

Integrado *Sistemas Operacionais*

Tampão circular

protoCore 0,00001

Prof. Otávio Gomes
otavio.gomes@unifei.edu.br

 /otavio-gomes



Tampões circulares



Amortecedores circulares

- “Espaços de memória “infinitos”
- Use a abordagem FIFO
- Armazenar dados temporários
- Pode ser implementado usando vetores ou listas vinculadas

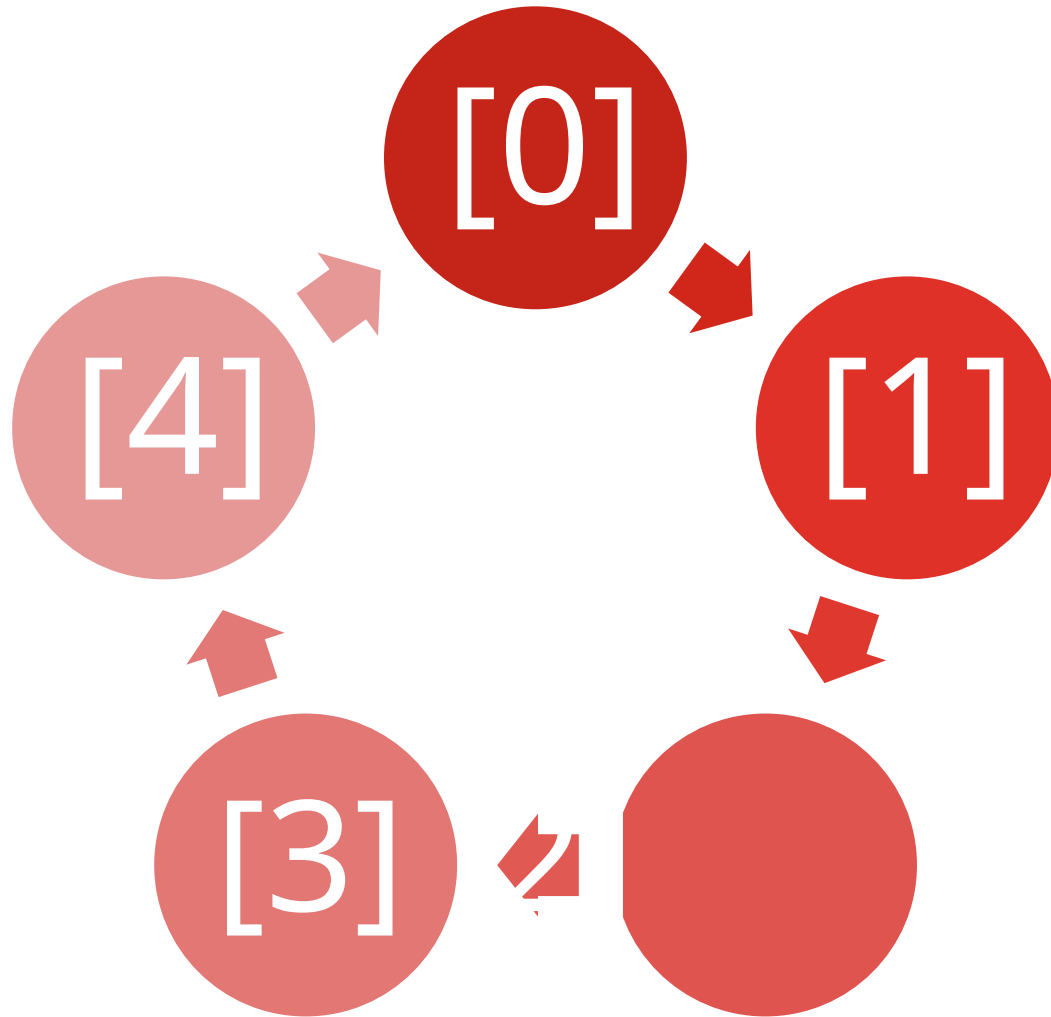


Amortecedores circulares

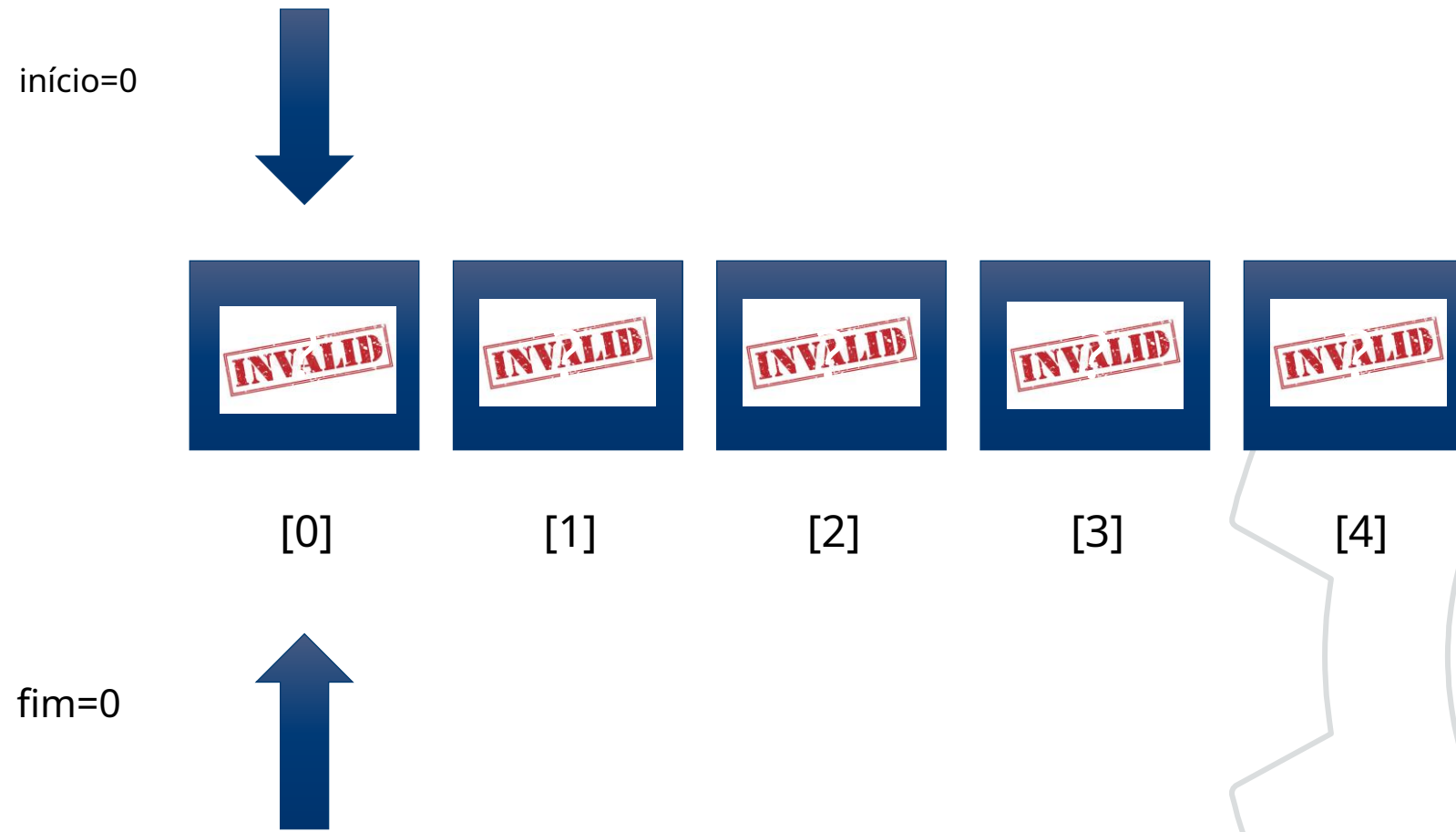
- Implementação vetorial
 - Usa menos espaço
 - É preciso ter cuidado especial ao andar de bicicleta
 - Problema para diferenciar cheio de vazio



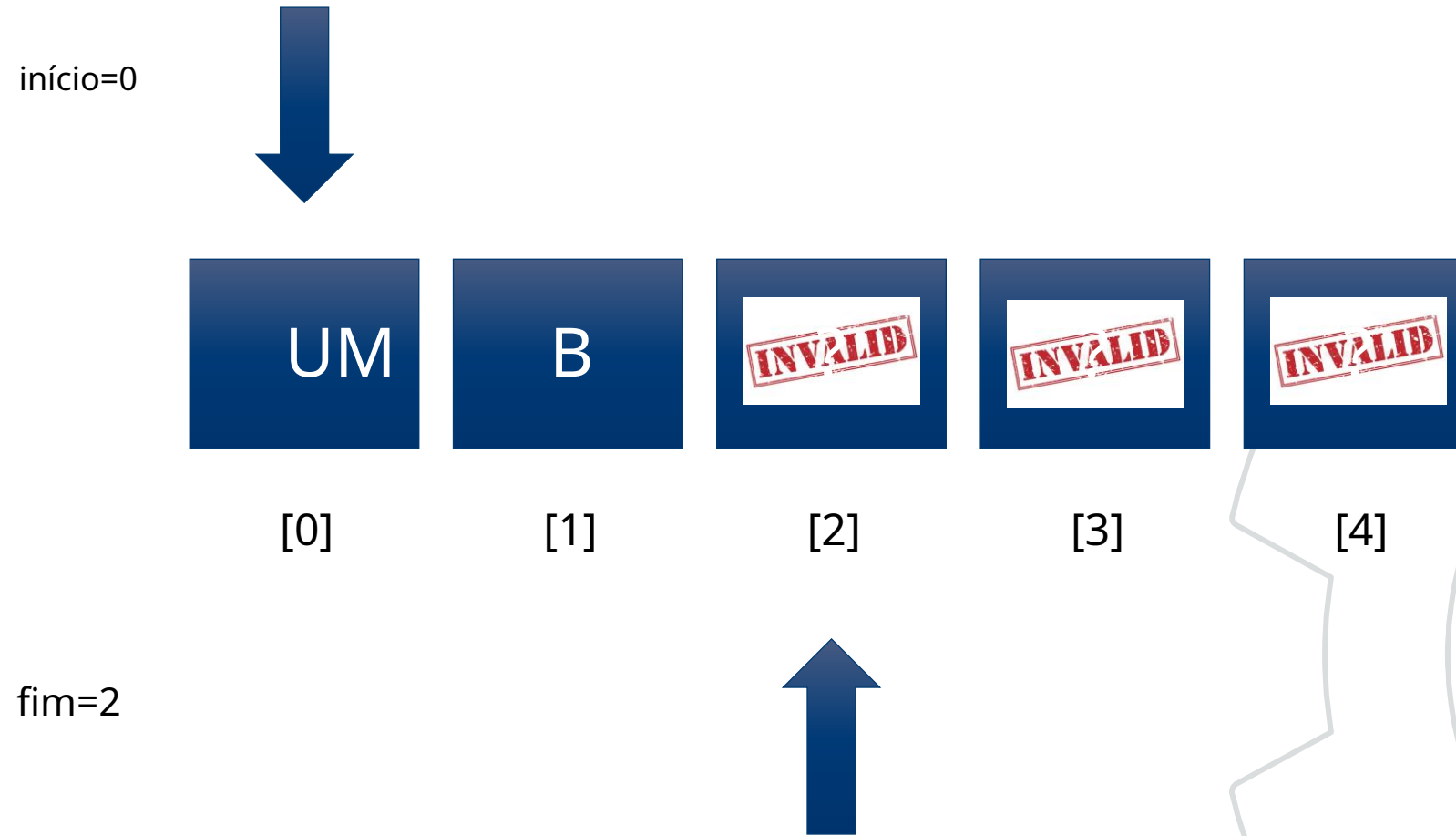
Amortecedores circulares



Buffer vazio



Adicionando 2 elementos



Removendo 1 elemento

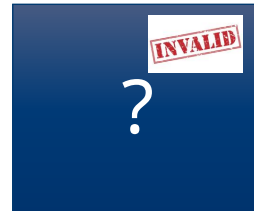
início=1



[0]



[1]



[2]

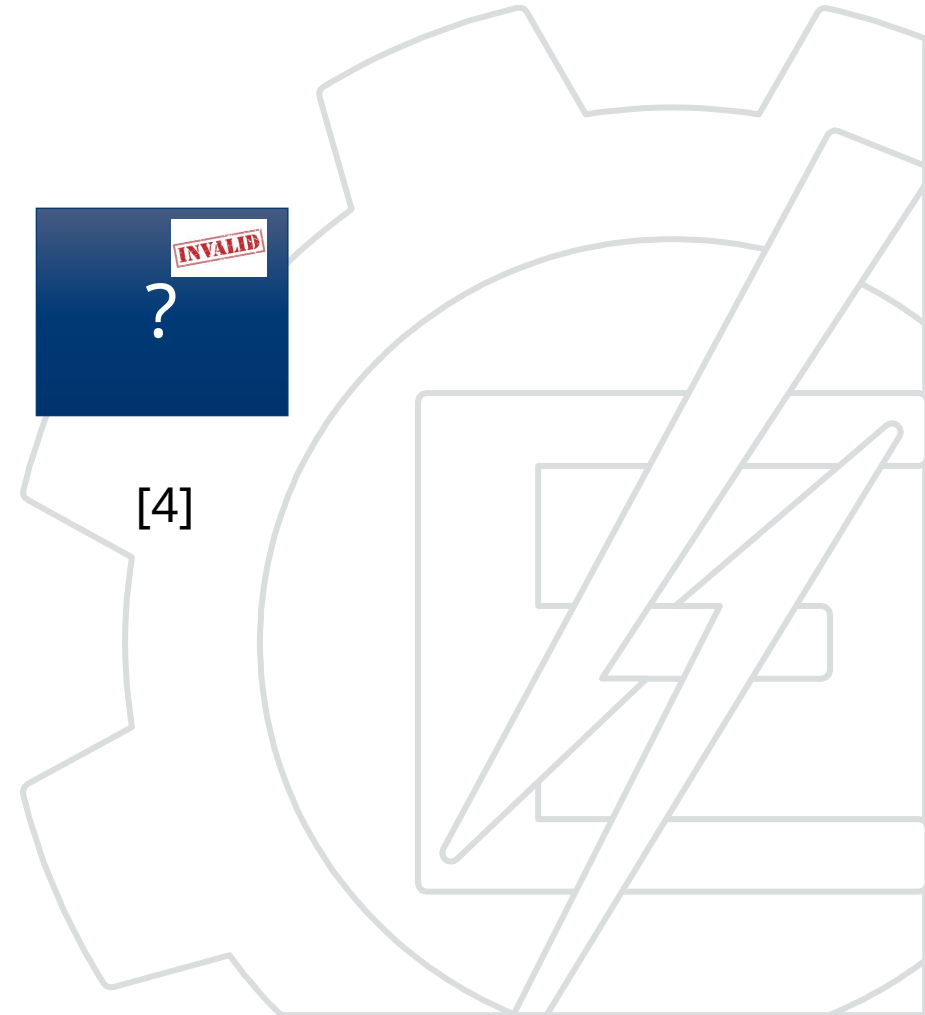


[3]

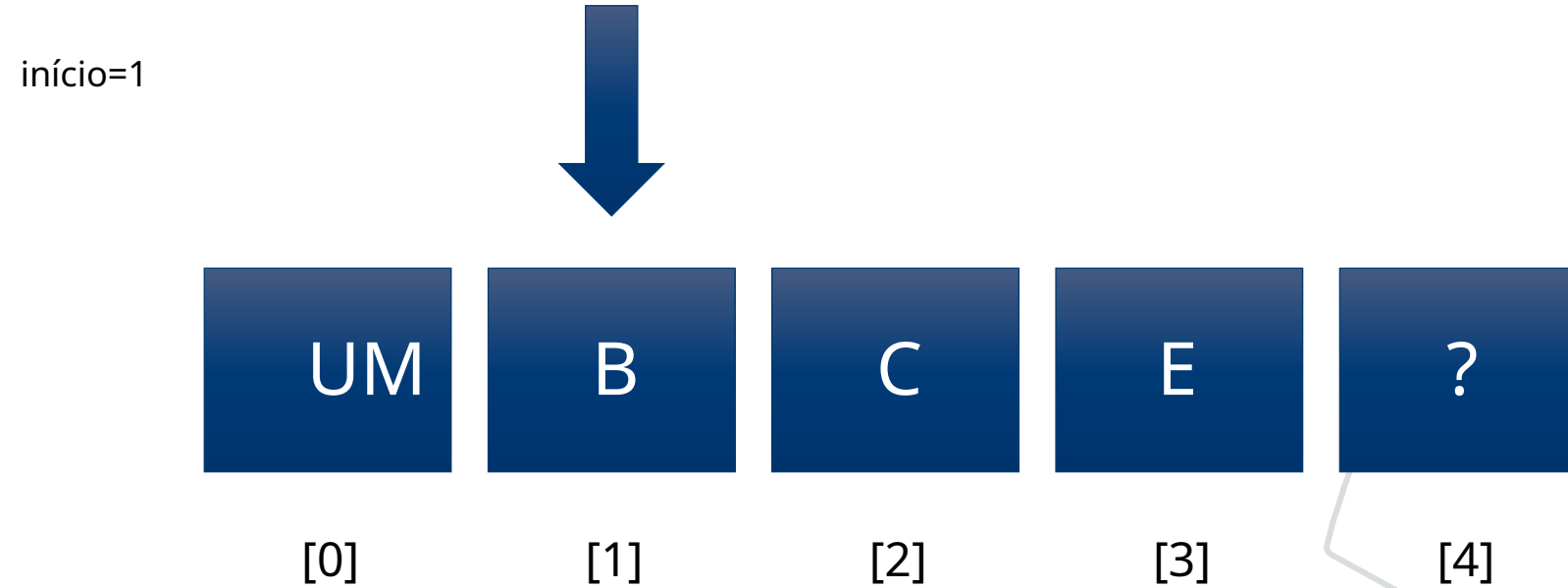


[4]

fim=2

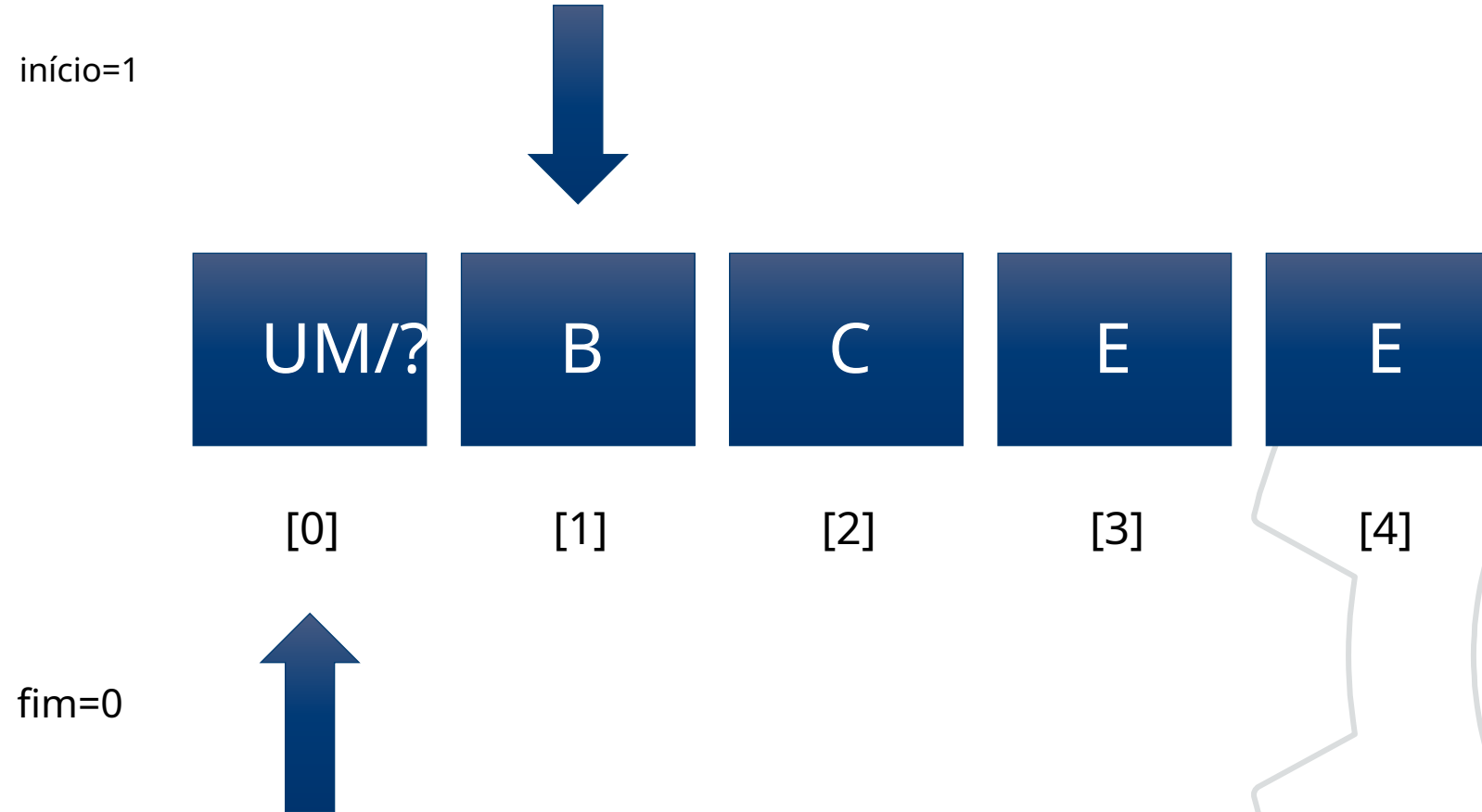


Adicionando 2 elementos



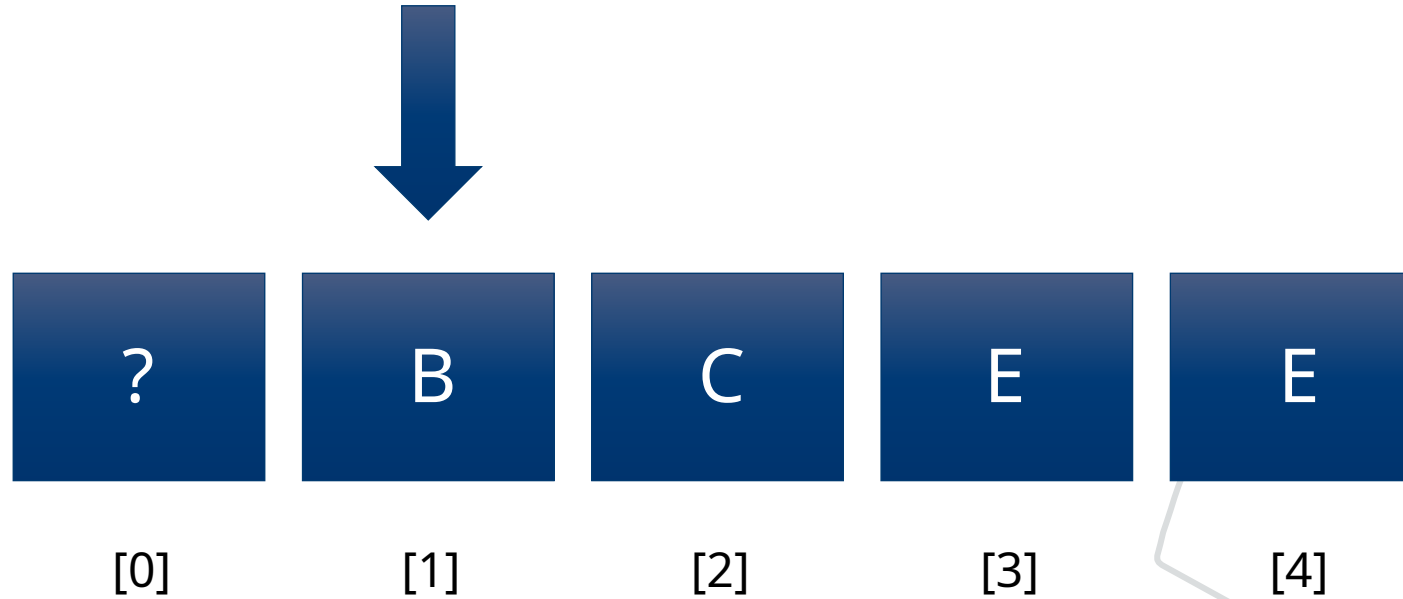
fim=4

Adicionando 1 elemento



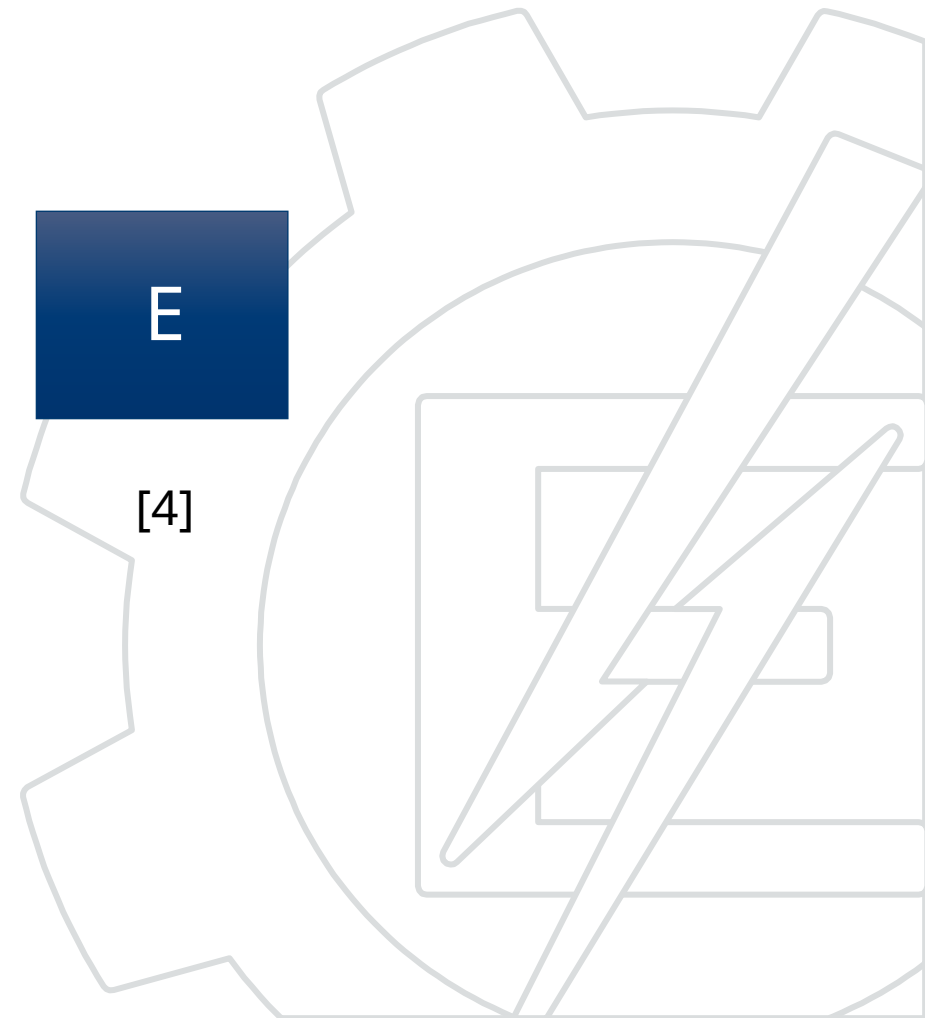
Buffer Completo

início=1



fim=0

Tampão: *Completo*



```
# definir TAMANHO_CB 10
```

```
Inteiro buffer_circular[TAMANHO_CB];
```

```
Inteiro índice=0;
```

```
para(;;){
```

```
    //faça qualquer coisa com o buffer
```

```
    buffer_circular[índice] = índice; //
```

```
    incrementa o índice
```

```
    índice= (índice+1)%TAMANHO_CB;
```

```
}
```



```
# definir TAMANHO_CB 10
```

```
Inteiro buffer_circular[TAMANHO_CB];
```

```
Inteiro índice=0;
```

```
para(;;){
```

```
    //faça qualquer coisa com o buffer
```

```
    buffer_circular[índice] = índice; //
```

```
    incrementa o índice
```

```
    índice= (índice+1)%TAMANHO_CB; //índice++;
```

```
}
```

```
# definir TAMANHO_CB 10
Inteiro buffer_circular[TAMANHO_CB];
inteiro começar=0, fim=0;
```

```
Caracteres Adicionar Buff(int novos dados) {
```

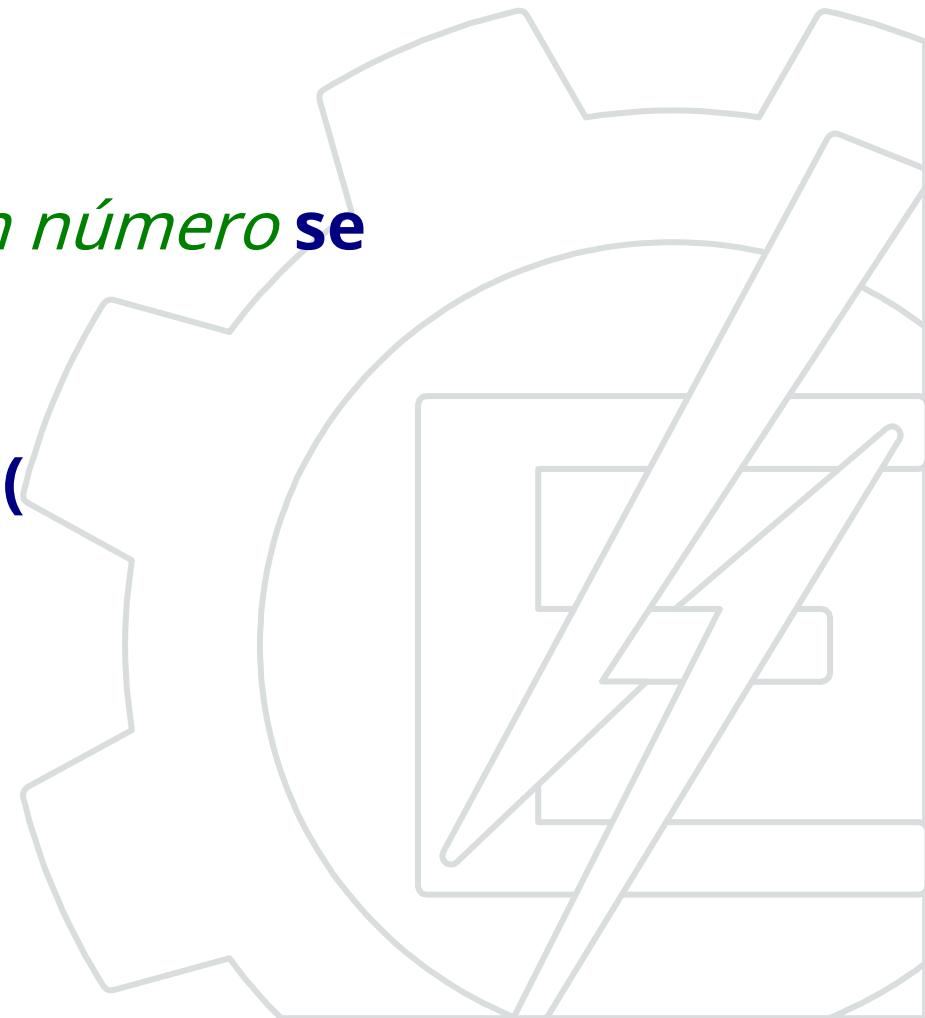
```
    //verifique se há espaço para adicionar um número se
    ( ((fim+1)%TAMANHO_CB) !=começar) {
```

```
        buffer_circular[fim] =novos dados; fim= (
        fim+1)%TAMANHO_CB; retornar
        SUCESSO;
```

```
    }
```

```
    retornar FALHAR;
```

```
}
```



```
# definir TAMANHO_CB 10
Inteiro buffer_circular[TAMANHO_CB];
inteiro começar=0, fim=0;
```

```
Caracteres Adicionar Buff(int novos dados) {
```

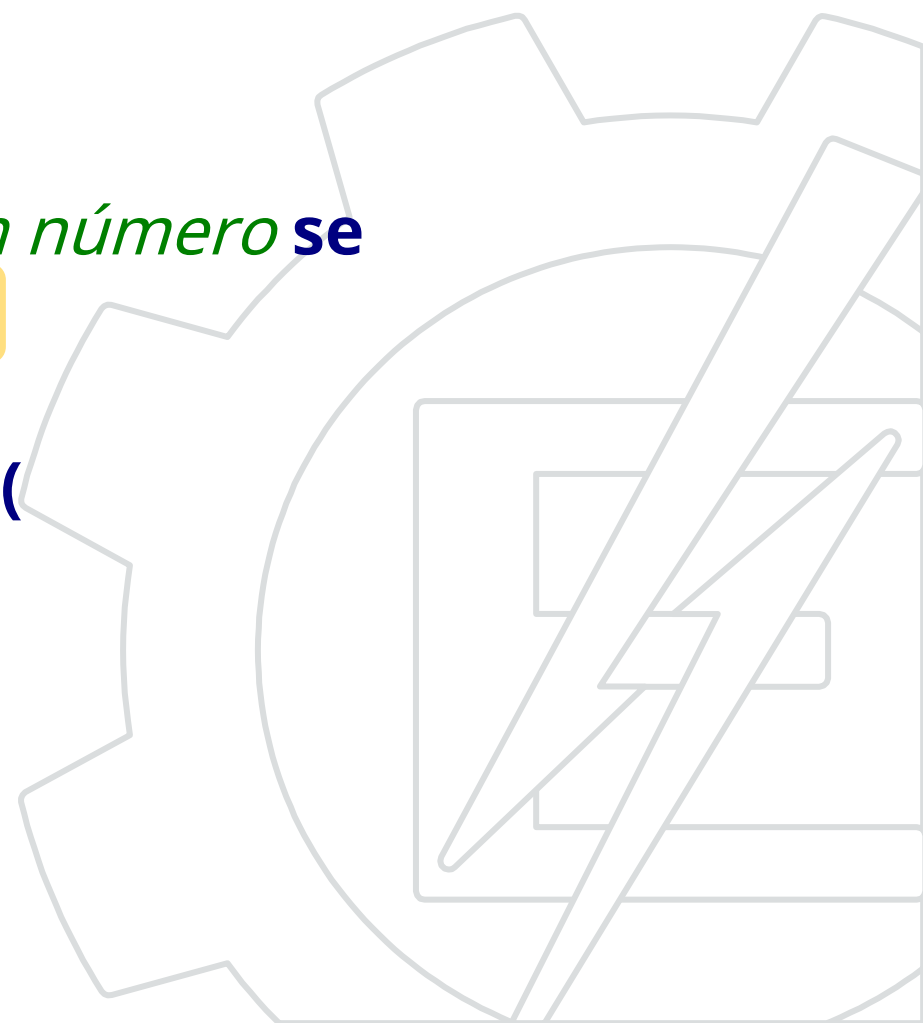
```
//verifique se há espaço para adicionar um número se
( ((fim+1)%TAMANHO_CB) !=começar) {
```

```
    buffer_circular[fim] =novos dados; fim= (
    fim+1)%TAMANHO_CB; retornar
    SUCESSO;
```

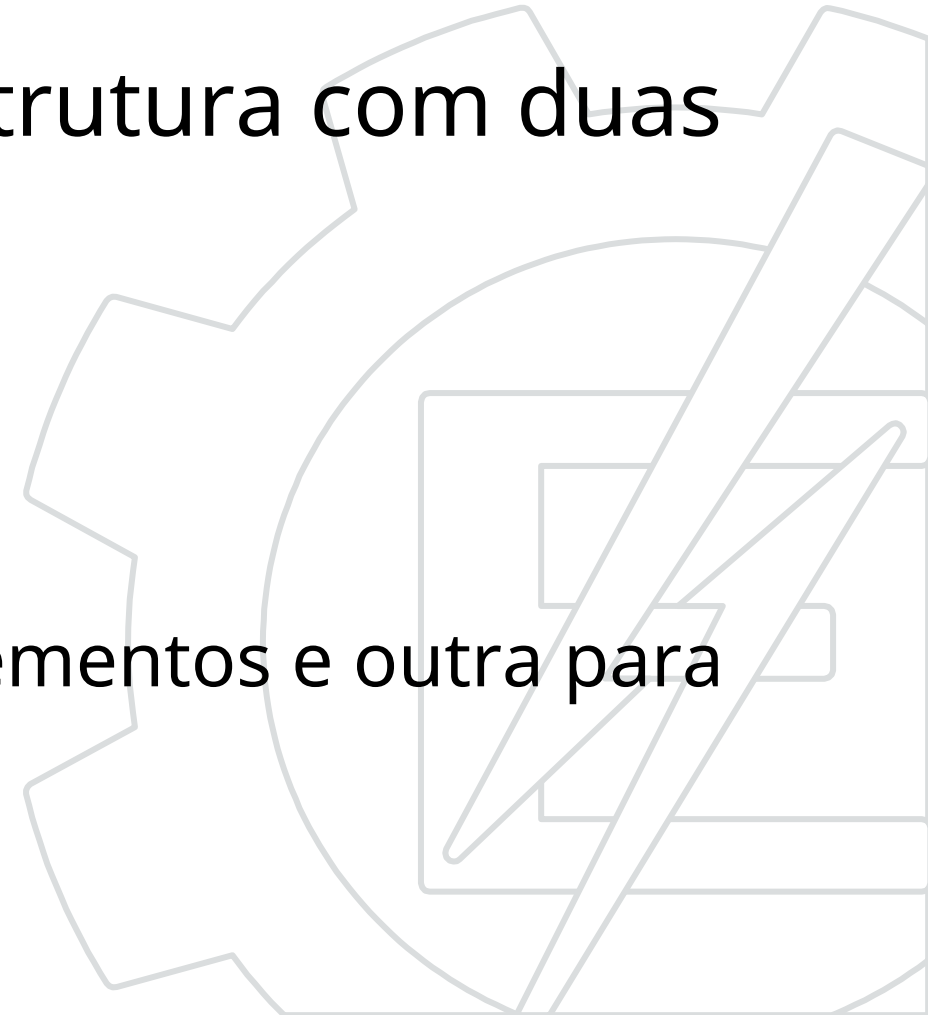
```
}
```

```
    retornar FALHAR;
```

```
}
```



- Implementar um buffer circular
 - Use um vetor de 10 posições
- Cada elemento do vetor é uma estrutura com duas variáveis
 - `char * Nome do Processo;`
 - `int Tempo;`
- Crie uma função para adicionar novos elementos e outra para remover os elementos mais antigos.




```
tipo definido estrutura{  
    Caracteres*nome do processo;  
    Inteiro tempo;  
}processo;
```

```
//declaração de buffer circular  
# definir BUFFERSIZE 10 buffer de processo[  
TAMANHO DO AMORTECEDOR];
```

```
//Declaração de ponteiros de acesso  
Inteiro início=0,fim=0;
```



```
top - 19:00:06 up 7:47, 1 user, load average: 0.65, 0.57, 0.51
Tasks: 198 total, 2 running, 196 sleeping, 0 stopped, 0 zombie
%Cpu(s): 12.6 us, 0.6 sy, 0.0 ni, 86.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 11894.0 total, 1511.5 free, 5763.0 used, 4619.4 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 5706.3 avail Mem
```

| PID | USER | PR | NI | VIRT | RES | SHR | S | %CPU | %MEM | TIME+ | COMMAND |
|-------|------|-----|-----|---------|--------|--------|---|------|------|-----------|-----------------|
| 2844 | root | 20 | 0 | 4169800 | 1.9g | 152908 | R | 46.8 | 16.4 | 126:45.88 | Web Content |
| 2758 | root | 20 | 0 | 2560304 | 462792 | 162424 | S | 5.6 | 3.8 | 49:01.37 | firefox-esr |
| 1383 | root | 20 | 0 | 521120 | 113500 | 82664 | S | 0.3 | 0.9 | 12:35.23 | Xorg |
| 2494 | root | 20 | 0 | 6347740 | 1.6g | 37592 | S | 0.3 | 13.7 | 31:22.93 | java |
| 3030 | root | 20 | 0 | 625936 | 50372 | 31888 | S | 0.3 | 0.4 | 0:34.02 | gnome-terminal- |
| 11209 | root | -51 | 0 | 17868 | 3504 | 3028 | R | 0.3 | 0.0 | 0:00.03 | top |
| 1 | root | 20 | 0 | 202592 | 8988 | 6760 | S | 0.0 | 0.1 | 0:17.10 | systemd |
| 2 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.02 | kthreadd |
| 3 | root | 0 | -20 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.00 | rcu_gp |
| 5 | root | 0 | -20 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.48 | kworker/0:0H |
| 7 | root | 0 | -20 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.00 | mm_percpu_wq |
| 8 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.35 | ksoftirqd/0 |
| 9 | root | 20 | 0 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:12.91 | rcu_sched |
| 10 | root | 20 | 0 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.00 | rcu_bh |
| 11 | root | rt | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.01 | migration/0 |
| 12 | root | rt | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.08 | watchdog/0 |
| 13 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | cpuhp/0 |
| 14 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | cpuhp/1 |
| 15 | root | rt | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.09 | watchdog/1 |
| 16 | root | rt | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | migration/1 |
| 17 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.71 | ksoftirqd/1 |
| 19 | root | 0 | -20 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.00 | kworker/1:0H |
| 20 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | cpuhp/2 |
| 21 | root | rt | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.09 | watchdog/2 |
| 22 | root | rt | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.01 | migration/2 |
| 23 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.38 | ksoftirqd/2 |
| 25 | root | 0 | -20 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.00 | kworker/2:0H |

//Função para adicionar processo ao buffer

vazio adicionarProc(Caracteres*nnome,Inteirotempo){

//Verificação de posição (completa?)

se(((fim+1)%TAMANHO DO AMORTECEDOR) !=começar){

//Atualização da posição atual

amortecedor[fim].nome do processo=nnome;

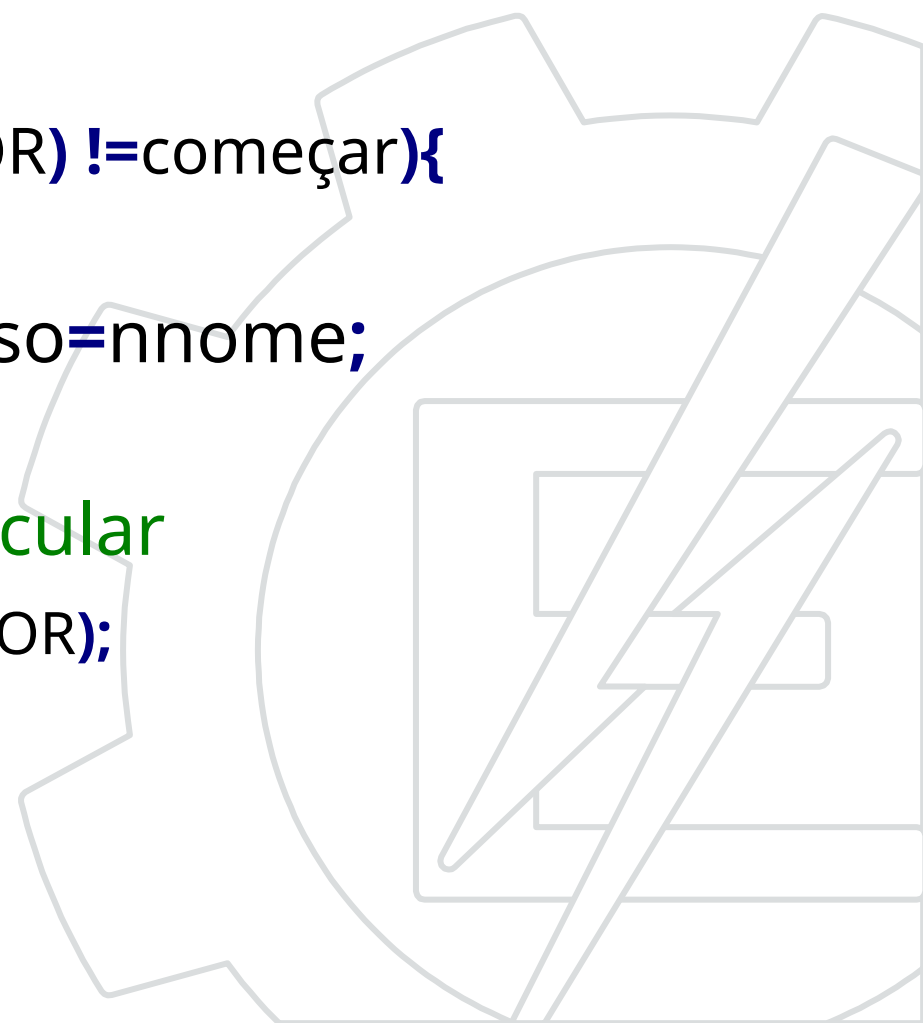
amortecedor[fim].tempo=tempo;

//incrementa a posição do buffer circular

fim= (fim+1)%(TAMANHO DO AMORTECEDOR);

}

}



//Função para adicionar processo ao buffer

vazio adicionarProc(Caracteres*nnome,Inteirotempo){

//Verificação de posição (completa?)

se(((fim+1)%TAMANHO DO AMORTECEDOR) !=começar){

//Atualização da posição atual

amortecedor[fim].nome do processo=nnome;

amortecedor[fim].tempo=tempo;

//incrementa a posição do buffer circular

fim= (fim+1)%(TAMANHO DO AMORTECEDOR);

}

}

//Função para adicionar processo ao buffer

vazio adicionarProc(**Caracteres***nnome,**Inteiro**tempo){

//Verificação de posição (completa?)

se(((fim+1)%TAMANHO DO AMORTECEDOR) !=começar){

//Atualização da posição atual

amortecedor[fim].nome do processo=nnome;

amortecedor[fim].tempo=tempo;

//incrementa a posição do buffer circular

fim= (fim+1)%(TAMANHO DO AMORTECEDOR);

}

}

```
//Função para remover processo do buffer  
vazioremoveProc(vazio){
```

```
//Verificação de posição (vazio?)
```

```
se(fim!=começar){
```

```
    //incrementa a posição inicial do buffer circular
```

```
    começar= (começar+1)%(TAMANHO DO AMORTECEDOR);
```

```
}
```

```
}
```

```
# incluir "stdio.h"
```

```
vazio principal(vazio){
```

```
    adicionarProc("proc1",0);
```

```
    adicionarProc("proc2",1);
```

```
    adicionarProc("proc3",2);
```

```
    removerProc();
```

```
    removerProc();
```

```
    removerProc();
```

```
}
```

Exercício





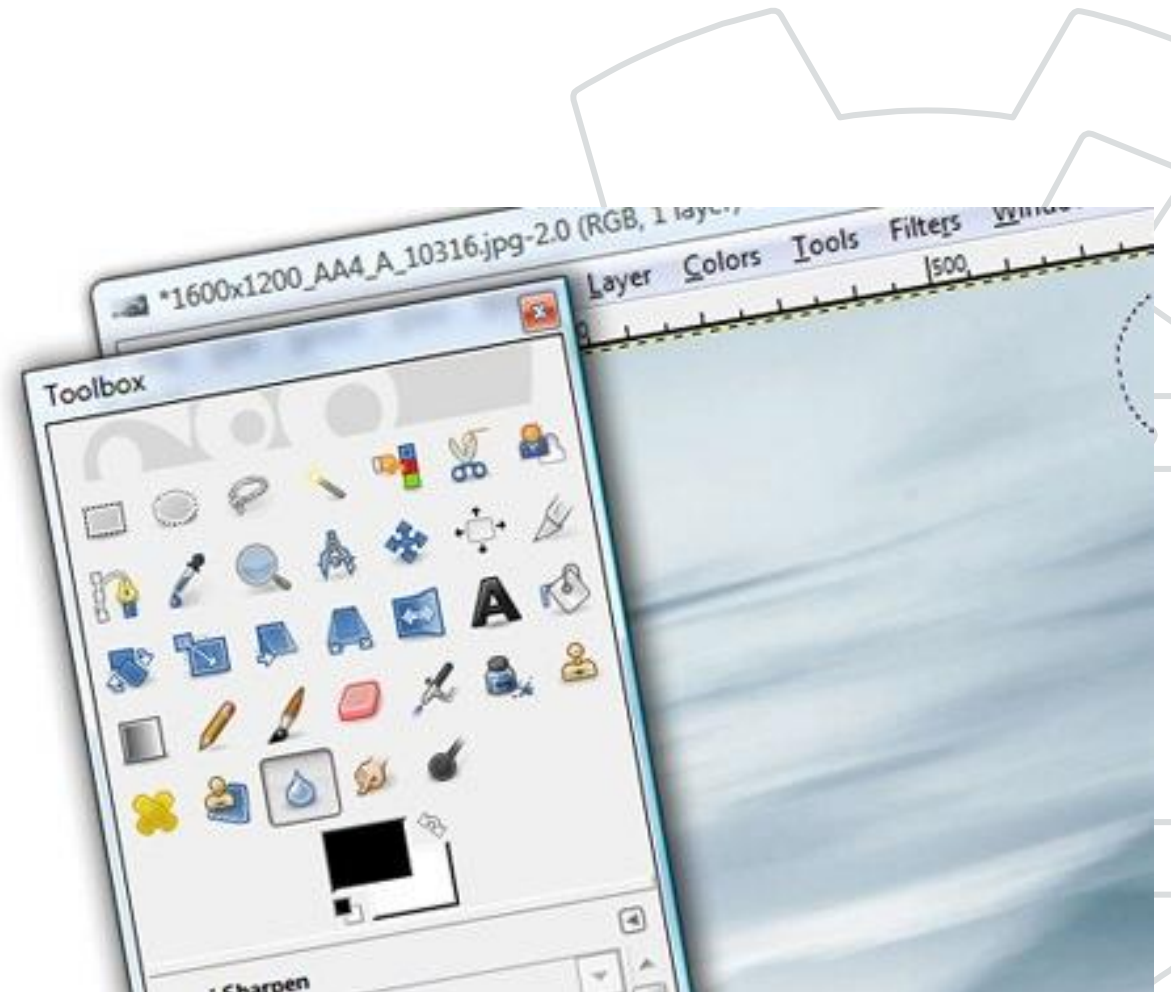
Motores de software

Kernel – abstração e gerenciamento



Motores de software

- **Meta:**
 - Crie um editor de imagens que possa escolher a função certa para chamar
- 1ª Implementação
 - Use um parâmetro de opção como um operador de troca



```
imagemBorrão(imagemnImg){  
imagemAfiado(imagemnImg){
```

```
imagemMotor do Editor de Imagens(imagemnImg,Inteirooptar){  
    imagemtemperatura;  
    trocar(optar){  
        caso1:  
            temperatura=Afiado(nImg);  
            quebrar;  
        caso2:  
            temperatura=Borrão(nImg);  
            quebrar;  
    }  
    retornartemperatura;  
}
```



```
imagemBorrão(imagemnImg){  
imagemAfiado(imagemnImg){
```

```
imagemMotor do Editor de Imagens(imagemnImg,Inteirooptar){  
    imagemtemperatura;  
    trocar(optar){  
        caso1:  
            temperatura=Afiado(nImg);  
            quebrar;  
        caso2:  
            temperatura=Borrão(nImg);  
            quebrar;  
    }  
    retornar temperatura;  
}
```

Por que não?

```
Afiado(nImg);  
    ou  
Borrão(nImg);
```

```
imagemBorrão(imagemnImg){}  
imagemAfiado(imagemnImg){}
```

```
imagemMotor do Editor de Imagens(imagemnImg,Inteirooptar){  
    imagemtemperatura;  
    trocar(optar){  
        caso1:  
            temperatura=Afiado(nImg);  
            quebrar;  
        caso2:  
            temperatura=Borrão(nImg);  
            quebrar;  
    }  
    retornar temperatura;  
}
```

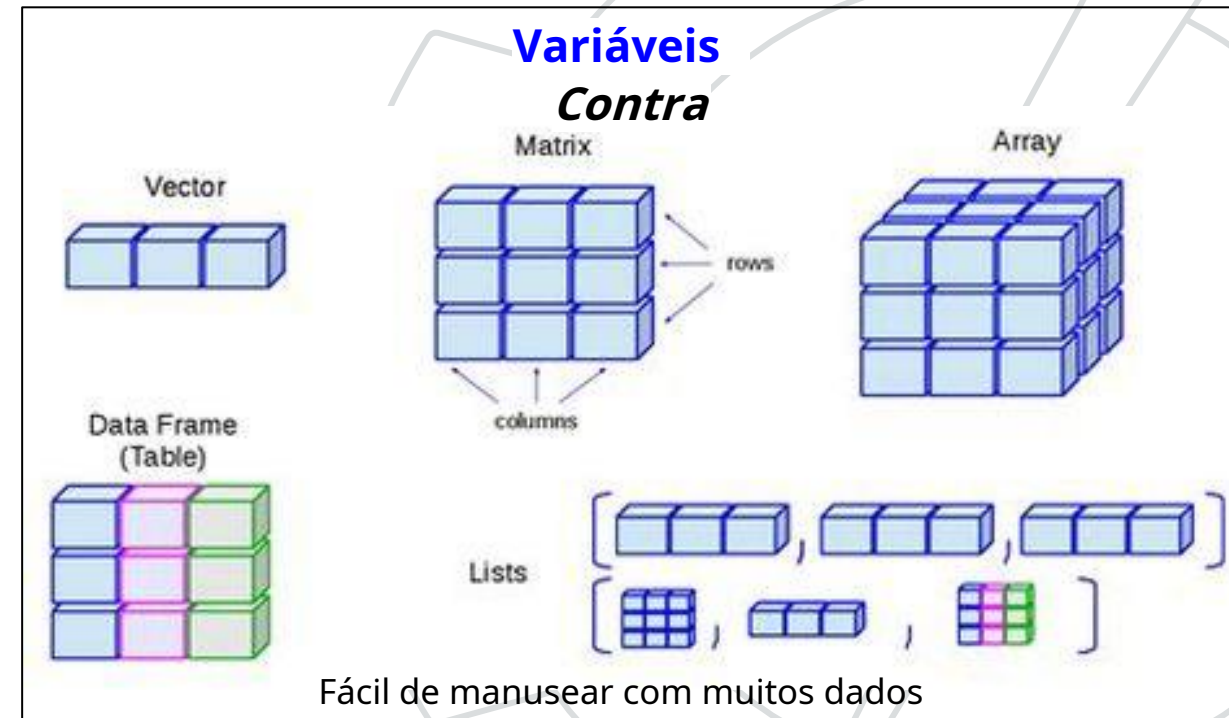
PEGAREDEFINIR métodos

- Permissões
- Resolução de imagem
- Tipo de arquivo

Variáveis
Contra
Tampão circular

```
imagemBorrão(imagemnImg){}
imagemAfiado(imagemnImg){}
```

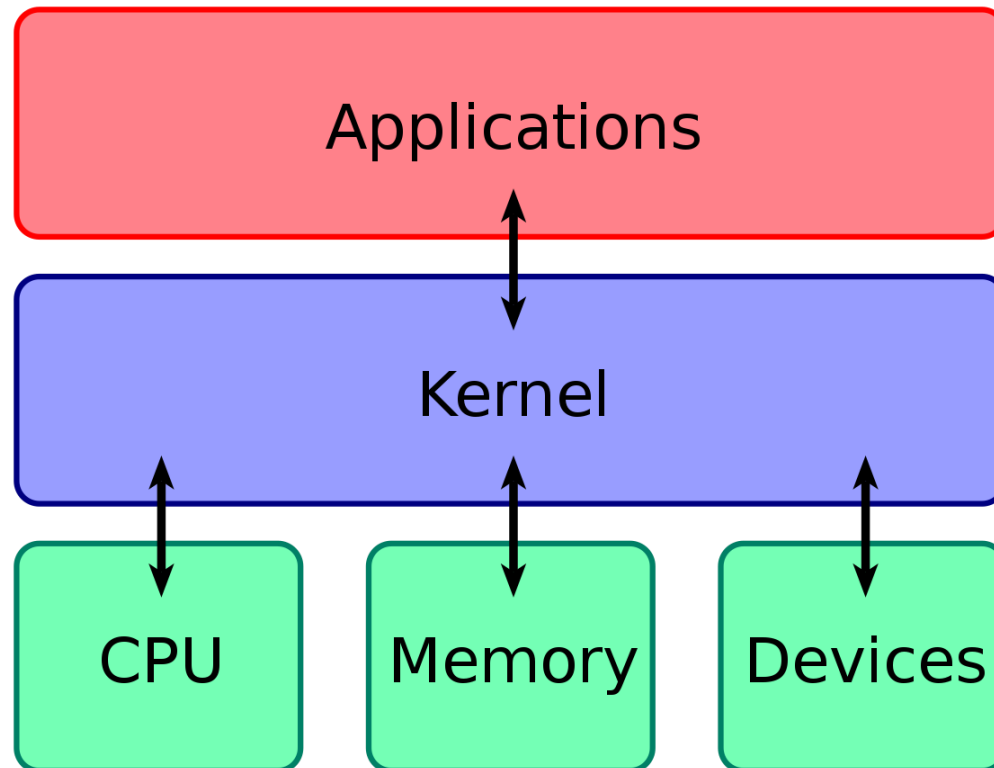
```
imagemMotor do Editor de Imagens(imagemnImg,Inteirooptar){
    imagemtemperatura;
    trocar(optar){
        caso1:
            temperatura=Afiado(nImg);
            quebrar;
        caso2:
            temperatura=Borrão(nImg);
            quebrar;
    }
    retornar temperatura;
}
```



Ponteiros de função

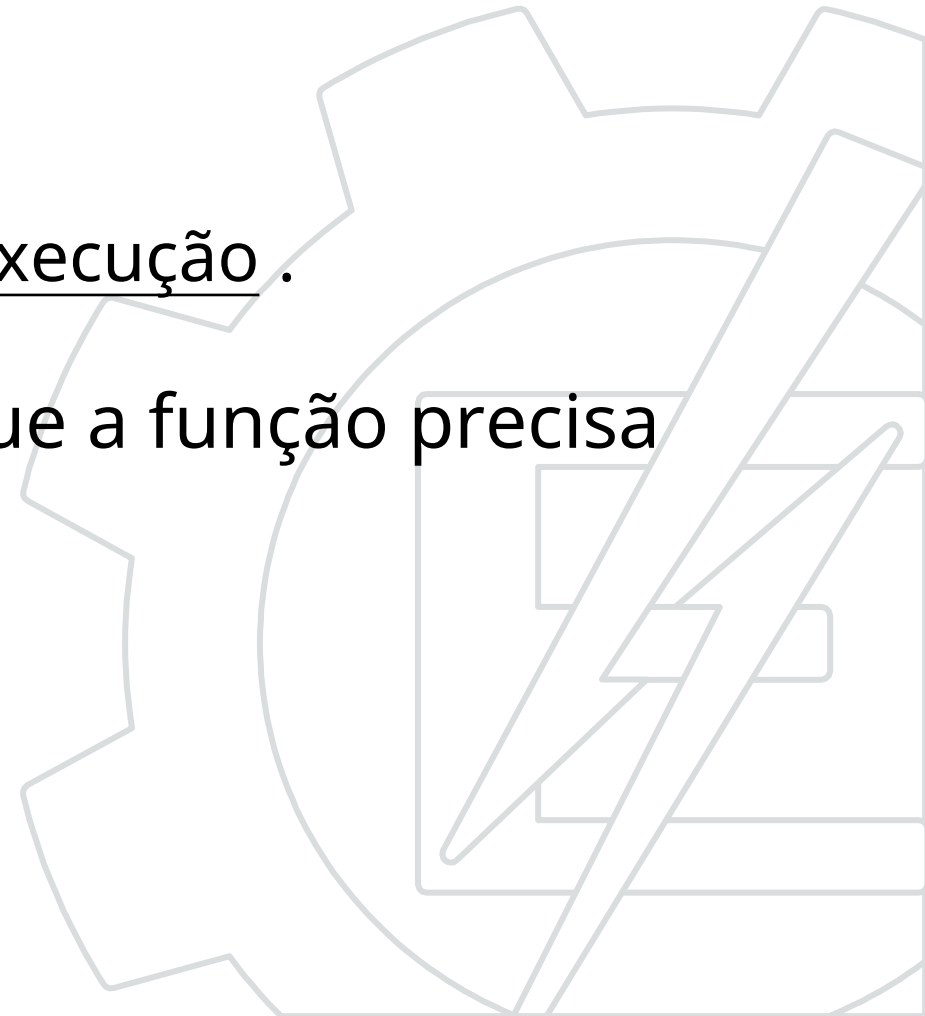


Como executar uma função que não é conhecido em tempo de compilação ?



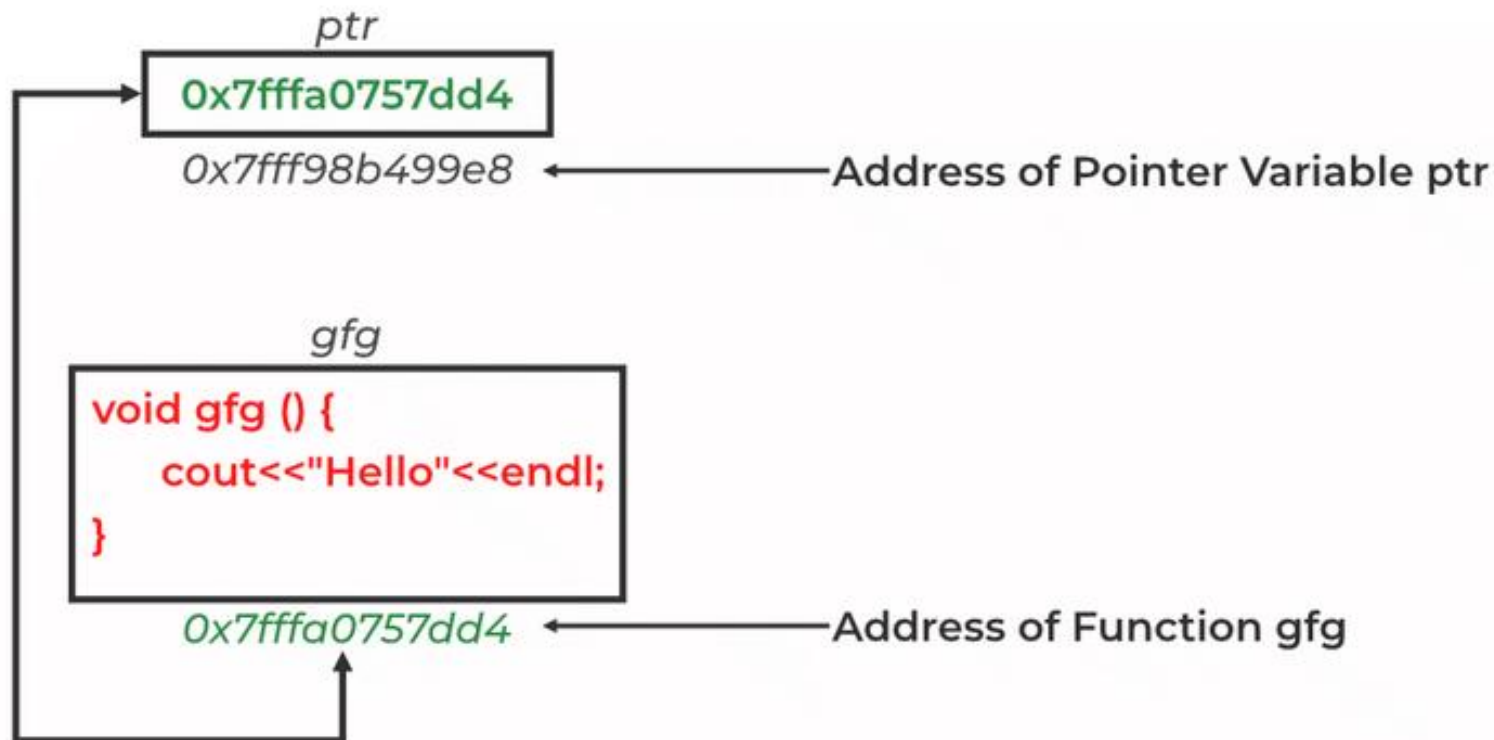
Como executar uma função que não é conhecido em tempo de compilação ?

- Saiba o endereço da função em tempo de execução .
- Empilhe corretamente os parâmetros que a função precisa
- Faça um chamada de função para este endereço



Ponteiros de função

- Trabalhar *quase* como um ponteiro normal
- Sua manipulação obedece a todas as regras de manipulação de ponteiros
- Mantenha o endereço do ponto inicial de uma função em vez do endereço de uma variável



Ponteiros de função

- Trabalhar *quase* como um ponteiro normal
- Sua manipulação obedece a todas as regras de manipulação de ponteiros
- Mantenha o endereço do ponto inicial de uma função em vez do endereço de uma variável
- O compilador não precisa saber a assinatura da função para passar os parâmetros corretos e o valor de retorno.
- Declaração estranha (é melhor use um typedef)

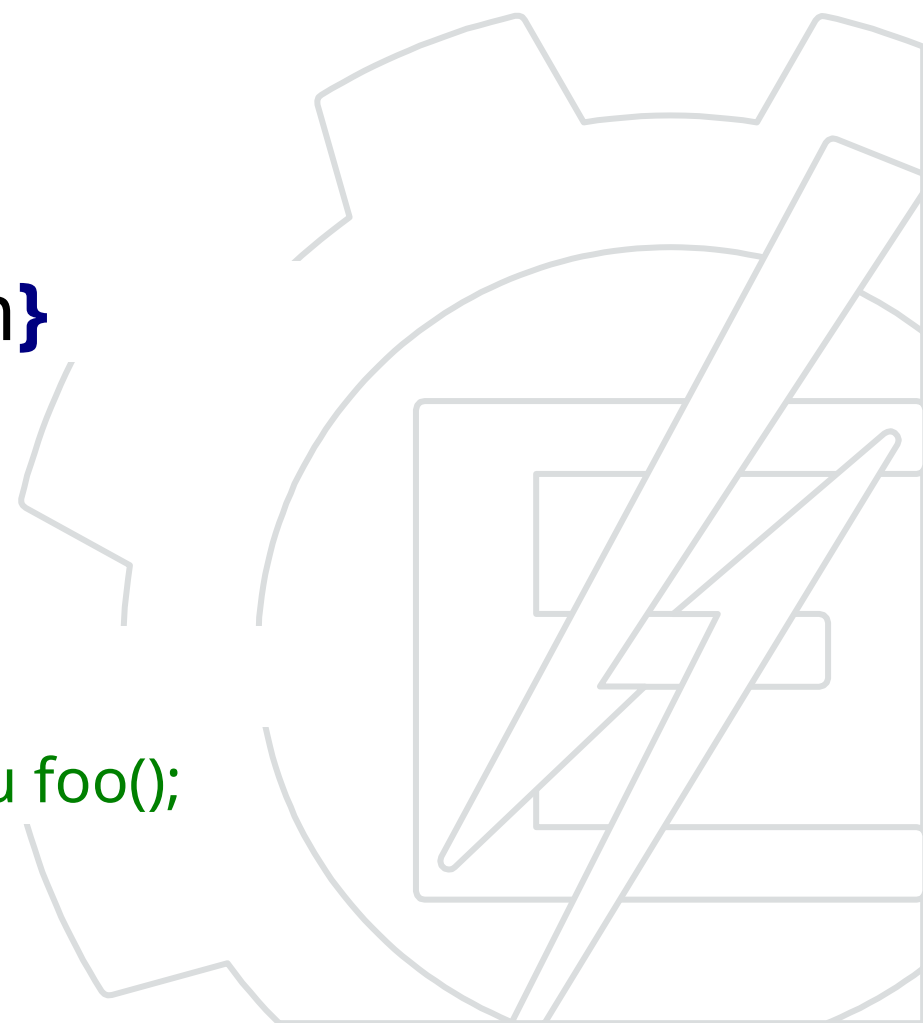
```
//definindo o tipo pointerTest //é um  
//ponteiro para uma função que: //  
//    não recebe nenhum parâmetro  
//    não retorna nenhum parâmetro  
vazio(*Teste de ponteiro)(vazio);
```



```
//definindo o tipo pointerTest //é um  
// ponteiro para uma função que: //  
//      não recebe nenhum parâmetro  
//      não retorna nenhum parâmetro  
vazio(*Teste de ponteiro)(vazio);
```

```
//Função a ser chamada  
vazio não(vazio){__asm NOP __endasm}
```

```
//criando uma variável pointerTest;  
(*Teste de ponteiro)(vazio)foo;  
foo=não; //foo recebe o endereço de nop  
(*foo)(); //chamando a função via ponteiro // ou foo();
```



```
//definindo o tipo pointerTest //é um  
ponteiro para uma função que: //
```

```
    não recebe nenhum parâmetro
```

```
//    não retorna nenhum parâmetro
```

```
//vazio(*Teste de ponteiro)(vazio); tipo  
definido vazio(*Teste de ponteiro)(vazio);
```



```
//definindo o tipo pointerTest //é um  
ponteiro para uma função que: //
```

```
    não recebe nenhum parâmetro
```

```
//    não retorna nenhum parâmetro
```

```
tipo definidovazio(*Teste de ponteiro)(vazio);
```

```
//Função a ser chamada
```

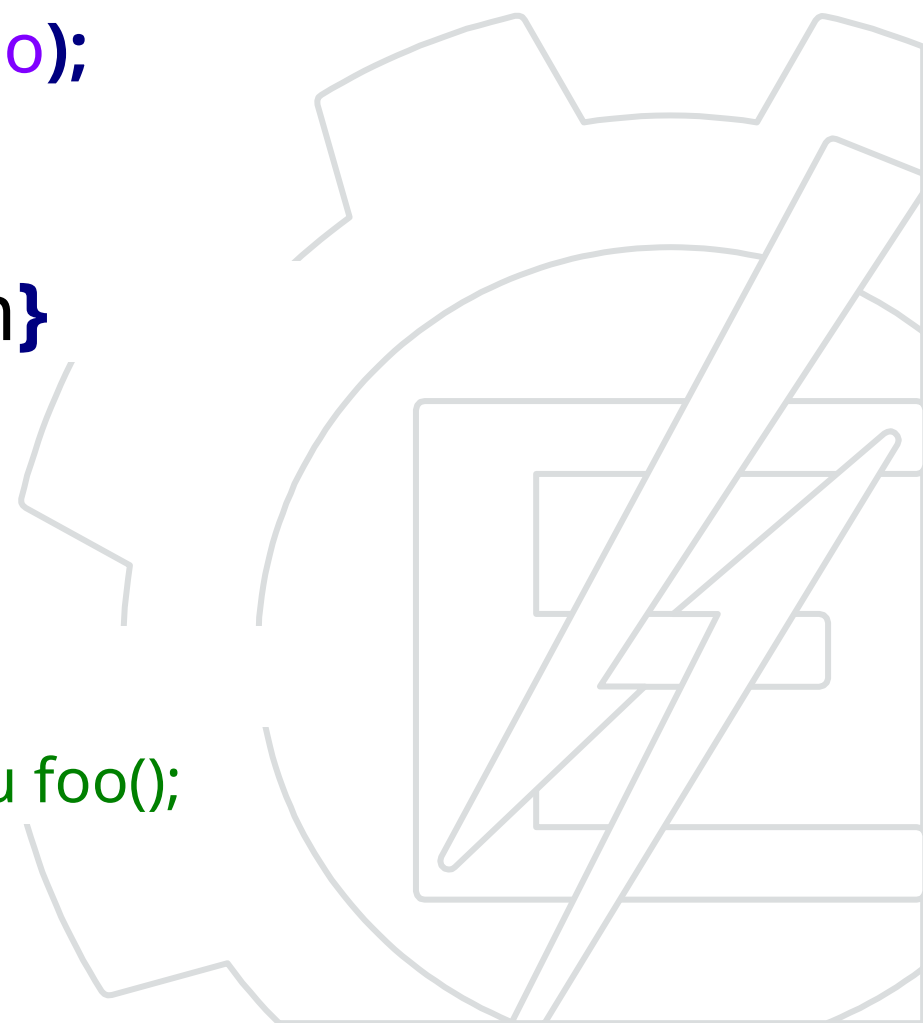
```
vazio(vazio){__asm NOP __endasm}
```

```
//criando uma variável pointerTest;
```

```
ponteiroTeste foo;
```

```
foo=não; //foo recebe o endereço de nop
```

```
(*foo)(); //chamando a função via ponteiro // ou foo();
```



Ponteiros de função

Recodificar o mecanismo do editor de imagens usando ponteiros de função



```
imagemBorrão(imagemnImg){}
```

```
imagemAfiado(imagemnImg){}
```

```
typedef imagem (*função ptr)(imagemnImg);
```

```
//motor de edição de imagens
```

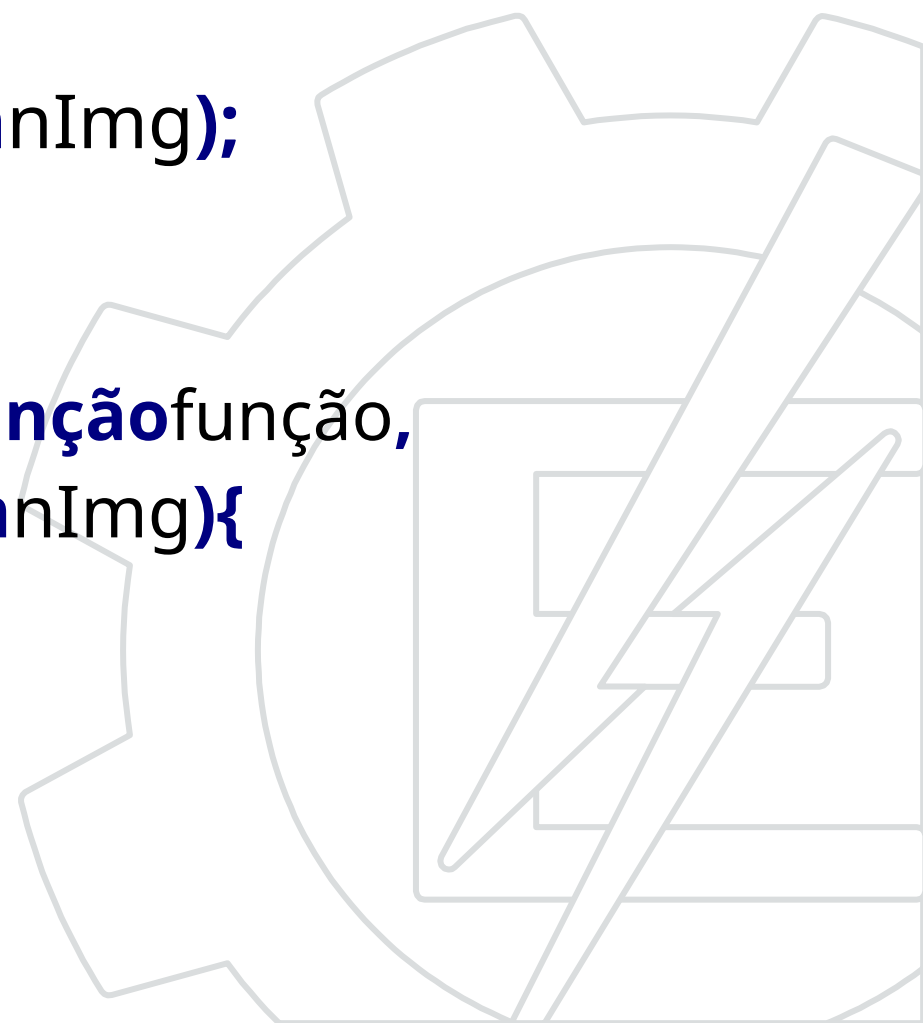
```
imagemMotor do Editor de Imagens(ptrFunçãofunção,  
                                imagemnImg){
```

```
    imagemtemperatura;
```

```
    temperatura= (*função)(nImg);
```

```
    retornar temperatura;
```

```
}
```




```
imagemBorrão(imagemnImg){}
```

```
imagemAfiado(imagemnImg){}
```

```
typedef imagem (*função ptr)(imagemnImg);
```

```
//motor de edição de imagens
```

```
imagemMotor do Editor de Imagens(ptrFunçãofunção,  
                                imagemnImg){
```

```
    imagemtemperatura;
```

```
    temperatura= (*função)(nImg);
```

```
    retornar temperatura;
```

```
}
```

Retornos:

imagem

Gravação
função para ptr & imagem

Ele executa a função A C com a imagem
função retorna o res

visão

Ponteiros de função

• Bom

- Novas adições de funções não alterar o motor
- O motor só precisa ser testado uma vez
- Pode alterar as implementações da função dinamicamente

• Ruim

- Mais complexo código (função ponteiros não são fáceis de entender para iniciantes)
- Provável *insetos*
- Falta de garantias de tempo de compilação (assinatura de função)

Ponteiros de função

```
int mais(int a, int b) {retornar a+b; } int  
menos(int a, int b) {retornar ab; }
```

```
int principal() {  
    int (*função)(int, int);  
    func = mais;  
    printf("%d\n", func(2,5));  
    retornar 0;  
}
```



Ponteiros de função

```
int mais(int a, int b) {retornar a+b; } int menos(int  
a, int b) {retornar ab; } int vezes(int a, int b)  
{retornar a*b; } int dividir(int a, int b) {retornar a/  
b; }
```

```
int principal() {  
    int (*função)(int, int);  
    func = mais;  
    printf("%d\n", func(2,5));  
    retornar 0;  
}
```



Ponteiros de função

```
int mais(int a, int b) {retornar a+b; } int menos(int  
a, int b) {retornar ab; } int vezes(int a, int b)  
{retornar a*b; } int dividir(int a, int b) {retornar a/  
b; }
```

Somente novos recursos/funções devem ser testados

```
int principal() {  
    int (*função)(int, int);  
    func = mais;  
    printf("%d\n", func(2,5));  
    retornar 0;  
}
```

Exercício

Usando ponteiros de função



Exercício

- Atualize a última estrutura de classe para incluir um ponteiro de função como um de seus membros.
- Crie uma função (ExecProc) que execute o ponteiro armazenado na "primeira" posição preenchida do buffer circular.
- Crie um main que execute os comandos para o lado:
- Crie três funções diferentes, cada uma imprimindo uma frase diferente.

```
# incluir "stdio.h"  
vazioprincipal (vazio){  
    adicionarProc(p1);  
    adicionarProc(p2);  
    adicionarProc(p3);  
    ExeProc();  
    RemoveProc();  
    ExeProc();  
    RemoveProc();  
    ExeProc();  
    RemoveProc();  
}
```

```
# incluir "stdio.h"
```

```
vazio principal(vazio){
```

```
    adicionarProc("proc1",0);
```

```
    adicionarProc("proc2",1);
```

```
    adicionarProc("proc3",2);
```

```
    removerProc();
```

```
    removerProc();
```

```
    removerProc();
```

```
}
```

Exercício



Exercício

- Atualize a última estrutura de classe para incluir um ponteiro de função como um de seus membros.
- Crie uma função (ExecProc) que execute o ponteiro armazenado na "primeira" posição preenchida do buffer circular.
- Crie um main que execute os comandos para o lado:
- Crie três funções diferentes, cada uma imprimindo uma frase diferente.

```
# incluir "stdio.h"  
vazioprincipal (vazio){  
    adicionarProc(p1);  
    adicionarProc(p2);  
    adicionarProc(p3);  
    ExeProc();  
    RemoveProc();  
    ExeProc();  
    RemoveProc();  
    ExeProc();  
    RemoveProc();  
}
```

```
typedef int (*função ptr)(vazio);
```

```
estrutura typedef{
```

```
    Caracteresnome;
```

```
    Inteirotempo;
```

```
    função ptrfunção;
```

```
}processo;
```

```
# definir BUFFERSIZE 10 buffer de  
processo[BUFFERSIZE];
```

```
Inteiroinício=0, fim=0;
```



```
vazioadicionarProc(processo*nProcesso,InteironTempo,função ptr  
fPonteiro){
```

```
    se( ((fim+1)%BUFFERSIZE) != início){
```

```
        buffer[fim].nome = nProcesso;
```

```
        buffer[fim].tempo = nTempo;
```

```
        buffer[fim].pFunc = fPointer; fim =  
        (fim+1)%(BUFFERSIZE);
```

```
    }
```

Exercício

```
}
```



```
vazioremoveProc (vazio){  
    se(início != fim){  
        início = (início + 1)%(BUFFERSIZE);  
    }  
}
```

```
vazioexecutar(vazio){  
    se(início != fim){  
        buffer[início].func();  
    }  
}
```

Exercício



```
vaziofunc1(vazio){imprimirf("f1 \n");}
```

```
vaziofunc2(vazio){imprimirf("f2 \n");}
```

```
vaziofunc3(vazio){imprimirf("f3 \n");}
```



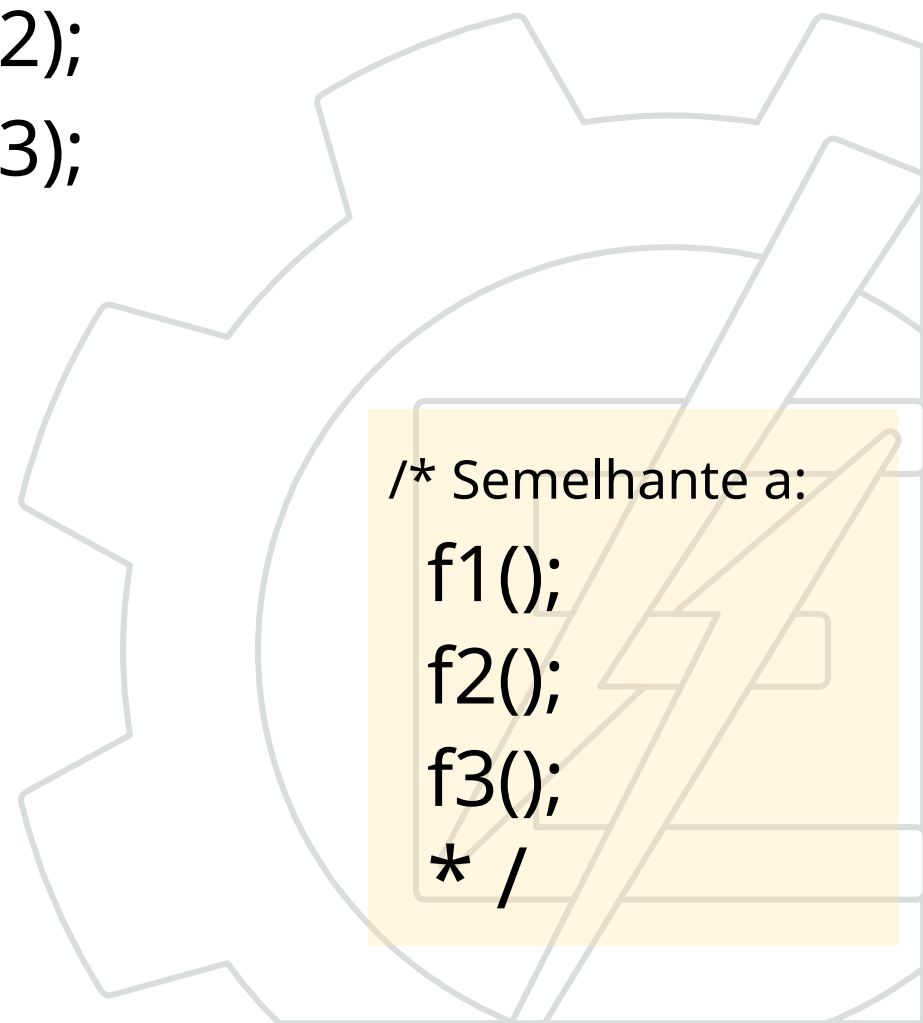
incluir "stdio.h"

```
vazioprincipal (vazio){  
    adicionarProc("Processo 1", 1, func1);  
    adicionarProc("Processo 2", 2, func2);  
    adicionarProc("Processo 3", 3, func3);  
    exec();  
    removerProc();  
    executar();  
    removerProc();  
    executar();  
    removerProc();  
}
```

incluir "stdio.h"

```
vazioprincipal (vazio){  
    adicionarProc("Processo 1", 1, func1);  
    adicionarProc("Processo 2", 2, func2);  
    adicionarProc("Processo 3", 3, func3);  
    exec();  
    removerProc();  
    executar();  
    removerProc();  
    executar();  
    removerProc();  
}
```

Exercício



```
/* Semelhante a:  
f1();  
f2();  
f3();  
* /
```

Exercício

Passo a passo




```
typedef int (*função ptr)(vazio);
```

```
estrutura typedef{  
    Caracteres nome;  
    Inteiro tempo;  
    função ptr função;  
}processo;
```

```
# definir BUFFERSIZE 10 buffer de  
processo[BUFFERSIZE];
```

```
Inteiro início=0, fim=0;
```

amortecedor

| | nome | tempo | função |
|---|------------|---------|------------|
| | Caracteres | Inteiro | função ptr |
| 0 | ? | ? | ? |
| 1 | ? | ? | ? |
| 2 | ? | ? | ? |
| 3 | ? | ? | ? |
| 4 | ? | ? | ? |
| 5 | ? | ? | ? |
| 6 | ? | ? | ? |
| 7 | ? | ? | ? |
| 8 | ? | ? | ? |
| 9 | ? | ? | ? |

começar

0

fim

0

```
# incluir "stdio.h"
```

```
vazio principal (vazio){  
    adicionarProc("Processo 1", 1, func1);  
    adicionarProc("Processo 2", 2, func2);  
    adicionarProc("Processo 3", 3, func3);  
    exec();  
    removerProc();  
    executar();  
    removerProc();  
    executar();  
    removerProc();  
}
```

amortecedor

| | nome | tempo | função |
|---|------------|---------|------------|
| | Caracteres | Inteiro | função ptr |
| 0 | ? | ? | ? |
| 1 | ? | ? | ? |
| 2 | ? | ? | ? |
| 3 | ? | ? | ? |
| 4 | ? | ? | ? |
| 5 | ? | ? | ? |
| 6 | ? | ? | ? |
| 7 | ? | ? | ? |
| 8 | ? | ? | ? |
| 9 | ? | ? | ? |

começar

0

fim

0

```
vazio adicionarProc(processo*nProcesso,InteironTempo,  
função ptrfPonteiro){
```

```
    se( ((fim+1)%BUFFERSIZE) != início){
```

```
        buffer[fim].nome = nProcesso;
```

```
        buffer[fim].tempo = nTempo;
```

```
        buffer[fim].func = fPonteiro;
```

```
        fim = (fim+1)%(BUFFERSIZE);
```

```
    }
```

```
}
```

amortecedor

nome**tempo****função**

Caracteres

Inteiro

função ptr

| | | | |
|---|---|---|---|
| 0 | ? | ? | ? |
| 1 | ? | ? | ? |
| 2 | ? | ? | ? |
| 3 | ? | ? | ? |
| 4 | ? | ? | ? |
| 5 | ? | ? | ? |
| 6 | ? | ? | ? |
| 7 | ? | ? | ? |
| 8 | ? | ? | ? |
| 9 | ? | ? | ? |

começar

0

fim

0

```
vazio adicionarProc(processo*nProcesso,InteironTempo,  
função ptrponteiro){
```

```
    se( ((fim+1)%BUFFERSIZE) != início){
```

```
        buffer[fim].nome = nProcesso;
```

```
        buffer[fim].tempo = nTempo;
```

```
        buffer[end].func = ponteiro;
```

```
        fim = (fim+1)%(BUFFERSIZE);
```

```
    }
```

```
}
```

amortecedor

nome**tempo****função**

Caracteres

Inteiro

função ptr

| | | | |
|---|------------|---|---------|
| 0 | Processo 1 | 1 | função1 |
| 1 | ? | ? | ? |
| 2 | ? | ? | ? |
| 3 | ? | ? | ? |
| 4 | ? | ? | ? |
| 5 | ? | ? | ? |
| 6 | ? | ? | ? |
| 7 | ? | ? | ? |
| 8 | ? | ? | ? |
| 9 | ? | ? | ? |

começar

0

fim

1

```
vazio adicionarProc(processo*nProcesso,InteironTempo,  
função ptrponteiro){
```

```
    se( ((fim+1)%BUFFERSIZE) != início){
```

```
        buffer[fim].nome = nProcesso;
```

```
        buffer[fim].tempo = nTempo;
```

```
        buffer[end].func = ponteiro;
```

```
        fim = (fim+1)%(BUFFERSIZE);
```

```
    }
```

```
}
```

amortecedor

| | nome <small>Caracteres</small> | tempo <small>Inteiro</small> | função <small>função ptr</small> |
|---|--|--|--|
| 0 | Processo 1 | 1 | função1 |
| 1 | Processo2 | 2 | função2 |
| 2 | ? | ? | ? |
| 3 | ? | ? | ? |
| 4 | ? | ? | ? |
| 5 | ? | ? | ? |
| 6 | ? | ? | ? |
| 7 | ? | ? | ? |
| 8 | ? | ? | ? |
| 9 | ? | ? | ? |

começar

0

fim

2

```
vazio adicionarProc(processo*nProcesso,InteironTempo,  
função ptrponteiro){
```

```
    se( ((fim+1)%BUFFERSIZE) != início){
```

```
        buffer[fim].nome = nProcesso;
```

```
        buffer[fim].tempo = nTempo;
```

```
        buffer[end].func = ponteiro;
```

```
        fim = (fim+1)%(BUFFERSIZE);
```

```
    }
```

```
}
```

amortecedor

| | nome | tempo | função |
|---|-------------|--------------|---------------|
| | Caracteres | Inteiro | função ptr |
| 0 | Processo 1 | 1 | função1 |
| 1 | Processo2 | 2 | função2 |
| 2 | Processo3 | 3 | função3 |
| 3 | ? | ? | ? |
| 4 | ? | ? | ? |
| 5 | ? | ? | ? |
| 6 | ? | ? | ? |
| 7 | ? | ? | ? |
| 8 | ? | ? | ? |
| 9 | ? | ? | ? |

começar

0

fim

3

```
# incluir "stdio.h"
```

```
vazio principal (vazio){  
    adicionarProc("Processo 1", 1, func1);  
    adicionarProc("Processo 2", 2, func2);  
    adicionarProc("Processo 3", 3, func3);  
    exec();  
    removerProc();  
    executar();  
    removerProc();  
    executar();  
    removerProc();  
}
```

amortecedor

| | nome | tempo | função |
|---|------------|---------|------------|
| | Caracteres | Inteiro | função ptr |
| 0 | Processo 1 | 1 | função1 |
| 1 | Processo2 | 2 | função2 |
| 2 | Processo3 | 3 | função3 |
| 3 | ? | ? | ? |
| 4 | ? | ? | ? |
| 5 | ? | ? | ? |
| 6 | ? | ? | ? |
| 7 | ? | ? | ? |
| 8 | ? | ? | ? |
| 9 | ? | ? | ? |

começar

0

fim

3

incluir "stdio.h"

```
vazioprincipal (vazio){\n    adicionarProc("Processo 1", 1, func1);\n    adicionarProc("Processo 2", 2, func2);\n    adicionarProc("Processo 3", 3, func3);\n    exec();\n    removerProc();\n    executar();\n    removerProc();\n    executar();\n    removerProc();\n}
```

amortecedor

| | nome | tempo | função |
|---|-------------|--------------|---------------|
| | Caracteres | Inteiro | função ptr |
| 0 | Processo 1 | 1 | função1 |
| 1 | Processo2 | 2 | função2 |
| 2 | Processo3 | 3 | função3 |
| 3 | ? | ? | ? |
| 4 | ? | ? | ? |
| 5 | ? | ? | ? |
| 6 | ? | ? | ? |
| 7 | ? | ? | ? |
| 8 | ? | ? | ? |
| 9 | ? | ? | ? |

começar

0

fim

3


```
vazioexecutar(vazio){  
    se(início != fim){  
        buffer[início].func();  
    }  
}
```

```
vazioremoverProc (vazio){  
    se(início != fim){  
        início = (início + 1)%(BUFFERSIZE);  
    }  
}
```

```
vaziofunc1(vazio){imprimirf("f1 \n");}
```

```
vaziofunc2(vazio){imprimirf("f2 \n");}
```

```
vaziofunc3(vazio){imprimirf("f3 \n");}
```



amortecedor

| | nome | tempo | função |
|--|-------------|--------------|---------------|
| | Caracteres | Inteiro | função ptr |

| | | | |
|---|------------|---|---------|
| 0 | Processo 1 | 1 | função1 |
| 1 | Processo2 | 2 | função2 |
| 2 | Processo3 | 3 | função3 |
| 3 | ? | ? | ? |
| 4 | ? | ? | ? |
| 5 | ? | ? | ? |
| 6 | ? | ? | ? |
| 7 | ? | ? | ? |
| 8 | ? | ? | ? |
| 9 | ? | ? | ? |

começar

0

fim

3

```
vazioexecutar(vazio){  
    se(início != fim){  
        buffer[início].func();  
    }  
}
```

```
vazioremoverProc (vazio){  
    se(início != fim){  
        início = (início + 1)%(BUFFERSIZE);  
    }  
}
```

```
vaziofunc1(vazio){imprimirf("f1 \n");}
```

```
vaziofunc2(vazio){imprimirf("f2 \n");}
```

```
vaziofunc3(vazio){imprimirf("f3 \n");}
```



amortecedor

| | nome <small>Caracteres</small> | tempo <small>Inteiro</small> | função <small>função ptr</small> |
|---|--|--|--|
| 0 | Processo 1 | 1 | função1 |
| 1 | Processo2 | 2 | função2 |
| 2 | Processo3 | 3 | função3 |
| 3 | ? | ? | ? |
| 4 | ? | ? | ? |
| 5 | ? | ? | ? |
| 6 | ? | ? | ? |
| 7 | ? | ? | ? |
| 8 | ? | ? | ? |
| 9 | ? | ? | ? |

começar

1

fim

3

```
# incluir "stdio.h"
```

```
vazio principal (vazio){
```

```
    adicionarProc("Processo 1", 1, func1);
```

```
    adicionarProc("Processo 2", 2, func2);
```

```
    adicionarProc("Processo 3", 3, func3);
```

```
    exec();
```

```
    removerProc();
```

```
    executar();
```

```
    removerProc();
```

```
    executar();
```

```
    removerProc();
```

```
}
```

amortecedor

nome

tempo

função

Caracteres

Inteiro

função ptr

| | | | |
|---|------------|---|---------|
| 0 | Processo 1 | 1 | função1 |
| 1 | Processo2 | 2 | função2 |
| 2 | Processo3 | 3 | função3 |
| 3 | ? | ? | ? |
| 4 | ? | ? | ? |
| 5 | ? | ? | ? |
| 6 | ? | ? | ? |
| 7 | ? | ? | ? |
| 8 | ? | ? | ? |
| 9 | ? | ? | ? |

começar

1

fim

3

```
vazioexecutar(vazio){  
    se(início != fim){  
        buffer[início].func();  
    }  
}
```

```
vazioremoverProc (vazio){  
    se(início != fim){  
        início = (início + 1)%(BUFFERSIZE);  
    }  
}
```

```
vaziofunc1(vazio){imprimirf("f1 \n");}
```

```
vaziofunc2(vazio){imprimirf("f2 \n");}
```

```
vaziofunc3(vazio){imprimirf("f3 \n");}
```



amortecedor

| | nome <small>Caracteres</small> | tempo <small>Inteiro</small> | função <small>função ptr</small> |
|---|--|--|--|
| 0 | Processo 1 | 1 | função1 |
| 1 | Processo2 | 2 | função2 |
| 2 | Processo3 | 3 | função3 |
| 3 | ? | ? | ? |
| 4 | ? | ? | ? |
| 5 | ? | ? | ? |
| 6 | ? | ? | ? |
| 7 | ? | ? | ? |
| 8 | ? | ? | ? |
| 9 | ? | ? | ? |

começar

1

fim

3

```
vazioexecutar(vazio){  
    se(início != fim){  
        buffer[início].func();  
    }  
}
```

```
vazioremoverProc (vazio){  
    se(início != fim){  
        início = (início + 1)%(BUFFERSIZE);  
    }  
}
```

```
vaziofunc1(vazio){imprimirf("f1 \n");}
```

```
vaziofunc2(vazio){imprimirf("f2 \n");}
```

```
vaziofunc3(vazio){imprimirf("f3 \n");}
```



amortecedor

| | nome <small>Caracteres</small> | tempo <small>Inteiro</small> | função <small>função ptr</small> |
|---|--|--|--|
| 0 | Processo 1 | 1 | função1 |
| 1 | Processo2 | 2 | função2 |
| 2 | Processo3 | 3 | função3 |
| 3 | ? | ? | ? |
| 4 | ? | ? | ? |
| 5 | ? | ? | ? |
| 6 | ? | ? | ? |
| 7 | ? | ? | ? |
| 8 | ? | ? | ? |
| 9 | ? | ? | ? |

começar

2

fim

3

incluir "stdio.h"

```
vazioprincipal (vazio){  
    adicionarProc("Processo 1", 1, func1);  
    adicionarProc("Processo 2", 2, func2);  
    adicionarProc("Processo 3", 3, func3);  
    exec();  
    removerProc();  
    executar();  
    removerProc();  
    executar();  
    removerProc();  
}
```

amortecedor

| | nome | tempo | função |
|---|------------|---------|------------|
| | Caracteres | Inteiro | função ptr |
| 0 | Processo 1 | 1 | função1 |
| 1 | Processo2 | 2 | função2 |
| 2 | Processo3 | 3 | função3 |
| 3 | ? | ? | ? |
| 4 | ? | ? | ? |
| 5 | ? | ? | ? |
| 6 | ? | ? | ? |
| 7 | ? | ? | ? |
| 8 | ? | ? | ? |
| 9 | ? | ? | ? |

começar

2

fim

3

```
vazioexecutar(vazio){  
    se(início != fim){  
        buffer[início].func();  
    }  
}
```

```
vazioremoverProc (vazio){  
    se(início != fim){  
        início = (início + 1)%(BUFFERSIZE);  
    }  
}
```

```
vaziofunc1(vazio){imprimirf("f1 \n");}
```

```
vaziofunc2(vazio){imprimirf("f2 \n");}
```

```
vaziofunc3(vazio){imprimirf("f3 \n");}
```



amortecedor

| | nome | tempo | função |
|---|-------------|--------------|---------------|
| | Caracteres | Inteiro | função ptr |
| 0 | Processo 1 | 1 | função1 |
| 1 | Processo2 | 2 | função2 |
| 2 | Processo3 | 3 | função3 |
| 3 | ? | ? | ? |
| 4 | ? | ? | ? |
| 5 | ? | ? | ? |
| 6 | ? | ? | ? |
| 7 | ? | ? | ? |
| 8 | ? | ? | ? |
| 9 | ? | ? | ? |

começar

2

fim

3

```
vazioexecutar(vazio){  
    se(início != fim){  
        buffer[início].func();  
    }  
}
```

```
vazioremoveProc (vazio){  
    se(início != fim){  
        início = (início + 1)%(BUFFERSIZE);  
    }  
}
```

```
vaziofunc1(vazio){imprimirf("f1 \n");}
```

```
vaziofunc2(vazio){imprimirf("f2 \n");}
```

```
vaziofunc3(vazio){imprimirf("f3 \n");}
```



amortecedor

| | nome <small>Caracteres</small> | tempo <small>Inteiro</small> | função <small>função ptr</small> |
|---|--|--|--|
| 0 | Processo 1 | 1 | função1 |
| 1 | Processo2 | 2 | função2 |
| 2 | Processo3 | 3 | função3 |
| 3 | ? | ? | ? |
| 4 | ? | ? | ? |
| 5 | ? | ? | ? |
| 6 | ? | ? | ? |
| 7 | ? | ? | ? |
| 8 | ? | ? | ? |
| 9 | ? | ? | ? |

começar

3

fim

3


```
# incluir "stdio.h"
```

```
vazio principal (vazio){  
    adicionarProc("Processo 1", 1, func1);  
    adicionarProc("Processo 2", 2, func2);  
    adicionarProc("Processo 3", 3, func3);  
    exec();  
    removerProc();  
    executar();  
    removerProc();  
    executar();  
    removerProc();  
}
```

amortecedor

| | nome | tempo | função |
|---|------------|---------|------------|
| | Caracteres | Inteiro | função ptr |
| 0 | Processo 1 | 1 | função1 |
| 1 | Processo2 | 2 | função2 |
| 2 | Processo3 | 3 | função3 |
| 3 | ? | ? | ? |
| 4 | ? | ? | ? |
| 5 | ? | ? | ? |
| 6 | ? | ? | ? |
| 7 | ? | ? | ? |
| 8 | ? | ? | ? |
| 9 | ? | ? | ? |

começar

3

fim

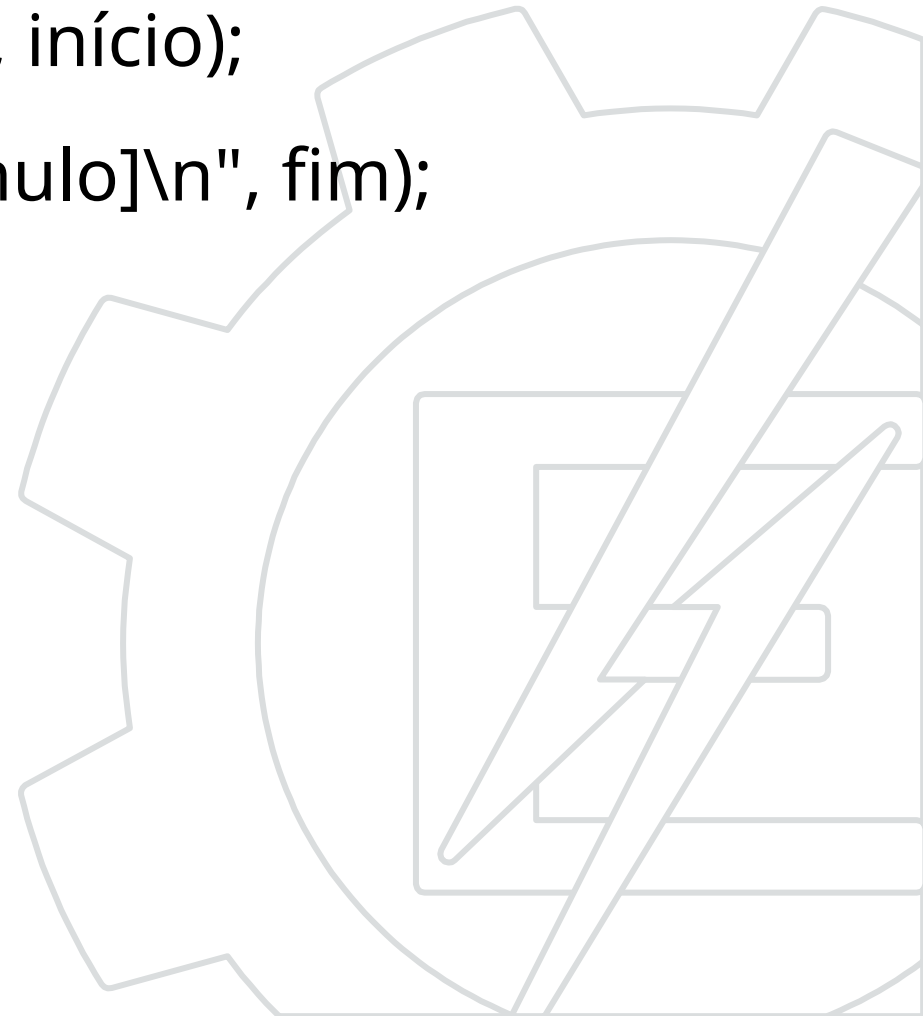
3

Exercício

Opções / Mudanças



```
vazioexecutar(vazio){  
    se(início != fim){  
        imprimirf("Processo - ID atual%e\n", início);  
        imprimirf("Processo - Último ID%e[nulo]\n", fim);  
        imprimirf("<-----\n", fim);  
        amortecedor[começar].função();  
        imprimirf("----->\n", fim);  
    }  
}
```



incluir "stdio.h"

```
vazioprincipal (vazio){  
    adicionarProc("Processo 1", 1, func1);  
    adicionarProc("Processo 2", 2, func2);  
    adicionarProc("Processo 3", 3, func3);  
    exec();  
    executar();  
    removerProc();  
    executar();  
    removerProc();  
    executar();  
    removerProc();  
}
```



incluir "stdio.h"

vazioprincipal (**vazio**){

adicionarProc("Processo 1", 1, func1);

adicionarProc("Processo 2", 2, func2);

adicionarProc("Processo 3", 3, func3);

exec();

removerProc();

adicionarProc("Processo 4", 4, func1);

adicionarProc("Processo 5", 5, func2);

exec();

removerProc();

executar();

adicionarProc("Processo 5", 5,

func2); removeProc();

}



incluir "stdio.h"

```
vazio principal (vazio){  
    adicionarProc("Processo 1", 1, func1);  
    adicionarProc("Processo 2", 2, func2);  
    adicionarProc("Processo 3", 3, func3);  
    exec();  
    removerProc();  
    se (<condição>){  
        removerProc();  
        adicionarProc("Processo 4", 4, func1);  
        senão{  
            adicionarProc("Processo 5", 5, função2);  
        }  
    }  
    executar();  
    removerProc();  
    executar();  
    removerProc();  
}
```



Linus Torvalds

núcleo 0,01 (1991)



Bibliografia

- Denardin, GB; Barriquello, CH **Sistemas operacionais de tempo real e sua aplicação em sistemas embarcados**. 1ª ed. Editora Blucher. ISBN: 9788521213970. <https://plataforma.bvirtual.com.br/Acervo/Publicacao/169968>
- Tanenbaum, AS **Sistemas Operacionais Modernos**. 3ª ed. 674 páginas. São Paulo: Pearson. ISBN: 9788576052371.
<https://plataforma.bvirtual.com.br/Acervo/Publicacao/1233>
- Almeida, Moraes, Serafim e Gomes. **Programação de Sistemas Embarcados**. 2ª ed. Editora GEN LTC. ISBN: 9788595159105.
<https://cengagebrasil.vitalsource.com/books/9788595159112>



Integrado

Sistemas Operacionais

Tampão circular

protoCore 0,00001

Prof. Otávio Gomes
otavio.gomes@unifei.edu.br

 /otavio-gomes

