FAMEPI

Faculdade do Amazonas de Ensino, Pesquisa e Inovação

FUNDAÇÃO MATIAS MACHLINE

Algoritmos e Estrutura de Dados 2

Prof Pedro Corrêa

Objetivo da aula

Utilizar pesquisas em Estruturas de Dados do tipo Lista implementada através de vetor

Sumário

Sumário

Algoritmos e Estrutura de Dados 2

- Necessidade de recuperar informação a partir de uma grande massa de informação previamente armazenada.
- A informação é dividida em registros.
- Cada registro possui uma chave para ser usada na pesquisa.
- **Objetivo da pesquisa**: Encontrar uma ou mais ocorrências de registros com chaves iguais à chave de pesquisa.
- **Resultado: pesquisa com sucesso / sem sucesso.**

FAMEPI
Faculdade do Amazonas de Ensino, Pesquisa e Inovação
FUNDAÇÃO MATIAS MACHLINE

Algoritmos e Estrutura de Dados 2

Algoritmos de Pesquisa

- É importante considerar os algoritmos de pesquisa como tipos abstratos de dados (TADs), de tal forma que haja uma independência de implementação para as operações.

Algoritmos e Estrutura de Dados 2

Dicionário

- Dicionário é um TAD com as operações:
  1. Todas as operações previstas para uma lista implementada com vetor
     - **Pesquisa**

- Analogia com um dicionário da língua portuguesa:
  - Chaves de pesquisa ⇔ palavras a pesquisar
  - Registros ⇔ entradas associadas como *pronúncia, definição, sinônimos, outras informações

FAMEPI
Faculdade do Amazonas de Ensino, Pesquisa e Inovação
FUNDAÇÃO MATIAS MACHLINE

Sumário

Sumário

Algoritmos e Estrutura de Dados 2

Pesquisa sequencial

- Método de pesquisa mais simples: a partir do primeiro registro, pesquise sequencialmente até encontrar a chave procurada;
  - Então mostre e **pare** (se não houver outra chave igual).
  - Então mostre e **continue** (se houver outra chave igual).
- Armazenamento do conjunto de registros é feito por meio de uma lista implementada através de um vetor (array).

Algoritmos e Estrutura de Dados 2

- **Definindo as estruturas**

tipo_item

Cod
Nome
Quant

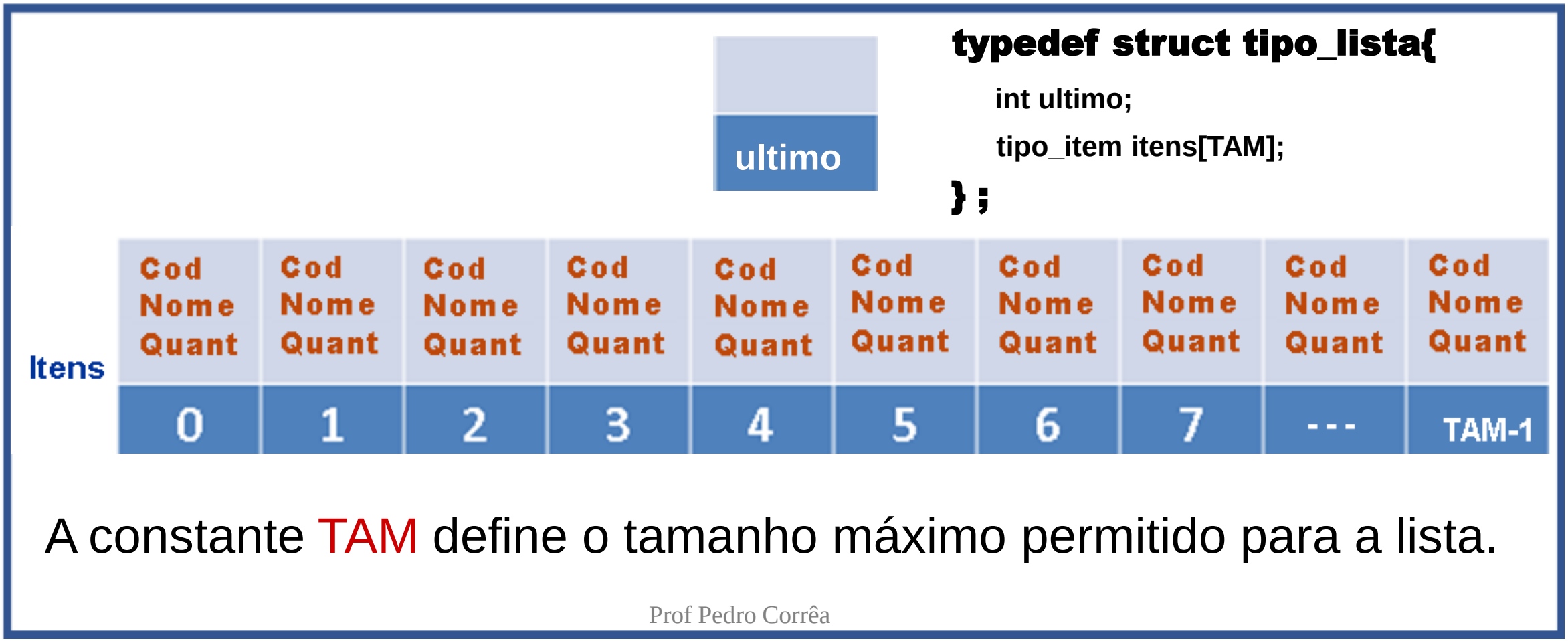
```
typedef struct tipo_item{
   int cod; // será único
   char nome[30];
   int quant;
} ;
```


FAMEPI
Faculdade do Amazonas de Ensino, Pesquisa e Inovação
FUNDAÇÃO MATIAS MACHLINE

Algoritmos e Estrutura de Dados 2

tipo_lista

- Definindo as estruturas



A constante TAM define o tamanho máximo permitido para a lista.

Algoritmos e Estrutura de Dados 2

Pesquisa sequencial

- A busca (find):
  - Pesquisa mostra os dados do registro que contém a chave x;
  - Caso não esteja presente, informa a ausência.

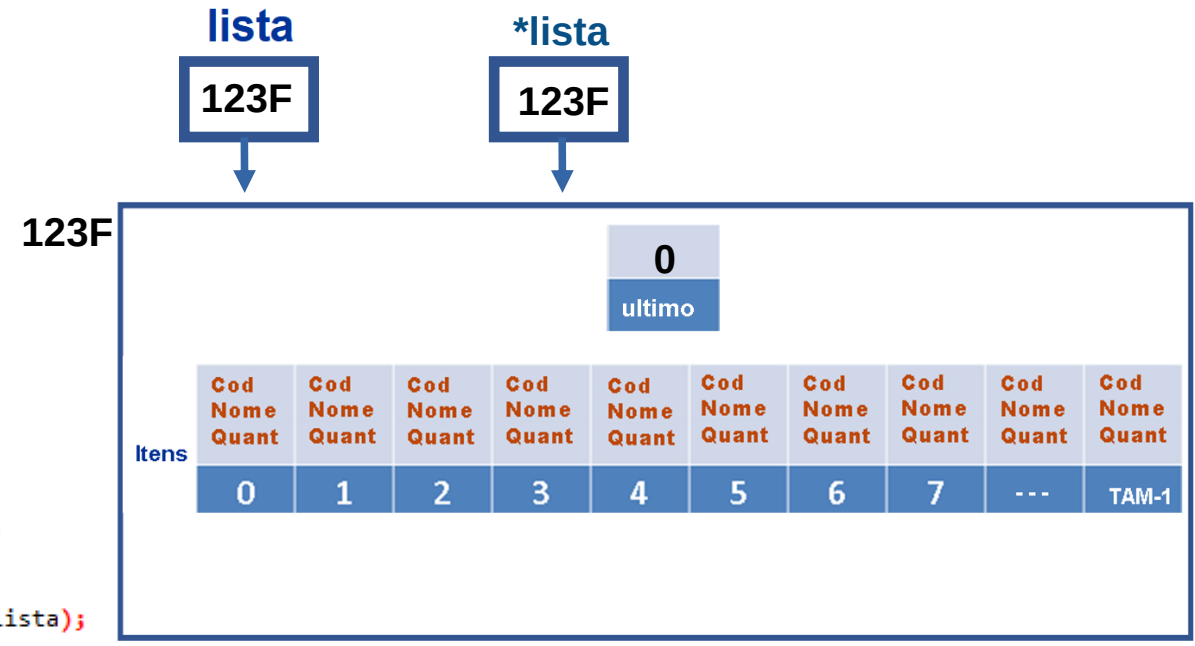
Algoritmos e Estrutura de Dados 2

Pesquisa sequencial

Implementação

```c
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>// para system("cls")
#include <conio.h>// para getch()
#include <string.h>// para strcmp()

#define TAM 5 // constante tamanho vetor

typedef struct tipo_item{
    int cod; // será único
    char nome[30];
    int quant;
};

typedef struct tipo_lista{
    tipo_item itens[TAM];
    int total;
};

void flvazia(tipo_lista *lista){
    lista -> ultimo = 0;
}
```

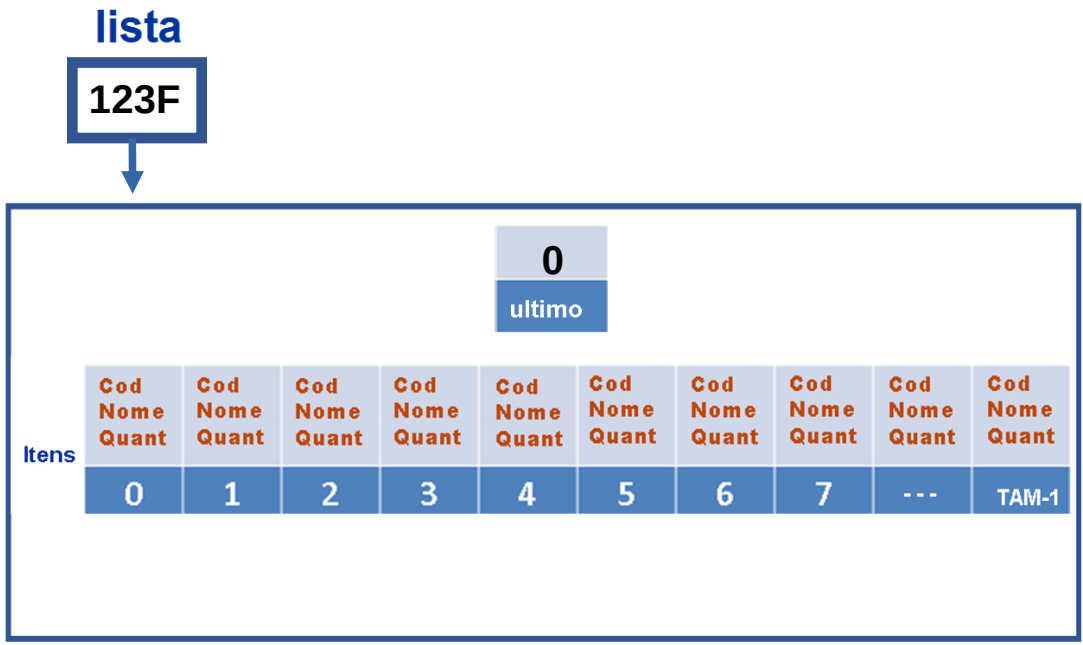
```c
int main( ) {
    setlocale(LC_ALL, "");

    int op;

    tipo lista lista;

    flvazia(&lista);

    do{
        system("cls");

        op = menu();

        switch(op){

            case 0: return(0);
                    break;

            case 1: inserir(&lista);
                    break;

            case 2: mostrar(&lista);
                    break;

            case 3: pesquisar(&lista);
                    break;

            case 4: remover(&lista);
                    break;

            case 5: inserirPos(&lista);
                    break;

            default:
                printf("\n\n OPÇÃO INVÁLIDA !!!");
                getch();
        }

    }while(1);

    return(0);
}
```



FAMEPI
Faculdade do Amazonas de Ensino, Pesquisa e Inovação
FUNDAÇÃO MATIAS MACHLINE

```c
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>// para system("cls")
#include <conio.h>// para getch()
#include <string.h>// para strcmp()

#define TAM 5 // constante tamanho vetor

typedef struct tipo_item{
    int cod; // será único
    char nome[30];
    int quant;
};

typedef struct tipo_lista{
    tipo_item itens[TAM];
    int total;
};


void flvazia(tipo_lista *lista){
    lista -> ultimo = 0;
}

int menu(){
    int op;

    printf("\n\n ********* MENU *********");

    printf("\n [1] - INSERIR");

    printf("\n [2] - MOSTRAR");

    printf("\n [3] - PESQUISAR");

    printf("\n [4] - REMOVER");

    printf("\n [5] - INSERIR EM UMA POSIÇÃO");

    printf("\n [0] - SAIR");

    printf("\n\n Digite sua opção: ");
    scanf("%d", &op);

    return(op);
}
```

```c
int main( ) {
    setlocale(LC_ALL, "");

    int op;

    tipo_lista lista;

    flvazia(&lista);

    do{
        system("cls");

        op = menu();

        switch(op){

            case 0: return(0);
                    break;

            case 1: inserir(&lista);
                    break;

            case 2: mostrar(&lista);
                    break;

            case 3: pesquisar(&lista);
                    break;

            case 4: remover(&lista);
                    break;

            case 5: inserirPos(&lista);
                    break;

            default:
                printf("\n\n OPÇÃO INVÁLIDA !!!");
                getch();
        }

    }while(1);

    return(0);
}
```




```
********** MENU **********
[1] - INSERIR
[2] - MOSTRAR
[3] - PESQUISAR
[4] - REMOVER
[5] - INSERIR EM UMA POSIÇÃO
[0] - SAIR

Digite sua opção: _
```

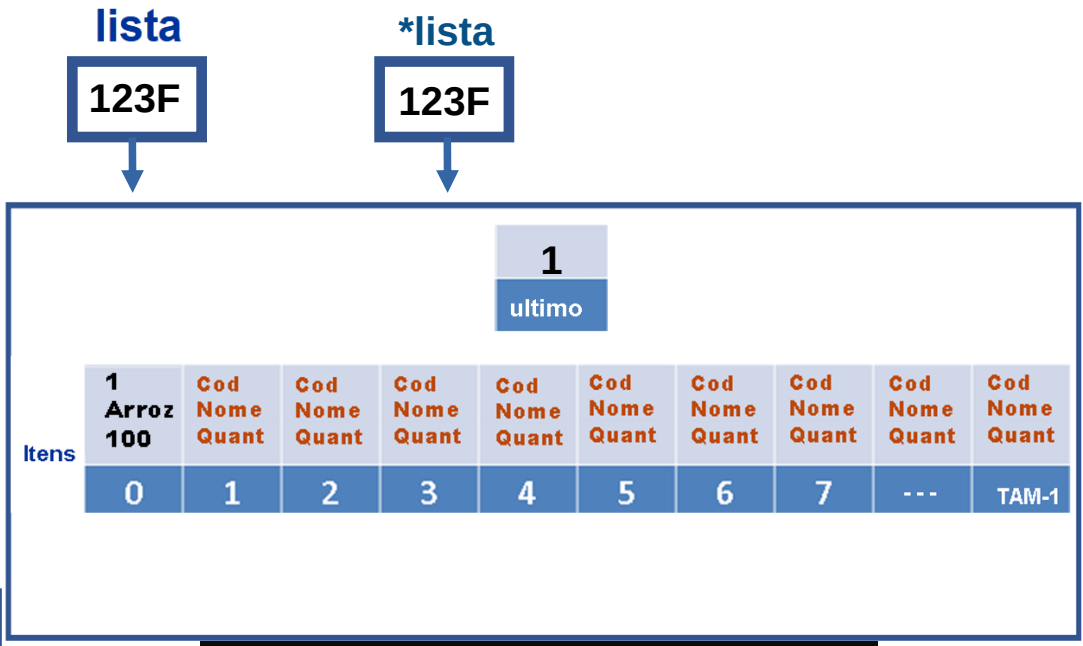

FAMEPI
Faculdade do Amazonas de Ensino, Pesquisa e Inovação
FUNDAÇÃO MATIAS MACHLINE

```c
tipo_item pegarItem(){

    static int cod = 1;

    tipo_item item;

    item.cod = cod;

    cod++;

    fflush(stdin);

    printf("\n\n Digite o nome do item: ");

    gets(item.nome);

    printf("\n\n Digite a quantidade do item: ");

    scanf("%d", &item.quant);

    return(item);
}

 int cheia(tipo_lista *lista){
     return(lista -> ultimo == TAM);
 }


void inserir(tipo_lista *lista){

    if(cheia(lista)){
        printf("\n\n Impossível inserir! Motivo: LISTA CHEIA!!!");
        getch();
        return;
    }

    lista -> itens[lista->ultimo] = pegarItem();

    lista -> ultimo++;

    printf("\n\n ITEM INSERIDO COM SUCESSO!!!");

    getch();
}
```

2

1

```c
int main( ) {
    setlocale(LC_ALL, "");

    int op;

    tipo_lista lista;

    flvazia(&lista);

    do{
        system("cls");

        op = menu();

        switch(op){

            case 0: return(0);
                    break;

            case 1: inserir(&lista);
                    break;

            case 2: mostrar(&lista);
                    break;

            case 3: pesquisar(&lista);
                    break;

            case 4: remover(&lista);
                    break;

            case 5: inserirPos(&lista);
                    break;

            default:
                printf("\n\n OPÇÃO INVÁLIDA !!!");
                getch();
        }

    }while(1);

    return(0);
}
```




```
********** MENU **********
[1] - INSERIR
[2] - MOSTRAR
[3] - PESQUISAR
[4] - REMOVER
[5] - INSERIR EM UMA POSIÇÃO
[0] - SAIR

Digite sua opção: 1

Digite o nome do item: Arroz


Digite a quantidade do item: 100


ITEM INSERIDO COM SUCESSO!!!
```

```c
int vazia(tipo_lista *lista){
    return(lista -> ultimo == 0);
}
```

```c
void mostrar(tipo lista *lista){

    if(vazia(lista)){
        printf("\n\n Impossível mortrar a lista. Motivo: LISTA VAZIA !!!");
        getch();
        return;
    }

    printf("\n\n === Total de itens na lista: %d ===", lista->ultimo);

    for(int i = 0; i < lista->ultimo; i++){

        printf("\n\n === %dº ITEM ===", i + 1);

        printf("\n Cod: %d", lista->itens[i].cod);

        printf("\n Nome: %s", lista->itens[i].nome);

        printf("\n Quantidade: %d", lista->itens[i].quant);
    }

    getch();
}
```

```c
int main( ) {
    setlocale(LC_ALL, "");

    int op;

    tipo_lista lista;

    flvazia(&lista);

    do{
        system("cls");

        op = menu();

        switch(op){

            case 0: return(0);
                    break;

            case 1: inserir(&lista);
                    break;

            case 2: mostrar(&lista);
                    break;

            case 3: pesquisar(&lista);
                    break;

            case 4: remover(&lista);
                    break;

            case 5: inserirPos(&lista);
                    break;

            default:
                printf("\n\n OPÇÃO INVÁLIDA !!!");
                getch();
        }

    }while(1);

    return(0);
}
```


```
=== Total de itens na lista: 4 ===

=== 1º ITEM ===
Cod: 1
Nome: Arroz
Quantidade: 100

=== 2º ITEM ===
Cod: 2
Nome: Farinha
Quantidade: 200

=== 3º ITEM ===
Cod: 3
Nome: FeijÆo
Quantidade: 300

=== 4º ITEM ===
Cod: 4
Nome: Carne
Quantidade: 400
```

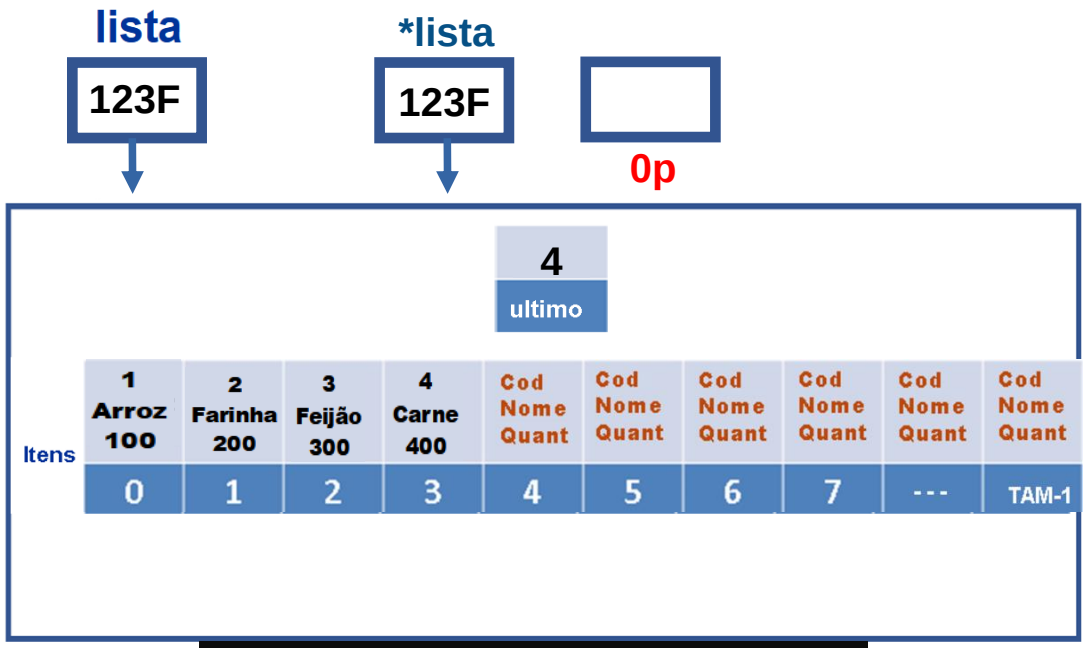
```
********** MENU **********
[1] - INSERIR
[2] - MOSTRAR
[3] - PESQUISAR
[4] - REMOVER
[5] - INSERIR EM UMA POSIÇÃO
[0] - SAIR

Digite sua opção: 2
```


FAMEPI
Faculdade do Amazonas de Ensino, Pesquisa e Inovação
FUNDAÇÃO MATIAS MACHLINE

```
********* MENU DE PESQUISA *********
[1] - PESQUISAR POR CÓDIGO
[2] - PESQUISAR POR NOME
[3] - PESQUISAR POR QUANTIDADE
[0] - SAIR DA PESQUISA

Digite sua opção: 1
```

```c
void pesquisar(tipo_lista *lista){
    if(vazia(lista)){
        printf("\n\n Impossível pesquisar na lista. Motivo: LISTA
        getch();                                          VAZIA !!!");
        return;
    }

    int op;

    printf("\n\n ********* MENU DE PESQUISA *********");

    printf("\n [1] - PESQUISAR POR CÓDIGO");

    printf("\n [2] - PESQUISAR POR NOME");

    printf("\n [3] - PESQUISAR POR QUANTIDADE");

    printf("\n [0] - SAIR DA PESQUISA");

    printf("\n\n Digite sua opção: ");
    scanf("%d", &op);

    switch(op){

            case 0: return;
                    break;
            case 1: pesqCod(lista);
                    break;
            case 2: pesqNome(lista);
                    break;
            case 3: pesqQuant(lista);
                    break;
            default:
                printf("\n\n OPÇÃO INVÁLIDA !!!");
                getch();
        }
    getch();
}
```

```c
int main( ) {
    setlocale(LC_ALL, "");

    int op;

    tipo_lista lista;

    flvazia(&lista);

    do{
        system("cls");

        op = menu();

        switch(op){

            case 0: return(0);
                    break;

            case 1: inserir(&lista);
                    break;

            case 2: mostrar(&lista);
                    break;

            case 3: pesquisar(&lista)
                    break;

            case 4: remover(&lista);
                    break;

            case 5: inserirPos(&lista);
                    break;

            default:
                printf("\n\n OPÇÃO INVÁLIDA !!!");
                getch();
        }

    }while(1);

    return(0);
}
```

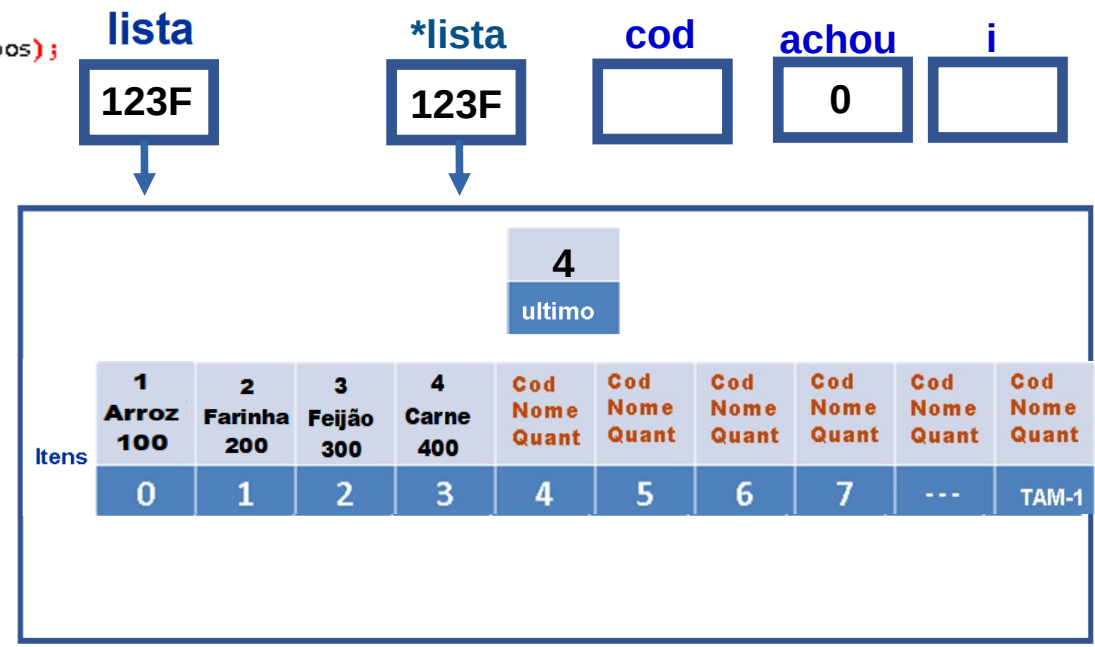



```
********** MENU **********
[1] - INSERIR
[2] - MOSTRAR
[3] - PESQUISAR
[4] - REMOVER
[5] - INSERIR EM UMA POSIÇÃO
[0] - SAIR

Digite sua opção: 3
```


FAMEPI
Faculdade do Amazonas de Ensino, Pesquisa e Inovação
FUNDAÇÃO MATIAS MACHLINE

```
Digite o Código para pesquisa: 4


===== ITEM ENCONTRADO NA POSIÇÃO 4 ===
Código: 4
Nome: Carne
Quantidade: 400
```

```
void mostrarItem(tipo_item item, int pos){

    printf("\n\n ===== ITEM ENCONTRADO NA POSIÇÃO %d ===", pos);

    printf("\n Código: %d", item.cod);

    printf("\n Nome: %s", item.nome);

    printf("\n Quantidade: %d", item.quant);
}
```




```
void pesquisar(tipo_lista *lista){
    if(vazia(lista)){
        printf("\n\n Impossível pesquisar na lista. Motivo: LISTA VAZIA !!!");
        getch();
        return;
    }

    int op;

    printf("\n\n ********* MENU DE PESQUISA *********");

    printf("\n [1] - PESQUISAR POR CÓDIGO");

    printf("\n [2] - PESQUISAR POR NOME");

    printf("\n [3] - PESQUISAR POR QUANTIDADE");

    printf("\n [0] - SAIR DA PESQUISA");

    printf("\n\n Digite sua opção: ");
    scanf("%d", &op);

    switch(op){
            case 0: return;
                    break;
            case 1: pesqCod(lista);
                    break;
            case 2: pesqNome(lista);
                    break;
            case 3: pesqQuant(lista);
                    break;
            default:
                printf("\n\n OPÇÃO INVÁLIDA !!!");
                getch();
        }
    getch();
}
```

```
void pesqCod(tipo_lista *lista){

    int cod, achou = 0, i;

    printf("\n\n Digite o Código para pesquisa: ");
    scanf("%d", &cod);

    for(i = 0; i < lista->ultimo; i ++){
        if(lista->itens[i].cod == cod){
            mostrarItem(lista->itens[i], i+1);
            achou = 1;
        }
    }

    if(!achou)
        printf("\n\n O item de codigo %d não existe na lista !!!", cod);
}
```

```
O item de codigo 5 não existe na lista !!!
```

pesqQuant igual a pesqCod

```
void pesqCod(tipo_lista *lista){

    int cod, achou = 0, i;

    printf("\n\n Digite o Código para pesquisa: ");
    scanf("%d", &cod);

    for(i = 0; i < lista->ultimo; i ++){
        if(lista->itens[i].cod == cod){
            mostrarItem(lista->itens[i], i+1);
            achou = 1;
        }
    }

    if(!achou)
        printf("\n\n O item de codigo %d não existe na lista !!!", cod);
}
```

```
printf("\n\n Digite a Quantidade para pesquisa: ");
scanf("%d", &quant);

if(lista->itens[i].quant == quant){
```

Detalhe de pesqNome

```
fflush(stdin);
printf("\n\n Digite o Nome para pesquisa: ");
gets(nomePesq);

if( ! strcmp ( lista->itens[i].nome , nomePesq ) ) {
```

Algoritmos e Estrutura de Dados 2

Pesquisa sequencial

- **Análise:**
  - Pesquisa com sucesso:
    - melhor caso : C(n) = 1
    - pior caso: C(n) = n
    - caso médio: C(n) = (n + 1) / 2
  - Pesquisa sem sucesso:
    - C (n) = n + 1.

O algoritmo de pesquisa sequencial é a melhor escolha para o problema de **pesquisa com n < 25.**

FAMEPI
Faculdade do Amazonas de Ensino, Pesquisa e Inovação
FUNDAÇÃO MATIAS MACHLINE

Sumário

Sumário

Algoritmos e Estrutura de Dados 2

Pesquisa binária

- Para uma massa de dados **grande** e se os registros estiverem **em ordem**

  ➔ Utilizar um método muito mais eficiente para encontrar o dado procurado:

  a pesquisa binária

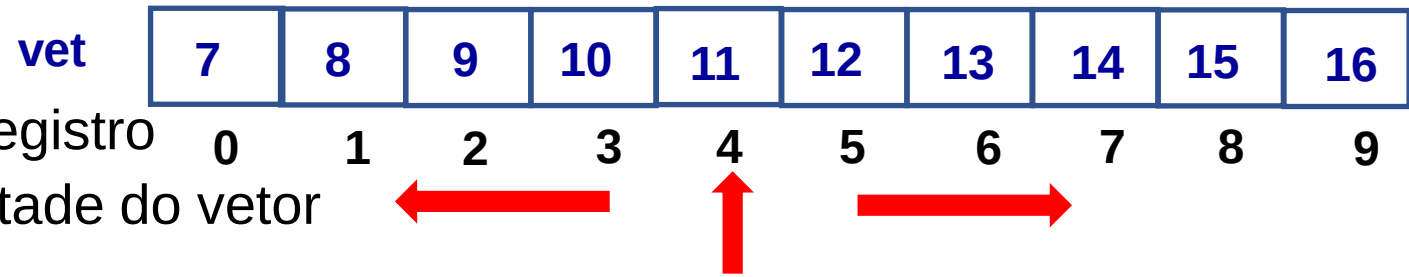
- Caso os registros não estejam em ordem: pode-se utilizar um dos métodos de ordenação existentes e que serão estudados nas próximas aulas.
- A eficiência deste método está na utilização da estratégia de "**dividir para conquistar**"

Algoritmos e Estrutura de Dados 2

Pesquisa binária

- Lembrando: os registros precisam estar em ordem
- Estratégia da pesquisa para saber se uma chave está presente no vetor:

1. Comparar a chave de pesq com a chave do registro que está na posição do meio do vetor.
2. Se a chave é menor então o registro procurado está na primeira metade do vetor
3. Se a chave é maior então o registro procurado está na segunda metade do vetor.
4. Repetir com a metade escolhida **até** que a chave **seja encontrada ou** que se constate que a chave **não existe** no vetor.



FAMEPI
Faculdade do Amazonas de Ensino, Pesquisa e Inovação
FUNDAÇÃO MATIAS MACHLINE

Exemplo de Pesquisa Binária: G

Pesquisa binária

	0	1	2	3	4	5	6	7
Chaves iniciais:	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*
	A	*B*	*C*	***D***	*E*	*F*	*G*	*H*
					E	***F***	*G*	*H*
							G	*H*

FAMEPI
Faculdade do Amazonas de Ensino, Pesquisa e Inovação
FUNDAÇÃO MATIAS MACHLINE

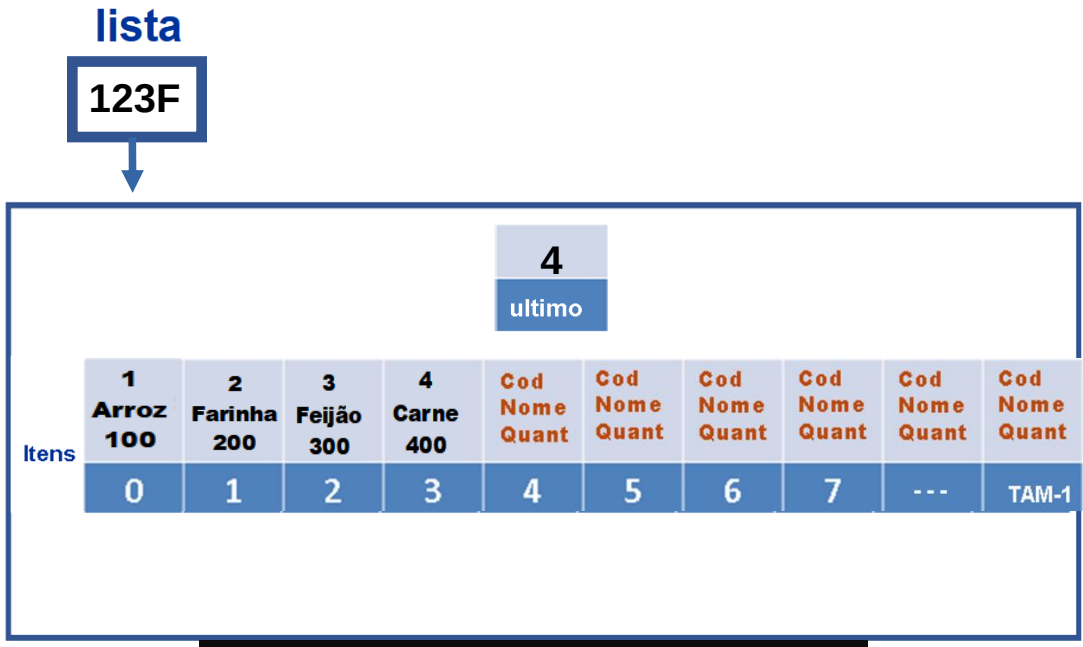
Algoritmos e Estrutura de Dados 2

Pesquisa binária

Implementação

FAMEPI
Faculdade do Amazonas de Ensino, Pesquisa e Inovação
FUNDAÇÃO MATIAS MACHLINE

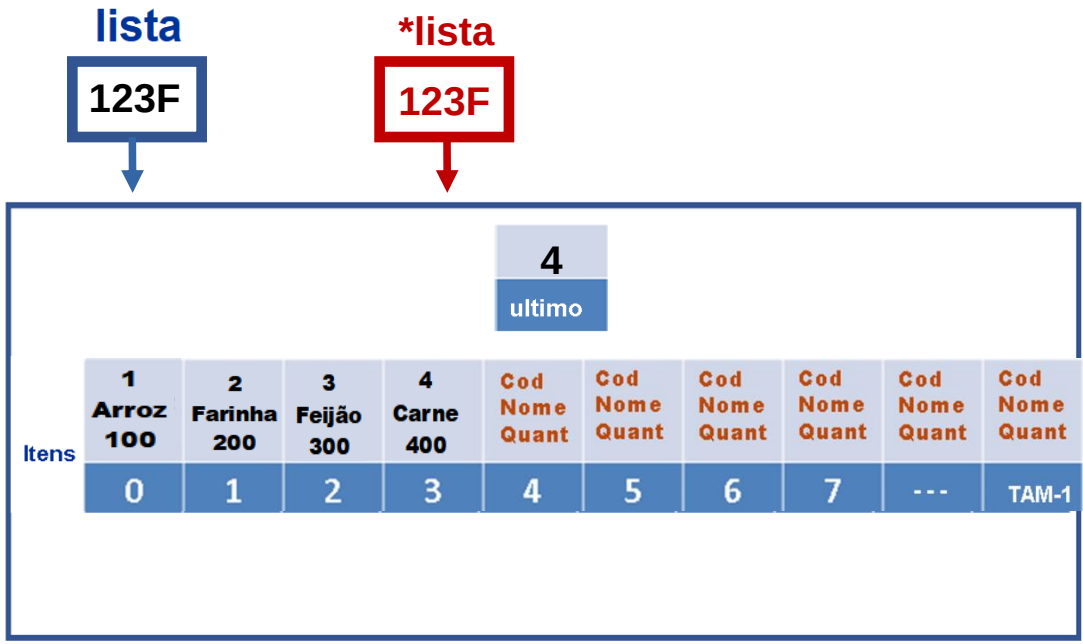
```c
int main( ) {
    setlocale(LC_ALL, "");

    int op;

    tipo_lista lista;

    flvazia(&lista);

    do{
        system("cls");

        op = menu();

        switch(op){

            case 0: return(0);
                    break;

            case 1: inserir(&lista);
                    break;

            case 2: mostrar(&lista);
                    break;

            case 3: pesquisar(&lista)
                    break;

            case 4: remover(&lista);
                    break;

            case 5: inserirPos(&lista);
                    break;

            default:
                printf("\n\n OPÇÃO INVÁLIDA !!!");
                getch();
        }

    }while(1);

    return(0);
}
```




```
********** MENU **********
[1] - INSERIR
[2] - MOSTRAR
[3] - PESQUISAR
[4] - REMOVER
[5] - INSERIR EM UMA POSIÇÃO
[0] - SAIR

Digite sua opção: 3
```

Pesquisa binária

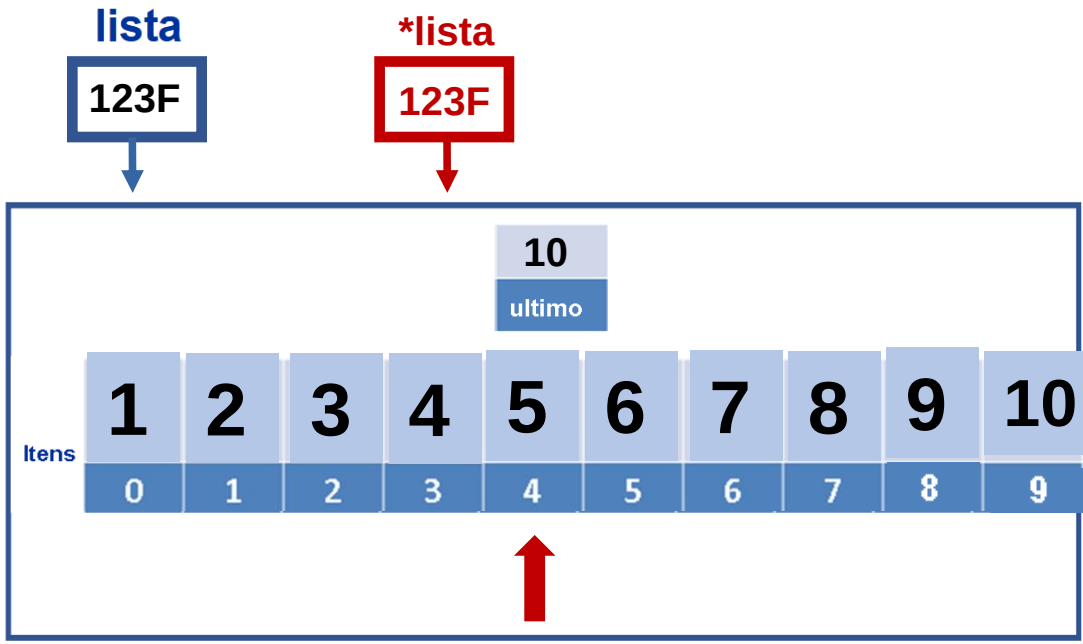
Somente pelo campo ordenado




```
void pesquisar(tipo_lista *lista){
    if(vazia(lista)){
        printf("\n\n Impossível pesquisar na lista.
        getch();                    Motivo: LISTA VAZIA !!!");
        return;
    }

    int cod, achou = 0;

    int esq, dir, meio;

    printf("\n\n Digite o Código para pesquisa binária: ");
    scanf("%d", &cod);

    esq = 0; dir = lista->ultimo -1; // limites iniciais
    do{

        meio = (esq + dir) / 2;

        if(cod > lista->itens[meio].cod) esq = meio + 1;

        else dir = meio - 1;

    }while(cod != lista->itens[meio].cod && esq <= dir);

    if(cod == lista->itens[meio].cod){

        mostrarItem(lista->itens[meio], meio + 1);

        achou = 1;  // encontrou o item com o cod
    }
    if(!achou)
        printf("\n\n Não existe item com o ódigo %d na lista !!!", cod);
    getch();
}
```

Pesquisa binária

```
void pesquisar(tipo_lista *lista){
    if(vazia(lista)){
        printf("\n\n Impossível pesquisar na lista.
        getch();                    Motivo: LISTA VAZIA !!!");
        return;
    }

    int cod, achou = 0;

    int esq, dir, meio;

    printf("\n\n Digite o Código para pesquisa binária: ");
    scanf("%d", &cod);

    esq = 0; dir = lista->ultimo -1; // limites iniciais
    do{

        meio = (esq + dir) / 2;

        if(cod > lista->itens[meio].cod) esq = meio + 1;

        else dir = meio - 1;

    }while(cod != lista->itens[meio].cod && esq <= dir);

    if(cod == lista->itens[meio].cod){

        mostrarItem(lista->itens[meio], meio + 1);

        achou = 1;  // encontrou o item com o cod
    }
```



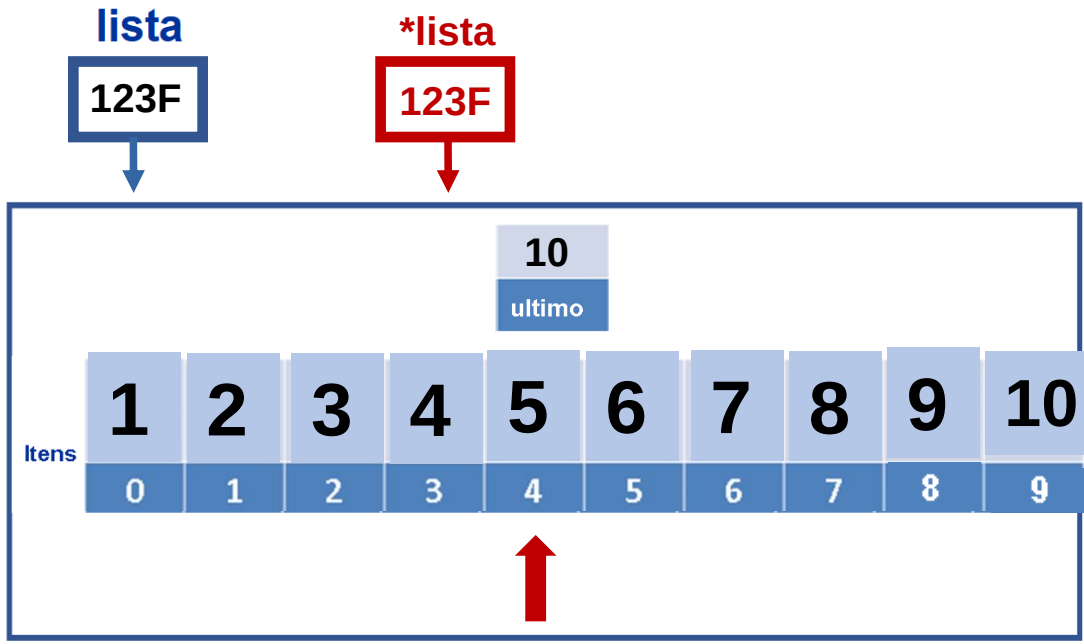
8	0	0	9	4
cod	achou	esq	dir	meio

```
Digite o Código para pesquisa binária: 8
```

```
    if(!achou)
        printf("\n\n Não existe item com o ódigo %d na lista !!!", cod);
    getch();
}
```

Pesquisa binária

```
void pesquisar(tipo_lista *lista){
    if(vazia(lista)){
        printf("\n\n Impossível pesquisar na lista.
        getch();                    Motivo: LISTA VAZIA !!!");
        return;
    }

    int cod, achou = 0;

    int esq, dir, meio;

    printf("\n\n Digite o Código para pesquisa binária: ");
    scanf("%d", &cod);

    esq = 0; dir = lista->ultimo -1; // limites iniciais
    do{

        meio = (esq + dir) / 2;

        if(cod > lista->itens[meio].cod) esq = meio + 1;

        else dir = meio - 1;

    }while(cod != lista->itens[meio].cod && esq <= dir);

    if(cod == lista->itens[meio].cod){

        mostrarItem(lista->itens[meio], meio + 1);

        achou = 1;  // encontrou o item com o cod
    }
```

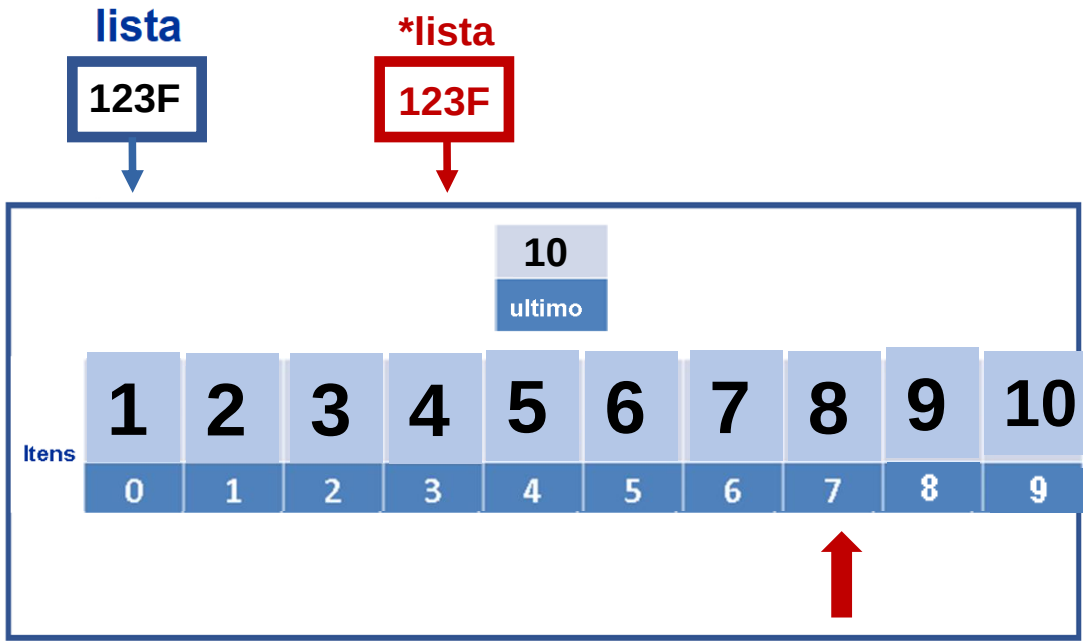


8	0	5	9	4
cod	achou	esq	dir	meio

```
    if(!achou)
        printf("\n\n Não existe item com o ódigo %d na lista !!!", cod);
    getch();
}
```

Pesquisa binária

```
void pesquisar(tipo_lista *lista){
    if(vazia(lista)){
        printf("\n\n Impossível pesquisar na lista.
        getch();                    Motivo: LISTA VAZIA !!!");
        return;
    }

    int cod, achou = 0;

    int esq, dir, meio;

    printf("\n\n Digite o Código para pesquisa binária: ");
    scanf("%d", &cod);

    esq = 0; dir = lista->ultimo -1; // limites iniciais
    do{

        meio = (esq + dir) / 2;

        if(cod > lista->itens[meio].cod) esq = meio + 1;

        else dir = meio - 1;

    }while(cod != lista->itens[meio].cod && esq <= dir);

    if(cod == lista->itens[meio].cod){

        mostrarItem(lista->itens[meio], meio + 1);

        achou = 1;  // encontrou o item com o cod
    }
```



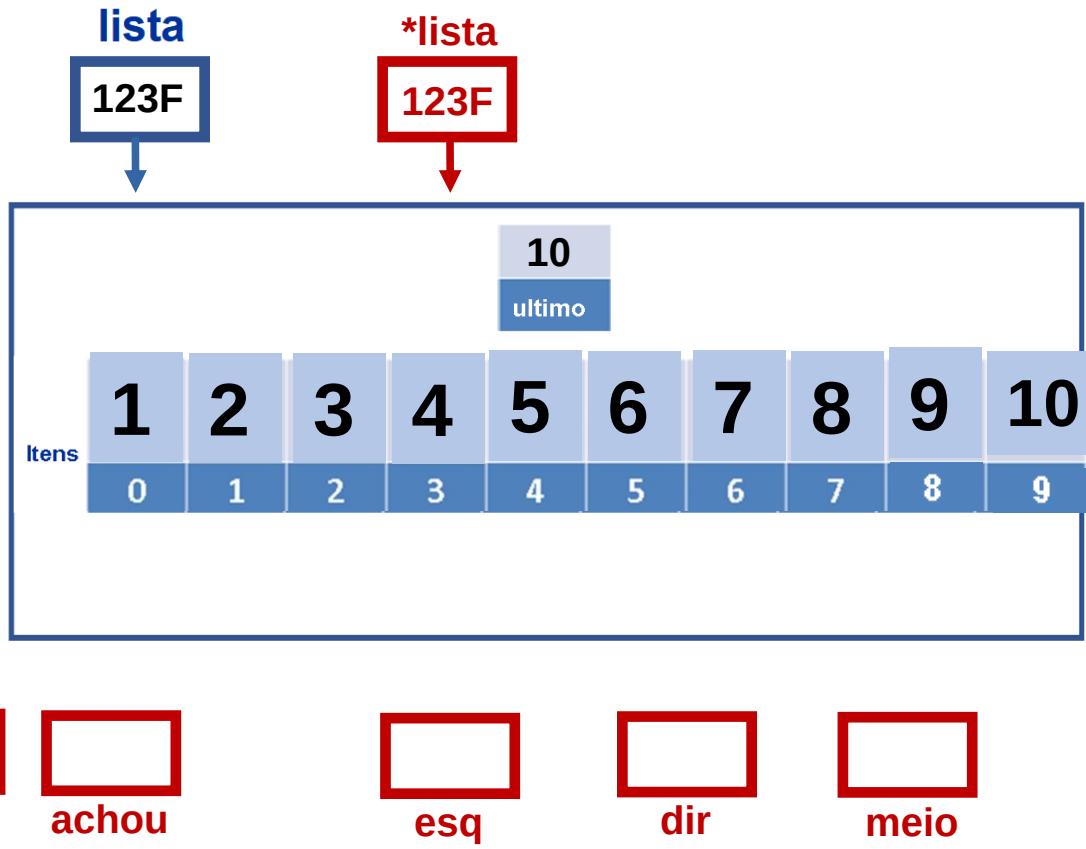

```
===== ITEM ENCONTRADO NA POSIÇÃO 8 ===
Código: 8
Nome: Maionese
Quantidade: 900
```

```
    if(!achou)
        printf("\n\n Não existe item com o ódigo %d na lista !!!", cod);
    getch();
}
```

Pesquisa binária

```
void pesquisar(tipo_lista *lista){
    if(vazia(lista)){
        printf("\n\n Impossível pesquisar na lista.
        getch();                    Motivo: LISTA VAZIA !!!");
        return;
    }

    int cod, achou = 0;

    int esq, dir, meio;

    printf("\n\n Digite o Código para pesquisa binária: ");
    scanf("%d", &cod);

    esq = 0; dir = lista->ultimo -1; // limites iniciais
    do{

        meio = (esq + dir) / 2;

        if(cod > lista->itens[meio].cod) esq = meio + 1;

        else dir = meio - 1;

    }while(cod != lista->itens[meio].cod && esq <= dir);

    if(cod == lista->itens[meio].cod){

        mostrarItem(lista->itens[meio], meio + 1);

        achou = 1;  // encontrou o item com o cod
    }
    if(!achou)
        printf("\n\n Não existe item com o ódigo %d na lista !!!", cod);
    getch();
}
```

Testar para cod não existente. Ex: 11



Sumário

Sumário

Pesquisa Binária

Consequentemente, a pesquisa binária **não deve** ser usada em aplicações muito dinâmicas.

- Análise
  - A cada iteração do algoritmo, é feito a comparação.
  - **Logo**:
    - no melhor caso, o número comparações é igual a 1
    - no pior caso, o número comparações é igual a cerca de $\log_2 n$

    (se n = 1024 ➔ $\log_2 1024 = X$ ➔ $2^x = 1024$ ➔ $2^x = 2^{10}$ ➔ $x = 10$)

  - **Ressalva**: o custo para manter o vetor ordenado é alto: a cada inserção na posição p do vetor implica no **deslocamento** dos registros a partir da posição p para as posições seguintes.

FAMEPI
Faculdade do Amazonas de Ensino, Pesquisa e Inovação
FUNDAÇÃO MATIAS MACHLINE