CENTRO UNIVERSITÁRIO UNIFAVIP | WYDEN

Estrutura de Dados

Alocação Dinâmica de Memória

Wolney Henrique Queiroz Freitas
wolney.freitas@professores.unifavip.edu.br

Roteiro de Aula

- Alocação Dinâmica de Memória
  - Conceito;
  - Função malloc;
  - Função calloc;
  - Função realloc.

Relembrando Conceitos

- Uma variável é uma posição de memória que armazena um dado que pode ser usado pelo programa;

- Alocação pré-definida de memória de acordo com o tamanho da variável.

Relembrando Conceitos

- Arrays são agrupamentos sequenciais de dados de um mesmo tipo na memória;
- Um ponteiro guarda o endereço de um dado na memória;
- O nome do array é um ponteiro para o primeiro elemento do array.

Conceito de Alocação de Memória

- A linguagem C permite alocar (reservar) dinamicamente (em tempo de execução) blocos de memórias utilizando ponteiros;

- Funções
  - malloc;
  - calloc;
  - realloc;
  - free;

- Operador sizeof.

Operador sizeof

- Alocar memória do tipo int é diferente de alocar memória do tipo char:
  - Char: 1 byte;
  - Int: 4 bytes;
  - Float: 4 bytes;
  - Double: 8 bytes.

- O operador ajuda sizeof ajuda a saber quantos bytes serão necessários para alocar uma certa quantidade de dados, dependendo de seu tipo.

Operador sizeof

```c
#include <stdio.h>

struct aluno {
    int matricula;
    char nome[80];
};

main(){
    printf("\nchar: %d", sizeof(char)); // 1
    printf("\nint: %d", sizeof(int)); // 4
    printf("\nfloat: %d", sizeof(float)); // 4
    printf("\ndouble: %d", sizeof(double)); // 8
    printf("\nstruct aluno: %d", sizeof(struct aluno)); // 84
}
```

Função malloc()

- Serve para alocar memória durante a execução do programa;

- Ela faz o pedido de memória ao computador e retorna um ponteiro com o endereço do início do espaço de memória alocado.

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

Função malloc()

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

main(){

    // Cria array de 50 inteiros (200 bytes)
    // Retorna null caso ocorra erro
    int *v = (int*) malloc(200);

    // Cria array de 200 caracteres
    char *c = (char*) malloc(200);
}
```

Função malloc()

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

main(){

    // Quero 50 posições de inteiros
    int *v = (int*) malloc(50 * sizeof(int));

    // Quero 30 posições de char
    char *c = (char*) malloc(30 * sizeof(int));

    if(v == NULL || c == NULL){
        printf("Falha na alocação de memória.");
    }

    // Desaloca a memória
    free(v);
    free(c);
}
```

Função malloc()

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

main(){

    int *v = (int*) malloc(3 * sizeof(int));

    int i;
    for(i = 0; i < 3; i++){
        scanf("%d", &v[i]);
    }

    for(i = 0; i < 3; i++){
        printf("\n%d", v[i]);
    }
}
```

Vamos Praticar

- Utilizando a função malloc, aloque um vetor de 5 posições para cada tipo abaixo:
  - Float;
  - Int;
  - Char;
  - Struct (Crie sua própria struct).

Função calloc()

- Semelhante ao malloc();
- O que altera é que calloc preenche todo o espaço de memória alocado com o bit 0;
- O protótipo da função também é diferente:

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

Função calloc()

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

main(){

    // Vetor de inteiros de 50 posições
    int *v = (int*) calloc(50, 4);

    // Vetor de caracteres de 30 posições
    char *c = (char*) calloc(30, 1);
}
```

Função calloc()

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

main(){

    // Vetor de inteiros de 50 posições
    int *v = (int*) calloc(50, sizeof(int));

    // Vetor de caracteres de 30 posições
    char *c = (char*) calloc(30, sizeof(char));
}
```

Vamos Praticar

- Utilizando a função calloc, aloque um vetor de 5 posições para cada tipo abaixo:
  - Float;
  - Int;
  - Char;
  - Struct (Crie sua própria struct).

Função realloc()

- Serve para alocar e realocar memória durante a execução do programa;

- É possível redimensionar a memória previamente alocada.

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

Função realloc()

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

struct aluno{
    int matricula;
};

main(){

    struct aluno *m = (struct aluno*) malloc(3 * sizeof(struct aluno));

    struct aluno *m2 = (struct aluno*) realloc(m, 10 * sizeof(struct aluno));
}
```

Função realloc()

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

main(){

    // Se passar um ponteiro NULO, a função funciona
    // semelhante a um malloc
    int *p = (int *) realloc(NULL, 50*sizeof(int));

    int *p2 = (int *) malloc(50*sizeof(int));
}
```

Função realloc()

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

main(){

    int *p = (int *) malloc(50*sizeof(int));

    // Se realocar um ponteiro com 0 bytes
    // realloc funcionará semelhante a free()
    realloc(p, 0);
    free(p);
}
```

Vamos Praticar

- Crie um vetor de struct (crie sua própria struct) de 3 posições com a função malloc;
- Leia valores para preencher seu vetor e o exiba;
- Por fim, use a função realloc para adicionar mais 2 posições ao seu vetor, leia e exiba os valores do vetor realocado.

Faci facid FACIMP FBV Martha Falcão ISL UNIFAVIP UNIMETROCAMP RUY BARBOSA AREA1 UniFBV UniFanor