Pilhas

Disciplina: Estrutura de Dados

Luciano Moraes Da Luz Brum

Universidade Federal do Pampa – Unipampa – Campus Bagé

Email: lucianobrum18@gmail.com

Tópicos

- O que é uma pilha?
- Aplicações.
- Interface do tipo Pilha.
- Implementação de pilhas com vetor (contiguidade física).
- Implementação de pilhas com listas encadeadas (alocação dinâmica).
- Resumo.

O que é uma pilha?

- Pilha é uma estrutura de dados em que:
  - O último elemento inserido é o primeiro a ser retirado da pilha.
- **Analogia: Pilha de pratos para lavar !**

O que é uma pilha?

O que é uma pilha?

- Ideia fundamental: O acesso aos elementos da pilha são feitos a partir do topo. Isso implica que:

- **Quando um elemento é inserido, ele se torna o topo da pilha.**

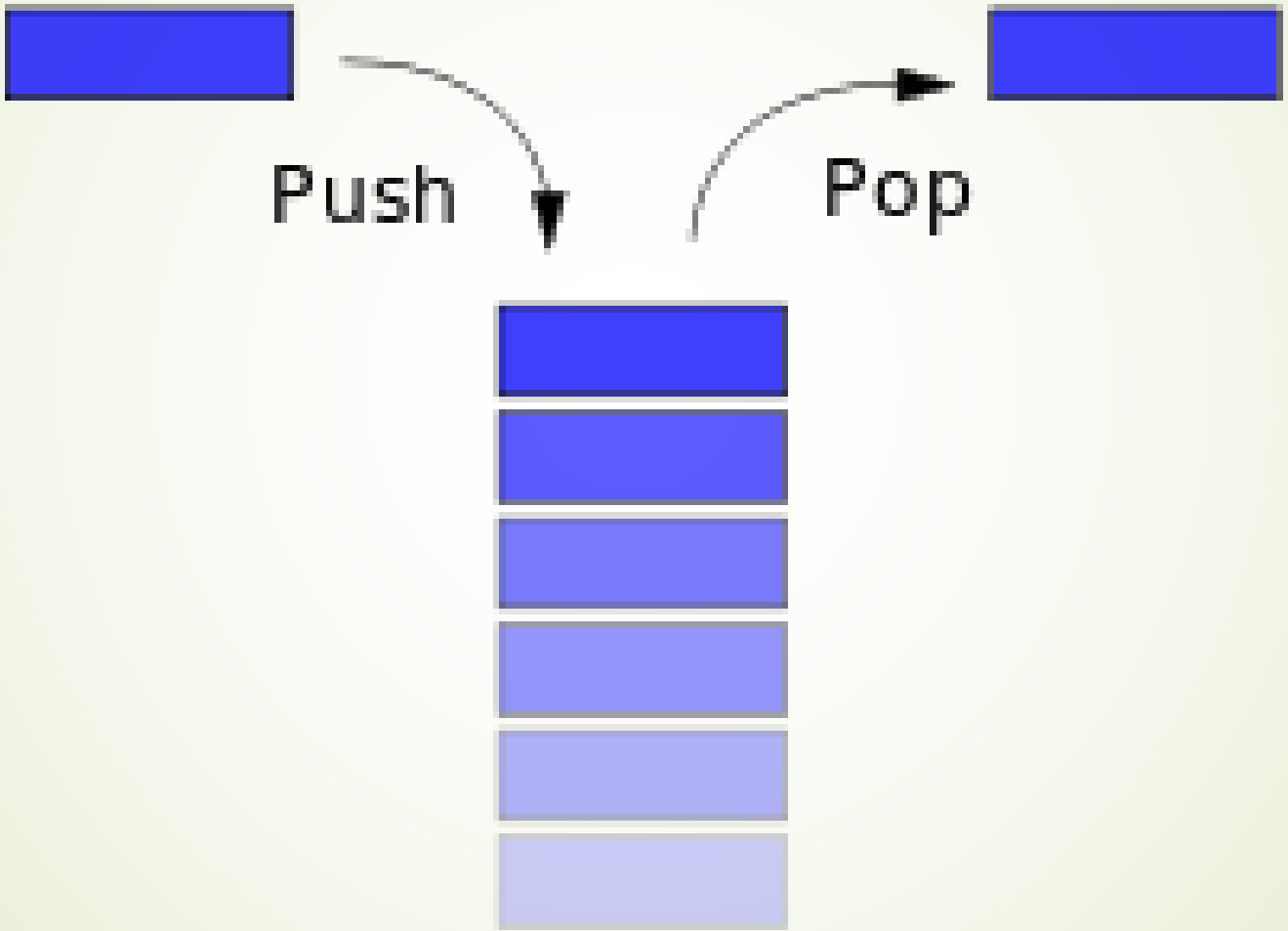
- **O único elemento que pode ser retirado é o do topo.**

- **Segue estratégia LIFO (Last in – First out).**

O que é uma pilha?

- A pilha possui duas operações básicas:
  - **Operação para empilhar um elemento, inserindo-o no topo (Push);**
  - **Operação para desempilhar um elemento, retirando-o do topo (Pop);**

O que é uma pilha?



Tópicos

- ~~O que é uma pilha?~~
- **Aplicações.**
- **Interface do tipo Pilha.**
- **Implementação de pilhas com vetor (contiguidade física).**
- **Implementação de pilhas com listas encadeadas (alocação dinâmica).**
- **Resumo.**

Aplicações

- **Pilha de execução da linguagem C:**
  - Variáveis locais de funções são dispostas em uma pilha.
  - Uma função só tem acesso às variáveis da função que está no topo (não é possível acessar variáveis da função locais às outras funções.

Aplicações

- **Calculadoras da HP:**
  - Expressões pós-fixadas: para avaliar expressões como (1+2)*(4-1), podemos digitar 1 2 + 4 1 - *. Operandos, operadores e resultados são empilhados e desempilhados todo instante.

- **Compiladores:**
  - Muitos compiladores utilizam pilhas para análise sintática de expressões, blocos de programas e afins.

Tópicos

- ~~O que é uma pilha?~~
- ~~Aplicações.~~
- **Interface do tipo Pilha.**
- **Implementação de pilhas com vetor (contiguidade física).**
- **Implementação de pilhas com listas encadeadas (alocação dinâmica).**
- **Resumo.**

Interface do tipo Pilha

- **Vamos considerar a implementação de 5 operações:**
  - Criar pilha vazia;
  - Inserir elemento no topo (*push*);
  - Remover elemento do topo (*pop*);
  - Verificar se a pilha está vazia;
  - Liberar a estrutura da pilha;

Interface do tipo Pilha

- **Podemos criar o arquivo pilha.h, que representa a interface da pilha:**

```
typedef struct pilha Pilha;
Pilha* pilha_cria (void);
void pilha_push (Pilha *p, float v);
float pilha_pop (Pilha *p);
int pilha_vazia (Pilha *p);
void pilha_libera (Pilha *p);
```

Tópicos

- ~~O que é uma pilha?~~
- ~~Aplicações.~~
- ~~Interface do tipo Pilha.~~
- **Implementação de pilhas com vetor (contiguidade física).**
- **Implementação de pilhas com listas encadeadas (alocação dinâmica).**
- **Resumo.**

Implementação de Pilhas com Vetor

- Se sabemos de antemão o número máximo de elementos na pilha, podemos implementar a pilha com **vetores.**

- Se vamos armazenar os elementos em um vetor, podemos dizer que elementos inseridos ocupam posições iniciais do vetor.

- O elemento do topo será o último elemento armazenado (posição n-1).

Implementação de Pilhas com Vetor

- Podemos ter a seguinte representação para o elemento pilha:

```
#define TAM 50
struct pilha{
    int n;
    float vet[TAM];
}
```

Tamanho do vetor

N° de elementos

Elementos

Implementação de Pilhas com Vetor

- A função para criar a pilha:

```
Pilha* pilha_cria (void){
    Pilha *p = (Pilha*) malloc(sizeof(Pilha));
    p->n = 0;
    return p;
}
```

Implementação de Pilhas com Vetor

- A função para inserir um elemento na pilha (push – empilhar):

```
void pilha_push (Pilha *p, float v){
        if(p->n == TAM){
                printf("Capacidade da pilha estourou.\n");
                exit(1);
        }
        p->vet[p->n]=v;
        p->n++;
}
```

Implementação de Pilhas com Vetor

- A função para retirar um elemento na pilha (pop – desempilhar):

```
float pilha_pop (Pilha *p){
        float v;

        if(pilha_vazia(p)){
                printf("Pilha já vazia.\n");
                exit(1);
        }

        v = p->vet[(p->n)-1];
        p->n--;
        return v;

}
```

Implementação de Pilhas com Vetor

- A função para verificar se a pilha está vazia:

```
int pilha_vazia (Pilha *p){
    return (p->n == 0); //1 = verdadeiro e 0 = falso
}
```

- A função para liberar a pilha da memória:

```
void pilha_libera(Pilha *p){
    free(p);
}
```

Tópicos

- ~~O que é uma pilha?~~
- ~~Aplicações.~~
- ~~Interface do tipo Pilha.~~
- ~~Implementação de pilhas com vetor (contiguidade física) .~~
- **Implementação de pilhas com listas encadeadas (alocação dinâmica).**
- **Resumo.**

Implementação de Pilhas com Lista

- Se não sabemos de antemão o número máximo de elementos na pilha, podemos implementar a pilha com uma **estrutura de dados dinâmica**, por exemplo, uma **lista.**

- Os elementos são armazenados na lista e a pilha é apenas um ponteiro para o primeiro nó da lista.

Implementação de Pilhas com Lista

- Podemos ter a seguinte representação para os elementos lista e pilha:

```
struct lista{
    float info;
    struct lista *prox;
};
typedef struct lista Lista;
struct pilha{
    Lista *prim;
};
```

Implementação de Pilhas com Lista

- A função para criar a pilha:

```
Pilha* pilha_cria (void){
    Pilha *p = (Pilha*) malloc(sizeof(Pilha));
    p->prim = NULL;
    return p;
}
```

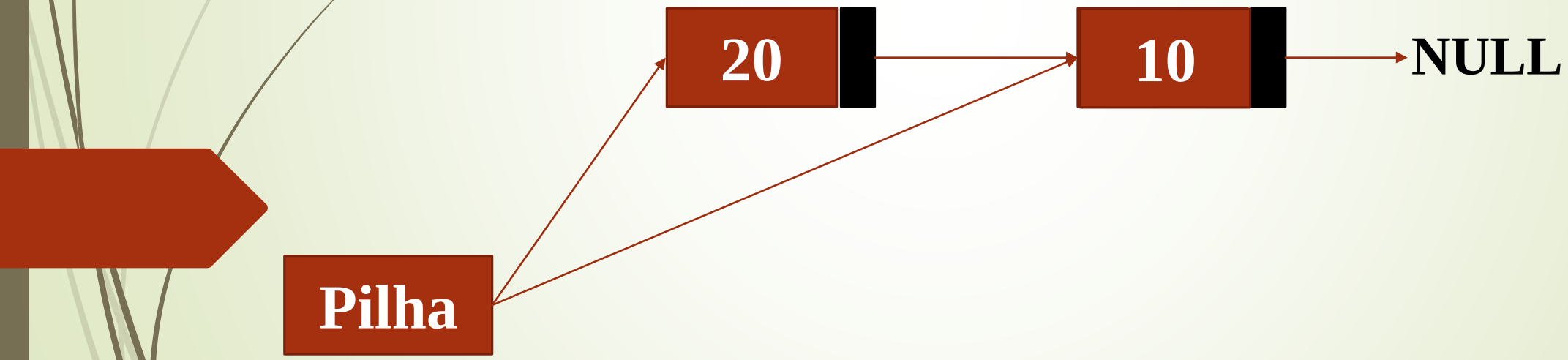
Implementação de Pilhas com Lista

- A função para inserir um elemento na pilha (push – empilhar):

```
void pilha_push (Pilha *p, float v){
    Lista *n= (Lista*) malloc(sizeof(Lista));
    n->info = v;
    n->prox = p->prim;
    p->prim = n;
}
```

Implementação de Pilhas com Lista

Inserindo o elemento 10 e depois o 20.



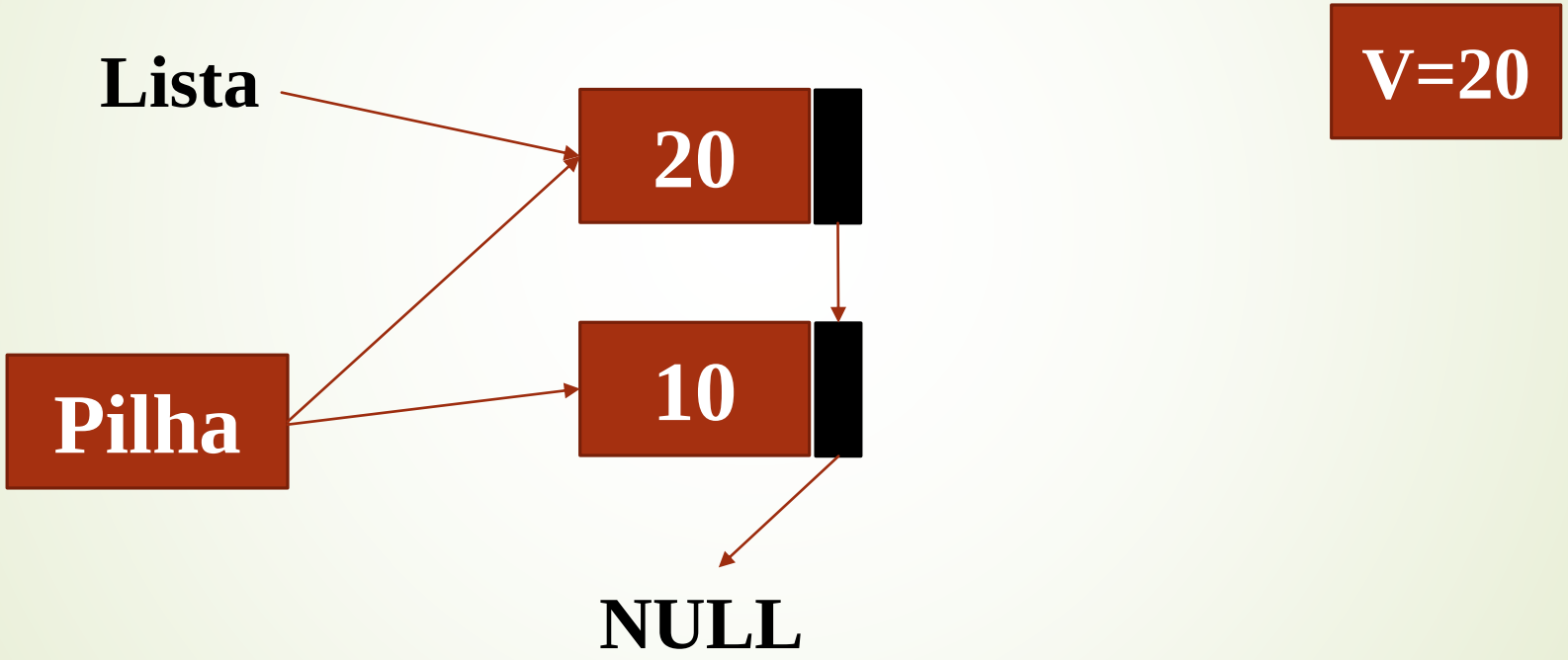
Implementação de Pilhas com Lista

- A função para retirar um elemento na pilha (pop – desempilhar):

```
float pilha_pop (Pilha *p){
    Lista *t;
    float v;
    if(pilha_vazia(p)){
            printf("Pilha Vazia!.\n");
            exit(1);
    }
    t = p->prim;
    v = t->info;
    p->prim = t->prox;
    free(t);
    return v;
}
```

Implementação de Pilhas com Lista

Retirando os elementos da pilha.



Implementação de Pilhas com Lista

- A função para verificar se a pilha está vazia:

```
int pilha_vazia (Pilha *p){
    return (p->prim == NULL); //1 = verdadeiro e 0 = falso
}
```

- A função para liberar a pilha deve também liberar a lista:

```
void pilha_libera(Pilha *p){
    Lista *q = p->prim;
    while(q!=NULL){
        Lista *t = q->prox;
        free(q);
        q = t;
    }
    free(p);
}
```

Implementação de Pilhas com Lista

- Talvez seja interessante termos uma função que mostre todos os elementos da pilha, para fins de teste:

```
void pilha_imprime(Pilha *p){
    int i;
    for(i = (p->n) -1; i>=0; i--){
        printf("%f\n", p->vet[i]);
    }
}
void pilha_imprime(Pilha *p){
    Lista *q;
    for(q = p->prim; q!=NULL; q=q->prox){
        printf("%f\n", q->info);
    }
}
```

Tópicos

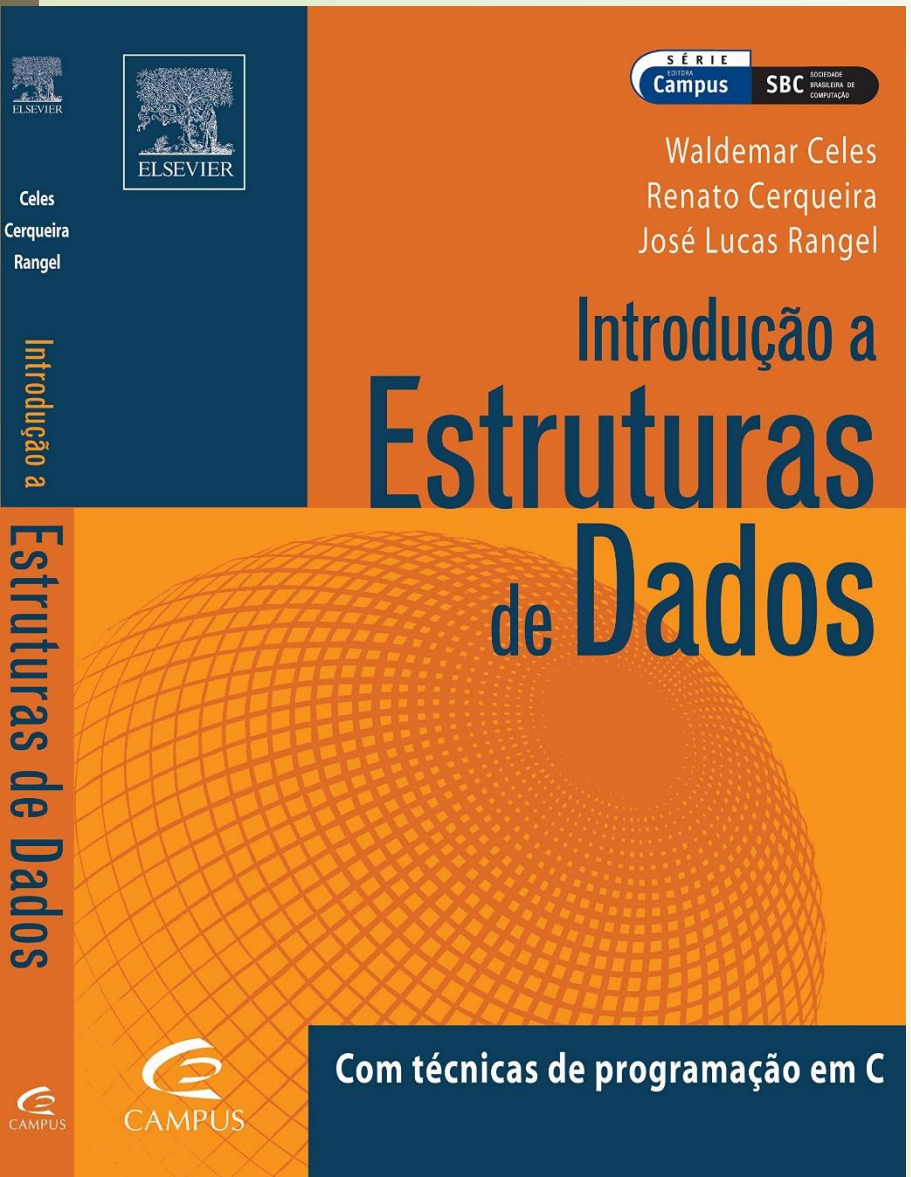
- ~~O que é uma pilha?~~
- ~~Aplicações.~~
- ~~Interface do tipo Pilha.~~
- ~~Implementação de pilhas com vetor (contiguidade física).~~
- ~~Implementação de pilhas com listas encadeadas (alocação dinâmica).~~
- **Resumo.**

Resumo

- **Foi demonstrado:**
  - **Funcionamento de uma pilha com vetor e lista;**
  - **Aplicações;**
  - **Operações básicas e interface;**
  - **Exemplos em C;**

Dúvidas ?

Referências



- CELES, Waldemar; CERQUEIRA, Renato; RANGEL, José Lucas. Introdução a Estruturas de Dados ~~c~~om técnicas de programação em C. Rio de Janeiro: Elsevier (Campus), 2004. 4ª Reimpressão. 294 p.