CMP1054 – ED 1

Aula 06 – Fila circular usando arranjos

Fila

- É uma lista linear em que todas as inserções são realizadas em um extremo da lista, e todas as retiradas e, geralmente, os acessos são realizados no outro extremo da lista.
- O modelo intuitivo de uma fila é o de uma fila de espera em que as pessoas no início da fila são servidas primeiro e as pessoas que chegam entram no fim da fila.
- São chamadas listas fifo (“first-in”, “first-out”).

Fila

- Existe uma ordem linear para filas que é a “ordem de chegada”.
- São utilizadas quando desejamos processar itens de acordo com a ordem “primeiro-que-chega, primeiro-atendido”.
- Sistemas operacionais utilizam filas para regular a ordem na qual tarefas devem receber processamento e recursos devem ser alocados a processos.

TAD Filas

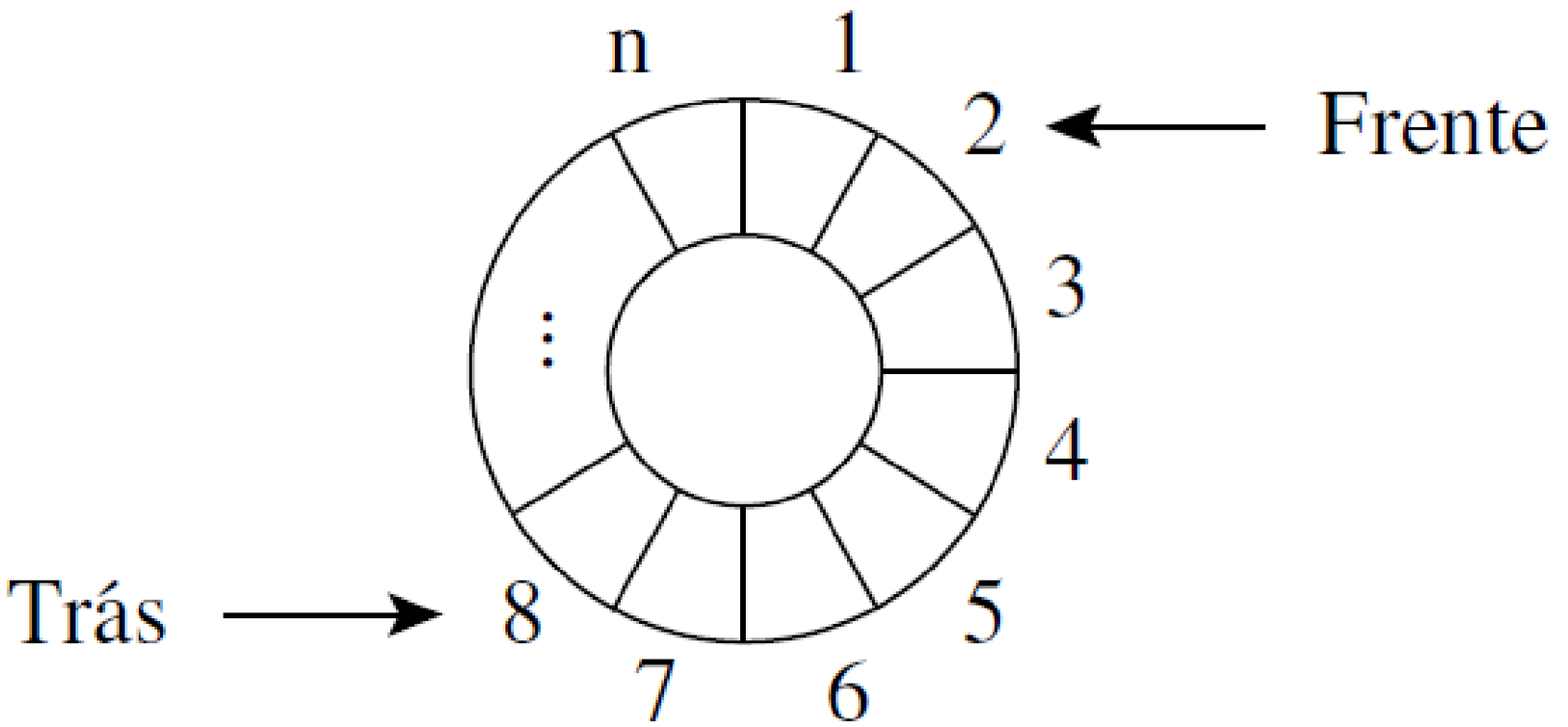
1. **Criar** uma fila vazia.
2. **Enfileirar** o item x no final da fila.
3. **Desenfileirar**. Essa função retorna o item x no início da fila e o retira da fila.
4. **Verificar** se a fila está vazia. Essa função retorna true se a fila está vazia; do contrário, retorna false.

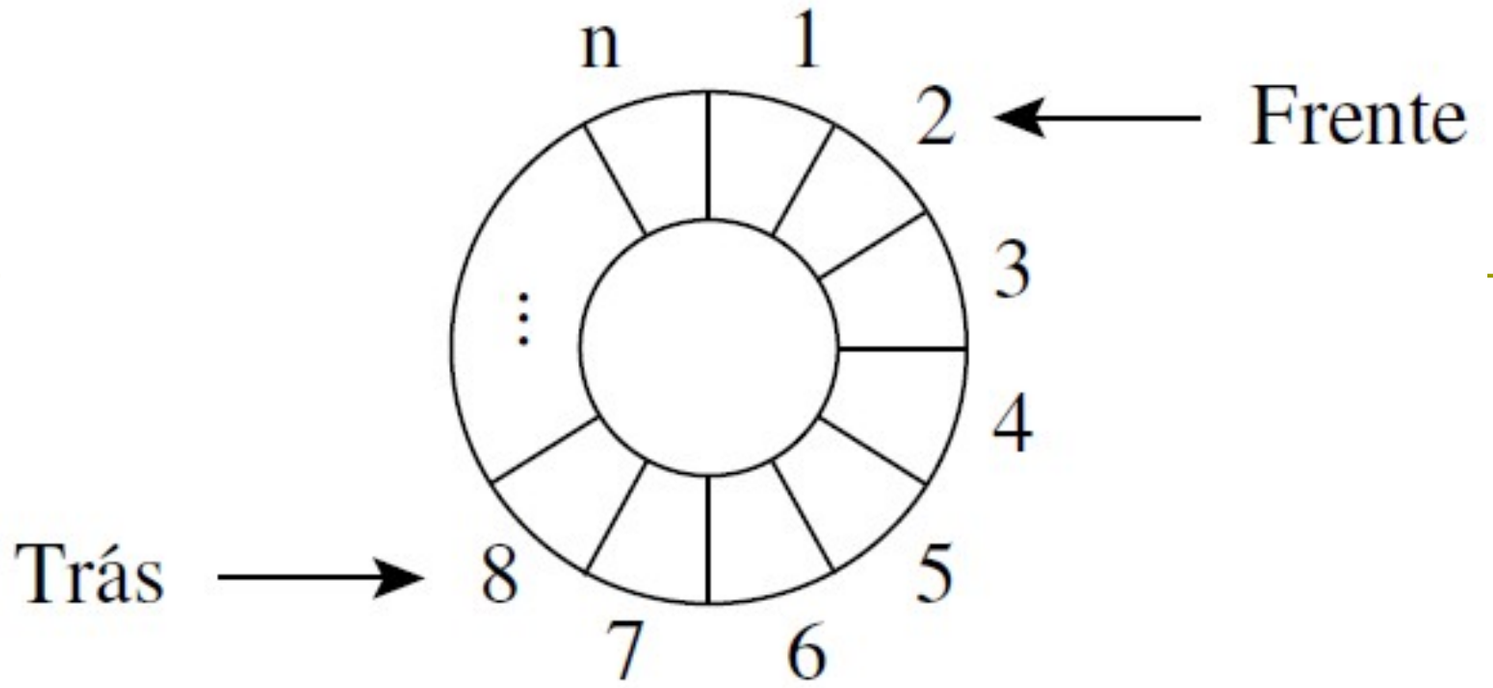
Implementação de Filas por meio de Arranjos

- Os itens são armazenados em posições contíguas de memória.
- A operação enfileira faz a parte de trás da fila expandir-se.
- A operação desenfileira faz a parte da frente da fila contrair-se.
- A fila tende a caminhar pela memória do computador, ocupando espaço na parte de trás e descartando espaço na parte da frente.
- Com poucas inserções e retiradas, a fila vai ao encontro do limite do espaço da memória alocado para ela.

Implementação de Filas por meio de Arranjos

- **Solução: imaginar o arranjo como um círculo. A primeira posição segue a última.**





- A fila se encontra em posições contíguas de memória, em alguma posição do círculo, delimitada pelos apontadores frente e trás.
- Para enfileirar, basta mover o apontador trás uma posição no sentido horário.
- Para desenfileirar, basta mover o apontador frente uma posição no sentido horário.

Estrutura e operações sobre Filas Usando Arranjos

- O tamanho máximo do arranjo circular é definido pela constante **maxTam**.
- Os outros campos da classe **Fila** contêm referências para a parte da frente e de trás da fila.
- Nos casos de fila cheia e fila vazia, os apontadores **frente** e **trás** apontam para a mesma posição do círculo.
- Uma saída para distinguir as duas situações é deixar uma posição vazia no arranjo.
- Neste caso, a fila está cheia quando **trás+1** for igual a **frente**.
- A implementação utiliza aritmética modular nos procedimentos **enfileira** e **desenfileira** (operador % em C++).

Estrutura e operações sobre Filas Usando Arranjos

```
#ifndef CELULA_H_
#define CELULA_H_
#include <string>
class Celula {
 private:
  int chave;
  std::string nome;
 public:
  Celula();
  int getChave(void);
  std::string getNome(void);
  bool setChave(int key);
  bool setName(std::string name);
  virtual ~Celula();
};

#endif /* CELULA_H_ */
```

Estrutura e operações sobre Filas Usando Arranjos

```
#include "Celula.h"
using namespace std;

Celula::Celula() {
 chave = 0;
 nome.clear();
}

int Celula::getChave(void) {
 return(chave);
}

string Celula::getNome(void) {
 return(nome);
}
```

```
bool Celula::setChave(int key) {
 chave = key;
 return (true);
}

bool Celula::setName(string name)
{
 nome = name;
 return (true);
}

Celula::~Celula() {
}
```

Estrutura e operações sobre Filas Usando Arranjos

```
#ifndef FILA_H_
#define FILA_H_

#include "Celula.h"

class Fila {
private:
Celula ** vetItens;
int   frente, atras, maxTam;
public:
 // Cria uma Fila vazia de 100 nós
 Fila ();
 // Cria uma Fila vazia de maxTam nós
 Fila (int tamanho);
 bool enfileira (Celula* x);
 Celula *desenfileira ();
 bool vazia ();
 void imprime ();
 virtual ~Fila ();
};
#endif /* FILA_H_ */
```

Estrutura e operações sobre Filas Usando Arranjos

```
#include "Fila.h"
#include <iostream>
using namespace std;

Fila::Fila() {
 maxTam = 100;
 vetItens = new Celula*[maxTam];
 frente = 0;
 atras = frente;
}
```

Estrutura e operações sobre Filas Usando Arranjos

```
Fila::Fila(int tamanho) {
 maxTam = tamanho;
 vetItens = new Celula*[maxTam];
 frente = 0;
 atras = frente;
}

bool Fila::enfileira (Celula* x) {
 if ((this->atras + 1) % this->maxTam == this->frente)
 { cout << "O vetor está cheio."; return false;}
 this->vetItens[this->atras] = x;
 this->atras = (this->atras + 1) % this->maxTam;
 return true;
}
```

Estrutura e operações sobre Filas Usando Arranjos

```
Celula* Fila::desenfileira () {
 if (this->vazia ()) {
    cout << "Erro: A fila esta vazia."<< endl; return 0;
 }
 Celula *item = this->vetItens[this->frente];
 this->vetItens[this->frente] = 0; // transfere a posse da memória
 this->frente = (this->frente + 1) % this->maxTam;
 return item;
}

bool Fila::vazia () {
 return (this->frente == this->atras);
}
```

Estrutura e operações sobre Filas Usando Arranjos

```
void Fila::imprime () {
  if(this->vazia()) cout << "A fila está vazia." << endl;
  Else { int i;
       for (i=this->frente; i!=this->atras; i=(i + 1)%this->maxTam) {
       cout << this->vetItens[i]->getChave() << endl;
       cout << this->vetItens[i]->getNome() << endl;}
  }
}

Fila::~Fila() {
  int i;
  for (i=this->frente; i!=this->atras; i=(i + 1)%this->maxTam)
     delete this->vetItens[i];
  delete[] this->vetItens;
}
```