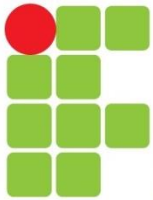

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
MATO GROSSO DO SUL

Algoritmos II

Alocação Dinâmica

Prof. Rogério Alves dos Santos Antoniassi
Prof. Alex Fernando de Araujo

Introdução

- Diz respeito ao processo de reservar/liberar memória para variáveis e estruturas de dados durante a execução de um programa.

- Na alocação estática, o tamanho é fixo em tempo de compilação, já na alocação dinâmica a memória cresce ou diminua conforme a necessidade.

- Permite ter mais eficiência no aproveitamento de memória.

- Uso da <stdlib.h>

malloc()

- malloc (*memory allocation*) é a forma mais direta de solicitar memória. Reservando o número de bytes solicitados e retorna um ponteiro para o primeiro byte do espaço (bloco) reservado.

- Sintaxe:
  - void* malloc(tamanho_em_bytes);

- Retorna um ponteiro do tipo void* (início do bloco alocado). Se ela retorna NULL (ex: Falta de memória).

Obs: O conteúdo do espaço alocado é indefinido (contém lixo), dessa forma deve ser inicializado.

Exemplo

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

int main(void) {

    int *p_num;

    p_num = malloc(sizeof(int));

    if (p_num == NULL) {
        printf("Erro ao alocar memória! \n");
        return 1;
    }

    *p_num = 42;

    printf("O valor é: %d \n", *p_num);
    printf("Endereço de memória: %p\n", (void *)p_num);

    free(p_num);
    return 0;
}
```

calloc()

- calloc (*contiguous allocation*) ideal para alocar vetores.

- Sintaxe:
  - void* calloc(num_elementos, tamanho_elemento).

- calloc() além de alocar o espaço de memória solicitado, inicializa o espaço alocado com 0, já malloc() não.

Exemplo

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

int main(void) {

    float *p_notas;
    int num = 4;

    p_notas = (float *) calloc(num, sizeof(float));

    if (p_notas == NULL) {
        printf("Erro ao alocar memória! \n");
        return 1;
    }

    printf("Vetor inicializado:\n");
    for (int cont = 0; cont <= num; cont++) {
        printf("%d nota: %.1f \n", cont + 1, p_notas[cont]);
    }

    free(p_notas);

    return 0;
}
```

realloc() - Redimensionando a memória

- realloc() permite aumentar ou diminuir e um bloco de memória previamente alocado.

- Sintaxe:
  - void* realloc(ponteiro, novo_tamanho).

- Faz o redimensionamento do bloco de memória apontado por ponteiro. O conteúdo original é preservado até o limite do menor tamanho.

Dica: Isso pode mover o bloco de memória de local no caso de pouco espaço, dessa forma, é importante reatribuir o ponteiro.

Exemplo

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

int main(void) {
    int *vet = NULL;
    int val, tamanho = 0;

    printf("Informe os números ou -1 para sair:\n");

    while (1) {
        scanf("%d", &val);
        if (val == -1) break;

        tamanho++;
        int *temp = realloc(vet, tamanho * sizeof(int));

        if (temp == NULL) {
            printf("Falha ao realocar memória! \n");
            free(vet);
            return 1;
        }
        vet = temp;
        vet[tamanho - 1] = val;
    }

    printf("\n Números informados: ");
    for(int cont = 0; cont < tamanho; cont++) {
        printf("%d ", vet[cont]);
    }

    free(vet);
    return 0;
}
```

free()

- Devolve o bloco de memória apontado por um ponteiro para o heap (liberando para uso futuro).

- Sintaxe:
  - void free(void* ponteiro).

- Caso não usar o free() pode gerar um vazamento de memória (*memory leak*). O programa continua consumindo memória sem devolvê-la (Pode usar toda a memória e travar).

Referências

- ASCENCIO, Ana Fernanda Gomes; DE CAMPOS, Edilene Aparecida Veneruchi. **Fundamentos da programação de computadores**. Pearson Educación, 2008.
- BACKES, André. ***Linguagem C: completa e descomplicada***. 2. ed., 4. imp. Rio de Janeiro: LTC, 2023.
- CORMEN, Thomas H. et al. **Algoritmos: teoria e prática**. Rio de Janeiro: Elsevier, c2012.
- DAMAS, Luís. **Linguagem C**. 10. ed. Rio de Janeiro: LTC, 2007.
- EVARISTO, Jaime. **Aprendendo a programar: programando em linguagem C**. Rio de Janeiro: Book Express, 2001. 205 p.
- MEDINA, Marco. **Algoritmos e programação: teoria e prática**. São Paulo: Novatec, 2006.
- MIZRAHI, Victorine Viviane. **Treinamento em Linguagem C**. 2. ed. São Paulo: Pearson, 2008.
- LAUREANO, Marcos. **Estrutura de Dados com Algoritmos e C**. Rio de Janeiro: Brasport, 2008.