Funções para alocação dinâmica de vetores (exemplos)

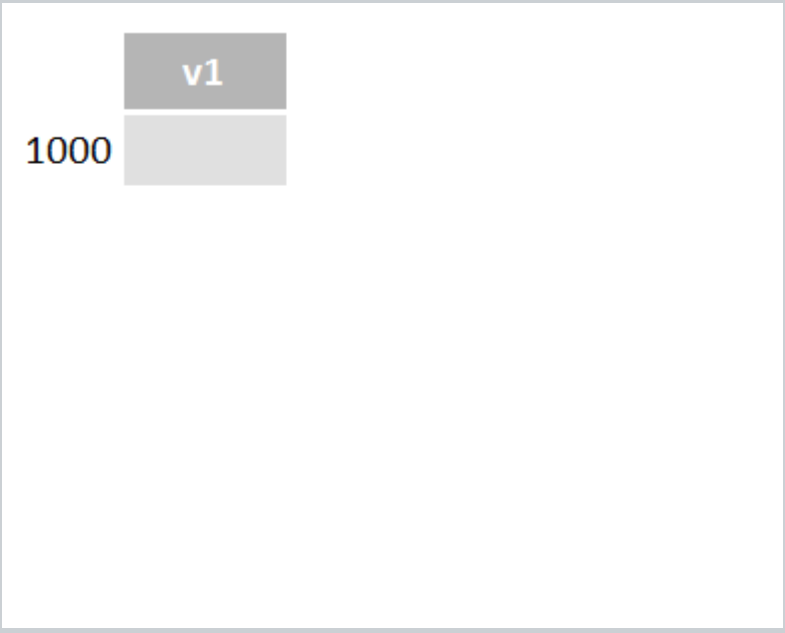
UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Na imagem ao lado podemos ver três implementações diferentes para uma função que aloca dinamicamente um vetor com *n* elementos. Analisaremos como cada função funciona separadamente.

Começando pela função **aloca1*. Esta função recebe como parâmetro o número de elementos que o vetor alocado deve conter, em outras palavras, quanta memória terá que ser alocada para esta variável. Esta função então retorna um ponteiro que aponta para o endereço de *v*, que contem o endereço do primeiro elemento do bloco de memória alocado.

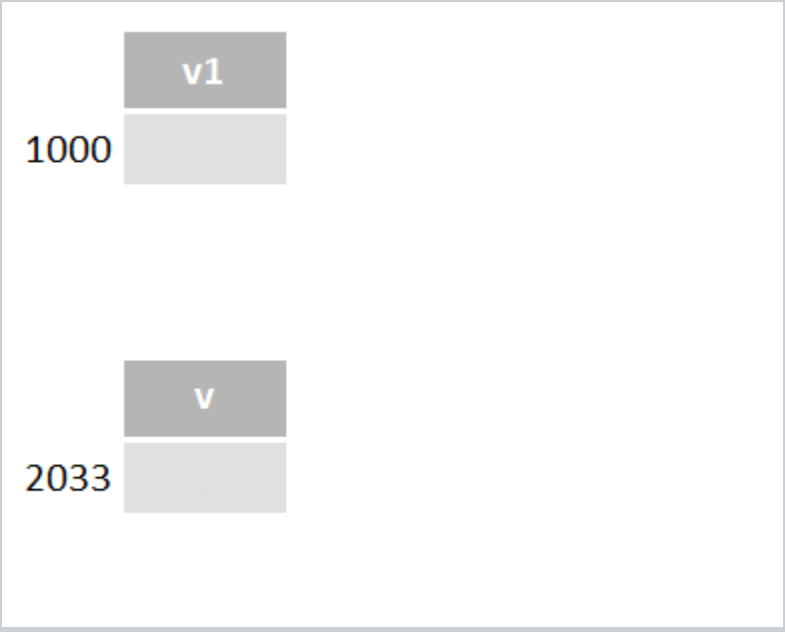
```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   int *aloca1(int n){
5       int *v = (int *)malloc(sizeof(int)*n);
6       return v;
7   }
8
9   void aloca2(int *v, int n){
10      v = (int *)malloc(sizeof(int)*n);
11  }
12
13  void aloca3(int **v, int n){
14      *v = (int *)malloc(sizeof(int)*n);
15  }
16
17
18  int main(){
19
20      int *v1 = aloca1(10);
21      v1[3] = 10;
22      printf("%d\n", v1[3]);
23
24
25      int *v2;
26      aloca2(v2, 10);
27      //v2[3] = 11;
28      //printf("%d\n", v2[3]);
29
30      int *v3;
31      aloca3(&v3,10);
32      v3[3] = 12;
33      printf("%d\n", v3[3]);
34
35  }
36
```

Abordando o funcionamento da função **aloca1* de forma mais detalhada, ela é usada na função *main* para realizar a inicialização do ponteiro **v1 (representado graficamente abaixo no endereço de memória 1000)*. É passado para a função o valor inteiro 10 como parâmetro que determina o número de elementos que o vetor resultante deve conter.



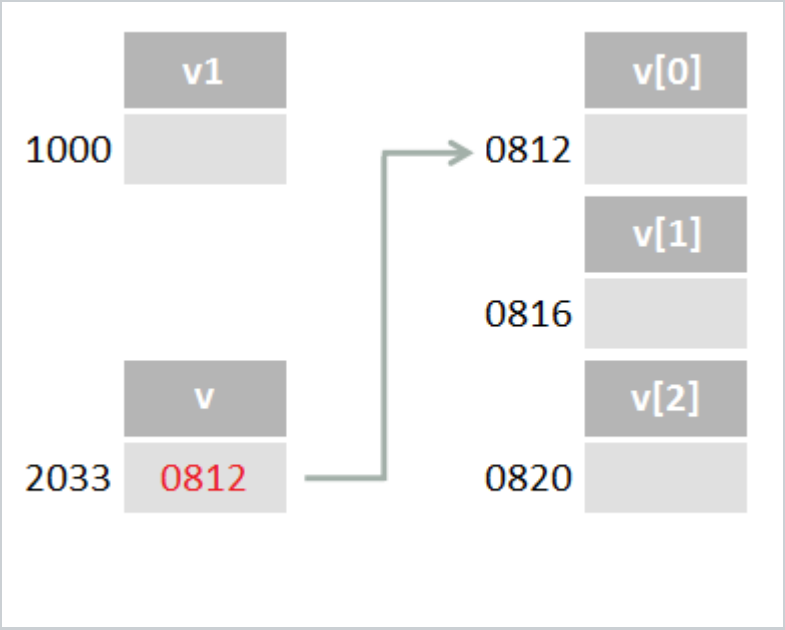

```c
#include <stdio.h>
#include <stdlib.h>

int *aloca1(int n){
    int *v = (int *)malloc(sizeof(int)*n);
    return v;
}

void aloca2(int *v, int n){
    v = (int *)malloc(sizeof(int)*n);
}

void aloca3(int **v, int n){
    *v = (int *)malloc(sizeof(int)*n);
}


int main(){

    int *v1 = aloca1(10);
    v1[3] = 10;
    printf("%d\n", v1[3]);


    int *v2;
    aloca2(v2, 10);
    //v2[3] = 11;
    //printf("%d\n", v2[3]);

    int *v3;
    aloca3(&v3,10);
    v3[3] = 12;
    printf("%d\n", v3[3]);

}
```

A função **aloca1* recebe o valor inteiro 10 como o parâmetro *n* e declara o ponteiro **v* (que foi alocado automaticamente no endereço 2033). Ela chama a função *malloc*, para realizar a inicialização de **v*. O parâmetro *n* é repassado para a função *malloc*.



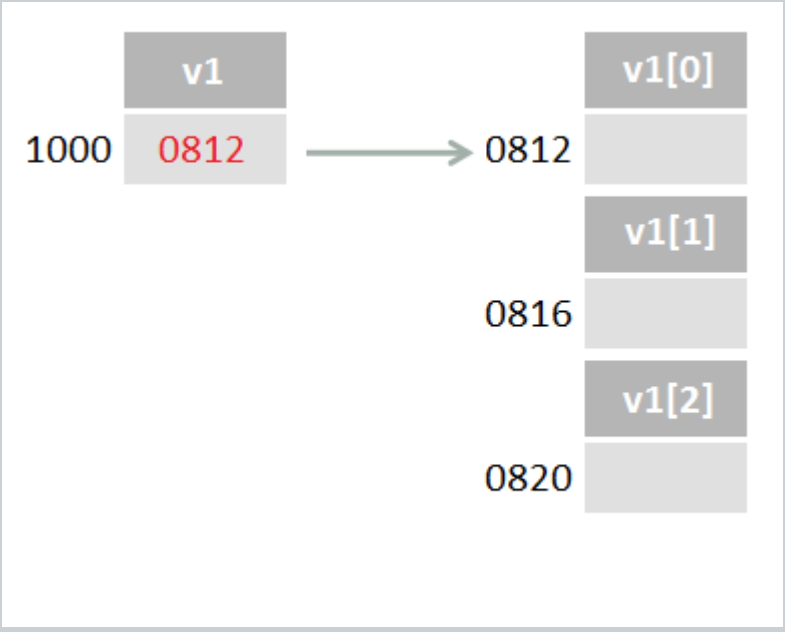

```c
#include <stdio.h>
#include <stdlib.h>

int *aloca1(int n){
    int *v = (int *)malloc(sizeof(int)*n);
    return v;
}

void aloca2(int *v, int n){
    v = (int *)malloc(sizeof(int)*n);
}

void aloca3(int **v, int n){
    *v = (int *)malloc(sizeof(int)*n);
}


int main(){

    int *v1 = aloca1(10);
    v1[3] = 10;
    printf("%d\n", v1[3]);


    int *v2;
    aloca2(v2, 10);
    //v2[3] = 11;
    //printf("%d\n", v2[3]);

    int *v3;
    aloca3(&v3,10);
    v3[3] = 12;
    printf("%d\n", v3[3]);

}
```

Ao terminar a alocação dinâmica, a função *malloc* retorna o endereço (0812) do primeiro elemento do bloco de memória, que é então atribuído ao ponteiro **v*.




```c
#include <stdio.h>
#include <stdlib.h>

int *aloca1(int n){
    int *v = (int *)malloc(sizeof(int)*n);
    return v;
}

void aloca2(int *v, int n){
    v = (int *)malloc(sizeof(int)*n);
}

void aloca3(int **v, int n){
    *v = (int *)malloc(sizeof(int)*n);
}


int main(){

    int *v1 = aloca1(10);
    v1[3] = 10;
    printf("%d\n", v1[3]);


    int *v2;
    aloca2(v2, 10);
    //v2[3] = 11;
    //printf("%d\n", v2[3]);

    int *v3;
    aloca3(&v3,10);
    v3[3] = 12;
    printf("%d\n", v3[3]);

}
```

Por fim a função **aloca1* retorna o endereço de memória apontado por **v*, que é então atribuído ao ponteiro **v1*, que passa a apontar para o primeiro elemento do bloco de memória alocado dinamicamente. Concluindo assim a alocação dinâmica do vetor *v1* composto por 10 elementos inteiros.




```c
#include <stdio.h>
#include <stdlib.h>

int *aloca1(int n){
    int *v = (int *)malloc(sizeof(int)*n);
    return v;
}

void aloca2(int *v, int n){
    v = (int *)malloc(sizeof(int)*n);
}

void aloca3(int **v, int n){
    *v = (int *)malloc(sizeof(int)*n);
}


int main(){

    int *v1 = aloca1(10);
    v1[3] = 10;
    printf("%d\n", v1[3]);


    int *v2;
    aloca2(v2, 10);
    //v2[3] = 11;
    //printf("%d\n", v2[3]);

    int *v3;
    aloca3(&v3,10);
    v3[3] = 12;
    printf("%d\n", v3[3]);

}
```

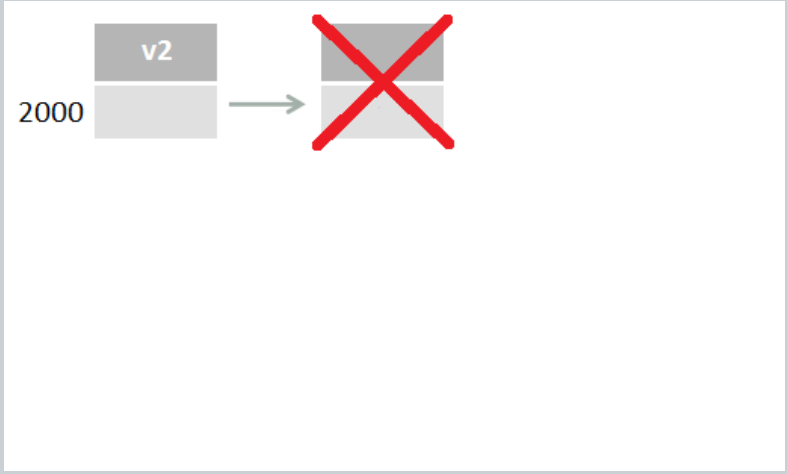
A função *aloca2* está incorreta, nela são recebidos como parâmetros, não só o número de elementos que o vetor deve conter, mas também a variável cuja memória deve ser alocada. Esta função tenta alocar o vetor alterando a variável que recebeu como um de seus parâmetros, porém foi implementada de forma incorreta.

Primeiramente, a variável *v2* não foi passada por referência de forma correta, o que significa que não é afetada pelo que ocorre dentro da função e, sendo assim, continua não inicializada após o fim da execução da função *aloca2*. Além disso, a função comete outro erro na hora de alocar a memória, alterando o valor de *v* (endereço apontado por *v) em vez de **v* (conteúdo do endereço apontado por **v*). Isso gera um vetor "solto" que não é usado e continua ocupando memória.

```c
#include <stdio.h>
#include <stdlib.h>

int *aloca1(int n){
    int *v = (int *)malloc(sizeof(int)*n);
    return v;
}

void aloca2(int *v, int n){
    v = (int *)malloc(sizeof(int)*n);
}

void aloca3(int **v, int n){
    *v = (int *)malloc(sizeof(int)*n);
}


int main(){

    int *v1 = aloca1(10);
    v1[3] = 10;
    printf("%d\n", v1[3]);


    int *v2;
    aloca2(v2, 10);
    //v2[3] = 11;
    //printf("%d\n", v2[3]);

    int *v3;
    aloca3(&v3,10);
    v3[3] = 12;
    printf("%d\n", v3[3]);

}
```

Para tornar mais fácil de se entender por que a função foi implementada de forma incorreta representaremos as variáveis na memória de forma gráfica.

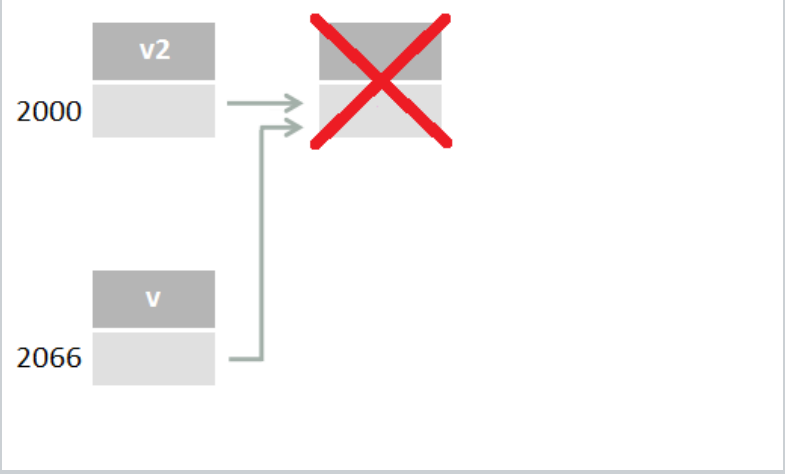
A função *aloca2* é chamada após a declaração do ponteiro **v2*. Aqui já podemos ver o primeiro erro. Em vez de passar o endereço do ponteiro **v2* (&v2) para a função, foi passado o valor contido nele (v2).




```c
#include <stdio.h>
#include <stdlib.h>

int *aloca1(int n){
    int *v = (int *)malloc(sizeof(int)*n);
    return v;
}

void aloca2(int *v, int n){
    v = (int *)malloc(sizeof(int)*n);
}

void aloca3(int **v, int n){
    *v = (int *)malloc(sizeof(int)*n);
}


int main(){

    int *v1 = aloca1(10);
    v1[3] = 10;
    printf("%d\n", v1[3]);


    int *v2;
    aloca2(v2, 10);
    //v2[3] = 11;
    //printf("%d\n", v2[3]);

    int *v3;
    aloca3(&v3,10);
    v3[3] = 12;
    printf("%d\n", v3[3]);

}
```

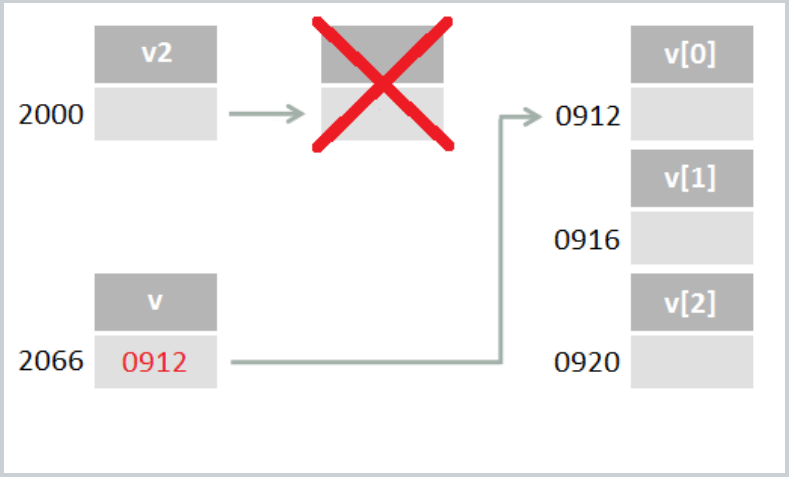
Por causa disso, o ponteiro *v* declarado dentro da função *aloca2* apontará para um endereço inválido quando deveria apontara para o endereço de *v2* (&v2) e a função não poderá alterar o ponteiro *v2*.

Mesmo que esse erro fosse corrigido ainda há outro erro que impediria o funcionamento adequado da função.



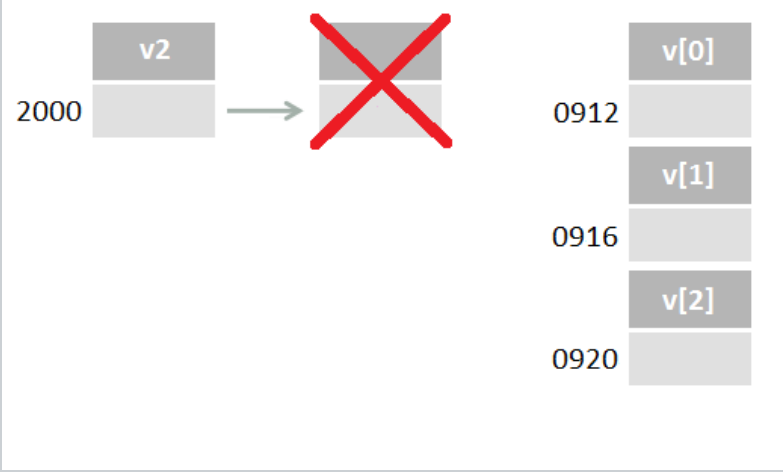

```c
#include <stdio.h>
#include <stdlib.h>

int *aloca1(int n){
    int *v = (int *)malloc(sizeof(int)*n);
    return v;
}

void aloca2(int *v, int n){
    v = (int *)malloc(sizeof(int)*n);
}

void aloca3(int **v, int n){
    *v = (int *)malloc(sizeof(int)*n);
}


int main(){

    int *v1 = aloca1(10);
    v1[3] = 10;
    printf("%d\n", v1[3]);


    int *v2;
    aloca2(v2, 10);
    //v2[3] = 11;
    //printf("%d\n", v2[3]);

    int *v3;
    aloca3(&v3,10);
    v3[3] = 12;
    printf("%d\n", v3[3]);

}
```

Esse segundo erro ocorre quando o endereço retornado pela função *malloc* é atribuído à *v* em vez de **v*. Mesmo que o primeiro erro fosse corrigido e **v* apontasse para **v2*, este segundo erro faria com que o ponteiro **v* passasse a apontar para o bloco de memória alocado em vez de alterar o endereço apontado por **v2* para ser o endereço do bloco de memória alocado.




```c
#include <stdio.h>
#include <stdlib.h>

int *aloca1(int n){
    int *v = (int *)malloc(sizeof(int)*n);
    return v;
}

void aloca2(int *v, int n){
    v = (int *)malloc(sizeof(int)*n);
}

void aloca3(int **v, int n){
    *v = (int *)malloc(sizeof(int)*n);
}


int main(){

    int *v1 = aloca1(10);
    v1[3] = 10;
    printf("%d\n", v1[3]);


    int *v2;
    aloca2(v2, 10);
    //v2[3] = 11;
    //printf("%d\n", v2[3]);

    int *v3;
    aloca3(&v3,10);
    v3[3] = 12;
    printf("%d\n", v3[3]);

}
```

Por causa destes erros, ao fim da execução da função *v2* continua apontando para um endereço de memória inválido e foi alocado um vetor “solto” na memória.



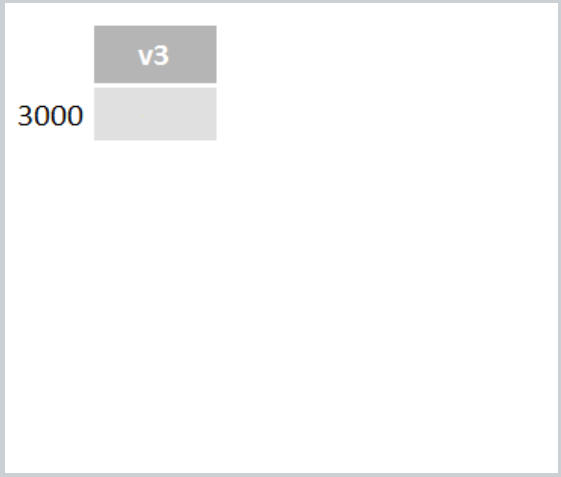

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   int *aloca1(int n){
5       int *v = (int *)malloc(sizeof(int)*n);
6       return v;
7   }
8
9   void aloca2(int *v, int n){
10      v = (int *)malloc(sizeof(int)*n);
11  }
12
13  void aloca3(int **v, int n){
14      *v = (int *)malloc(sizeof(int)*n);
15  }
16
17
18  int main(){
19
20      int *v1 = aloca1(10);
21      v1[3] = 10;
22      printf("%d\n", v1[3]);
23
24
25      int *v2;
26      aloca2(v2, 10);
27      //v2[3] = 11;
28      //printf("%d\n", v2[3]);
29
30      int *v3;
31      aloca3(&v3,10);
32      v3[3] = 12;
33      printf("%d\n", v3[3]);
34
35  }
36
```

Por fim a função *aloca3* apresenta a forma correta de implementar a função *aloca2*, corrigindo ambos os erros presentes nela.

```c
#include <stdio.h>
#include <stdlib.h>

int *aloca1(int n){
    int *v = (int *)malloc(sizeof(int)*n);
    return v;
}

void aloca2(int *v, int n){
    v = (int *)malloc(sizeof(int)*n);
}

void aloca3(int **v, int n){
    *v = (int *)malloc(sizeof(int)*n);
}


int main(){

    int *v1 = aloca1(10);
    v1[3] = 10;
    printf("%d\n", v1[3]);


    int *v2;
    aloca2(v2, 10);
    //v2[3] = 11;
    //printf("%d\n", v2[3]);

    int *v3;
    aloca3(&v3,10);
    v3[3] = 12;
    printf("%d\n", v3[3]);

}
```

Faremos novamente a representação gráfica das variáveis na memória para tornar o entendimento da função mais fácil.

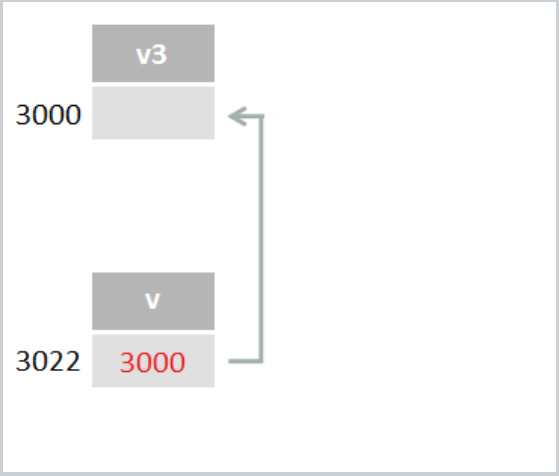
Assim como a função *aloca2*, a função *aloca3* recebe como parâmetros um inteiro e uma variável não inicializada.




```c
#include <stdio.h>
#include <stdlib.h>

int *aloca1(int n){
    int *v = (int *)malloc(sizeof(int)*n);
    return v;
}

void aloca2(int *v, int n){
    v = (int *)malloc(sizeof(int)*n);
}

void aloca3(int **v, int n){
    *v = (int *)malloc(sizeof(int)*n);
}


int main(){

    int *v1 = aloca1(10);
    v1[3] = 10;
    printf("%d\n", v1[3]);


    int *v2;
    aloca2(v2, 10);
    //v2[3] = 11;
    //printf("%d\n", v2[3]);

    int *v3;
    aloca3(&v3,10);
    v3[3] = 12;
    printf("%d\n", v3[3]);

}
```

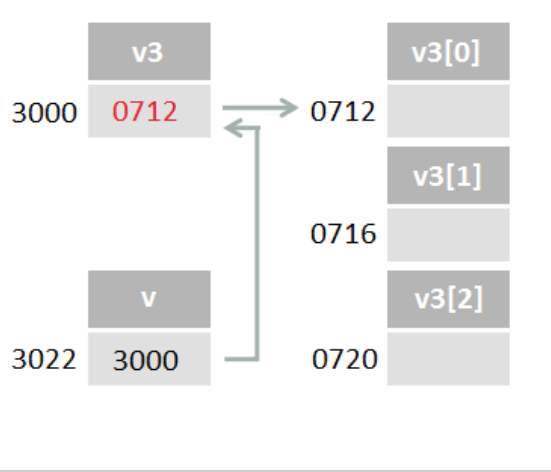
Porém, no caso da *aloca3* foi realiza a passagem de parâmetros por referência de forma correta, sendo assim, a função recebe o endereço do ponteiro **v3* (&v3), o que permitirá que ela altere esta variável.

O ponteiro de ponteiro ***v* recebe o endereço de **v3*, passando então a apontar para **v3*.



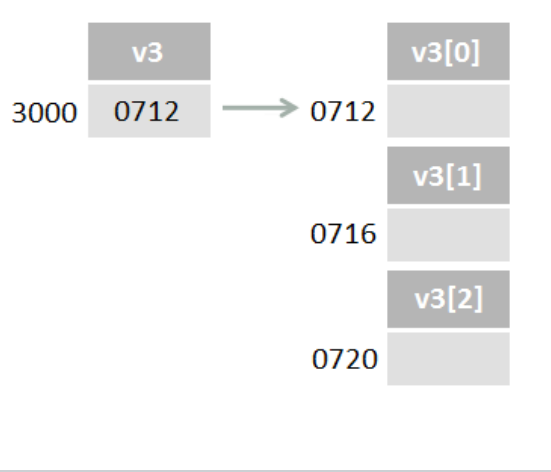

```c
#include <stdio.h>
#include <stdlib.h>

int *aloca1(int n){
    int *v = (int *)malloc(sizeof(int)*n);
    return v;
}

void aloca2(int *v, int n){
    v = (int *)malloc(sizeof(int)*n);
}

void aloca3(int **v, int n){
    *v = (int *)malloc(sizeof(int)*n);
}


int main(){

    int *v1 = aloca1(10);
    v1[3] = 10;
    printf("%d\n", v1[3]);


    int *v2;
    aloca2(v2, 10);
    //v2[3] = 11;
    //printf("%d\n", v2[3]);

    int *v3;
    aloca3(&v3,10);
    v3[3] = 12;
    printf("%d\n", v3[3]);

}
```

A função *malloc* é chamada para alocar um bloco de memória e o endereço retornado por ela é atribuído ao endereço de memória apontado por *v*, o que altera o endereço de memória apontado por **v3* que passa a apontar para o bloco de memória alocado.




```c
#include <stdio.h>
#include <stdlib.h>

int *aloca1(int n){
    int *v = (int *)malloc(sizeof(int)*n);
    return v;
}

void aloca2(int *v, int n){
    v = (int *)malloc(sizeof(int)*n);
}

void aloca3(int **v, int n){
    *v = (int *)malloc(sizeof(int)*n);
}


int main(){

    int *v1 = aloca1(10);
    v1[3] = 10;
    printf("%d\n", v1[3]);


    int *v2;
    aloca2(v2, 10);
    //v2[3] = 11;
    //printf("%d\n", v2[3]);

    int *v3;
    aloca3(&v3,10);
    v3[3] = 12;
    printf("%d\n", v3[3]);

}
```

Ao fim da execução da função *aloca3* o vetor foi alocado com sucesso e esta sendo corretamente apontado por **v3.*




```c
#include <stdio.h>
#include <stdlib.h>

int *aloca1(int n){
    int *v = (int *)malloc(sizeof(int)*n);
    return v;
}

void aloca2(int *v, int n){
    v = (int *)malloc(sizeof(int)*n);
}

void aloca3(int **v, int n){
    *v = (int *)malloc(sizeof(int)*n);
}


int main(){

    int *v1 = aloca1(10);
    v1[3] = 10;
    printf("%d\n", v1[3]);


    int *v2;
    aloca2(v2, 10);
    //v2[3] = 11;
    //printf("%d\n", v2[3]);

    int *v3;
    aloca3(&v3,10);
    v3[3] = 12;
    printf("%d\n", v3[3]);

}
```

Obrigado!

UNIVERSIDADE FEDERAL
DE SANTA CATARINA