ALGORITMOS III

Prof. Ms. Ronan Loschi.

ronan.loschi@unifasar.edu.br

31-98759-9555

UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA

FILA e FILA CIRCULAR EM

C++

DEFINIÇÃO:

Uma **FILA** é uma lista linear de dados acessada na ordem em que o **primeiro elemento que entra na lista é o primeiro a ser retirado**. Também é chamada de lista **FIFO** (**First In, First Out**).

Um exemplo comum, é uma fila de banco. Elementos podem ser adicionados às filas e retirados. A cada adição, consequentemente, aumenta a fila, e a cada saída, diminui-se a fila; então uma fila pode ficar com comprimento 0.

OBJETIVO:

Onde as FILAS são mais utilizadas?

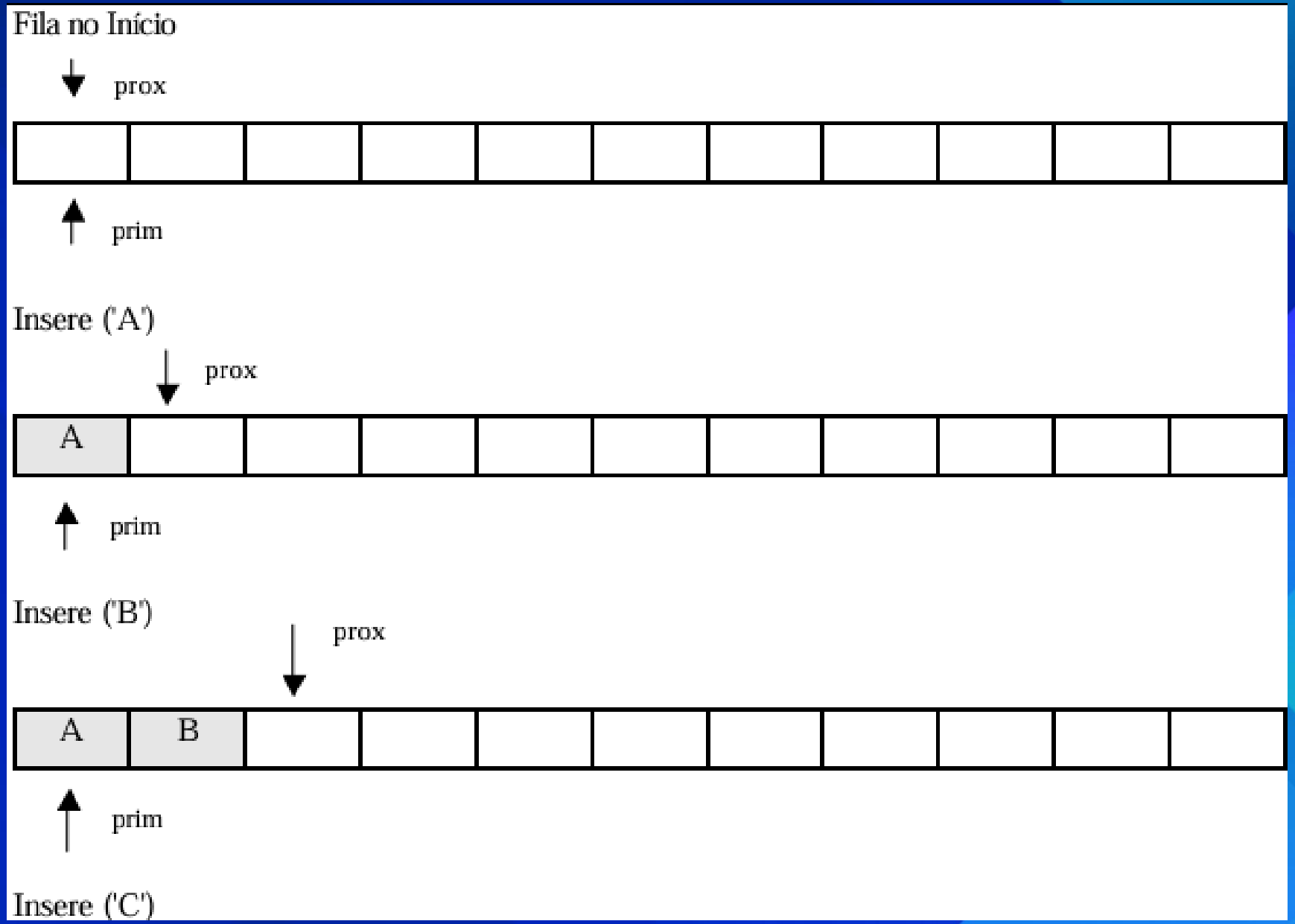
- **Sistemas de atendimento** (senhas, call centers, hospitais)
- **Processamento de tarefas** em segundo plano (ex: tarefas em servidores)
- **Impressoras** (ordem de envio dos documentos)
- **Buffers de dados** (como em streamings de áudio ou vídeo)
- **Algoritmos de grafos**, como **BFS (Busca em Largura)**
- **Sistemas operacionais**, em filas de processos

EXEMPLO:

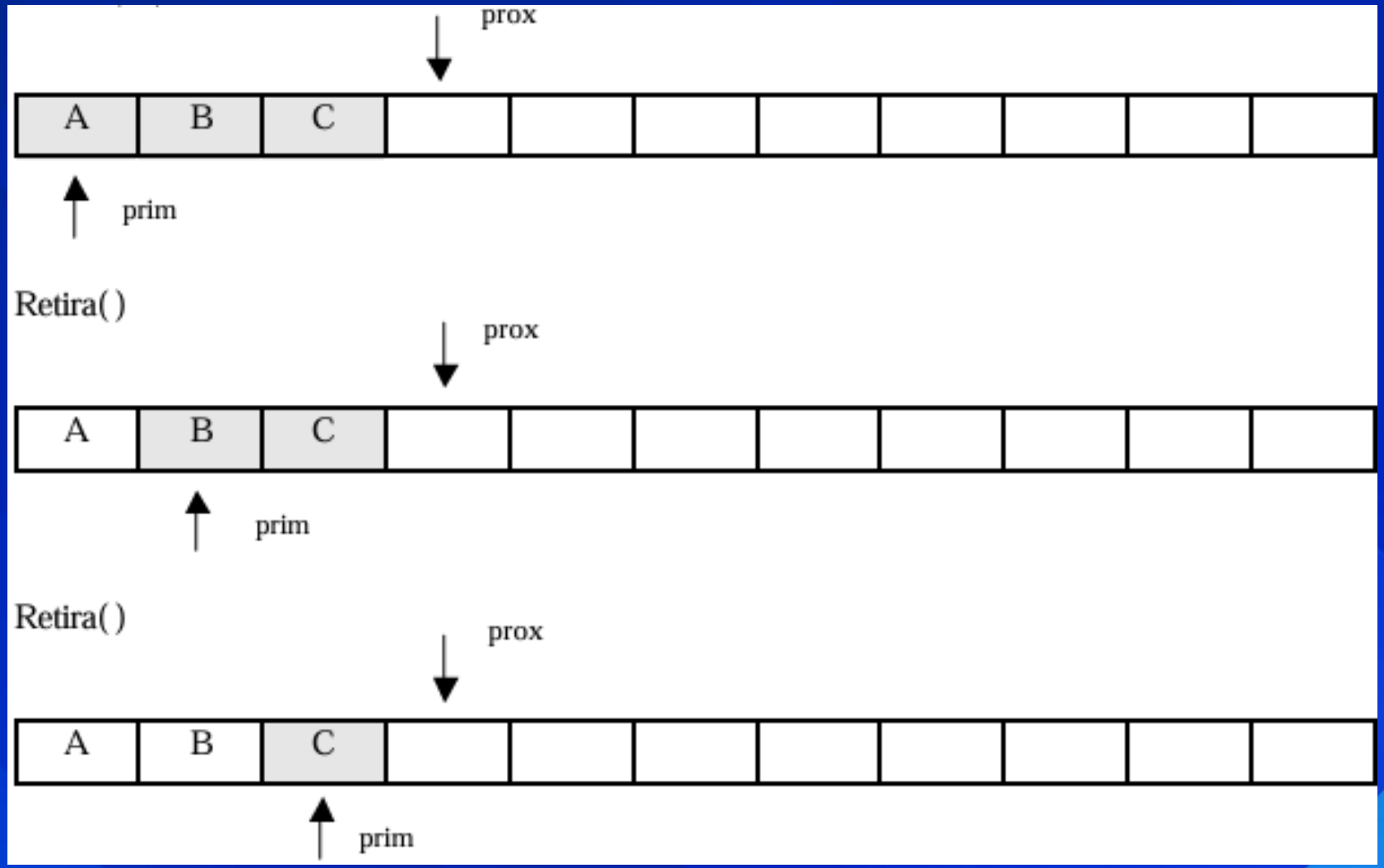
Supondo duas funções Insere() e Retira(), que inserem e retiram respectivamente elementos da fila, temos:

Ação	Conteúdo da Fila
Insere(A)	[A]
Insere(B)	[A B]
Insere(C)	[A B C]
Retira( )	[B C]
Insere(D)	[B C D]
Retira( )	[C D]
Retira( )	[D]
Retira( )	[]

EXEMPLO - FILA:



EXEMPLO - FILA:



SINTAXE COM C++:

```
#include <queue>
```

Importa a biblioteca **padrão de filas da linguagem C++,** que está contida na STL (Standard Template Library). Ela oferece implementações prontas de estruturas de dados fila (queue) e fila de prioridade (priority_queue).

EXEMPLO:

O que essa biblioteca permite fazer?
#include <queue>

você pode:

- Criar filas do tipo **FIFO** com std::queue
- Adicionar elementos ao final da fila com .push()
- Remover elementos da frente com .pop()
- Consultar o primeiro elemento com .front()
- Verificar o tamanho com .size()
- Saber se a fila está vazia com .empty()
- Retorna o último elemento . back()

IMPORTANTE:

Em **C++,** não é possível inserir elementos no meio de uma **Fila** diretamente, porque ela é projetada como uma estrutura de fila FIFO (First-In, First-Out).

Isso significa que:

- Você só pode adicionar elementos no final com .push()
- Só pode remover da frente com .pop()
- Não há acesso direto a elementos internos da fila

SINTAXE – DECLARAÇÃO DE UMA FILA DINÂMICA:

```
#include <queue>
std::queue<int> fila;
```

Você está **criando uma estrutura de dados dinâmica**, que **cresce e encolhe sozinha conforme você adiciona ou remove elementos**, sem precisar definir previamente o tamanho (como ocorre com vetores ou arrays).

SINTAXE – Inserir elemento na fila → push():

```
#include <queue>

std::queue<int> fila;

fila.push(10);  // Insere o valor 10 no final da fila
```

SINTAXE – Retirar elemento do INÍCIO da fila → pop( ):

```
#include <queue>

std::queue<int> fila;

fila.pop(10);  // retira o valor 10 (primeiro elemento) da fila
```

Obs. **Não retorna** o valor removido.

SINTAXE – Consultar o primeiro elemento (sem remover) → front():

```
#include <queue>

std::queue<int> fila;

int primeiro = fila.front( );  // Acessa o primeiro da fila
```

Obs. O conteúdo, primeiro elemento da fila, vai para a variável indicada, no caso a variável do tipo int primeiro.

SINTAXE – Verifica o tamanho da fila → size():

```
// Verifica o tamanho da fila

    fila.size( ) ;

// Mostra o último elemento inserido (fim da fila)

    fila.back();
```

SINTAXE – // Verifica se a fila está vazia → empty( ):

```
if (fila.empty()) {
    cout << "A fila está vazia." << endl;
} else {
    cout << "A fila NÃO está vazia." << endl;
}
```

EXEMPLO - Exemplo 1 – Fila de números inteiros simples:

```cpp
#include <iostream>
#include <queue>
using namespace std;

int main() {
    queue<int> fila;

    fila.push(5);
    fila.push(10);
    fila.push(15);

    cout << "primeiro elemento da da fila: " << fila.front() << endl;  // 5

    fila.pop(); // Retira o primeiro elemento da fila

    cout << "Novo primeiro elemento da fila: " << fila.front() << endl; // 10

    return 0;
}
```

EXEMPLO - Exemplo 2 – Simulando atendimento em uma fila:

```cpp
#include <iostream>
#include <queue>
#include <string>
using namespace std;

int main() {
    queue<string> filaAtendimento;

    filaAtendimento.push("João");
    filaAtendimento.push("Maria");
    filaAtendimento.push("Carlos");

    while (!filaAtendimento.empty()) {
        cout << "Atendendo: " << filaAtendimento.front() << endl; // Atende o primeiro (front) da fila
        filaAtendimento.pop(); // Retira o primeiro elemento após ser atendido (Fila "ainda")
    }

    return 0;
}
```

Exemplo 3 – Contando o tamanho da fila e mostrando todos:

```cpp
#include <iostream>
#include <queue>
using namespace std;

int main() {
    queue<int> fila;

    for (int i = 1; i <= 5; i++) {
        fila.push(i * 10);
    }

    cout << "Tamanho da fila: " << fila.size() << endl; // Tamanho da fila

    while (!fila.empty()) {
        cout << fila.front() << " "; // Omprime o primeiro
        fila.pop(); //retira o primeiro
    }

    return 0;
}
```

Exemplo 4 – Contando o tamanho da fila e mostrando o último da fila:

```cpp
#include <iostream>
#include <queue>
using namespace std;

int main() {
    queue<int> fila;

    // Adicionando elementos na fila
    fila.push(10);
    fila.push(20);
    fila.push(30);

    // Verifica o tamanho da fila
    cout << "Tamanho da fila: " << fila.size() << endl;

    // Verifica se a fila está vazia
    if (fila.empty()) {
        cout << "A fila está vazia." << endl;
    } else {
        cout << "A fila NÃO está vazia." << endl;
    }

    // Mostra o último elemento inserido (fim da fila)
    cout << "Último elemento da fila: " << fila.back() << endl;

    return 0;
}
```

UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA

Exemplo 5 – Completo:

```cpp
#include <iostream>
#include <queue>
using namespace std;

int main() {
    queue<int> fila;

    // Adicionando elementos ao final da fila
    fila.push(5);
    fila.push(10);
    fila.push(15);

    // Consultando o primeiro e o último elemento
    cout << "Primeiro elemento (front): " << fila.front() << endl;
    cout << "Último elemento (back): " << fila.back() << endl;

    // Verificando o tamanho da fila
    cout << "Tamanho da fila: " << fila.size() << endl;

    // Verificando se a fila está vazia
    if (fila.empty()) {
        cout << "A fila está vazia." << endl;
    } else {
        cout << "A fila NÃO está vazia." << endl;
    }

    // Removendo elementos da frente da fila
    cout << "Removendo elementos da fila:" << endl;
    while (!fila.empty()) {
        cout << "Removido: " << fila.front() << endl;
        fila.pop();
    }

    // Verificando novamente se a fila está vazia
    cout << "A fila está vazia agora? " << (fila.empty() ? "Sim" : "Não") << endl;

    return 0;
}
```

UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA

Exemplo 6 – Exemplo com entrada do usuário:

```cpp
#include <iostream>
#include <queue>
using namespace std;

int main() {
    queue<int> fila;
    int n, valor;

    cout << "Quantos elementos deseja inserir na fila? ";
    cin >> n;

    // Adicionando elementos digitados pelo usuário
    for (int i = 0; i < n; i++) {
        cout << "Digite o elemento #" << (i + 1) << ": ";
        cin >> valor;
        fila.push(valor); // push()
    }

    // Exibindo o primeiro e o último elemento
    if (!fila.empty()) {
        cout << "\nPrimeiro elemento (front): " << fila.front() << endl;
        cout << "Último elemento (back): " << fila.back() << endl;
    }

    // Verificando o tamanho
    cout << "Tamanho da fila: " << fila.size() << endl;

    // Verificando se está vazia
    cout << "A fila está vazia? " << (fila.empty() ? "Sim" : "Não") << endl;

    // Removendo elementos da fila
    cout << "\nRemovendo os elementos da fila:\n";
    while (!fila.empty()) {
        cout << "Removido: " << fila.front() << endl;
        fila.pop(); // pop()
    }

    // Verificando novamente se está vazia
    cout << "Fila vazia após remoção? " << (fila.empty() ? "Sim" : "Não") << endl;

    return 0;
}
```

UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA

FILA CIRCULAR EM C++

DEFINIÇÃO:

Uma fila circular **é uma variação da fila convencional** em que o último elemento está logicamente **ligado** ao primeiro, formando um ciclo contínuo na estrutura de dados.

Ela é geralmente implementada com um **vetor fixo** (array).

DEFINIÇÃO:

Por que usar fila circular?

Eficiência de memória: evita que posições “vazias” à frente da fila fiquem inutilizadas após várias remoções.

Desempenho constante: ideal em contextos onde o tamanho máximo da fila **é conhecido** (ex: buffers circulares, drivers, sistemas embarcados).

DEFINIÇÃO:

Uma fila circular **é uma variação da fila convencional** em que o último elemento está logicamente **ligado** ao primeiro, formando um ciclo contínuo na estrutura de dados.

Ela é geralmente implementada com um **vetor fixo** (array).

Diferença fila comum X fila circular:

Característica	Fila Comum	Fila Circular
Crescimento	Apenas para frente	Retorna ao início se houver espaço
Uso de espaço	Pode desperdiçar espaço	Usa o vetor inteiro eficientemente
Inserção (push)	No final	No final, com rotação circular
Remoção (pop)	Da frente	Da frente, com rotação circular

EXEMPLOS

```cpp
#include <iostream>
using namespace std;

#define TAM 3  // Tamanho fixo da fila

int main() {
    int fila[TAM];     // Vetor que representa a fila
    int inicio = 0;    // Índice do início da fila
    int fim = 0;       // Índice do final da fila
    int tamanho = 0;   // Quantidade de elementos na fila

// Inserindo 3 elementos na fila circular
fila[fim] = 10;                // Coloca o valor 10 na posição indicada por 'fim' (posição 0)
fim = (fim + 1) % TAM;         // Avança o índice 'fim' para a próxima posição da fila
                               // O operador % (módulo) garante que, ao chegar no final do vetor,
                               // ele volte para o início (posição 0), mantendo o comportamento circular
tamanho++;                     // Aumenta a variável 'tamanho' para indicar que a fila agora tem 1 elemento

fila[fim] = 20;                // Insere o valor 20 na nova posição de 'fim' (posição 1)
fim = (fim + 1) % TAM;         // Avança 'fim' circularmente (irá para a posição 2)
tamanho++;                     // Agora a fila tem 2 elementos

fila[fim] = 30;                // Insere o valor 30 na posição 2 do vetor
fim = (fim + 1) % TAM;         // Como TAM = 3, agora fim volta para 0 (posição inicial do vetor)
                               // Isso mostra o comportamento circular – o fim "dá a volta"
tamanho++;                     // A fila agora está cheia com 3 elementos

    // Removendo 1 elemento (o da frente, que é 10)
    inicio = (inicio + 1) % TAM;  // Avança início de forma circular
    tamanho--;

    // Inserindo mais 1 elemento (reaproveitando espaço no início do vetor)
    fila[fim] = 40;
    fim = (fim + 1) % TAM;
    tamanho++;

    // Exibindo a fila circular
    cout << "Fila circular: ";
    for (int i = 0; i < tamanho; i++) {
        // Acesso circular aos elementos
        cout << fila[(inicio + i) % TAM] << " ";
    }
    cout << endl;

    return 0;
}
```

FILA CIRCULAR Exemplo 1 – inserir e remover em fila circular pequena

fim = (fim + 1) % TAM é o coração da fila circular: ele impede que o índice saia do vetor, fazendo o "retorno ao início".

tamanho++ controla a quantidade de elementos válidos — evita confundir posição livre com ocupada.

UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA

Exemplo 2 – Fila circular com entrada do usuário

```cpp
#include <iostream>
using namespace std;

#define TAM 2  // Tamanho fixo da fila

int main() {
    int fila[TAM];      // Vetor da fila circular
    int inicio = 0;     // Início da fila
    int fim = 0;        // Final da fila
    int tamanho = 0;    // Quantidade de elementos na fila
    int valor;          // Valor digitado pelo usuário

    // Inserção de valores digitados pelo usuário
    for (int i = 0; i < TAM; i++) {
        cout << "Digite um valor: ";
        cin >> valor;

        fila[fim] = valor;              // Insere na posição 'fim'
        fim = (fim + 1) % TAM;          // Avança fim circularmente
        tamanho++;
    }

    // Exibindo a fila circular
    cout << "Fila circular: ";
    for (int i = 0; i < tamanho; i++) {
        // Imprime os elementos na ordem da fila, respeitando o início
        cout << fila[(inicio + i) % TAM] << " ";
    }
    cout << endl;

    return 0;
}
```

UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA

EXRCÍCIOS - FILA

1) Fila comum – Atendimento em um consultório

Enunciado:

Um consultório médico quer organizar o atendimento dos pacientes pela ordem de chegada. Implemente um programa em C++ que:

1. Adicione três pacientes: "João", "Maria" e "Carlos".
2. Mostre o próximo a ser atendido.
3. Remova o primeiro paciente (foi atendido).
4. Mostre o novo próximo paciente

2) Fila comum – Impressora de documentos

Enunciado: Uma impressora doméstica imprime os documentos na ordem em que são enviados. Faça um programa que:

Leia 3 nomes de arquivos do usuário.

Insira cada nome em uma fila.

Mostre a quantidade de documentos na fila.

Imprima o primeiro (removendo da fila) e mostre o próximo.

3) Fila comum – Lanchonete

Enunciado: Em uma lanchonete, os pedidos são atendidos na ordem de chegada. Implemente:

Inserção de 2 pedidos do usuário.
Verificação se a fila está vazia.
Exibição do último pedido da fila.

EXRCÍCIOS – FILA CIRCULAR

1) Fila Circular – Armazém com 3 caixas

Enunciado: Um armazém tem 3 caixas para empacotar produtos. Quando todas estão cheias e uma é liberada, a próxima caixa disponível é usada no formato circular.

Insira 3 códigos de produto.
Libere 1 caixa.
Insira mais 1 produto e exiba a fila circular.

1) Fila Circular – Posto de vacinação com 2 senhas

Enunciado:Um posto de saúde distribui senhas para vacinação com uma fila circular de 2 posições. Faça um programa que:

Leia 2 senhas do teclado.
Libere 1 pessoa.
Adicione mais 1 senha e mostre a fila circular.

OBRIGADO!

UNIFASAR
CENTRO UNIVERSITÁRIO SANTA RITA