Laboratório de Introdução à Ciência da Computação I

Aula 14 – Alocação dinâmica de memória

Professor:
Jó Ueyama

Funções para alocação de memória

- malloc(), calloc(), realloc(), free()
- São funções utilizadas para trabalhar com alocação dinâmica (em tempo de execução) de memória
- A memória é alocada a partir de uma área conhecida como heap

Malloc

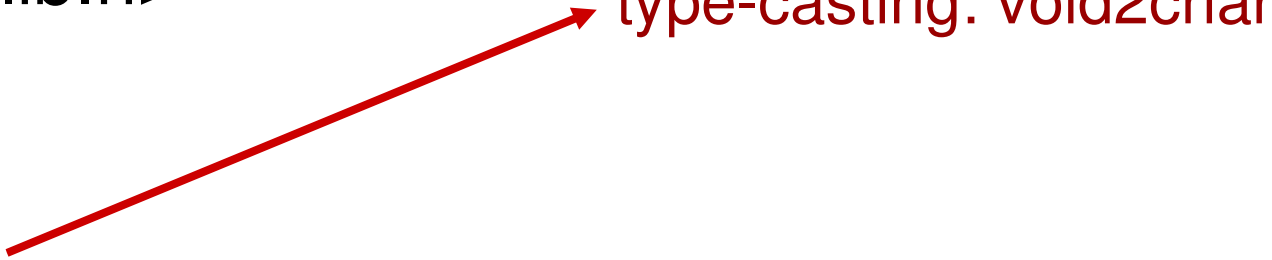
//void *malloc(size_t size);

- Size = tamanho do bloco de memória em bytes
- Retorna um ponteiro para o bloco de memória alocado
- Quando não conseguir alocar a memória, retorna um ponteiro nulo
- A região alocada contém valores desconhecidos
- Sempre verifique o valor de retorno!

Malloc

```
#include <stdlib.h>


char *str;
if((str = (char *)malloc(100)) == NULL)
{
   printf("Espaco insuficiente para alocar buffer \n");
    exit(1);
}
printf("Espaco alocado para str\n");
```



Malloc

```
#include <stdlib.h>

int *num;
if((num = (int *)malloc(50 * sizeof(int))) ==NULL)
{
    printf("Espaco insuficiente para alocar buffer \n");
    exit(1);
}
printf("Espaco alocado para num\n");
```

Calloc

//void * calloc (size_t num, size_t size);

- A função calloc() aloca um bloco de memória para um “array” de *num* elementos, sendo cada elemento de tamanho *size*
- A região da memória alocada é inicializada com o valor zero
- A função retorna um ponteiro para o primeiro byte
- Se não houver alocação, retorna um ponteiro nulo

Calloc

```
#include <stdlib.h>
unsigned int num;
int *ptr;
printf("Digite o numero de variaveis do tipo int: ");
scanf("%d", &num);
if((num = (int *)calloc(num, sizeof(int))) == NULL)
{
    printf("Espaco insuficiente para alocar \“num\” \n");
    exit(1);
}
printf("Espaco alocado com o calloc\n");
```

Realloc

//void * realloc (void * ptr, size_t size);

- A função realloc() aumenta ou reduz o tamanho de um bloco de memória previamente alocado com malloc() ou calloc()
- O argumento *ptr* aponta para o bloco original de memória e o *size* indica o novo tamanho desejado em bytes

Realloc

- Possíveis retornos:
  - Se houver espaço para expandir, a memória adicional é alocada e retorna *ptr*
  - Se não houver espaço suficiente para expandir o bloco atual, um novo bloco de tamanho size é alocado numa outra região da memória e o conteúdo do bloco original é copiado para o novo. O espaço de memória do bloco original é liberado e a função retorna um ponteiro para o novo bloco

Realloc

- Possíveis retornos (continuação):
  - Se o argumento *size* for zero, a memória indicada por *ptr* é liberada e a função retorna NULL
  - Se não houver memória suficiente para a realocação (nem para um novo bloco), a função retorna NULL e o bloco original permanece inalterado
  - Se o argumento *ptr* for NULL, a função atua como um malloc()

Exemplo: calloc seguido de realloc

```
unsigned int num; int *ptr;
printf("Digite o numero de variaveis do tipo int: ");
scanf("%d", &num);
if((ptr = (int *)calloc(num, sizeof(int))) == NULL){
      printf("Espaco insuficiente para alocar \"num\" \n");
      exit(1);
 }
//duplica o tamanho da região alocada para ptr
if((ptr = (int *)realloc(ptr, 2*num*sizeof(int))) == NULL){
      printf("Espaco insuficiente para alocar \"num\" \n");
       exit(1);
}
printf("Novo espaço \“realocado\“ com sucesso\n");
```

Free

//void free (void * ptr);

- “Desaloca”/libera um espaço de memória previamente alocado usando *malloc*, *calloc* ou *realloc*, tornando-o disponível para uso futuro
- A função deixa o valor de *ptr* inalterado, porém apontando para uma região inválida (note que o ponteiro não se torna NULL)
- Se for passado um ponteiro nulo, nenhuma ação será realizada
- Ex: free(ptr);

Exercício

- Escreva um programa que aloque a memória para cada caractere digitado pelo usuário e imprima no final o texto completo

- Implemente um código capaz de realizar a soma de duas matrizes, sendo os tamanhos e os conteúdos das mesmas informado pelo usuário

Referência

- http://www.cplusplus.com/reference/clibrary/cstdlib/r