


Alocação dinâmica

- Para que serve?




Alocação dinâmica

- Cadastro de funcionários de uma empresa?

```
struct funcionarios[1000];
```

Pode haver desperdício ou faltar espaço.



Malloc (stdlib.h)

```
void *malloc ( unsigned int num );
```

A função malloc() retorna um ponteiro genérico

//array de 50 inteiros

```
int *v = (int *) malloc ( 50 );
```

//string de 200 caracteres

```
char *c = (char *) malloc( 200 );
```

Para reservar memória dinamicamente

- Para reservar memória dinamicamente em um vetor, usa-se ponteiros da seguinte forma:

```
float *x;
```

```
x = ( float * ) malloc ( n * sizeof ( float ) ) ;
```

n é o numero de elementos que vai ter o array

Para reservar memória dinamicamente

- Para liberar a memória alocada dinamicamente:

```
float *x;
```

```
x = ( float * ) malloc ( n * sizeof ( float ) ) ;
```

```
//n é o numero de elementos que vai ter o array
```

```
free(x);
```

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


int main()
{

    //Se não tiver memória suficiente,
    //retorna NULL
    int *p = (int *) malloc( 5 * sizeof(int) );
    if ( p == NULL ) exit(1);

    int i;
    for ( i = 0; i < 5; i++ )
        scanf( "%d", &p[i] );

    //Libera memória
    free(p);

    return 0;
}
```



Calloc (stdlib.h)

```
void *calloc (unsigned int num, unsigned int size );
```

num: número de unidades alocadas

size: tamanho de cada unidade

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

int main()
{

    //Se não tiver memória suficiente,
    //retorna NULL
    int *p = (int *) calloc( 5, sizeof(int) );
    if ( p == NULL ) exit(1);

    int i;
    for ( i = 0; i < 5; i++ )
        scanf( "%d", &p[i] );

    //Libera memória
    free(p);

    return 0;
}
```




malloc vs. calloc

- Malloc

- ☐ Apenas aloca a memória

- Calloc

- ☐ Aloca a memória
- ☐ Inicializa todos os bits da memória com zero

realloc (stdlib.h)

- Útil para **alocar** ou **realocar** memória durante a execução do programa.
 - Ex.: Aumentar a quantidade de memória já alocada.

`void *realloc (void *ptr, unsigned int num);`

OS DADOS EM PTR **NÃO** SÃO PERDIDOS



realloc

```
void *realloc ( void *ptr, unsigned int num );
```

- ptr: ponteiro para bloco já alocado anteriormente
- num: número de bytes a ser alocado
- Retorna ponteiro para primeira posição do array ou NULL caso não seja possível alocar

```
int *v = (int *) malloc ( 50 * sizeof(int) );
```

```
Int *ptr;
```

```
ptr = v;
```

realloc

- Se `ptr == NULL`, então `realloc` funciona como `malloc`

```
int *p;
```

```
p = (int *) realloc( NULL, n * sizeof ( int ) );
```

```
p = (int *) malloc( n * sizeof(int) );
```

realloc

- Realloc pode ser utilizado para liberar a memória, ou seja, diminuir o espaço alocado

```
int *p = ( int * ) malloc ( n * sizeof ( int ) );  
p = ( int * ) realloc ( p, 0 );
```

realloc

- Realloc pode ser utilizado para liberar a memória

```
int *p = ( int * ) malloc ( 5000 * sizeof ( int ) );  
p = ( int * ) realloc ( p, 10000000 * sizeof ( int ) );
```

QUAL O PROBLEMA DO CÓDIGO ACIMA?



realloc

- Realloc pode ser utilizado para liberar a memória

- `int *p = (int *) malloc (50 * sizeof (int));`
- `p = (int *) realloc (p, 100 * sizeof (int));`

PONTEIRO PODE SER NULL E O VETOR ALOCADO
ANTERIORMENTE É PERDIDO



Exercício de alocação de memória

Elabore um programa que crie um novo tipo chamado pessoa, com os campos nome, sexo e ponteiro para idade. Leia do usuário o tamanho de um vetor de pessoas e faça esse tamanho ser para todos os vetores. Em seguida, faça a alocação dinâmica desse vetor de struct, usando malloc, calloc e realloc, mas com ponteiros diferentes, ou seja, serão criados 3 vetores. Em seguida preencha cada campo de cada vetor de pessoa e exiba os dados na tela.