

2-CALLOC

- Quase não ha diferença entre o calloc e o malloc.

```
int main(){//inicio_main
printf("\n*****\n");

int qtd, *p;

printf("Informe a quantidade de elementos para o vetor:\n");
scanf("%d",&qtd);

p = (int*)calloc(qtd, sizeof(int)); // em vez de * recebe 2 parametros.

if(p){
printf("A variavel 'p' ocupa %ld bytes em memoria.\n", qtd * sizeof(int));
}else{
printf("Erro: Memoria insuficiente!!!");
}

printf("\n*****\n");
printf("\n");
printf("\n");
return 0;
}

//fim_main
/*****ANOTAÇÕES*****/
* calloc(qtd, sizeof(int)); calloc(a,b) -> a * b;
*
```

- Malloc() recebe 1 valor(parametro)
- Calloc() recebe 2 valores.

-
- Quando o Malloc vai alocar uma memoria, ele ira alocar as quantidades de espaços desejadas, so que pode ser que ainda exista lixo nesses espaços (quase sempre), esses valores são mantidos e so são substituidos quando inicializamos;

```
PROG_C > 516(alocacao_dinamica_memoria) > 2-CALLOC > C: p68.c > (0) main()
1 //include <time.h>
2 //calloc
3
4 int main(){//inicio_main
5 printf("\n*****\n");
6
7 int qtd, *p;
8
9 printf("Informe a quantidade de elementos para o vetor:\n");
10 scanf("%d",&qtd);
11
12 p = (int*)malloc(qtd * sizeof(int)); // recebe 1 parametro. | 3 x 4 == 12 bytes
13 // em vez de * recebe 2 parametros. | 3 x 4 ==
14
15 if(p){
16 //p[0] = 6;
17 //p[1] = 12;
18 //p[2] = 34;
19 printf("A variavel 'p' ocupa %ld bytes em memoria.\n", qtd * sizeof(int));
20 for(int i = 0; i < qtd; i++){
21 printf("Valor de p[%d] = %d\n",i,p[i]);
22 }
23 }else{
24 printf("Erro: Memoria insuficiente!!!");
25 }
26
27 printf("\n*****\n");
28 printf("\n");
29 printf("\n");
30 return 0;
31 }
32 //fim_main
33
34 * calloc(qtd, sizeof(int)); calloc(a,b) -> a * b;
35 *
36 * Malloc
37 * 13571548
38 * 13571536
39 * 33554434
```

- Ha diferença é que o **malloc()** não remove lixo da memoria.
- Ja o **calloc()**, zera o espaço em memoria.

[illegible]