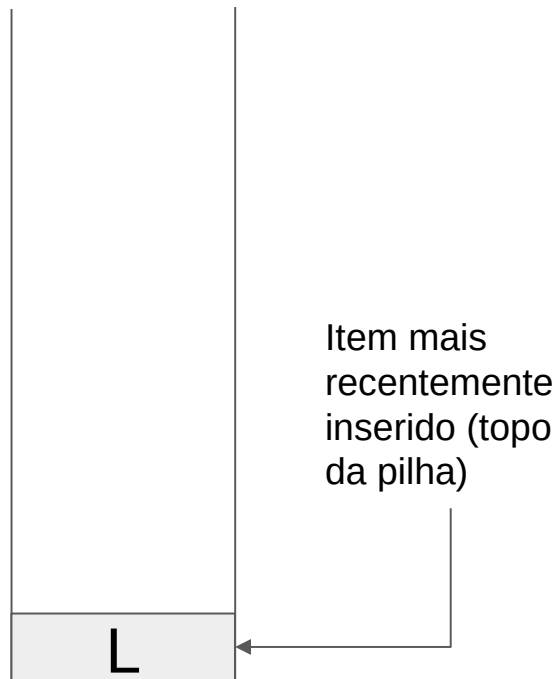
Processamento:

- Se é uma letra: insere na pilha
- Se é um * : remove da pilha

Saída:



Pilha



Exemplo de funcionamento

Entrada:

* S T I * N ...

Processamento:

- Se é uma letra: insere na pilha
- Se é um * : remove da pilha

Saída:



Pilha



Item mais recentemente inserido (topo da pilha)

Exemplo de funcionamento

Entrada:

S T I * N ...

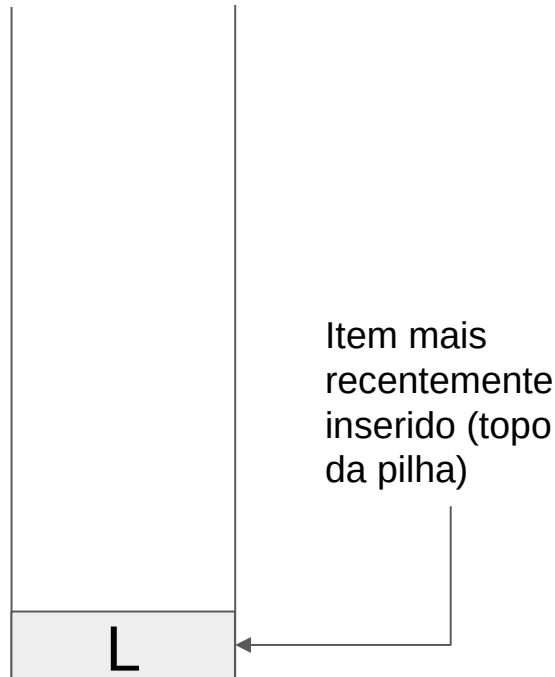
Processamento:

- Se é uma letra: insere na pilha
- Se é um * : remove da pilha

Saída:

A

Pilha



Exemplo de funcionamento

Entrada:

T I * N ...

Processamento:

- Se é uma letra: insere na pilha
- Se é um * : remove da pilha

Saída:

A

Pilha



Item mais recentemente inserido (topo da pilha)

Exemplo de funcionamento

Entrada:

I * N ...

Processamento:

- Se é uma letra: insere na pilha
- Se é um * : remove da pilha

Saída:

A

Pilha



Item mais recentemente inserido (topo da pilha)

Exemplo de funcionamento

Entrada:

* N ...

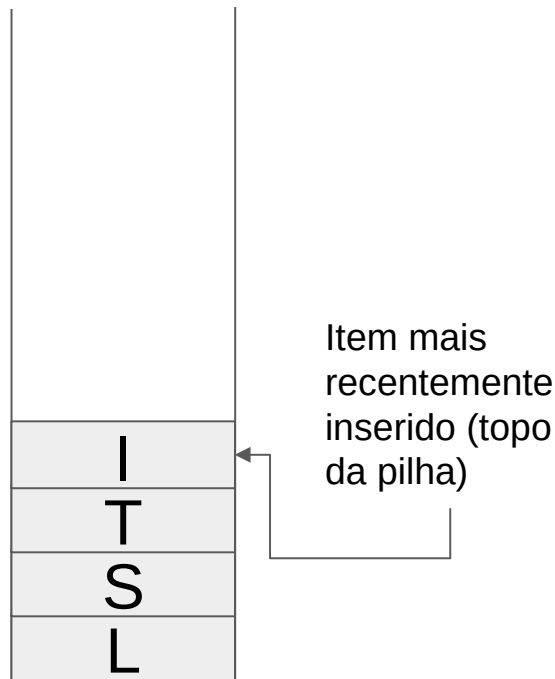
Processamento:

- Se é uma letra: insere na pilha
- Se é um * : remove da pilha

Saída:

A

Pilha



Exemplo de funcionamento

Entrada:

N ...

Processamento:

- Se é uma letra: insere na pilha
- Se é um * : remove da pilha

Saída:

A I

Pilha



Item mais recentemente inserido (topo da pilha)



Exemplo de funcionamento

Entrada:

...

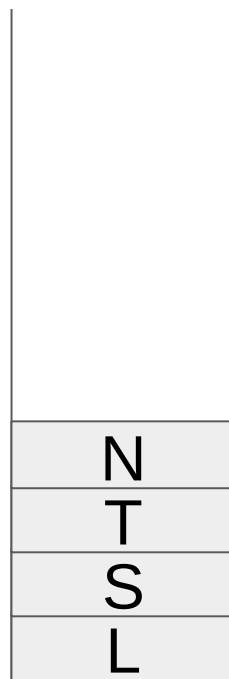
Processamento:

- Se é uma letra: insere na pilha
- Se é um * : remove da pilha

Saída:

Al

Pilha



Item mais recentemente inserido (topo da pilha)

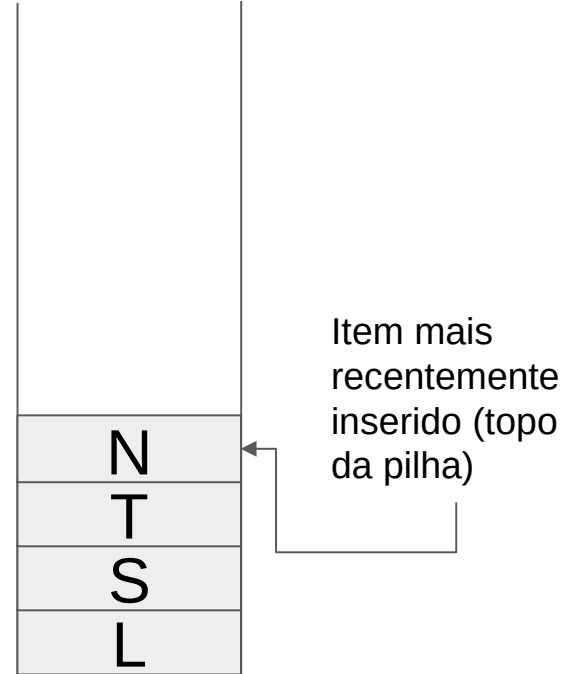
Exemplo de funcionamento

Entrada:

...

O processamento continuaria...

Pilha



Exemplo de aplicação

Problema: Verificar se as chaves, colchetes e parênteses de um programa estão corretamente emparelhados.

Entrada (após pré-processamento):

`() { [] }`

Saída:

Correto

Entrada (após pré-processamento):

`() [{ }]`

Saída:

Incorreto

Exemplo de aplicação

Entrada:

`() { [] }`

Solução:

- Se é (, { ou [: insere na pilha
- Se é), } ou]: remove da pilha e compara se deu match



Saída:



Pilha



Pilha vazia

Exemplo de aplicação

Entrada:

) { [] }

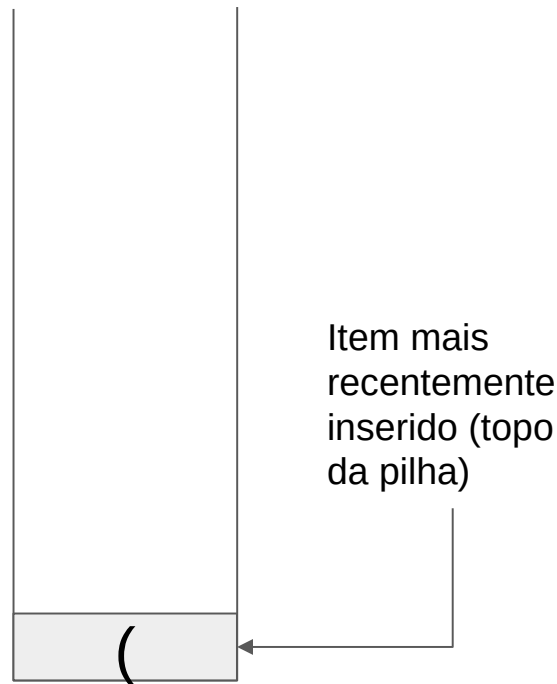
Solução:

- Se é (, { ou [: insere na pilha
- Se é), } ou]: remove da pilha e compara se deu match

Saída:



Pilha



Exemplo de aplicação

Entrada:

{ [] }

Solução:

- Se é (, { ou [: insere na pilha
- Se é), } ou]: remove da pilha e compara se deu match



Saída:



Pilha



Pilha vazia

Exemplo de aplicação

Entrada:

[] }

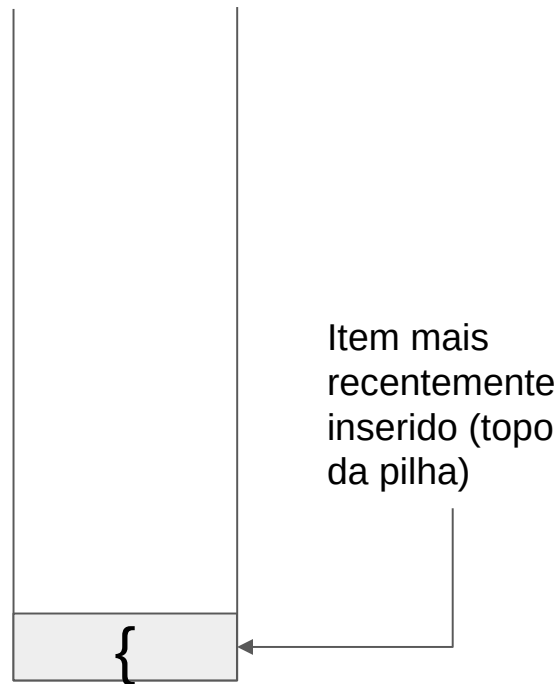
Solução:

- Se é (, { ou [: insere na pilha
- Se é), } ou]: remove da pilha e compara se deu match

Saída:



Pilha



Exemplo de aplicação

Entrada:

] }

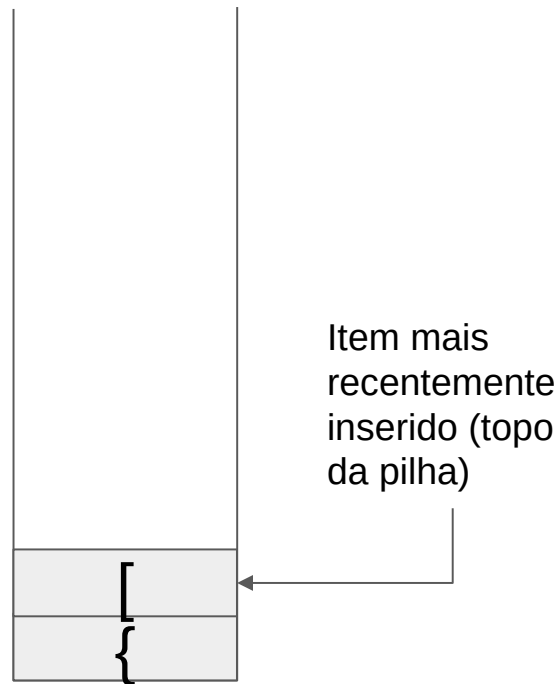
Solução:

- Se é (, { ou [: insere na pilha
- Se é), } ou]: remove da pilha e compara se deu match

Saída:



Pilha



Exemplo de aplicação

Entrada:

}

Solução:

- Se é (, { ou [: insere na pilha
- Se é), } ou]: remove da pilha e compara se deu match



Saída:



Pilha



Item mais recentemente inserido (topo da pilha)

Exemplo de aplicação

Entrada:

Solução:

- Se é (, { ou [: insere na pilha
- Se é), } ou]: remove da pilha e compara se deu match



Saída:



Pilha



Pilha vazia

Exemplo de aplicação

Entrada:

Solução:

- Se é (, { ou [: insere na pilha
- Se é), } ou]: remove da pilha e compara se deu match

Saída:

Correto

Pilha

Pilha vazia

Implementação

- Comumente, uma pilha é implementada de duas maneiras: usando um vetor ou usando uma lista encadeada simples
- Usando um vetor
 - Desvantagem: É necessário definir um tamanho máximo da pilha; uso ineficiente da memória total alocada
 - Vantagem: Inserção e remoção de itens não requerem alocação e liberação de memória
- Usando uma lista encadeada simples
 - Desvantagem: Inserção e remoção de itens requerem alocação e liberação de memória
 - Vantagem: Uso mais eficiente da memória total alocada

Implementação - lista encadeada simples

- Podemos declarar os seguintes tipos:

```
typedef int Item;
```

```
typedef struct elemPilha {  
    Item item;  
    struct elemPilha *proximo;  
} ElemPilha;
```

```
typedef struct {           // ou  
    ElemPilha *topo;       // ElemPilha *top  
} Pilha;                  //
```


Implementação - lista encadeada simples

- Podemos declarar os seguintes tipos:

```
typedef int Item;
```

```
typedef struct elemPilha {  
    Item item;  
    struct elemPilha *proximo;  
} ElemPilha;
```

```
typedef struct {  
    ElemPilha *topo;  
} Pilha;
```

```
Pilha *pilha;
```

OU

```
Pilha *pilha, *topo;
```

Implementação - lista encadeada simples

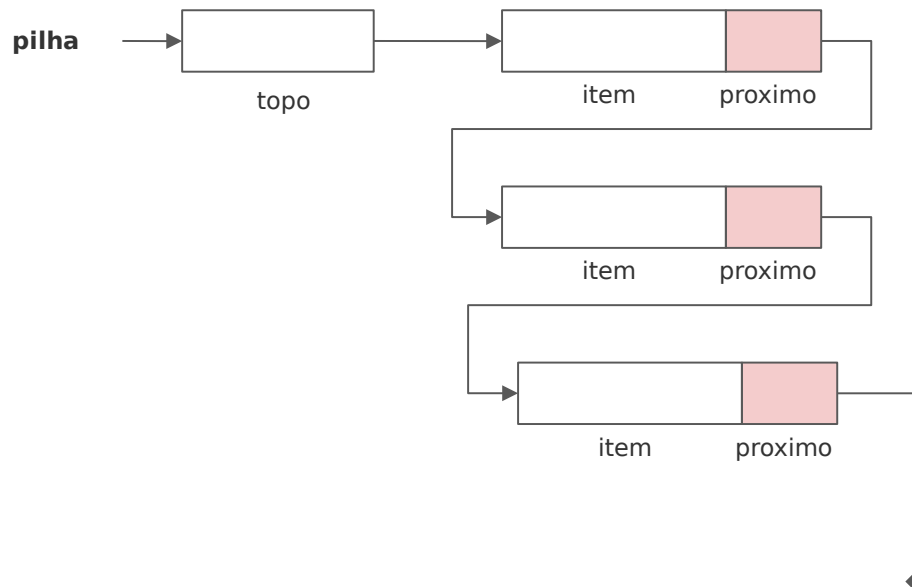
- Podemos declarar os seguintes tipos:

```
typedef int Item;
```

```
typedef struct elemPilha {  
    Item item;  
    struct elemPilha *proximo;  
} ElemPilha;
```

```
typedef struct {  
    ElemPilha *topo;  
} Pilha;
```

```
Pilha *pilha;
```



Implementação - lista encadeada simples

- Operações:

```
void inserePilha(Pilha *pilha, Item item)
```

```
void removePilha(Pilha *pilha, Item *item)
```

```
void inicializaPilha(Pilha *pilha)
```

```
int pilhaVazia(Pilha *pilha)
```

```
void liberaPilha(Pilha *pilha)
```

Implementação - lista encadeada simples

- Operação de inserir um novo elemento na pilha:

```
void inserePilha(Pilha *pilha, Item item) {  
    ElemPilha *aux;  
  
    // Cria um novo elemento da lista encadeada que representa a pilha e  
    // armazena neste novo elemento o item a ser inserido na pilha  
    aux = malloc(sizeof(ElemPilha));  
    aux->item = item;  
  
    // Insere o novo elemento no inicio da lista encadeada que representa a  
    // pilha  
    aux->proximo = pilha->topo;  
    pilha->topo = aux;  
}
```

Implementação - lista encadeada simples

- Operação de remover um elemento da pilha:

```
void removePilha(Pilha *pilha, Item *item) {  
    ElemPilha *aux;  
  
    // Verificar se a pilha esta vazia!  
  
    // Armazena o item a ser removido da pilha  
    *item = pilha->topo->item; // ATENCAO: Depende da definicao do tipo do item  
  
    // Armazena o primeiro elemento da lista encadeada que representa a pilha  
    // e remove este primeiro elemento da lista  
    aux = pilha->topo;  
    pilha->topo = pilha->topo->proximo;  
  
    // Libera a memoria alocada para o elemento removido  
    free(aux);  
}
```

Implementação - lista encadeada simples

- Operação de inicializar a pilha:

```
void inicializaPilha(Pilha *pilha) {  
    pilha->topo = NULL;  
}
```

Implementação - lista encadeada simples

- Operação de testar se a pilha está vazia:

```
int pilhaVazia(Pilha *pilha) {  
    return (pilha->topo == NULL);  
}
```

Implementação - lista encadeada simples

- Operação de destruir a pilha (liberar a memória alocada para a pilha):

```
void liberaPilha(Pilha *pilha) {  
    ElemPilha *aux;  
  
    while (pilha->topo != NULL) {  
        // Armazena o primeiro elemento da lista encadeada que representa a  
        // pilha e remove este primeiro elemento da lista  
        aux = pilha->topo;  
        pilha->topo = pilha->topo->proximo;  
  
        // Libera a memoria alocada para o elemento removido  
        free(aux);  
    }  
}
```


Implementação - lista encadeada simples

- Usando a pilha:

```
int main() {  
    Pilha pilha;  
    Item item;  
  
    inicializaPilha(&pilha);  
  
    for (int i = 0; i < 10; i++) {  
        item = i;  
  
        printf("Inserindo na pilha o item %d.\n", item);  
        inserePilha(&pilha, item);  
    }  
  
    // Continua no proximo slide
```

Implementação - lista encadeada simples

- Usando a pilha:

```
// Continuacao do slide anterior
```

```
while (pilhaVazia(&pilha) == 0) {  
    removePilha(&pilha, &item);  
    printf("Item %d removido da pilha.\n", item);  
}
```

```
liberaPilha(&pilha); // Sem efeito se a pilha ja estiver vazia
```

```
return 0;  
}
```

Exercícios

1. Faça um programa que receba uma string, caracter por caracter
 - Cada caracter é colocado em uma pilha
 - No fim da entrada, esvazie a pilha imprimindo os caracteres armazenados