FAMEPI

Faculdade do Amazonas de Ensino, Pesquisa e Inovação

FUNDAÇÃO MATIAS MACHLINE

Algoritmos e Estrutura de Dados 2

Prof Pedro Corrêa

Objetivo da aula

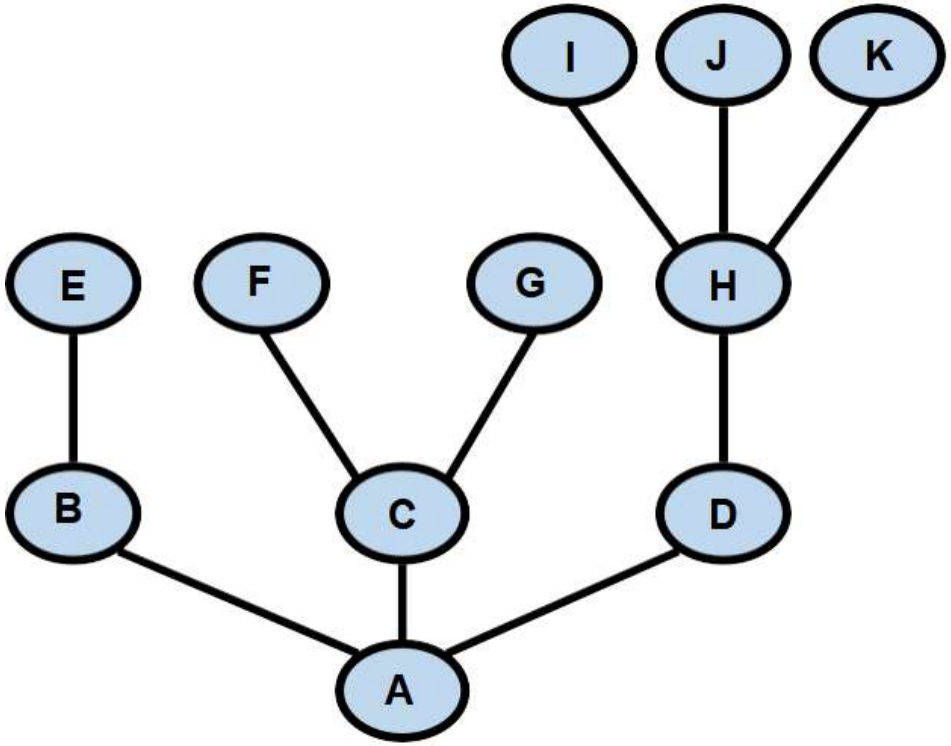
Utilizar a Estrutura de Dados do tipo Árvore Binária de Pesquisa

Sumário

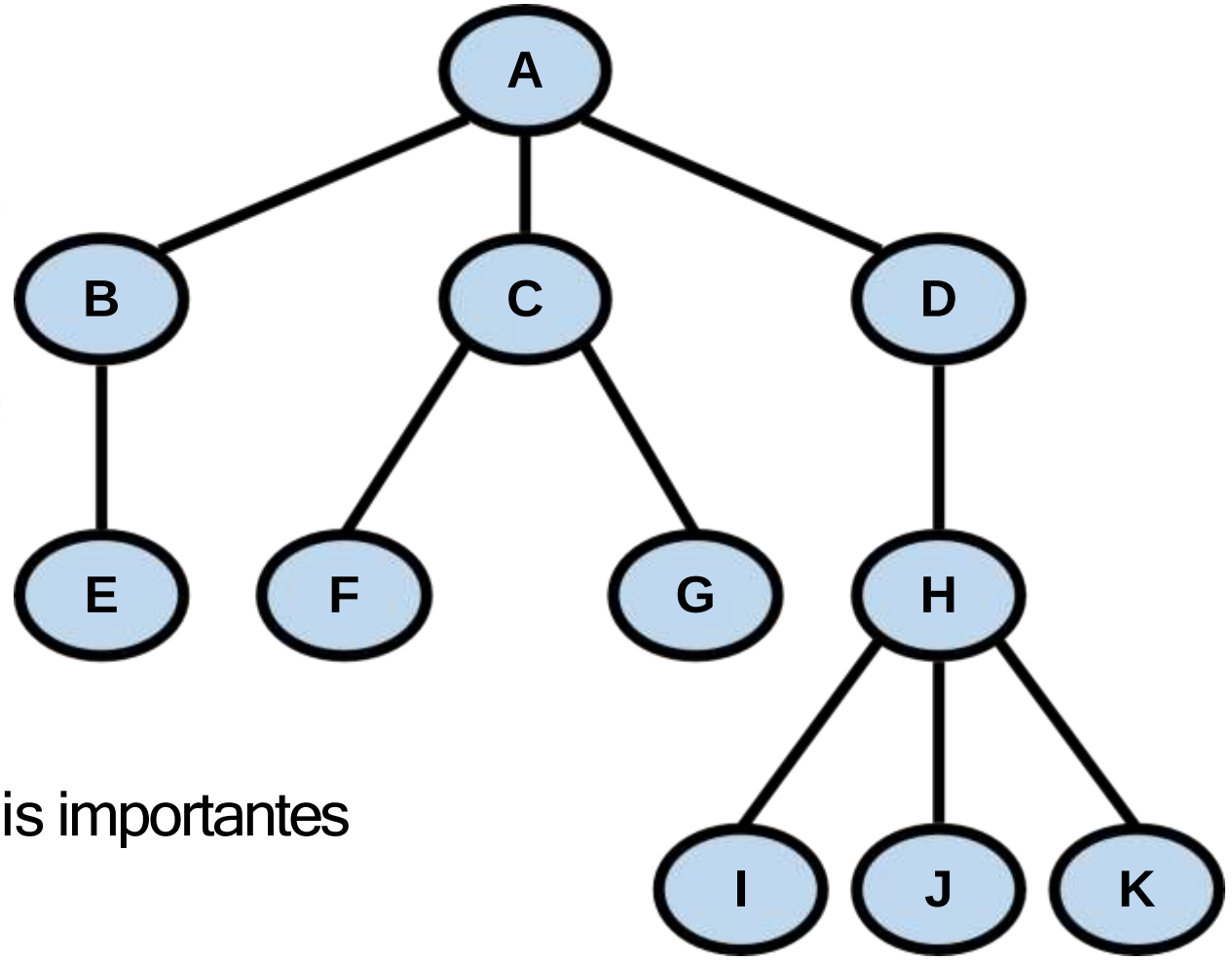
ÁRVORES

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

ÁRVORES



ÁRVORES

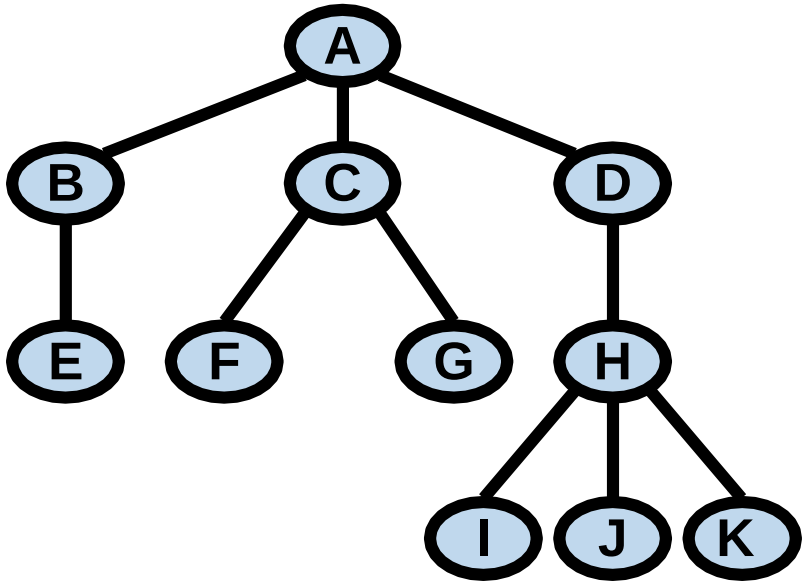


Constituem uma das estruturas mais importantes da área de computação.

São usadas em diversas aplicações

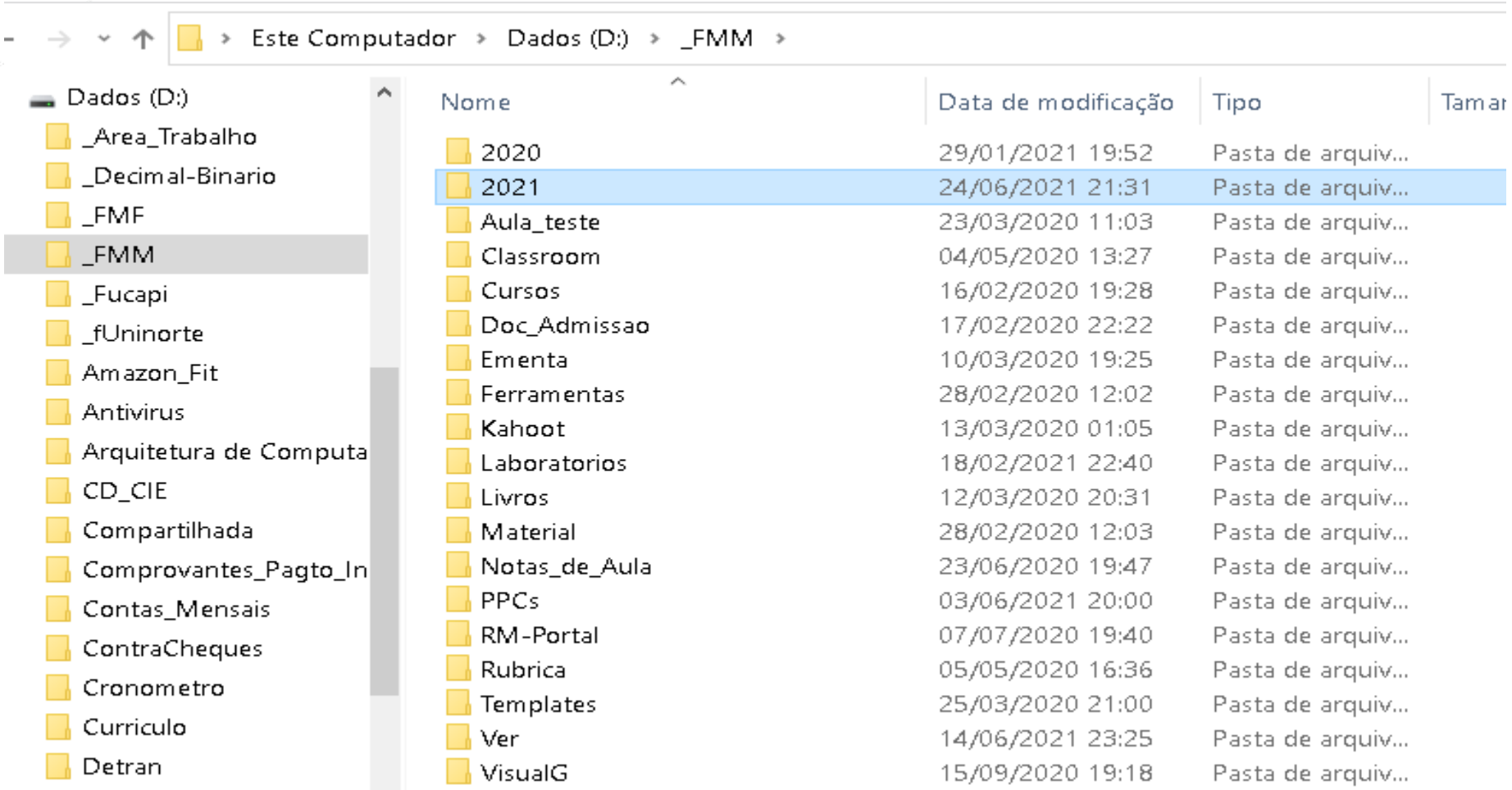
ÁRVORES

São formadas por um conjunto de nós que possuem relacionamento de **Hierarquia** ou **Subordinação**



EXEMPLOS DE APLICAÇÃO

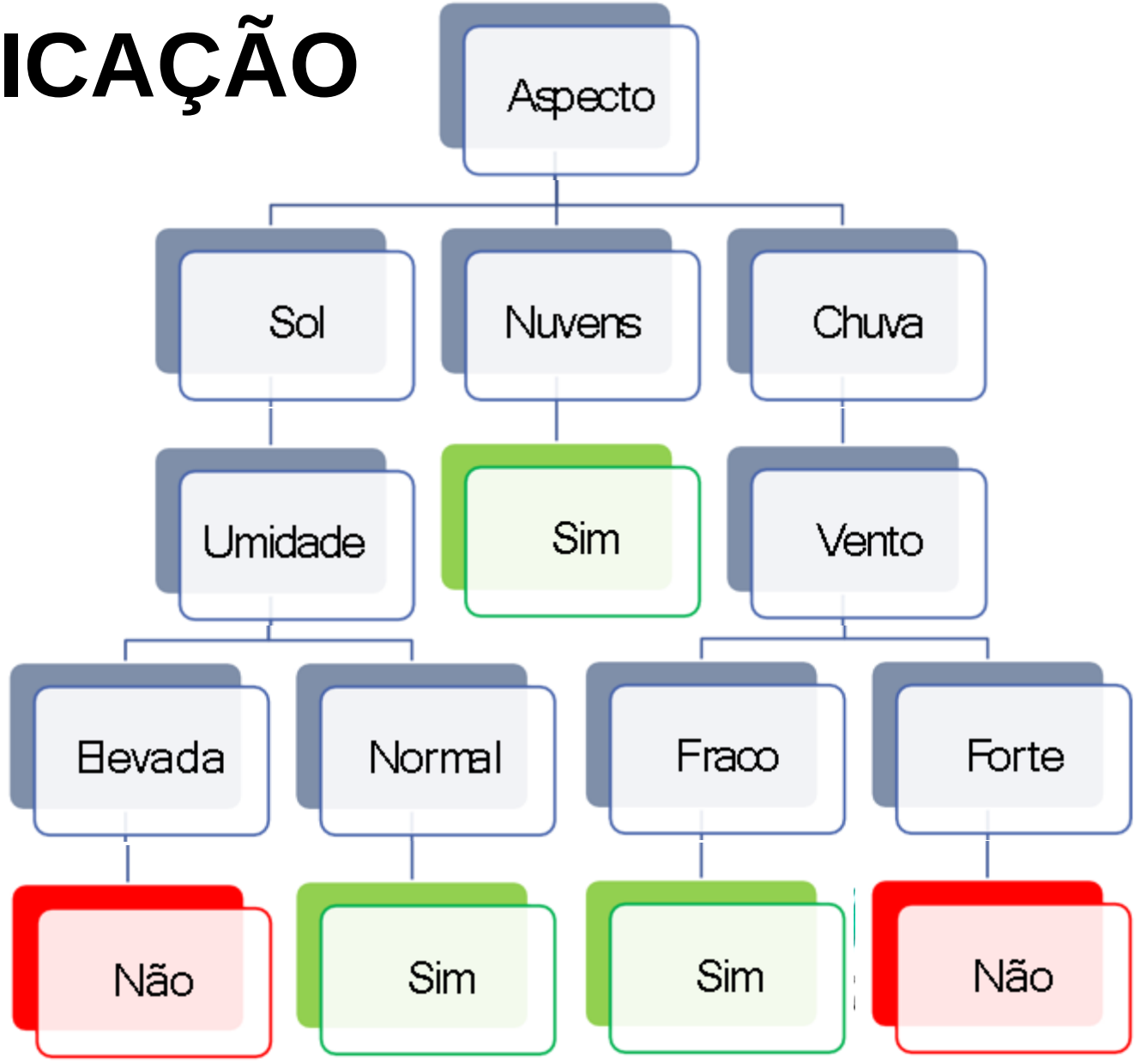
Sistema de arquivos



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

EXEMPLOS DE APLICAÇÃO

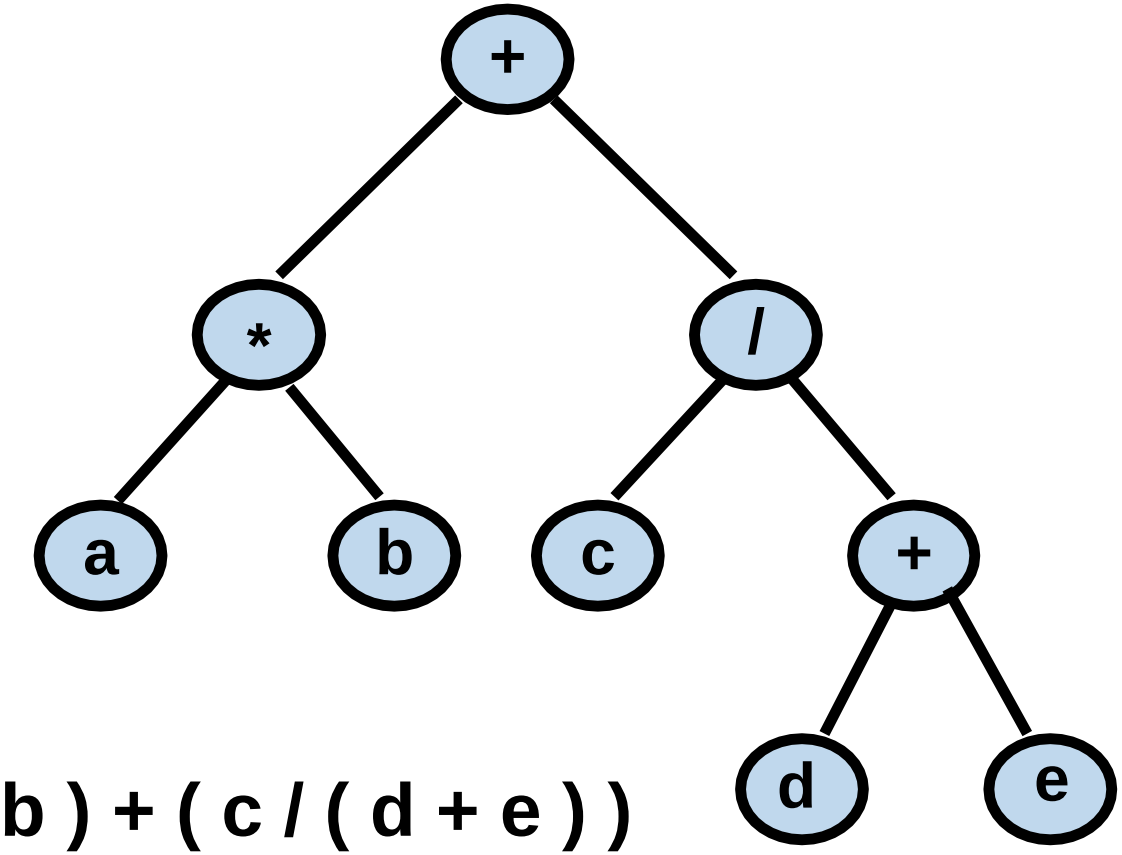
Árvore de decisão para jogar tênis



EXEMPLOS DE APLICAÇÃO

Árvore de derivação

- Usada pelos compiladores



Expressão aritmética: (a * b) + (c / (d + e))

DEFINIÇÃO FORMAL

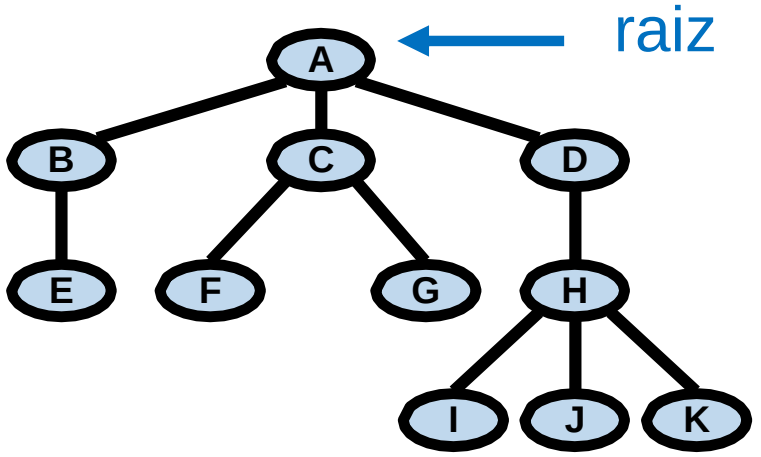
Conjunto finito de zero ou mais nós (**nodos** ou **vértices**), tal que:

Se o número de nós é maior do que zero:

- Existe um nó denominado **raiz** da árvore
- Os demais nós formam $m \geq 0$ conjuntos disjuntos $S_1, S_2, ..., S_m$, onde cada um destes é uma árvore (S_i são denominadas **sub-árvores**)

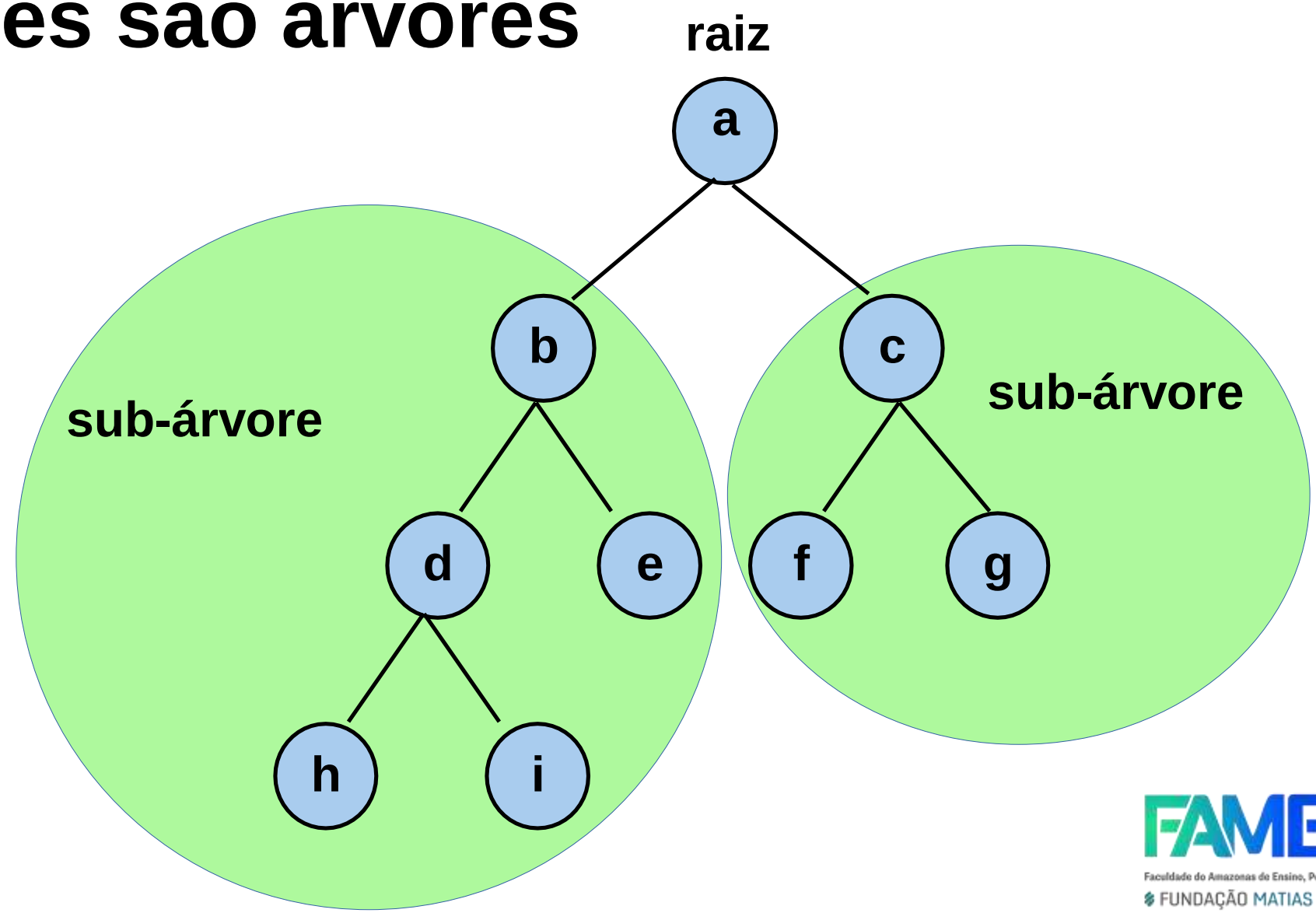
Se o número de nós é igual a zero:

- **Árvore vazia**

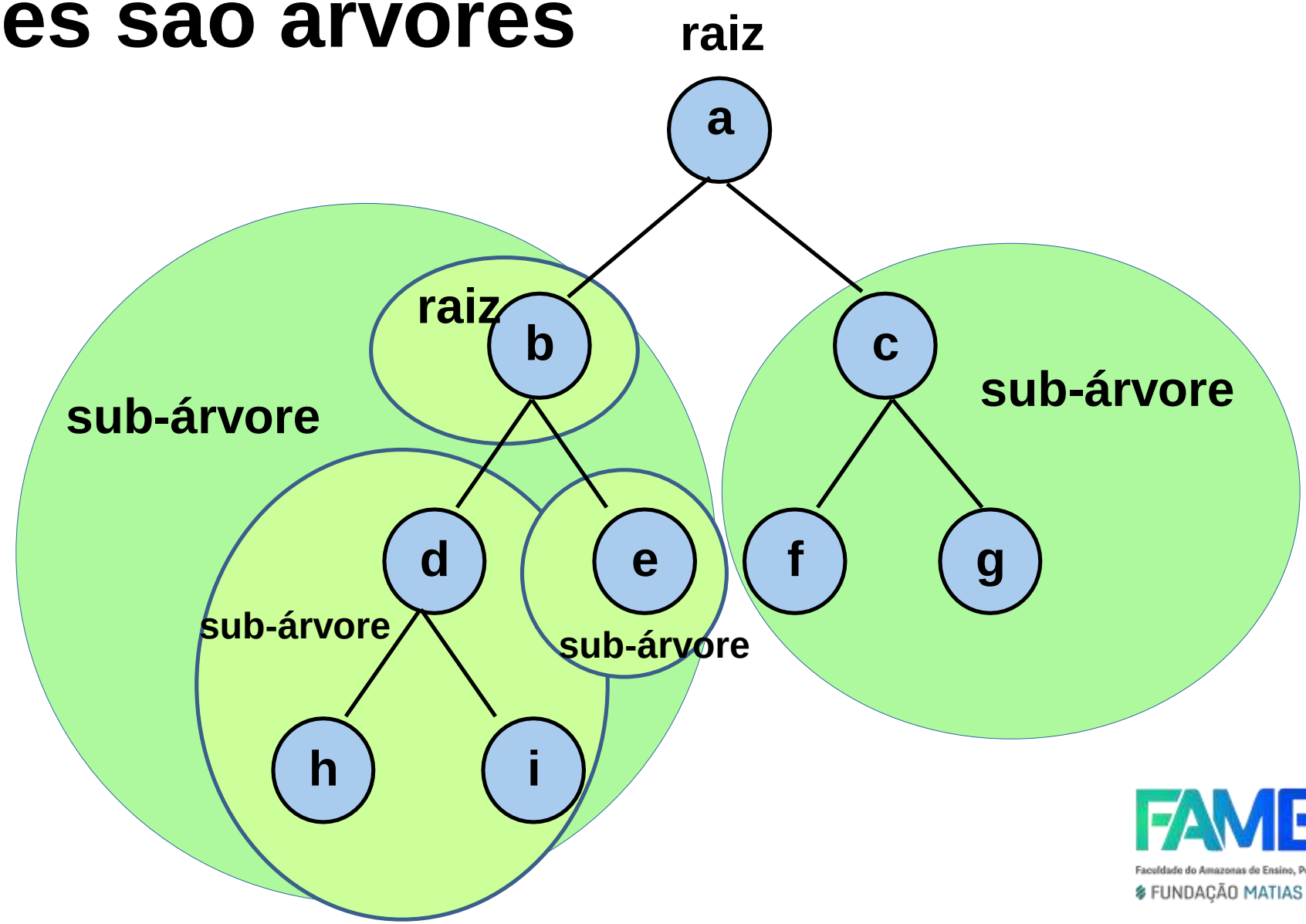


FAMEPI
Faculdade do Amazonas de Ensino, Pesquisa e Inovação
FUNDAÇÃO MATIAS MACHLINE

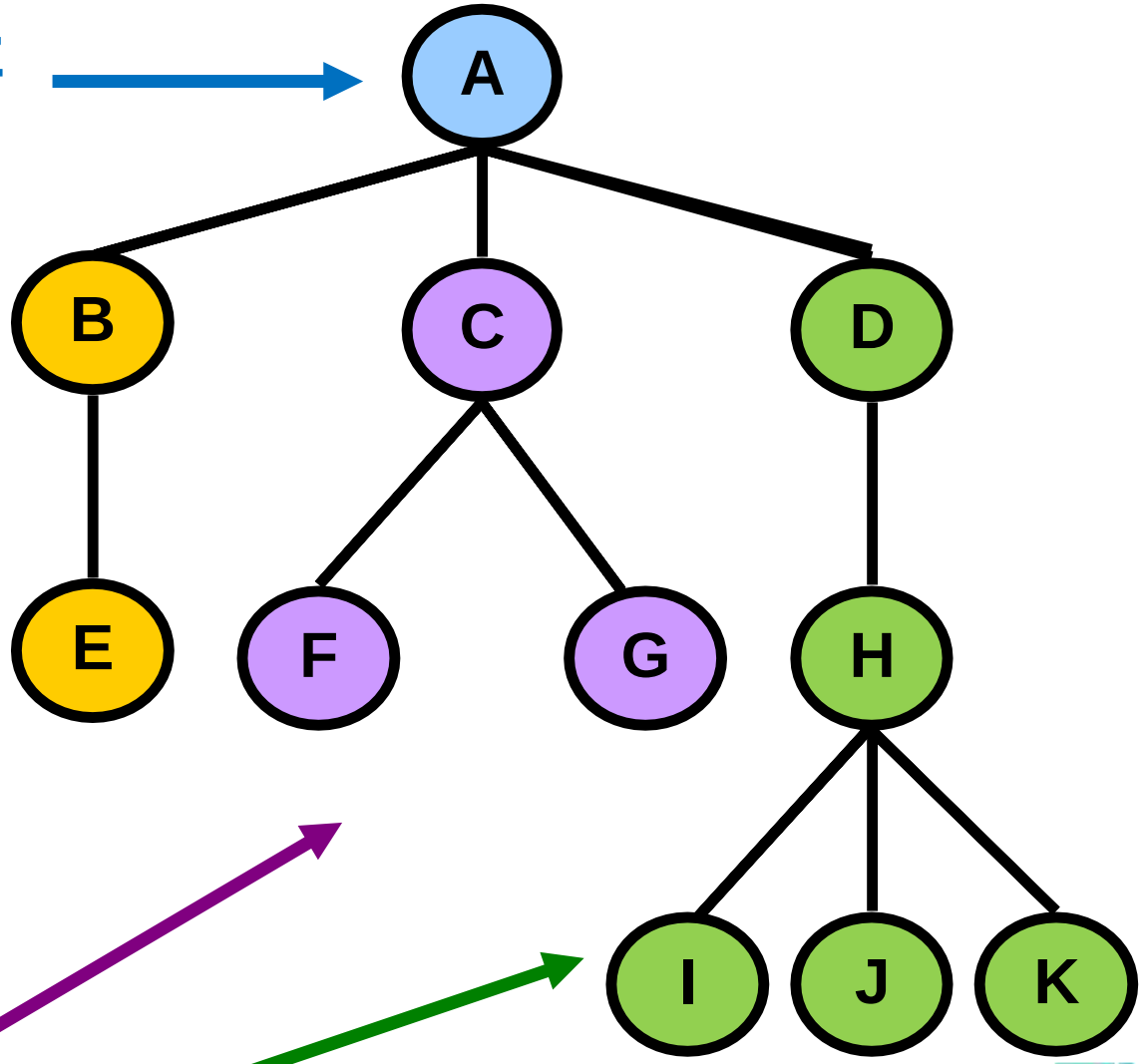
Sub-árvores são árvores



Sub-árvores são árvores

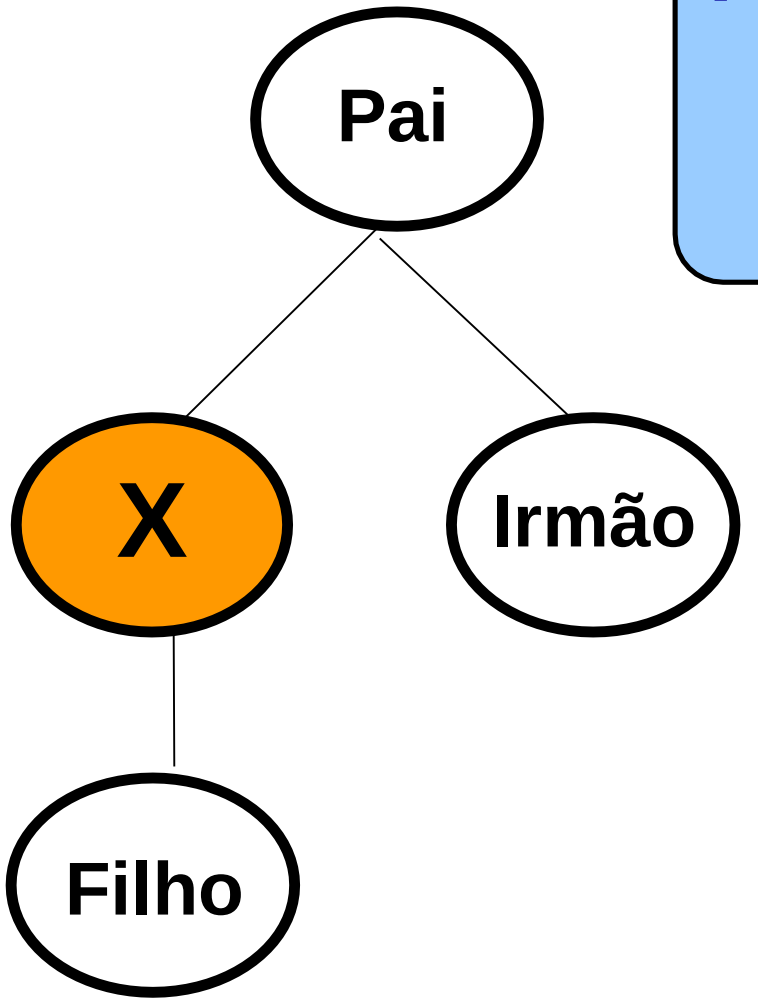


TERMINOLOGIA



TERMINOLOGIA

Usando o nó **X** como referencial



Pai
= mãe
= ascendente
= antecessor

Irmão
= irmã

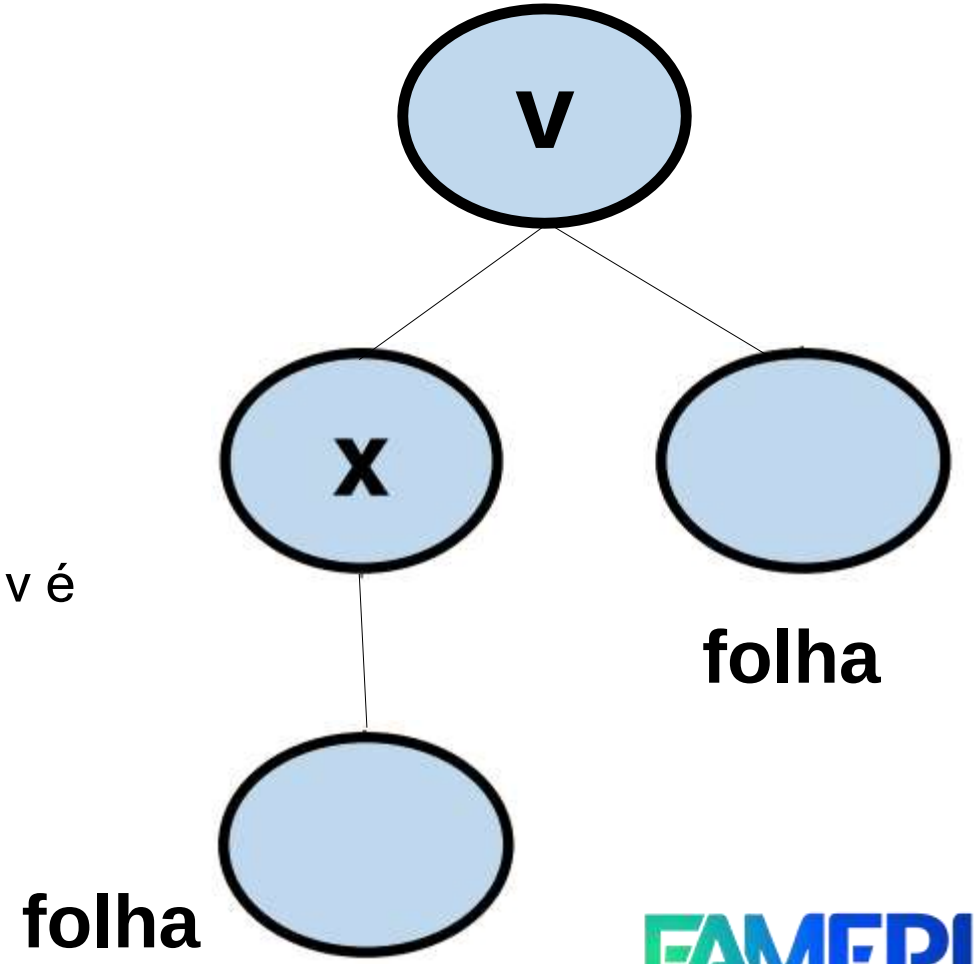
Filho
= filha
= descendente
= sucessor

TERMINOLOGIA

Um nó é ancestral e descendente de si mesmo

Se **x** pertence à subárvore enraizada em **v**:

- x é **descendente** de v
- v é **ancestral** de x
- Se x é diferente de v, x é **descendente próprio** de v, e v é **ancestral próprio** de x
- Um **nó folha** não possui descendentes próprios



TERMINOLOGIA

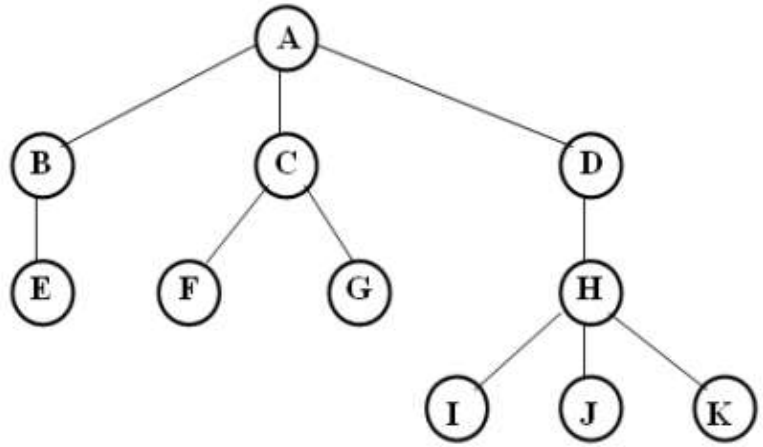
Resumindo

- Um nó n_i é ancestral e descendente de si mesmo.

- Se há um caminho de n_i **para** n_j, então n_i é ancestral de n_j e n_j é descendente de n_i.

- Ancestral próprio ou descendente próprio de um nó é um ancestral ou descendente, respectivamente, diferente desse nó.

- Na árvore ao lado:
  - Ancestrais próprios de J: A, D, H
  - Descendentes próprios de D: H, I, J, K

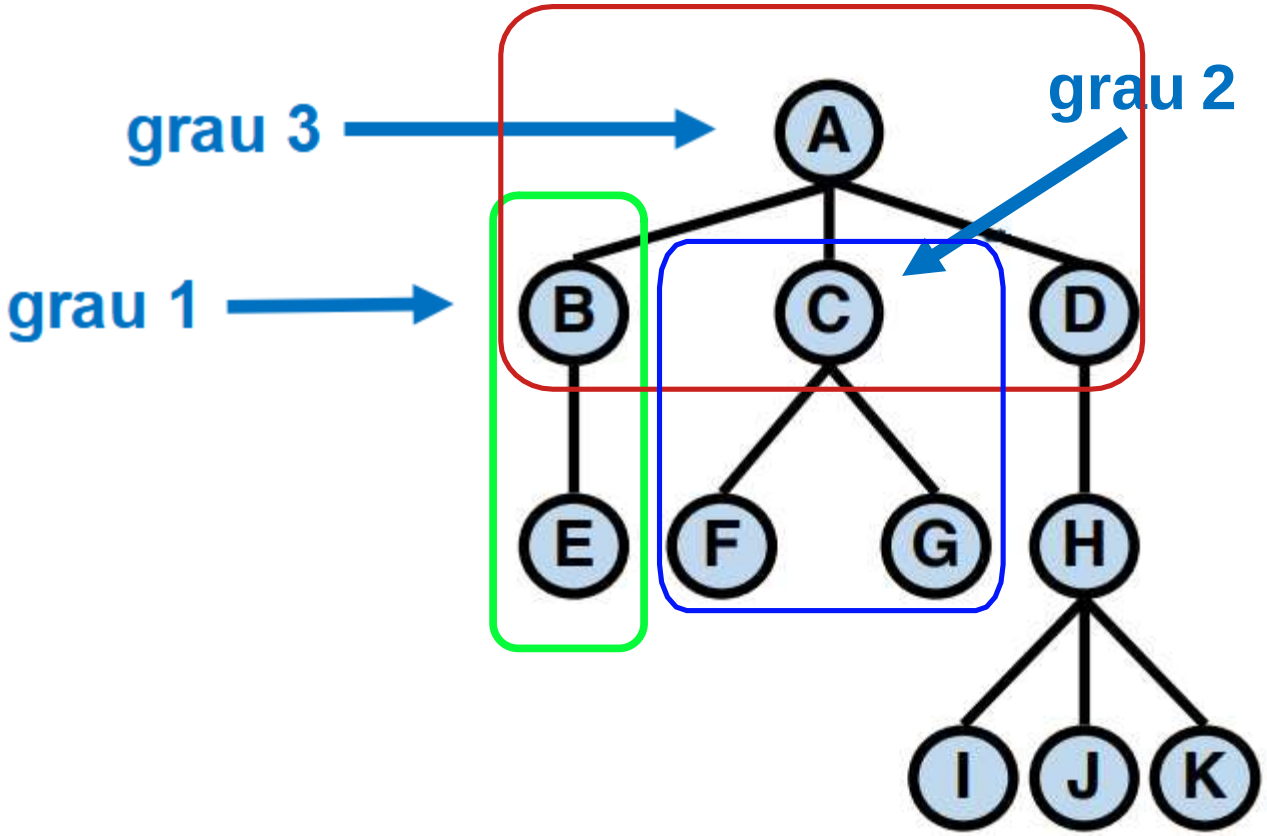


TERMINOLOGIA

Grau de um nó

Ou grau de saída

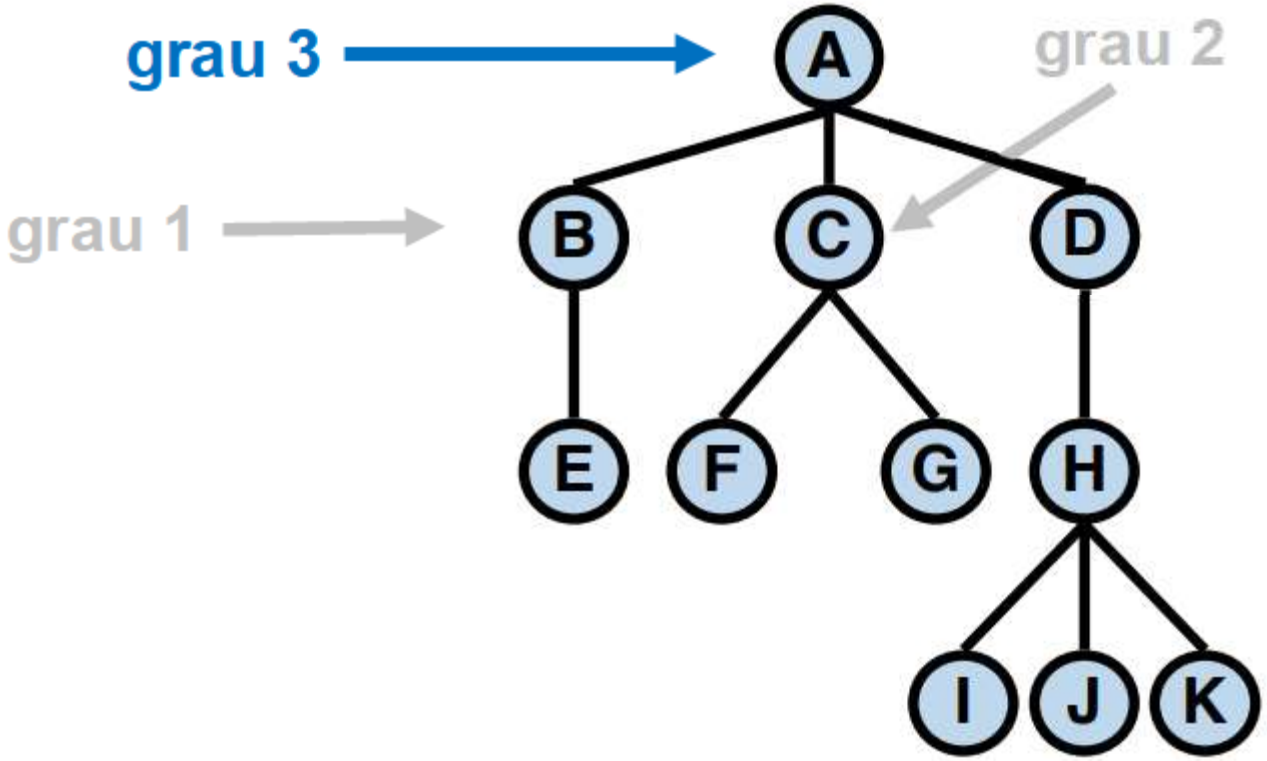
- Número de sub-árvores do nó ou
- Número de filhos de um nó



TERMINOLOGIA

Grau da árvore

Máximo entre os graus de seus nós

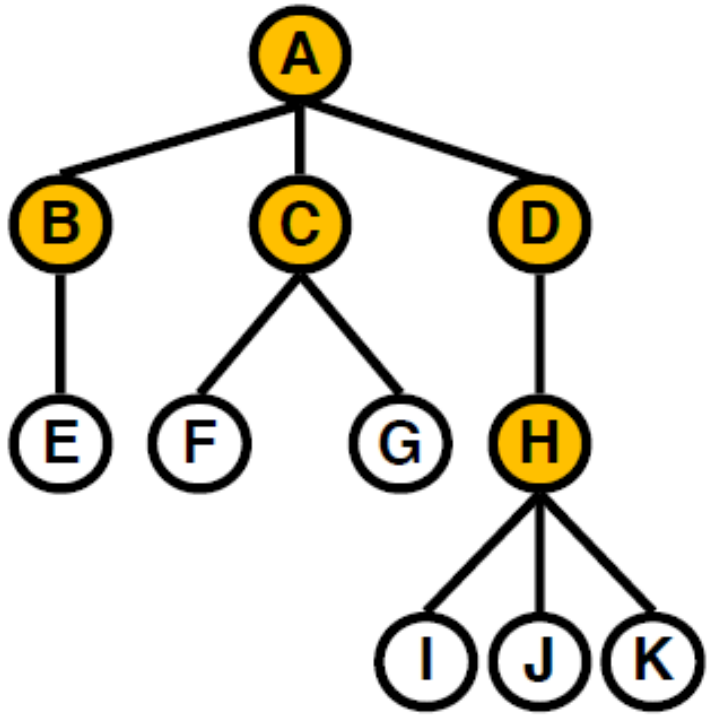


TERMINOLOGIA

Nó interno

Nó interno (ou nó de derivação)

- nó com grau maior do que zero (tem pelo menos um filho)

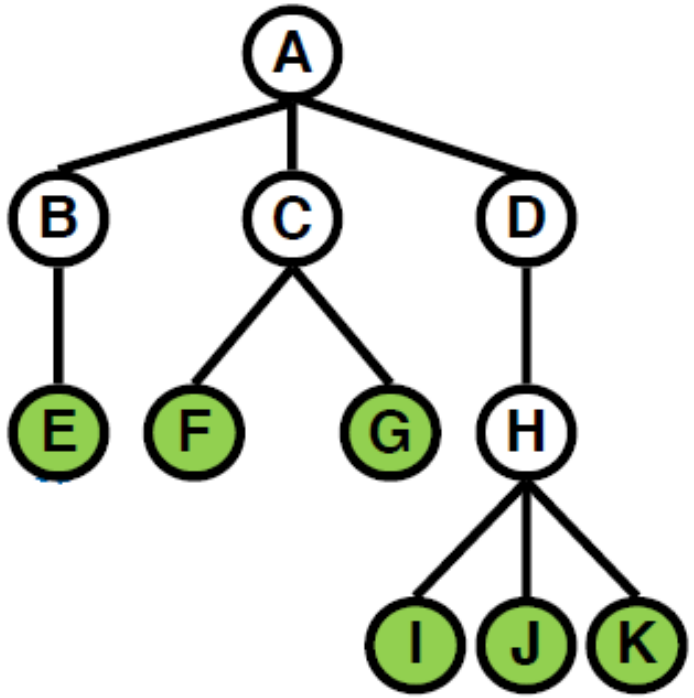


TERMINOLOGIA

Nó folha

Nó folha (nó terminal ou externo)

nó com grau igual a zero (nó sem filhos)

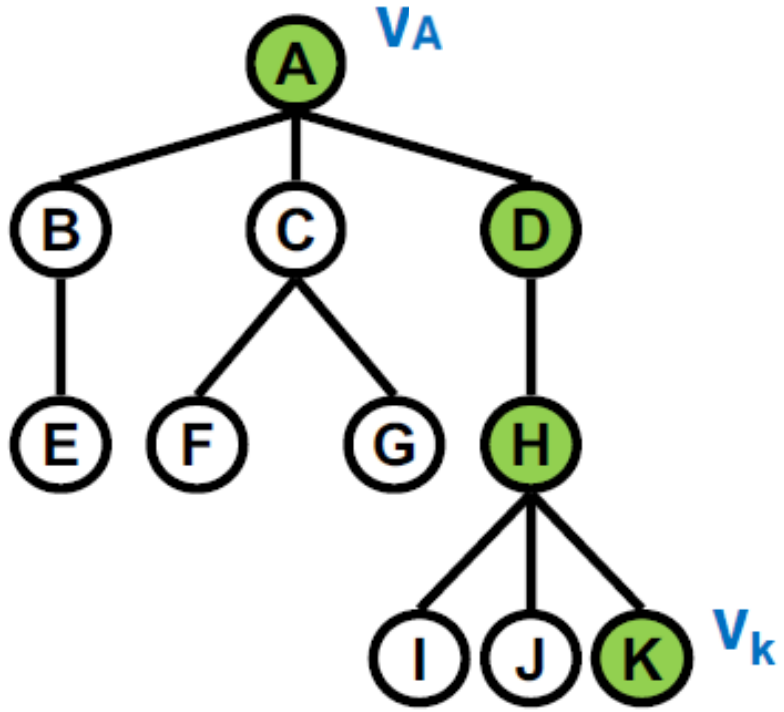


TERMINOLOGIA

Caminho

É uma sequência de nós consecutivos distintos entre dois nós

Caminho = A - D - H - K



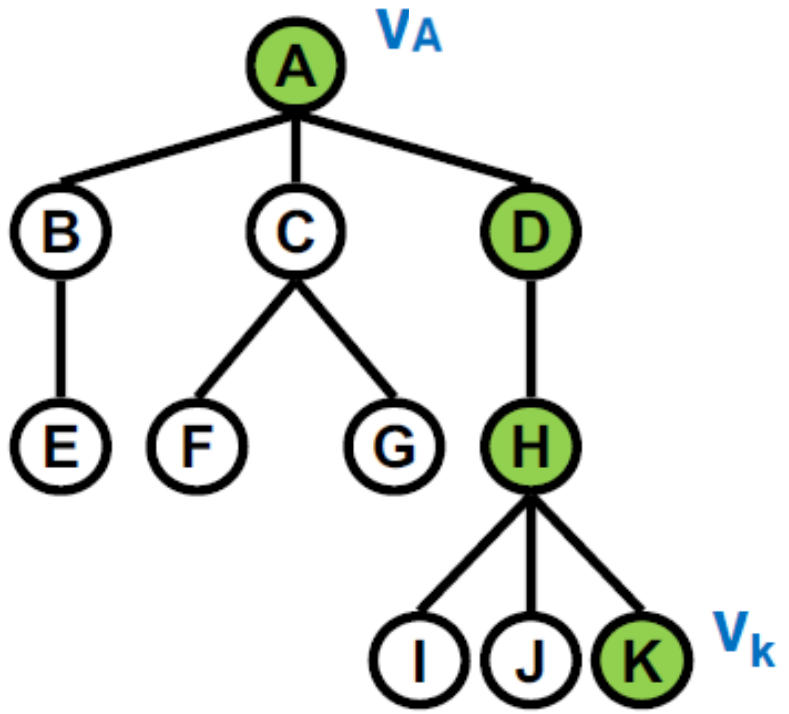
TERMINOLOGIA

Comprimento do caminho

Comprimento do caminho é o número de ligações (arestas) entre os nós do caminho

É igual ao número de nós - 1

Comprimento do caminho= 3

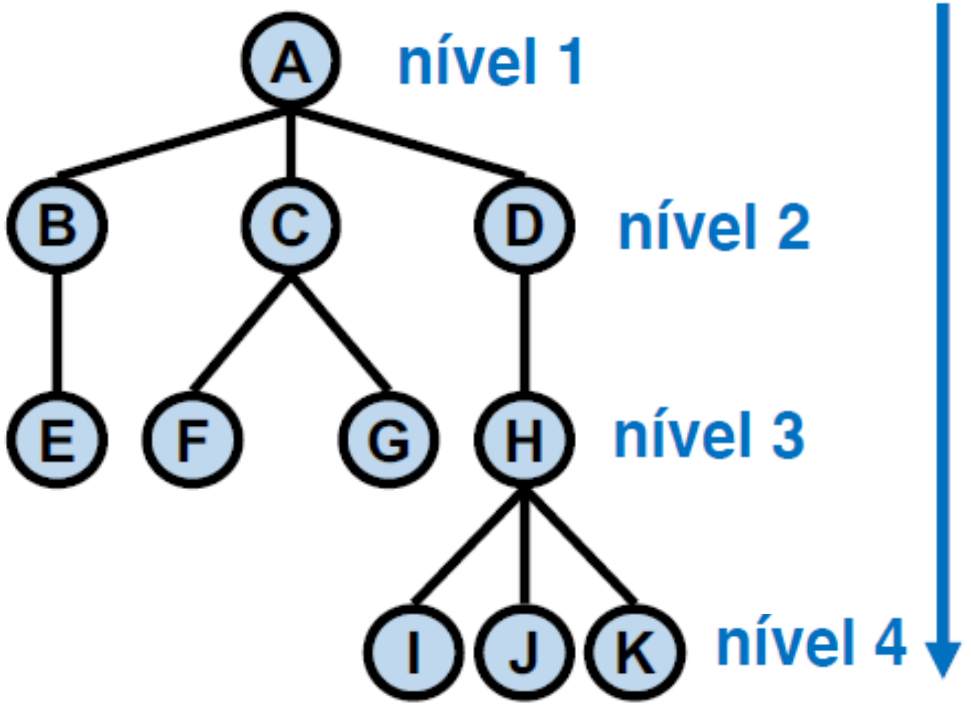


TERMINOLOGIA

Nível

Número de ligações entre a raiz e o nó, acrescido de uma unidade

É igual ao número de nós

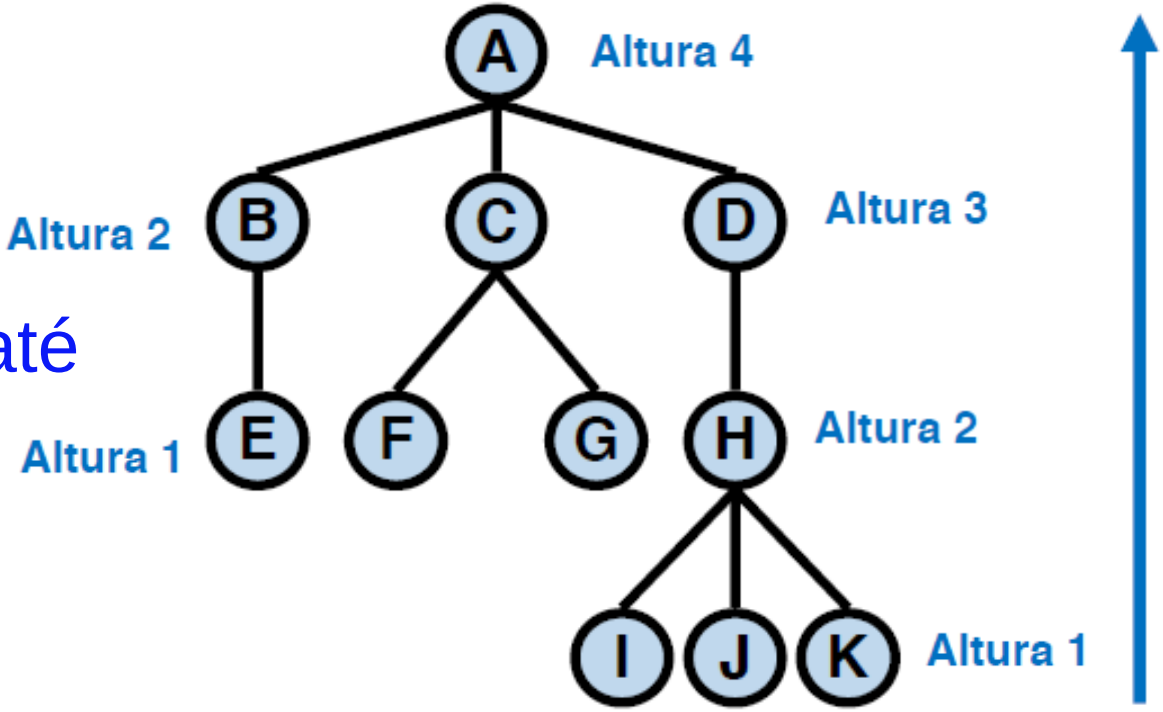


TERMINOLOGIA

Altura de um nó

Altura (profundidade) de um nó:
- número de ligações entre o nó até o nó folha descendente dele

Altura de um nó folha é igual a 1

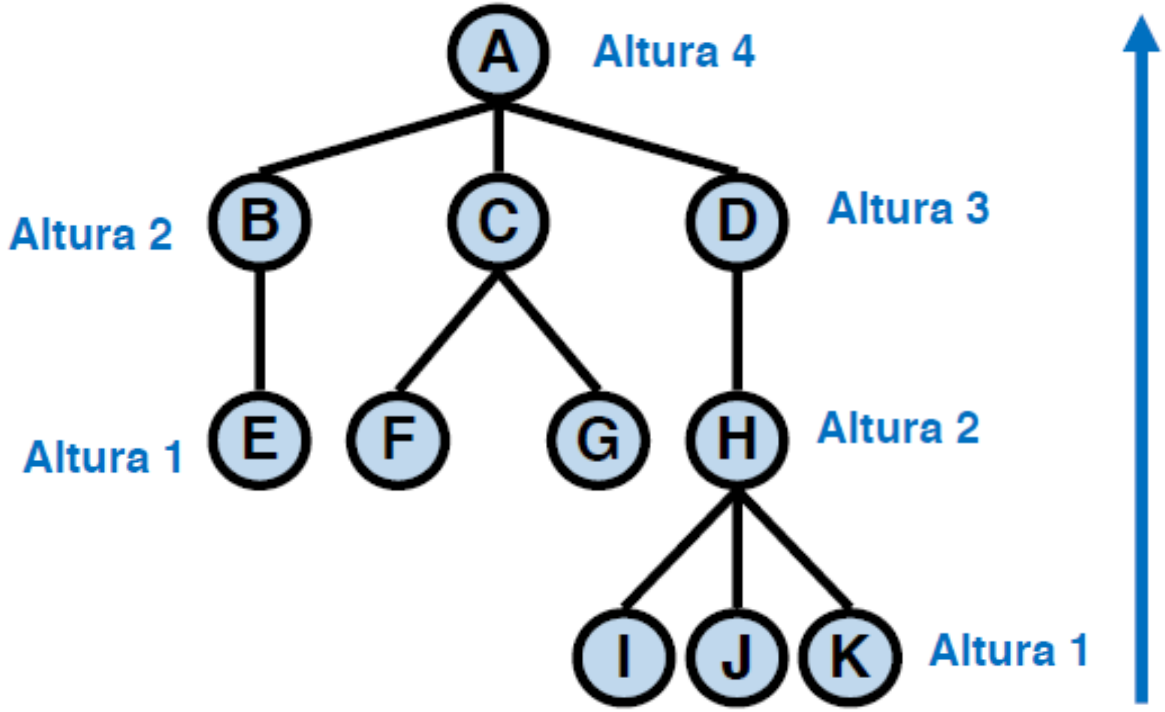


TERMINOLOGIA

Altura da árvore

Altura (profundidade) **da árvore**:
- Maior nível dentre seus nós

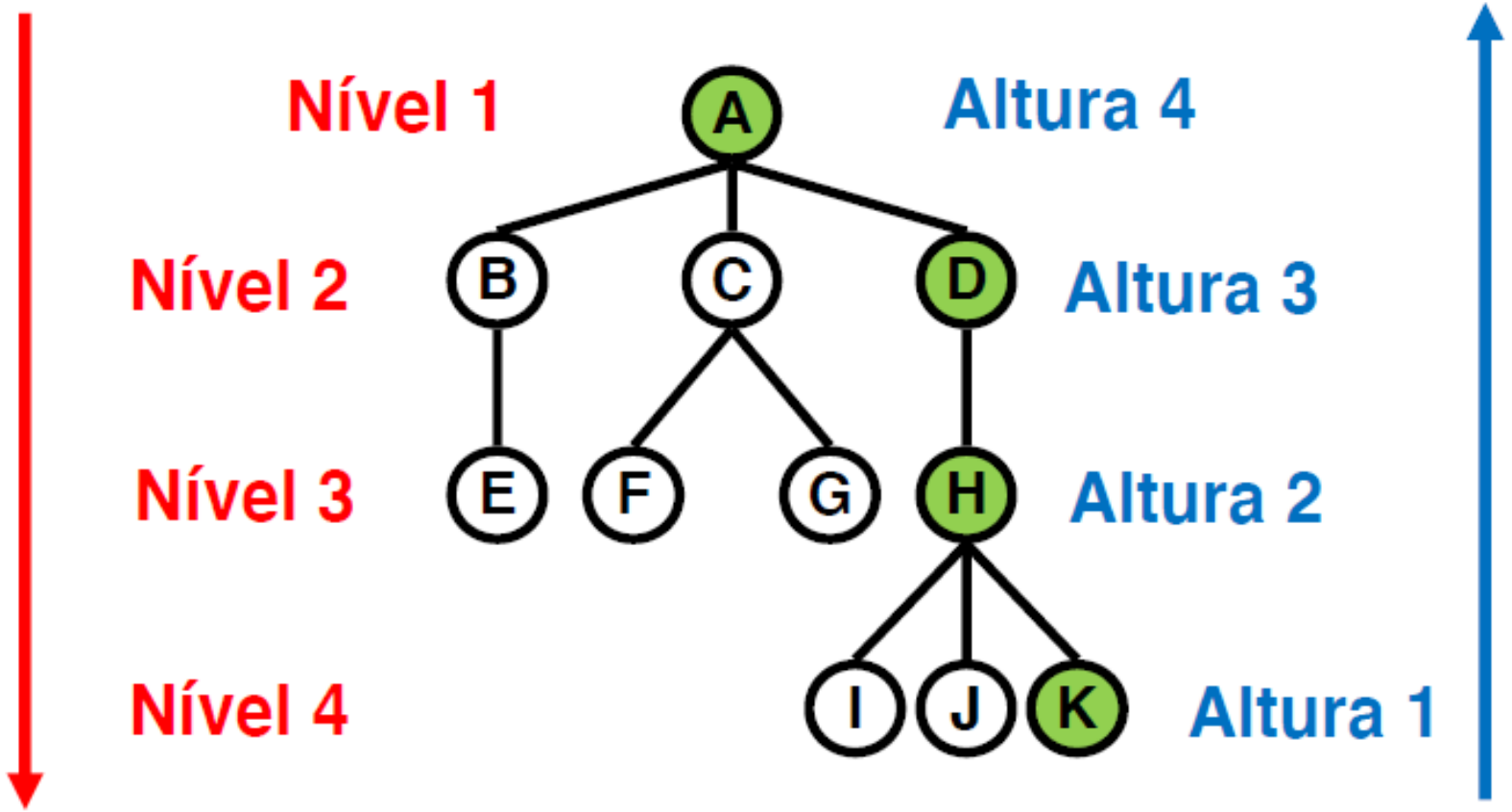
Equivale a altura do nó raiz



Altura da Árvore = 4

TERMINOLOGIA

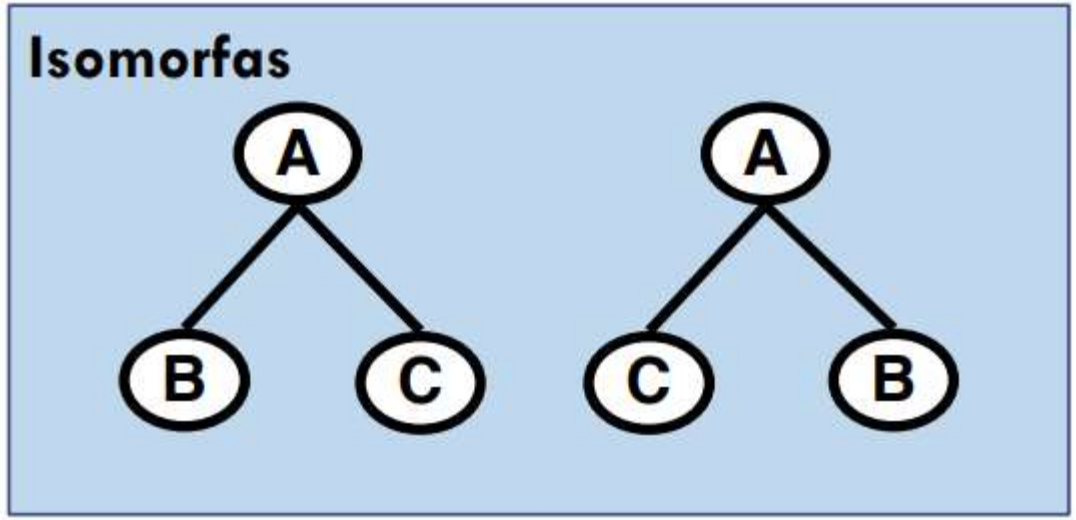
Altura X nível



TERMINOLOGIA

Árvore isomorfa

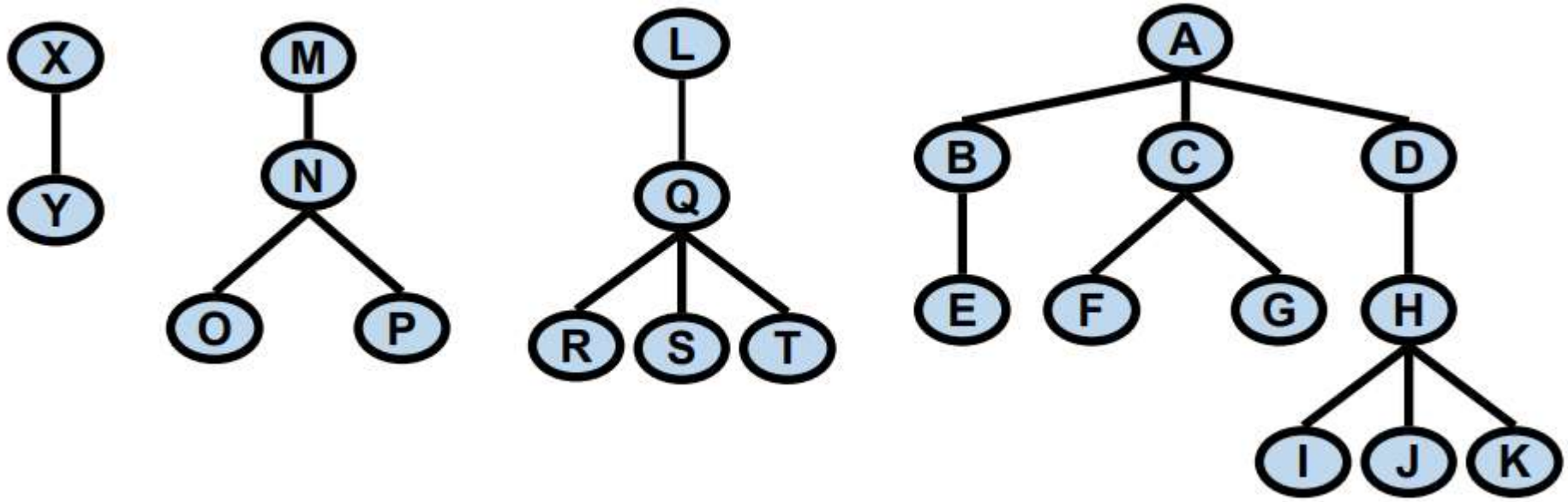
Duas árvores são isomorfas quando puderem se tornar coincidentes pela permutação da ordem das subárvores



TERMINOLOGIA

Floresta

Floresta = Um conjunto de árvores

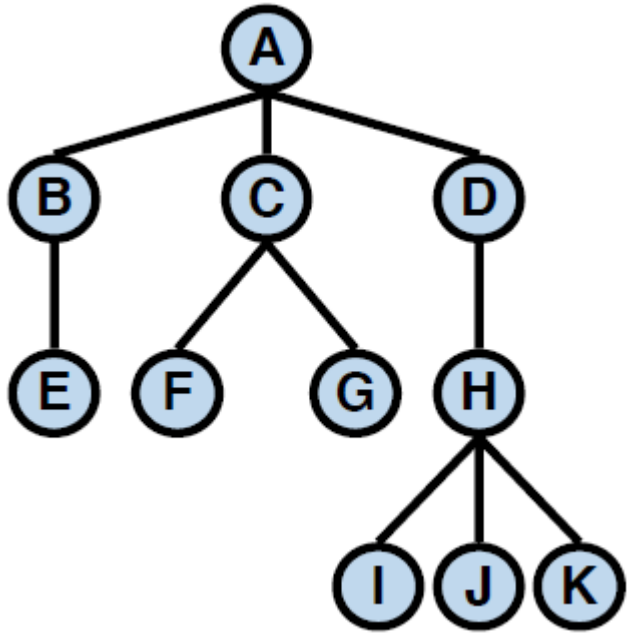


FAMEPI
Faculdade do Amazonas de Ensino, Pesquisa e Inovação
FUNDAÇÃO MATIAS MACHLINE

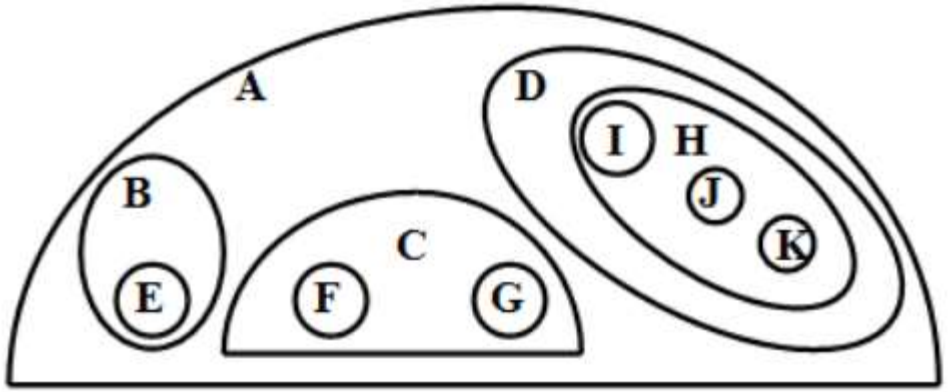
TERMINOLOGIA

Formas de representação

a) Forma convencional



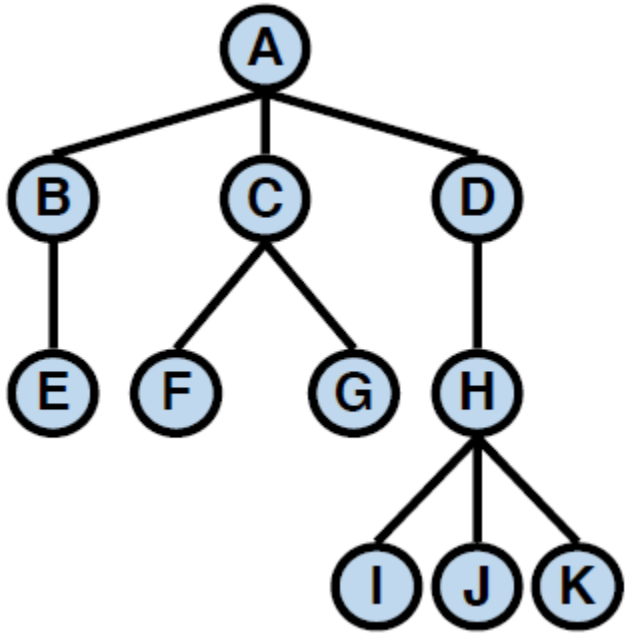
b) Diagrama de conjuntos



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

TERMINOLOGIA

Formas de representação



c) Forma parentética

Coloca-se entre parêntesis a raiz, seguida das formas parentéticas de suas sub-árvores, ordenadas da esquerda para a direita

(A (B (E)) (C (F) (G)) (D (H (I) (J) (K))))

TERMINOLOGIA

Formas de representação

d) Forma tabulada

Diretoria
- Departamento de Fabricação
  - Seção de Prensas
  - Seção de Tornos
  - Seção de Ferramentaria
  - Seção de Fornos
  - Seção de Banhos Químicos
- Departamento de Engenharia de Produção
  - Seção de Desenvolvimento de Projetos
  - Seção de Desenhos
  - Seção de Controle de Qualidade
- Departamento de Manutenção
  - Seção de Eletricistas
  - Seção de Mecânica
  - Seção de Hidráulica
  - Seção de Instalações Prediais

- Cada nó aparece numa linha, e seus filhos são listados com uma tabulação a mais que a desse nó.
  - Ex: organograma de uma empresa, índice de livros, etc.

TERMINOLOGIA

Formas de representação

e) Forma numerada
(ou itemizada)

1 – Estruturas de dados

1.1 – Listas lineares
1.1.1 – Estrutura contígua
1.1.2 – Estrutura encadeada
1.1.3 – Pilhas e filas

1.2 – Árvores
1.2.1 – Definições
1.2.2 – Estruturas de dados
1.2.3 – Árvores binárias

1.3 – Grafos
1.3.1 – Grafos orientados
1.3.2 – Grafos não orientados

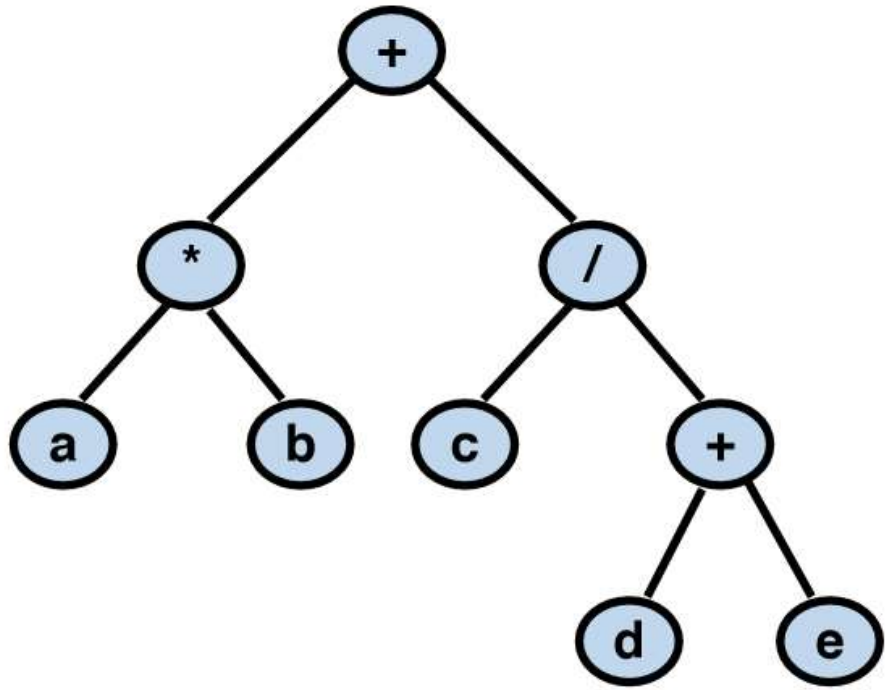
- Semelhante à anterior: a numeração de um nó tem como prefixo o número de seu pai, e como sufixo um número que o diferencie dos irmãos.

ÁRVORES BINÁRIAS

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

ÁRVORES BINÁRIAS

Árvores binárias são uma das árvores mais usadas em computação



Expressão aritmética: (a * b) + (c / (d + e))

ÁRVORES BINÁRIAS

DEFINIÇÃO

Conjunto finito T de zero ou mais nós (nodos), tal que:

Se número de nós é maior do que zero

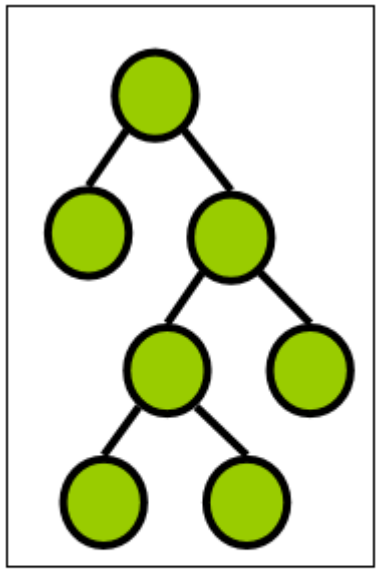
- existe um nó denominado raiz da árvore
- os demais nós formam 2 conjuntos disjuntos S_1, S_2 (subárvore da esquerda e subárvore da direita) onde cada um destes é uma árvore binária

Se número de nós é igual a zero

- árvore vazia

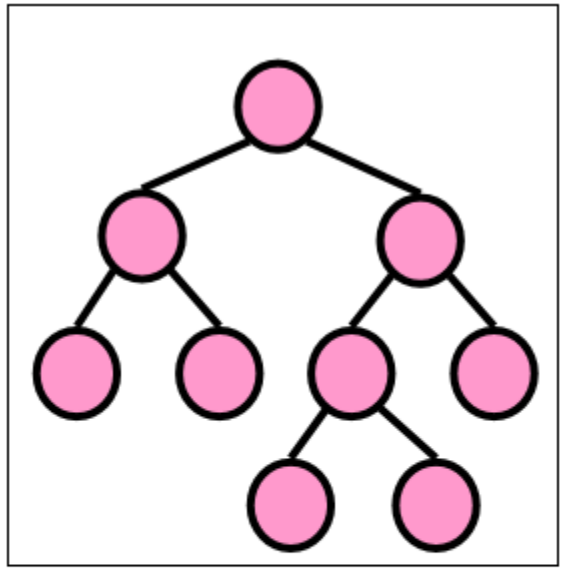
ÁRVORES BINÁRIAS

TIPOS DE ÁRVORES BINÁRIAS



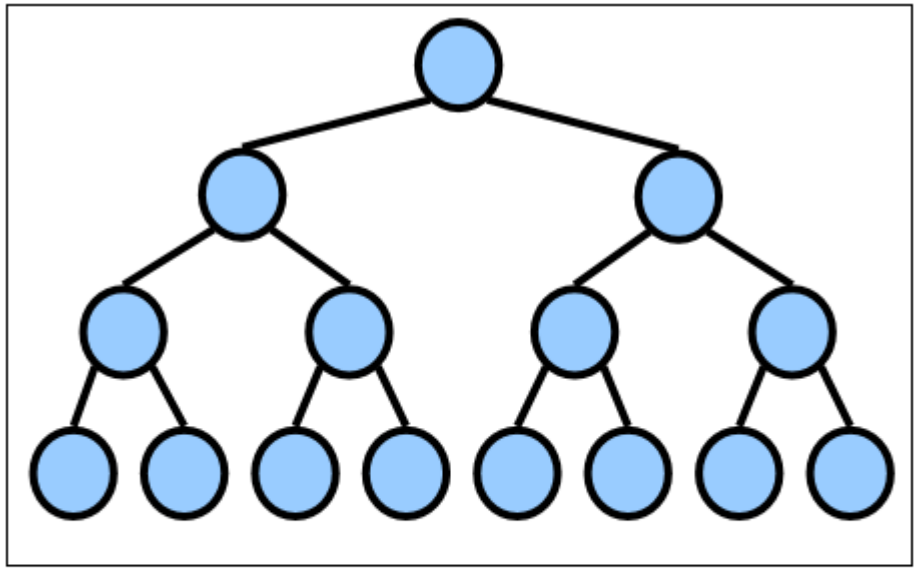
Estritamente Binária

0 ou 2 filhos



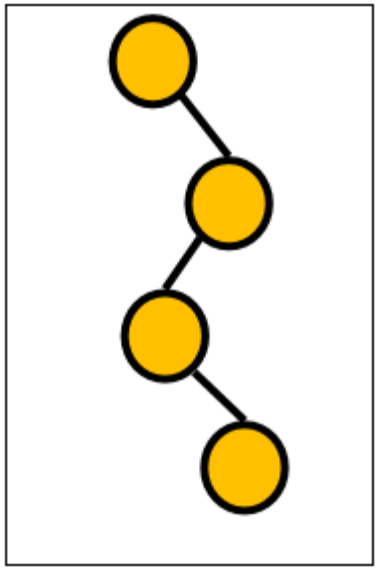
Binária Completa

Sub-árvores vazias apenas no último ou penúltimo nível



Binária Cheia

Sub-árvores vazias somente no último nível



Zigue Zague

Nós internos com 1 subárvore vazia

ÁRVORES BINÁRIAS CAMINHAMENTOS

Um caminhamento (ou percurso) define uma sequência de nós

Cada nó passa a ter um nó seguinte, ou um nó anterior, ou ambos (exceto árvore com 1 só nó)

Sequência de nós depende do caminhamento

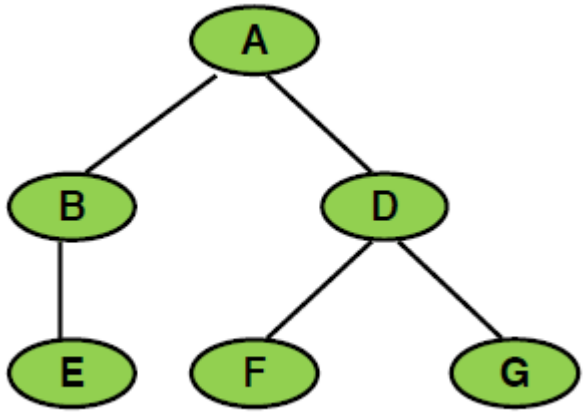
Exemplo:

Caminhamento 1:

- A – B – D – E – F - G

Caminhamento 2:

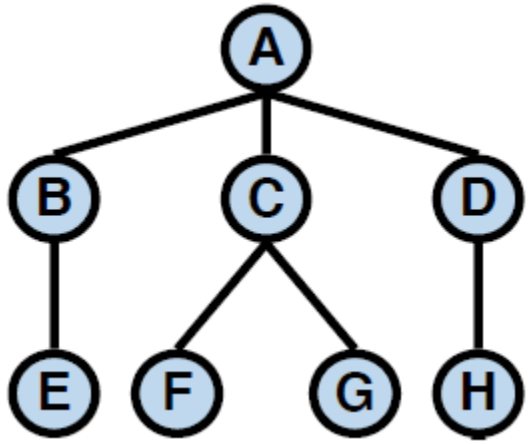
- A – B – E – D- F- G



ÁRVORES BINÁRIAS CAMINHAMENTOS

- Ordenação de todos os nós de uma árvore
  - Existem formas de se ordenar e de se percorrer sistematicamente todos os nós de uma árvore:
    - Ordenação ou percurso por nível — Busca em largura
    - Ordenação ou percurso em pré-ordem
    - Ordenação ou percurso em ordem central (ou in-ordem)
    - Ordenação ou percurso em pós-ordem

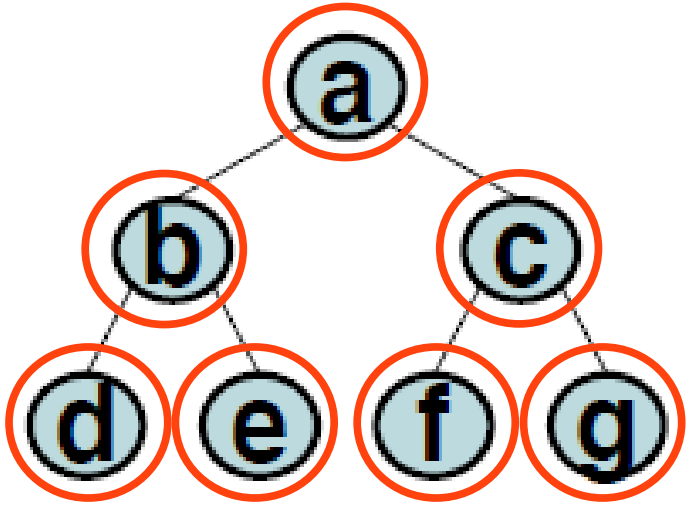
Busca em profundidade



ÁRVORES BINÁRIAS CAMINHAMENTOS

A B C D E F G

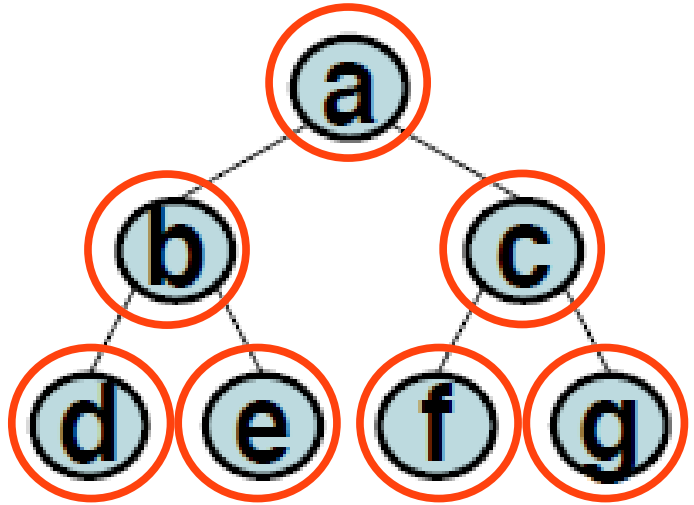
- Ordenação de todos os nós de uma árvore
  - Ordenação ou percurso por nível — Busca em largura
    - Passos:
      - Primeiramente, visita-se a raiz
      - Depois visitam-se todos os filhos da raiz, da esquerda para a direita
      - Depois os descendentes dos filhos da esquerda para a direita, e assim por diante.



ÁRVORES BINÁRIAS CAMINHAMENTOS

A B D E C F G

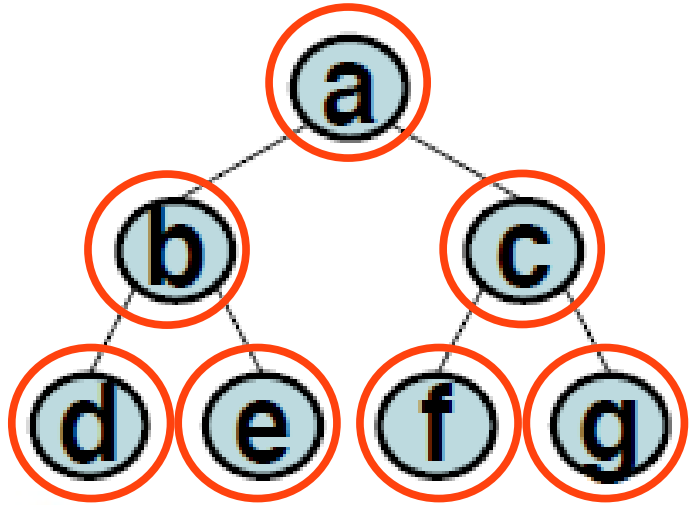
- Ordenação de todos os nós de uma árvore
  - Ordenação ou percurso em pré-ordem
    - Passos:
      1. Vistar a raiz.
      2. Percorrer a sua subárvore esquerda em pré-ordem.
      3. Percorrer a sua subárvore direita em pré-ordem.



ÁRVORES BINÁRIAS CAMINHAMENTOS

D B E A F C G

- Ordenação de todos os nós de uma árvore
  - Ordenação ou percurso em ordem central (ou in-ordem)
    - Passos:



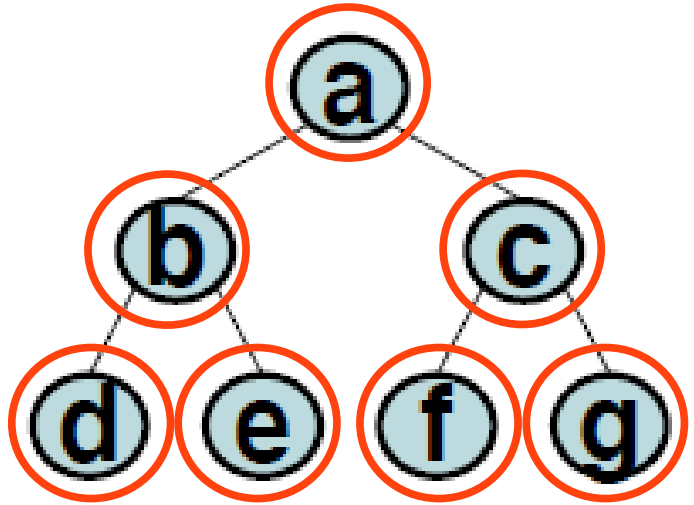
1. Percorrer a sua subárvore esquerda em in-ordem.
2. Vistar a raiz.
3. Percorrer a sua subárvore direita em in-ordem.

É conhecida também pelo nome de ordem simétrica.

D E B F G C A

ÁRVORES BINÁRIAS CAMINHAMENTOS

- Ordenação de todos os nós de uma árvore
  - Ordenação ou percurso em pós-ordem
    - Passos:



1. Percorrer a sua subárvore esquerda em pós-ordem.
2. Percorrer a sua subárvore direita em pós-ordem.
3. Vistar a raiz.

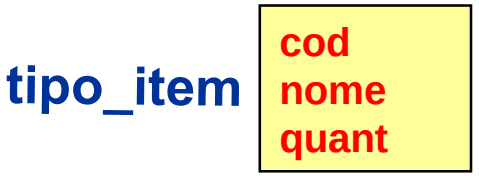
ÁRVORES BINÁRIAS

IMPLEMENTAÇÃO

ÁRVORES BINÁRIAS

- **Definindo as estruturas**

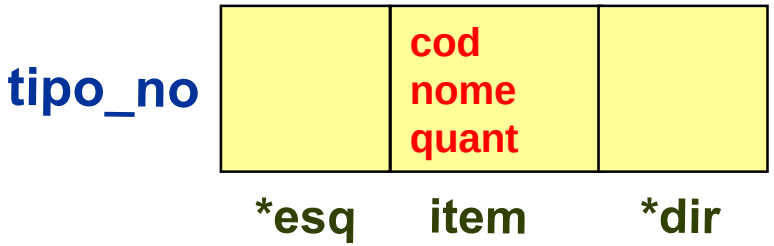
```
typedef struct tipo_item {
int cod;
char nome[30];
int quant;
};
```



ÁRVORES BINÁRIAS

- **Definindo as estruturas**

```
typedef struct tipo_no {
    tipo_item item;
tipo_no *esq;
tipo_no *dir;
};
```



ÁRVORES BINÁRIAS

Operações

inserir – Insere um item na árvore

mostrar – Mostra o conteúdo ordenado da árvore

pesquisar – Pesquisa um item na árvore

excluir – Remove um item da árvore

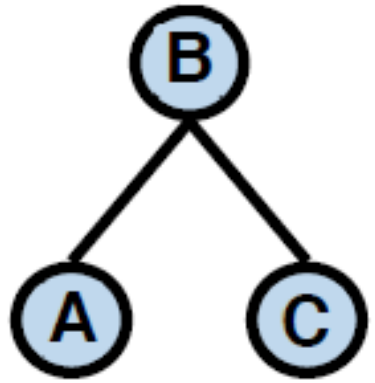
mostrarItem – mostra um item

Regra da árvore binária de busca

Cada nó de referência tem palavras chaves (cod) nas seguintes condições:

= Menores à esquerda da raiz

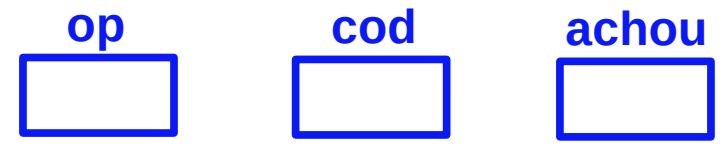
= Maiores à direita da raiz



Código enviado com o slide

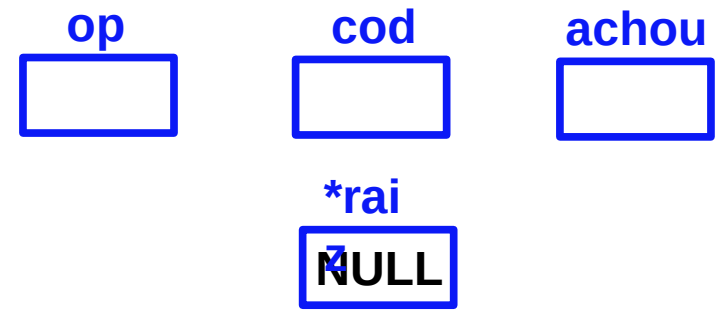
```c
int main( ) {
    setlocale(LC_ALL, "");

    int op, cod, achou;

    tipo_no *raiz = NULL;

    tipo_item aux;

    do{
        system("cls");

        op = menu();

        switch(op){

            case 0: printf("\n\n\n\t Fim do Programa. \n\n");
                    getch();
                    return(0);

            case 1: aux = preencher_item();
                    if(!raiz) raiz = inserir(aux, raiz, raiz);
                    else inserir(aux, raiz, raiz);
                    break;

            case 2: printf("\n\n CONTEÚDO DA ÁRVORE: ");
                    mostrar(raiz);
                    getch(); // segura a tela
                    break;

            case 3: printf("\n\n PESQUISA NA ÁRVORE ");
                    printf("\n\n Digite o código do item a pesquisar: ");
                    scanf("%d", &cod);
                    achou = 0;
                    pesquisar(&achou, cod, raiz);
                    if(!achou) printf("\n\n O item de código %d não está na árvore", cod);
                    getch(); // segura a tela
                    break;

            case 4: printf("\n\n EXCLUIR DA ÁRVORE ");
                    printf("\n\n Digite o código do item a excluir: ");
                    scanf("%d", &cod);
                    achou = 0;
                    raiz = excluir(&achou, cod, raiz);
                    if(!achou) printf("\n\n O item de código %d não es
                    else printf("\n\n O item de código %d foi excluído
                    getch(); // segura a tela
                    break;

            default:
                    printf("\n\n OPÇÃO INVÁLIDA !!!");
                    getch();
            }// fim switch
        }while(1);//fim do-while
        return(0);
    }// fim main
```

ÁRVORES BINÁRIAS




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

    int op, cod, achou;

    tipo_no *raiz = NULL;

    tipo_item aux;

    do{
        system("cls");

        op = menu();

        switch(op){

            case 0: printf("\n\n\n\t Fim do Programa. \n\n");
                    getch();
                    return(0);

            case 1: aux = preencher_item();
                    if(!raiz) raiz = inserir(aux, raiz, raiz);
                    else inserir(aux, raiz, raiz);
                    break;

            case 2: printf("\n\n CONTEÚDO DA ÁRVORE: ");
                    mostrar(raiz);
                    getch(); // segura a tela
                    break;

            case 3: printf("\n\n PESQUISA NA ÁRVORE ");
                    printf("\n\n Digite o código do item a pesquisar: ");
                    scanf("%d", &cod);
                    achou = 0;
                    pesquisar(&achou, cod, raiz);
                    if(!achou) printf("\n\n O item de código %d não está na árvore", c
                    getch(); // segura a tela
                    break;
```

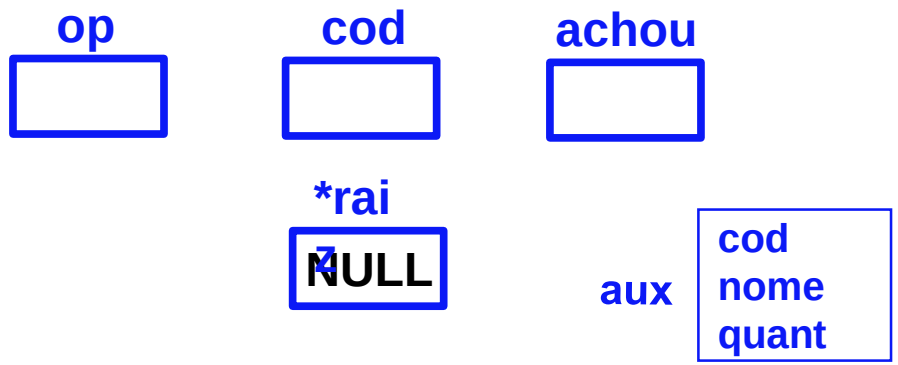
ÁRVORES BINÁRIAS




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

    int op, cod, achou;

    tipo_no *raiz = NULL;

    tipo_item aux;

    do{
        system("cls");

        op = menu();

        switch(op){

            case 0: printf("\n\n\n\t Fim do Programa. \n\n");
                    getch();
                    return(0);

            case 1: aux = preencher_item();
                    if(!raiz) raiz = inserir(aux, raiz, raiz);
                    else inserir(aux, raiz, raiz);
                    break;

            case 2: printf("\n\n CONTEÚDO DA ÁRVORE: ");
                    mostrar(raiz);
                    getch(); // segura a tela
                    break;

            case 3: printf("\n\n PESQUISA NA ÁRVORE ");
                    printf("\n\n Digite o código do item a pesquisar: ");
                    scanf("%d", &cod);
                    achou = 0;
                    pesquisar(&achou, cod, raiz);
                    if(!achou) printf("\n\n O item de código %d não está na árvore", c
                    getch(); // segura a tela
                    break;
```

ÁRVORES BINÁRIAS

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

    int op, cod, achou;

    tipo_no *raiz = NULL;

    tipo_item aux;

    do{
        system("cls");

        op = menu();

        switch(op){

            case 0: printf("\n\n\n\t Fim do Programa. \n\n");
                    getch();
                    return(0);

            case 1: aux = preencher_item();
                    if(!raiz) raiz = inserir(aux, raiz, raiz);
                    else inserir(aux, raiz, raiz);
                    break;

            case 2: printf("\n\n CONTEÚDO DA ÁRVORE: ");
                    mostrar(raiz);
                    getch(); // segura a tela
                    break;

            case 3: printf("\n\n PESQUISA NA ÁRVORE ");
                    printf("\n\n Digite o código do item a pesquisar: ");
                    scanf("%d", &cod);
                    achou = 0;
                    pesquisar(&achou, cod, raiz);
                    if(!achou) printf("\n\n O item de código %d não está na árvore", c
                    getch(); // segura a tela
                    break;
```



ÁRVORES BINÁRIAS

IMPLEMENTAÇÃO

INSERIR NA ÁRVORE

ÁRVORES BINÁRIAS

Procedimento para Inserir na Árvore

- Onde inserir?
  - Atingir um ponteiro nulo em um processo de pesquisa significa uma pesquisa sem sucesso.
  - O ponteiro nulo atingido é o ponto de inserção.

- Como inserir?
  1. Cria registro.
  2. Procurar o lugar na árvore.
  3. Se registro não tiver na árvore, insere-o.

ÁRVORES BINÁRIAS

Procedimento para inserir na árvore

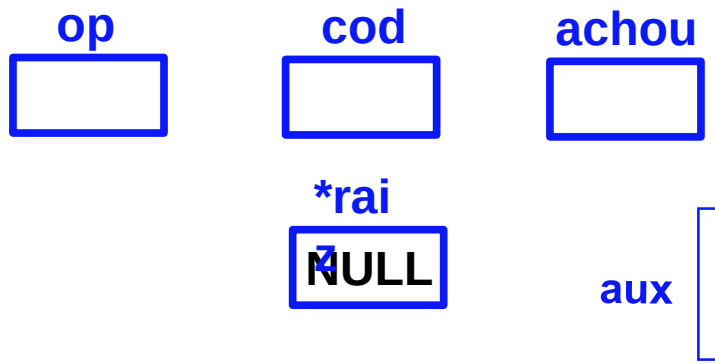
- Onde inserir?
  - Atingir um ponteiro nulo em um processo de pesquisa significa uma pesquisa sem sucesso.
  - O ponteiro nulo atingido é o ponto de inserção.

- Como inserir?
  1. Cria registro.
  2. Procurar o lugar na árvore.
  3. Se registro não tiver na árvore, insere-o.

ÁRVORES BINÁRIAS

```c
int menu(){
    int op;

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

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

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

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

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

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

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

    return(op);
}
```

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

    int op, cod, achou;

    tipo_no *raiz = NULL;

    tipo_item aux;

do{
    system("cls");

    op = menu();

    switch(op){

        case 0: printf("\n\n\n\t Fim do Programa. \n\n");
                getch();
                return(0);

        case 1: aux = preencher_item();
                if(!raiz) raiz = inserir(aux, raiz, raiz);
                else inserir(aux, raiz, raiz);
                break;

        case 2: printf("\n\n CONTEÚDO DA ÁRVORE: ");
                mostrar(raiz);
                getch(); // segura a tela
                break;

        case 3: printf("\n\n PESQUISA NA ÁRVORE ");
                printf("\n\n Digite o código do item a pesquisar: ");
                scanf("%d", &cod);
                achou = 0;
                pesquisar(&achou, cod, raiz);
                if(!achou) printf("\n\n O item de código %d não está na árvore", c
                getch(); // segura a tela
                break;
```




```
********** MENU ******
[1] - INSERIR
[2] - MOSTRAR
[3] - PESQUISAR
[4] - EXCLUIR
[0] - SAIR

Digite sua opção: 1
```

ÁRVORES BINÁRIAS

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

    int op, cod, achou;

    tipo_no *raiz = NULL;

    tipo_item aux;

    do{
        system("cls");

        op = menu();

        switch(op){

            case 0: printf("\n\n\n\t Fim do Programa. \n\n");
                    getch();
                    return(0);

            case 1: aux = preencher_item();
                    if(!raiz) raiz = inserir(aux, raiz, raiz);
                    else inserir(aux, raiz, raiz);
                    break;

            case 2: printf("\n\n CONTEÚDO DA ÁRVORE: ");
                    mostrar(raiz);
                    getch(); // segura a tela
                    break;

            case 3: printf("\n\n PESQUISA NA ÁRVORE ");
                    printf("\n\n Digite o código do item a pesquisar: ");
                    scanf("%d", &cod);
                    achou = 0;
                    pesquisar(&achou, cod, raiz);
                    if(!achou) printf("\n\n O item de código %d não está na árvore", c
                    getch(); // segura a tela
                    break;
```

```c
tipo_item preencher_item(){

    tipo_item item;
    printf("\n\n ***** PREENCHENDO ITEM A SER INSERIDO *****");

    printf("\n\n Digite o código: ");
    scanf("%d", &item.cod);

    fflush(stdin); // limpar o buffer do teclado
    printf("\n\n Digite o nome do item: ");
    gets(item.nome);

    printf("\n\n Digite a quantidade: ");
    scanf("%d", &item.quant);

    return(item);
}
```

op: 1

cod:

achou:

*rai: NULL

aux: cod, nome, quant

```
***** PREENCHENDO ITEM A SER INSERIDO *****

Digite o código: 5

Digite o nome do item: Arroz

Digite a quantidade: 500
```

INÁRIAS

Item: cod, nome, quant

```c
tipo_item preencher_item(){

    tipo_item item;
    printf("\n\n ***** PREENCHENDO ITEM A SER INSERIDO *****");

    printf("\n\n Digite o código: ");
    scanf("%d", &item.cod);

    fflush(stdin); // limpar o buffer do teclado
    printf("\n\n Digite o nome do item: ");
    gets(item.nome);

    printf("\n\n Digite a quantidade: ");
    scanf("%d", &item.quant);

    return(item);
}
```

op: 1

cod

achou

*rai: NULL

aux: cod, nome, quant

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

    int op, cod, achou;

    tipo_no *raiz = NULL;

    tipo_item aux;

    do{
        system("cls");

        op = menu();

        switch(op){

            case 0: printf("\n\n\n\t Fim do Programa. \n\n");
                    getch();
                    return(0);

            case 1: aux = preencher_item();
                    if(!raiz) raiz = inserir(aux, raiz, raiz);
                    else inserir(aux, raiz, raiz);
                    break;

            case 2: printf("\n\n CONTEÚDO DA ÁRVORE: ");
                    mostrar(raiz);
                    getch(); // segura a tela
                    break;

            case 3: printf("\n\n PESQUISA NA ÁRVORE ");
                    printf("\n\n Digite o código do item a pesquisar: ");
                    scanf("%d", &cod);
                    achou = 0;
                    pesquisar(&achou, cod, raiz);
                    if(!achou) printf("\n\n O item de código %d não está na árvore", c
                    getch(); // segura a tela
                    break;
```

ÁRVORES BINÁRIAS

Item | 5 Arroz 500

op: 1 | cod: | achou:

*rai: NULL

aux: cod, nome, quant

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

    int op, cod, achou;

    tipo_no *raiz = NULL;

    tipo_item aux;

    do{
        system("cls");

        op = menu();

        switch(op){

            case 0: printf("\n\n\n\t Fim do Programa. \n\n");
                    getch();
                    return(0);

            case 1: aux = preencher_item();
                    if(!raiz) raiz = inserir(aux, raiz, raiz);
                    else inserir(aux, raiz, raiz);
                    break;

            case 2: printf("\n\n CONTEÚDO DA ÁRVORE: ");
                    mostrar(raiz);
                    getch(); // segura a tela
                    break;

            case 3: printf("\n\n PESQUISA NA ÁRVORE ");
                    printf("\n\n Digite o código do item a pesquisar: ");
                    scanf("%d", &cod);
                    achou = 0;
                    pesquisar(&achou, cod, raiz);
                    if(!achou) printf("\n\n O item de código %d não está na árvore", c
                    getch(); // segura a tela
                    break;
```

```c
tipo_item preencher_item(){

    tipo_item item;
    printf("\n\n ***** PREENCHENDO ITEM A SER INSERIDO *****");

    printf("\n\n Digite o código: ");
    scanf("%d", &item.cod);

    fflush(stdin); // limpar o buffer do teclado
    printf("\n\n Digite o nome do item: ");
    gets(item.nome);

    printf("\n\n Digite a quantidade: ");
    scanf("%d", &item.quant);

    return(item);
}
```

ÁRVORES BINÁRIAS

Item | 5 Arroz 500

op: 1 | cod: | achou:

*rai: NULL

aux: 5 Arroz 500

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

    int op, cod, achou;

    tipo_no *raiz = NULL;

    tipo_item aux;

    do{
        system("cls");

        op = menu();

        switch(op){

            case 0: printf("\n\n\n\t Fim do Programa. \n\n");
                    getch();
                    return(0);

            case 1: aux = preencher_item();
                    if(!raiz) raiz = inserir(aux, raiz, raiz);
                    else inserir(aux, raiz, raiz);
                    break;

            case 2: printf("\n\n CONTEÚDO DA ÁRVORE: ");
                    mostrar(raiz);
                    getch(); // segura a tela
                    break;

            case 3: printf("\n\n PESQUISA NA ÁRVORE ");
                    printf("\n\n Digite o código do item a pesquisar: ");
                    scanf("%d", &cod);
                    achou = 0;
                    pesquisar(&achou, cod, raiz);
                    if(!achou) printf("\n\n O item de código %d não está na árvore", c
                    getch(); // segura a tela
                    break;
```

```c
tipo_item preencher_item(){

    tipo_item item;
    printf("\n\n ***** PREENCHENDO ITEM A SER INSERIDO *****");

    printf("\n\n Digite o código: ");
    scanf("%d", &item.cod);

    fflush(stdin); // limpar o buffer do teclado
    printf("\n\n Digite o nome do item: ");
    gets(item.nome);

    printf("\n\n Digite a quantidade: ");
    scanf("%d", &item.quant);

    return(item);
}
```

ÁRVORES BINÁRIAS

```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```

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

    int op, cod, achou;

    tipo_no *raiz = NULL;

    tipo_item aux;

    do{
        system("cls");

        op = menu();

        switch(op){

            case 0: printf("\n\n\n\t Fim do Programa. \n\n");
                    getch();
                    return(0);

            case 1: aux = preencher_item();
                    if(!raiz) raiz = inserir(aux, raiz, raiz);
                    else inserir(aux, raiz, raiz);
                    break;

            case 2: printf("\n\n CONTEÚDO DA ÁRVORE: ");
                    mostrar(raiz);
                    getch(); // segura a tela
                    break;

            case 3: printf("\n\n PESQUISA NA ÁRVORE ");
                    printf("\n\n Digite o código do item a pesquisar: ");
                    scanf("%d", &cod);
                    achou = 0;
                    pesquisar(&achou, cod, raiz);
                    if(!achou) printf("\n\n O item de código %d não está na árvore", c
                    getch(); // segura a tela
                    break;
```

op: 1

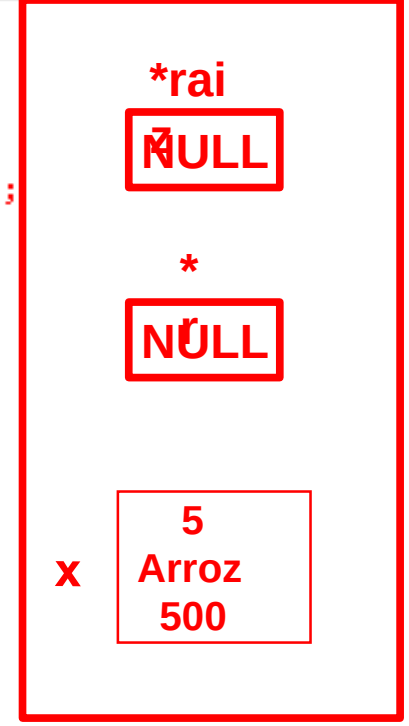
cod:

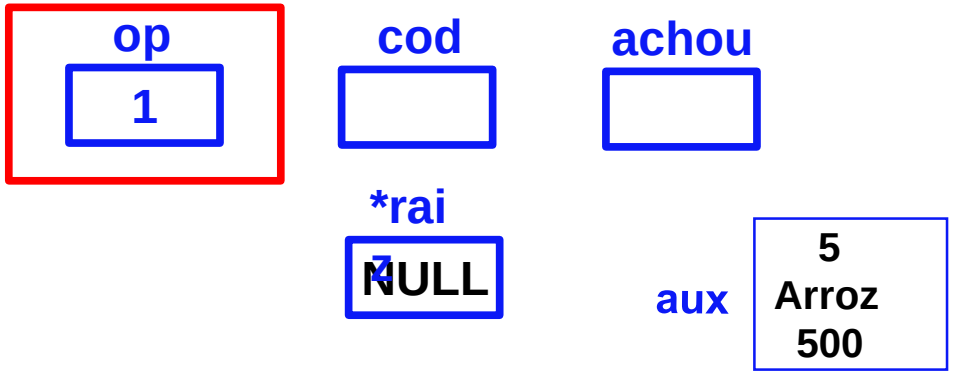
achou:

*rai: NULL

aux: 5, Arroz, 500

```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```






```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

```c
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){
    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```

*rai
NULL

*
NULL

x
5
Arroz
500

op
1

cod

achou

*rai
NULL

aux
5
Arroz
500

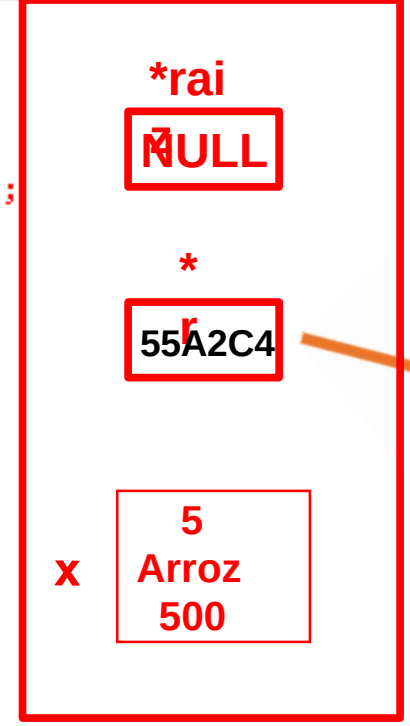
*esq item *dir

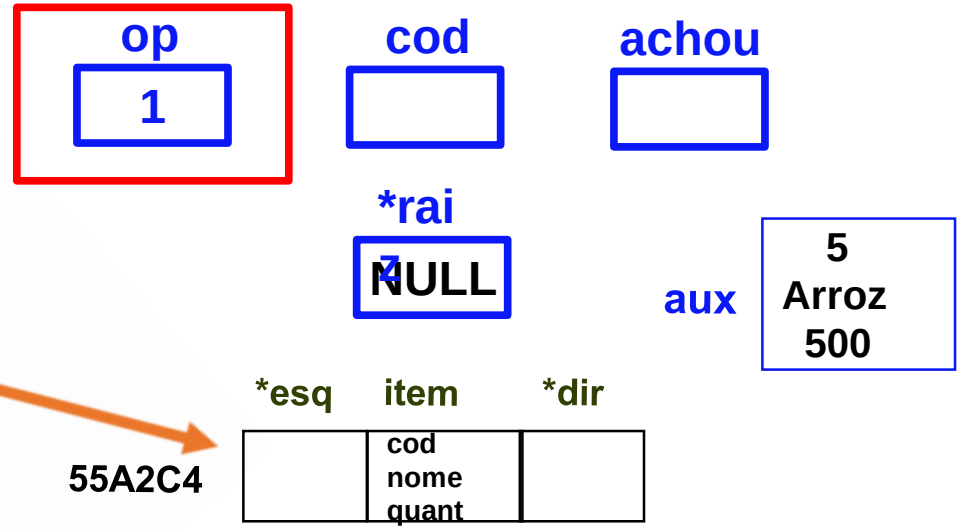
55A2C4

cod
nome
quant

```c
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

```c
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){
    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```






```c
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```

*rai
NULL

*
r
55A2C4

x
5
Arroz
500

op
1

cod

achou

*rai
NULL

aux
5
Arroz
500

*esq item *dir

55A2C4

cod
nome
quant

```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```

*rai
NULL

*
r
55A2C4

x
5
Arroz
500

op
1

cod

achou

*rai
NULL

aux
5
Arroz
500

*esq	item	*dir
NULL	5 Arroz 500	NULL

55A2C4

```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

```c
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```

*rai
NULL

*
r
55A2C4

x
5
Arroz
500

op
1

cod

achou

*rai
NULL

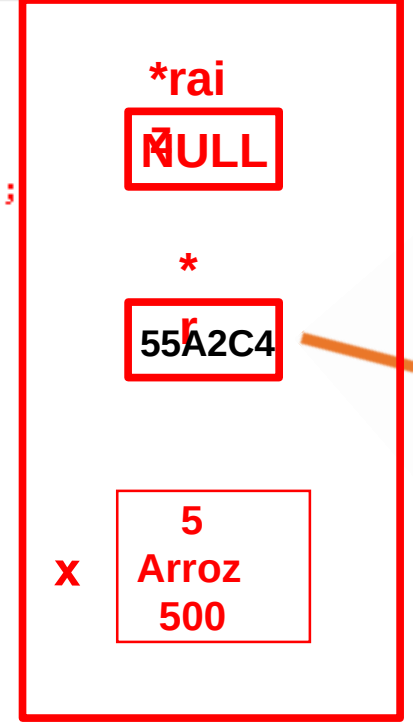
aux
5
Arroz
500

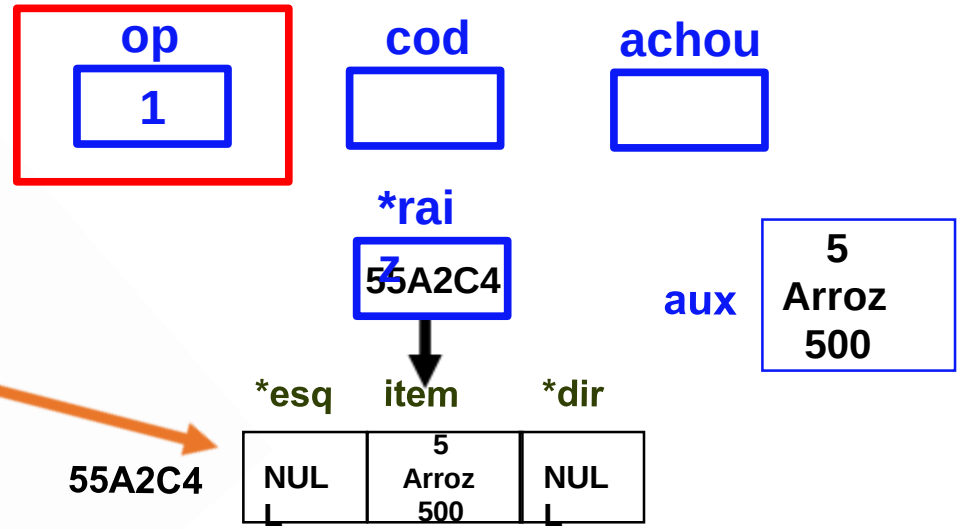
	*esq	item	*dir
55A2C4	NULL	5 Arroz 500	NULL

```
Item inserido com sucesso!!!
```

```c
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

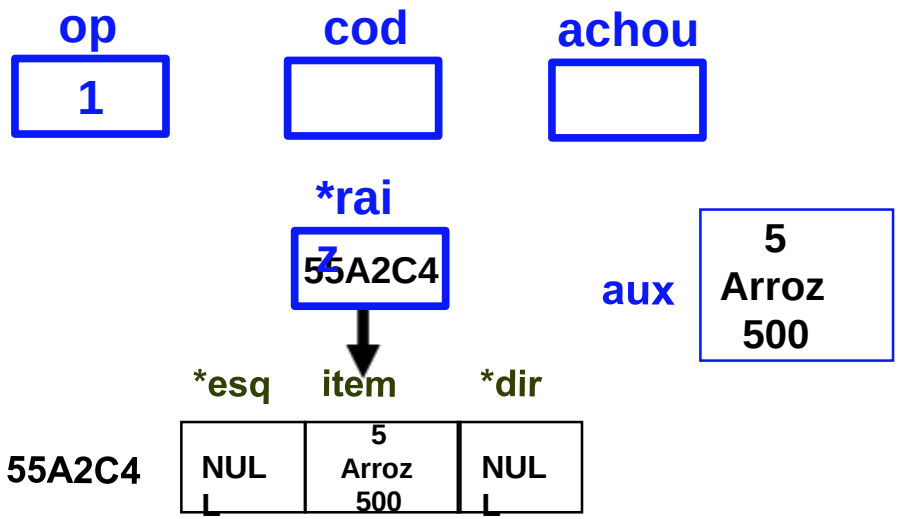
```c
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```



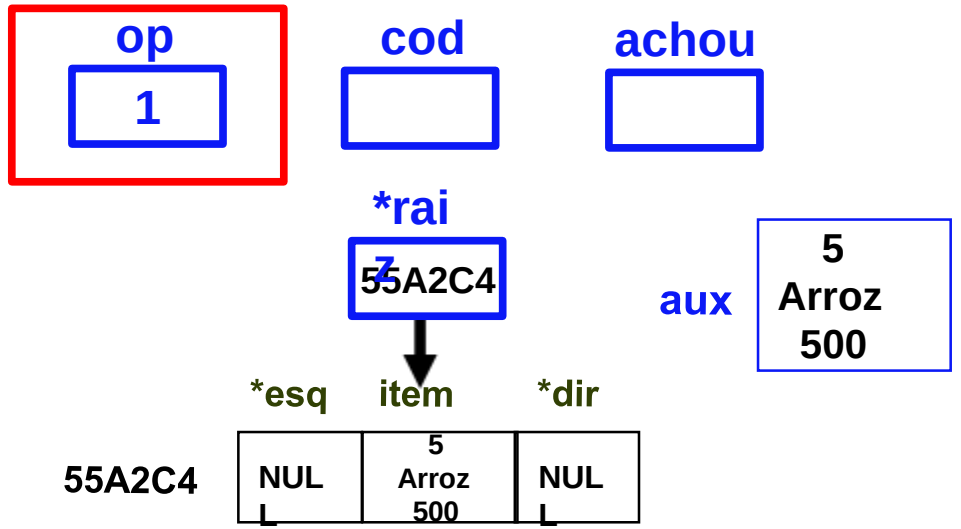



```c
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

ÁRVORES BINÁRIAS



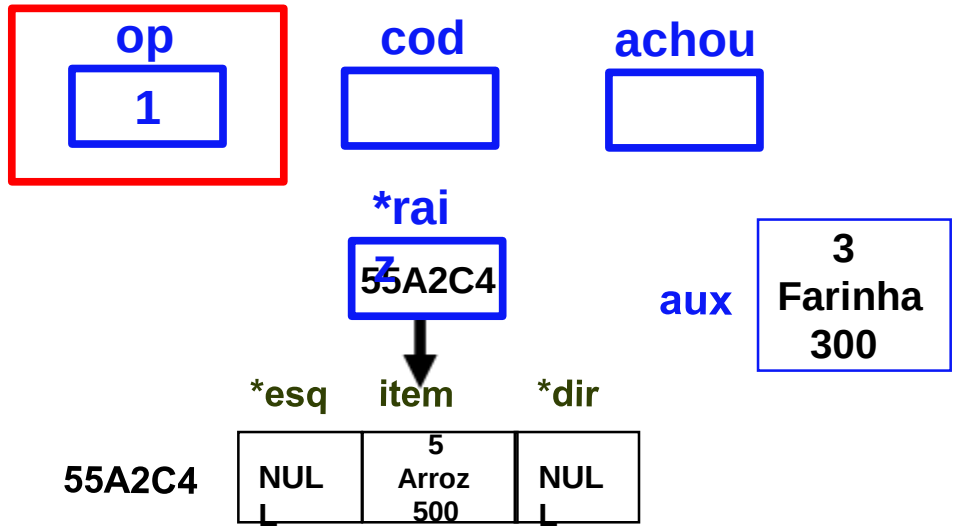
ÁRVORES BINÁRIAS




```
********* MENU *********
[1] - INSERIR
[2] - MOSTRAR
[3] - PESQUISAR
[4] - EXCLUIR
[0] - SAIR

Digite sua opção: 1
```

```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

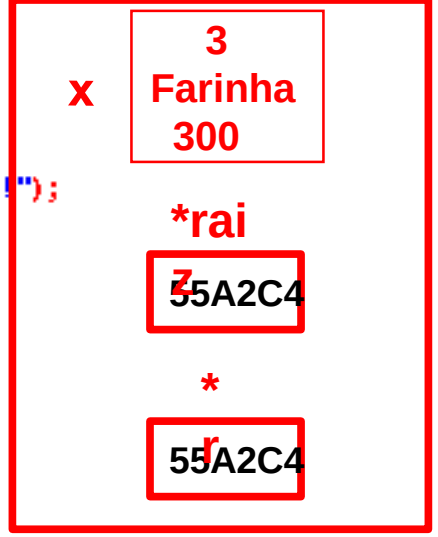
ÁRVORES BINÁRIAS

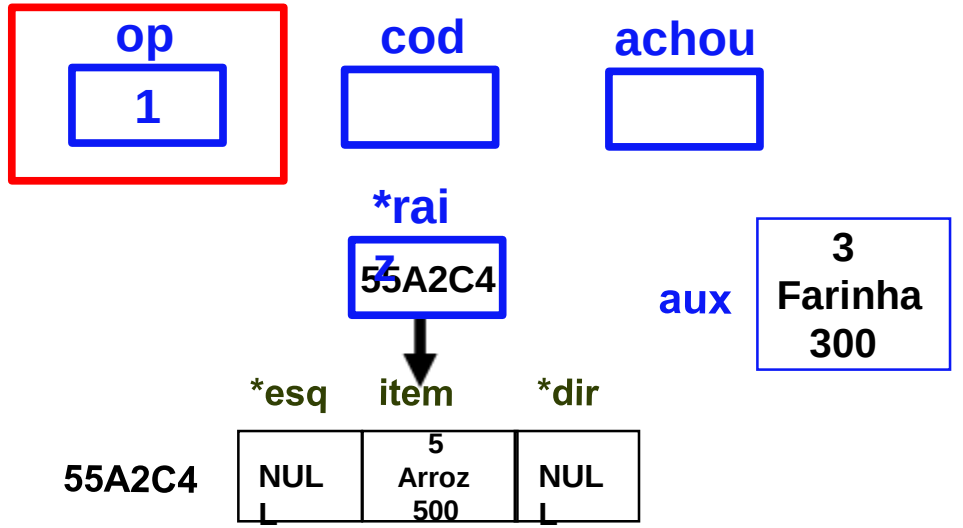



```
********* MENU *********
[1] - INSERIR
[2] - MOSTRAR
[3] - PESQUISAR
[4] - EXCLUIR
[0] - SAIR

Digite sua opção: 1
```

```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

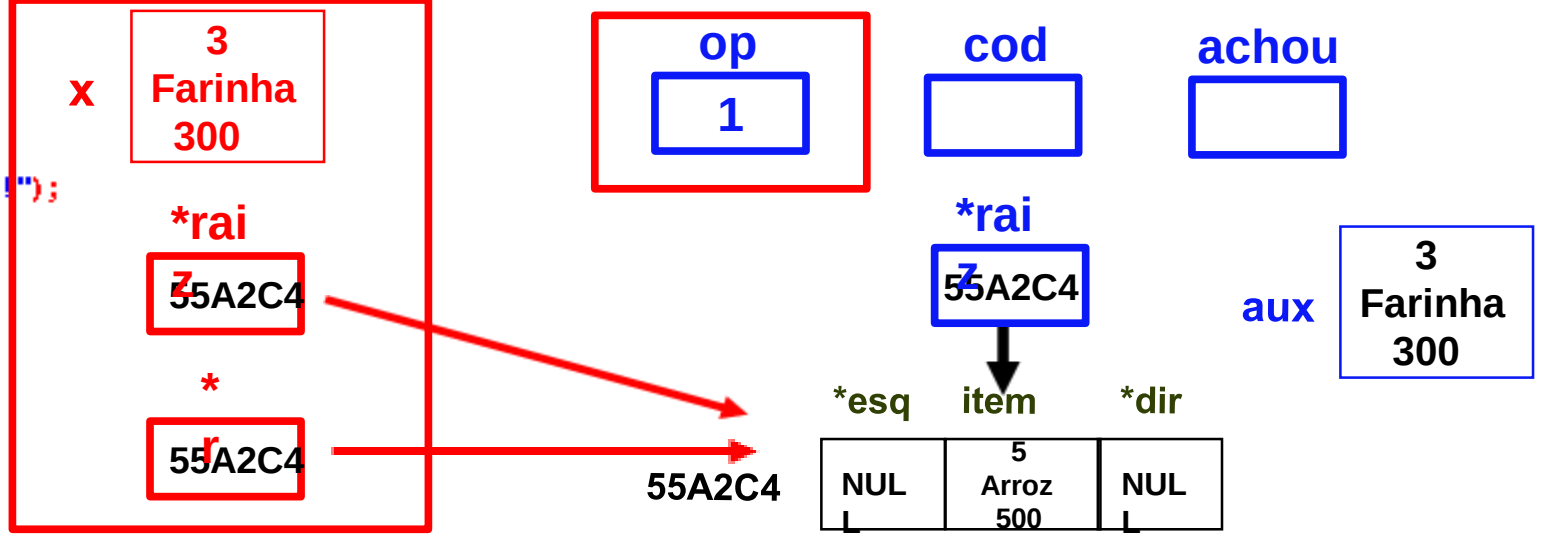
```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```





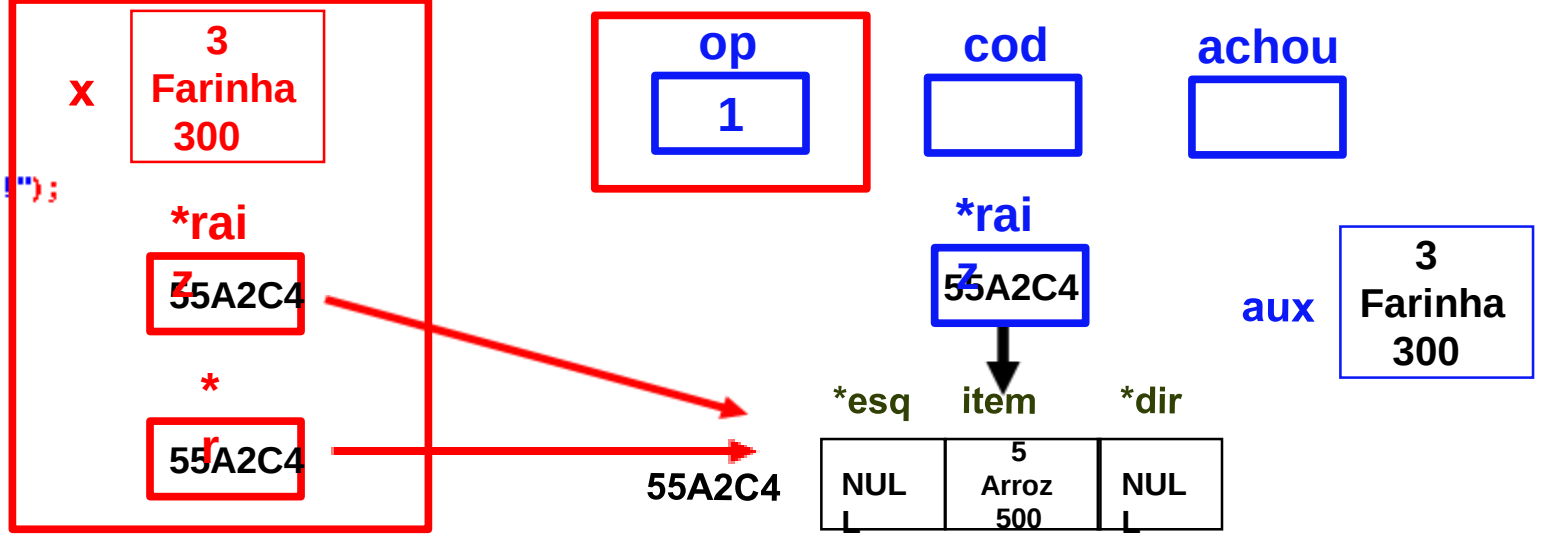

```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```




```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```




```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```

x: 3 Farinha 300

*rai: 55A2C4

*: 55A2C4

op: 1

cod

achou

*rai: 55A2C4

aux: 3 Farinha 300

*esq	item	*dir
NULL	5 Arroz 500	NULL

55A2C4

*rai: 55A2C4

*: NULL

x: 3 Farinha 300

```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){
    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```

x → 3 Farinha 300

*rai → 55A2C4

* r → 55A2C4

op = 1

cod

achou

*rai → 55A2C4

aux → 3 Farinha 300

*esq	item	*dir
NULL	5 Arroz 500	NULL

55A2C4

*rai → 55A2C4

* r → NULL

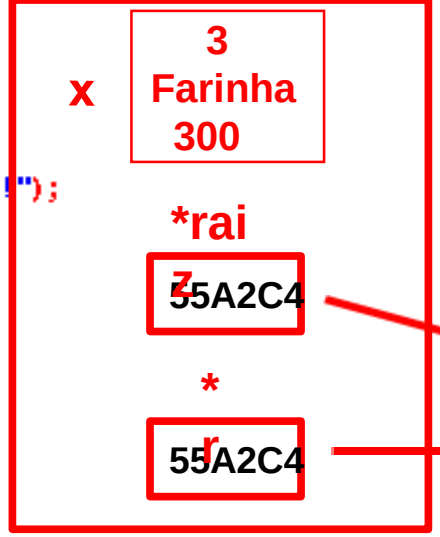
x → 3 Farinha 300

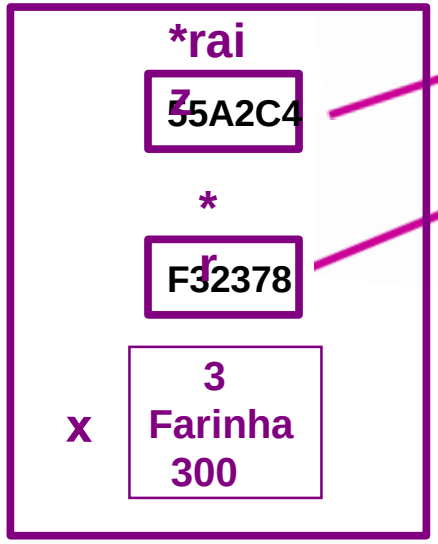
F32378

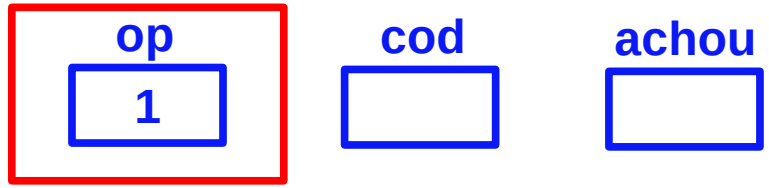
*esq	item	*dir
	cod nome quant	

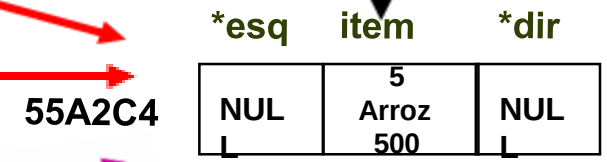
```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

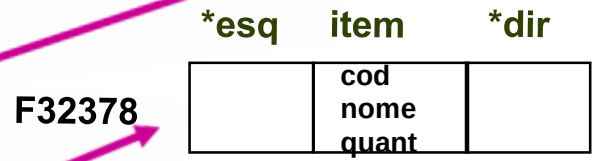
```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){
  if(!r){
      r = (tipo_no *) malloc(sizeof(tipo_no));
      if(!r){
          printf("\n\n ERRO: Problema de espaço na memória !!!");
          getch(); // segura a tela
          exit(0); // fecha o programa
      }

      r->esq = NULL;
      r->dir = NULL;
      //r->esq = r->dir = NULL; // atribuição multipla
      r->item = x;

      if(!raiz){
        printf("\n\n Item inserido com sucesso!!!");
        getch();
       return r; // primeira entrada => raiz da árvore
      }

      if(x.cod < raiz->item.cod) raiz->esq = r;

      else raiz->dir = r;

      printf("\n\n Item inserido com sucesso!!!");
      getch();

      return r;

  }

  if(x.cod < r->item.cod) inserir(x , r, r->esq);

  else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

  else{
      printf("\n\n IMPOSSÍVEL INSERIR !!!");
      printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
      getch(); // segura a tela
  }
} // fim inserir()
```












```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```

x: 3 Farinha 300

*rai: z 55A2C4

*r: r 55A2C4

op: 1

cod:

achou:

*rai: z 55A2C4

aux: 3 Farinha 300

*esq	item	*dir
NULL	5 Arroz 500	NULL

55A2C4

*rai: z 55A2C4

*r: r F32378

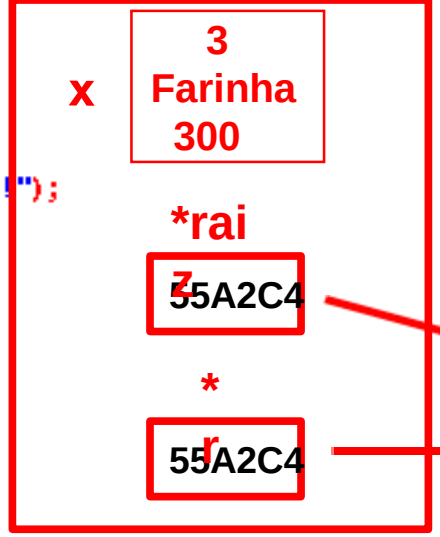
F32378

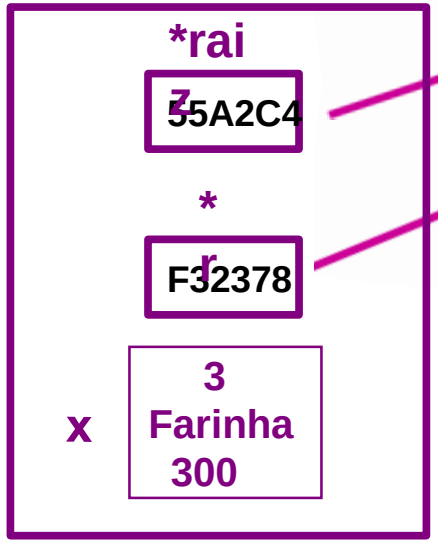
*esq	item	*dir
	cod nome quant	

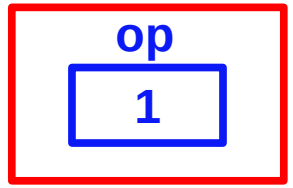
x: 3 Farinha 300

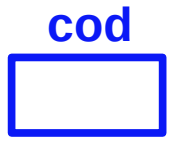
```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

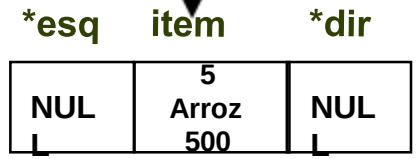
```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```

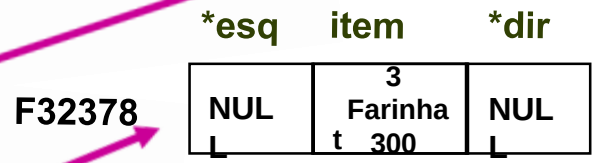






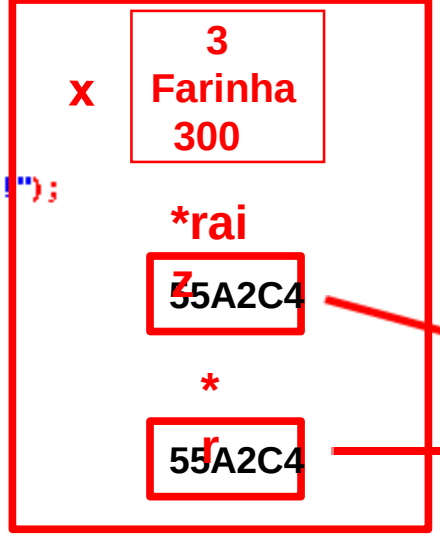


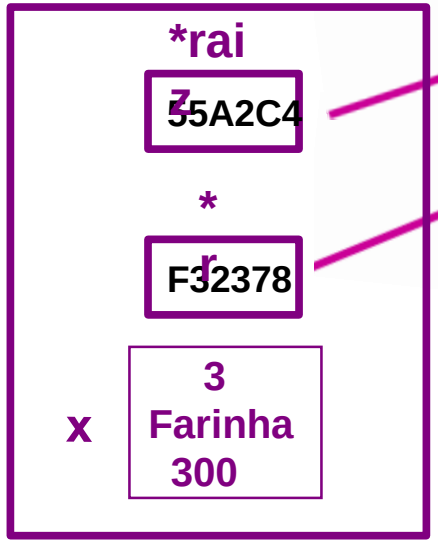


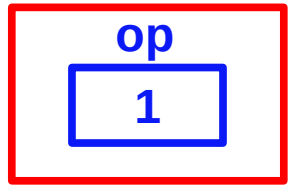


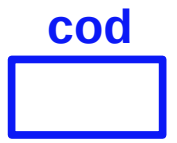

```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

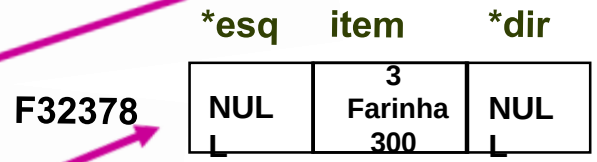
```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

      if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

   if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```





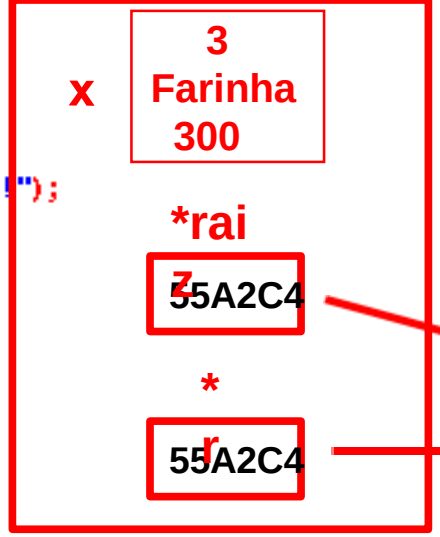


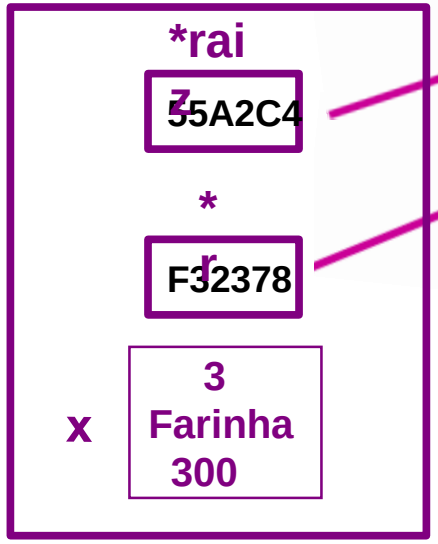


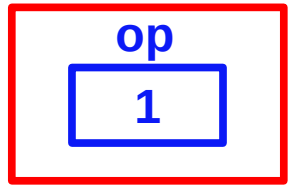


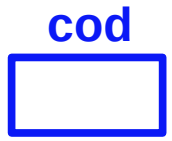

```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

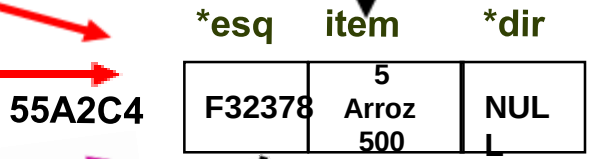
```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

      if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

   if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```

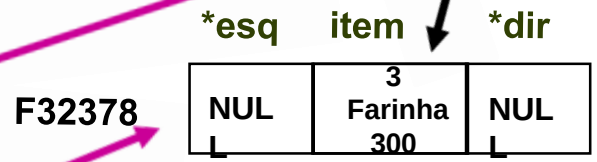






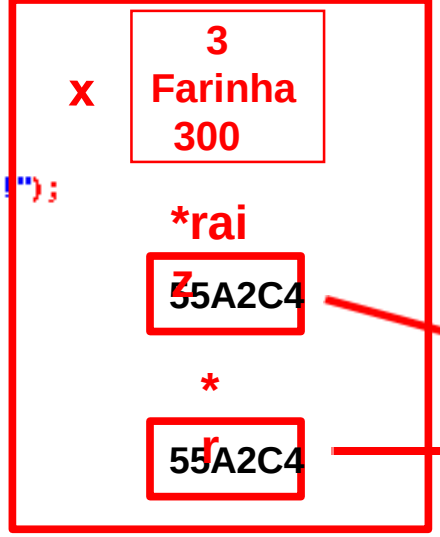


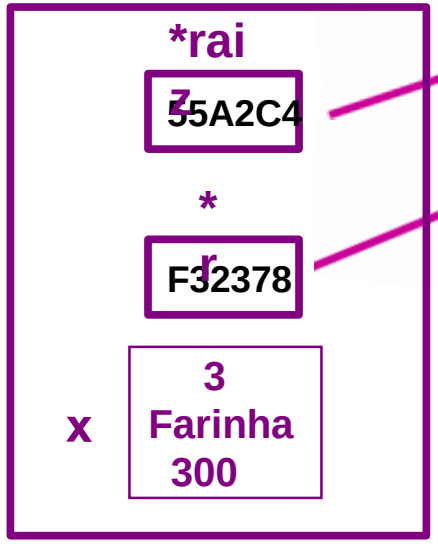


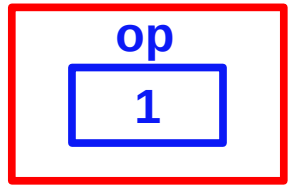


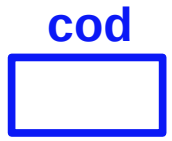

```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

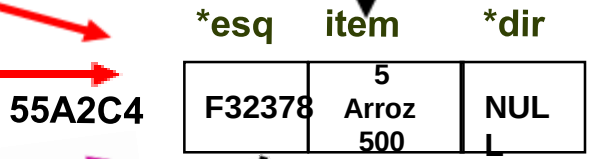
```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;
    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```

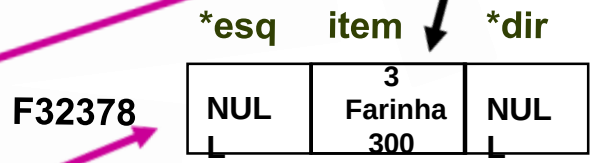








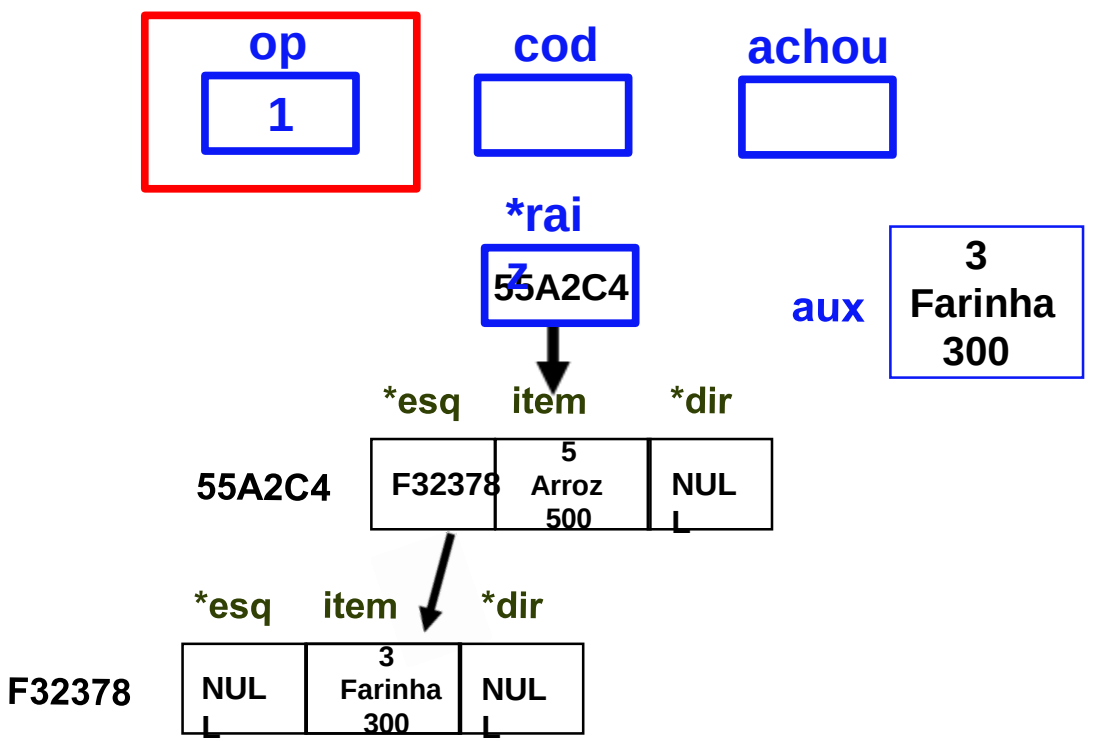





```
Item inserido com sucesso!!!
```

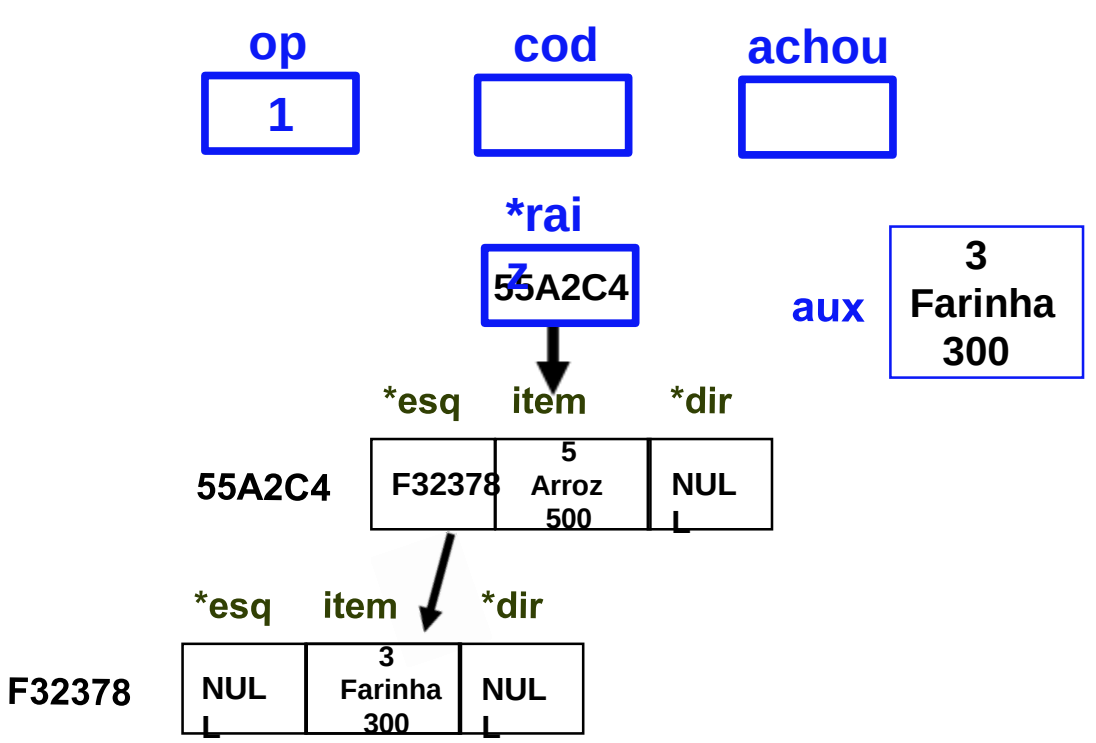
```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

ÁRVORES BINÁRIAS



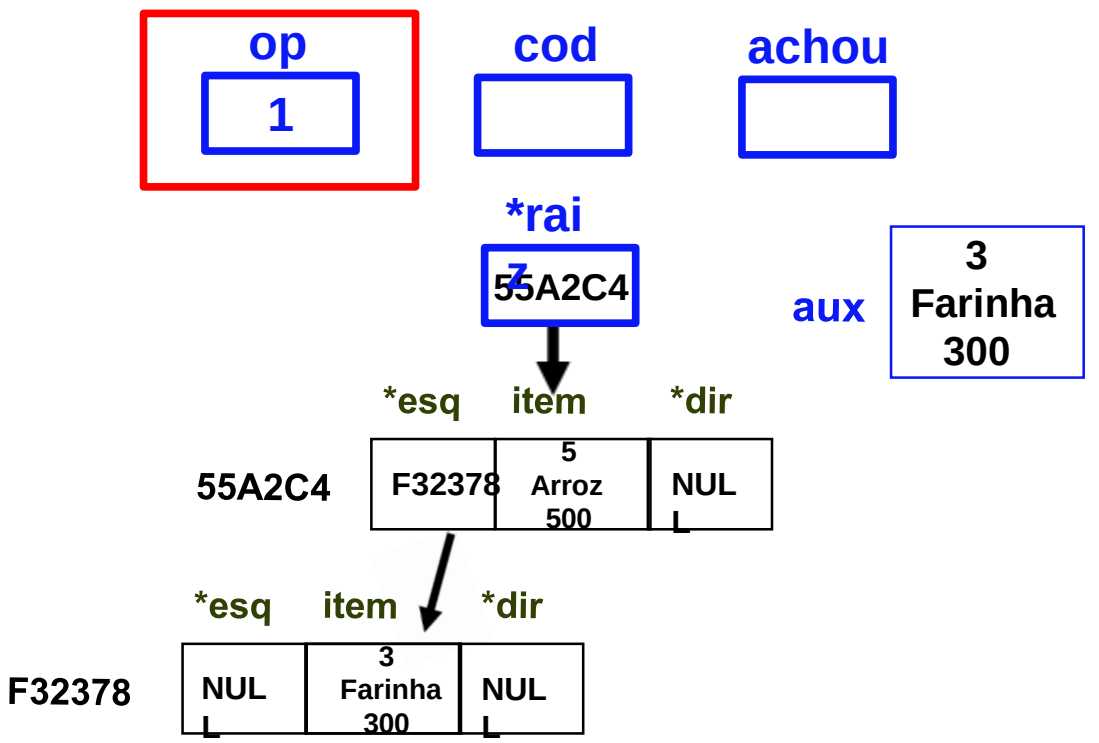

```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

ÁRVORES BINÁRIAS



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

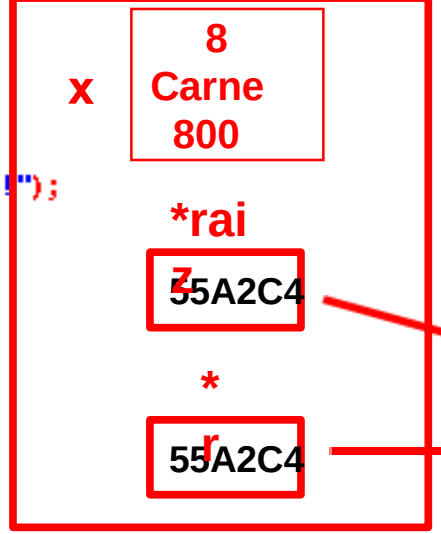
ÁRVORES BINÁRIAS

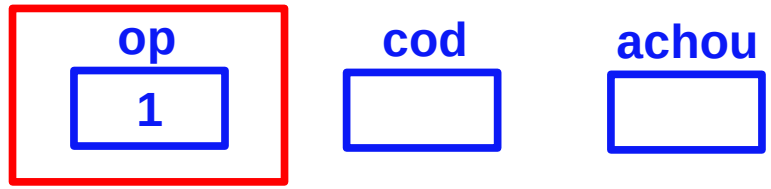


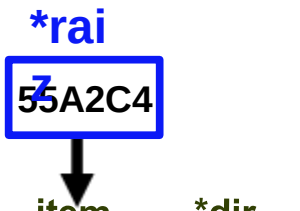

```
********* MENU *********
[1] - INSERIR
[2] - MOSTRAR
[3] - PESQUISAR
[4] - EXCLUIR
[0] - SAIR

Digite sua opção: 1
```

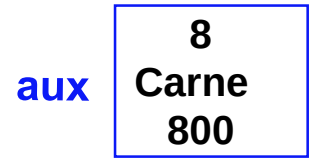
```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

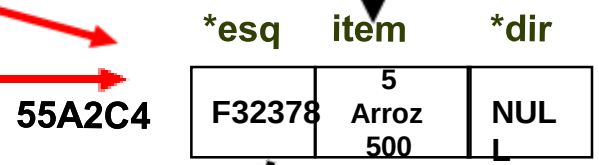
```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```

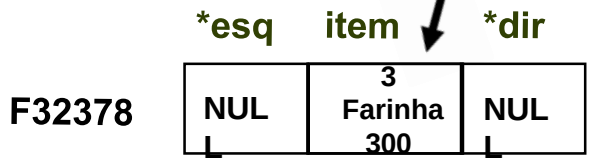






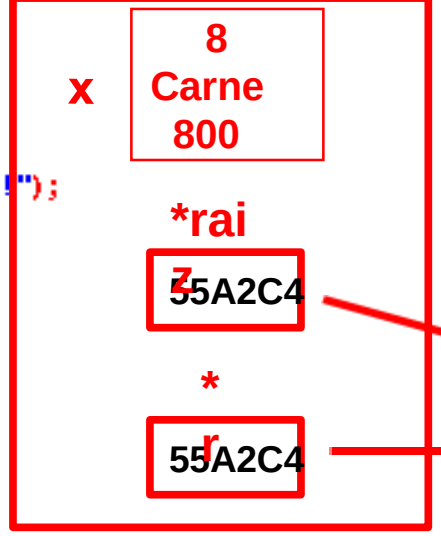


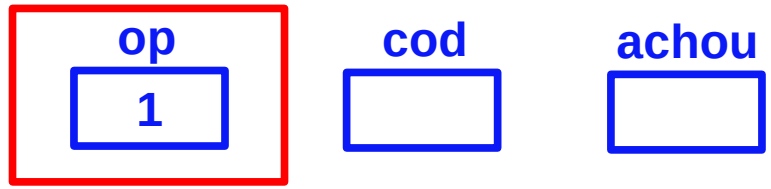


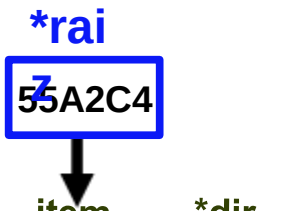


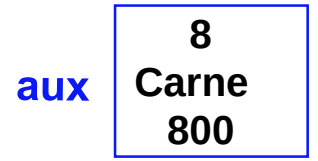

```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

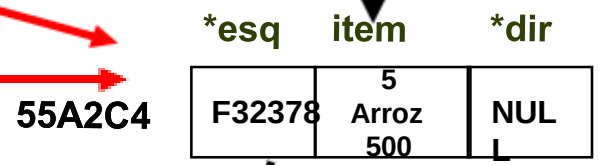
```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```

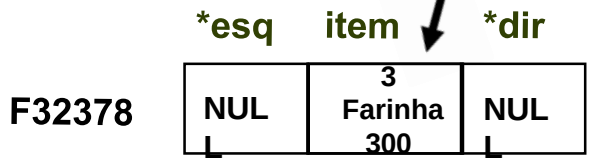






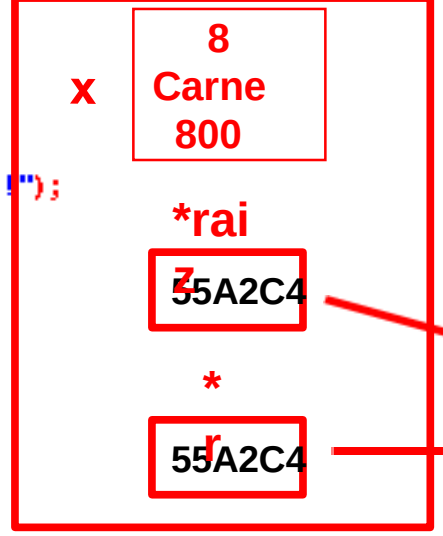


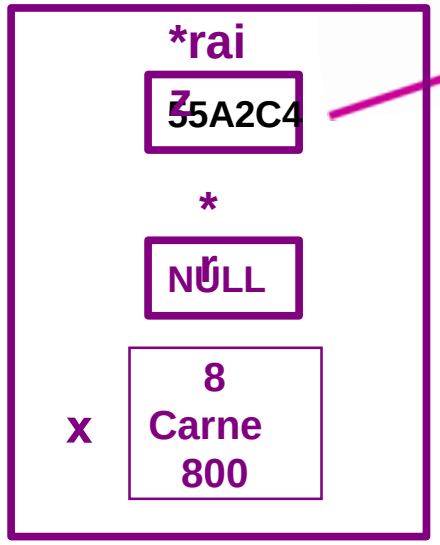


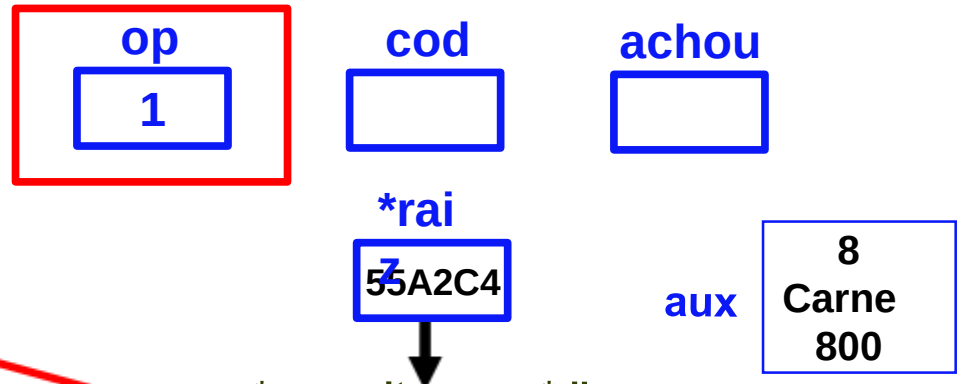


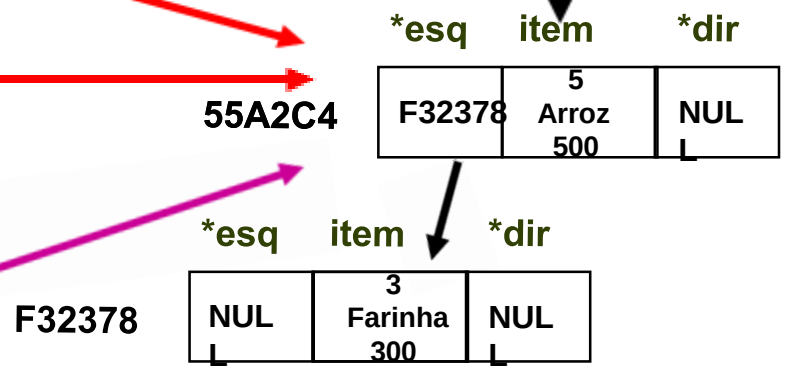

```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```



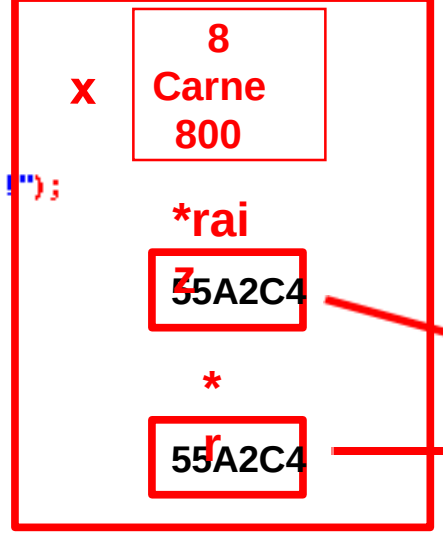


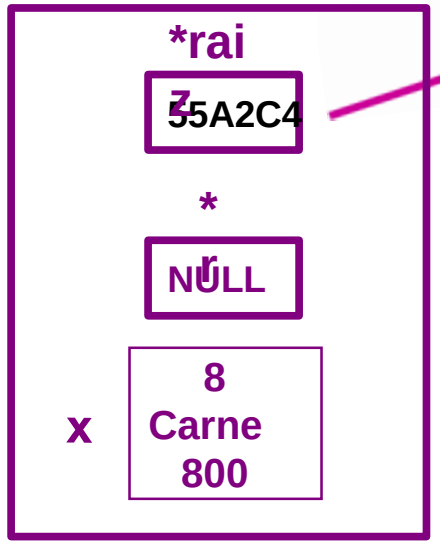


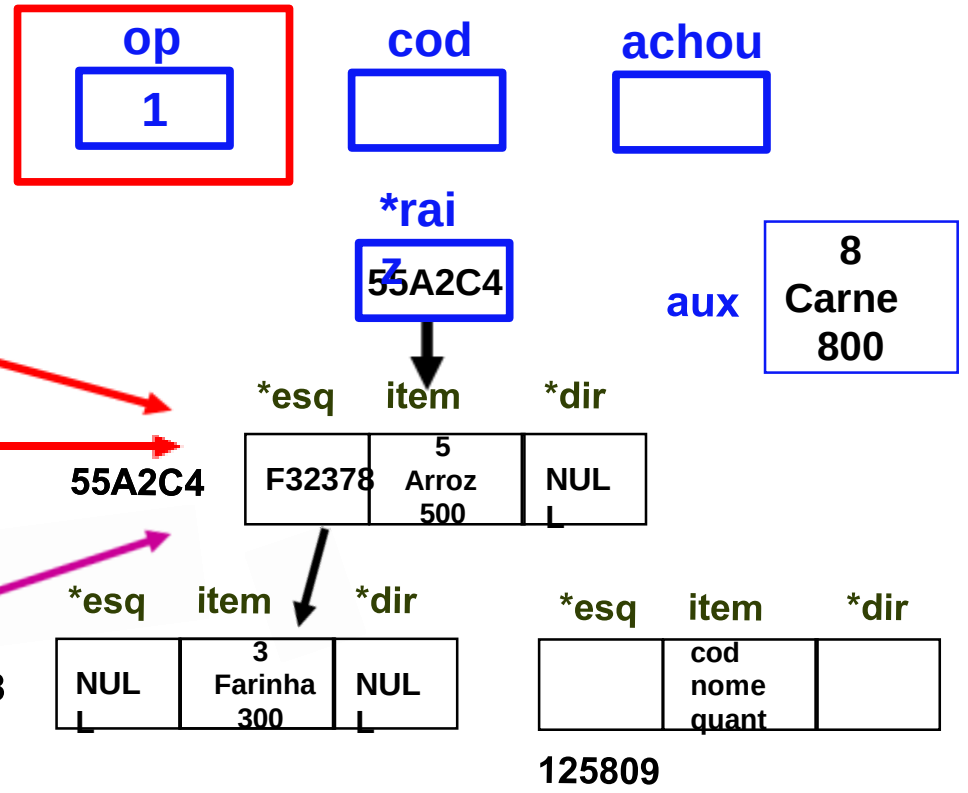



```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){
    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
          return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```

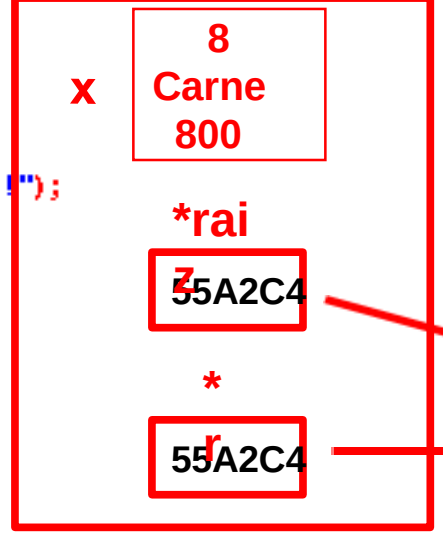


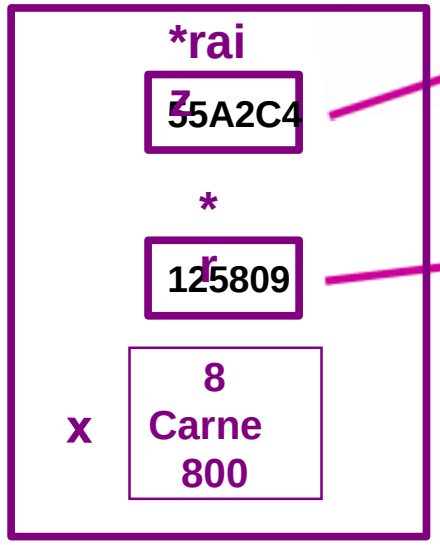


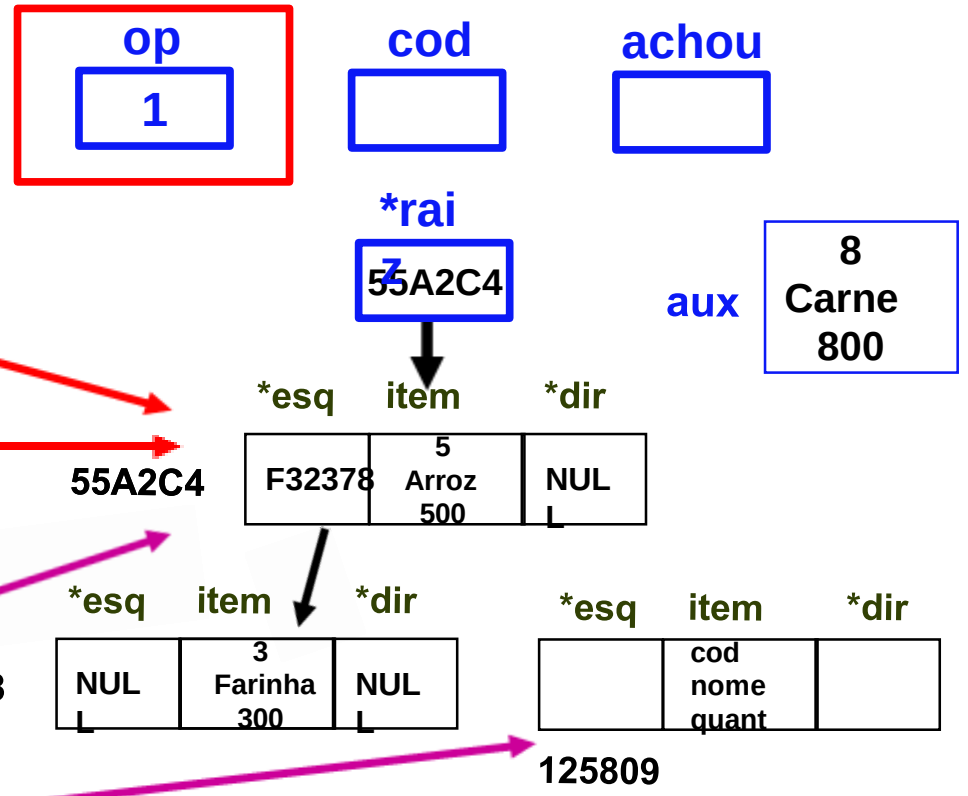



```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){
    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
          return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```

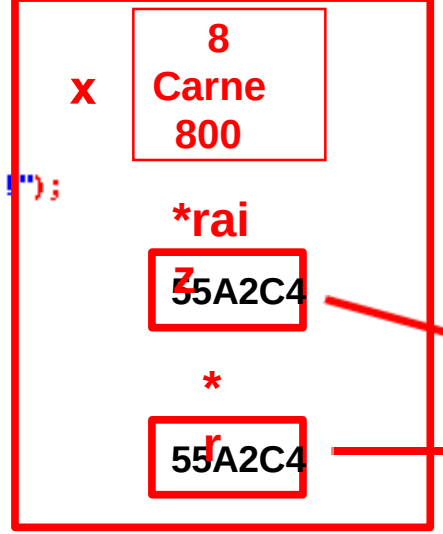


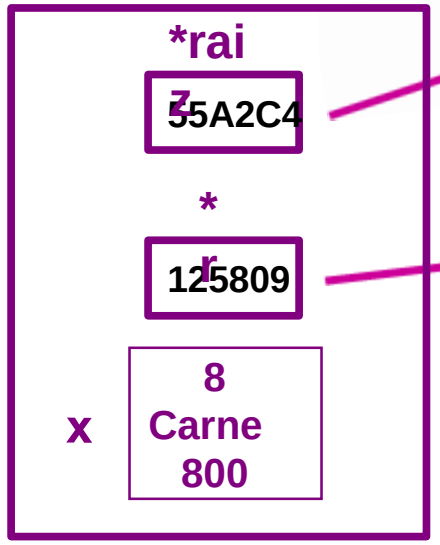


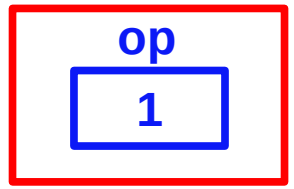


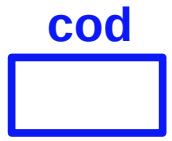

```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

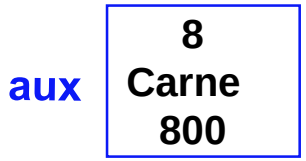
```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```

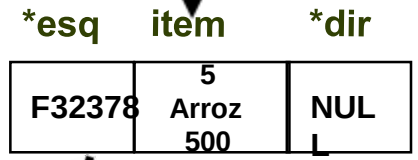


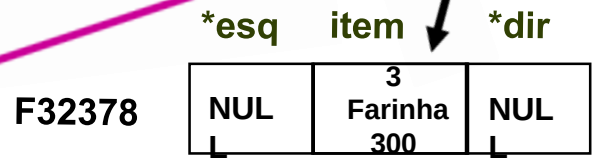


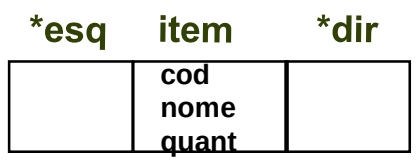






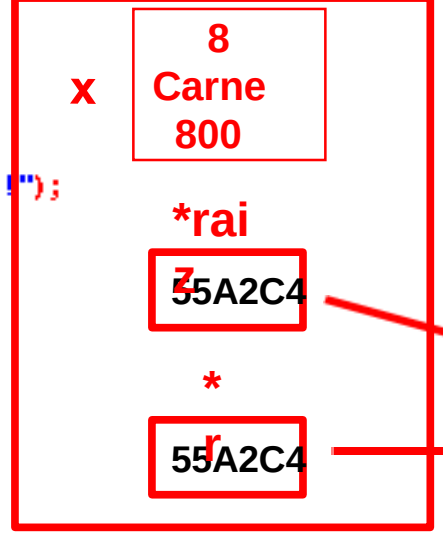


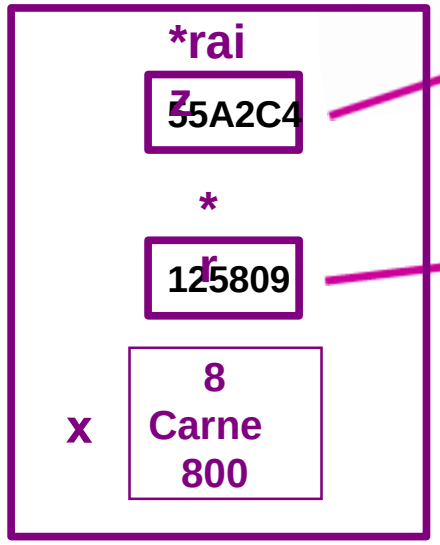


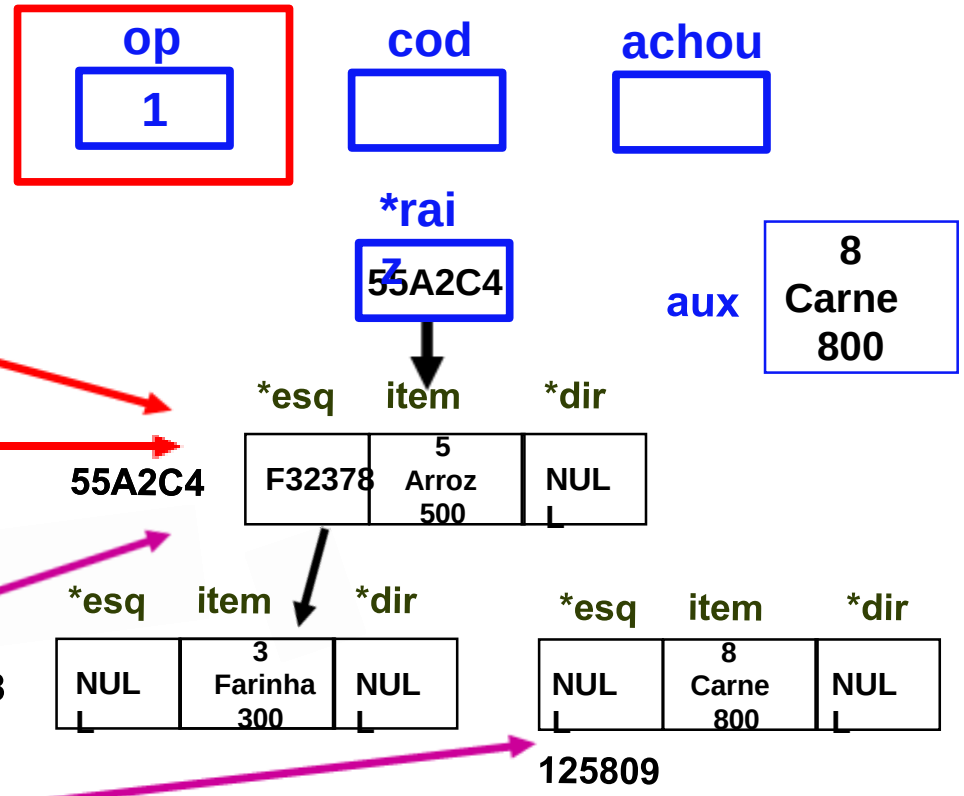



```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```

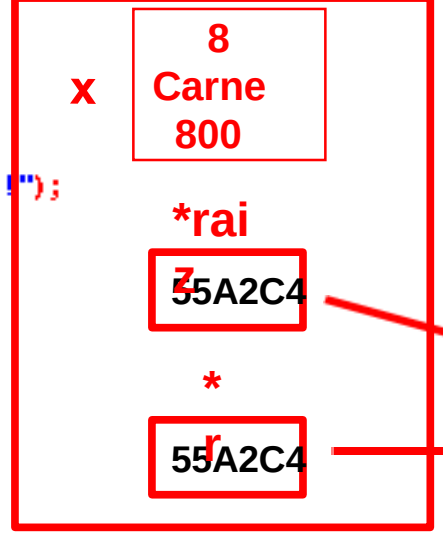


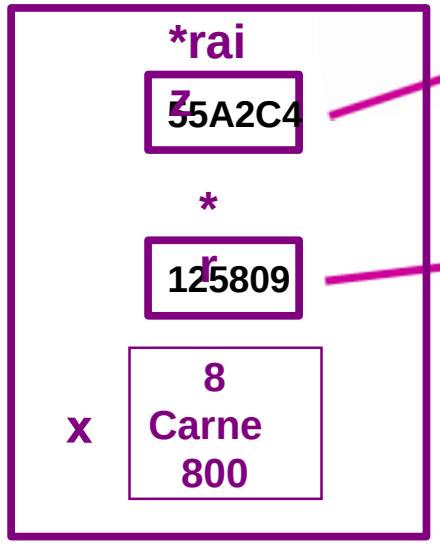


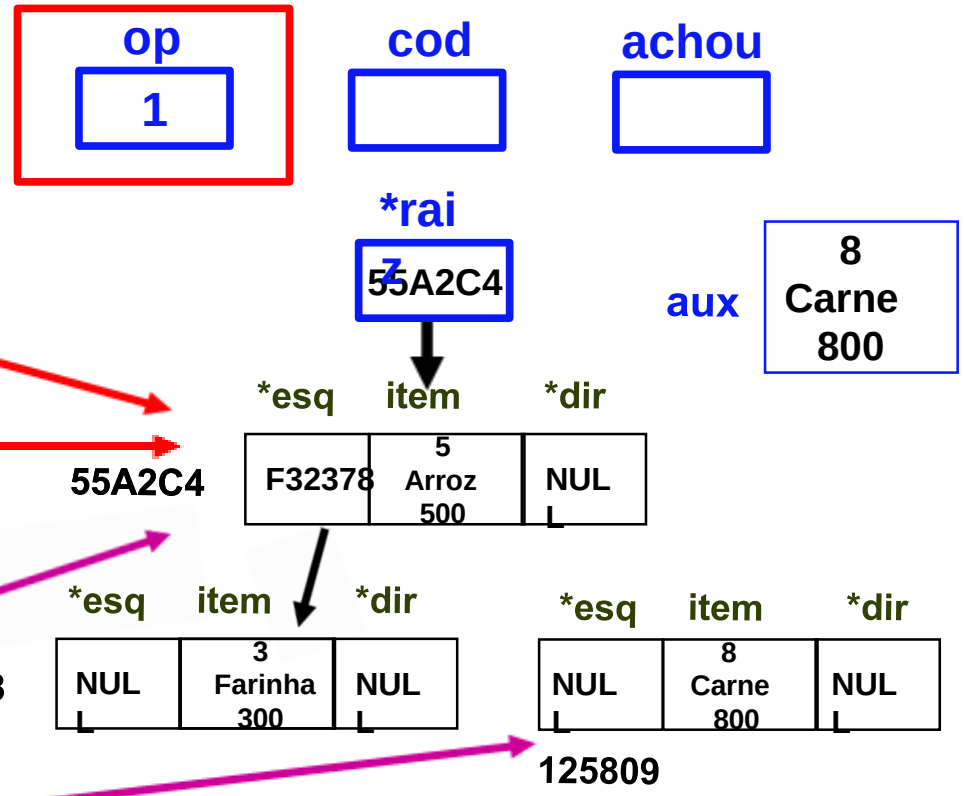



```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```

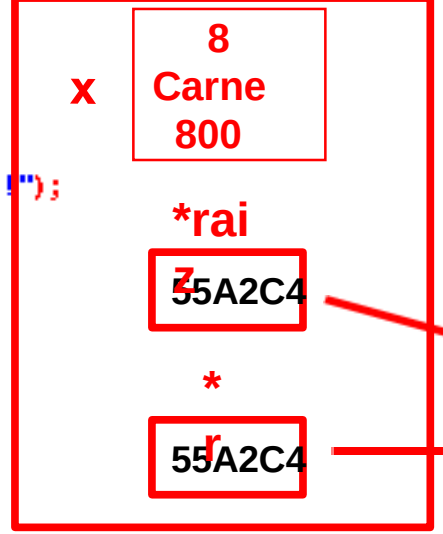


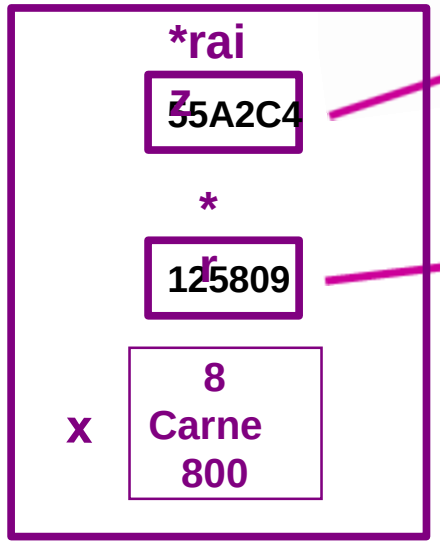


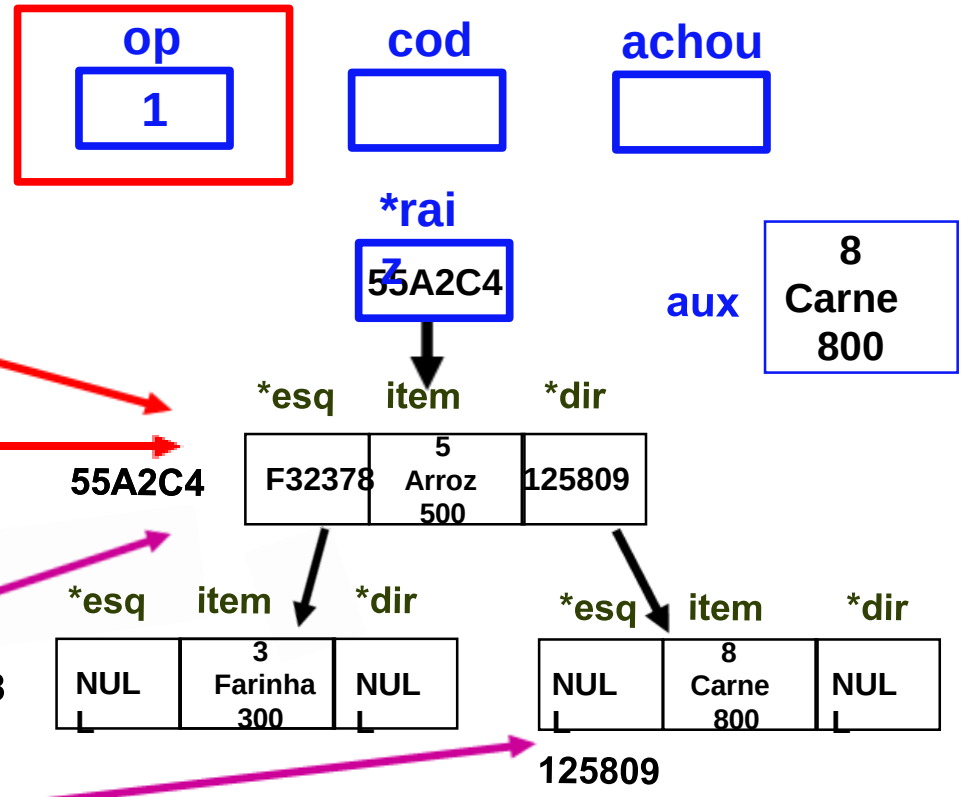



```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;

    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```

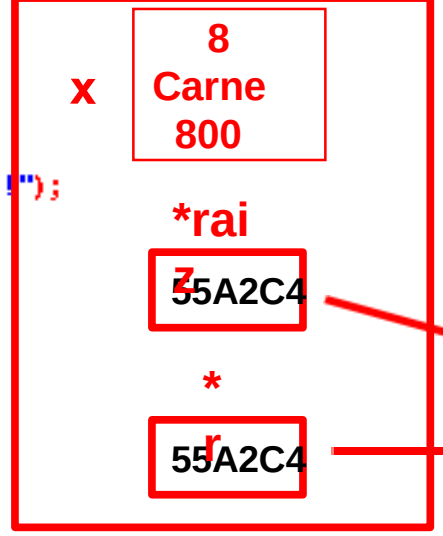


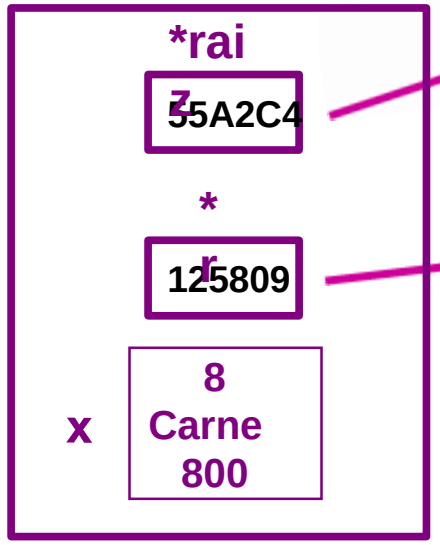


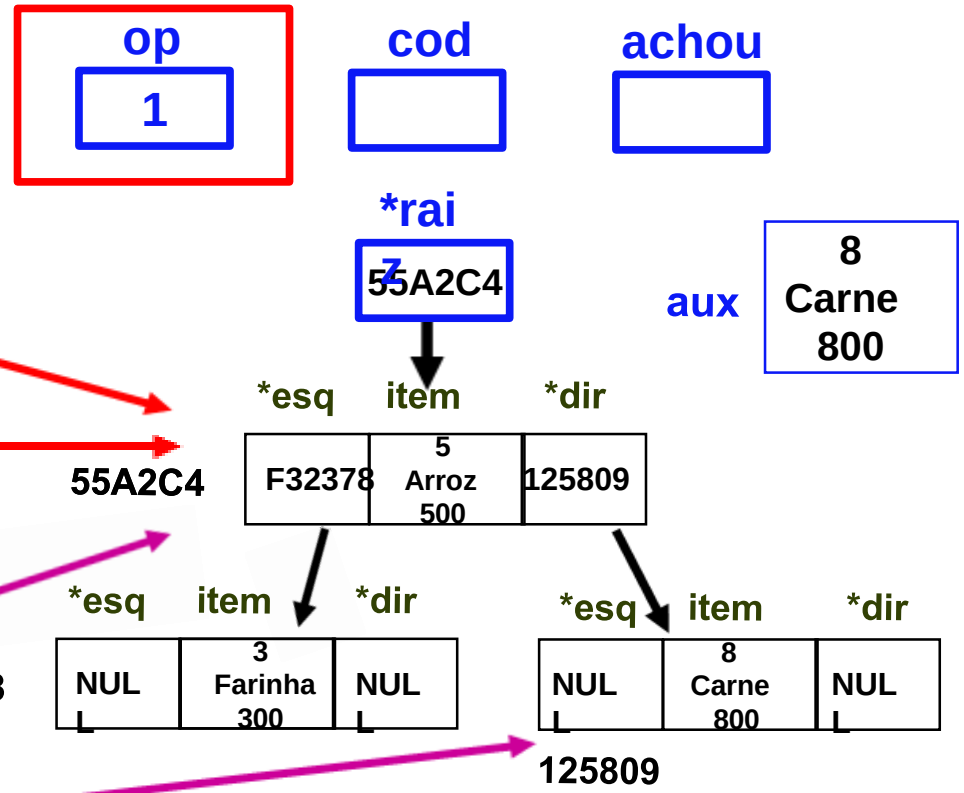


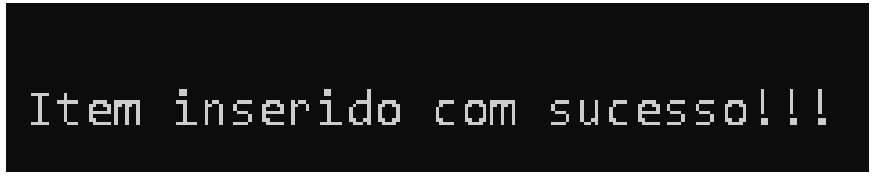

```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

```
tipo_no* inserir(tipo_item x, tipo_no *raiz, tipo_no *r){

    if(!r){
        r = (tipo_no *) malloc(sizeof(tipo_no));
        if(!r){
            printf("\n\n ERRO: Problema de espaço na memória !!!");
            getch(); // segura a tela
            exit(0); // fecha o programa
        }

        r->esq = NULL;
        r->dir = NULL;
        //r->esq = r->dir = NULL; // atribuição multipla
        r->item = x;

        if(!raiz){
          printf("\n\n Item inserido com sucesso!!!");
          getch();
         return r; // primeira entrada => raiz da árvore
        }

        if(x.cod < raiz->item.cod) raiz->esq = r;

        else raiz->dir = r;

        printf("\n\n Item inserido com sucesso!!!");
        getch();

        return r;
    }

    if(x.cod < r->item.cod) inserir(x , r, r->esq);

    else if(x.cod > r->item.cod)  inserir(x , r, r->dir);

    else{
        printf("\n\n IMPOSSÍVEL INSERIR !!!");
        printf("\n\n MOTIVO: Já existe um item com o código %d na árvore", x.cod);
        getch(); // segura a tela
    }
} // fim inserir()
```



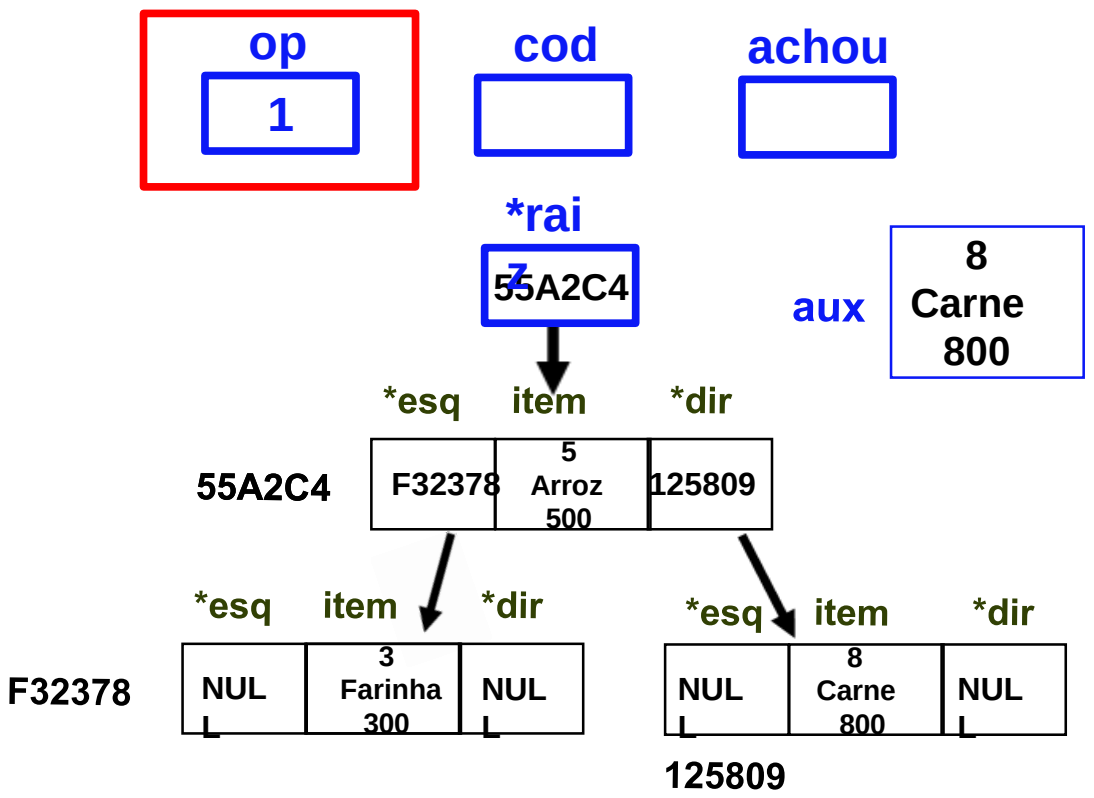






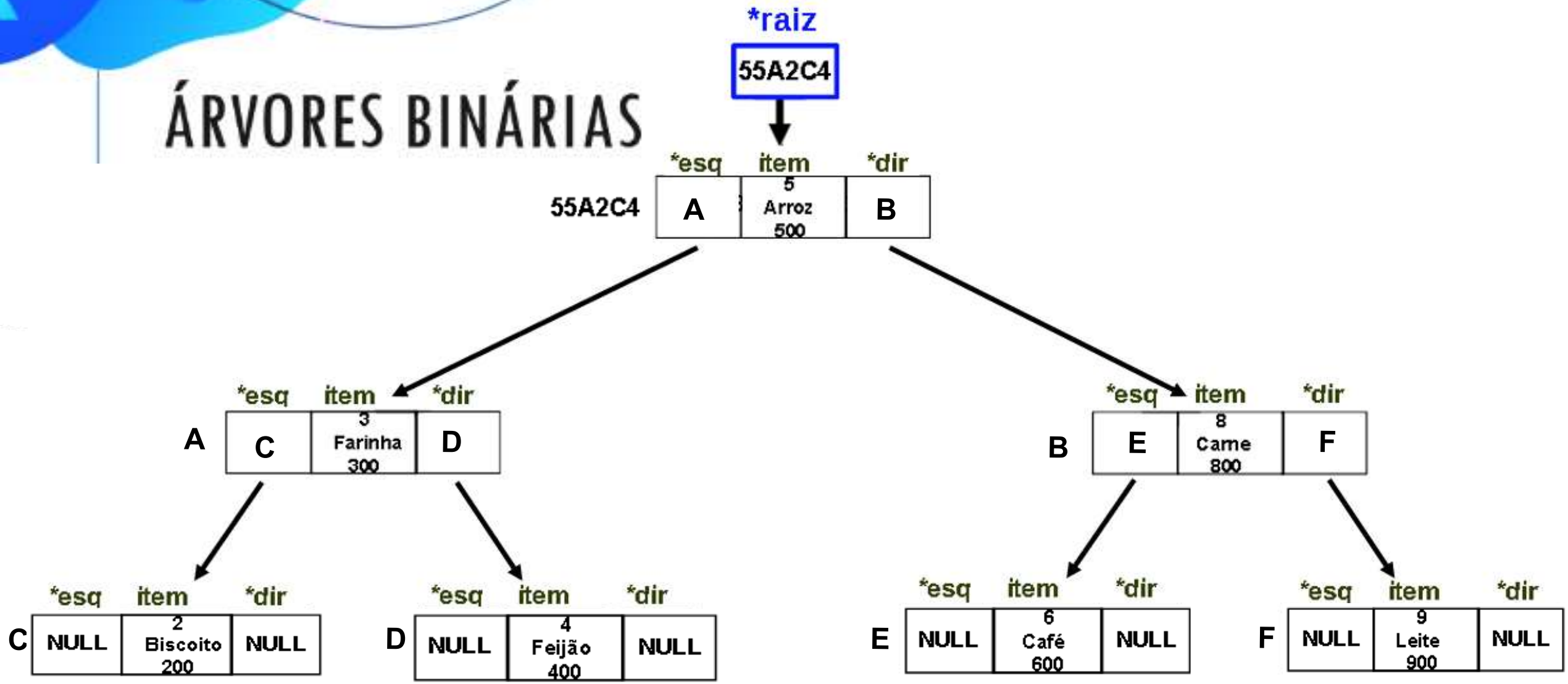

```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

ÁRVORES BINÁRIAS




```
case 1: aux = preencher_item();
        if(!raiz) raiz = inserir(aux, raiz, raiz);
        else inserir(aux, raiz, raiz);
        break;
```

ÁRVORES BINÁRIAS



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

ÁRVORES BINÁRIAS

IMPLEMENTAÇÃO

MOSTRAR O CONTEÚDO DA ÁRVORE

ÁRVORES BINÁRIAS

```c
void mostrar(tipo_no *raiz){

    if(!raiz) return;

    mostrar(raiz->esq);

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

    printf("\n Cod: %d  ", raiz->item.cod);

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

    printf("\n Quantidade: %d  \n", raiz->item.cod);

    mostrar(raiz->dir);
}
```

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

    int op, cod, achou;

    tipo_no *raiz = NULL;

    tipo_item aux;

    do{
        system("cls");

        op = menu();

        switch(op){

            case 0: printf("\n\n\n\t Fim do Programa. \n\n");
                    getch();
                    return(0);

            case 1: aux = preencher_item();
                    if(!raiz) raiz = inserir(aux, raiz, raiz);
                    else inserir(aux, raiz, raiz);
                    break;

            case 2: printf("\n\n CONTEÚDO DA ÁRVORE: ");
                    mostrar(raiz);
                    getch(); // segura a tela
                    break;

            case 3: printf("\n\n PESQUISA NA ÁRVORE ");
                    printf("\n\n Digite o código do item a pesquisar: ");
                    scanf("%d", &cod);
                    achou = 0;
                    pesquisar(&achou, cod, raiz);
                    if(!achou) printf("\n\n O item de código %d não está na árvore", c
                    getch(); // segura a tela
                    break;
```

op: 2

cod:

achou:

*rai: 55A2C4

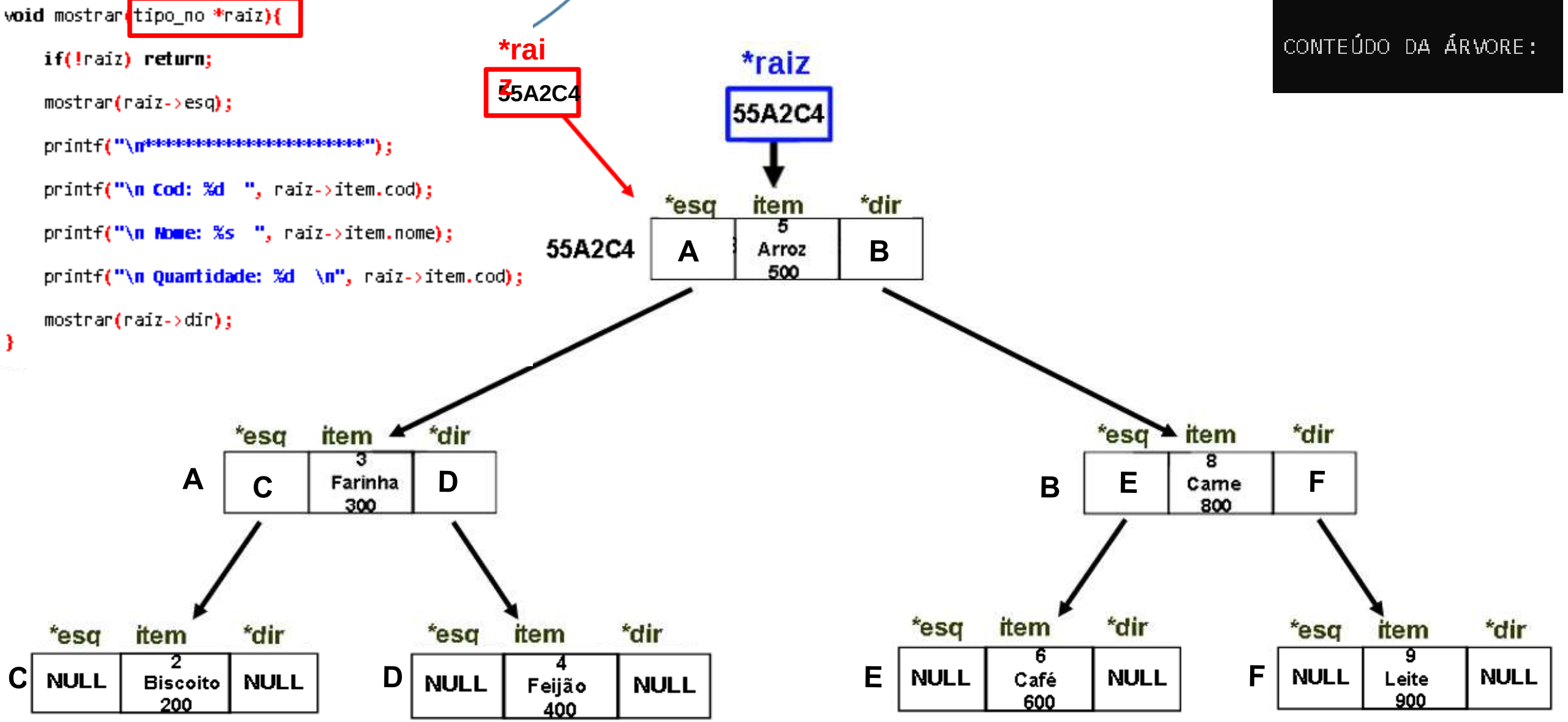
aux: cod, nome, quant

```
********** MENU **********
[1] - INSERIR
[2] - MOSTRAR
[3] - PESQUISAR
[4] - EXCLUIR
[0] - SAIR

Digite sua opção: 2
```

```
CONTEÚDO DA ÁRVORE:
```

```c
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

CONTEÚDO DA ÁRVORE:



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

```
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

CONTEÚDO DA ÁRVORE:

*rai
55A2C4

*raiz
55A2C4

	*esq	item	*dir
55A2C4	A	5 Arroz 500	B

*rai
A

	*esq	item	*dir
A	C	3 Farinha 300	D

	*esq	item	*dir
B	E	8 Carne 800	F

	*esq	item	*dir
C	NULL	2 Biscoito 200	NULL

	*esq	item	*dir
D	NULL	4 Feijão 400	NULL

	*esq	item	*dir
E	NULL	6 Café 600	NULL

	*esq	item	*dir
F	NULL	9 Leite 900	NULL

```c
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

CONTEÚDO DA ÁRVORE:

*rai
55A2C4

*raiz
55A2C4

	*esq	item	*dir
55A2C4	A	5 Arroz 500	B

*rai
A

	*esq	item	*dir
A	C	3 Farinha 300	D

*rai
C

	*esq	item	*dir
C	NULL	2 Biscoito 200	NULL

	*esq	item	*dir
D	NULL	4 Feijão 400	NULL

	*esq	item	*dir
B	E	8 Carne 800	F

	*esq	item	*dir
E	NULL	6 Café 600	NULL

	*esq	item	*dir
F	NULL	9 Leite 900	NULL

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

```c
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

CONTEÚDO DA ÁRVORE:

*rai
55A2C4

*raiz
55A2C4

	*esq	item	*dir
55A2C4	A	5 Arroz 500	B

*rai
A

	*esq	item	*dir
A	C	3 Farinha 300	D

*rai
C

	*esq	item	*dir
C	NULL	2 Biscoito 200	NULL

	*esq	item	*dir
D	NULL	4 Feijão 400	NULL

	*esq	item	*dir
B	E	8 Carne 800	F

	*esq	item	*dir
E	NULL	6 Café 600	NULL

	*esq	item	*dir
F	NULL	9 Leite 900	NULL

*rai
NULL

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

```c
void mostrar(tipo_no *raiz){
   if(!raiz) return;
   mostrar(raiz->esq);
   printf("\n**********************");
   printf("\n Cod: %d  ", raiz->item.cod);
   printf("\n Nome: %s  ", raiz->item.nome);
   printf("\n Quantidade: %d  \n", raiz->item.cod);
   mostrar(raiz->dir);
}
```

CONTEÚDO DA ÁRVORE:

*rai
55A2C4

*raiz
55A2C4

55A2C4	*esq	item	*dir
	A	5 Arroz 500	B

*rai
A

A	*esq	item	*dir
	C	3 Farinha 300	D

B	*esq	item	*dir
	E	8 Carne 800	F

*rai
C

C	*esq	item	*dir
	NULL	2 Biscoito 200	NULL

D	*esq	item	*dir
	NULL	4 Feijão 400	NULL

E	*esq	item	*dir
	NULL	6 Café 600	NULL

F	*esq	item	*dir
	NULL	9 Leite 900	NULL

*rai
NULL

```
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

```
CONTEÚDO DA ÁRVORE:

**********************
Cod: 2
Nome: Biscoito
Quantidade: 200
```

*rai
z 55A2C4

*raiz
55A2C4

55A2C4	*esq	item	*dir
	A	5 Arroz 500	B

*rai
z A

A	*esq	item	*dir
	C	3 Farinha 300	D

*rai
z C

C	*esq	item	*dir
	NULL	2 Biscoito 200	NULL

D	*esq	item	*dir
	NULL	4 Feijão 400	NULL

B	*esq	item	*dir
	E	8 Carne 800	F

E	*esq	item	*dir
	NULL	6 Café 600	NULL

F	*esq	item	*dir
	NULL	9 Leite 900	NULL

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

```
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

```
CONTEÚDO DA ÁRVORE:

**********************
Cod: 2
Nome: Biscoito
Quantidade: 200
```

*rai
55A2C4

*raiz
55A2C4

55A2C4 | *esq: A | item: 5, Arroz, 500 | *dir: B

*rai
A

A | *esq: C | item: 3, Farinha, 300 | *dir: D

*rai
C

C | *esq: NULL | item: 2, Biscoito, 200 | *dir: NULL

D | *esq: NULL | item: 4, Feijão, 400 | *dir: NULL

B | *esq: E | item: 8, Carne, 800 | *dir: F

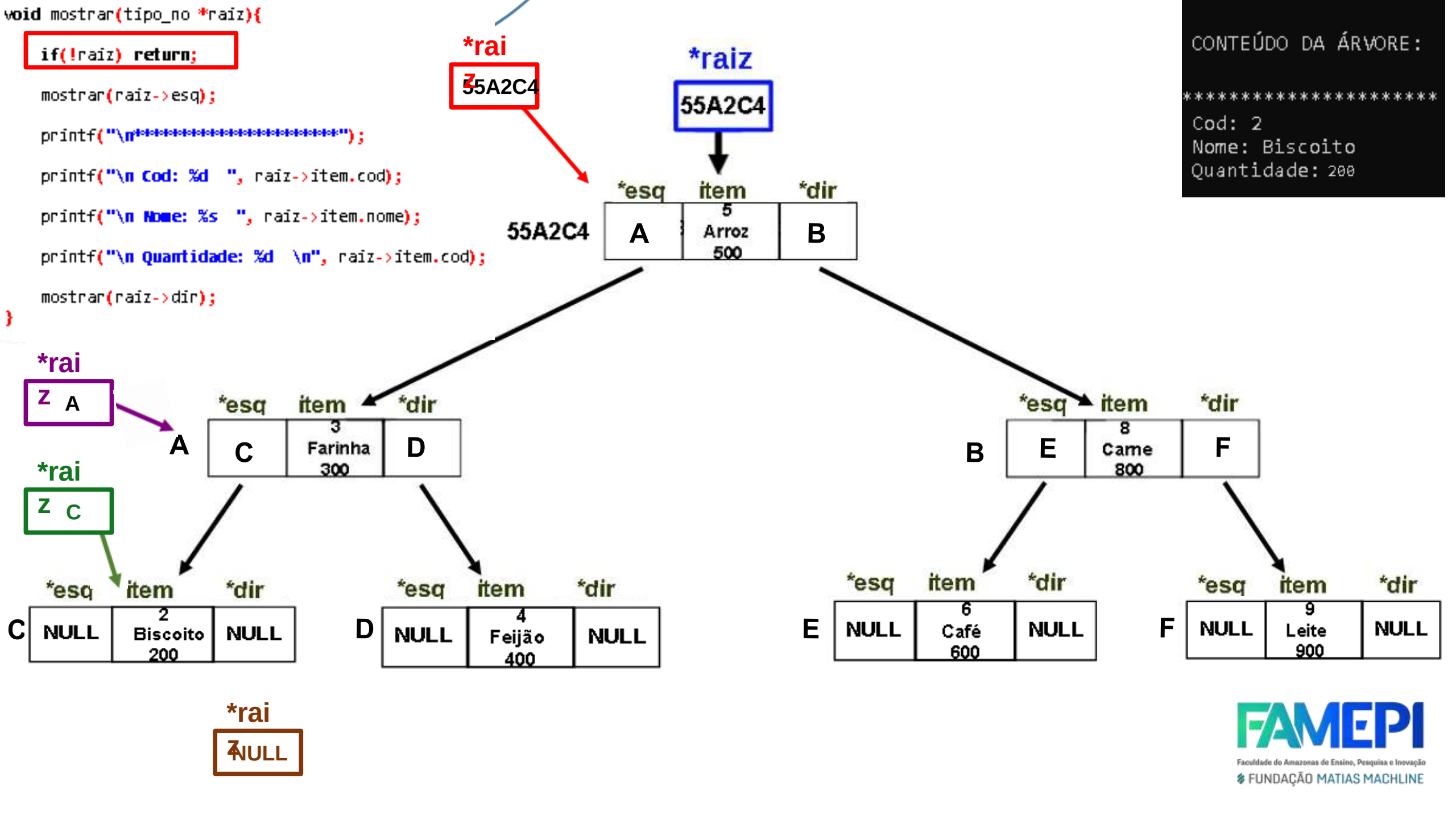
E | *esq: NULL | item: 6, Café, 600 | *dir: NULL

F | *esq: NULL | item: 9, Leite, 900 | *dir: NULL

*rai
NULL

```
void mostrar(tipo_no *raiz){
   if(!raiz) return;
   mostrar(raiz->esq);
   printf("\n**********************");
   printf("\n Cod: %d  ", raiz->item.cod);
   printf("\n Nome: %s  ", raiz->item.nome);
   printf("\n Quantidade: %d  \n", raiz->item.cod);
   mostrar(raiz->dir);
}
```

```
CONTEÚDO DA ÁRVORE:

**********************
Cod: 2
Nome: Biscoito
Quantidade: 200
```



```
void mostrar(tipo_no *raiz){
   if(!raiz) return;
   mostrar(raiz->esq);
   printf("\n**********************");
   printf("\n Cod: %d  ", raiz->item.cod);
   printf("\n Nome: %s  ", raiz->item.nome);
   printf("\n Quantidade: %d  \n", raiz->item.cod);
   mostrar(raiz->dir);
}
```

```
CONTEÚDO DA ÁRVORE:

**********************
Cod: 2
Nome: Biscoito
Quantidade: 200
```

*rai
z 55A2C4

*raiz
55A2C4

	*esq	item	*dir
55A2C4	A	5 Arroz 500	B

*rai
z A

	*esq	item	*dir
A	C	3 Farinha 300	D

	*esq	item	*dir
B	E	8 Carne 800	F

*rai
z C

	*esq	item	*dir
C	NULL	2 Biscoito 200	NULL
D	NULL	4 Feijão 400	NULL
E	NULL	6 Café 600	NULL
F	NULL	9 Leite 900	NULL

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

```
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

```
CONTEÚDO DA ÁRVORE:

**********************
Cod: 2
Nome: Biscoito
Quantidade: 200
**********************
Cod: 3
Nome: Farinha
Quantidade: 300
```

*rai
z 55A2C4

*raiz
55A2C4

	*esq	item	*dir
55A2C4	A	5 Arroz 500	B

*rai
z A

	*esq	item	*dir
A	C	3 Farinha 300	D
B	E	8 Carne 800	F
C	NULL	2 Biscoito 200	NULL
D	NULL	4 Feijão 400	NULL
E	NULL	6 Café 600	NULL
F	NULL	9 Leite 900	NULL

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

```
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

```
CONTEÚDO DA ÁRVORE:

**********************
Cod: 2
Nome: Biscoito
Quantidade: 200
**********************
Cod: 3
Nome: Farinha
Quantidade: 300
```

*rai
Z 55A2C4

*raiz
55A2C4

55A2C4 | *esq: A | item: 5 Arroz 500 | *dir: B

*rai
Z A

A | *esq: C | item: 3 Farinha 300 | *dir: D

*rai
Z D

B | *esq: E | item: 8 Carne 800 | *dir: F

C | *esq: NULL | item: 2 Biscoito 200 | *dir: NULL

D | *esq: NULL | item: 4 Feijão 400 | *dir: NULL

E | *esq: NULL | item: 6 Café 600 | *dir: NULL

F | *esq: NULL | item: 9 Leite 900 | *dir: NULL

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

```
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

```
CONTEÚDO DA ÁRVORE:

**********************
Cod: 2
Nome: Biscoito
Quantidade: 200
**********************
Cod: 3
Nome: Farinha
Quantidade: 300
```

*rai
55A2C4

*raiz
55A2C4

55A2C4	*esq	item	*dir
	A	5 Arroz 500	B

*rai
A

A	*esq	item	*dir
	C	3 Farinha 300	D

*rai
D

B	*esq	item	*dir
	E	8 Carne 800	F

C	*esq	item	*dir
	NULL	2 Biscoito 200	NULL

D	*esq	item	*dir
	NULL	4 Feijão 400	NULL

E	*esq	item	*dir
	NULL	6 Café 600	NULL

F	*esq	item	*dir
	NULL	9 Leite 900	NULL

*rai
NULL

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

```
void mostrar(tipo_no *raiz){
   if(!raiz) return;
   mostrar(raiz->esq);
   printf("\n**********************");
   printf("\n Cod: %d  ", raiz->item.cod);
   printf("\n Nome: %s  ", raiz->item.nome);
   printf("\n Quantidade: %d  \n", raiz->item.cod);
   mostrar(raiz->dir);
}
```

```
CONTEÚDO DA ÁRVORE:

**********************
Cod: 2
Nome: Biscoito
Quantidade: 200
**********************
Cod: 3
Nome: Farinha
Quantidade: 300
```

*rai
55A2C4

*raiz
55A2C4

55A2C4	*esq	item	*dir
	A	5 Arroz 500	B

*rai
A

A	*esq	item	*dir
	C	3 Farinha 300	D

B	*esq	item	*dir
	E	8 Carne 800	F

*rai
D

C	*esq	item	*dir
	NULL	2 Biscoito 200	NULL

D	*esq	item	*dir
	NULL	4 Feijão 400	NULL

E	*esq	item	*dir
	NULL	6 Café 600	NULL

F	*esq	item	*dir
	NULL	9 Leite 900	NULL

*rai
NULL

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

```
void mostrar(tipo_no *raiz){
   if(!raiz) return;
   mostrar(raiz->esq);
   printf("\n**********************");
   printf("\n Cod: %d  ", raiz->item.cod);
   printf("\n Nome: %s  ", raiz->item.nome);
   printf("\n Quantidade: %d  \n", raiz->item.cod);
   mostrar(raiz->dir);
}
```

```
CONTEÚDO DA ÁRVORE:

**********************
Cod: 2
Nome: Biscoito
Quantidade: 200
**********************
Cod: 3
Nome: Farinha
Quantidade: 300
```

*rai
Z 55A2C4

*raiz
55A2C4

55A2C4	*esq	item	*dir
	A	5 Arroz 500	B

*rai
Z A

A	*esq	item	*dir
	C	3 Farinha 300	D

*rai
Z D

B	*esq	item	*dir
	E	8 Carne 800	F

C	*esq	item	*dir
	NULL	2 Biscoito 200	NULL

D	*esq	item	*dir
	NULL	4 Feijão 400	NULL

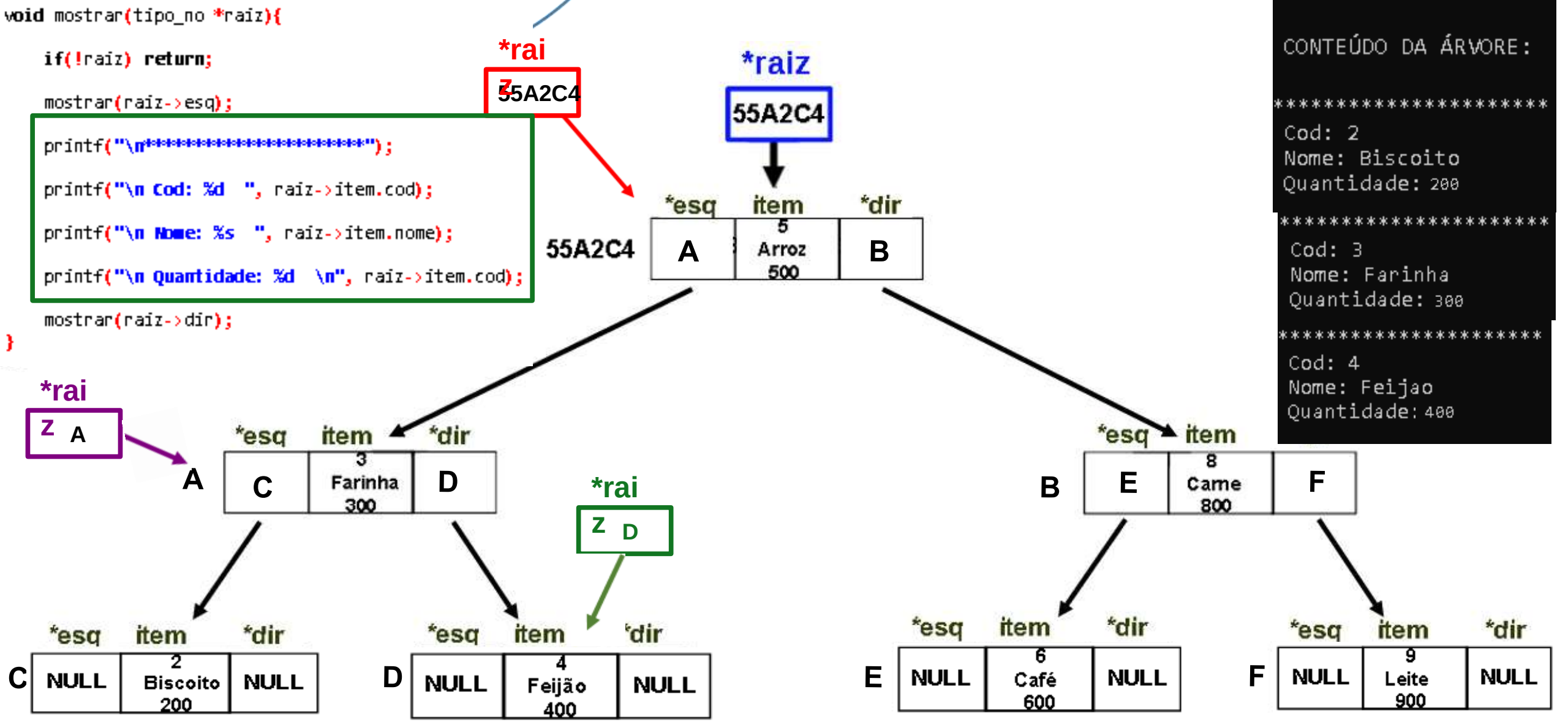
E	*esq	item	*dir
	NULL	6 Café 600	NULL

F	*esq	item	*dir
	NULL	9 Leite 900	NULL

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

```c
void mostrar(tipo_no *raiz){

    if(!raiz) return;

    mostrar(raiz->esq);

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

    printf("\n Cod: %d  ", raiz->item.cod);

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

    printf("\n Quantidade: %d  \n", raiz->item.cod);

    mostrar(raiz->dir);
}
```

```
CONTEÚDO DA ÁRVORE:

**********************
Cod: 2
Nome: Biscoito
Quantidade: 200
**********************
Cod: 3
Nome: Farinha
Quantidade: 300
**********************
Cod: 4
Nome: Feijao
Quantidade: 400
```



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

```
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

```
CONTEÚDO DA ÁRVORE:

**********************
Cod: 2
Nome: Biscoito
Quantidade: 200
**********************
Cod: 3
Nome: Farinha
Quantidade: 300
**********************
Cod: 4
Nome: Feijao
Quantidade: 400
```

*rai
Z 55A2C4

*raiz
55A2C4

55A2C4

*esq	item	*dir
A	5 Arroz 500	B

*rai
Z A

A

*esq	item	*dir
C	3 Farinha 300	D

*rai
Z D

B

*esq	item	*dir
E	8 Carne 800	F

C

*esq	item	*dir
NULL	2 Biscoito 200	NULL

D

*esq	item	*dir
NULL	4 Feijão 400	NULL

E

*esq	item	*dir
NULL	6 Café 600	NULL

F

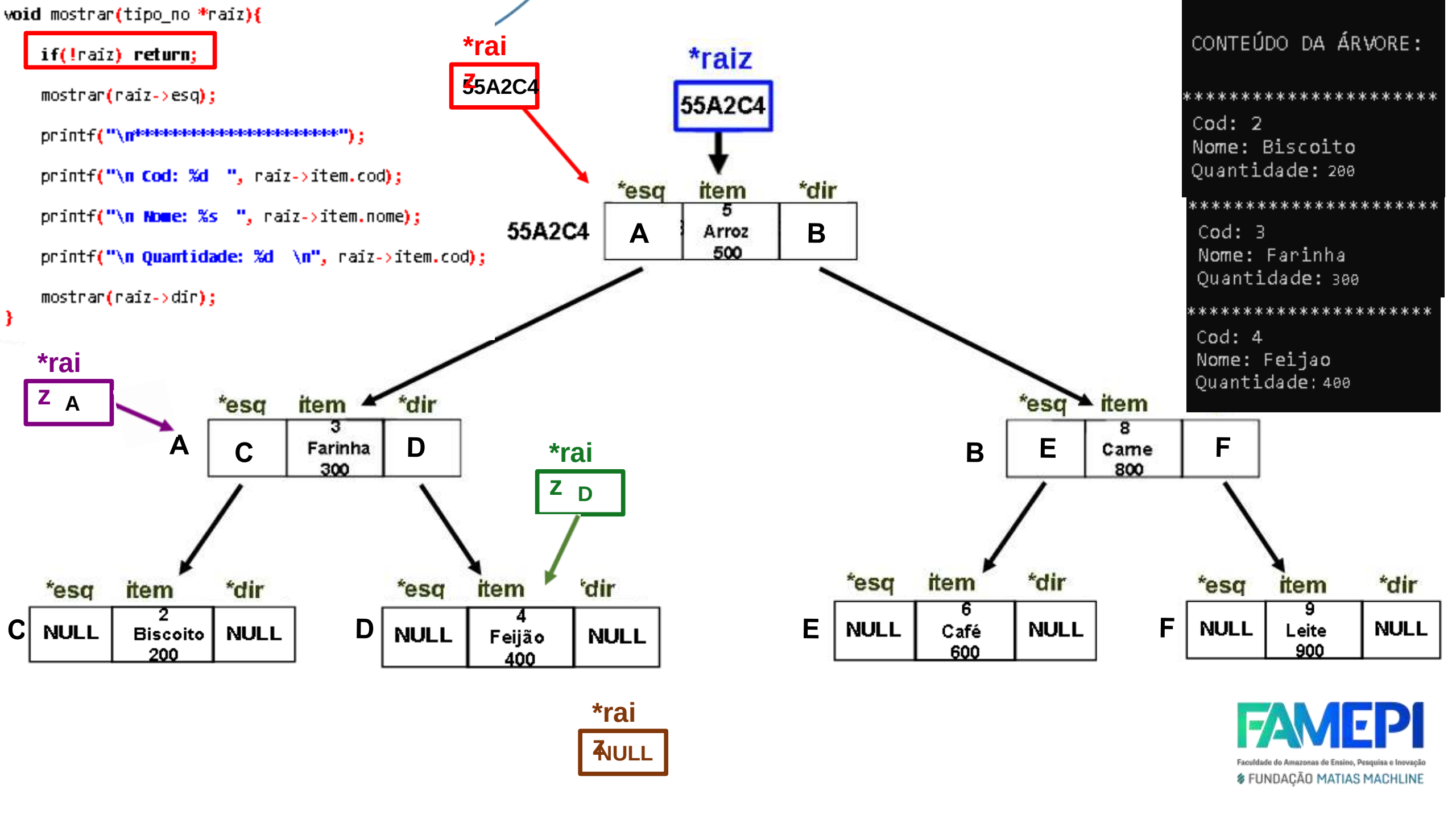
*esq	item	*dir
NULL	9 Leite 900	NULL

*rai
Z NULL

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

```c
void mostrar(tipo_no *raiz){
   if(!raiz) return;
   mostrar(raiz->esq);
   printf("\n**********************");
   printf("\n Cod: %d  ", raiz->item.cod);
   printf("\n Nome: %s  ", raiz->item.nome);
   printf("\n Quantidade: %d  \n", raiz->item.cod);
   mostrar(raiz->dir);
}
```

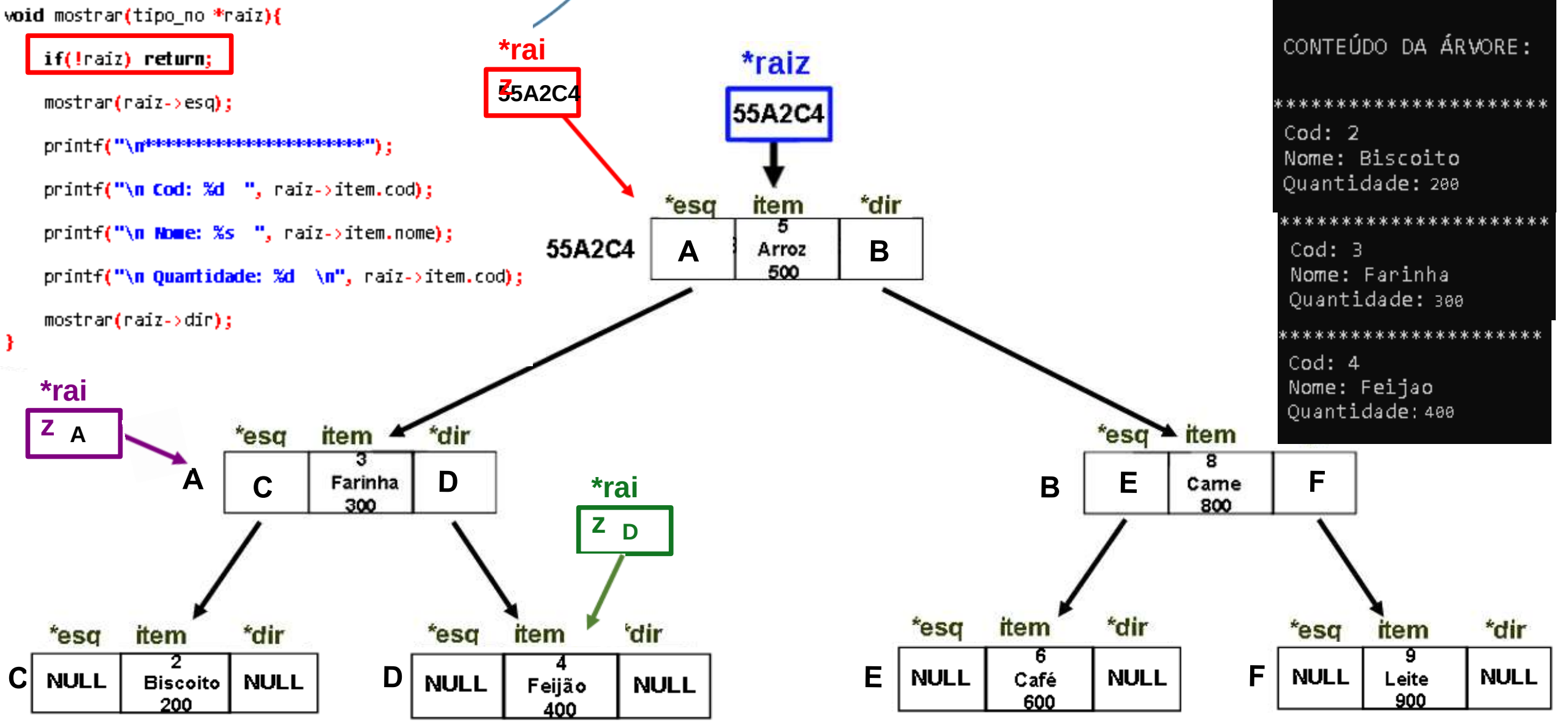
```
CONTEÚDO DA ÁRVORE:

**********************
Cod: 2
Nome: Biscoito
Quantidade: 200
**********************
Cod: 3
Nome: Farinha
Quantidade: 300
**********************
Cod: 4
Nome: Feijao
Quantidade: 400
```



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

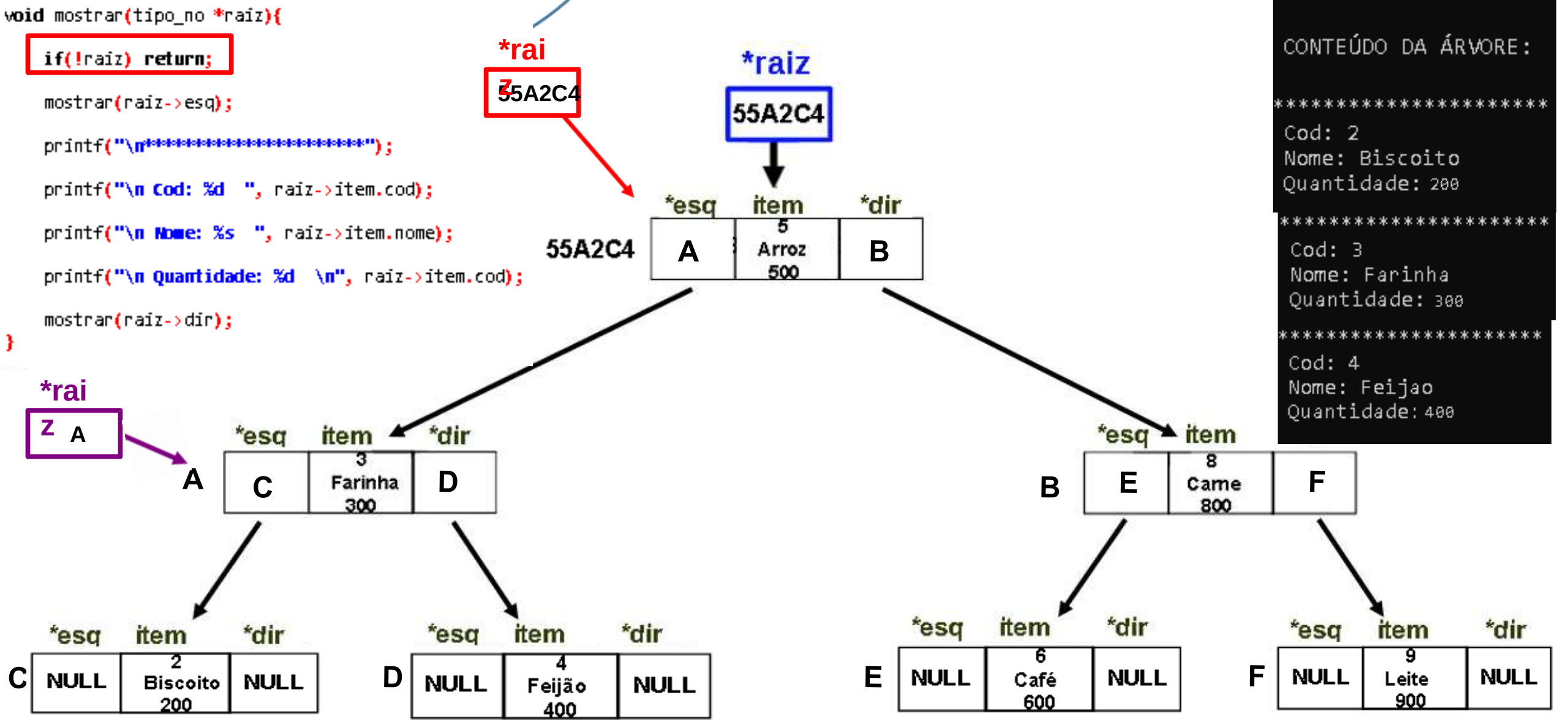
```c
void mostrar(tipo_no *raiz){
   if(!raiz) return;
   mostrar(raiz->esq);
   printf("\n**********************");
   printf("\n Cod: %d  ", raiz->item.cod);
   printf("\n Nome: %s  ", raiz->item.nome);
   printf("\n Quantidade: %d  \n", raiz->item.cod);
   mostrar(raiz->dir);
}
```

```
CONTEÚDO DA ÁRVORE:

**********************
Cod: 2
Nome: Biscoito
Quantidade: 200
**********************
Cod: 3
Nome: Farinha
Quantidade: 300
**********************
Cod: 4
Nome: Feijao
Quantidade: 400
```



```c
void mostrar(tipo_no *raiz){
   if(!raiz) return;
   mostrar(raiz->esq);
   printf("\n**********************");
   printf("\n Cod: %d  ", raiz->item.cod);
   printf("\n Nome: %s  ", raiz->item.nome);
   printf("\n Quantidade: %d  \n", raiz->item.cod);
   mostrar(raiz->dir);
}
```

```
CONTEÚDO DA ÁRVORE:

**********************
Cod: 2
Nome: Biscoito
Quantidade: 200
**********************
Cod: 3
Nome: Farinha
Quantidade: 300
**********************
Cod: 4
Nome: Feijao
Quantidade: 400
```

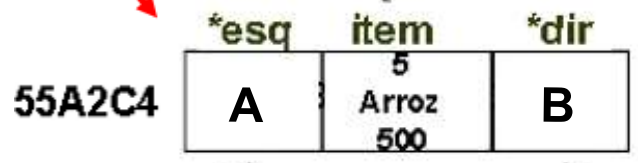


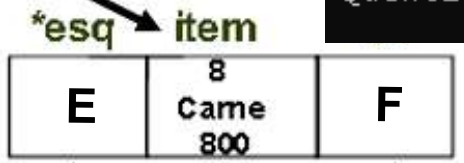
FAMEPI
Faculdade do Amazonas de Ensino, Pesquisa e Inovação
FUNDAÇÃO MATIAS MACHLINE

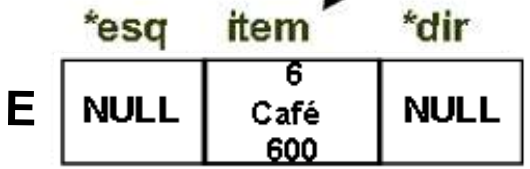
```c
void mostrar(tipo_no *raiz){
   if(!raiz) return;
   mostrar(raiz->esq);
   printf("\n**********************");
   printf("\n Cod: %d  ", raiz->item.cod);
   printf("\n Nome: %s  ", raiz->item.nome);
   printf("\n Quantidade: %d  \n", raiz->item.cod);
   mostrar(raiz->dir);
}
```

*rai
55A2C4

*raiz
55A2C4








```
CONTEÚDO DA ÁRVORE:

**********************
Cod: 2
Nome: Biscoito
Quantidade: 200
**********************
Cod: 3
Nome: Farinha
Quantidade: 300
**********************
Cod: 4
Nome: Feijao
Quantidade: 400
```

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

```c
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

*rai
55A2C4

*raiz
55A2C4

	*esq	item	*dir
55A2C4	A	5 Arroz 500	B
A	C	3 Farinha 300	D
B	E	8 Carne 800	
C	NULL	2 Biscoito 200	NULL
D	NULL	4 Feijão 400	NULL
E	NULL	6 Café 600	NULL
F	NULL	9 Leite 900	NULL

```
CONTEÚDO DA ÁRVORE:

**********************
Cod: 2
Nome: Biscoito
Quantidade: 200
**********************
Cod: 3
Nome: Farinha
Quantidade: 300
**********************
Cod: 4
Nome: Feijao
Quantidade: 400
**********************
Cod: 5
Nome: Arroz
Quantidade: 500
```

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

```
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

*rai
55A2C4

*raiz
55A2C4

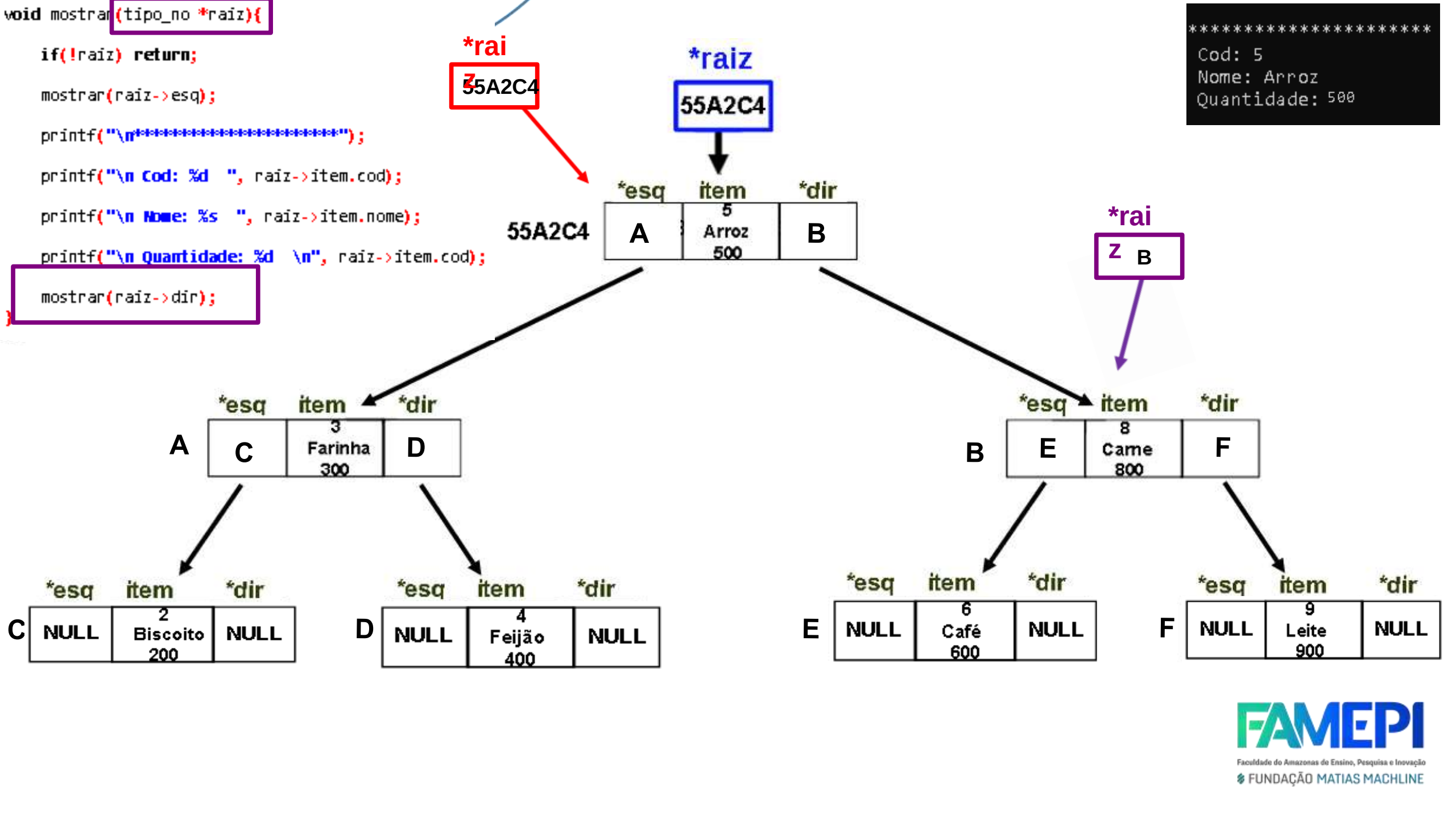
	*esq	item	*dir
55A2C4	A	5 Arroz 500	B
A	C	3 Farinha 300	D
B	E	8 Carne 800	
C	NULL	2 Biscoito 200	NULL
D	NULL	4 Feijão 400	NULL
E	NULL	6 Café 600	NULL
F	NULL	9 Leite 900	NULL

```
CONTEÚDO DA ÁRVORE:

**********************
Cod: 2
Nome: Biscoito
Quantidade: 200
**********************
Cod: 3
Nome: Farinha
Quantidade: 300
**********************
Cod: 4
Nome: Feijao
Quantidade: 400
**********************
Cod: 5
Nome: Arroz
Quantidade: 500
```

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

```
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

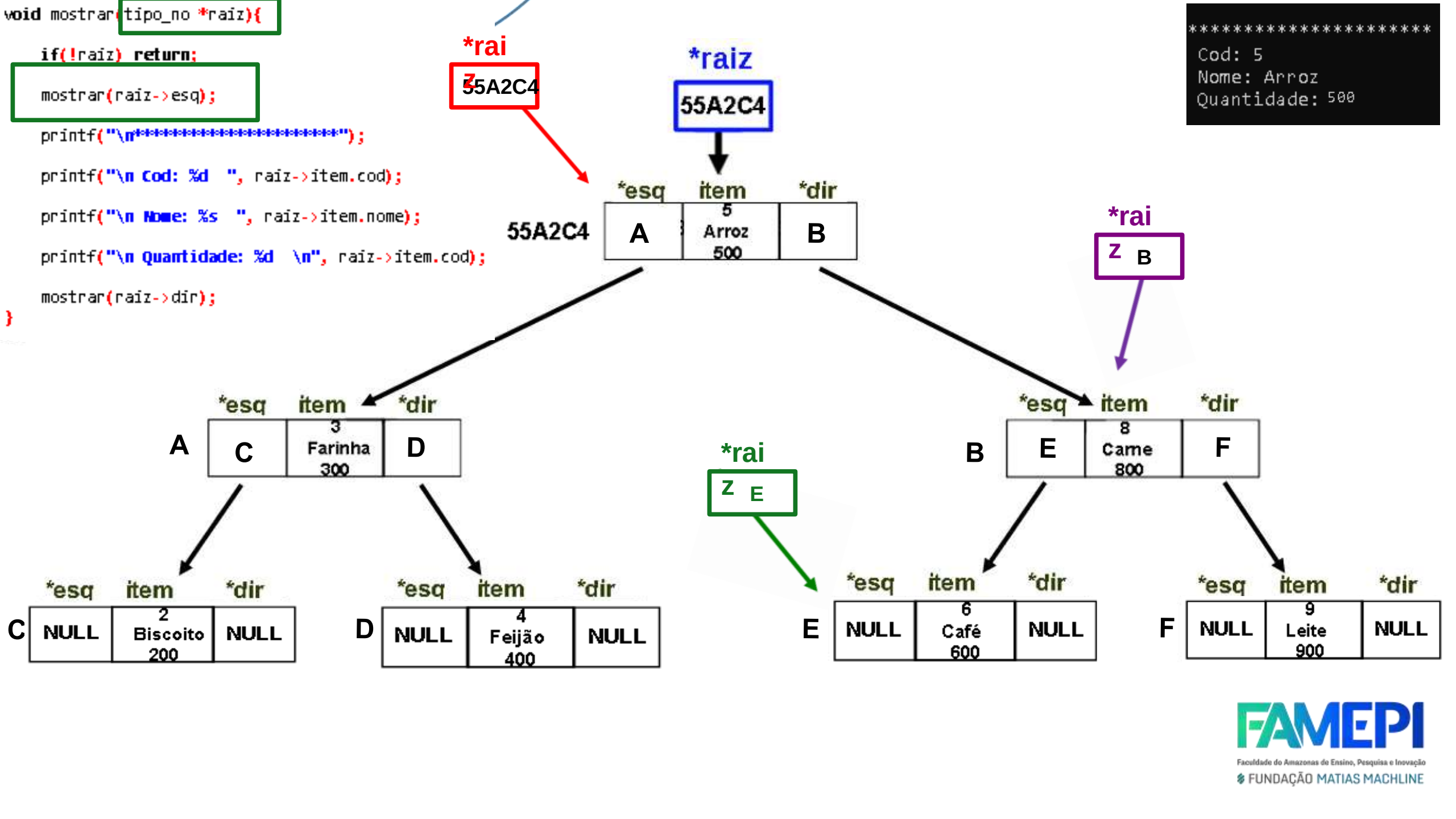
```
**********************
Cod: 5
Nome: Arroz
Quantidade: 500
```



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

```c
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

```
**********************
Cod: 5
Nome: Arroz
Quantidade: 500
```



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

```c
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

```
**********************
Cod: 5
Nome: Arroz
Quantidade: 500
```

*rai
Z 55A2C4

*raiz
55A2C4

	*esq	item	*dir
55A2C4	A	5 Arroz 500	B

*rai
Z B

	*esq	item	*dir
A	C	3 Farinha 300	D

*rai
Z E

	*esq	item	*dir
B	E	8 Carne 800	F

	*esq	item	*dir
C	NULL	2 Biscoito 200	NULL

	*esq	item	*dir
D	NULL	4 Feijão 400	NULL

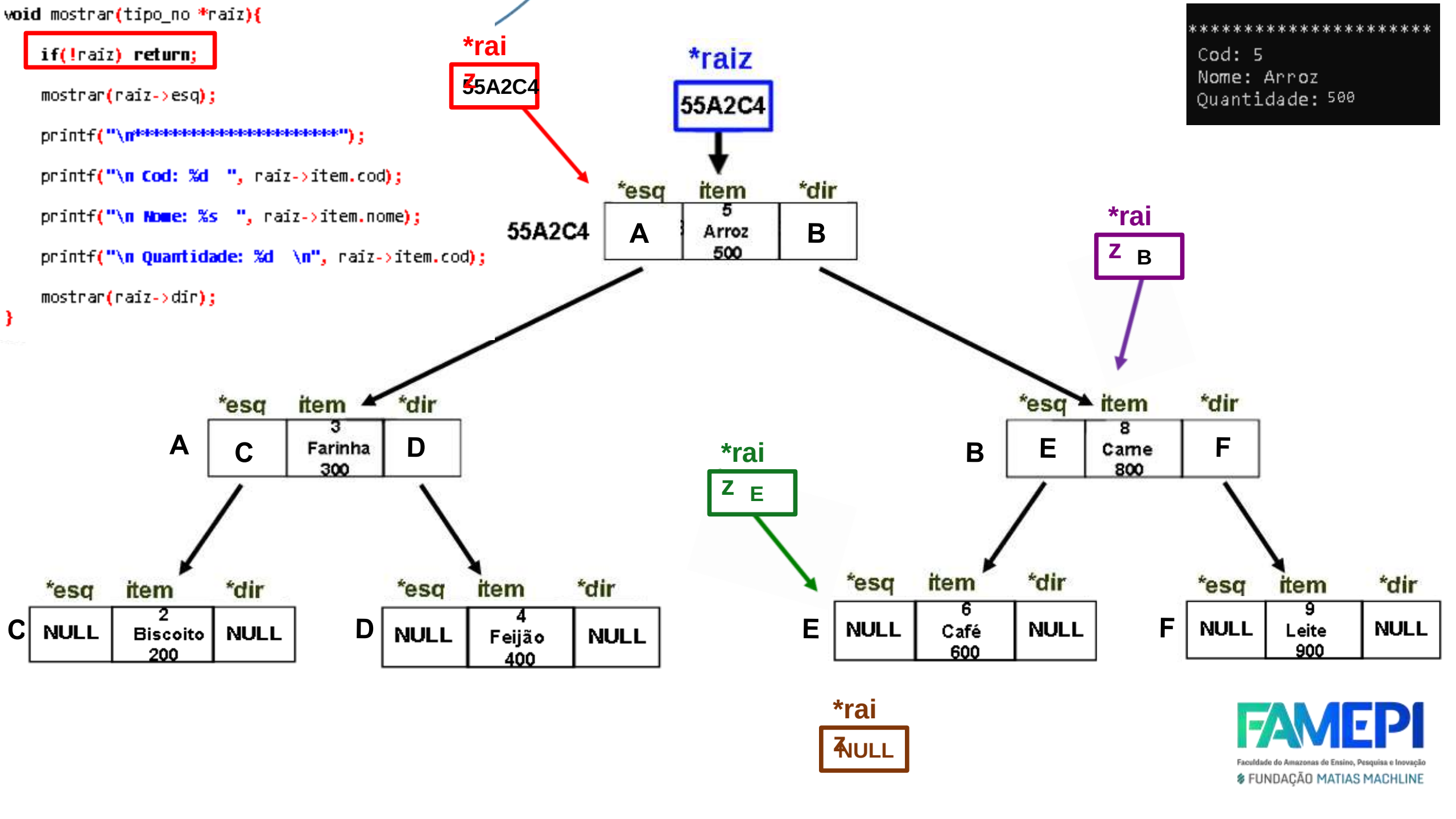
	*esq	item	*dir
E	NULL	6 Café 600	NULL

	*esq	item	*dir
F	NULL	9 Leite 900	NULL

*rai
Z NULL

```c
void mostrar(tipo_no *raiz){
   if(!raiz) return;
   mostrar(raiz->esq);
   printf("\n**********************");
   printf("\n Cod: %d  ", raiz->item.cod);
   printf("\n Nome: %s  ", raiz->item.nome);
   printf("\n Quantidade: %d  \n", raiz->item.cod);
   mostrar(raiz->dir);
}
```

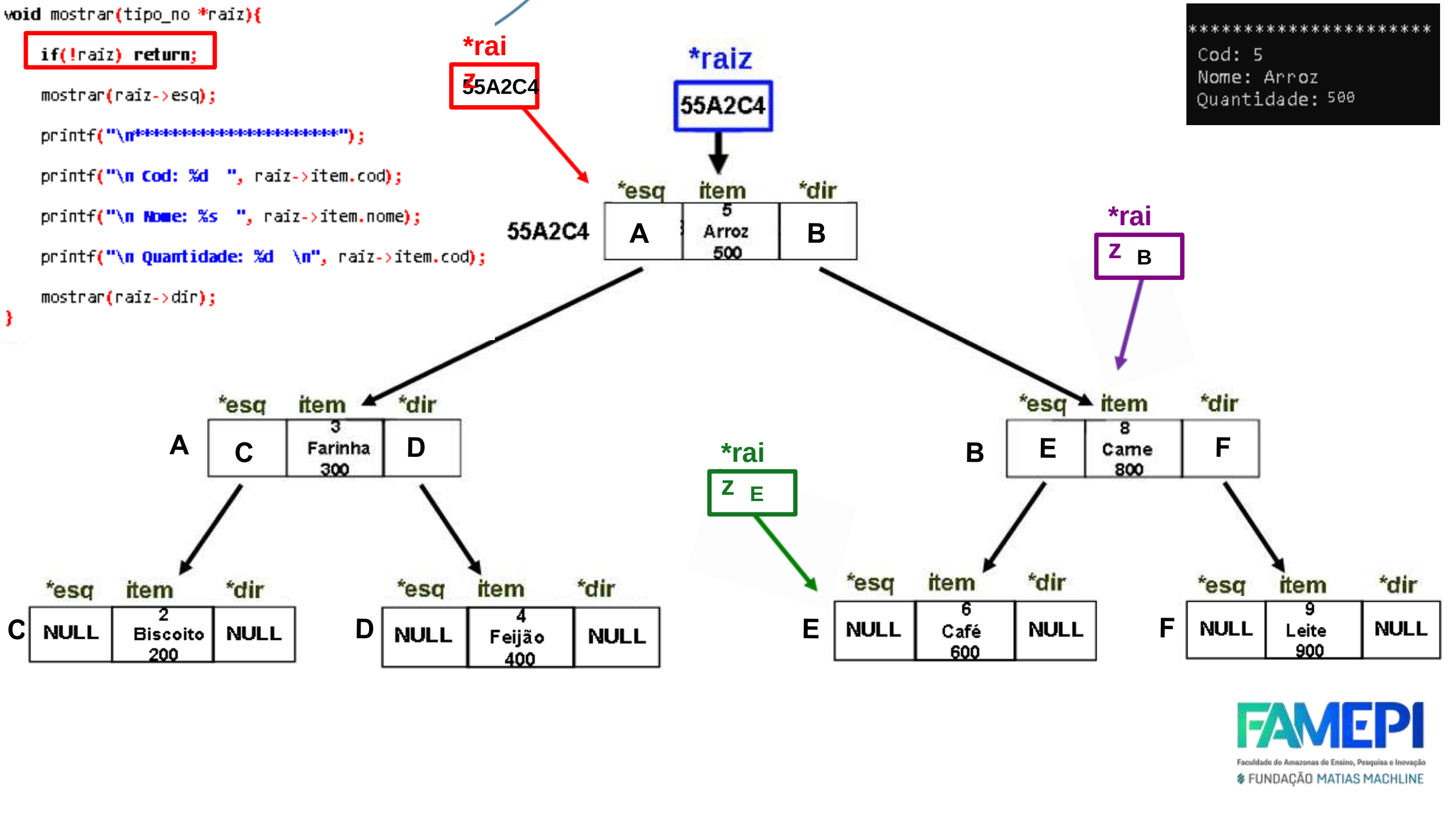
```
**********************
Cod: 5
Nome: Arroz
Quantidade: 500
```



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

```c
void mostrar(tipo_no *raiz){
   if(!raiz) return;
   mostrar(raiz->esq);
   printf("\n**********************");
   printf("\n Cod: %d  ", raiz->item.cod);
   printf("\n Nome: %s  ", raiz->item.nome);
   printf("\n Quantidade: %d  \n", raiz->item.cod);
   mostrar(raiz->dir);
}
```

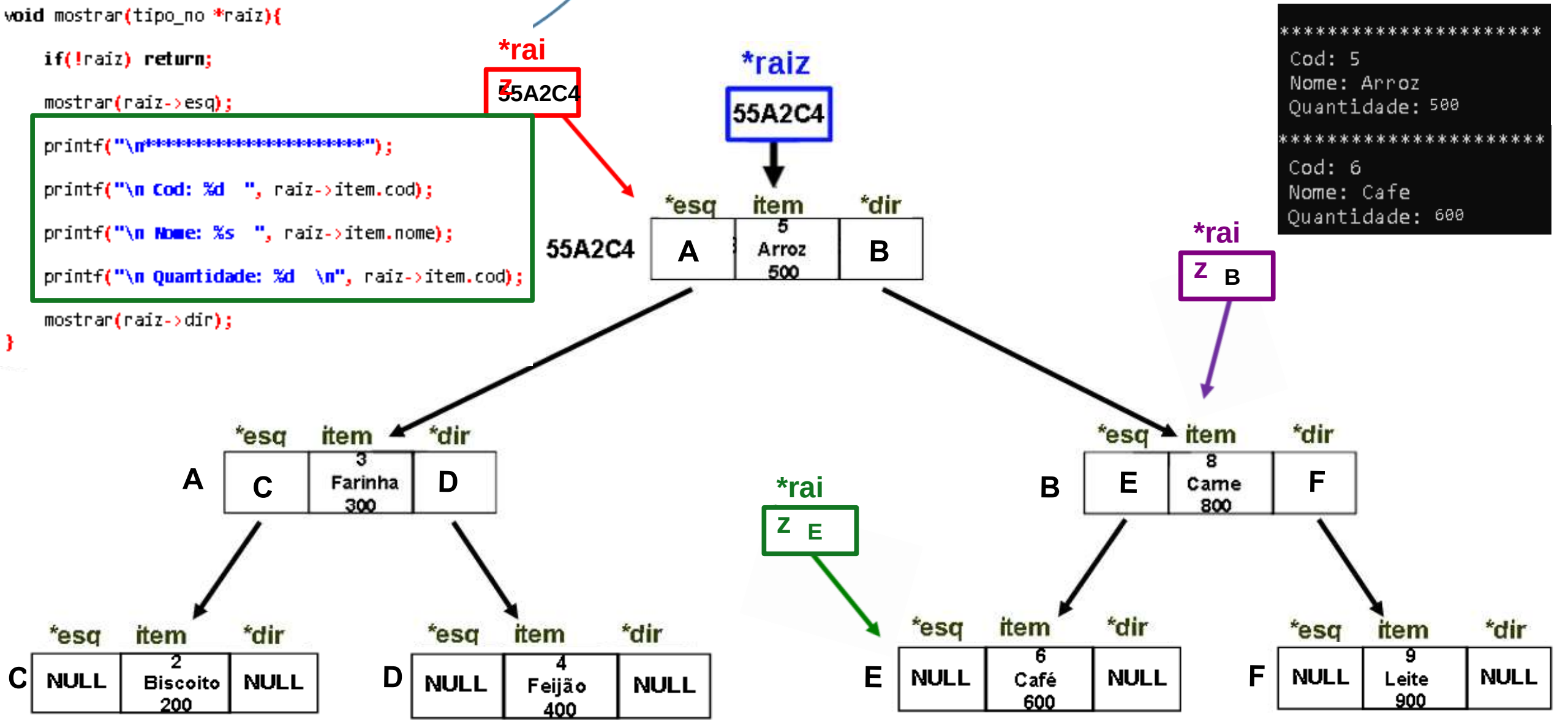
```
**********************
Cod: 5
Nome: Arroz
Quantidade: 500
```



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

```c
void mostrar(tipo_no *raiz){

    if(!raiz) return;

    mostrar(raiz->esq);

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

    printf("\n Cod: %d  ", raiz->item.cod);

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

    printf("\n Quantidade: %d  \n", raiz->item.cod);

    mostrar(raiz->dir);
}
```

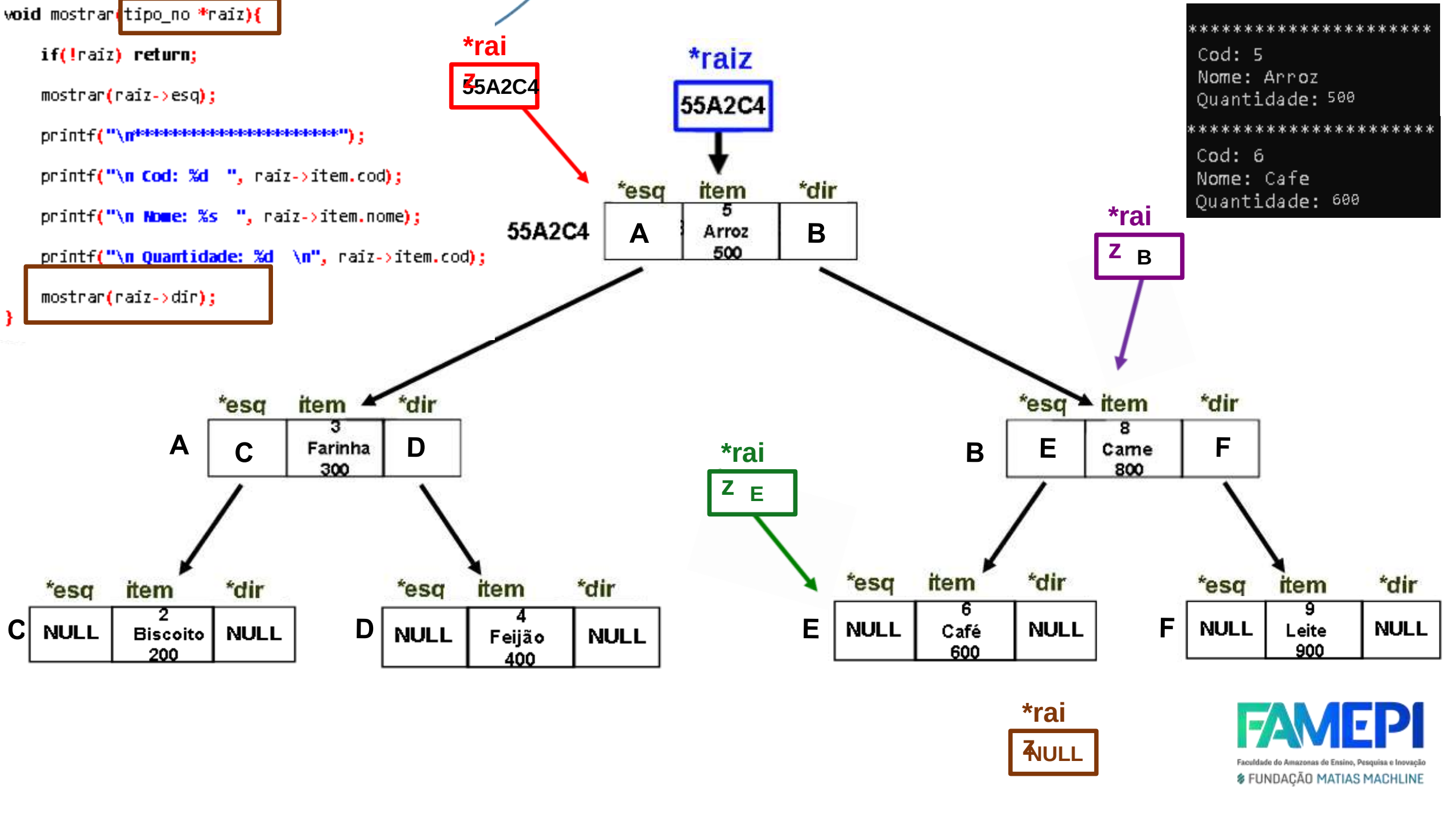
```
**********************
 Cod: 5
 Nome: Arroz
 Quantidade: 500
**********************
 Cod: 6
 Nome: Cafe
 Quantidade: 600
```



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

```
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

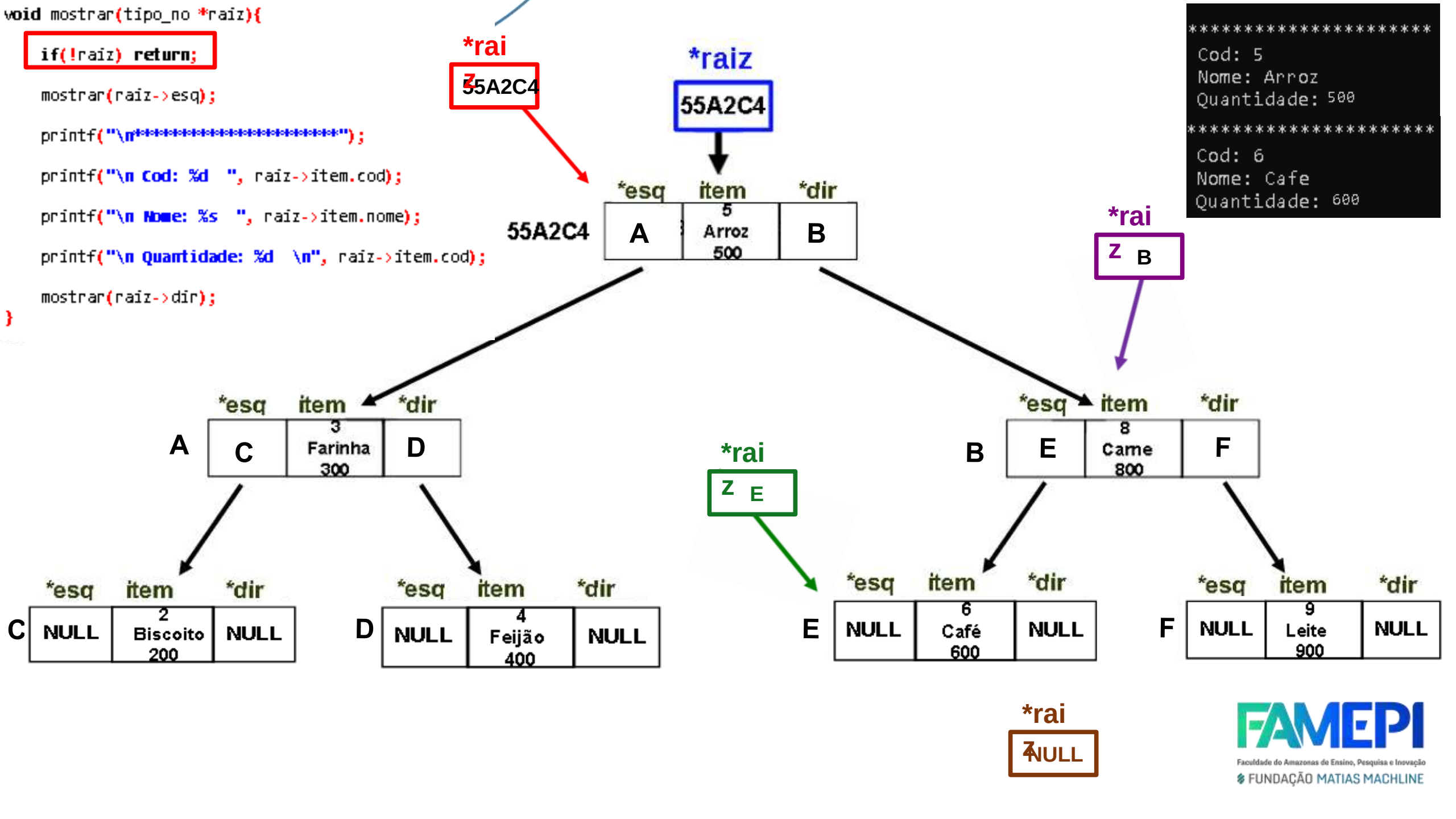
```
**********************
 Cod: 5
 Nome: Arroz
 Quantidade: 500
**********************
 Cod: 6
 Nome: Cafe
 Quantidade: 600
```



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

```c
void mostrar(tipo_no *raiz){
   if(!raiz) return;
   mostrar(raiz->esq);
   printf("\n**********************");
   printf("\n Cod: %d  ", raiz->item.cod);
   printf("\n Nome: %s  ", raiz->item.nome);
   printf("\n Quantidade: %d  \n", raiz->item.cod);
   mostrar(raiz->dir);
}
```

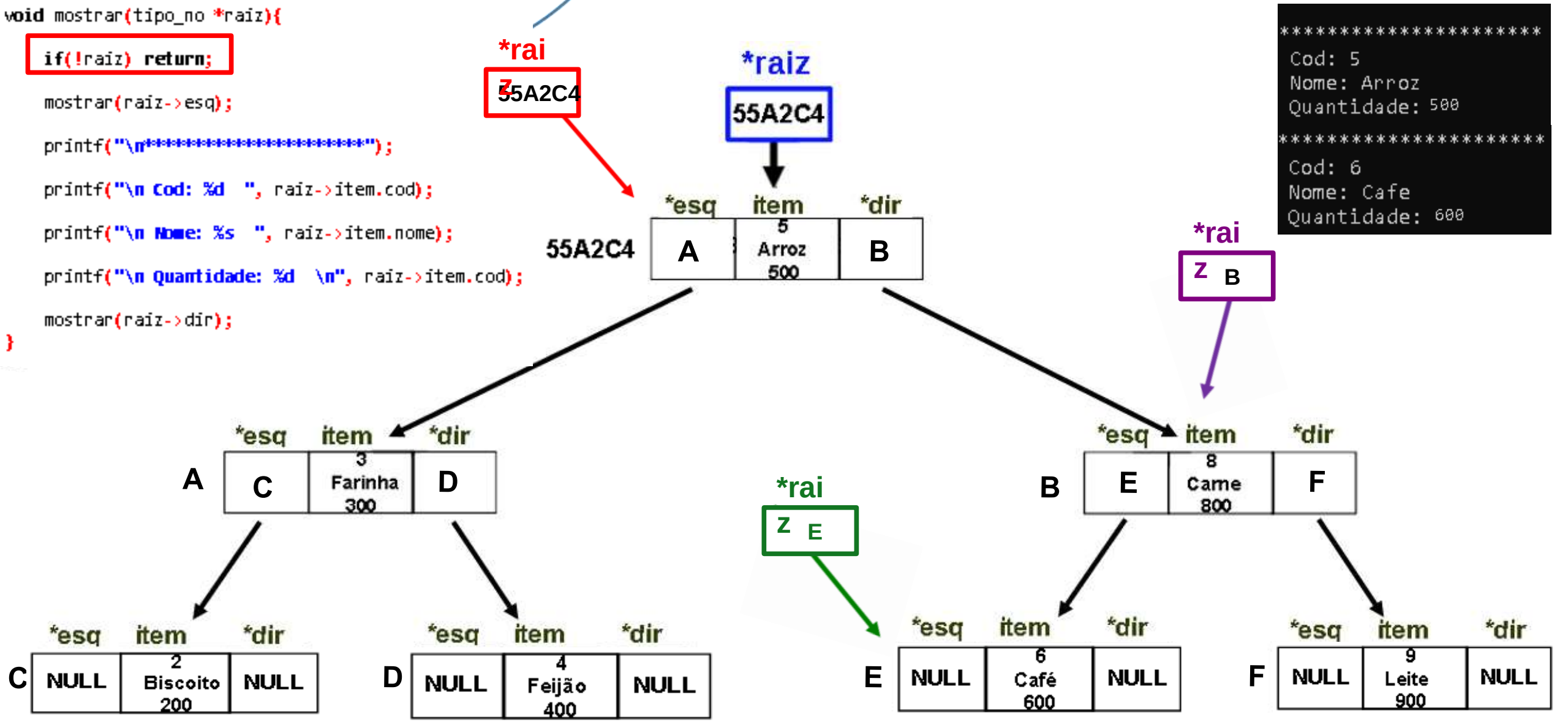
```
**********************
 Cod: 5
 Nome: Arroz
 Quantidade: 500
**********************
 Cod: 6
 Nome: Cafe
 Quantidade: 600
```



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

```c
void mostrar(tipo_no *raiz){
   if(!raiz) return;
   mostrar(raiz->esq);
   printf("\n**********************");
   printf("\n Cod: %d  ", raiz->item.cod);
   printf("\n Nome: %s  ", raiz->item.nome);
   printf("\n Quantidade: %d  \n", raiz->item.cod);
   mostrar(raiz->dir);
}
```

```
**********************
 Cod: 5
 Nome: Arroz
 Quantidade: 500
**********************
 Cod: 6
 Nome: Cafe
 Quantidade: 600
```



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

```c
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

*rai
z 55A2C4

*raiz
55A2C4

	*esq	item	*dir
55A2C4	A	5 Arroz 500	B

	*esq	item	*dir
A	C	3 Farinha 300	D

	*esq	item	*dir
C	NULL	2 Biscoito 200	NULL

	*esq	item	*dir
D	NULL	4 Feijão 400	NULL

*rai
z B

	*esq	item	*dir
B	E	8 Carne 800	F

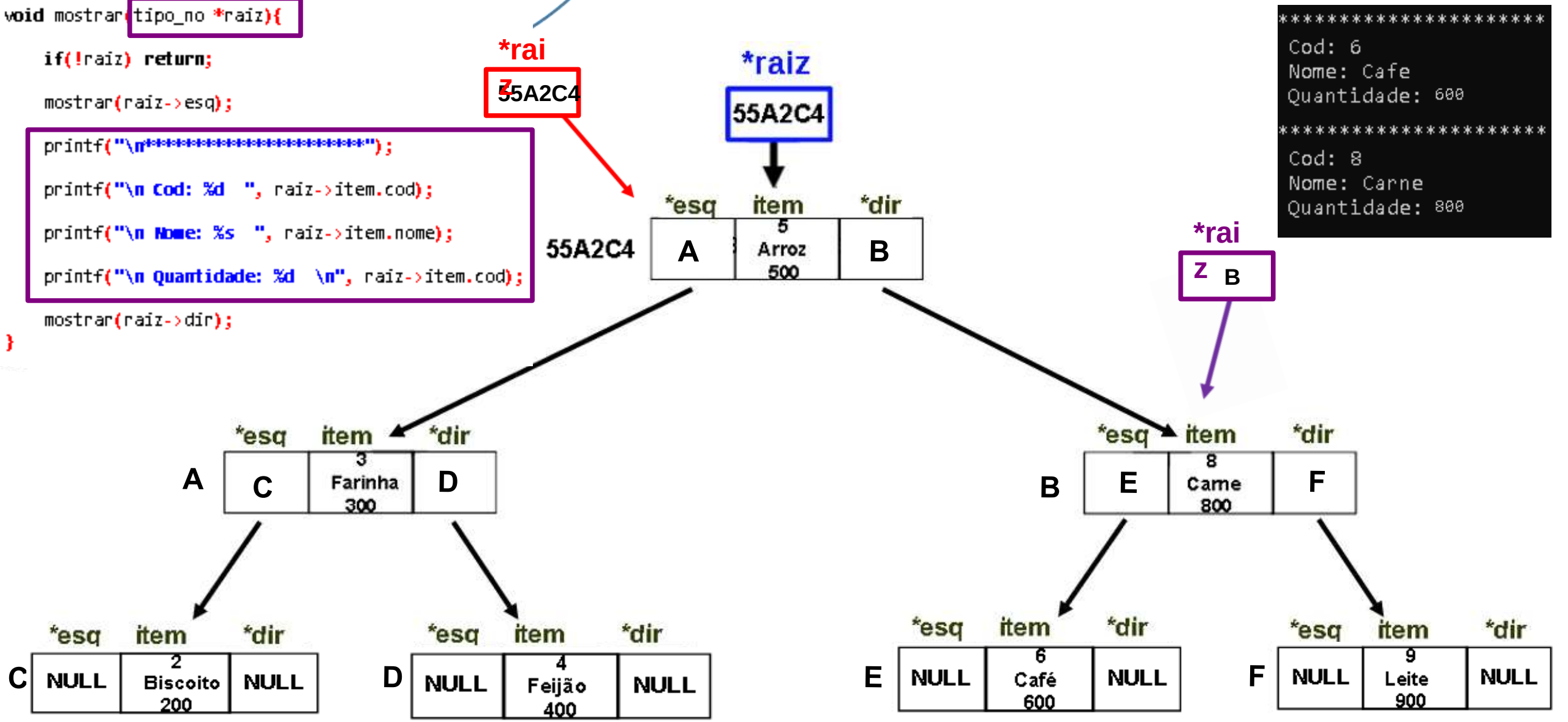
	*esq	item	*dir
E	NULL	6 Café 600	NULL

	*esq	item	*dir
F	NULL	9 Leite 900	NULL

```
**********************
 Cod: 5
 Nome: Arroz
 Quantidade: 500
**********************
 Cod: 6
 Nome: Cafe
 Quantidade: 600
**********************
 Cod: 8
 Nome: Carne
 Quantidade: 800
```

```
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

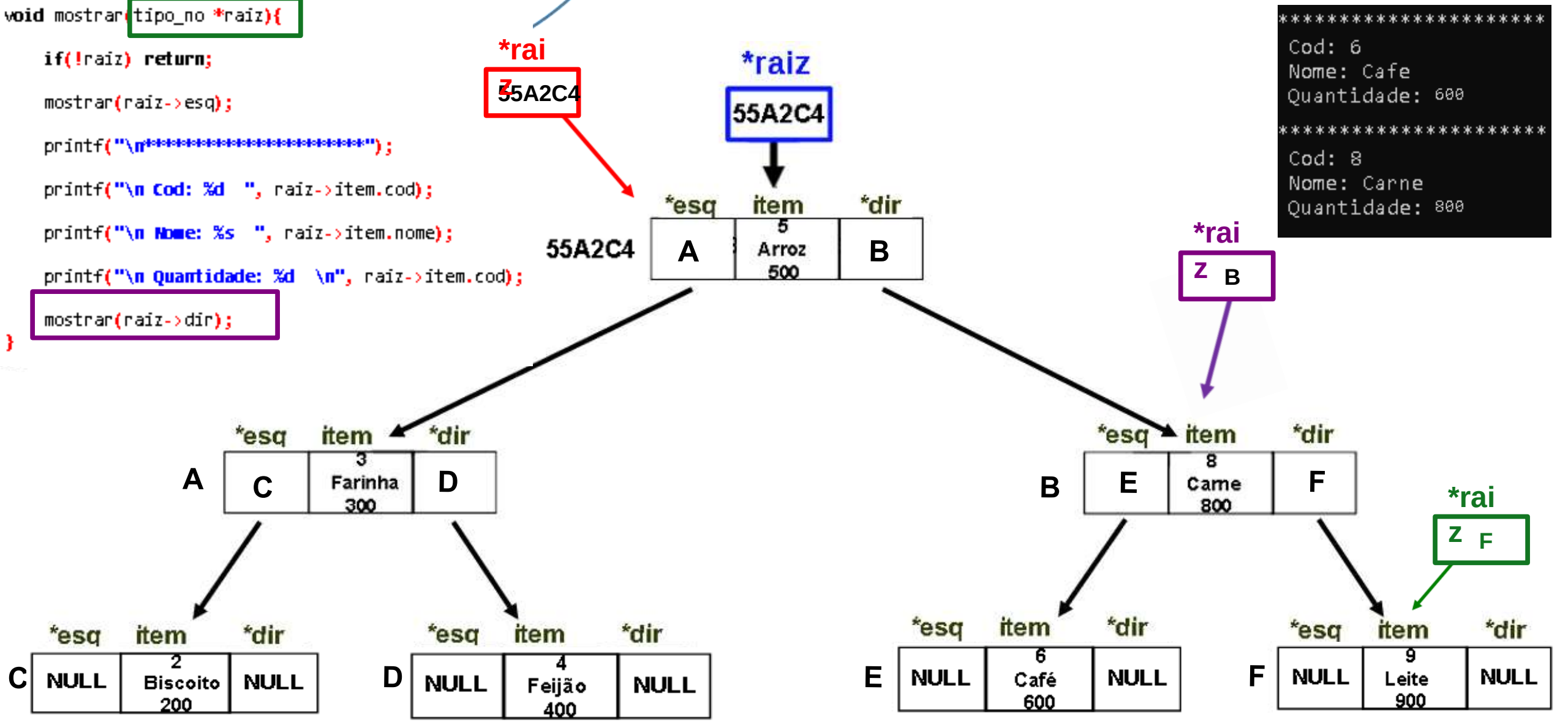
```
**********************
Cod: 6
Nome: Cafe
Quantidade: 600
**********************
Cod: 8
Nome: Carne
Quantidade: 800
```



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

```c
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

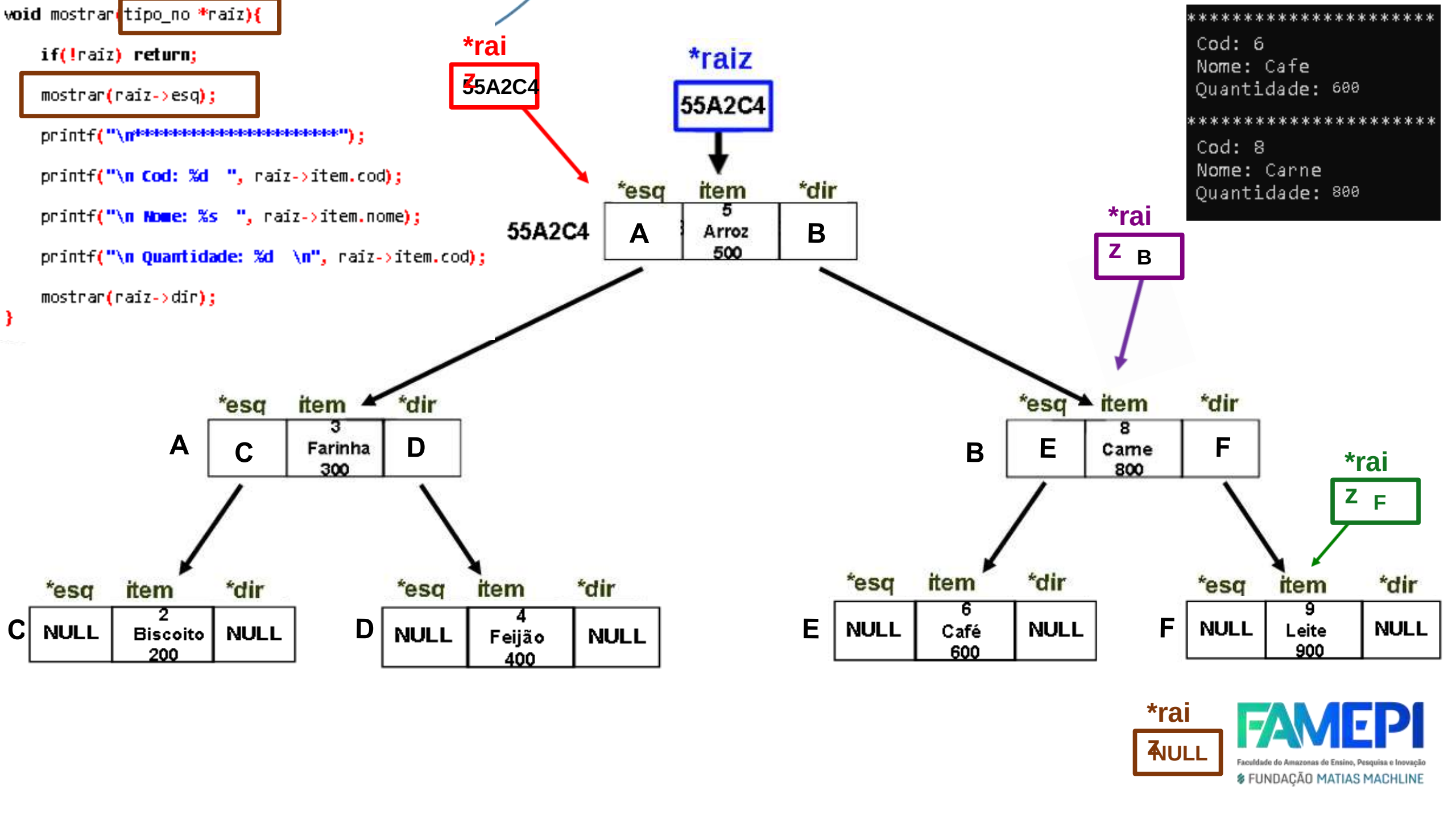
```
**********************
Cod: 6
Nome: Cafe
Quantidade: 600
**********************
Cod: 8
Nome: Carne
Quantidade: 800
```



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

```c
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

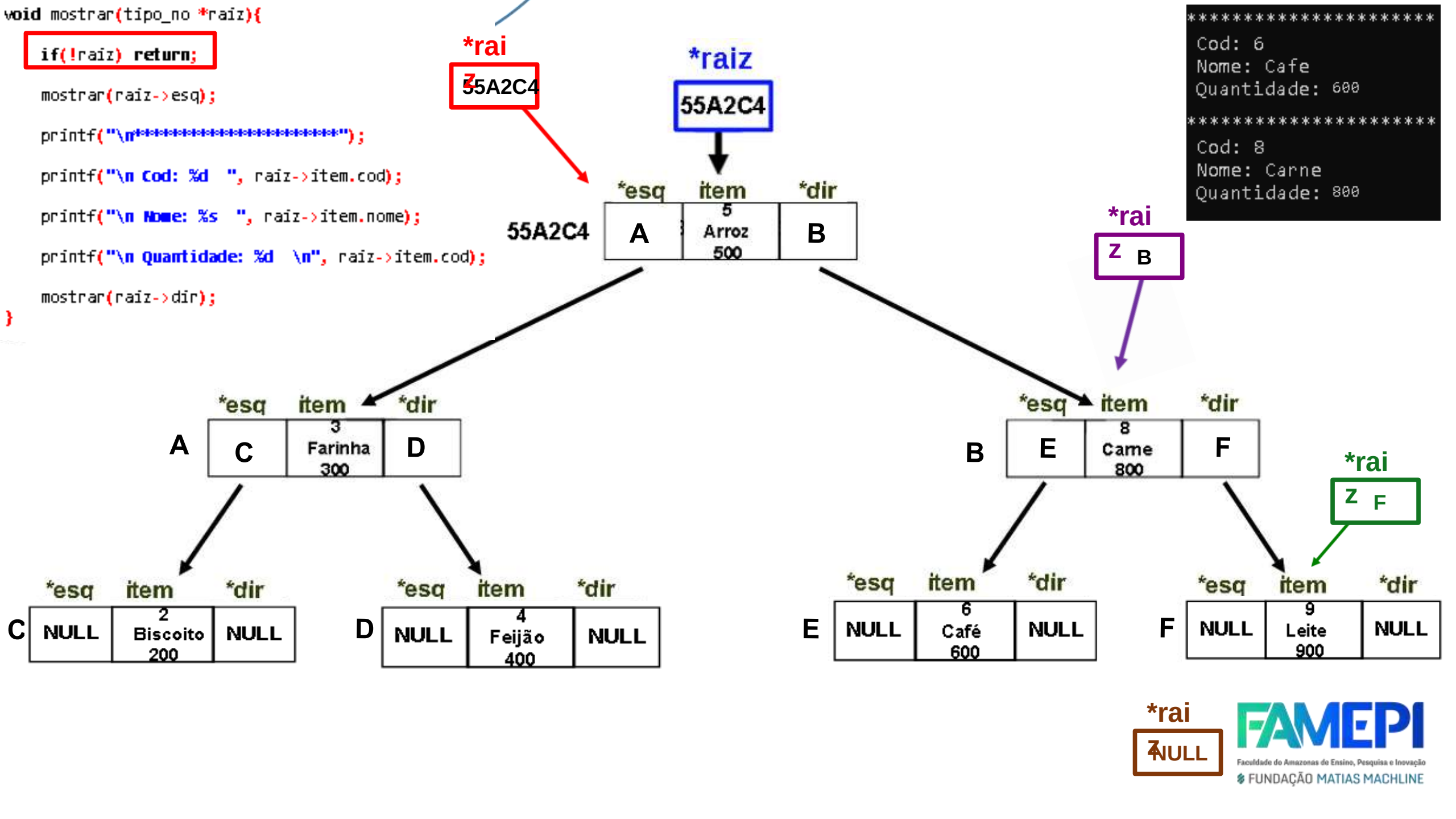
```
**********************
Cod: 6
Nome: Cafe
Quantidade: 600
**********************
Cod: 8
Nome: Carne
Quantidade: 800
```



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

```c
void mostrar(tipo_no *raiz){
   if(!raiz) return;
   mostrar(raiz->esq);
   printf("\n**********************");
   printf("\n Cod: %d  ", raiz->item.cod);
   printf("\n Nome: %s  ", raiz->item.nome);
   printf("\n Quantidade: %d  \n", raiz->item.cod);
   mostrar(raiz->dir);
}
```

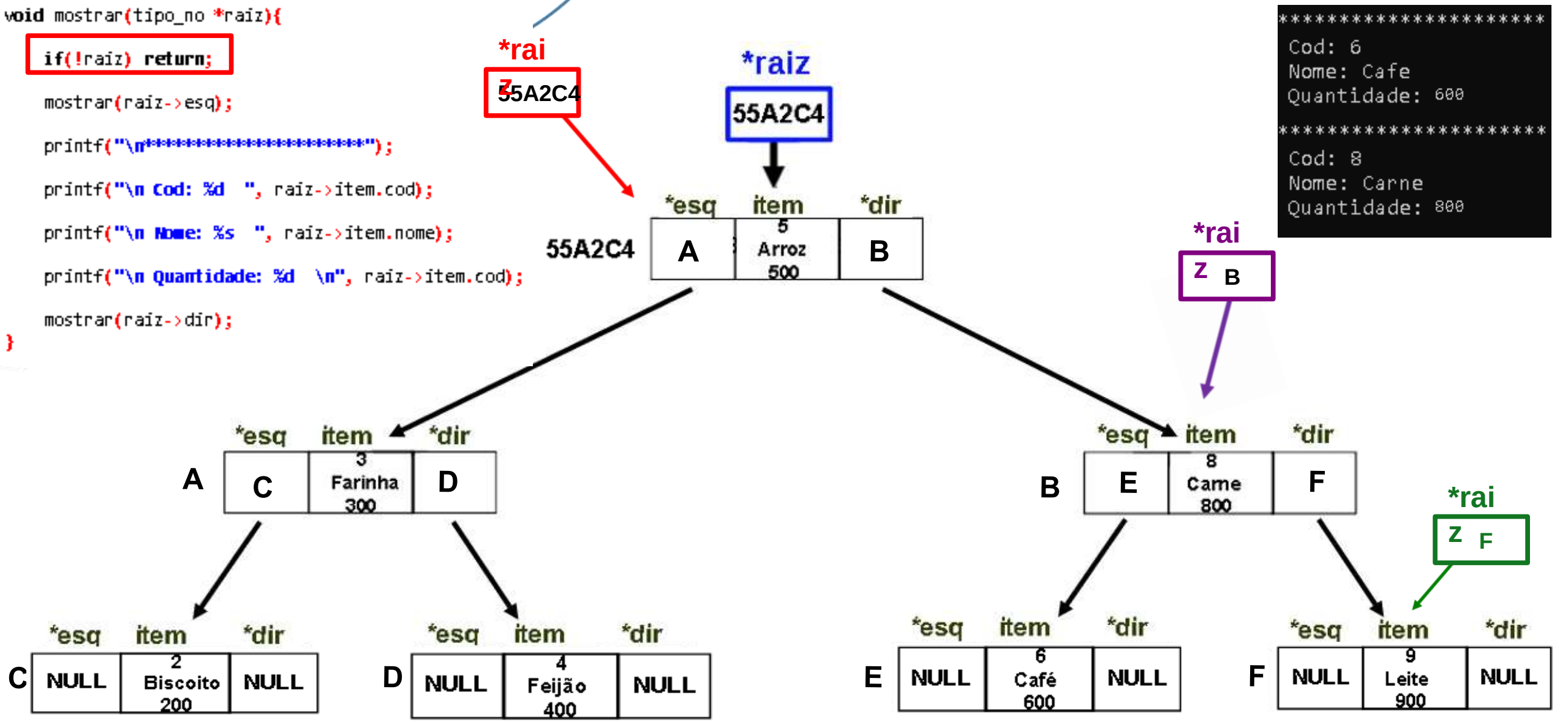
```
**********************
Cod: 6
Nome: Cafe
Quantidade: 600
**********************
Cod: 8
Nome: Carne
Quantidade: 800
```



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

```c
void mostrar(tipo_no *raiz){
   if(!raiz) return;
   mostrar(raiz->esq);
   printf("\n**********************");
   printf("\n Cod: %d  ", raiz->item.cod);
   printf("\n Nome: %s  ", raiz->item.nome);
   printf("\n Quantidade: %d  \n", raiz->item.cod);
   mostrar(raiz->dir);
}
```

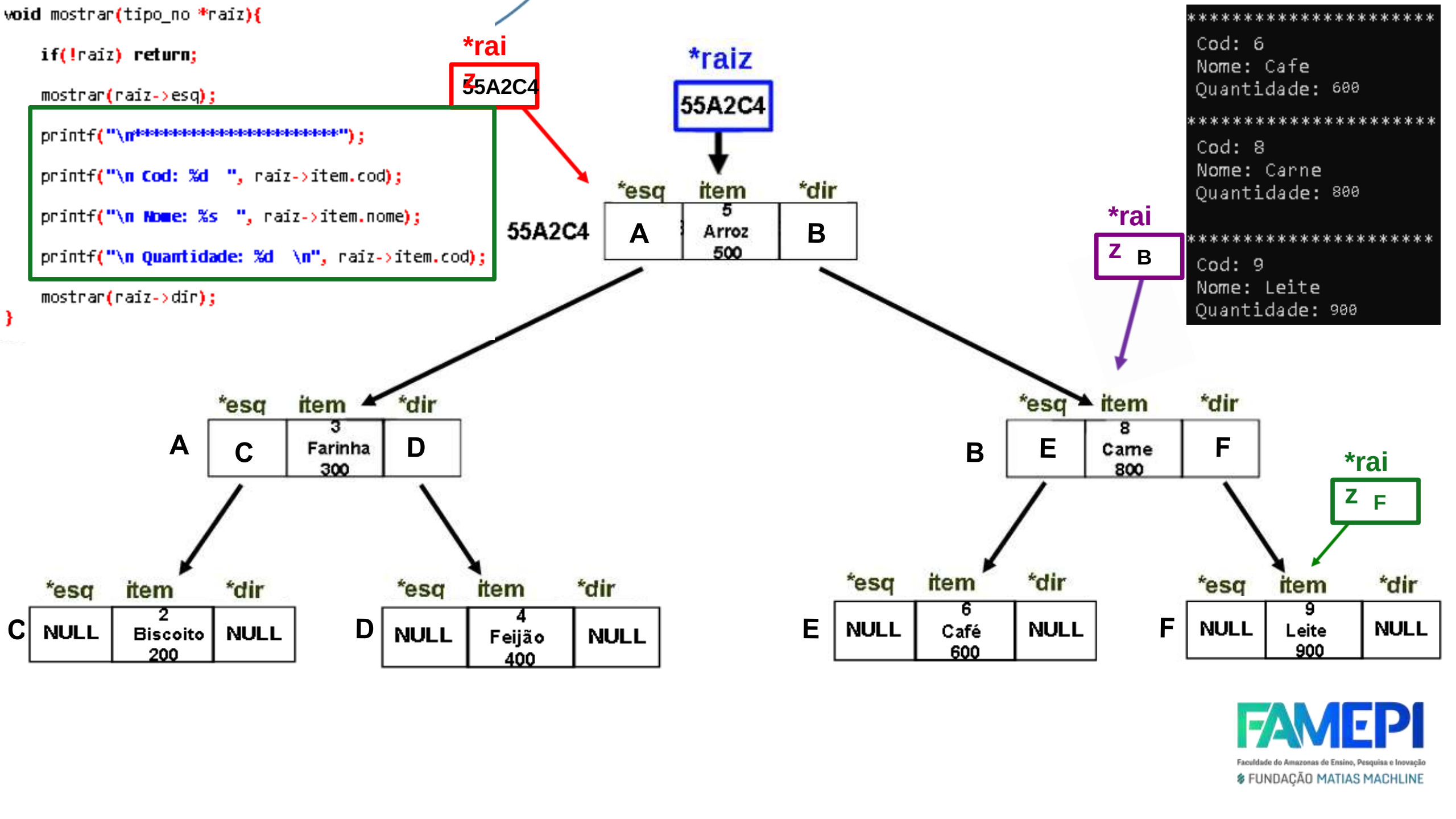
```
**********************
Cod: 6
Nome: Cafe
Quantidade: 600
**********************
Cod: 8
Nome: Carne
Quantidade: 800
```



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

```c
void mostrar(tipo_no *raiz){

    if(!raiz) return;

    mostrar(raiz->esq);

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

    printf("\n Cod: %d  ", raiz->item.cod);

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

    printf("\n Quantidade: %d  \n", raiz->item.cod);

    mostrar(raiz->dir);
}
```

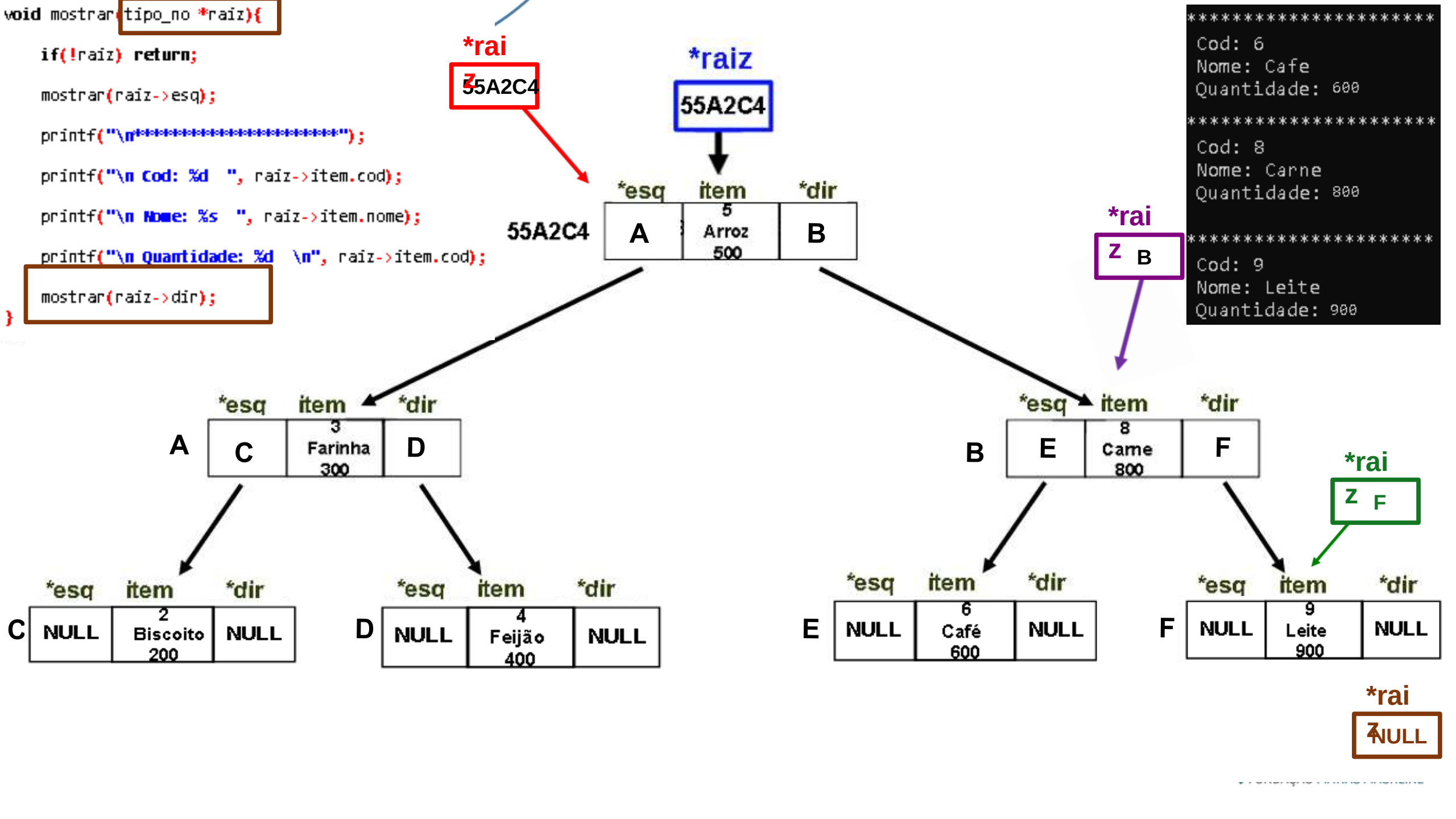
```
**********************
Cod: 6
Nome: Cafe
Quantidade: 600
**********************
Cod: 8
Nome: Carne
Quantidade: 800
**********************
Cod: 9
Nome: Leite
Quantidade: 900
```



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

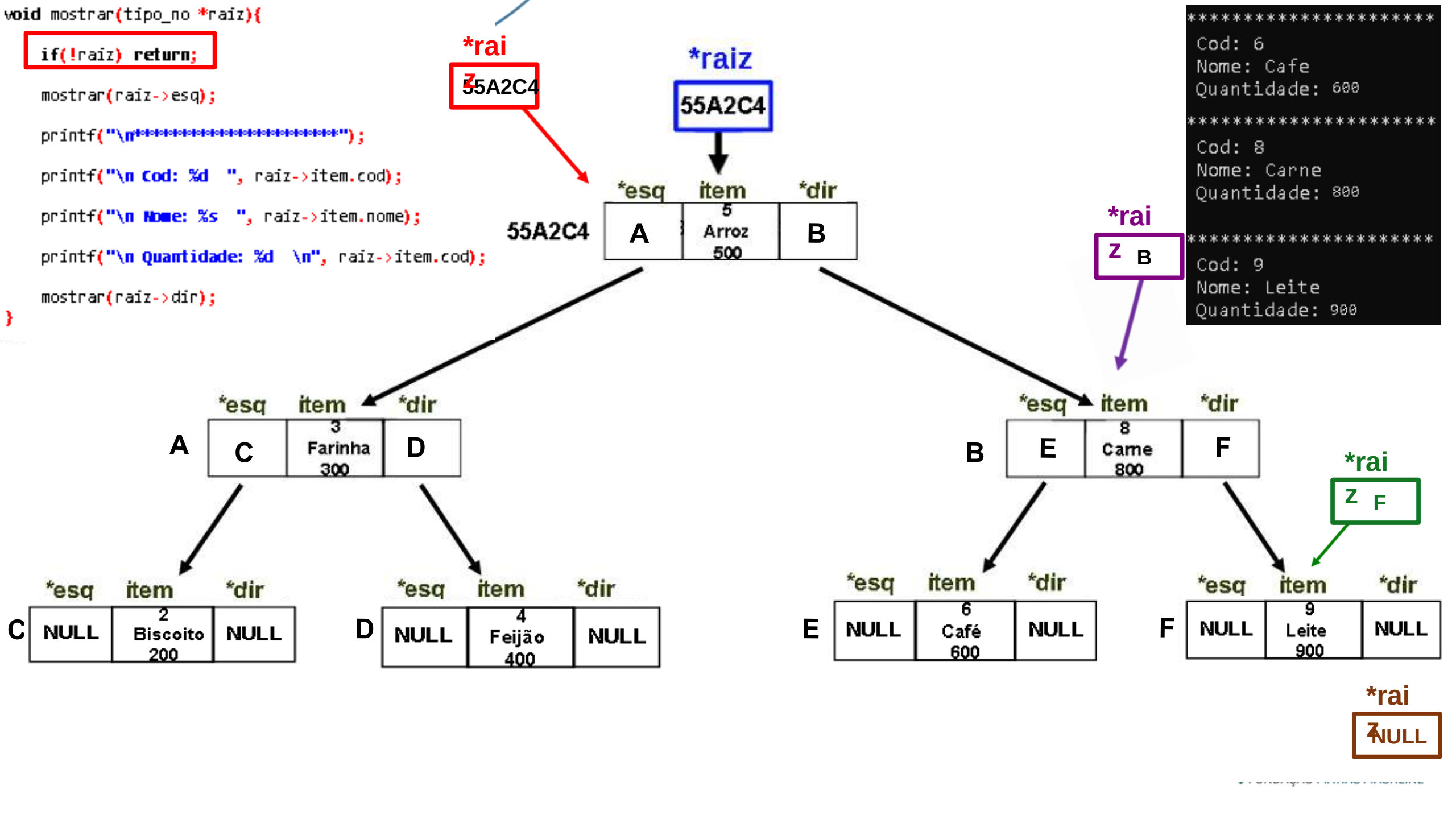
```
void mostrar(tipo_no *raiz){
    if(!raiz) return;
    mostrar(raiz->esq);
    printf("\n**********************");
    printf("\n Cod: %d  ", raiz->item.cod);
    printf("\n Nome: %s  ", raiz->item.nome);
    printf("\n Quantidade: %d  \n", raiz->item.cod);
    mostrar(raiz->dir);
}
```

```
**********************
Cod: 6
Nome: Cafe
Quantidade: 600
**********************
Cod: 8
Nome: Carne
Quantidade: 800
**********************
Cod: 9
Nome: Leite
Quantidade: 900
```



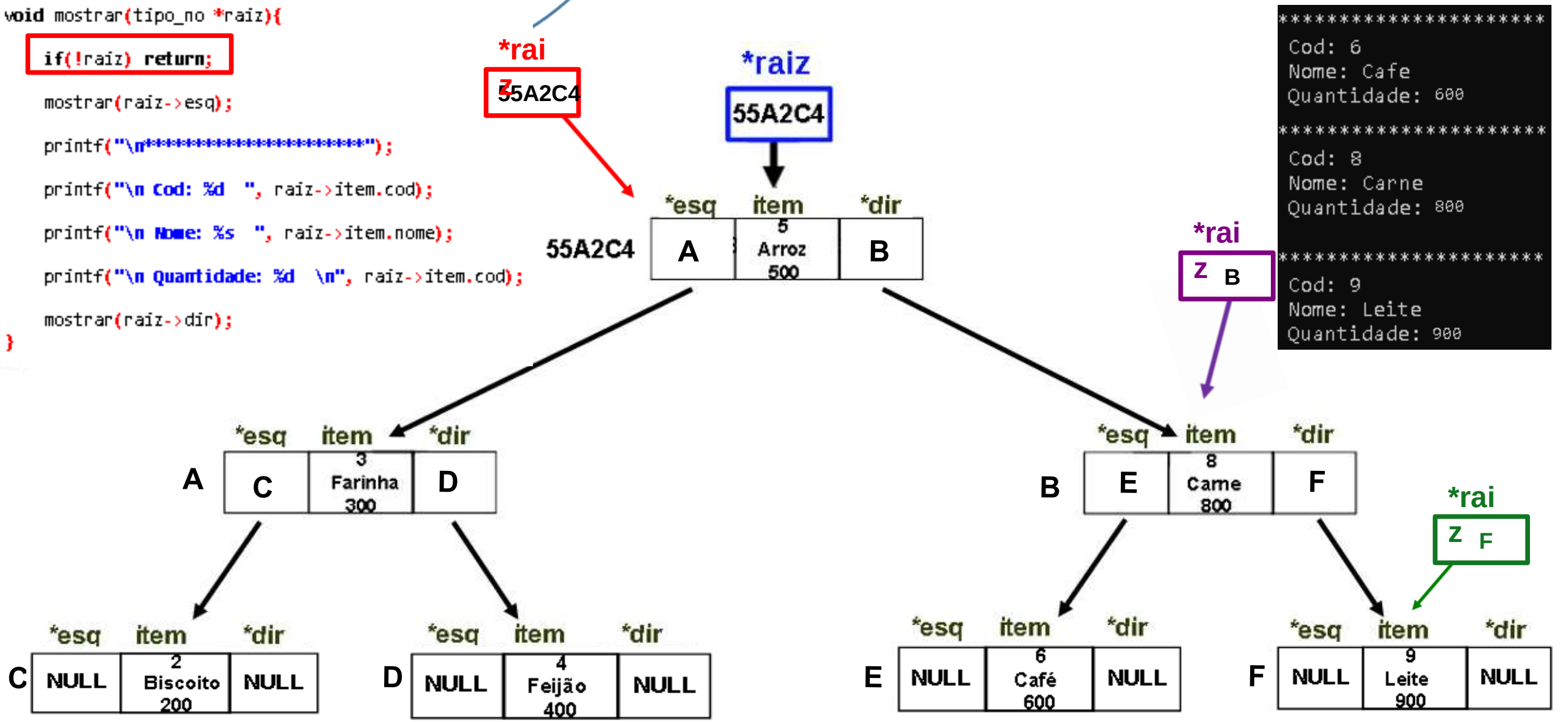
```c
void mostrar(tipo_no *raiz){
   if(!raiz) return;
   mostrar(raiz->esq);
   printf("\n**********************");
   printf("\n Cod: %d  ", raiz->item.cod);
   printf("\n Nome: %s  ", raiz->item.nome);
   printf("\n Quantidade: %d  \n", raiz->item.cod);
   mostrar(raiz->dir);
}
```

```
**********************
Cod: 6
Nome: Cafe
Quantidade: 600
**********************
Cod: 8
Nome: Carne
Quantidade: 800
**********************
Cod: 9
Nome: Leite
Quantidade: 900
```



```c
void mostrar(tipo_no *raiz){
   if(!raiz) return;
   mostrar(raiz->esq);
   printf("\n**********************");
   printf("\n Cod: %d  ", raiz->item.cod);
   printf("\n Nome: %s  ", raiz->item.nome);
   printf("\n Quantidade: %d  \n", raiz->item.cod);
   mostrar(raiz->dir);
}
```

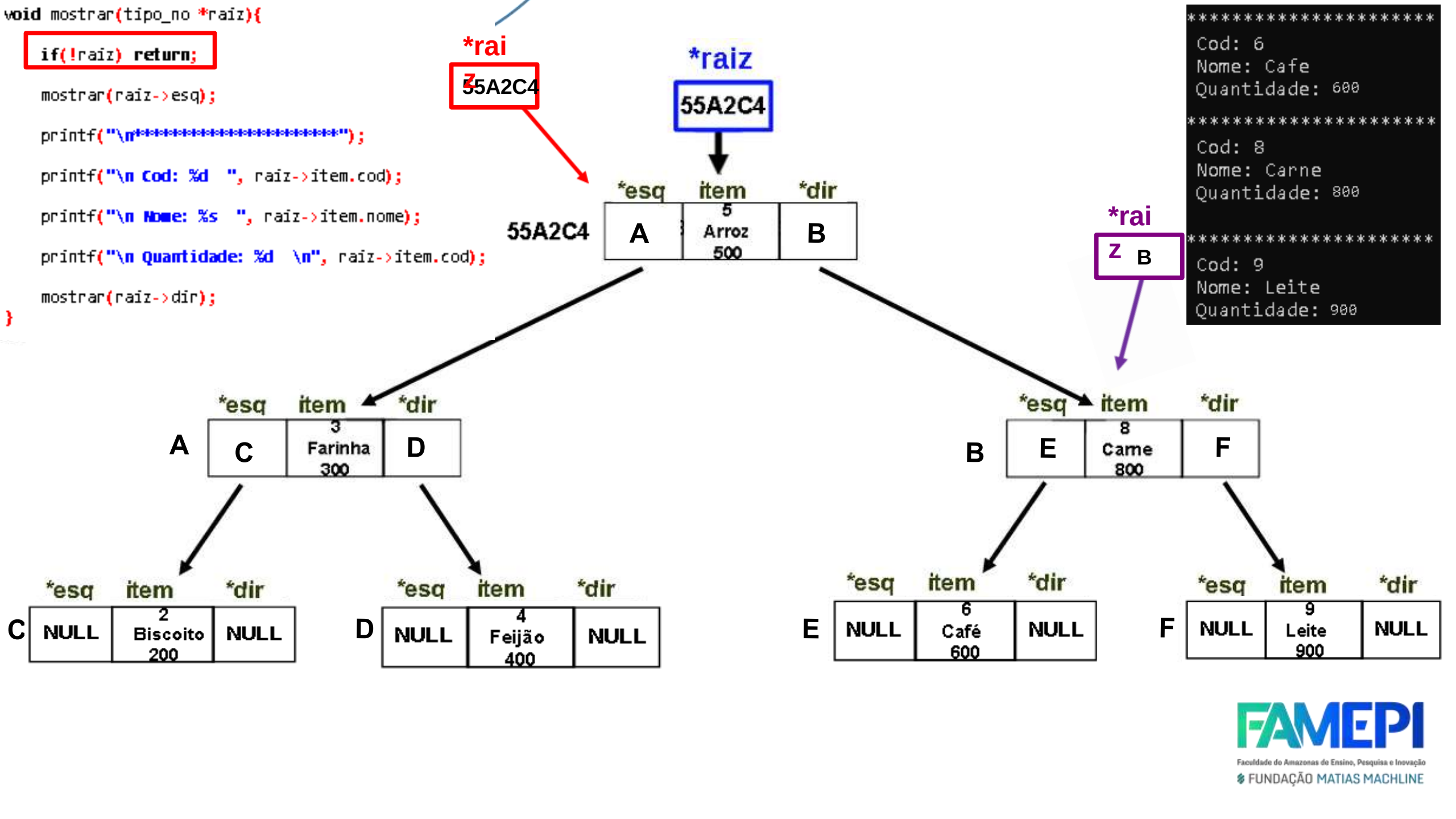
```
**********************
Cod: 6
Nome: Cafe
Quantidade: 600
**********************
Cod: 8
Nome: Carne
Quantidade: 800
**********************
Cod: 9
Nome: Leite
Quantidade: 900
```



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

```c
void mostrar(tipo_no *raiz){
   if(!raiz) return;
   mostrar(raiz->esq);
   printf("\n**********************");
   printf("\n Cod: %d  ", raiz->item.cod);
   printf("\n Nome: %s  ", raiz->item.nome);
   printf("\n Quantidade: %d  \n", raiz->item.cod);
   mostrar(raiz->dir);
}
```

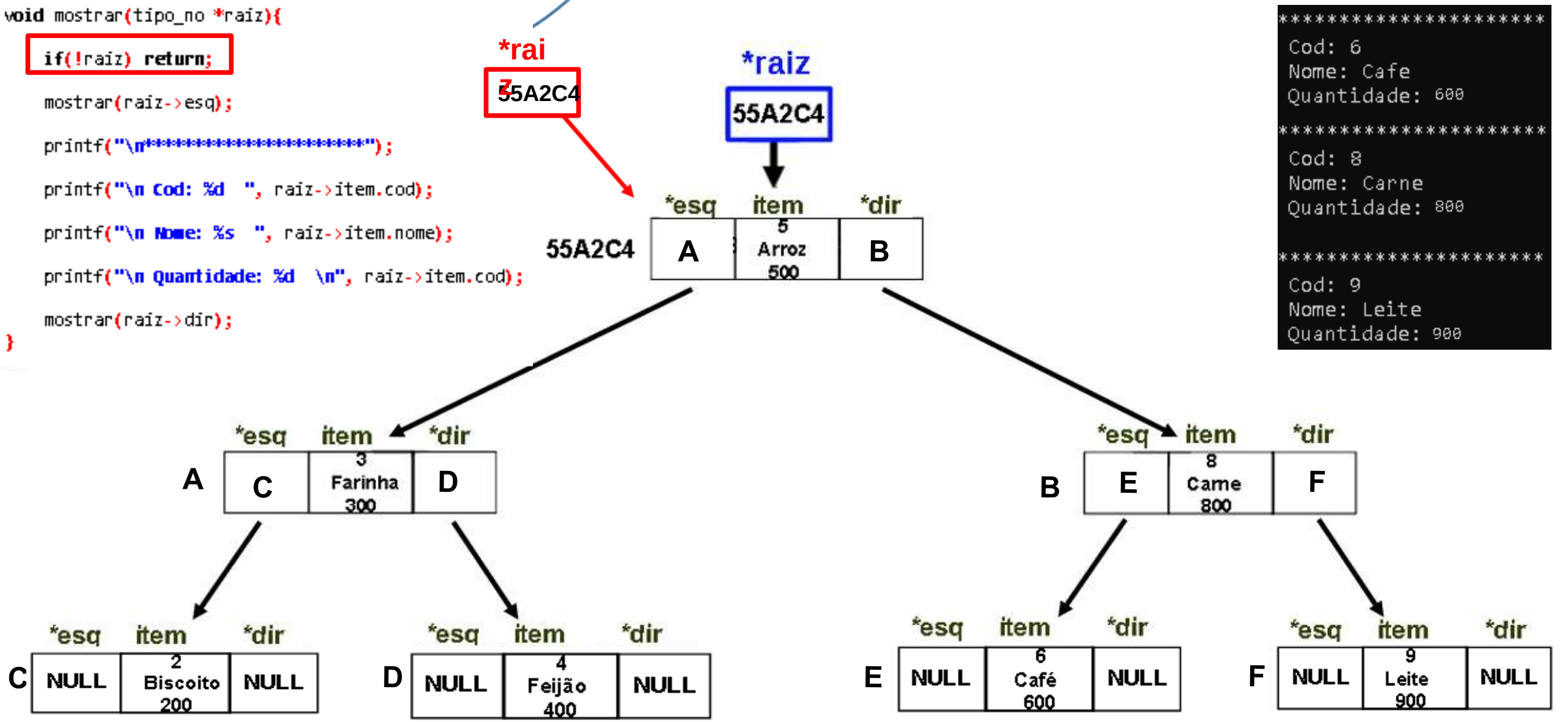
```
**********************
Cod: 6
Nome: Cafe
Quantidade: 600
**********************
Cod: 8
Nome: Carne
Quantidade: 800
**********************
Cod: 9
Nome: Leite
Quantidade: 900
```



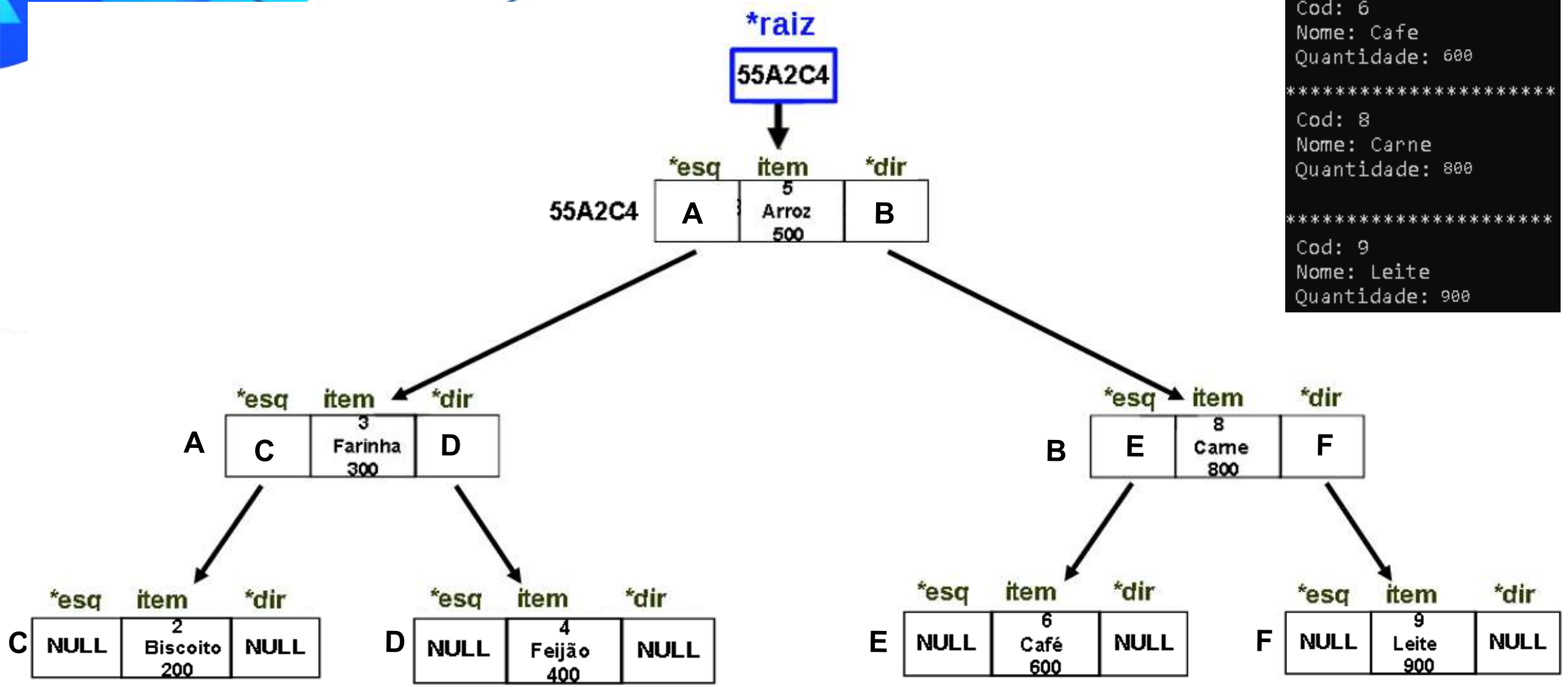
FAMEPI
Faculdade do Amazonas de Ensino, Pesquisa e Inovação
FUNDAÇÃO MATIAS MACHLINE

```c
void mostrar(tipo_no *raiz){
   if(!raiz) return;
   mostrar(raiz->esq);
   printf("\n**********************");
   printf("\n Cod: %d  ", raiz->item.cod);
   printf("\n Nome: %s  ", raiz->item.nome);
   printf("\n Quantidade: %d  \n", raiz->item.cod);
   mostrar(raiz->dir);
}
```

```
**********************
 Cod: 6
 Nome: Cafe
 Quantidade: 600
**********************
 Cod: 8
 Nome: Carne
 Quantidade: 800
**********************
 Cod: 9
 Nome: Leite
 Quantidade: 900
```



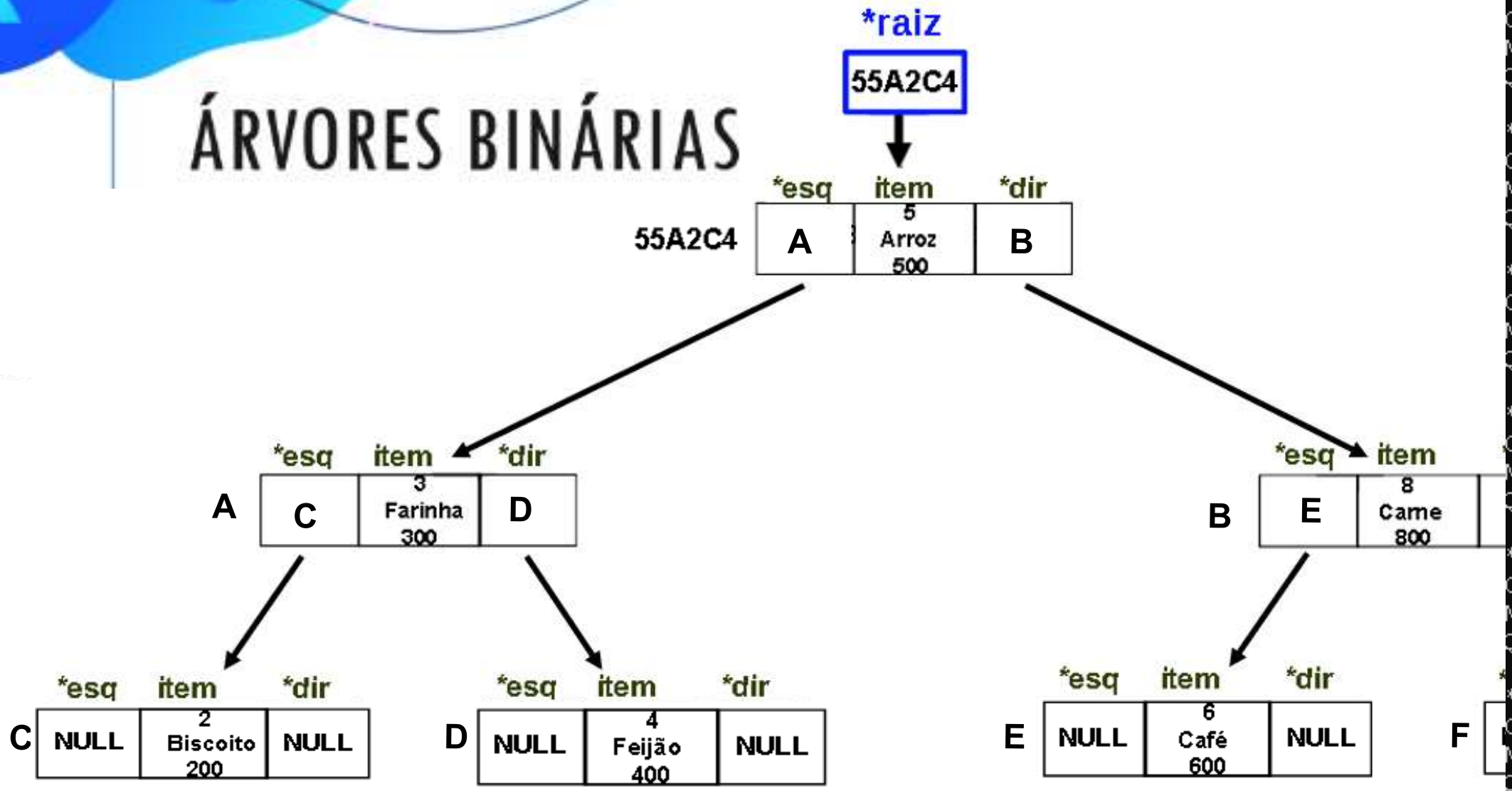
FAMEPI
Faculdade do Amazonas de Ensino, Pesquisa e Inovação
FUNDAÇÃO MATIAS MACHLINE




```
************************
Cod: 6
Nome: Cafe
Quantidade: 600
************************
Cod: 8
Nome: Carne
Quantidade: 800

************************
Cod: 9
Nome: Leite
Quantidade: 900
```

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

ÁRVORES BINÁRIAS




```
CONTEÚDO DA ÁRVORE:
**********************
Cod: 2
Nome: Biscoito
Quantidade: 200

**********************
Cod: 3
Nome: Farinha
Quantidade: 300

**********************
Cod: 4
Nome: Feijao
Quantidade: 400

**********************
Cod: 5
Nome: Arroz
Quantidade: 500

**********************
Cod: 6
Nome: Cafe
Quantidade: 600

**********************
Cod: 8
Nome: Carne
Quantidade: 800

**********************
Cod: 9
Nome: Leite
Quantidade: 900
```

ÁRVORES BINÁRIAS

IMPLEMENTAÇÃO

PESQUISAR NA ÁRVORE

ÁRVORES BINÁRIAS

Procedimento para pesquisar na árvore

- Para encontrar um registro com uma chave x:
  - Compare-a com a chave que está na raiz.
  - Se x é menor, vá para a subárvore esquerda.
  - Se x é maior, vá para a subárvore direita.
  - Repita o processo recursivamente, até que a chave procurada seja encontrada ou um nó folha seja atingido.

ÁRVORES BINÁRIAS

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

    int op, cod;

    tipo_no *raiz = NULL;

    tipo_item aux;

    do{
        system("cls");

        op = menu();

        switch(op){

            case 0: printf("\n\n\n\t Fim do Progra
                    return(0);

            case 1: aux = preencher_item();
                    if(!raiz) raiz = inserir(aux, raiz, raiz);
                    else inserir(aux, raiz, raiz);
                    break;

            case 2: printf("\n\n CONTEÚDO DA ÁRVORE: ");
                    mostrar(raiz);
                    getch(); // segura a tela
                    break;

            case 3: printf("\n\n PESQUISA NA ÁRVORE ");
                    printf("\n\n Digite o código do item a pesquisar: ");
                    scanf("%d", &cod);
                    pesquisar(cod, raiz);
                    getch(); // segura a tela
                    break;
```

op: 3

cod:

*rai: 55A2C4

```
********* MENU *********
[1] - INSERIR
[2] - MOSTRAR
[3] - PESQUISAR
[4] - EXCLUIR
[0] - SAIR

Digite sua opção: 3
```

ÁRVORES BINÁRIAS

```c
void pesquisar(int codPesq, tipo_no *raiz){

    if(!raiz) {
        printf("\n\n O item de código %d não está na árvore", codPesq);
        return;
    }

    if(codPesq < raiz->item.cod){
       pesquisar(codPesq, raiz->esq);
       return;
    }

    if(codPesq > raiz->item.cod)
       pesquisar(codPesq, raiz->dir);

    else{
        printf("\n\n *** ITEM ENCONTRADO ***");
        printf("\n\n Código: %d  ", raiz->item.cod);
        printf("\n\n Nome: %s  ", raiz->item.nome);
        printf("\n\n Quantidade: %d  ", raiz->item.quant);
    }
}
```

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

    int op, cod;

    tipo_no *raiz = NULL;

    tipo_item aux;

    do{
        system
        op = m

        switch(op){

            case 0: printf("\n\n\n\t Fim do Progra
                    return(0);

            case 1: aux = preencher_item();
                    if(!raiz) raiz = inserir(aux, raiz, raiz);
                    else inserir(aux, raiz, raiz);
                    break;

            case 2: printf("\n\n CONTEÚDO DA ÁRVORE: ");
                    mostrar(raiz);
                    getch(); // segura a tela
                    break;

            case 3: printf("\n\n PESQUISA NA ÁRVORE ");
                    printf("\n\n Digite o código do item a pesquisar: ");
                    scanf("%d", &cod);
                    pesquisar(cod, raiz);
                    getch(); // segura a tela
                    break;
```

op: 3

cod: 4

*rai: 55A2C4

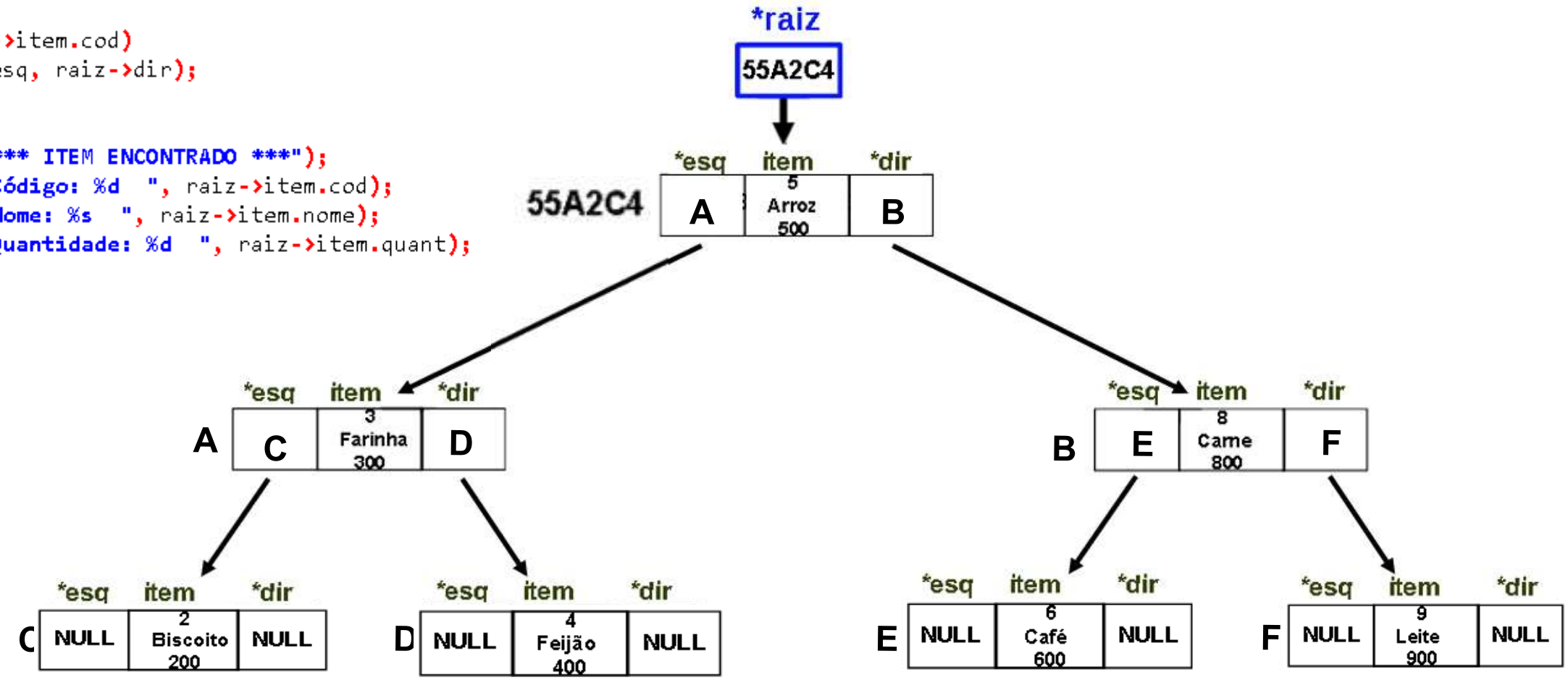
```
PESQUISA NA ÁRVORE

Digite o código do item a pesquisar: 4
```

```
********* MENU *********
[1] - INSERIR
[2] - MOSTRAR
[3] - PESQUISAR
[4] - EXCLUIR
[0] - SAIR

Digite sua opção: 3
```

```c
void pesquisar(int codPesq, tipo_no *raiz){

    if(!raiz) {
        printf("\n\n O item de código %d não está na árvore", codPesq);
        return;
    }

    if(codPesq < raiz->item.cod){
       pesquisar(codPesq, raiz->esq);
       return;
    }

    if(codPesq > raiz->item.cod)
       pesquisar(codPesq, raiz->dir);

    else{
        printf("\n\n *** ITEM ENCONTRADO ***");
        printf("\n\n Código: %d  ", raiz->item.cod);
        printf("\n\n Nome: %s  ", raiz->item.nome);
        printf("\n\n Quantidade: %d  ", raiz->item.quant);
    }
}
```

cod

4

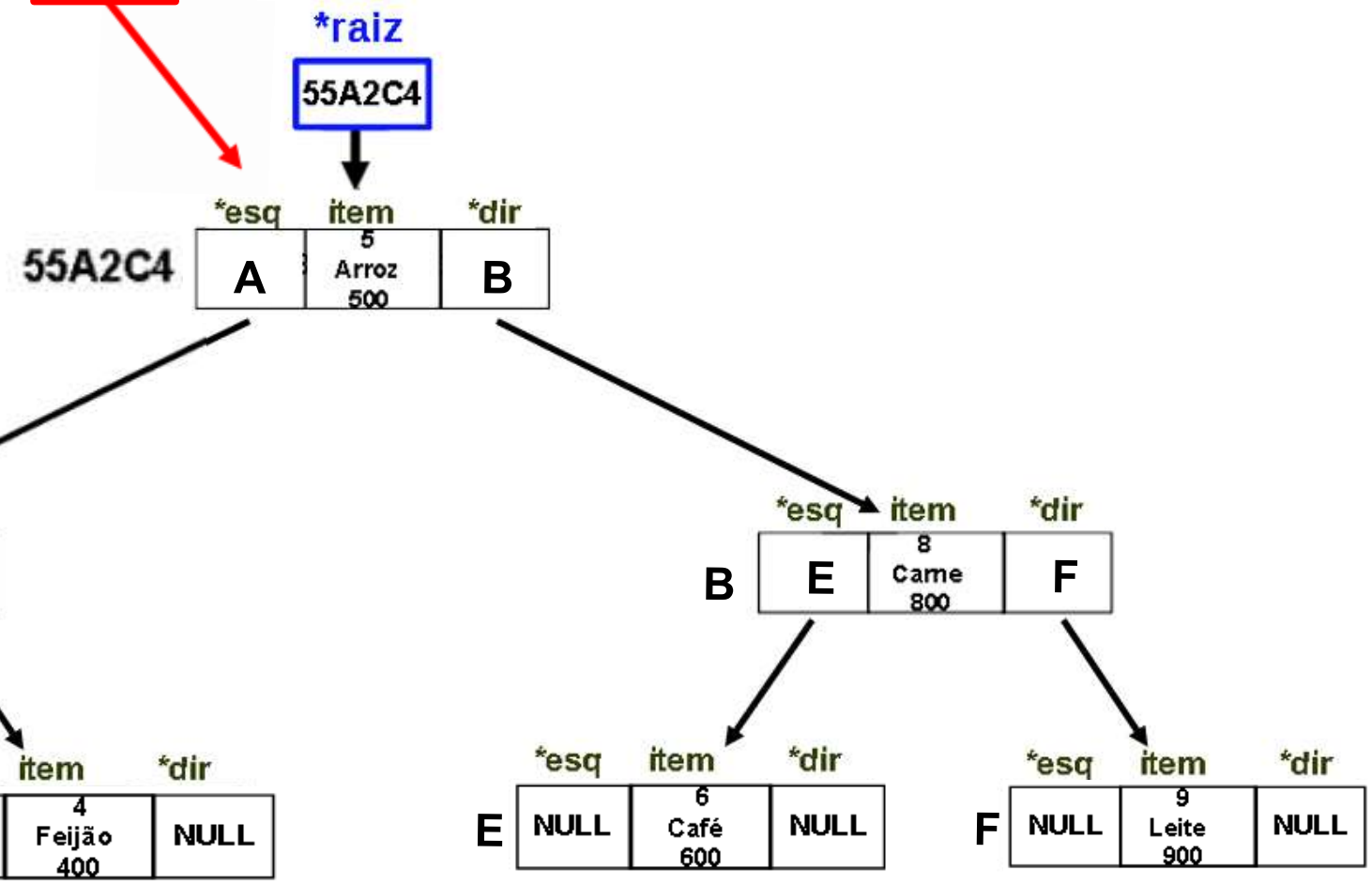


```c
void pesquisar(int codPesq, tipo_no *raiz){

    if(!raiz) {
        printf("\n\n O item de código %d não está na árvore", codPesq);
        return;
    }

    if(codPesq < raiz->item.cod){
       pesquisar(codPesq, raiz->esq);
       return;
    }

    if(codPesq > raiz->item.cod)
       pesquisar(codPesq, raiz->dir);

    else{
        printf("\n\n *** ITEM ENCONTRADO ***");
        printf("\n\n Código: %d  ", raiz->item.cod);
        printf("\n\n Nome: %s  ", raiz->item.nome);
        printf("\n\n Quantidade: %d  ", raiz->item.quant);
    }
}
```

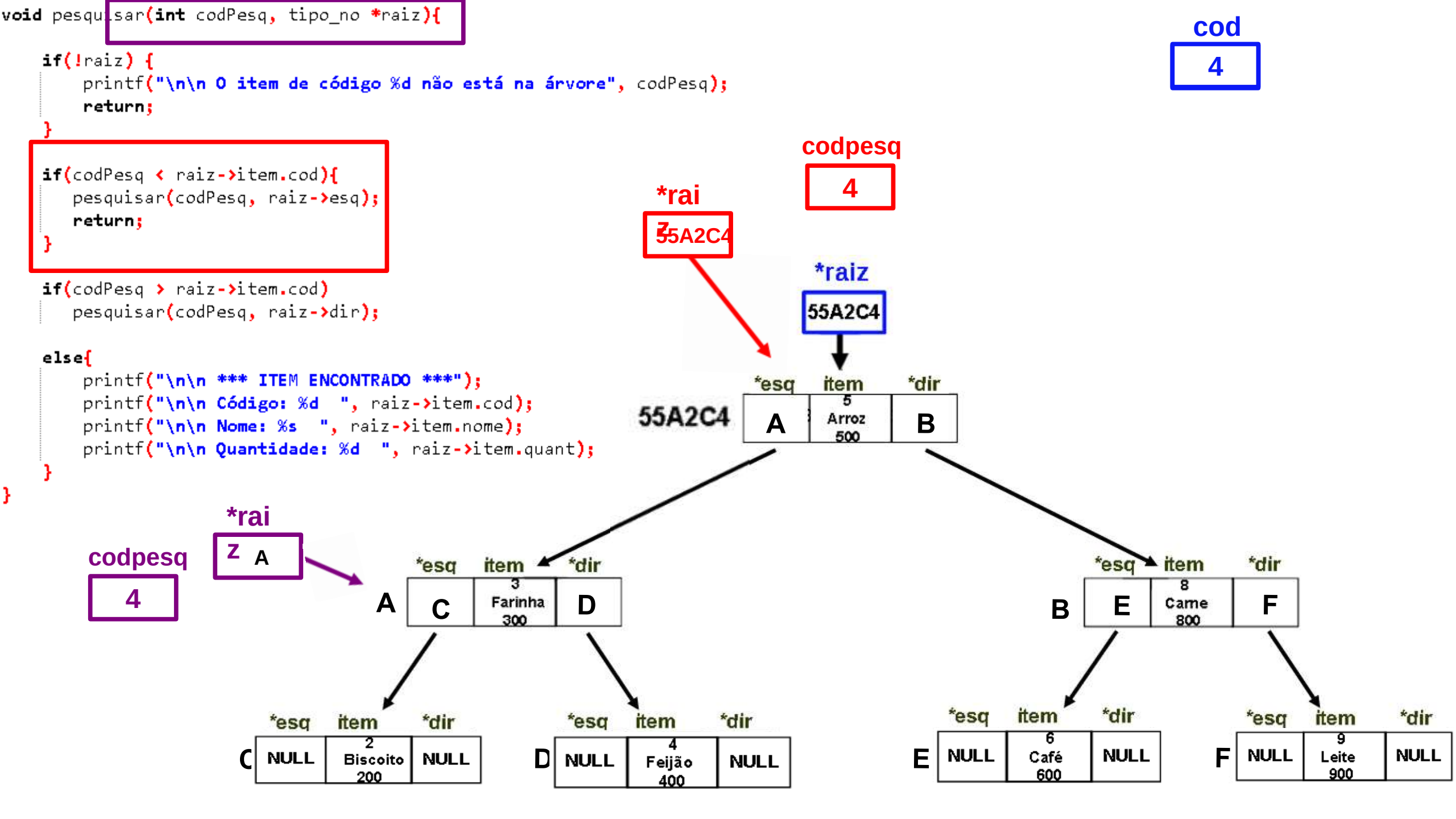
cod: 4

codpesq: 4

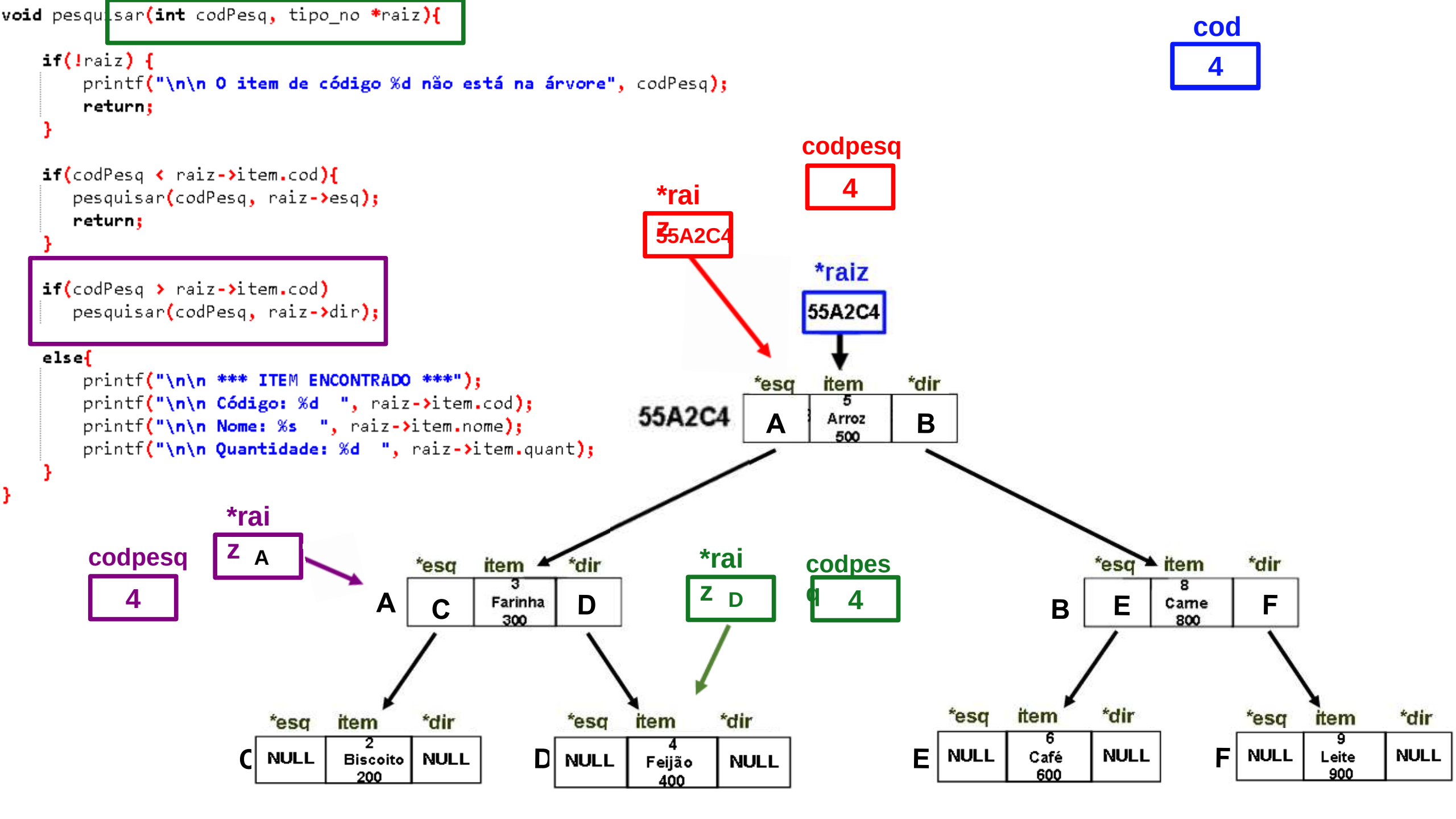
*rai: ~~7~~55A2C4



```c
void pesquisar(int codPesq, tipo_no *raiz){

    if(!raiz) {
        printf("\n\n O item de código %d não está na árvore", codPesq);
        return;
    }

    if(codPesq < raiz->item.cod){
       pesquisar(codPesq, raiz->esq);
       return;
    }

    if(codPesq > raiz->item.cod)
       pesquisar(codPesq, raiz->dir);

    else{
        printf("\n\n *** ITEM ENCONTRADO ***");
        printf("\n\n Código: %d  ", raiz->item.cod);
        printf("\n\n Nome: %s  ", raiz->item.nome);
        printf("\n\n Quantidade: %d  ", raiz->item.quant);
    }
}
```



```c
void pesquisar(int codPesq, tipo_no *raiz){

    if(!raiz) {
        printf("\n\n O item de código %d não está na árvore", codPesq);
        return;
    }

    if(codPesq < raiz->item.cod){
       pesquisar(codPesq, raiz->esq);
       return;
    }

    if(codPesq > raiz->item.cod)
       pesquisar(codPesq, raiz->dir);

    else{
        printf("\n\n *** ITEM ENCONTRADO ***");
        printf("\n\n Código: %d  ", raiz->item.cod);
        printf("\n\n Nome: %s  ", raiz->item.nome);
        printf("\n\n Quantidade: %d  ", raiz->item.quant);
    }
}
```



```c
void pesquisar(int codPesq, tipo_no *raiz){

    if(!raiz) {
        printf("\n\n O item de código %d não está na árvore", codPesq);
        return;
    }

    if(codPesq < raiz->item.cod){
       pesquisar(codPesq, raiz->esq);
       return;
    }

    if(codPesq > raiz->item.cod)
       pesquisar(codPesq, raiz->dir);

    else{
        printf("\n\n *** ITEM ENCONTRADO ***");
        printf("\n\n Código: %d  ", raiz->item.cod);
        printf("\n\n Nome: %s  ", raiz->item.nome);
        printf("\n\n Quantidade: %d  ", raiz->item.quant);
    }
}
```

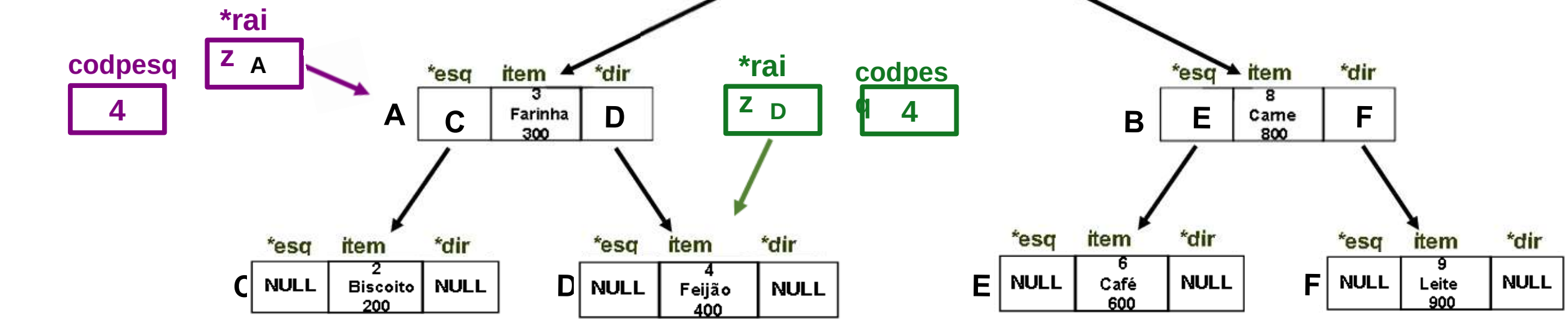
cod: 4

codpesq: 4

*rai: ~~55A2C4~~

*raiz: 55A2C4

```
*** ITEM ENCONTRADO ***

Código: 4

Nome: Feijao

Quantidade: 400
```



```c
void pesquisar(int codPesq, tipo_no *raiz){

    if(!raiz) {
        printf("\n\n O item de código %d não está na árvore", codPesq);
        return;
    }

    if(codPesq < raiz->item.cod){
       pesquisar(codPesq, raiz->esq);
       return;
    }

    if(codPesq > raiz->item.cod)
       pesquisar(codPesq, raiz->dir);

    else{
        printf("\n\n *** ITEM ENCONTRADO ***");
        printf("\n\n Código: %d  ", raiz->item.cod);
        printf("\n\n Nome: %s  ", raiz->item.nome);
        printf("\n\n Quantidade: %d  ", raiz->item.quant);
    }
}
```

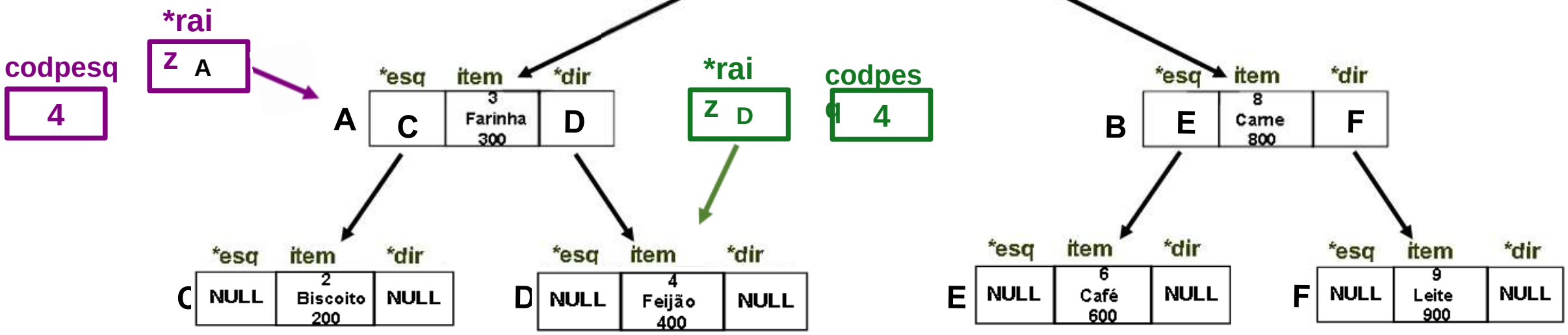
cod: 4

codpesq: 4

*rai: ~~55A2C4~~

*raiz: 55A2C4

```
*** ITEM ENCONTRADO ***

Código: 4

Nome: Feijao

Quantidade: 400
```



```c
void pesquisar(int codPesq, tipo_no *raiz){

    if(!raiz) {
        printf("\n\n O item de código %d não está na árvore", codPesq);
        return;
    }

    if(codPesq < raiz->item.cod){
       pesquisar(codPesq, raiz->esq);
       return;
    }

    if(codPesq > raiz->item.cod)
       pesquisar(codPesq, raiz->dir);

    else{
        printf("\n\n *** ITEM ENCONTRADO ***");
        printf("\n\n Código: %d  ", raiz->item.cod);
        printf("\n\n Nome: %s  ", raiz->item.nome);
        printf("\n\n Quantidade: %d  ", raiz->item.quant);
    }
}
```

cod

4

codpesq

4

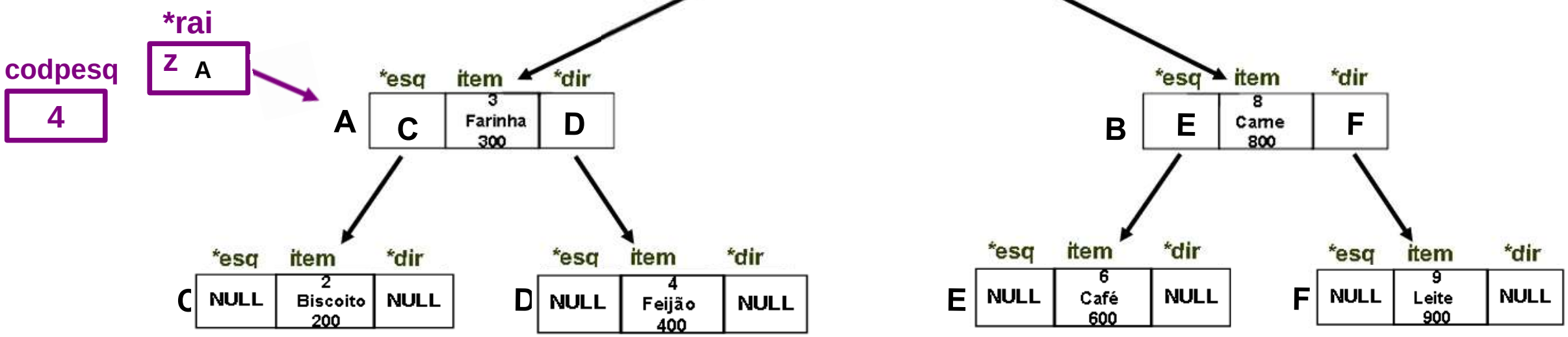
***rai**

~~55A2C4~~

***raiz**

55A2C4

```
*** ITEM ENCONTRADO ***

Código: 4

Nome: Feijao

Quantidade: 400
```



```c
void pesquisar(int codPesq, tipo_no *raiz){

    if(!raiz) {
        printf("\n\n O item de código %d não está na árvore", codPesq);
        return;
    }

    if(codPesq < raiz->item.cod){
       pesquisar(codPesq, raiz->esq);
       return;
    }

    if(codPesq > raiz->item.cod)
       pesquisar(codPesq, raiz->dir);

    else{
        printf("\n\n *** ITEM ENCONTRADO ***");
        printf("\n\n Código: %d  ", raiz->item.cod);
        printf("\n\n Nome: %s  ", raiz->item.nome);
        printf("\n\n Quantidade: %d  ", raiz->item.quant);
    }
}
```

cod

4

codpesq

4

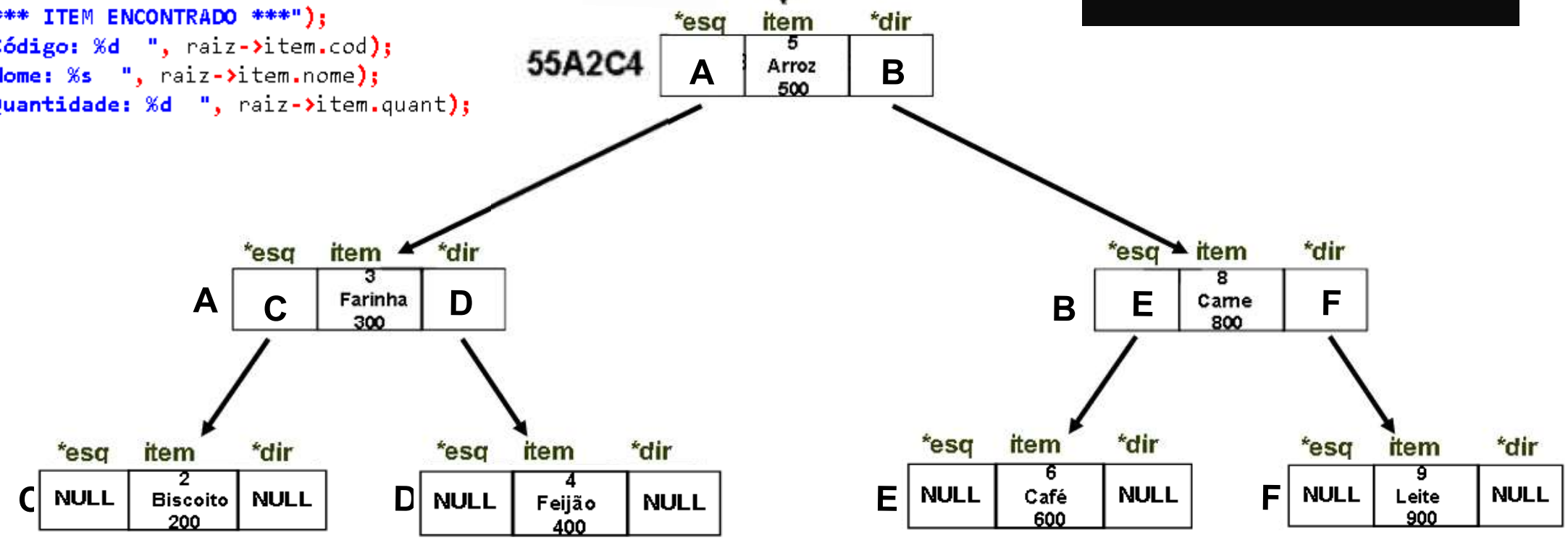
***rai**

55A2C4

***raiz**

55A2C4

```
*** ITEM ENCONTRADO ***

Código: 4

Nome: Feijao

Quantidade: 400
```



cod

4

```
*** ITEM ENCONTRADO ***

Código: 4

Nome: Feijao

Quantidade: 400
```