

# INSERÇÃO DA ÁRVORE BINÁRIA DE BUSCA

# Objetivo

- Compreender o funcionamento de um método de inserção em uma árvore binária de busca simples.

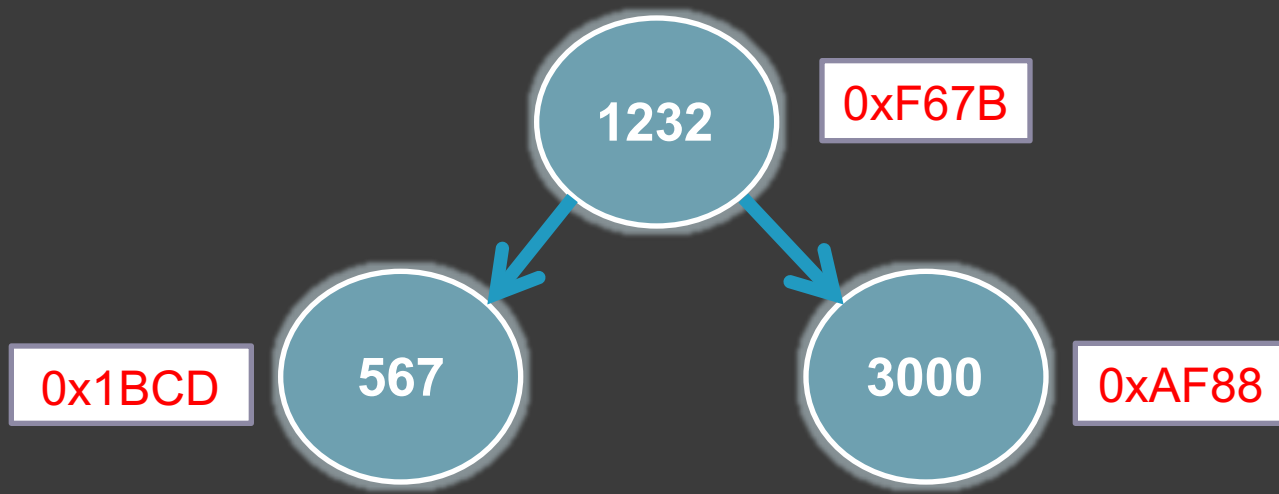
# Código-fonte Inserir

```
01. void insere(tipo_no *&no, int dado){
02.     if (no == NULL){
03.         no = new tipo_no;
04.         no->dado = dado;
05.         no->dir = NULL;
06.         no->esq = NULL;
07.     }else{
08.         if (dado > no->dado){
09.             insere (no->dir, dado);
10.         }else{
11.             if (dado < no->dado){
12.                 insere (no->esq, dado);
13.             }
14.         }
15.     }
16. }
```

EXECUÇÃO

# Teste de Mesa

- Imaginemos a seguinte situação



```
insere( arvore->raiz, 678 );
```

# Teste de Mesa


- Como será utilizada recursividade, para facilidade de entendimento, será demonstrado o estado da memória referente a cada chamada de função

**no = 0xF67B**

**dado = 678**

**PC = 2**

# Teste de Mesa

```
01. void insere(tipo_no *&no, int dado) {  
02.      if (no == NULL) {  
03.         no = new tipo_no;  
04.         no->dado = dado;  
05.         no->dir = NULL;  
06.         no->esq = NULL;  
07.     }else{  
08.         if (dado > no->dado) {  
09.             insere(&no->dir, dado);  
10.         }else{  
11.             if (dado < no->dado) {  
12.                 insere (&no->esq, dado);  
13.             }  
14.         }  
15.     }  
16. }
```

no possui o valor 0xF67B, que é o endereço de memória do nó raiz da árvore, logo não é igual a NULL.

# Teste de Mesa

```
01. void insere
02.     if (n
03.         no
04.         no
05.         no
06.         no-
07.     }else{
08.         → if (dado > no->dado) {
09.             insere (no->dir, dado);
10.         }else{
11.             if (dado < no->dado) {
12.                 insere (no->esq, dado);
13.             }
14.         }
15.     }
16. }
```

salta-se, para a linha 08

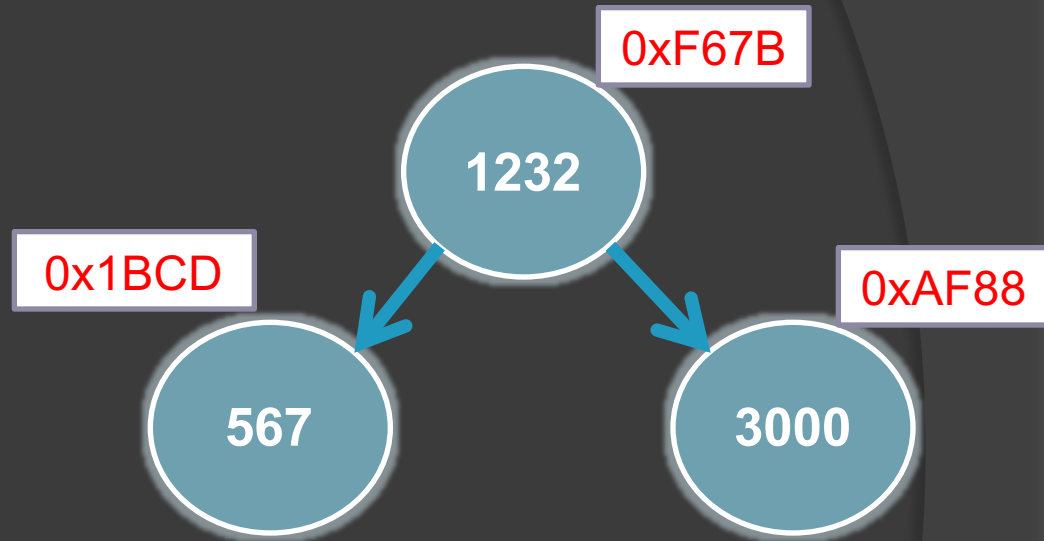


# Teste de Mesa

no = 0xF67B

dado = 678

PC = 8



# Teste de Mesa

```
01. void insere
02.     if (n
03.         no
04.         no
05.         no
06.         no-
07.     }else{
08.         → if (dado > no->dado) {
09.             insere (no->dir, dado);
10.         }else{
11.             if (dado < no->dado) {
12.                 insere (no->esq, dado);
13.             }
14.         }
15.     }
16. }
```

salta-se, para a linha 08, onde é verificado se o dado que existe no nó analisado (0xF67B), no caso 1232. Logo, a comparação poderia ser traduzida como  $678 > 1232$ ?

# Teste de Mesa

```
01. void insere(tipo_no *&no, int dado) {  
02.     if (no == NULL) {  
03.         no = new tipo_no;  
04.  
05.  
06.  
07.  
08.     }  
09.     insere (no->dir, dado);  
10.     else {  
11.         if (dado < no->dado) {  
12.             insere (no->esq, dado);  
13.         }  
14.     }  
15. }  
16. }
```

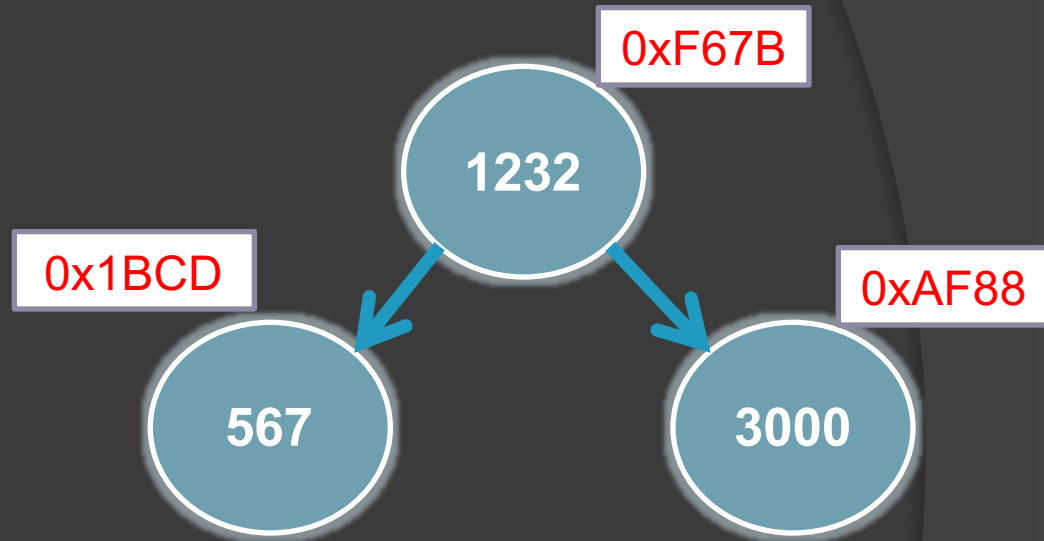
Como o teste é falso, salta-se para a linha 11.

# Teste de Mesa

no = 0xF67B

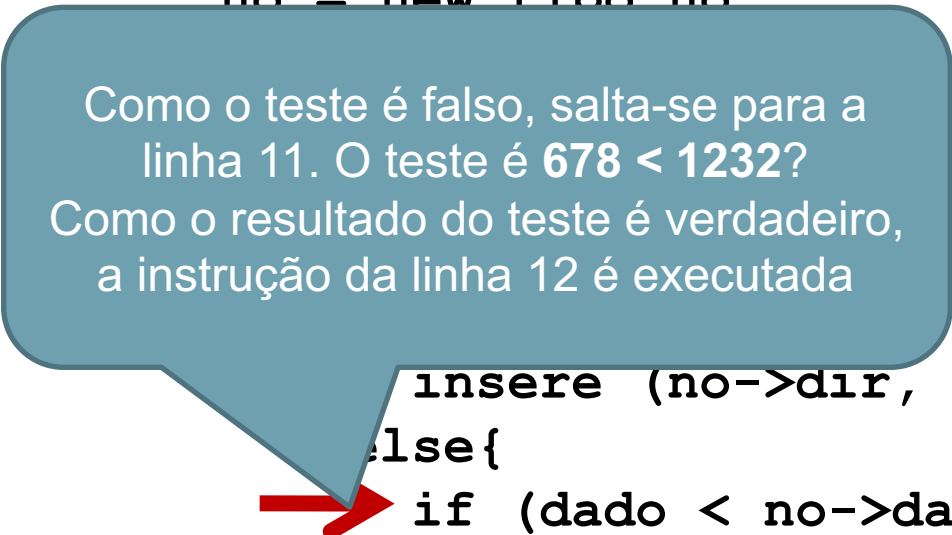
dado = 678

PC = 11



# Teste de Mesa

```
01. void insere(tipo_no *&no, int dado) {  
02.     if (no == NULL) {  
03.         no = new tipo_no;  
04.  
05.         Como o teste é falso, salta-se para a  
06.         linha 11. O teste é  $678 < 1232$ ?  
07.         Como o resultado do teste é verdadeiro,  
08.         a instrução da linha 12 é executada  
09.         {  
10.             insere (no->dir, dado);  
11.         }  
12.     }  
13.     }  
14. }  
15.  
16. }
```

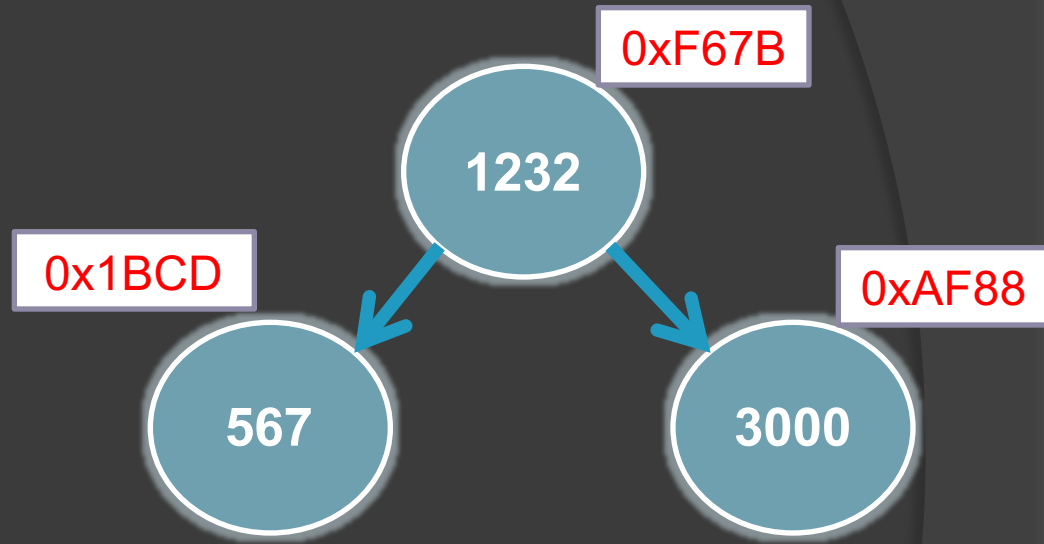


# Teste de Mesa

no = 0xF67B

dado = 678

PC = 12



# Teste de Mesa

```
01. void insere(tipo_no *&no, int dado) {  
02.     if (no == NULL) {  
03.         no = new tipo_no;  
04.         no->dado = dado;  
05.         no->esq = NULL;  
06.         no->dir = NULL;  
07.         if (dado > no->dado) {  
08.             insere (no->dir, dado);  
09.         }  
10.     }  
11.     if (dado < no->dado) {  
12.         insere (no->esq, dado);  
13.     }  
14. }  
15. }  
16. }
```

Uma chamada recursiva é realizada, passando como parâmetros o nó da esquerda e o novo índice 678.

# Teste de Mesa

**no = 0xF67B**

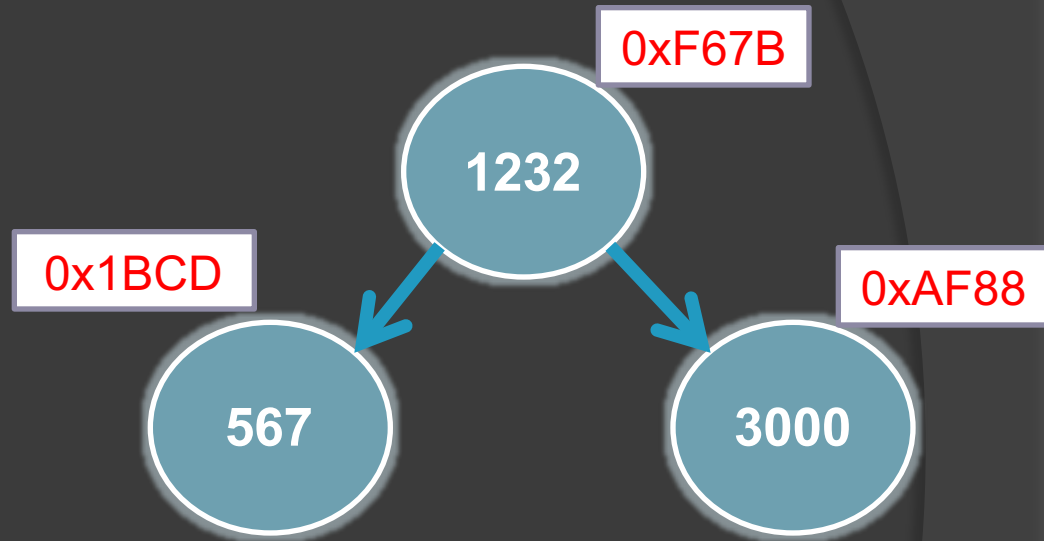
**dado = 678**

**PC = 12**

**no = 0x1BCD**

**dado = 678**

**PC = 2**






# Teste de Mesa

```
01. void insere(tipo_no *&no, int dado) {  
02.     → if (no == NULL) {  
03.         no = new tipo_no;  
04.         no->valor = dado;  
05.         no->esq = NULL;  
06.         no->dir = NULL;  
07.     }  
08.     if (no->esq == NULL) {  
09.         if (dado < no->valor) {  
10.             insere (no->esq, dado);  
11.         }  
12.     }  
13.     if (no->dir == NULL) {  
14.         if (dado > no->valor) {  
15.             insere (no->dir, dado);  
16.         }  
    }
```

no possui o valor 0x1BCD, que é o endereço de memória do nó a esquerda da raiz da árvore, logo não é igual a NULL.

# Teste de Mesa

```
01. void insere(tipo_no *&no, int dado) {  
02.     if (no == NULL) {  
03.         no = new tipo_no;  
04.         no->dado = dado;  
05.         no->dir = NULL;  
06.         no->esq = NULL;  
07.     }else{  
08.          if (dado > no->dado) {  
09.             insere (no->dir, dado);  
10.         }  
11.         if (dado < no->dado) {  
12.             insere (no->esq, dado);  
13.         }  
14.     }  
15. }
```

Salta-se para a linha 08

# Teste de Mesa

**no = 0xF67B**

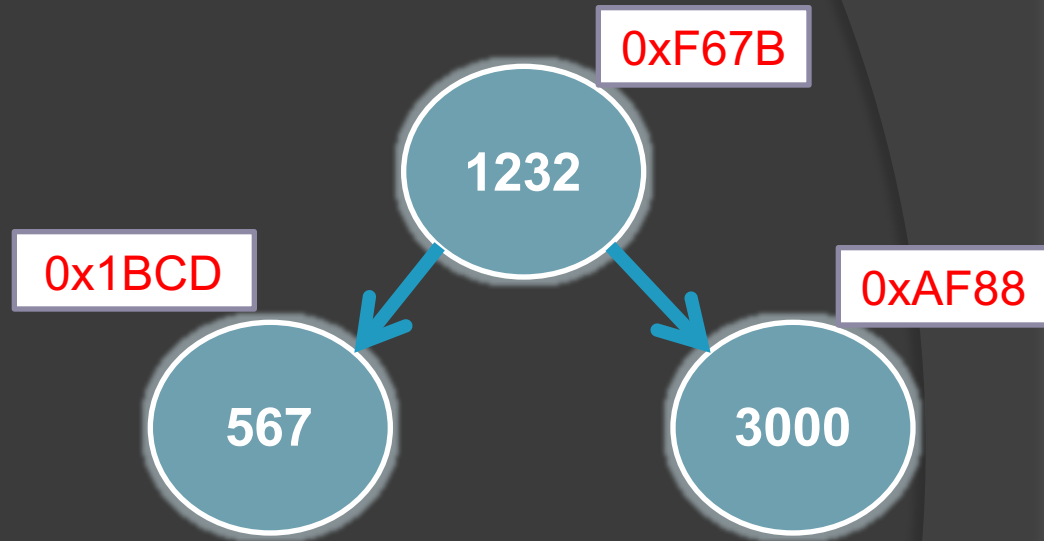
**dado = 678**

**PC = 12**


**no = 0x1BCD**

**dado = 678**

**PC = 8**




# Teste de Mesa

```
01. void insere(tipo_no *&no, int dado) {  
02.     if (no == NULL) {  
03.         no = new tipo_no;  
04.         no->dado = dado;  
05.         no->dir = NULL;  
06.         no->esq = NULL;  
07.     }else{  
08.          if (dado > no->dado) {  
09.             insere (no->dir, dado) ;  
10.         }  
11.         if (dado < no->dado) {  
12.             insere (no->esq, dado) ;  
13.         }  
14.     }  
15. }
```

Compara-se com o valor do dado  
presente no nó analisado.  
Logo, testa-se  $678 > 567$ ?

# Teste de Mesa

```
01. void insere(tipo_no *&no, int dado) {  
02.     if (no == NULL) {  
03.         no = new tipo_no;  
04.         no->dado = dado;  
05.         no->dir = NULL;  
06.         no->esq = NULL;  
07.     }else{  
08.         if (dado > no->dado) {  
09.              insere (no->dir, dado) ;  
10.         }else{  
11.             if (dado < no->dado) {  
12.                 insere (no->esq, dado) ;  
13.             }  
14.         }  
15.     }  
16. }
```

Como é verdadeira o teste, executa-se a instrução da linha 09

# Teste de Mesa

no = 0xF67B

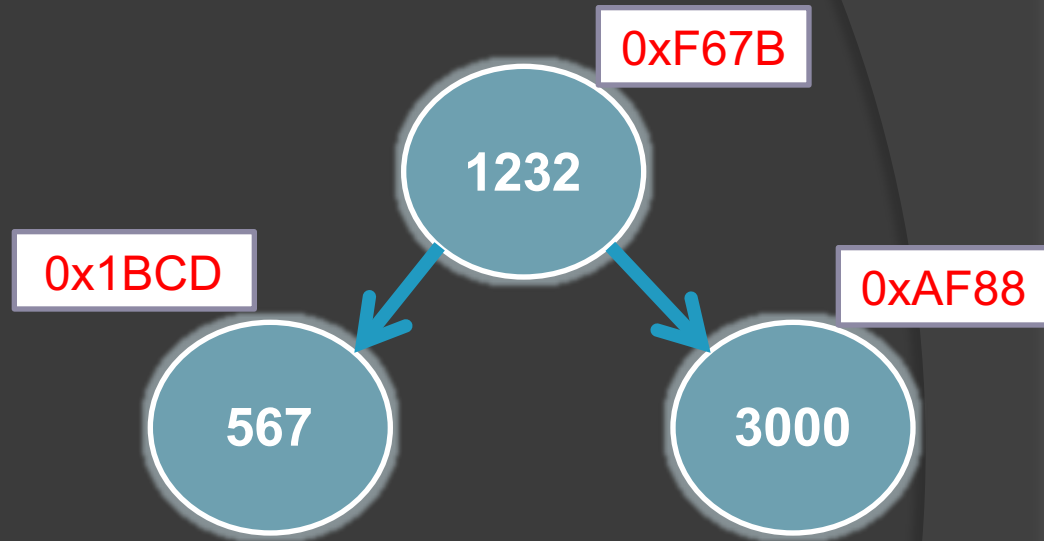
dado = 678

PC = 12


no = 0x1BCD

dado = 678

PC = 9



# Teste de Mesa

```
01. void insere(tipo_no *&no, int dado) {  
02.     if (no == NULL) {  
03.         no = new tipo_no;  
04.         no->dado = dado;  
05.         no->dir = NULL;  
06.         no->esq = NULL;  
07.     }else{  
08.         if (dado > no->dado) {  
09.              insere (no->dir, dado);  
10.         }else{  
11.             if (dado < no->dado) {  
12.                 insere (no->esq, dado);  
13.             }  
14.         }  
15.     }  
16. }
```

Na linha 09, há uma chamada recursiva, passando como parâmetro, o novo índice, 678; e o nó da direita do analisado, que está marcado como NULL

# Teste de Mesa

**no = 0xF67B**

**dado = 678**

**PC = 12**

**no = 0x1BCD**

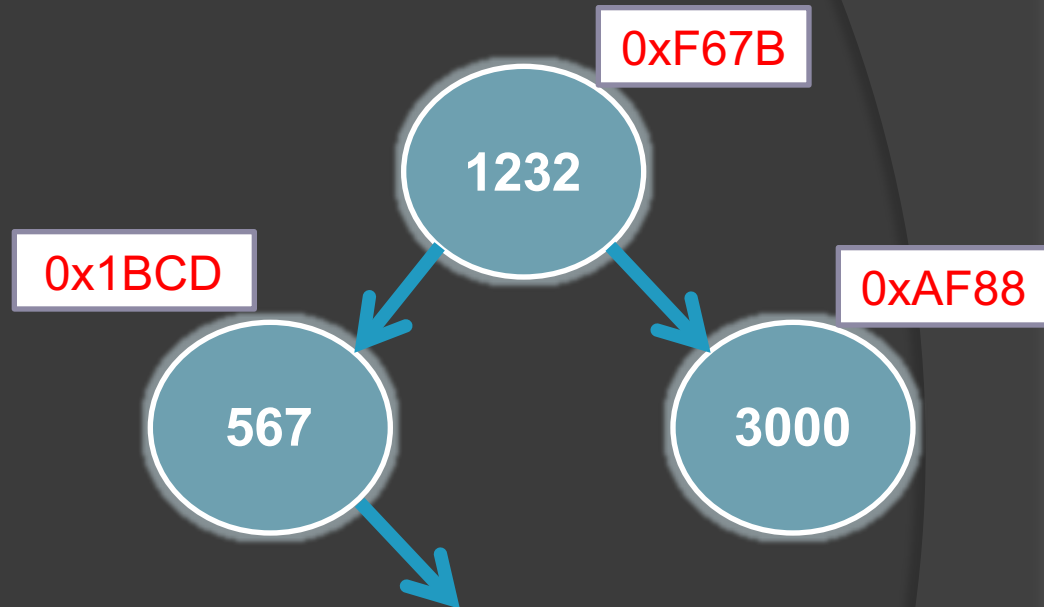
**dado = 678**

**PC = 9**

**no = NULL**


**dado = 678**

**PC = 2**





# Teste de Mesa

```
01. void insere(tipo_no *&no, int dado) {  
02.      if (no == NULL) {  
03.         no = new tipo_no;
```

o ponteiro nó possui NULL, logo a condição é verdadeira

```
do;  
L;  
L;  
08.         if (dado > no->dado) {  
09.             insere (no->dir, dado);  
10.         }else{  
11.             if (dado < no->dado) {  
12.                 insere (no->esq, dado);  
13.             }  
14.         }  
15.     }  
16. }
```

# Teste de Mesa

```
01. void insere(tipo_no *&no, int dado) {
02.     if (no == NULL) {
03.         → no = new tipo_no;
04.         no->dado = dado;
05.         no->dir = NULL;
06.         no->esq = NULL;
07.     } else {
08.         if (dado < no->dado) {
09.             insere (no->esq, dado);
10.         } else {
11.             if (dado < no->dado) {
12.                 insere (no->esq, dado);
13.             }
14.         }
15.     }
16. }
```

um novo nó é criado no parâmetro  
(passado por referência) à direita do nó  
0x1BCD

# Teste de Mesa

**no = 0xF67B**

**dado = 678**

**PC = 12**

**no = 0x1BCD**

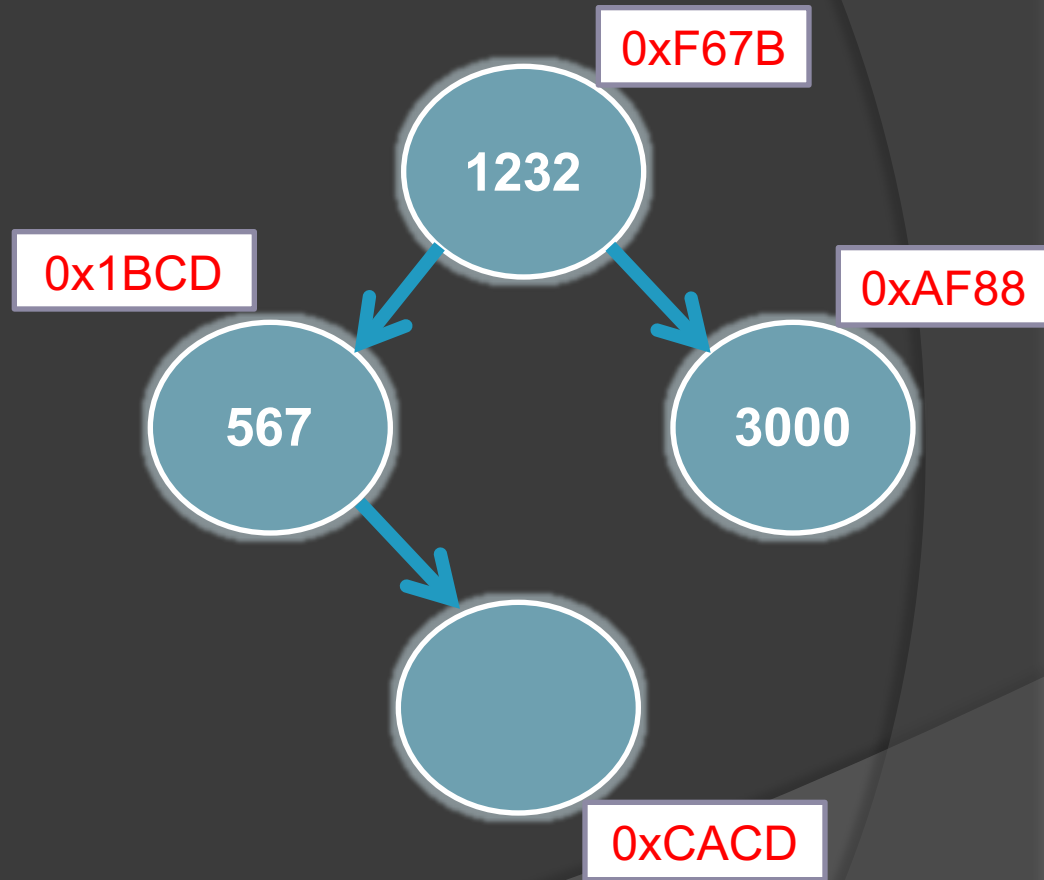
**dado = 678**

**PC = 9**

**no = 0xCACD**

**dado = 678**

**PC = 3**



# Teste de Mesa

**no = 0xF67B**

**dado = 678**

**PC = 12**

**no = 0x1BCD**

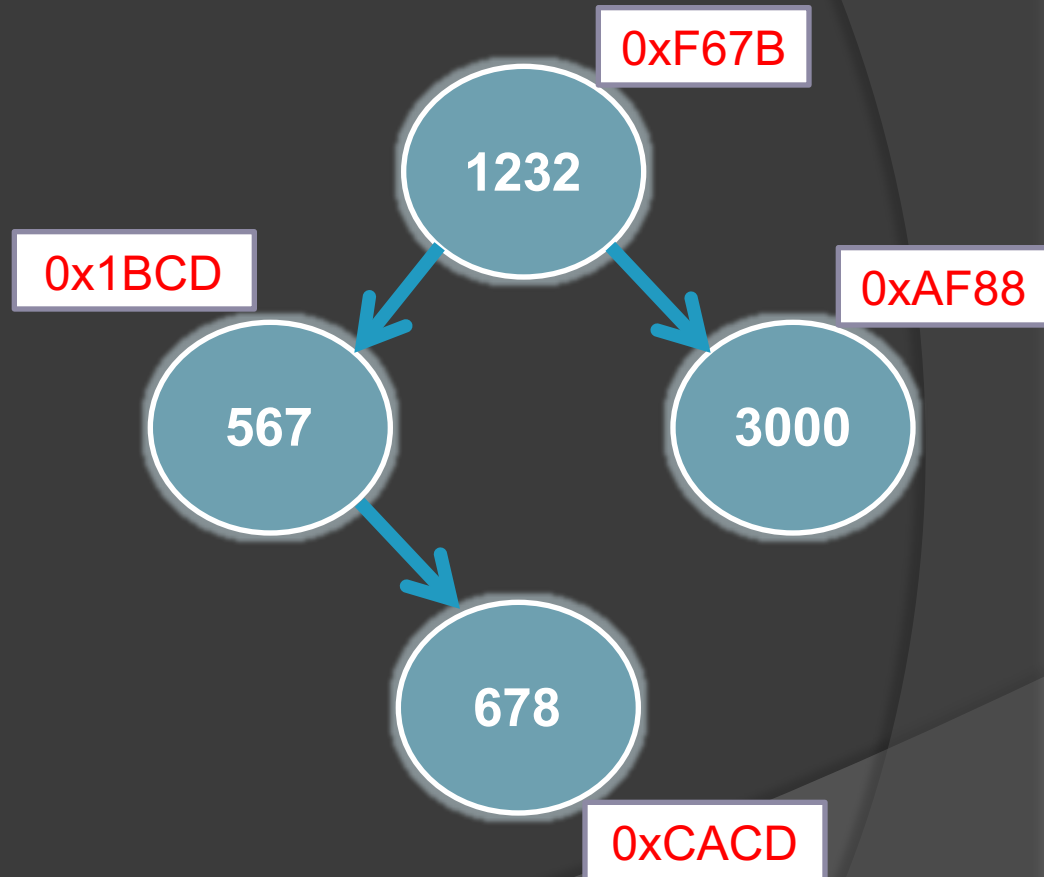
**dado = 678**

**PC = 9**


**no = NULL**

**dado = 678**

**PC = 3**



# Teste de Mesa

```
01. void insere(tipo_no *&no, int dado) {  
02.     if (no == NULL) {  
03.         no = new tipo_no;  
04.         no->dado = dado;  
05.          no->dir = NULL;  
06.         no->esq = NULL;  
07.     } else {  
08.         if (dado > no->dado) {  
09.             ;  
10.             ;  
11.             ;  
12.             ;  
13.         }  
14.     }  
15. }  
16. }
```

ponteiro dir do novo nó é marcado com NULL

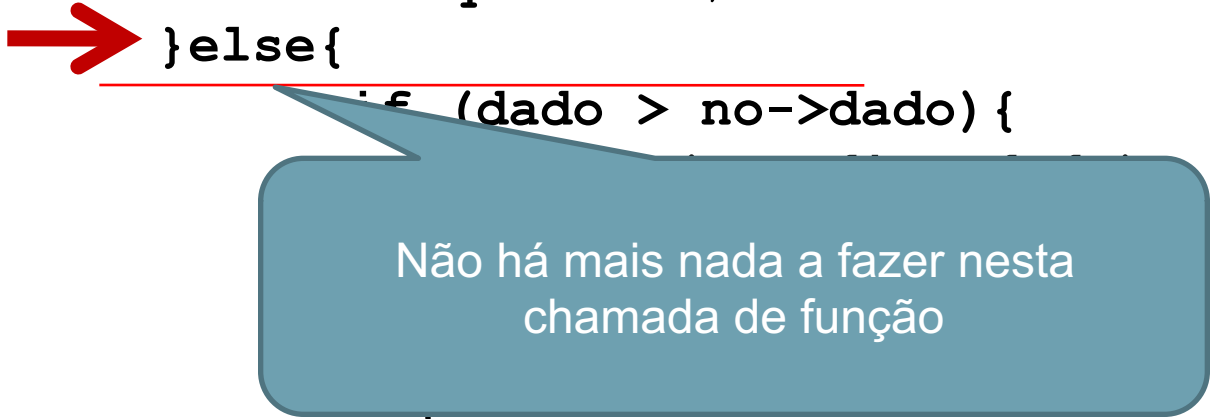
# Teste de Mesa

```
01. void insere(tipo_no *&no, int dado) {
02.     if (no == NULL) {
03.         no = new tipo_no;
04.         no->dado = dado;
05.         no->dir = NULL;
06.         → no->esq = NULL;
07.     } else {
08.         if (dado > no->dado) {
09.
10.
11.
12.         ;
13.     }
14. }
15.
16. }
```

ponteiro esq do novo nó é marcado com NULL

# Teste de Mesa

```
01. void insere(tipo_no *&no, int dado) {  
02.     if (no == NULL) {  
03.         no = new tipo_no;  
04.         no->dado = dado;  
05.         no->dir = NULL;  
06.         no->esq = NULL;  
07.     }else{  
08.         if (dado > no->dado) {  
09.             ;  
10.         }  
11.     }  
12. }  
13. ;  
14. }  
15. }  
16. }
```



Não há mais nada a fazer nesta chamada de função

# Teste de Mesa

no = 0xF67B

dado = 678

PC = 12

no = 0x1BCD

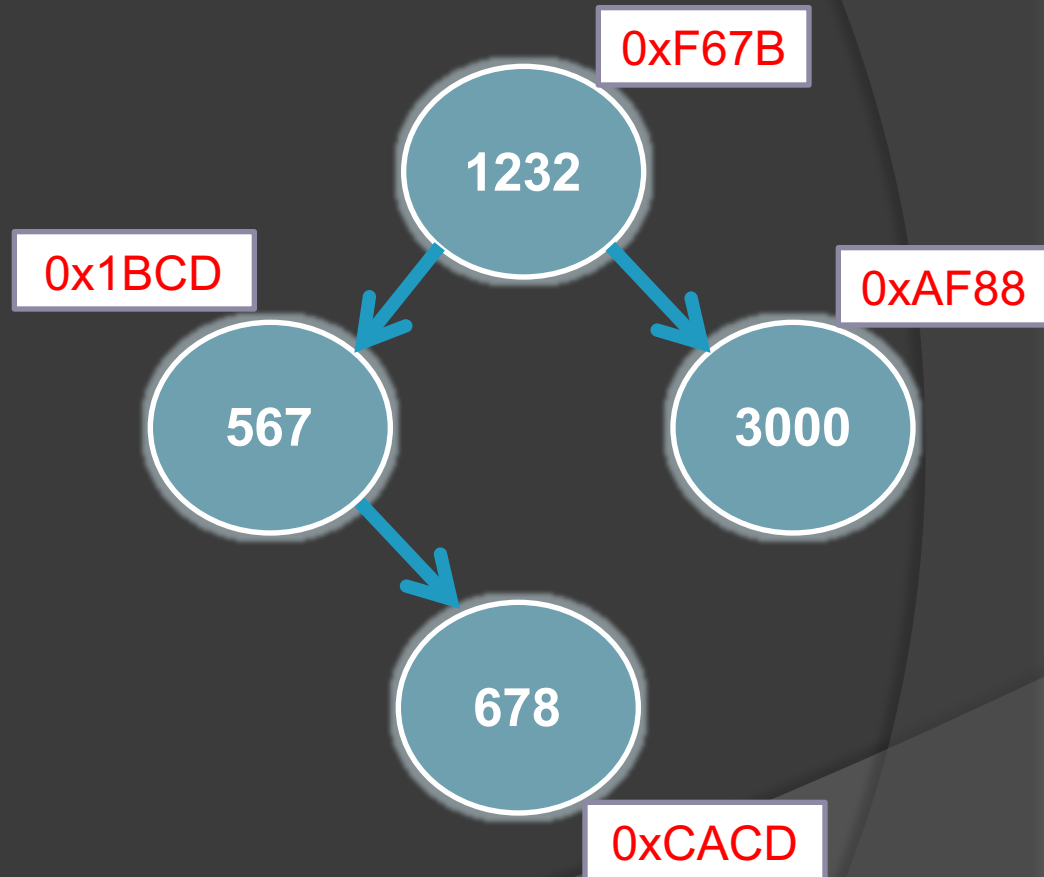
dado = 678

PC = 9

no = 0xCACD

dado = 678

PC = 7





# Teste de Mesa

no = 0xF67B

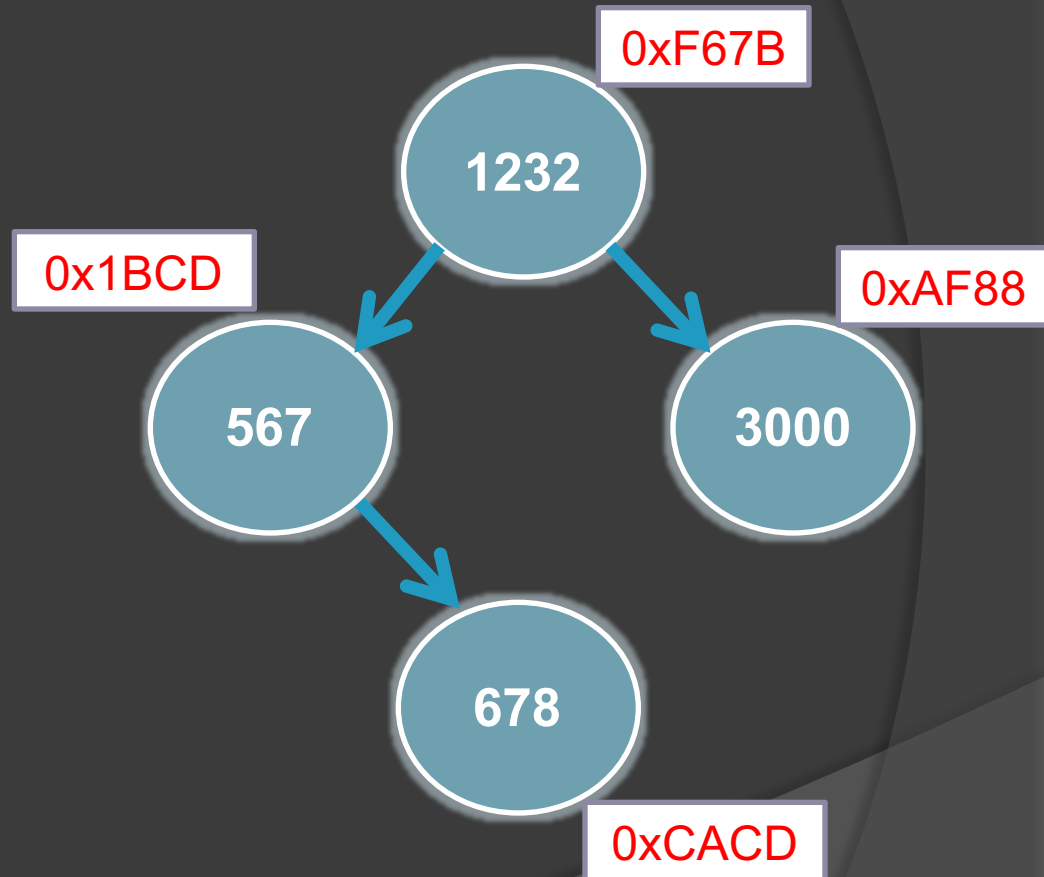
dado = 678

PC = 12

no = 0x1BCD

dado = 678


PC = 9



# Teste de Mesa

```
01. void insere(tipo_no *&no, int dado) {  
02.     if (no == NULL) {  
03.         no = new tipo_no;  
04.         // ...  
05.     }  
06.     if (dado > no->dado) {  
07.         insere (no->dir, dado);  
08.     } else {  
09.         if (dado < no->dado) {  
10.             insere (no->esq, dado);  
11.         }  
12.     }  
13. }  
14.  
15.  
16. }
```

Voltando a segunda chamada de função,  
não há mais nada a ser feito



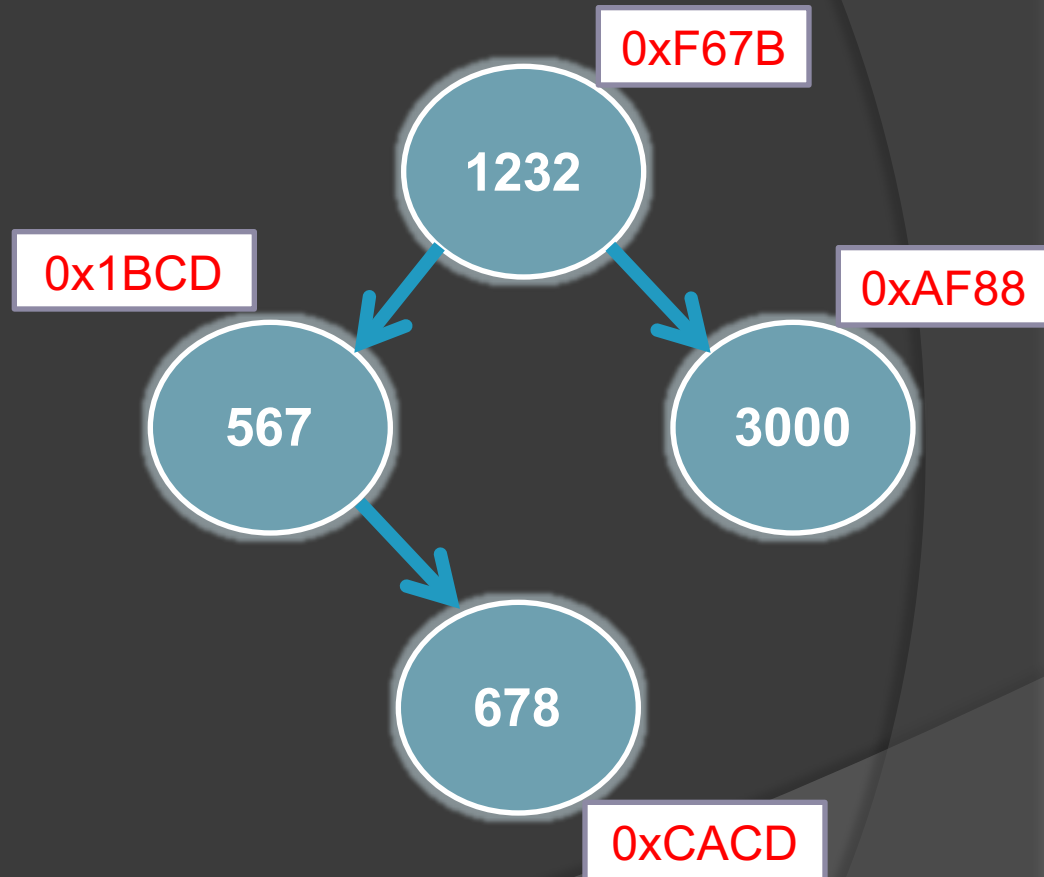
# Teste de Mesa

no = 0xF67B  
dado = 678

PC = 12

no = 0x1BCD  
dado = 678

PC = 10

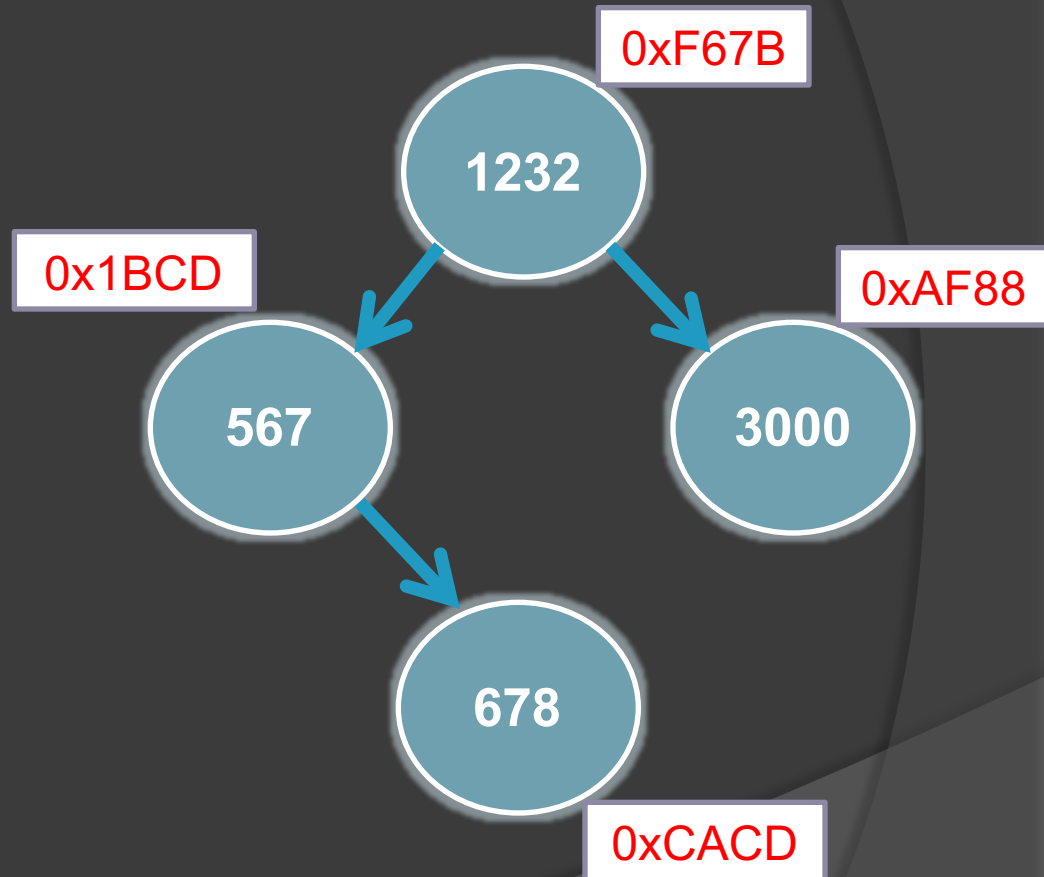


# Teste de Mesa

no = 0xF67B

dado = 678

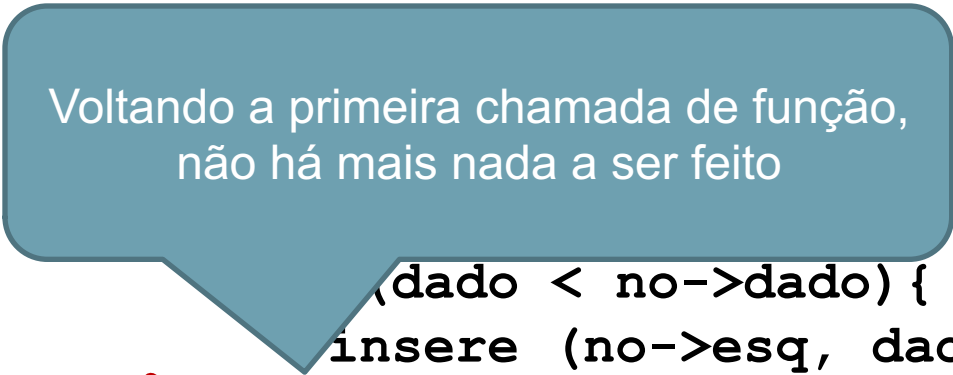
PC = 12



# Teste de Mesa

```
01. void insere(tipo_no *&no, int dado) {  
02.     if (no == NULL) {  
03.         no = new tipo_no;  
04.         no->dado = dado;  
05.         no->dir = NULL;  
06.         no->esq = NULL;  
07.     }  
08.     ;  
09.     if (dado < no->dado) {  
10.         insere (no->esq, dado) ;  
11.     }  
12. }  
13. }  
14. }  
15. }  
16. }
```

Voltando a primeira chamada de função,  
não há mais nada a ser feito

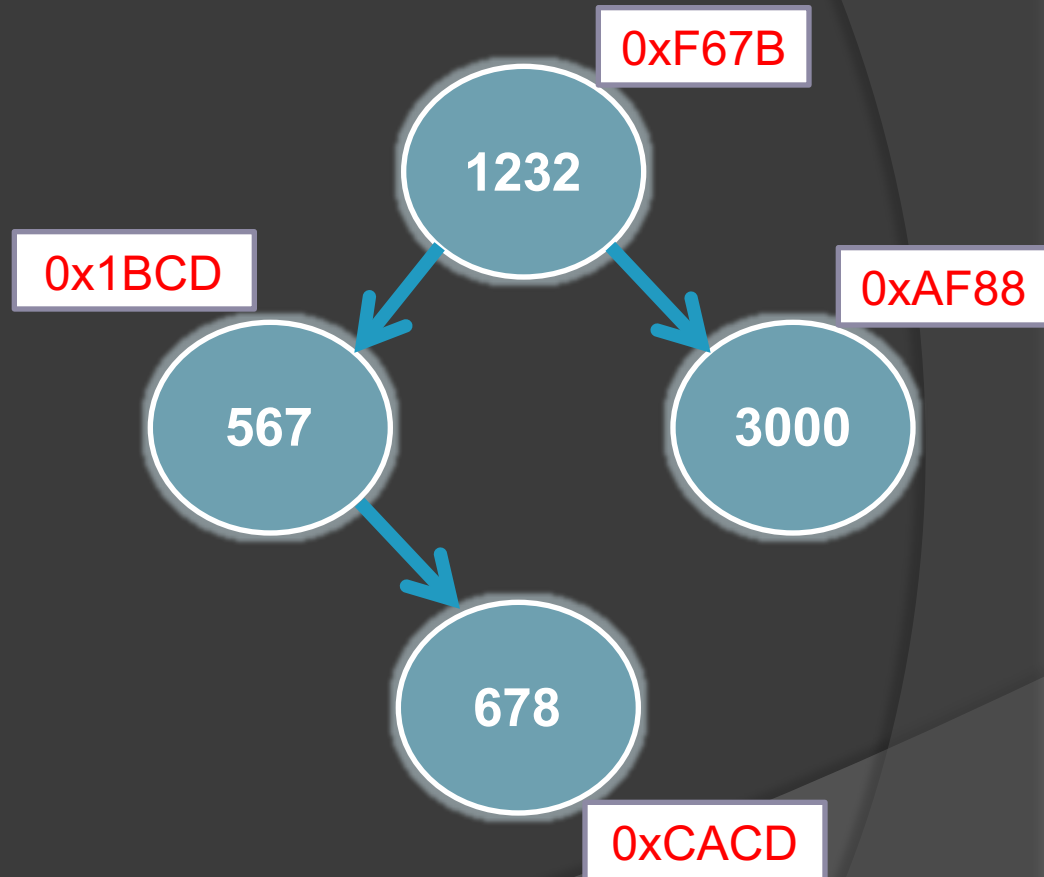


# Teste de Mesa

no = 0xF67B

dado = 678

PC = 13



# Teste de Mesa

PRONTO!!!

