Linguagem de Programação C

Memória Dinâmica

Cristiano Lehrer

http://www.ybadoo.com.br/

Alocação de Memória

```
void *malloc(size_t n_Bytes)
```

- onde `size_t` é normalmente definido como sendo

```
typedef unsigned int size_t
```

- A função `malloc` permite alocar o conjunto de *bytes* indicados pelo programador, devolvendo um ponteiro para o bloco de bytes criados ou `NULL`, caso a alocação falhe:
  - `void*` significa que o método retorna um ponteiro para qualquer tipo, isto é, retorna um endereço de memória.

Exemplo – sem Alocação Dinâmica

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

int main()
{
   char s[200];
   char outra[200];

   printf("Forneça o seu nome completo: ");
   fgets(s, 200, stdin);

   /* Copiar o nome para a segunda string */
   strcpy(outra, s);
   printf("Original: %sCópia: %s", s, outra);

   return 0;
}
```

Exemplo – com Alocação Dinâmica

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
    char s[200];
    char *ptr; // Ponteiro
    printf("Forneça o seu nome completo: ");
    fgets(s, 200, stdin);
    /* Alocar a memória necessária */
    ptr = (char *)malloc(strlen(s) + 1); // '\0' também conta
    if(ptr == NULL){
      printf("Problemas na Alocação de Memória!");
    }
    else {
       strcpy(ptr, s);
       printf("Original: %sCópia: %s", s, ptr);
       free(ptr); // Liberar a memória existente em ptr
    }
    return 0;
}
```

Realocação de Memória

```
void *realloc(void *ptr, size_t new_size)
```

- Permite alterar o número de *bytes* que estão associados a um bloco previamente criado utilizando a função `malloc`:
  - Se o bloco atualmente puder ser aumentado para suportar a nova dimensão, a memória adicional é também reservada, e é retornada `ptr`.
  - Se não existir espaço suficiente para prolongar o bloco, então é criado um novo bloco com a totalidade dos *bytes* necessários. Os dados são copiados para a nova localização, e é retornado o novo endereço.
  - Se o parâmetro `ptr` for igual a `NULL`, então a função comporta-se como a função `malloc`.
  - Se por algum motivo não for possível a alocação da memória ou se o número de *bytes* for igual a zero, é devolvido `NULL`.

Exemplo – Realocação de Memória

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
   char *str;

   str = (char *) malloc(10);
   strcpy(str, "Fulano");

   printf("String: %s\nEndereço: %p\n", str, str);
   str = (char *) realloc(str, 100);
   printf("String: %s\nEndereço: %p\n", str, str);
   free(str);

   return 0;
}
```

Liberação da Memória Alocada

```
void free(void *ptr)
```

- Libera a memória alocada através das funções `malloc` e `realloc`.