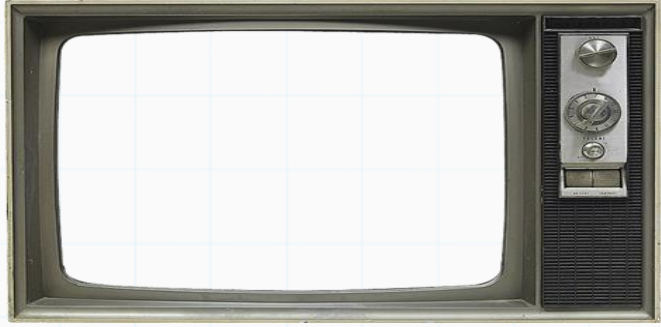
Programação Estruturada

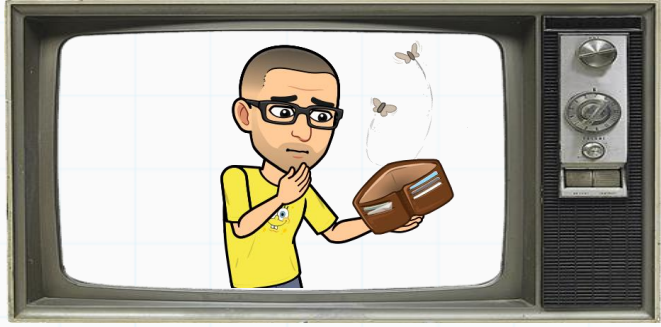


Professor : Yuri Frota

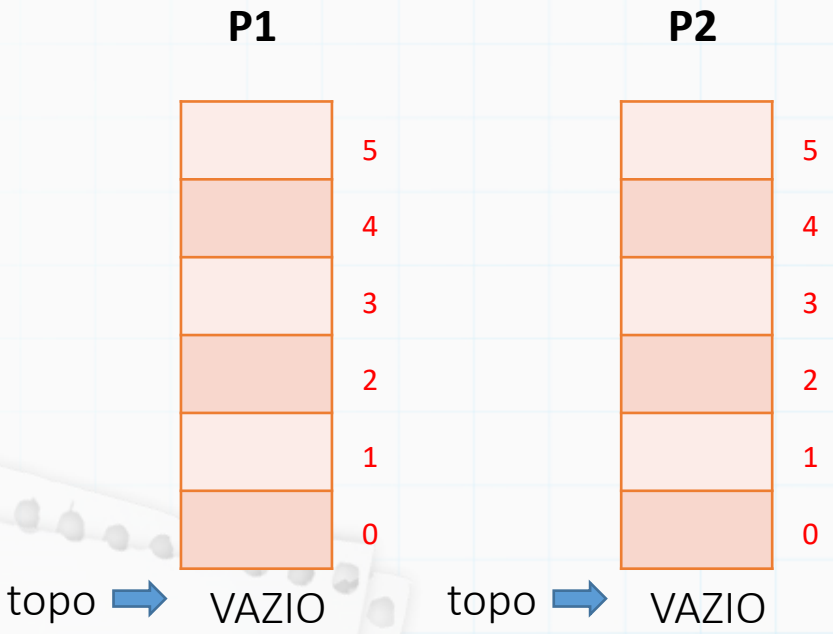
yuri@ic.uff.br

- Utilize o arquivo fornecido tad.c com o TAD básico de pilhas e filas para fazer as questões a seguir

TAD - Pilha

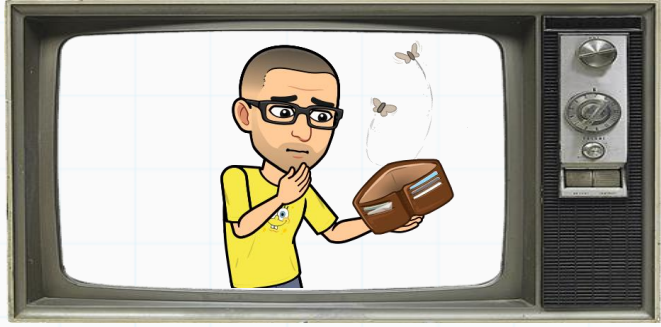


1) Filas e Pilhas: Dados duas pilhas P1 e P2, simule uma estrutura de fila, escrevendo as funções de entrar e sair da fila. Veja o exemplo:

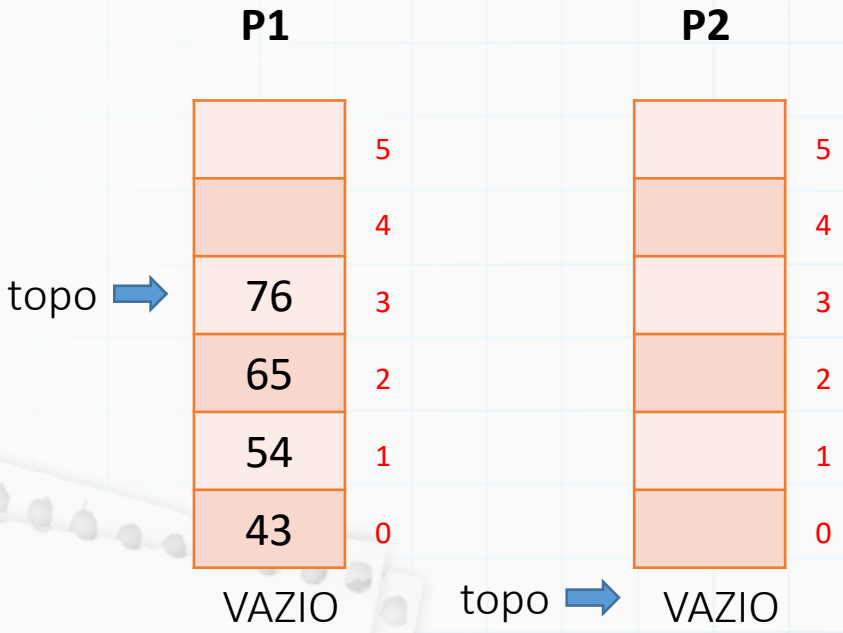


- Para entrar na fila, vamos sempre inserir na primeira pilha P1
  - Vamos inserir 43, 54, 65 e 76

TAD - Pilha

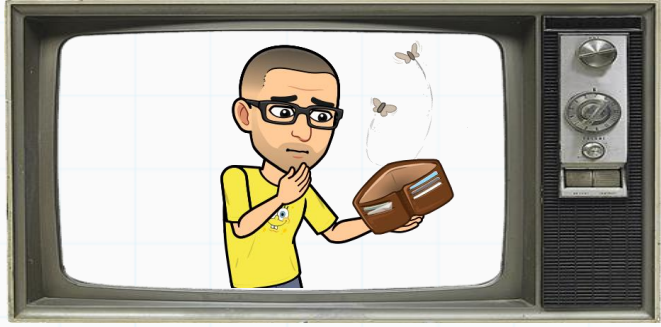


1) Filas e Pilhas: Dados duas pilhas P1 e P2, simule uma estrutura de fila, escrevendo as funções de entrar e sair da fila. Veja o exemplo:

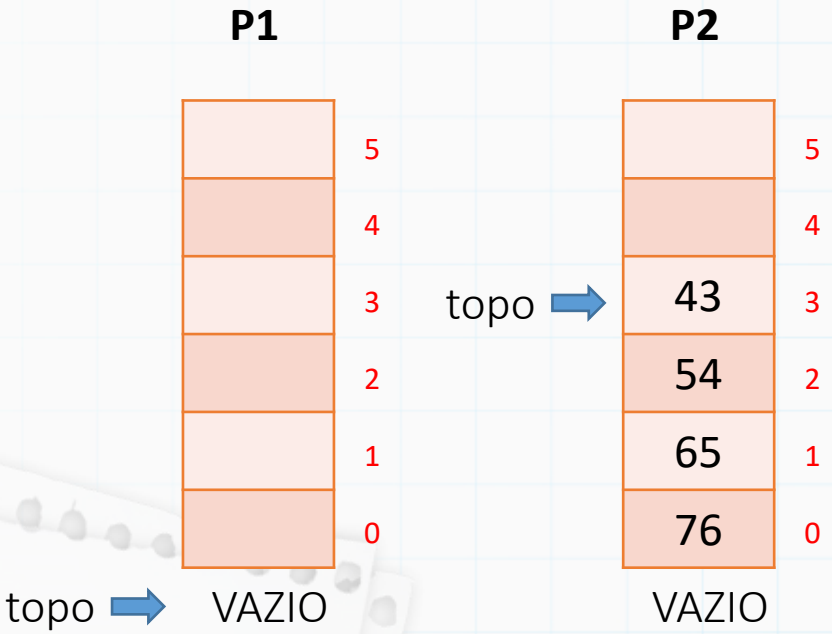


- Para entrar na fila, vamos sempre inserir na primeira pilha P1
  - Vamos inserir 43, 54, 65 e 76
- Para sair, o primeiro teria que ser o 43, então vamos jogar tudo em P2 (pop de P1 e push em P2)

TAD - Pilha

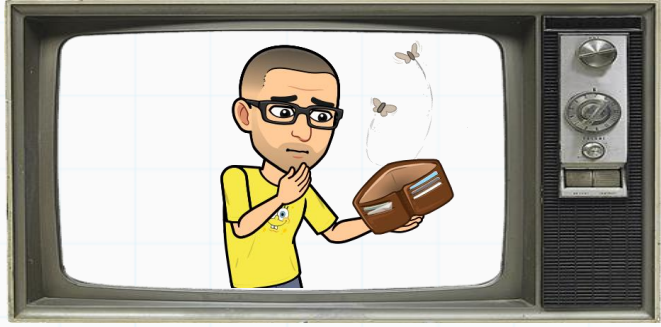


1) Filas e Pilhas: Dados duas pilhas P1 e P2, simule uma estrutura de fila, escrevendo as funções de entrar e sair da fila. Veja o exemplo:

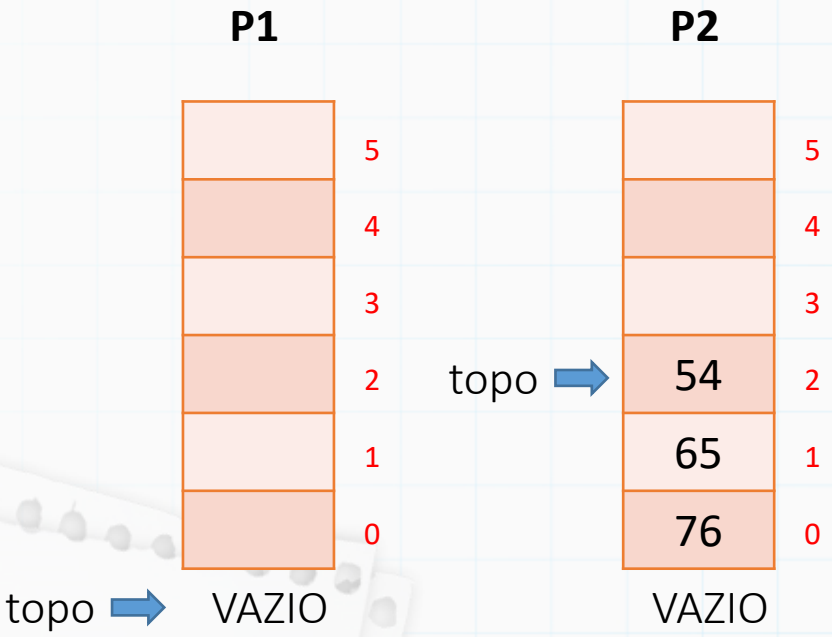


- Para entrar na fila, vamos sempre inserir na primeira pilha P1
  - Vamos inserir 43, 54, 65 e 76
- Para sair, o primeiro teria que ser o 43, então vamos jogar tudo em P2 (pop de P1 e push em P2), agora podemos tirar de P2 que equivale ao primeiro da fila

TAD - Pilha



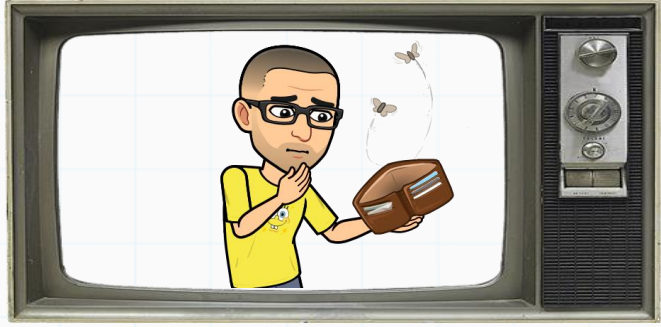
1) Filas e Pilhas: Dados duas pilhas P1 e P2, simule uma estrutura de fila, escrevendo as funções de entrar e sair da fila. Veja o exemplo:



- Para entrar na fila, vamos sempre inserir na primeira pilha P1
  - Vamos inserir 43, 54, 65 e 76
- Para sair, o primeiro teria que ser o 43, então vamos jogar tudo em P2 (pop de P1 e push em P2), agora podemos tirar de P2 que equivale ao primeiro da fila
- O processo de sair da fila pode ser definido como:
  - Se P2 não esta vazia, sai de P2
  - Senão, joga de P1 para P2 e tira de P2
  - Se P1 também vazia, então fila vazia.
- Veja a execução a seguir

TAD - Pilha

1) Filas e Pilhas



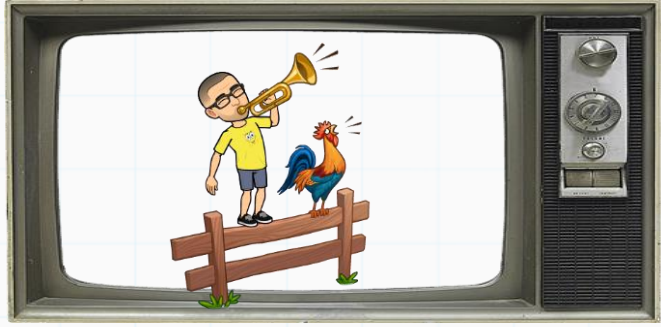
Código da main.c

```
int main()
{
    int tam=10;
    pilha P1, P2;
    cria_pilha(&P1, tam);
    cria_pilha(&P2, tam);

    entra_fila_pilha (&P1, &P2, 43);
    entra_fila_pilha (&P1, &P2, 54);
    entra_fila_pilha (&P1, &P2, 65);
    entra_fila_pilha (&P1, &P2, 76);

    printf("sai = %d\n", sai_fila_pilha (&P1, &P2));
    printf("sai = %d\n", sai_fila_pilha (&P1, &P2));

    entra_fila_pilha (&P1, &P2, 100);
    entra_fila_pilha (&P1, &P2, 200);

    printf("sai = %d\n", sai_fila_pilha (&P1, &P2));
    printf("sai = %d\n", sai_fila_pilha (&P1, &P2));
    printf("sai = %d\n", sai_fila_pilha (&P1, &P2));
    printf("sai = %d\n", sai_fila_pilha (&P1, &P2));
    printf("sai = %d\n", sai_fila_pilha (&P1, &P2));

    return 0;
}
```

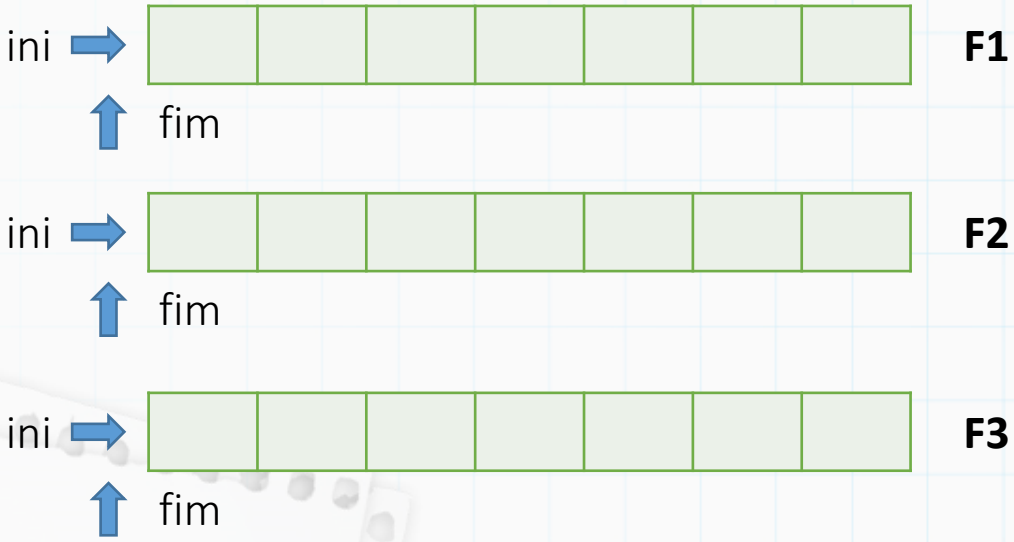
Exemplo de execução:

```
sai = 43
sai = 54
sai = 65
sai = 76
sai = 100
sai = 200
fila vazia
sai = -1
```

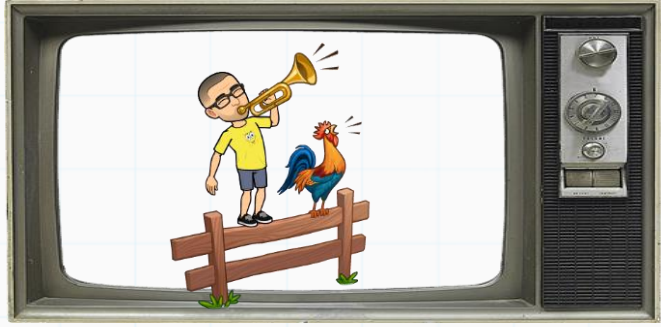
TAD - Pilha



2) Filas de Prioridade em 3 níveis: Vamos considerar uma fila que tem 3 níveis de prioridade (1,2 e 3), quer dizer que para alguém ser atendido na fila (sair da fila), primeiro são considerados as pessoas de menor prioridade (1), depois de 2, e por ultimo 3. Essa fila prioritária em 3 níveis pode ser implementada usando 3 filas de prioridade, veja o exemplo:

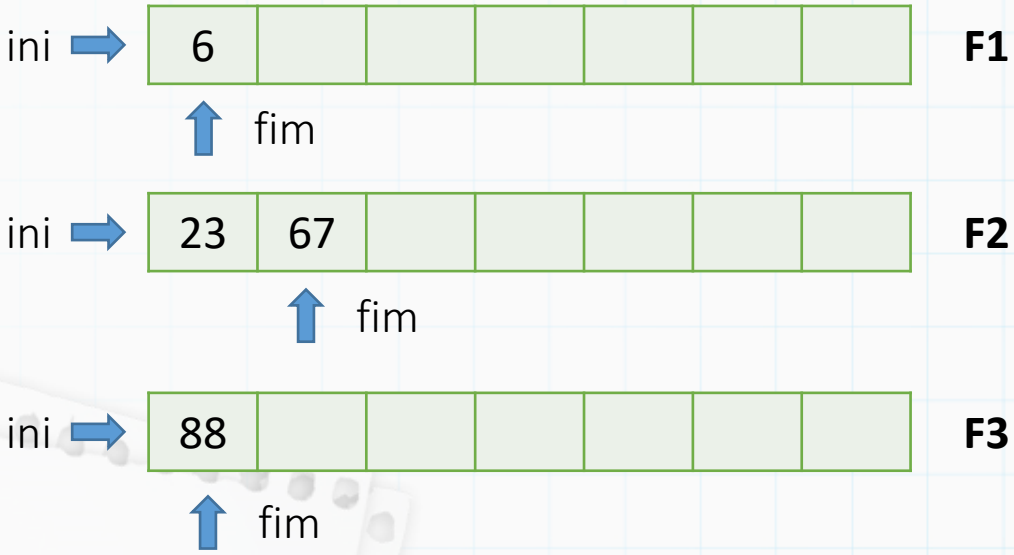


TAD - Pilha

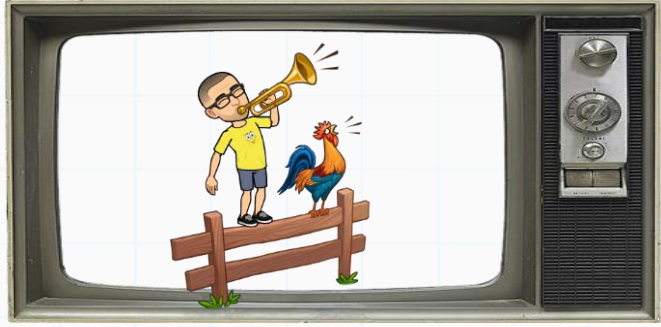


2) Filas de Prioridade em 3 níveis: Vamos considerar uma fila que tem 3 níveis de prioridade (1,2 e 3), quer dizer que para alguém ser atendido na fila (sair da fila), primeiro são considerados as pessoas de menor prioridade (1), depois de 2, e por ultimo 3. Essa fila prioritária em 3 níveis pode ser implementada usando 3 filas de prioridade, veja o exemplo:

Vamos inserir elemento 23 e 67 com prioridade 2
Vamos inserir elemento 6 com prioridade 1
Vamos inserir elemento 88 com prioridade 3



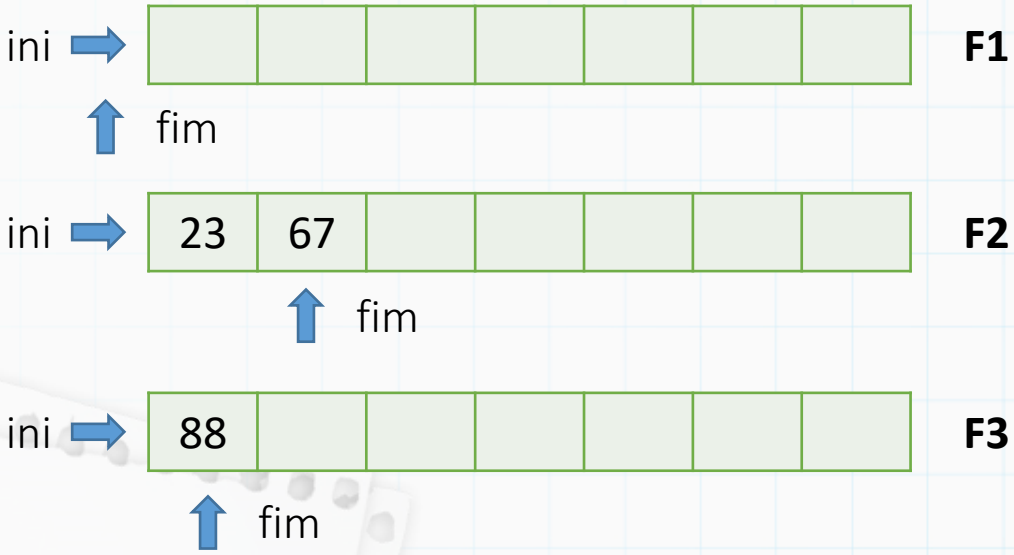
TAD - Pilha



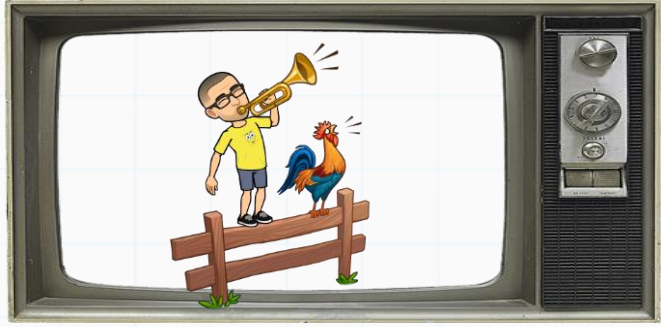
2) Filas de Prioridade em 3 níveis: Vamos considerar uma fila que tem 3 níveis de prioridade (1,2 e 3), quer dizer que para alguém ser atendido na fila (sair da fila), primeiro são considerados as pessoas de menor prioridade (1), depois de 2, e por ultimo 3. Essa fila prioritária em 3 níveis pode ser implementada usando 3 filas de prioridade, veja o exemplo:

Vamos inserir elemento 23 e 67 com prioridade 2
Vamos inserir elemento 6 com prioridade 1
Vamos inserir elemento 88 com prioridade 3

Vamos agora retirar elementos da fila, o primeiro a ser atendido (sair) será o 6,



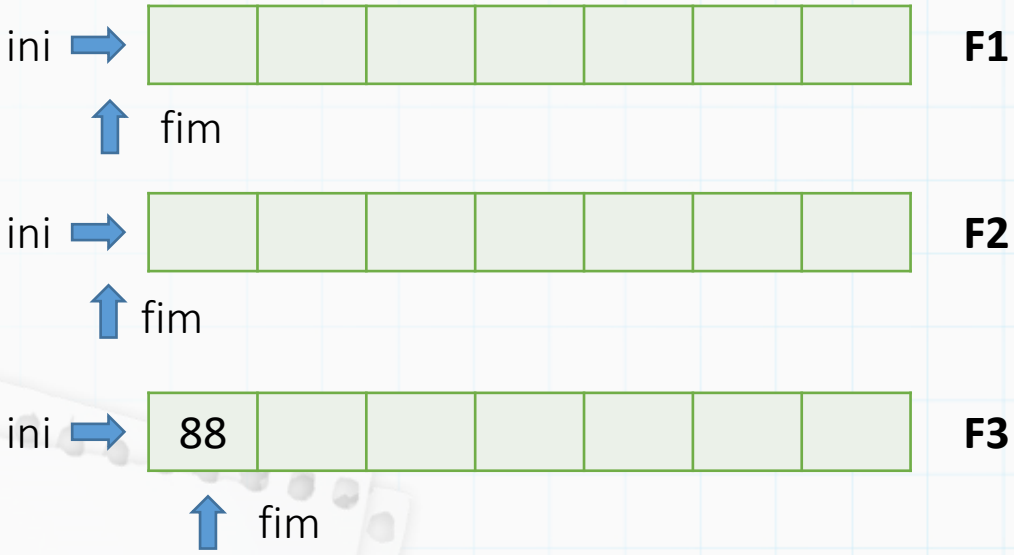
TAD - Pilha



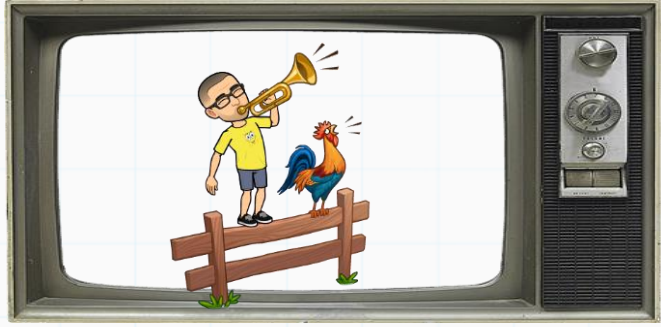
2) Filas de Prioridade em 3 níveis: Vamos considerar uma fila que tem 3 níveis de prioridade (1,2 e 3), quer dizer que para alguém ser atendido na fila (sair da fila), primeiro são considerados as pessoas de menor prioridade (1), depois de 2, e por ultimo 3. Essa fila prioritária em 3 níveis pode ser implementada usando 3 filas de prioridade, veja o exemplo:

Vamos inserir elemento 23 e 67 com prioridade 2
Vamos inserir elemento 6 com prioridade 1
Vamos inserir elemento 88 com prioridade 3

Vamos agora retirar elementos da fila, o primeiro a ser atendido (sair) será o 6, depois o 23 e 67

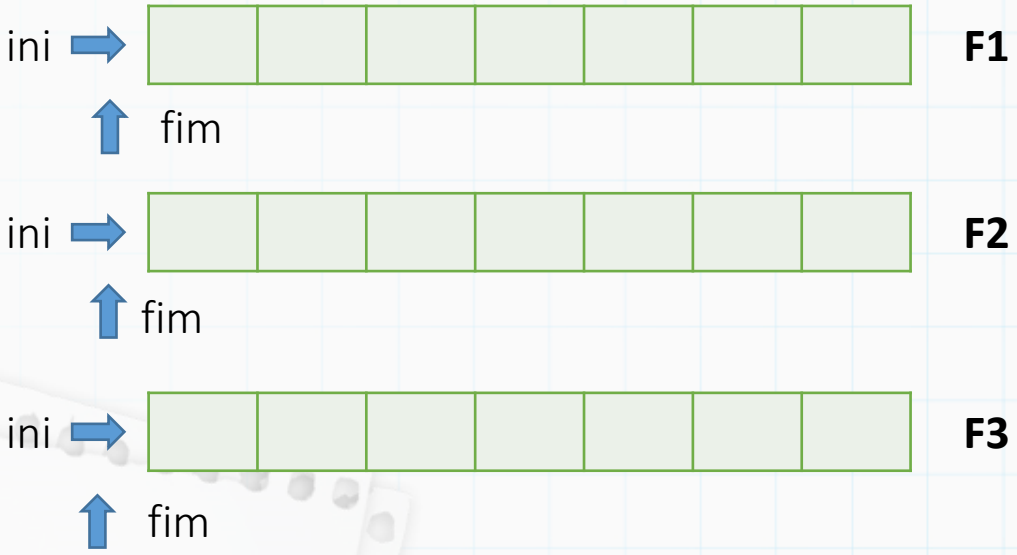


TAD - Pilha



2) Filas de Prioridade em 3 níveis: Vamos considerar uma fila que tem 3 níveis de prioridade (1,2 e 3), quer dizer que para alguém ser atendido na fila (sair da fila), primeiro são considerados as pessoas de menor prioridade (1), depois de 2, e por ultimo 3. Essa fila prioritária em 3 níveis pode ser implementada usando 3 filas de prioridade, veja o exemplo:

Vamos inserir elemento 23 e 67 com prioridade 2
Vamos inserir elemento 6 com prioridade 1
Vamos inserir elemento 88 com prioridade 3



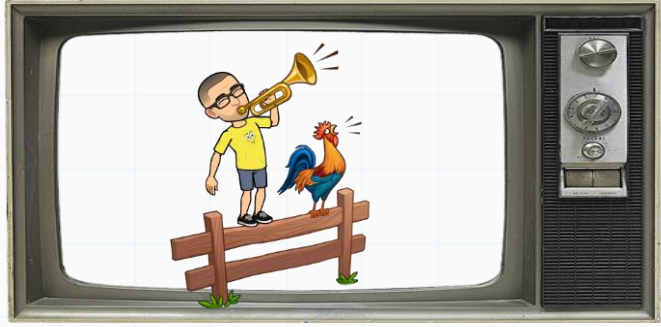
Vamos agora retirar elementos da fila, o primeiro a ser atendido (sair) será o 6, depois o 23 e 67, e por ultimo será o 88.

Implemente as funções:

```
void entra_filas(fila * F1, fila * F2, fila * F3, int el, int prio)
int sai_filas(fila * F1, fila * F2, fila * F3)
void imprime_filas(fila * F1, fila * F2, fila * F3)
```

Veja o exemplo de execução abaixo:

TAD - Pilha



2) Filas de Prioridade em 3 níveis:

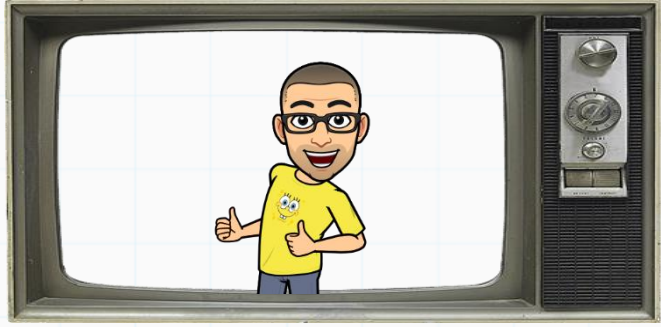
Código da main.c

```
int main()
{
    fila *F1 = aloca_fila();
    fila *F2 = aloca_fila();
    fila *F3 = aloca_fila();

    entra_filas(F1,F2,F3, 43, 2);
    entra_filas(F1,F2,F3, 100, 3);
    entra_filas(F1,F2,F3, 2, 1);
    entra_filas(F1,F2,F3, 6, 1);
    entra_filas(F1,F2,F3, 200, 3);
    entra_filas(F1,F2,F3, 7, 1);
    entra_filas(F1,F2,F3, 65, 2);
    entra_filas(F1,F2,F3, 300, 3);
    entra_filas(F1,F2,F3, 54, 2);
    imprime_filas(F1, F2, F3);

    while (vazia_filas(F1,F2,F3) == 0)
        printf("sai = %d\n", sai_filas(F1,F2,F3));

    F1 = exclui_fila (F1);
    F2 = exclui_fila (F2);
    F3 = exclui_fila (F3);


    return 0;
}
```

Exemplo de execução:

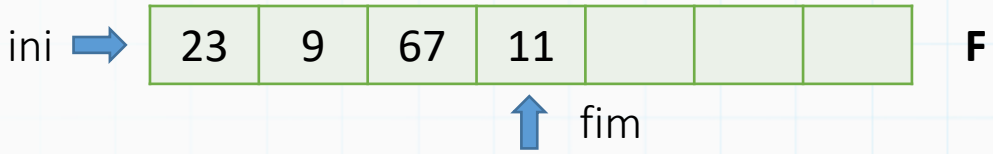
```
|  2 ||  6 ||  7 || 43 || 65 || 54 || 100 || 200 ||
300 |
sai = 2
sai = 6
sai = 7
sai = 43
sai = 65
sai = 54
sai = 100
sai = 200
sai = 300
```

TAD - Pilha



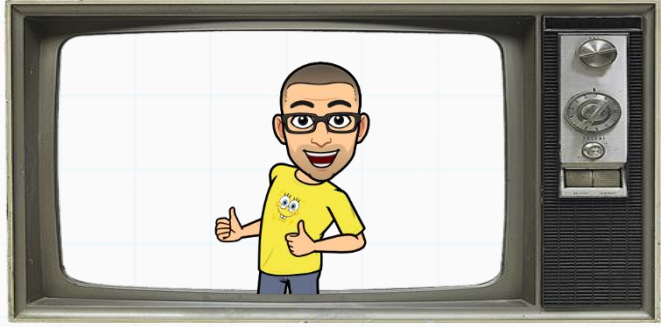
3) Filas de Prioridade por chave: Vamos considerar uma fila de prioridade por chave, o primeiro a ser atendido (sair da fila) é aquele que tenha a menor chave (valor), veja o exemplo:

Vamos inserir elemento 23, 9, 67 e 11



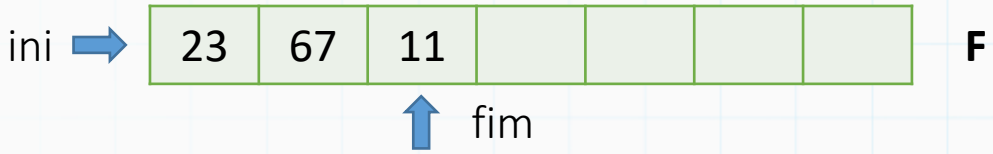
F

TAD - Pilha

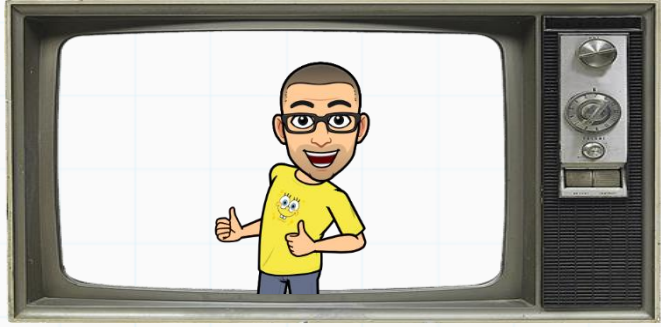


3) Filas de Prioridade por chave: Vamos considerar uma fila de prioridade por chave, o primeiro a ser atendido (sair da fila) é aquele que tenha a menor chave (valor), veja o exemplo:

Vamos inserir elemento 23, 9, 67 e 11
Vamos retirar agora o primeiro, que será o elemento 9 que tem menor prioridade

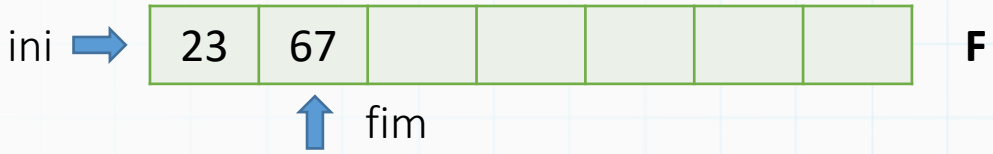


TAD - Pilha

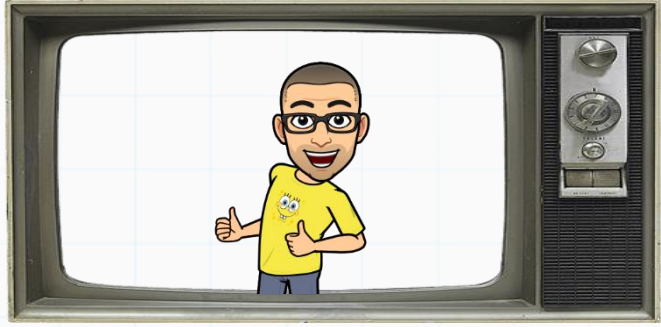


3) Filas de Prioridade por chave: Vamos considerar uma fila de prioridade por chave, o primeiro a ser atendido (sair da fila) é aquele que tenha a menor chave (valor), veja o exemplo:

Vamos inserir elemento 23, 9, 67 e 11
Vamos retirar agora o primeiro, que será o elemento 9 que tem menor prioridade, depois o 11

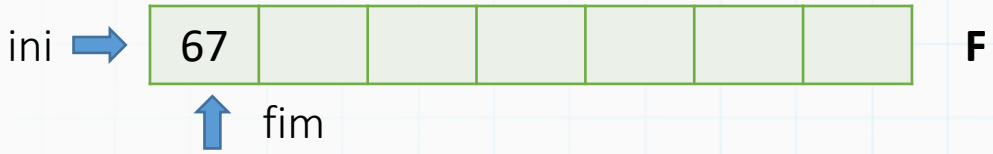


TAD - Pilha

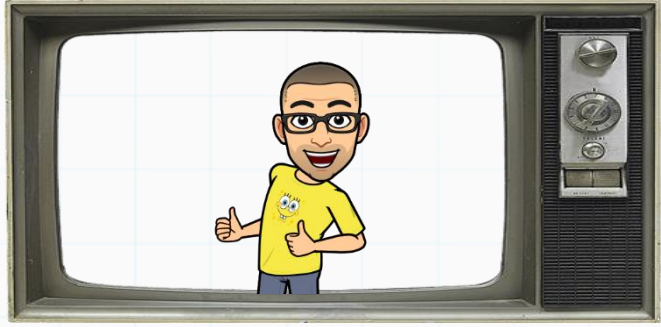


3) Filas de Prioridade por chave: Vamos considerar uma fila de prioridade por chave, o primeiro a ser atendido (sair da fila) é aquele que tenha a menor chave (valor), veja o exemplo:

Vamos inserir elemento 23, 9, 67 e 11
Vamos retirar agora o primeiro, que será o elemento 9 que tem menor prioridade, depois o 11, depois o 23

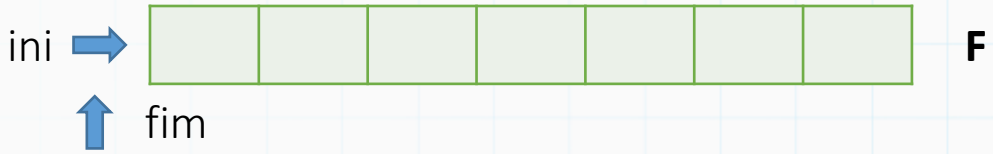


TAD - Pilha



3) Filas de Prioridade por chave: Vamos considerar uma fila de prioridade por chave, o primeiro a ser atendido (sair da fila) é aquele que tenha a menor chave (valor), veja o exemplo:

Vamos inserir elemento 23, 9, 67 e 11
Vamos retirar agora o primeiro, que será o elemento 9 que tem menor prioridade, depois o 11, depois o 23, e finalmente o 67



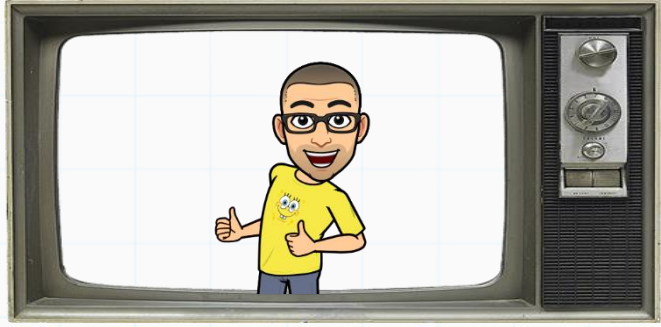
F

Implemente a função:

int sai_fila_prio(fila * F)

Veja o exemplo de execução abaixo:

TAD - Pilha



3) Filas de Prioridade por chave:

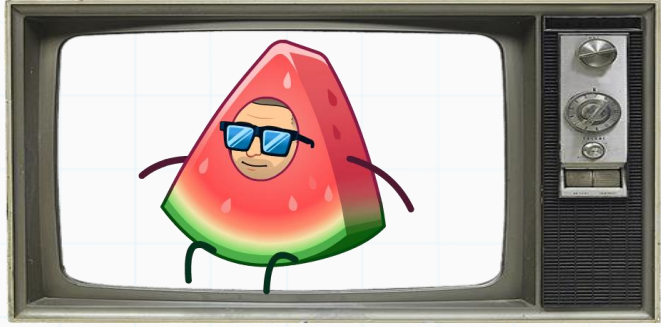
Código da main.c

```
int main()
{
    fila *F = aloca_fila();

    entra_fila(F, 43);
    entra_fila(F, 100);
    entra_fila(F, 2);
    entra_fila(F, 6);
    entra_fila(F, 200);
    imprime_fila(F);

    printf("sai = %d\n",sai_fila_prio(F));
    imprime_fila(F);
    printf("sai = %d\n",sai_fila_prio(F));
    imprime_fila(F);
    printf("sai = %d\n",sai_fila_prio(F));
    imprime_fila(F);
    printf("sai = %d\n",sai_fila_prio(F));
    imprime_fila(F);
    printf("sai = %d\n",sai_fila_prio(F));
    imprime_fila(F);
    printf("sai = %d\n",sai_fila_prio(F));
    imprime_fila(F);

    F = exclui_fila (F);
    return 0;
}
```

Exemplo de execução:

```
| 43 || 100 ||  2 ||  6 || 200 |
sai = 2
| 43 || 100 ||  6 || 200 |
sai = 6
| 43 || 100 || 200 |
sai = 43
| 100 || 200 |
sai = 100
| 200 |
sai = 200

fila vazia
sai = -1
```

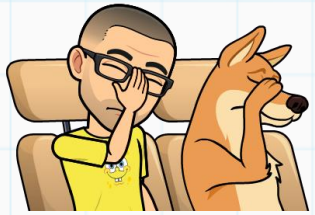
TAD - Pilha



4) Ordena Fila: Queremos escrever uma função em que vamos ordenar os elementos de uma fila de forma crescente, porem só podemos interagir com a fila usando as funções:

```
fila * aloca_fila(void)
fila * exclui_fila (fila* F)
int vazia_fila(fila * F)
int primeiro_fila(fila *F)
void entra_fila(fila *F, int el)
int sai_fila(fila *F)
```

Observe que a única forma de olhar o valor de um elemento da fila são com as funções primeiro_fila e sai_fila.

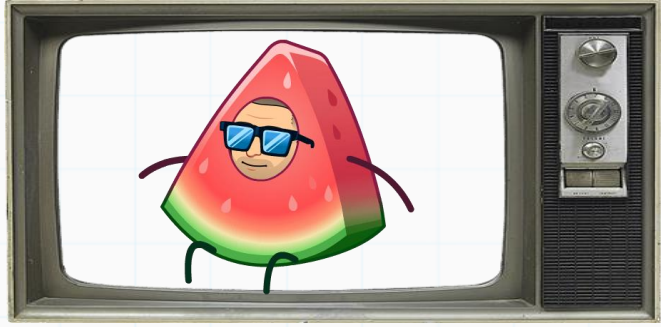


Além disso, a única memória auxiliar que você pode usar será uma fila auxiliar criada dentro da função de ordenação.

DICA: Você pode checar todos os elementos de uma fila, retirando e re-inserindo.

Vamos então escrever a função de ordenação. Veja o exemplo de execução a seguir:

TAD - Pilha



4) Ordena Fila:

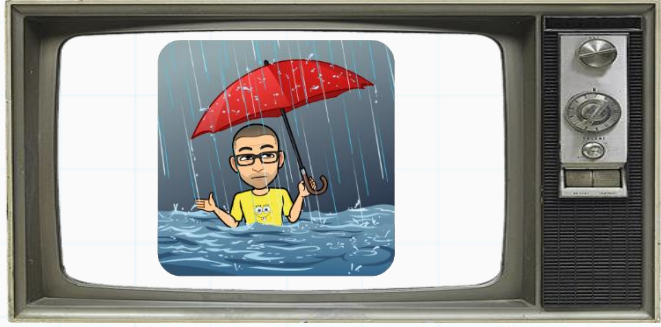
Código da main.c

```
int main()
{
    fila *F = aloca_fila();

    entra_fila(F, 43);
    entra_fila(F, 100);
    entra_fila(F, 2);
    entra_fila(F, 6);
    entra_fila(F, 200);
    entra_fila(F, 28);
    entra_fila(F, 8);
    imprime_fila(F);

    ordena_fila(F);
    imprime_fila(F);

    F = exclui_fila (F);
    return 0;
}}
```

Exemplo de execução:

```
| 43 || 100 ||  2 ||  6 || 200 || 28 ||  8 |
|  2 ||  6 ||  8 || 28 || 43 || 100 || 200 |
```

TAD - Pilha



5) Pinta tela: Dada uma matriz de inteiros que representa as cores dos pixels de uma tela de computador

1	1	2	2	4	4
1	1	1	2	5	3
4	4	1	3	3	3
5	5	1	3	1	1
7	3	3	3	4	4

Queremos indicar uma posição da tela (coordenada linha/coluna) em que uma nova cor será aplicada (novo valor), porem ao aplicarmos essa cor, ela irá também pintar de forma recursiva todos as cores ao seu redor (cima, baixo, esquerda e direita) que tinham a mesma cor original do ponto. Veja, se pintarmos a posição (2,3) com a cor 8 (antes era 3), teremos:

1	1	2	2	4	4
1	1	1	2	5	3
4	4	1	8	3	3
5	5	1	3	1	1
7	3	3	3	4	4

1	1	2	2	4	4
1	1	1	2	5	3
4	4	1	8	8	3
5	5	1	8	1	1
7	3	3	3	4	4

1	1	2	2	4	4
1	1	1	2	5	3
4	4	1	8	8	8
5	5	1	8	1	1
7	3	3	8	4	4

...

1	1	2	2	4	4
1	1	1	2	5	8
4	4	1	8	8	8
5	5	1	8	1	1
7	8	8	8	4	4

Este processo pode ser realizado utilizando uma fila e enfileirando vizinhos dos nos pintados. Veja o exemplo de execução:

TAD - Pilha

5) Pinta tela

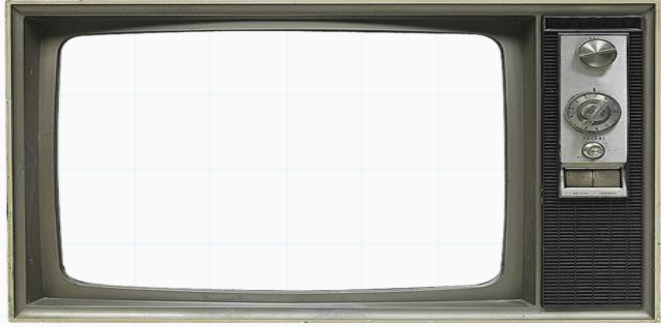
Código da main.c

```
int main()
{
    int m = 8;
    int n = 8;

    int tela[][8] = {   { 1, 1, 1, 1, 1, 1, 1, 1 },
                        { 1, 1, 1, 1, 1, 1, 0, 0 },
                        { 1, 0, 0, 1, 1, 0, 1, 1 },
                        { 1, 2, 2, 2, 2, 0, 1, 0 },
                        { 1, 1, 1, 2, 2, 0, 1, 0 },
                        { 4, 1, 1, 2, 2, 2, 2, 0 },
                        { 4, 1, 4, 1, 1, 2, 1, 1 },
                        { 4, 4, 4, 4, 4, 2, 2, 1 } };
    imprime_tela(tela, n, m);

    int x=4, y=4;
    int NovaCor = 3;
    pinta_tela(tela, n, m, x, y, NovaCor);
    imprime_tela(tela, n, m);

    x=7, y=1;
    NovaCor = 7;
    pinta_tela(tela, n, m, x, y, NovaCor);
    imprime_tela(tela, n, m);

    return 0;
}
```

Exemplo de execução:

```
1 1 1 1 1 1 0 0
1 0 0 1 1 0 1 1
1 2 2 2 2 0 1 0
1 1 1 2 2 0 1 0
4 1 1 2 2 2 2 0
4 1 4 1 1 2 1 1
4 4 4 4 4 2 2 1

1 1 1 1 1 1 1 1
1 1 1 1 1 1 0 0
1 0 0 1 1 0 1 1
1 3 3 3 3 0 1 0
1 1 1 3 3 0 1 0
4 1 1 3 3 3 3 0
4 1 4 1 1 3 1 1
4 4 4 4 4 3 3 1

1 1 1 1 1 1 1 1
1 1 1 1 1 1 0 0
1 0 0 1 1 0 1 1
1 3 3 3 3 0 1 0
1 1 1 3 3 0 1 0
7 1 1 3 3 3 3 0
7 1 7 1 1 3 1 1
7 7 7 7 7 3 3 1
```

Até a próxima



Slides baseados no curso de Aline Nascimento