Estrutura de Dados 1

Estruturas lineares - Pilha e Fila

Prof. Lucas Boaventura
lucas.boaventura@unb.br

Filas

- **O que é uma fila?**
  - Fila é uma estrutura de dados dinâmica
  - Permite Inserção de elementos
  - Permite Remoção de elementos

- “dinâmica” aqui se refere ao comportamento da estrutura (entradas e saídas), não necessariamente à alocação dinâmica de memória em C (isso vem depois)

Filas

- A fila segue uma regra específica de operação
- A remoção sempre ocorre no elemento mais antigo
- não removemos qualquer elemento, apenas o que entrou primeiro.

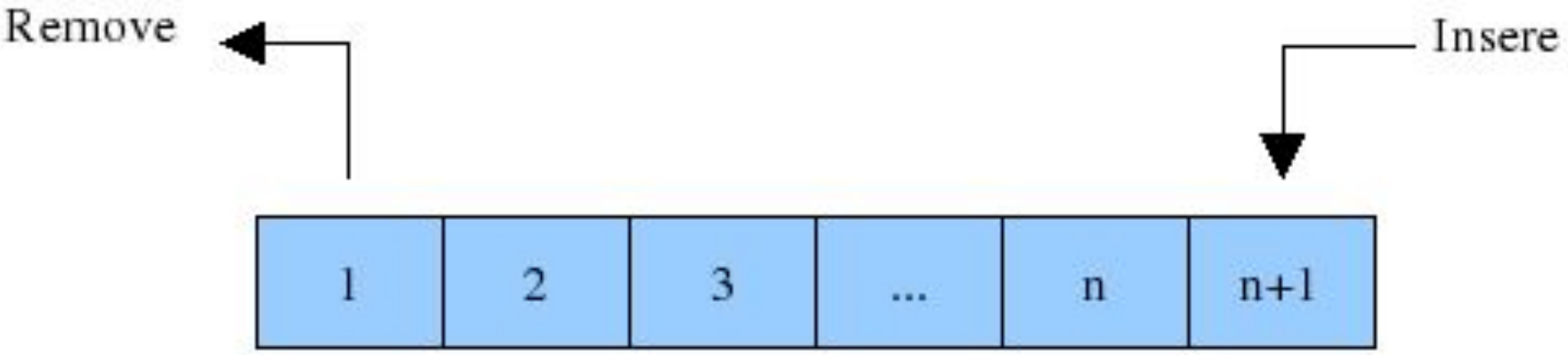
Filas

- FIFO – First In, First Out
- O primeiro elemento inserido é o primeiro a ser removido

- Situações reais similares:
  - Fila de Banco
  - Fila de processos no SO
  - Fila de pedidos de um restaurante

Filas

- Inserção ocorre no final da fila
- Remoção ocorre no início da fila



Filas em Vetores

- **Implementação em um vetor em C:**
  - fila[0..N-1]

- **O tipo dos elementos é irrelevante:**
  - inteiros;
  - bytes;
  - ponteiros;
  - structs, etc.

Filas em Vetores

- **A parte ocupada do vetor é:**
  - **fila[p .. u-1]**
  - p => posição do **primeiro elemento**
  - u => posição do **último elemento**

- p aponta para o próximo a sair
- u aponta para a próxima posição livre

Filas em Vetores

- **Fila vazia:**
  - p == u
- **Fila cheia:**
  - u == N (onde N é o tamanho do vetor)
- Lembrar que o vetor **anda** para a direita

Filas em Vetores

- A fila não começa necessariamente na posição 0
- o início “desloca” conforme removemos elementos

Filas em Vetores

- **Para remover um elemento da fila:**
  - Checar se a fila está vazia e depois remover

```
1 // 1ª opção
2 x = fila[p++];
3
4 // 2ª opção
5 x = fila[p];
6 p = p + 1;
```

Filas em Vetores

- **Para inserir um elemento y na fila:**
  - Inserção sempre ocorre no final, mesmo que a fila esteja vazia

```
1 // 1ª opção
2 fila[u++] = y;
3
4 // 2ª opção
5 fila[u] = y;
6 u = u + 1;
```

Filas em Vetores

- **Inserção só pode ocorrer se:**
  - u < N

- Caso contrário a fila está cheia (transborda) e ocorre um **overflow**

Filas em Vetores

- **Funções para encapsular a fila**
  - **Criar fila**
  - N é passado por clareza e padronização

```c
void criaFila(int fila[], int N, int *p, int *u) {
    *p = 0;
    *u = 0;
}
```

Filas em Vetores

- **Funções para encapsular a fila**
  - **Verifica se a fila está vazia**

```
1 int filaVazia(int p, int u) {
2     return (p == u);
3 }
```

Filas em Vetores

- **Funções para encapsular a fila**
  - **Verifica se a fila está cheia**

```
1 int filaCheia(int u, int N) {
2     return (u == N);
3 }
```

Filas em Vetores

- **Funções para encapsular a fila**
  - **Inserir elemento na fila**

```c
int colocanafila(int fila[], int N, int *u, int y) {
    if (filaCheia(*u, N)) return 0; // overflow

    fila[*u] = y;
    (*u)++;
    return 1; // sucesso
}
```

Filas em Vetores

- **Funções para encapsular a fila**
  - **Remover elemento na fila**

```
1 int tiradafila(int fila[], int *p, int u, int *x) {
2     if (filaVazia(*p, u)) return 0; // underflow
3
4     *x = fila[*p];
5     (*p)++;
6     return 1; // sucesso
7 }
```

```c
int main(void) {
    int fila[TAM];
    int p, u;
    int x;

    criaFila(fila, TAM, &p, &u);

    colocanafila(fila, TAM, &u, 10);
    colocanafila(fila, TAM, &u, 20);
    colocanafila(fila, TAM, &u, 30);

    for(int i = p; i < u; i++) printf("%d ", fila[i]);
    printf("\n--------------------\n");

    while (!filaVazia(p, u)) {
        tiradafila(fila, &p, u, &x);
        printf("Removido: %d\n", x);
    }

    for(int i = p; i < u; i++) printf("%d ", fila[i]);
    printf("--------------------\n");

    return 0;
}
```

Filas em Vetores

- Na implementação que fizemos, utilizando vetores, a fila anda para a direita no vetor

- Problema:
  - As posições não são reutilizadas
  - risco maior de overflow

- A fila pode ser implementada utilizando fila encadeada!

Pilha

- **O que é uma pilha?**
  - Pilha é uma estrutura de dados dinâmica
  - Permite Inserção de elementos
  - Permite Remoção de elementos

- “dinâmica” aqui se refere ao comportamento da estrutura (entradas e saídas), não necessariamente à alocação dinâmica de memória em C (isso vem depois)

Pilha

- **A pilha segue uma regra específica de operação**

- Sempre que ocorre uma remoção:
  - o elemento removido é o **mais recente**

- **Diferente da fila, aqui não removemos o mais antigo, mas sim o último que entrou**

Pilha

- **LIFO – Last In, First Out**
- **O último elemento inserido é o primeiro a ser removido**
- Essa é a regra fundamental da pilha, **só é possível retirar o que está no topo**
  - Pilha de pratos
  - Pilha de livros

Pilha

- **Pilha:**
  - **LIFO** (Last in, first out)
  - **Remove o elemento mais recente**

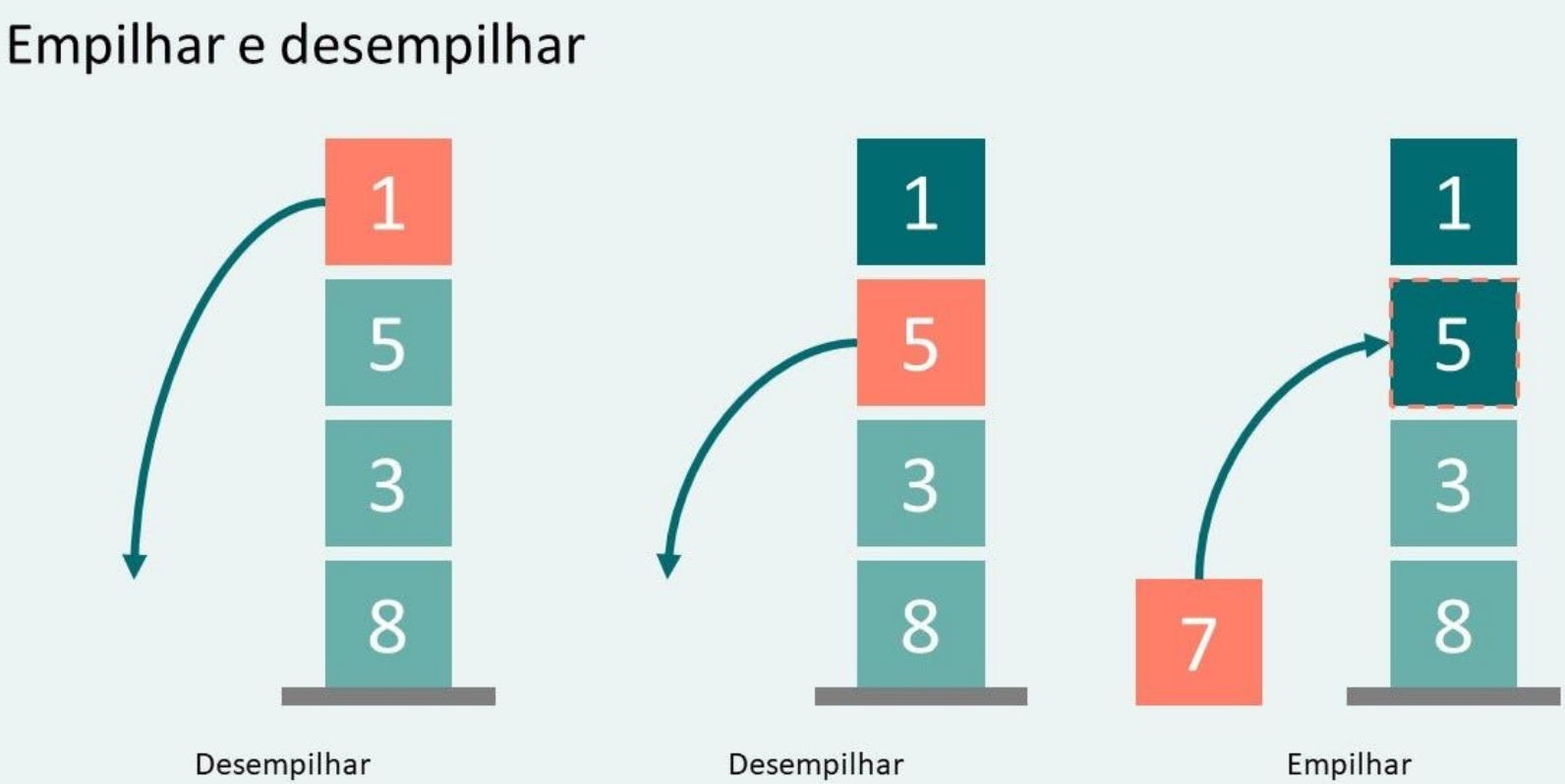
- **Fila:**
  - **FIFO** (first in, first out)
  - **Remove o elemento mais antigo**

Pilha

- **Dentro do conceito de estrutura de dados do tipo pilha costumamos chamar algumas operações por nomes específicos, como:**
  - Inserção: push
  - Remoção: pop
  - Extremidade: topo da pilha
- Diferente da fila, todas as operações acontecem em uma única extremidade.

Pilha

- **Exemplo:**



Pilha

- **Onde pilhas são usadas:**
  - **Chamadas de funções (call stack);**
  - **Algoritmos recursivos;**
  - **Desfazer/refazer (undo/redo),**
  - **etc.**

Pilha em Vetores

- **Pilha implementada em um vetor em C:**
  - pilha[0..N-1]

- **O tipo dos elementos é irrelevante:**
  - inteiros;
  - bytes;
  - ponteiros;
  - structs, etc.

Pilha em Vetores

- **A parte ocupada do vetor é:**
  - **pilha[0 … t-1]**
  - t => a **primeira posição livre**
  - t-1 => o **topo da pilha**

- Diferente da fila, a pilha cresce sempre a partir da posição 0

Pilha em Vetores

- **Pilha vazia:**
  - **t == 0**
- **Pilha cheia:**
  - **t == N**
- **A ideia de overflow e underflow continua**

Pilha em Vetores

- **Funções para encapsular a pilha**
  - **Criar pilha**

```
1 void criaPilha(int *t) {
2       *t = 0;
3 }
```

Pilha em Vetores

- **Funções para encapsular a pilha**
  - Verificar se a pilha está vazia

```
1 int pilhaVazia(int t) {
2     return (t == 0);
3 }
```

Pilha em Vetores

- **Funções para encapsular a pilha**
  - Verificar se a pilha está cheia

```
1 int pilhaCheia(int t, int N) {
2     return (t == N);
3 }
```

Pilha em Vetores

- **Funções para encapsular a pilha**
  - Empilhar elemento (push)
  - O topo cresce para a direita do vetor

```c
int empilha(int pilha[], int N, int *t, int y) {
    if (pilhaCheia(*t, N)) return 0; // overflow

    pilha[*t] = y;
    (*t)++;
    return 1; // sucesso
}
```

Pilha em Vetores

- **Funções para encapsular a pilha**
  - Desempilhar elemento (pop)
  - Decrementa e depois acessa o topo

```
1 int desempilha(int pilha[], int *t, int *x) {
2     if (pilhaVazia(*t)) return 0; // underflow
3
4     (*t)--;
5     *x = pilha[*t];
6     return 1; // sucesso
7 }
```

Pilha em Vetores

- Problema:
  - Tamanho fixo
  - As posições não são reutilizadas
  - Pode ocorrer overflow

- A fila pode ser implementada utilizando fila encadeada!

Dúvidas?

- lucas.boaventura@unb.br