


Unidade VII: Árvore Binária - Inserção em C com ponteiro



PUC Minas


Instituto de Ciências Exatas e Informática
Departamento de Ciência da Computação

- Estrutura de arquivos
- makefile
- Arquivos “no”
- Arquivos “arvorebinaria”

- **Estrutura de arquivos** 
- makefile
- Arquivos “no”
- Arquivos “arvorebinaria”

Estrutura de Arquivos

- no.h
- no.c
- arvorebinaria.c
- arvorebinaria.h
- principal.c
- makefile

- Estrutura de arquivos
- **makefile** 
- Arquivos “no”
- Arquivos “arvorebinaria”

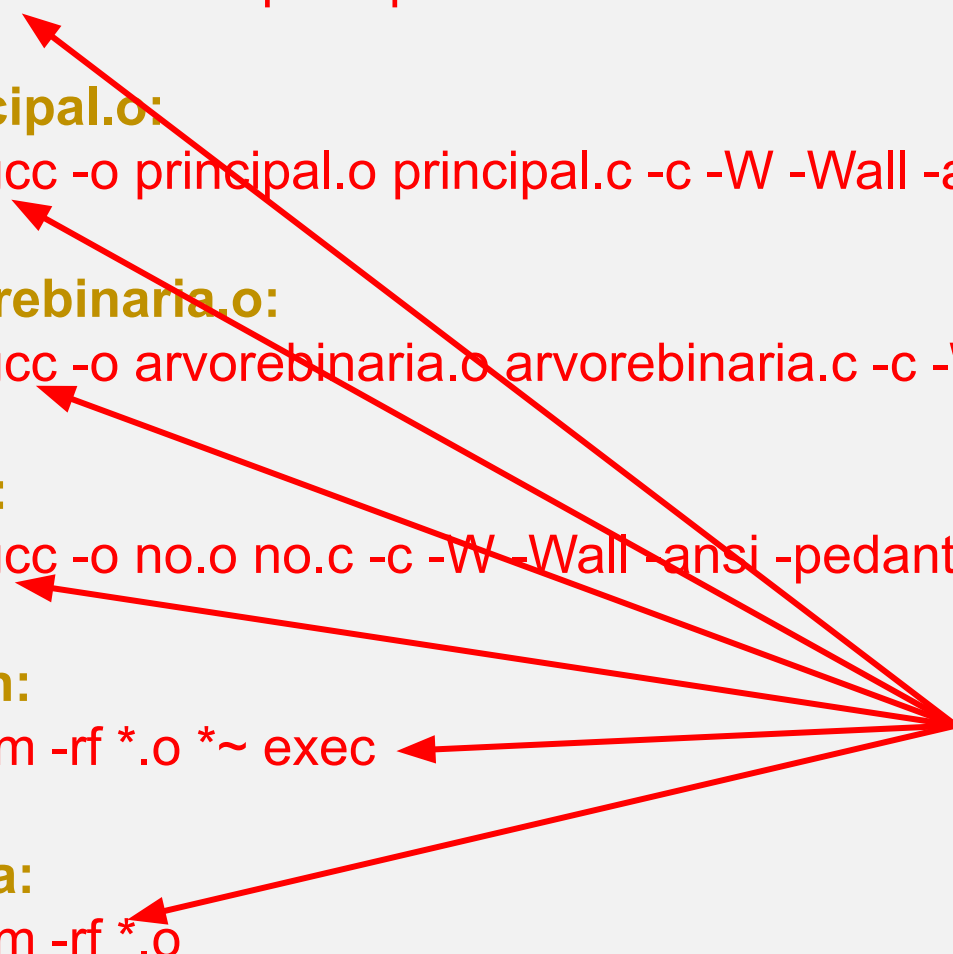
- Arquivo contendo um conjunto de diretivas usadas pela ferramenta de automação de compilação *make* para gerar um alvo / meta
- Nesse caso, os arquivos serão compilados digitando ***make***

```
1  all: exec
2
3  exec: principal.o arvorebinaria.o no.o
4      gcc -o exec principal.o arvorebinaria.o no.o
5
6  principal.o: principal.c
7      gcc -o principal.o principal.c -c -W -Wall -ansi -pedantic
8
9  arvorebinaria.o: arvorebinaria.c
10     gcc -o arvorebinaria.o arvorebinaria.c -c -W -Wall -ansi -pedantic
11
12 no.o: no.c
13     gcc -o no.o no.c -c -W -Wall -ansi -pedantic
14
15 clean:
16     rm -rf *.o *~ exec
17
18 limpa:
19     rm -rf *.o
```




```
1  all:
2
3  exec:
4      gcc -o exec principal.o arvorebinaria.o no.o
5
6  principal.o:
7      gcc -o principal.o principal.c -c -W -Wall -ansi -pedantic
8
9  arvorebinaria.o:
10     gcc -o arvorebinaria.o arvorebinaria.c -c -W -Wall -ansi -pedantic
11
12 no.o:
13     gcc -o no.o no.c -c -W -Wall -ansi -pedantic
14
15 clean:
16     rm -rf *.o *~ exec
17
18 limpa:
19     rm -rf *.o
```

comandos



```
1  all: exec
2
3  exec: principal.o arvorebinaria.o no.o
4
5
6  principal.o: principal.c
7
8
9  arvorebinaria.o: arvorebinaria.c
10
11
12  no.o: no.c
13
14
15  clean:
16
17
18  limpa:
19
```

pré-requisitos

```
graph LR
    PR[pré-requisitos] --> exec
    PR --> principal_o[principal.o]
    PR --> arvorebinaria_o[arvorebinaria.o]
    PR --> no_o[no.o]
```

```
1  all: exec
2
3  exec: principal.o arvorebinaria.o no.o
4      gcc -o exec principal.o arvorebinaria.o no.o
5
6  principal.o: principal.c
7      gcc -o principal.o principal.c -c -W -Wall -ansi -pedantic
8
9  arvorebinaria.o: arvorebinaria.c
10     gcc -o arvorebinaria.o arvorebinaria.c -c -W -Wall -ansi -pedantic
11
12 no.o: no.c
13     gcc -o no.o no.c -c -W -Wall -ansi -pedantic
14
15 clean:
16     rm -rf *.o *~ exec
17
18 limpa:
19     rm -rf *.o
```

Exercício Resolvido (6)

- Na pasta binariaC, digite a sequência de comandos abaixo e explique a saída na tela
 - 1) make all ; ls
 - 2) make clean ; ls
 - 3) make ; ls
 - 4) make clean ; ls
 - 5) make exec ; ls
 - 6) make limpa ; ls
 - 7) make no.o ; ls

Exercício Resolvido (6)

- Na pasta binariaC, digite a sequência de comandos abaixo e explique a saída na tela

- 1) **make all ; ls**
- 2) **make clean ; ls**
- 3) **make ; ls**
- 4) **make clean ; ls**
- 5) **make exec ; ls**
- 6) **make limpa ; ls**
- 7) **make no.o ; ls**

```
:$ make all ; ls
```

```
gcc -o principal.o principal.c -c -W -Wall -ansi -pedantic  
gcc -o arvorebinaria.o arvorebinaria.c -c -W -Wall -ansi -pedantic  
gcc -o no.o no.c -c -W -Wall -ansi -pedantic  
gcc -o exec principal.o arvorebinaria.o no.o
```

```
arvorebinaria.c arvorebinaria.h arvorebinaria.o exec makefile  
no.c no.h no.o principal.c principal.o
```

```
:$ make clean ; ls
```

```
rm -rf *.o *~ exec
```

```
arvorebinaria.c arvorebinaria.h makefile no.c no.h principal.c
```

Exercício Resolvido (6)

- Na pasta binariaC, digite a sequência de comandos abaixo e explique a saída na tela

- 1) `make all ; ls`
- 2) `make clean ; ls`
- 3) `make ; ls`
- 4) `make clean ; ls`
- 5) `make exec ; ls`
- 6) `make limpa ; ls`
- 7) `make no.o ; ls`

```
:$ make ; ls
```

```
gcc -o principal.o principal.c -c -W -Wall -ansi -pedantic  
gcc -o arvorebinaria.o arvorebinaria.c -c -W -Wall -ansi -pedantic  
gcc -o no.o no.c -c -W -Wall -ansi -pedantic  
gcc -o exec principal.o arvorebinaria.o no.o
```

```
arvorebinaria.c arvorebinaria.h arvorebinaria.o exec makefile  
no.c no.h no.o principal.c principal.o
```

```
:$ make clean ; ls
```

```
rm -rf *.o *~ exec
```

```
arvorebinaria.c arvorebinaria.h makefile no.c no.h principal.c
```

Exercício Resolvido (6)

- Na pasta binariaC, digite a sequência de comandos abaixo e explique a saída na tela

- 1) make all ; ls
- 2) make clean ; ls
- 3) make ; ls
- 4) make clean ; ls
- 5) **make exec ; ls**
- 6) **make limpa ; ls**
- 7) make no.o ; ls

```
:$ make exec ; ls
```

```
gcc -o principal.o principal.c -c -W -Wall -ansi -pedantic  
gcc -o arvorebinaria.o arvorebinaria.c -c -W -Wall -ansi -pedantic  
gcc -o no.o no.c -c -W -Wall -ansi -pedantic  
gcc -o exec principal.o arvorebinaria.o no.o
```

```
arvorebinaria.c arvorebinaria.h arvorebinaria.o exec makefile  
no.c no.h no.o principal.c principal.o
```

```
:$ make limpa ; ls
```

```
rm -rf *.o *
```

```
arvorebinaria.c arvorebinaria.h exec makefile no.c no.h  
principal.c
```

Exercício Resolvido (6)


- Na pasta binariaC, digite a sequência de comandos abaixo e explique a saída na tela

- 1) make all ; ls
- 2) make clean ; ls
- 3) make ; ls
- 4) make clean ; ls
- 5) make exec ; ls
- 6) make limpa ; ls
- 7) **make no.o ; ls**

```
:$ make no.o ; ls
```

```
gcc -o no.o no.c -c -W -Wall -ansi -pedantic
```

```
arvorebinaria.c arvorebinaria.h arvorebinaria.o exec makefile  
no.c no.h no.o principal.c principal.o
```


- Estrutura de arquivos
- makefile
- **Arquivos “no”** 
- Arquivos “arvorebinaria”


Arquivos “no”

//no.h

```
typedef struct No {  
    int elemento;  
    struct No *esq, *dir;  
} No;  
  
No* novoNo(int elemento);
```

//no.c

```
#include <stdlib.h>  
#include "no.h"  
  
No* novoNo(int elemento) {  
    No* novo = (No*) malloc(sizeof(No));  
    novo->elemento = elemento;  
    novo->esq = NULL;  
    novo->dir = NULL;  
    return novo;  
}
```

- Estrutura de arquivos
- makefile
- Arquivos “no”
- **Arquivos “arvorebinaria”** 

Arquivos “arvorebinaria”

```
//arvorebinaria.h
#include "no.h"
#define bool short
#define true 1
#define false 0

bool pesquisarRec(int, No*);
void caminharCentralRec(No*);
void caminharPreRec(No*);
void caminharPosRec(No*);
void inserirRec(int, No**);
void removerRec(int, No**);
void antecessor(No**, No**);
```

```
void start();
```

```
bool pesquisar(int);
void caminharCentral();
void caminharPre();
void caminharPos();
void inserir(int);
void remover(int);
```

```
//arvorebinaria.c
#include "no.h"
#include <err.h>
#include <stdlib.h>
#include <stdio.h>
#include "arvorebinaria.h"
```

```
No* raiz;
```

```
void start() {
    raiz = NULL;
}
```



Arquivos “arvorebinaria”

```
//arvorebinaria.h
#include "no.h"
#define bool short
#define true 1
#define false 0

bool pesquisarRec(int, No*);
void caminharCentralRec(No*);
void caminharPreRec(No*);
void caminharPosRec(No*);
void inserirRec(int, No**);
void removerRec(int, No**);
void antecessor(No**, No**);

void start();
bool pesquisar(int);
void caminharCentral();
void caminharPre();
void caminharPos();
void inserir(int);
void remover(int);
```

Como o C tem apenas a passagem de parâmetros por valor, neste material, optamos por fazer a inserção usando o endereço de ponteiro

Poderíamos, também, usar as duas estratégias implementadas em nosso código Java

Implementação da Função Inserir

- Anteriormente, em Java, apresentamos duas implementações do inserir()

```
No inserir(int x, No i) //Java
```

```
void inserir(int x, No i, No pai) //Java
```

- As implementações correspondentes em C seriam:

```
No* inserir(int x, No* i) //C
```

```
void inserir(int x, No* i, No* pai) //C
```

Implementação da Função Inserir

- Anteriormente, em Java, apresentamos duas implementações do inserir()

No inserir(int x, No i) //Java

void inserir(int x, No i, No pai) //Java

- As implementações correspondentes em C seriam:

No* inserir(int x, No* i) //C

void inserir(int x, No* i, No* pai) //C

Primeira Opção para o Inserir em C/Java

//código em Java

```
void inserir(int x) {  
    raiz = inserir(x, raiz);  
}
```

```
No inserir(int x, No i) {  
    if (i == null) {  
        i = new No(x);  
    } else if (x < i.elemento) {  
        i.esq = inserir(x, i.esq);  
    } else if (x > i.elemento) {  
        i.dir = inserir(x, i.dir);  
    } else {  
        throw new ("Erro!");  
    }  
    return i;  
}
```

//código em C

```
void inserir(int x) {  
    raiz = inserirRec(x, raiz);  
}  
  
No* inserirRec(int x, No* i) {  
    if (i == NULL) {  
        i = novoNo(x);  
    } else if (x < i->elemento) {  
        i->esq = inserirRec(x, i->esq);  
    } else if (x > i->elemento) {  
        i->dir = inserirRec(x, i->dir);  
    } else {  
        errx(1, "Erro ao inserir!");  
    }  
    return i;  
}
```


Implementação da Função Inserir

- Anteriormente, em Java, apresentamos duas implementações do inserir()

No inserir(int x, No i) //Java

void inserir(int x, No i, No pai) //Java

- As implementações correspondentes em C seriam:

No* inserir(int x, No* i) //C

void inserir(int x, No* i, No* pai) //C

Segunda Opção para o Inserir em C/Java

//código em Java

```
void inserirPai(int x) {
    if (raiz == null) {
        raiz = new No(x);
    } else if (x < raiz.elemento) {
        inserirPai(x, raiz.esq, raiz);
    } else if (x > raiz.elemento) {
        inserirPai(x, raiz.dir, raiz);
    } else { throw new("Erro!");
    } }

void inserirPai(int x, No i, No pai) {
    if (i == null) {
        if (x < pai.elemento){ pai.esq = new No(x);
        } else {                pai.dir = new No(x); }
    } else if (x < i.elemento) {
        inserirPai(x, i.esq, i);
    } else if (x > i.elemento) {
        inserirPai(x, i.dir, i);
    } else { throw new("Erro!");
    } }
```

//código em C

```
void inserirPai(int x) {
    if(raiz == NULL){
        raiz = novoNo(x);
    } else if(x < raiz->elemento){
        inserirPaiRec(x, raiz->esq, raiz);
    } else if(x > raiz->elemento){
        inserirPaiRec(x, raiz->dir, raiz);
    } else {  errx(1, "Erro ao inserir!");
    } }

void inserirPaiRec(int x, No* i, No* pai) {
    if (i == NULL) {
        if(x < i->elemento){ pai->esq = novoNo(x);
        } else {                pai->dir = novoNo(x); }
    } else if (x < i->elemento) {
        inserirPaiRec(x, i->esq, i);
    } else if (x > i->elemento) {
        inserirPaiRec(x, i->dir, i);
    } else {  errx(1, "Erro ao inserir!");
    } }
```

Arquivos “arvorebinaria”

```
//arvorebinaria.h
#include "no.h"
#define bool short
#define true 1
#define false 0

bool pesquisarRec(int, No*);
void caminharCentralRec(No*);
void caminharPreRec(No*);
void caminharPosRec(No*);
void inserirRec(int, No**);
void removerRec(int, No**);
void antecessor(No**, No**);

void start();
bool pesquisar(int);
void caminharCentral();
void caminharPre();
void caminharPos();
void inserir(int);
void remover(int);
```

```
//arvorebinaria.c
■■■

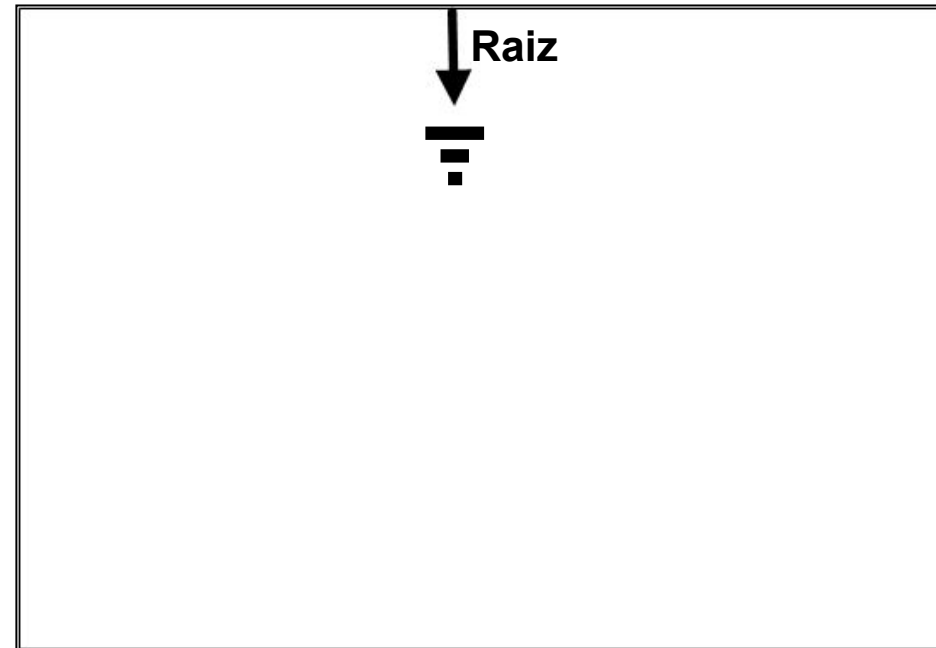
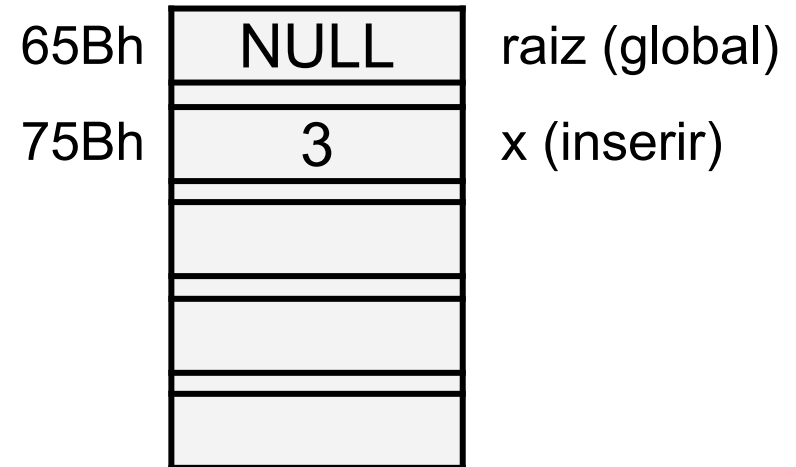
void inserir(int x) {
    inserirRec(x, &raiz);
}

void inserirRec(int x, No** i) {
    if (*i == NULL) {
        *i = novoNo(x);
    } else if (x < (*i)->elemento) {
        inserirRec(x, &((*i)->esq));
    } else if (x > (*i)->elemento) {
        inserirRec(x, &((*i)->dir));
    } else {
        errx(1, "Erro ao inserir!");
    }
}
```

Arquivos “arvorebinaria”

```
//arvorebinaria.c (supondo inserir o 3)
```

```
void inserir(int x) {  
    inserirRec(x, &raiz);  
}  
  
void inserirRec(int x, No** i) {  
    if (*i == NULL) {  
        *i = novoNo(x);  
    } else if (x < (*i)->elemento) {  
        inserirRec(x, &((*i)->esq));  
    } else if (x > (*i)->elemento) {  
        inserirRec(x, &((*i)->dir));  
    } else {  
        errx(1, "Erro ao inserir!");  
    }  
}
```

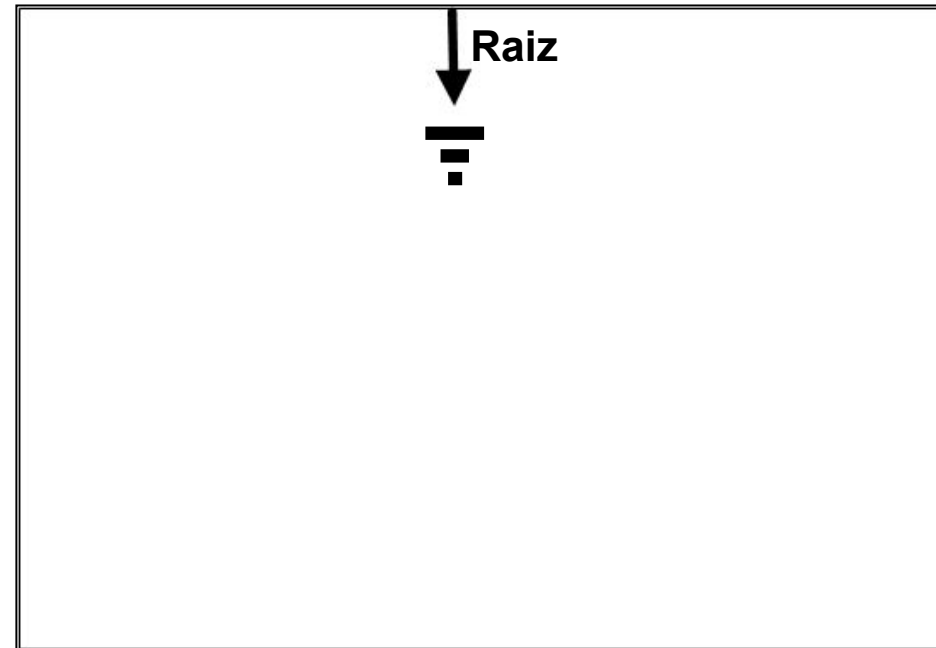
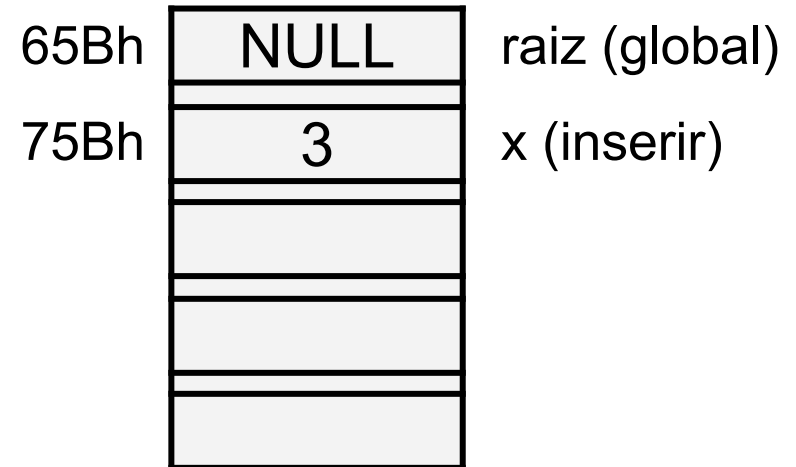


Arquivos “arvorebinaria”

```
//arvorebinaria.c (supondo inserir o 3)
```

```
void inserir(int x) {  
    inserirRec(x, &raiz);  
}
```

```
void inserirRec(int x, No** i) {  
    if (*i == NULL) {  
        *i = novoNo(x);  
    } else if (x < (*i)->elemento) {  
        inserirRec(x, &((*i)->esq));  
    } else if (x > (*i)->elemento) {  
        inserirRec(x, &((*i)->dir));  
    } else {  
        errx(1, "Erro ao inserir!");  
    }  
}
```



Arquivos “arvorebinaria”

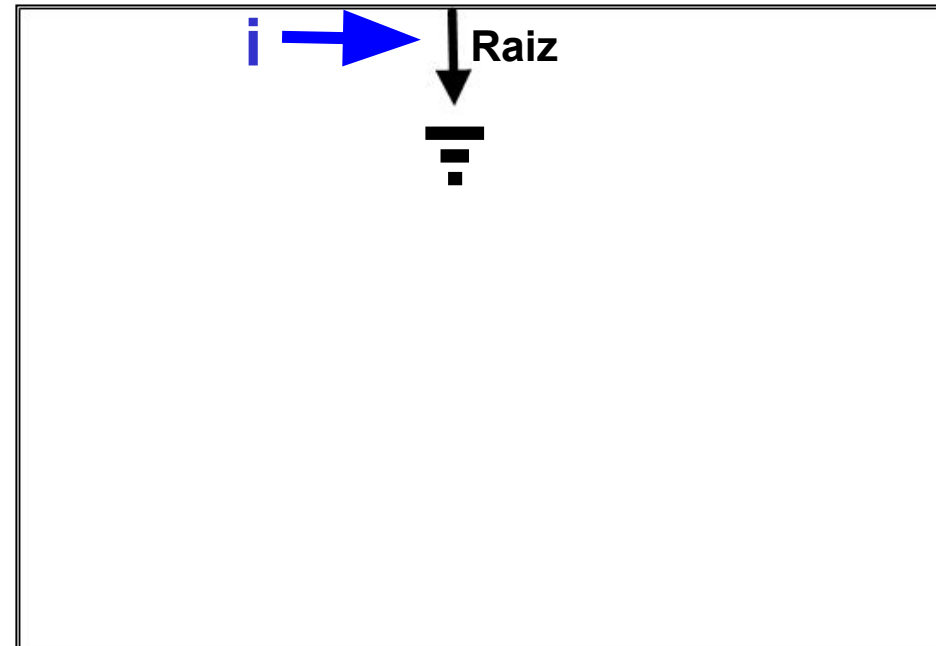
//arvorebinaria.c (supondo inserir o 3)

```
void inserir(int x) {
    inserirRec(x, &raiz);
}
```

```
void inserirRec(int x, No** i) {
```

```
    if (*i == NULL) {
        *i = novoNo(x);
    } else if (x < (*i)->elemento) {
        inserirRec(x, &((*i)->esq));
    } else if (x > (*i)->elemento) {
        inserirRec(x, &((*i)->dir));
    } else {
        errx(1, "Erro ao inserir!");
    }
}
```

65Bh	NULL	raiz (global)
75Bh	3	x (inserir)
800h	3	x (inserirRec)
811h	65Bh	i (inserirRec)



Arquivos “arvorebinaria”

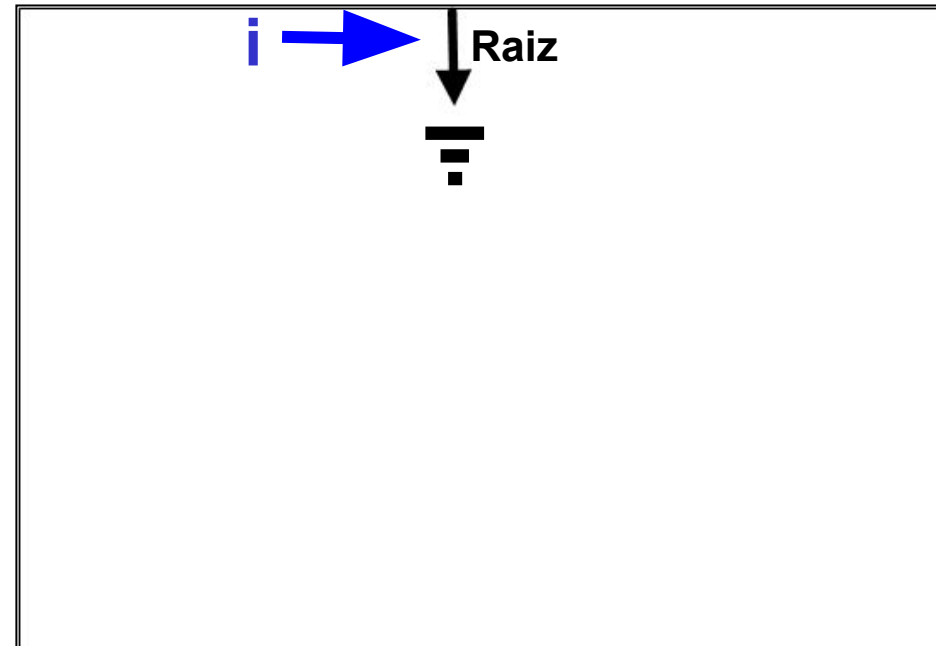
//arvorebinaria.c (supondo inserir o 3)

```
void inserir(int x) {
    inserirRec(x, &raiz);
}
```

```
void inserirRec(int x, No** i) {
    if (*i == NULL) {
```

```
        *i = novoNo(x);
    } else if (x < (*i)->elemento) {
        inserirRec(x, &((*i)->esq));
    } else if (x > (*i)->elemento) {
        inserirRec(x, &((*i)->dir));
    } else {
        errx(1, "Erro ao inserir!");
    }
}
```

65Bh	NULL	raiz (global)
75Bh	3	x (inserir)
800h	3	x (inserirRec)
811h	65Bh	i (inserirRec)



Arquivos “arvorebinaria”

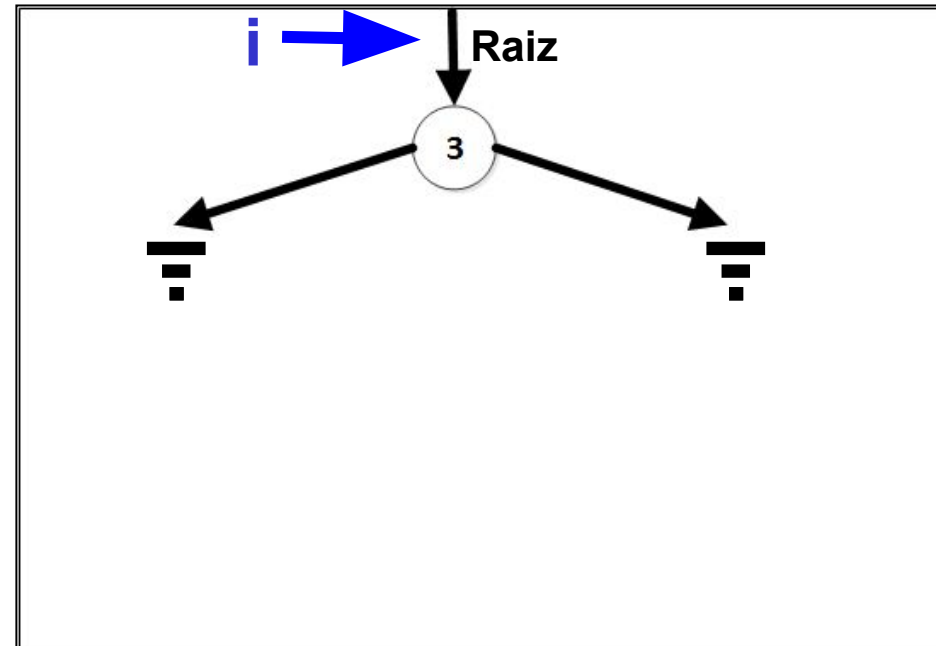
//arvorebinaria.c (supondo inserir o 3)

```
void inserir(int x) {
    inserirRec(x, &raiz);
}
```

```
void inserirRec(int x, No** i) {
    if (*i == NULL) {
        *i = novoNo(x);
```

```
    } else if (x < (*i)->elemento) {
        inserirRec(x, &((*i)->esq));
    } else if (x > (*i)->elemento) {
        inserirRec(x, &((*i)->dir));
    } else {
        errx(1, "Erro ao inserir!");
    }
}
```

65Bh	922h	raiz (global)
75Bh	3	x (inserir)
800h	3	x (inserirRec)
811h	65Bh	i (inserirRec)
922h	3/null/null	(novoNo)



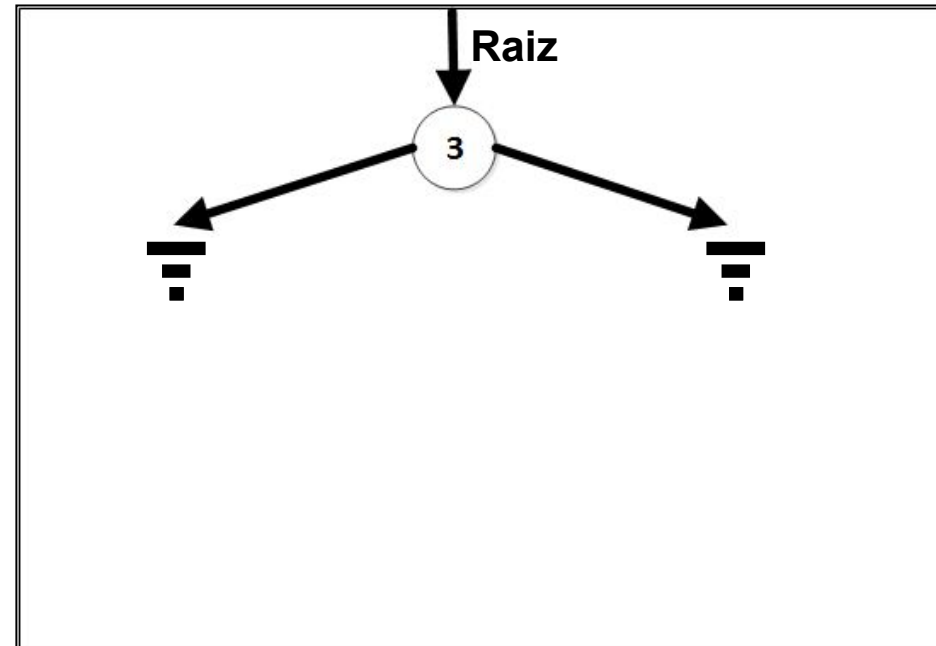
Arquivos “arvorebinaria”

//arvorebinaria.c (supondo inserir o 3)

```
void inserir(int x) {
    inserirRec(x, &raiz);
}

void inserirRec(int x, No** i) {
    if (*i == NULL) {
        *i = novoNo(x);
    } else if (x < (*i)->elemento) {
        inserirRec(x, &((*i)->esq));
    } else if (x > (*i)->elemento) {
        inserirRec(x, &((*i)->dir));
    } else {
        errx(1, "Erro ao inserir!");
    }
}
```

65Bh	922h	raiz (global)
75Bh	3	x (inserir)
800h	3	x (inserirRec)
811h	65Bh	i (inserirRec)
922h	3/null/null	(novoNo)



Arquivos “arvorebinaria”

//arvorebinaria.c (supondo inserir o 3)

```
void inserir(int x) {
    inserirRec(x, &raiz);
}
```

```
void inserirRec(int x, No** i) {
    if (*i == NULL) {
        *i = novoNo(x);
    } else if (x < (*i)->elemento) {
        inserirRec(x, &((*i)->esq));
    } else if (x > (*i)->elemento) {
        inserirRec(x, &((*i)->dir));
    } else {
        errx(1, "Erro ao inserir!");
    }
}
```

65Bh	922h	raiz (global)
75Bh	3	x (inserir)
800h	3	x (inserirRec)
811h	65Bh	i (inserirRec)
922h	3/null/null	(novoNo)

