

1-MALLOC

- Como estamos falando de alocação dinamica, temos estaticamente valores declarados.

```
PS C:\Users\Gael\Documents\TERIA_INDIVIDUAL\UBOY_FEP\UBOY_FEB08_C316(alocação_dinâmica_memoria)\3-PRATICO > gcc -c main.c
```

```
1 //Declaração de variáveis e constantes
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 /*
6 */
7
8 int main(){//inicio_main
9     printf("você é o número %d\n", 9);
10
11     int numero = 9;
12
13     printf("Número = %d\n "Bytes_em_Memoria = sizeof(numero), numero, sizeof(numero));
14
15     printf("você é o número %d\n", 9);
16
17     //declaramos variáveis
18     //estruturas_dados
19     //processamento_dados
20     //valida_dados
21     printf("você");
22     printf("você");
23     return 0;
24 }//fim_main
```

int numero

9

```

TERMINAL ... C:\C++> Cppz ...
UDPP\PROG_C\516(alocação_dinamica_memoria)\1-MALLOC> cd
C:\Users\Victor\Documents\TEORIA_INDIVIDUAL\UDPP\REP_
UDPP\PROG_C\516(alocação_dinamica_memoria)\1-MALLOC>
PS C:\Users\Victor\Documents\TEORIA_INDIVIDUAL\UDPP\REP_
UDPP\PROG_C\516(alocação_dinamica_memoria)\1-MALLOC> cd
cd /c .\p67.exe

*****

Numeros[0] = 35
*Bytes_em_Memoria = 4

Numeros[1] = 43
*Bytes_em_Memoria = 4

Numeros[2] = 2
*Bytes_em_Memoria = 4

Memoria Total numeros[3] = 12
*****

PS C:\Users\Victor\Documents\TEORIA_INDIVIDUAL\UDPP\REP_
UDPP\PROG_C\516(alocação_dinamica_memoria)\1-MALLOC>

```

- Nem sempre sabemos quantos valores haverá no vetor, talvez a gente precise perguntar ao usuario quantos valores ele quer informar.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 //
5
6 //
7
8 int main(){ //inicio main
9     printf("\n*****\n");
10
11     int numeros[3]; // 0...n-1 -> 0...2
12
13     for(int i = 0; i < 3; i++){
14         printf("Informe o valor para a posicao %d do vetor:\n",i);
15         scanf("%d", &numeros[i]);
16         scanf("\n");
17     }
18
19     for( int i = 0 ; i < 3 ; i++){
20         printf("Numeros[%d] = %d\n "Bytes_em_Memoria = %d\n\n", i, numeros[i], sizeof(numeros[i]));
21     }
22
23     printf("Memoria Total numeros[3] = %d\n", sizeof(numeros));
24
25     printf("\n*****\n");
26
27     //declarando variaveis
28     //memoria_dados
29     //processamento_dados
30     //saida_dados
31     printf("\n");
32     printf("\n");
33     return 0;
34 } //fim main

```

- Transformamos o processo em loop.

- Para utilizarmos essa alocação dinamica temos que importar uma outra biblioteca chamada `<stdlib.h>`

- Vamos tbm declara um ponteiro pois eh a partir dele que conseguimos fazer a alocação.

- Duas maneiras de fazer...

```

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

    int qtd, *p;

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

    int bytes = qtd * sizeof(int);

    p = (int*)malloc(bytes); // 3 x 4 = 12 bytes -> isso eh colocado no ponteiro e ele aloca todo o espaco necessario.
    // ponteiro ira receber um ponteiro do tipo int e o malloc(quantidade * tamanho em bytes de inteiro.)

    for(int i = 0; i < qtd; i++){
        printf("Informe o valor para a posicao %d do vetor:\n", i);
        scanf("%d", &p[i]);
        printf("\n");
    }

    for(int i = 0; i < qtd; i++){
        printf(" *p[%d] = %d\n *Bytes em Memoria = %d\n", i, p[i], sizeof(p[i]));
    }

    printf("Memoria Total p[%d] = %d", qtd, bytes);

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

    printf("\n");
    printf("\n");
    return 0;
} //fim_main

```

malloc retorna um ponteiro para um int

```

// C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
// C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c

Informe a quantidade de elementos para o vetor:
3
Informe o valor para a posicao 0 do vetor:
12
Informe o valor para a posicao 1 do vetor:
23
Informe o valor para a posicao 2 do vetor:
34

*q[0] = 12
*Bytes em Memoria = 4
*q[1] = 23
*Bytes em Memoria = 4
*q[2] = 34
*Bytes em Memoria = 4

Memoria Total p[3] = 12

// C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c

1 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
2 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
3 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
4 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
5 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
6 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
7 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
8 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
9 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
10 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
11 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
12 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
13 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
14 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
15 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
16 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
17 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
18 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
19 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
20 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
21 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
22 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
23 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
24 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
25 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
26 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
27 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
28 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
29 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
30 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
31 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
32 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
33 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
34 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
35 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
36 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
37 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
38 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
39 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
40 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
41 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
42 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
43 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
44 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
45 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
46 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
47 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
48 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
49 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
50 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
51 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
52 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
53 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
54 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
55 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
56 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
57 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
58 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
59 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
60 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
61 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
62 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
63 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
64 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
65 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
66 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
67 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
68 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
69 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
70 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
71 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
72 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
73 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
74 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
75 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
76 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
77 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
78 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
79 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
80 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
81 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
82 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
83 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
84 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
85 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
86 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
87 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
88 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
89 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
90 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
91 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
92 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
93 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
94 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
95 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
96 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
97 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
98 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
99 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c
100 // C:\Users\... \Documents\... \C1330\alocacao_dinamica_memoria\1-3-MALLOC.c

```

- Outra maneira

```

int qtd, *p;

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

p = (int*)malloc(qtd * sizeof(int)); //3 x 4 == 12 bytes

for(int i = 0; i < qtd; i++){
    printf("Informe o valor para a posicao %d do vetor: ", i);
    scanf("%d", &p[i]);
}

for(int i = 0; i < qtd; i++){
    printf("No vetor 'numeros[%d]' está o valor %d: \n", i, p[i]);
}

printf("A variável 'p' ocupa %d bytes em memória.\n", qtd * sizeof(int));

return 0;

```

- A partir do momento que fazemos a alocação de memoria, essa memoria ficará alocada independente se seu progama foi encerrado ou nao. Por isso, sempre temos que liberar a memoria.
free(p); // liberar a memoria (desalocar)

- O parametro de free é o proprio elemento que foi alocado.
- E por ultimo, temos que colocar nosso ponteiro recebendo NULL, uma medida de segurança para o ponteiro não ser reutilizado. Caso não seja anulado, um virus ou outro progama pode utilizar esse ponteiro e ele possui permissões dentro do seu codigo. Ao anulalo, desacoplamos o ponteiro ao endereço de memoria que ele foi alocado.

- No momento da alocação, pode ser que o dispositivo que o progama esteja sendo executado não tenha memoria disponivel, por isso antes de tentar fazer acesso a essa memoria, temos que checar.

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

/*
    int qtd; // variavel para perguntar ao usuario o quanto ele quer cadastras
    int numeros[3]; // 0..n-1 -> 0..2
*/

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*)malloc(qtd * sizeof(int)); // 3 x 4 = 12 bytes -> isso eh colocado no ponteiro e ele aloca todo o espaço
    // ponteiro ira receber um ponteiro do tipo int e o malloc(quantidade * tamanho em bytes de inteiro.)

    //Checagem
    if(p){
        for(int i = 0; i < qtd; i++){
            printf("Informe o valor para a posicao %d do vetor:\n", i);
            scanf("%d", &p[i]);
            printf("\n");
        }
        for(int i = 0; i < qtd; i++){
            printf(" *p[%d] = %d\n *Bytes_em_Memoria = %d\n", i, p[i], sizeof(p[i]));
        }

        printf("**Memoria Total p[%d] = %d", qtd, *p);
    } else {
        printf("***Erro de Alocação!***");
    }

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

    printf("\n");
    printf("\n");
    return 0;
} //fim_main

/*
    int qtd; // variavel para perguntar ao usuario o quanto ele quer cadastras
    int numeros[3]; // 0..n-1 -> 0..2
*/
```

- Vamos checar quando de memoria 1... elemento ira ocupar e observar o erro de alocação (sem memoria)!

```

1 // C:\Users\GleidyRui> gcc -std=c99 -c prog.c -o prog.o
2
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 int main() {
7     // Variáveis para perguntar ao usuário o quanto ele quer cadastrar
8     int qtde; // N. de elementos do vetor
9     int i;
10
11     // Inicializa o vetor
12     int *v;
13
14     // Pergunta a quantidade de elementos para o vetor
15     printf("Informe a quantidade de elementos para o vetor: ");
16     scanf("%d", &qtde);
17
18     // Aloca a memória
19     v = (int *) malloc(qtde * sizeof(int)); // 3 x 4 = 12 bytes -> isso de colocado no ponteiro e ele aloca todo o espaço
20     // quanto ele precisa um ponteiro do tipo int e o malloc(qtde * tamanho em bytes de inteiro)
21
22     // Verifica se a alocação foi bem sucedida
23     if (v == NULL) {
24         printf("Erro de Alocacao!\n");
25         return 1;
26     }
27
28     // Inicializa o vetor com zeros
29     for (i = 0; i < qtde; i++) {
30         v[i] = 0;
31     }
32
33     // Imprime o vetor
34     printf("Vetor: ");
35     for (i = 0; i < qtde; i++) {
36         printf("%d ", v[i]);
37     }
38     printf("\n");
39
40     // Libera a memória
41     free(v);
42
43     return 0;
44 }
45
46 // Compilado com sucesso!

```