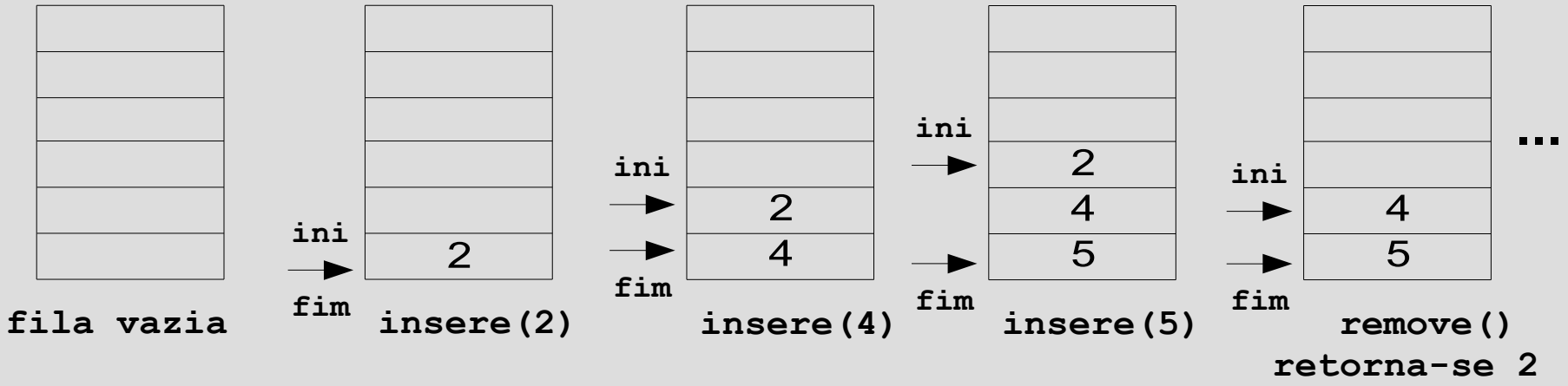
Estruturas de Dados

Fila

Fila

- Uma das estruturas de dados mais utilizadas é a fila
- A inserção/remoção de elementos utiliza a estratégia:
  - O primeiro que entra é o primeiro que sai (*First in, first out)*
- As operações básicas são:
  - Inserir no Fim da Fila
  - Remover do Início da Fila



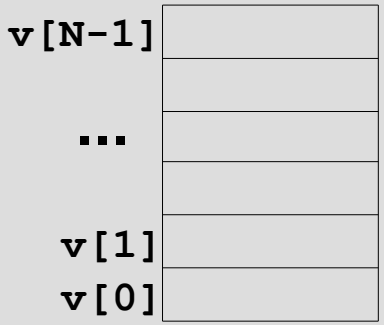
Tipo Abstrato de Dado Fila

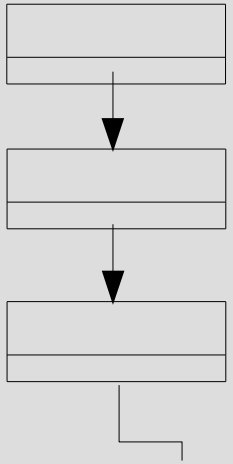
Podemos criar um TAD Fila de Inteiros. Para tanto, devemos criar o arquivo fila.h com o nome do tipo e os protótipos.

```
typedef struct fila Fila;

/*Função que cria uma Fila.*/
Fila* fila_cria(void);
/*Testa se uma Fila é vazia.*/
int fila_vazia(Fila *f);
/*Função que adiciona um elemento em uma Fila.*/
void fila_insere(Fila *f, int info);
/*Função que remove um elemento de uma Fila.*/
int fila_remove(Fila *f);
/*Função que imprime os elementos de uma Fila.*/
void fila_imprime(Fila *f);
/*Libera o espaço alocado para uma Fila.*/
void fila_libera(Fila *f);
```

Implementação do TAD Fila

- Podemos implementar a interface fila de duas maneiras:
  - Vetor, quando o número máximo de elementos é conhecido
  - Lista, quando não se sabe o número máximo de elementos



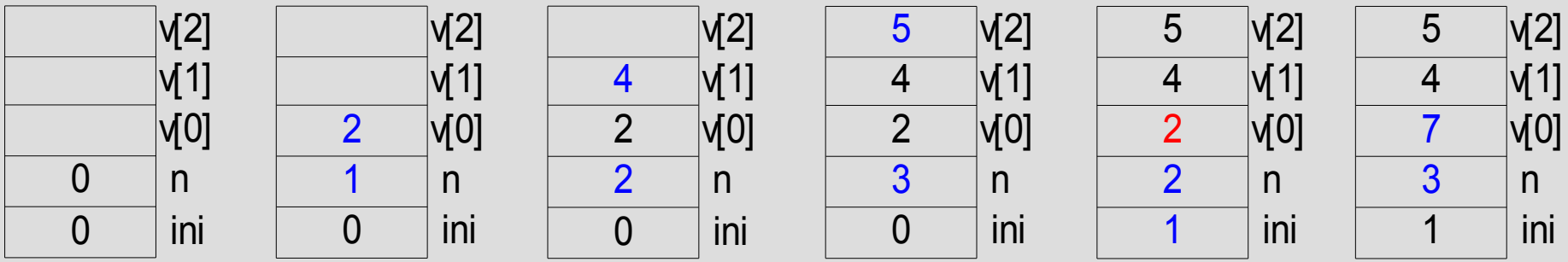


Implementação de Fila com Vetor

- A estrutura que representa o tipo fila deve ser composto pelo vetor, pelo elemento inicial e pelo número de elementos armazenados

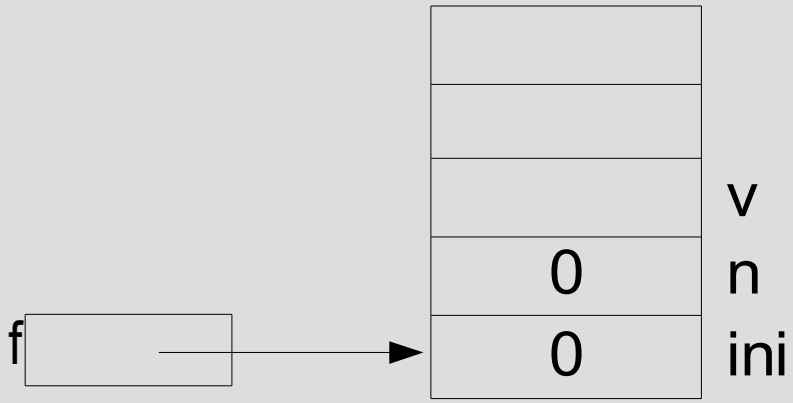
```
#define N 3
typedef struct fila{
   int n;
   int ini;
   int v[N];
}Fila;
```

```
Fila f; f.n=0; f.ini=0;
f.v[0] = 2; f.n++;
f.v[1] = 4; f.n++;
f.v[2] = 5; f.n++;
f.n--; f.ini++;
f.v[0] = 7; f.n++;
```



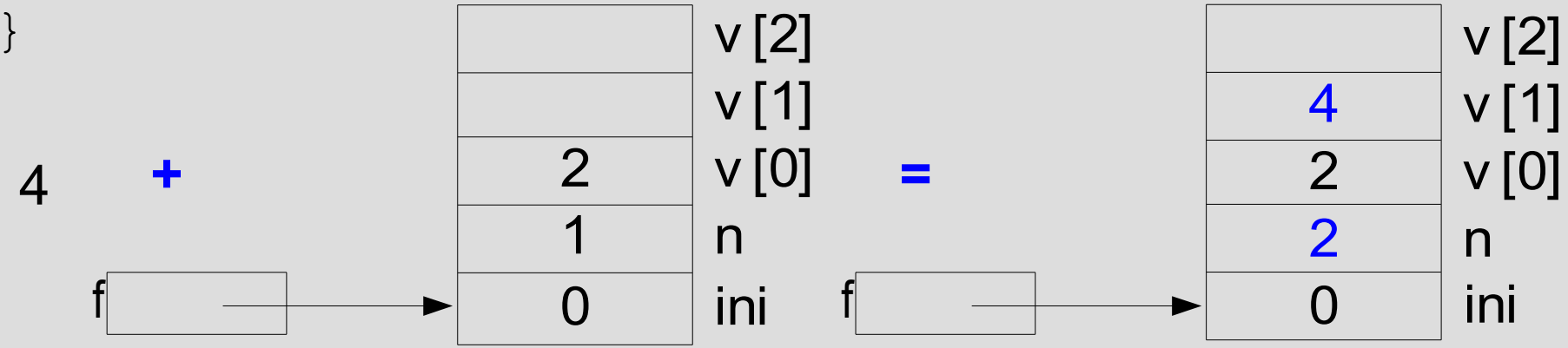
Implementação de Fila com Vetor – Função Cria Fila

```
Fila* fila_cria(void){
    Fila *f = (Fila*)malloc(sizeof(Fila));
    if(f==NULL){
       printf("Memoria insuficiente!!!\n");
       exit(1);
    }
    f->n = 0;
    f->ini = 0;
    return f;
}
```



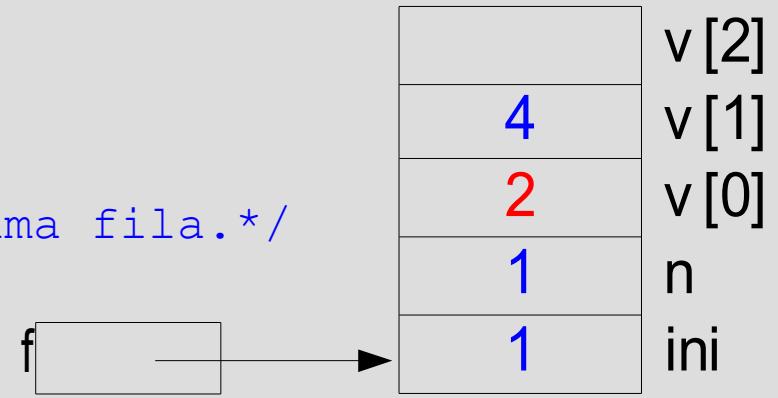
Implementação de Fila com Vetor – Função Insere

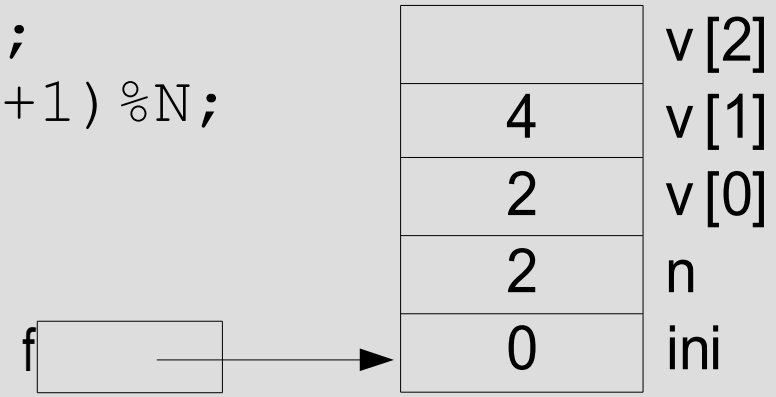
```
void fila_insere(Fila *f, int info){
    int fim;
    if(f->n==N){
        printf("Capacidade da Fila Estorou!!!\n");
        exit(1);
    }
    fim = (f->ini + f->n) % N;
    f->v[fim]= info;
    f->n++;
}
```



Implementação de Fila com Vetor – Função Vazia e Remove

```
/*Testa se uma fila é vazia.*/
int fila_vazia(Fila *f){
    return f->n==0;
}
/*Função que remove um elemento de uma fila.*/
int fila_remove(Fila *f){
    int a;
    if(fila_vazia(f)){
        printf("Fila Vazia!!!\n");
        exit(1);
    }
    a = f->v[f->ini];
    f->ini = (f->ini+1)%N;
    f->n--;
    return a;
}
```

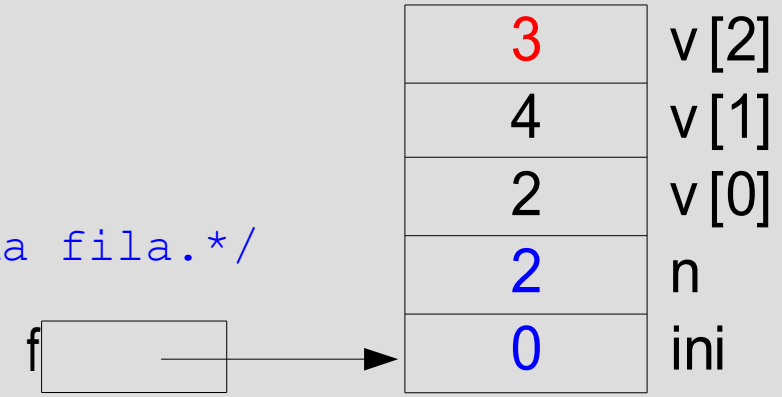


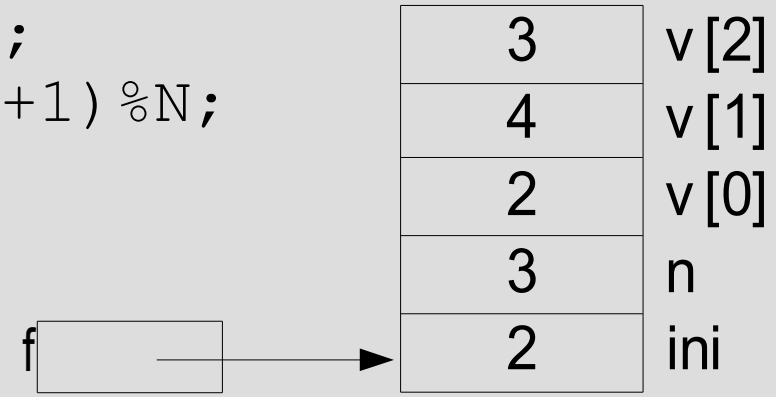


= 2

Implementação de Fila com Vetor – Função Vazia e Remove

```
/*Testa se uma fila é vazia.*/
int fila_vazia(Fila *f){
    return f->n==0;
}
/*Função que remove um elemento de uma fila.*/
int fila_remove(Fila *f){
    int a;
    if(fila_vazia(f)){
        printf("Fila Vazia!!!\n");
        exit(1);
    }
    a = f->v[f->ini];
    f->ini = (f->ini+1)%N;
    f->n--;
    return a;
}
```

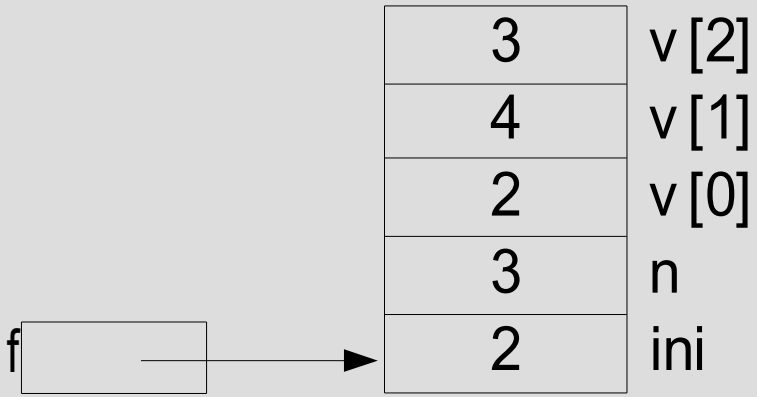


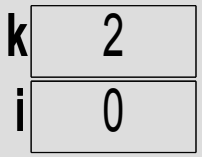


= 3

Implementação de Fila com Vetor
Função Imprime e Libera

```
void fila_imprime(Fila *f){
   int i;   int k;
   for(i = 0; i<f->n;i++){
      k = (f->ini+i) % N;
      printf("%d\n",f->v[k]);
   }

}
```





=

3
2
4

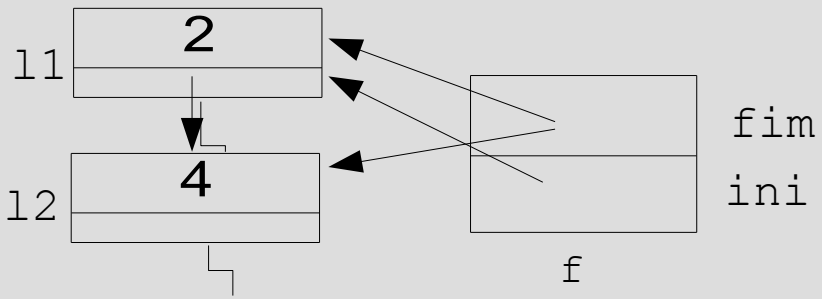
```
void fila_libera(Fila *f){
   free(f);
}
```

Implementação de Fila com Lista Encadeada

- A estrutura que representa o tipo fila é composta pelo início e pelo fim de uma lista.

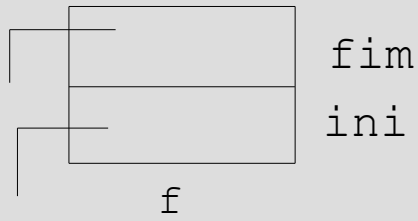
```
typedef struct lista Lista;
typedef struct fila Fila;

struct lista{
   int info;
   Lista *prox;
};

struct fila{
   Lista *ini;
   Lista *fim;
};
```

```
Fila f;

Lista l1; l1.info=2; l1.prox=NULL;
f.ini=&l1; f.fim=&l1;

Lista l2; l2.info=4; l2.prox=NULL;
f.fim->prox=&l2;f.fim=&l2;
```



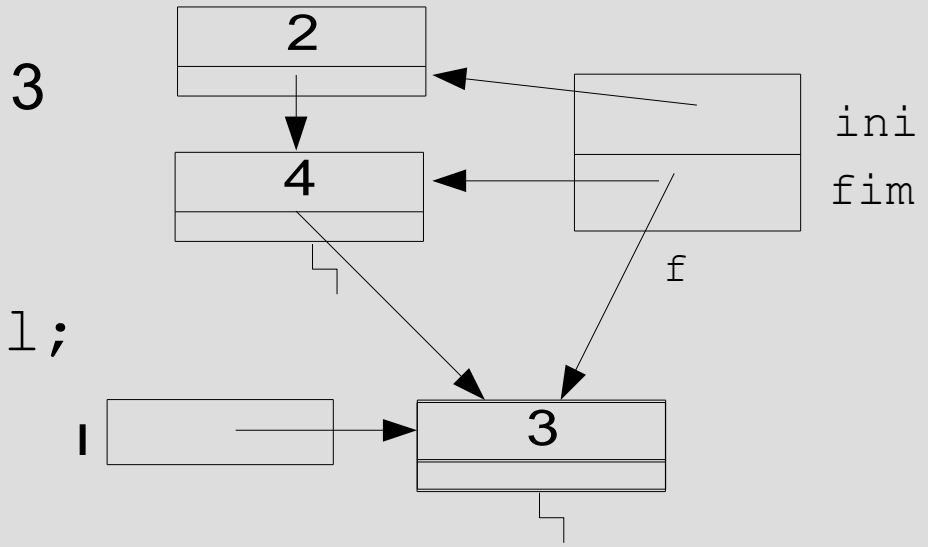
Implementação de Fila com Lista – Função Cria

```
Fila* fila_cria(void){
    Fila *f = (Fila*)malloc(sizeof(Fila));
    if(f==NULL){
        printf("Memoria insuficiente!!!\n");
        exit(1);
    }
    f->ini = NULL;
    f->fim = NULL;
    return f;
}
```



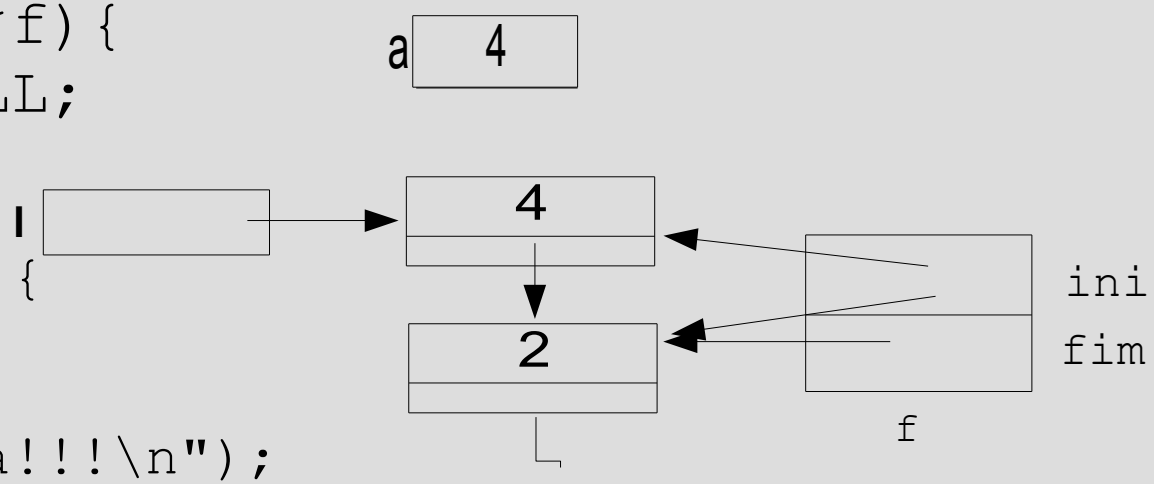
Implementação de Fila com Lista – Função Insere

```
void fila_insere(Fila *f, int info){
   Lista *l = (Lista*)malloc(sizeof(Lista));
   if(l==NULL){
      printf("Memoria insuficiente!!!\n");
      exit(1);
   }
   l->info = info;
   l->prox = NULL;
   if(!fila_vazia(f))
         f->fim->prox = l;
   else
         f->ini = l;
   f->fim = l;
}
```



Implementação de Fila com Lista – Função Vazia e Remove

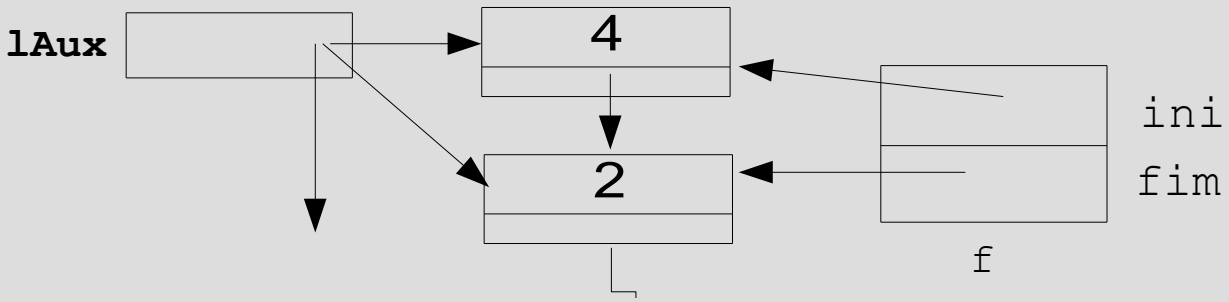
```
int fila_vazia(Fila *f){
    return f->ini==NULL;
}

int fila_remove(Fila *f){
    Lista *l;  int a;
    if(fila_vazia(f)){
        printf("Fila Vazia!!!\n");
        exit(1);
    }
    a = f->ini->info;
    l = f->ini;
    f->ini = f->ini->prox;
    free(l);
    if(fila_vazia(f))
        f->fim = NULL;
    return a;
}
```

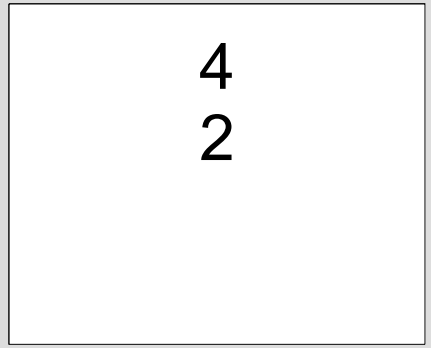


= 4

Implementação de Fila com Lista
Função Imprime

```
/*Função que imprime os elementos de uma fila.*/
void fila_imprime(Fila *f){
    Lista *lAux = f->ini;
    while(lAux!=NULL){
        printf("%f\n",lAux->info);
        lAux = lAux->prox;
    }
}
```





Implementação de Fila com Lista
Função Libera

```
/*Libera o espaço alocado para uma fila.*/
void fila_libera(Fila *f){
    Lista* l = f->ini;
    Lista* lAux;
    while(l!=NULL){
        lAux = l->prox;
        free(l);
        l = lAux;
    }
    free(f);
}
```