

Estrutura de Dados

Alocação Dinâmica

Prof. Luiz Gustavo Almeida Martins

Alocação Dinâmica

- Aloca **espaço de memória** para uma variável em **tempo de execução**
 - Permite o uso otimizado da memória
 - Muito usado para **variáveis indexadas** e **estruturas**
 - Alocação através das funções *malloc()*, *calloc()* e *realloc()*
- Função ***malloc()***
 - Presente na biblioteca ***stdlib.h***
 - **Entrada:** quantidade de bytes a ser alocado
 - **Saída:** endereço do **primeiro byte** alocado
 - Endereço do tipo **ponteiro genérico** (***void ****)
 - Deve ser convertido para o tipo do ponteiro desejado (**ex: *int ****)
 - Valor retornado deve ser guardado em uma variável ponteiro

Sintaxe:

ponteiro = (*tipo ponteiro **) ***malloc*** (*qtde. bytes*);

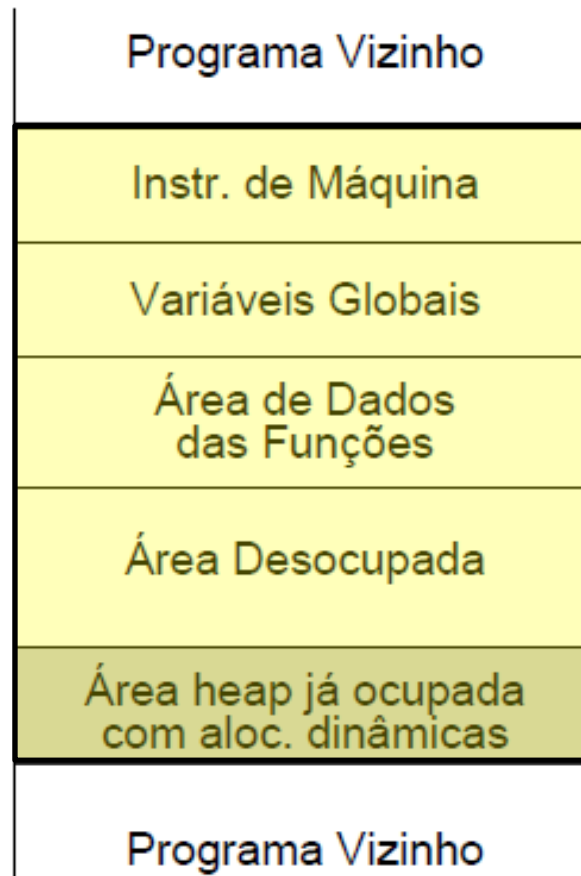
Alocação Dinâmica

- Quantidade de bytes alocados pode ser obtida através da função ***sizeof()***
 - **Entrada:** uma variável ou o **tipo do dado** que se deseja alocar (+ usual)
 - **Saída:** quantidade de bytes necessários para armazenar um dado do tipo definido
 - **Sintaxe:**

int sizeof(tipo do dado);

- Alocação usa uma região da memória chamada ***heap***
 - Bytes alocados são posicionados em **endereços contíguos**
- Operação de alocação pode falhar
 - Erro geralmente indica **falta de espaço na memória**
 - **Necessidade de verificação** antes de usar o ponteiro
*if (ponteiro == **NULL**)*

Layout de um Programa na Memória



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

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

```
int main () {
```

```
    int m, i, *A, *B, *C;
```

```
    printf("Entre tamanho dos vetores: ");
```

```
    scanf("%d", &m);
```

```
    A = (int *) malloc(m * sizeof(int)); // Aloca espaço para m números inteiros
```

```
    B = (int *) malloc(m * sizeof(int)); // Aloca espaço para m números inteiros
```

```
    C = (int *) malloc(m * sizeof(int)); // Aloca espaço para m números inteiros
```

```
    // Carrega os elementos do vetor A
```

```
    printf("\nVetor A: ");
```

```
    for (i = 0; i < m; i++)
```

```
        scanf("%d", &A[i]);
```

```
    // Carrega os elementos do vetor B
```

```
    printf("\nVetor B: ");
```

```
    for (i = 0; i < m; i++)
```

```
        scanf("%d", &B[i]);
```

```
    // Determina e apresenta os elementos do vetor C
```

```
    printf("\nVetor C: ");
```

```
    for (i = 0; i < m; i++) {
```

```
        C[i] = A[i] > B[i] ? A[i] : B[i];
```

```
        printf("%d ", C[i]);
```

```
    }
```

```
    return 0;
```

```
}
```

Exemplo

Slide adaptado do material do
Prof. Paulo Coelho

```
#include <stdio.h>
#include <stdlib.h>
int main () {
```

```
    int m, i, *A, *B, *C;
```

```
    printf("Entre tamanho dos vetores: ");
```

```
    scanf("%d", &m);
```

```
    A = (int *) malloc(m * sizeof(int)); // Aloca espaço para m números inteiros
```

```
    B = (int *) malloc(m * sizeof(int)); // Aloca espaço para m números inteiros
```

```
    C = (int *) malloc(m * sizeof(int)); // Aloca espaço para m números inteiros
```

```
    // Carrega os elementos do vetor A
```

```
    printf("\nVetor A: ");
```

```
    for (i = 0; i < m; i++)
```

```
        scanf("%d", &A[i]);
```

```
    // Carrega os elementos do vetor B
```

```
    printf("\nVetor B: ");
```

```
    for (i = 0; i < m; i++)
```

```
        scanf("%d", &B[i]);
```

```
    // Determina e apresenta os elementos do vetor C
```

```
    printf("\nVetor C: ");
```

```
    for (i = 0; i < m; i++) {
```

```
        C[i] = A[i] > B[i] ? A[i] : B[i];
```

```
        printf("%d ", C[i]);
```

```
    }
```

```
    return 0;
```

```
}
```

Exemplo

Slide adaptado do material do
Prof. Paulo Coelho

```
#include <stdio.h>
#include <stdlib.h>
int main () {
```

```
    int m, i, *A, *B, *C;
```

```
    printf("Entre tamanho dos vetores: ");
```

```
    scanf("%d", &m);
```

```
    A = (int *) malloc(m * sizeof(int)); // Aloca espaço para m números inteiros
```

```
    B = (int *) malloc(m * sizeof(int)); // Aloca espaço para m números inteiros
```

```
    C = (int *) malloc(m * sizeof(int)); // Aloca espaço para m números inteiros
```

```
    // Carrega os elementos do vetor A
```

```
    printf("\nVetor A: ");
```

```
    for (i = 0; i < m; i++)
```

```
        scanf("%d", &A[i]);
```

```
    // Carrega os elementos do vetor B
```

```
    printf("\nVetor B: ");
```

```
    for (i = 0; i < m; i++)
```

```
        scanf("%d", &B[i]);
```

```
    // Determina e apresenta os elementos do vetor C
```

```
    printf("\nVetor C: ");
```

```
    for (i = 0; i < m; i++) {
```

```
        C[i] = A[i] > B[i] ? A[i] : B[i];
```

```
        printf("%d ", C[i]);
```

```
    }
```

```
    return 0;
```

```
}
```

Exemplo

Slide adaptado do material do
Prof. Paulo Coelho

```
#include <stdio.h>
#include <stdlib.h>
int main () {
```

```
    int m, i, *A, *B, *C;
```

```
    printf("Entre tamanho dos vetores: ");
```

```
    scanf("%d", &m);
```

```
    A = (int *) malloc(m * sizeof(int)); // Aloca espaço para m números inteiros
```

```
    B = (int *) malloc(m * sizeof(int)); // Aloca espaço para m números inteiros
```

```
    C = (int *) malloc(m * sizeof(int)); // Aloca espaço para m números inteiros
```

```
    // Carrega os elementos do vetor A
```

```
    printf("\nVetor A: ");
```

```
    for (i = 0; i < m; i++)
```

```
        scanf("%d", &A[i]);
```

```
    // Carrega os elementos do vetor B
```

```
    printf("\nVetor B: ");
```

```
    for (i = 0; i < m; i++)
```

```
        scanf("%d", &B[i]);
```

```
    // Determina e apresenta os elementos do vetor C
```

```
    printf("\nVetor C: ");
```

```
    for (i = 0; i < m; i++) {
```

```
        C[i] = A[i] > B[i] ? A[i] : B[i];
```

```
        printf("%d ", C[i]);
```

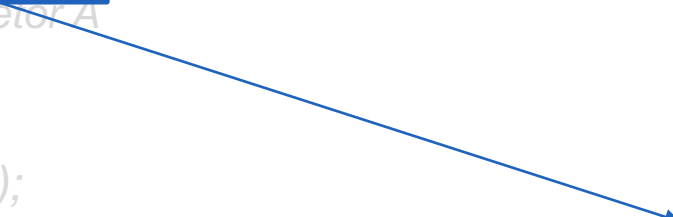
```
    }
```

```
    return 0;
```

```
}
```

Exemplo

Slide adaptado do material do
Prof. Paulo Coelho



Define o **tamanho em bytes** de **um** número inteiro


```
#include <stdio.h>
#include <stdlib.h>
int main () {
```

```
    int m, i, *A, *B, *C;
```

```
    printf("Entre tamanho dos vetores: ");
```

```
    scanf("%d", &m);
```

```
    A = (int *) malloc(m * sizeof(int)); // Aloca espaço para m números inteiros
```

```
    B = (int *) malloc(m * sizeof(int)); // Aloca espaço para m números inteiros
```

```
    C = (int *) malloc(m * sizeof(int)); // Aloca espaço para m números inteiros
```

```
    // Carrega os elementos do vetor A
```

```
    printf("\nVetor A: ");
```

```
    for (i = 0; i < m; i++)
```

```
        scanf("%d", &A[i]);
```

```
    // Carrega os elementos do vetor B
```

```
    printf("\nVetor B: ");
```

```
    for (i = 0; i < m; i++)
```

```
        scanf("%d", &B[i]);
```

```
    // Determina e apresenta os elementos do vetor C
```

```
    printf("\nVetor C: ");
```

```
    for (i = 0; i < m; i++) {
```

```
        C[i] = A[i] > B[i] ? A[i] : B[i];
```

```
        printf("%d ", C[i]);
```

```
    }
```

```
    return 0;
```

```
}
```

Exemplo

Slide adaptado do material do
Prof. Paulo Coelho



Define o quantidade de
elementos do vetor

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

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

```
int main () {
```

```
    int m, i, *A, *B, *C;
```

```
    printf("Entre tamanho dos vetores: ");
```

```
    scanf("%d", &m);
```

```
    A = (int *) malloc(m * sizeof(int)); // Conversão para o tipo de ponteiro desejado
```

```
    B = (int *) malloc(m * sizeof(int)); // Conversão para o tipo de ponteiro desejado
```

```
    C = (int *) malloc(m * sizeof(int)); // Conversão para o tipo de ponteiro desejado
```

```
    // Carrega os elementos do vetor A
```

```
    printf("\nVetor A: ");
```

```
    for (i = 0; i < m; i++)
```

```
        scanf("%d", &A[i]);
```

```
    // Carrega os elementos do vetor B
```

```
    printf("\nVetor B: ");
```

```
    for (i = 0; i < m; i++)
```

```
        scanf("%d", &B[i]);
```

```
    // Determina e apresenta os elementos do vetor C
```

```
    printf("\nVetor C: ");
```

```
    for (i = 0; i < m; i++) {
```

```
        C[i] = A[i] > B[i] ? A[i] : B[i];
```

```
        printf("%d ", C[i]);
```

```
    }
```

```
    return 0;
```

```
}
```

Exemplo

Slide adaptado do material do
Prof. Paulo Coelho

```
#include <stdio.h>
#include <stdlib.h>
int main () {
```

```
    int m, i, *A, *B, *C;
```

```
    printf("Entre tamanho dos vetores: ");
```

```
    scanf("%d", &m);
```

```
    A = (int *) malloc(m * sizeof(int)); // Conversão para o tipo de ponteiro desejado
```

```
    B = (int *) malloc(m * sizeof(int)); // Conversão para o tipo de ponteiro desejado
```

```
    C = (int *) malloc(m * sizeof(int)); // Conversão para o tipo de ponteiro desejado
```

```
    // Carrega os elementos do vetor A
```

```
    printf("\nVetor A: ");
```

```
    for (i = 0; i < m; i++)
```

```
        scanf("%d", &A[i]);
```

```
    // Carrega os elementos do vetor B
```

```
    printf("\nVetor B: ");
```

```
    for (i = 0; i < m; i++)
```

```
        scanf("%d", &B[i]);
```

```
    // Determina e apresenta os elementos do vetor C
```

```
    printf("\nVetor C: ");
```

```
    for (i = 0; i < m; i++) {
```

```
        C[i] = A[i] > B[i] ? A[i] : B[i];
```

```
        printf("%d ", C[i]);
```

```
    }
```

```
    return 0;
```

```
}
```

Exemplo

Slide adaptado do material do
Prof. Paulo Coelho

Conversão para um ponteiro
para o **tipo de dado definido**
no sizeof()

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

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

```
int main () {
```

```
    int m, i, *A, *B, *C;
```

```
    printf("Entre tamanho dos vetores: ");
```

```
    scanf("%d", &m);
```

```
    A = (int *) malloc(m * sizeof(int)); // Conversão para o tipo de ponteiro desejado
```

```
    B = (int *) malloc(m * sizeof(int)); // Conversão para o tipo de ponteiro desejado
```

```
    C = (int *) malloc(m * sizeof(int)); // Conversão para o tipo de ponteiro desejado
```

```
    // Carrega os elementos do vetor A
```

```
    printf("\nVetor A: ");
```

```
    for (i = 0; i < m; i++)
```

```
        scanf("%d", &A[i]);
```

```
    // Carrega os elementos do vetor B
```

```
    printf("\nVetor B: ");
```

```
    for (i = 0; i < m; i++)
```

```
        scanf("%d", &B[i]);
```

```
    // Determina e apresenta os elementos do vetor C
```

```
    printf("\nVetor C: ");
```

```
    for (i = 0; i < m; i++) {
```

```
        C[i] = A[i] > B[i] ? A[i] : B[i];
```

```
        printf("%d ", C[i]);
```

```
    }
```

```
    return 0;
```

```
}
```

Exemplo

Slide adaptado do material do
Prof. Paulo Coelho

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

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

```
int main () {
```

```
    int m, i, *A, *B, *C;
```

```
    printf("Entre tamanho dos vetores: ");
```

```
    scanf("%d", &m);
```

```
    A = (int *) malloc(m * sizeof(int)); // Conversão para o tipo de ponteiro desejado
```

```
    B = (int *) malloc(m * sizeof(int)); // Conversão para o tipo de ponteiro desejado
```

```
    C = (int *) malloc(m * sizeof(int)); // Conversão para o tipo de ponteiro desejado
```

```
    // Carrega os elementos do vetor A
```

```
    printf("\nVetor A: ");
```

```
    for (i = 0; i < m; i++)
```

```
        scanf("%d", &A[i]);
```

```
    // Carrega os elementos do vetor B
```

```
    printf("\nVetor B: ");
```

```
    for (i = 0; i < m; i++)
```

```
        scanf("%d", &B[i]);
```

```
    // Determina e apresenta os elementos do vetor C
```

```
    printf("\nVetor C: ");
```

```
    for (i = 0; i < m; i++) {
```

```
        C[i] = A[i] > B[i] ? A[i] : B[i];
```

```
        printf("%d ", C[i]);
```

```
    }
```

```
    return 0;
```

```
}
```

Exemplo

Slide adaptado do material do
Prof. Paulo Coelho

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

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

```
int main () {
```

```
→ int m, i, *A, *B, *C;
```

```
printf("Entre tamanho dos vetores: ");
```

```
scanf("%d", &m);
```

```
A = (int *) malloc(m * sizeof(int)); // Conversão para o tipo de ponteiro desejado
```

```
B = (int *) malloc(m * sizeof(int)); // Conversão para o tipo de ponteiro desejado
```

```
C → (int *) malloc(m * sizeof(int)); // Conversão para o tipo de ponteiro desejado
```

```
// Carrega os elementos do vetor A
```

```
printf("\nVetor A: ");
```

```
for (i = 0; i < m; i++)
```

```
    scanf("%d", &A[i]);
```

```
// Carrega os elementos do vetor B
```

```
printf("\nVetor B: ");
```

```
for (i = 0; i < m; i++)
```

```
    scanf("%d", &B[i]);
```

```
// Determina e apresenta os elementos do vetor C
```

```
printf("\nVetor C: ");
```

```
for (i = 0; i < m; i++) {
```

```
    C[i] = A[i] > B[i] ? A[i] : B[i];
```

```
    printf("%d ", C[i]);
```

```
}
```

```
return 0;
```

```
}
```

Exemplo

Slide adaptado do material do
Prof. Paulo Coelho

```
#include <stdio.h>
#include <stdlib.h>
int main () {
```

```
    int m, i, *A, *B, *C;
```

```
    printf("Entre tamanho dos vetores: ");
```

```
    scanf("%d", &m);
```

```
    A = (int *) malloc(m * sizeof(int)); // Aloca espaço para m números inteiros
```

```
    B = (int *) malloc(m * sizeof(int)); // Aloca espaço para m números inteiros
```

```
    C = (int *) malloc(m * sizeof(int)); // Aloca espaço para m números inteiros
```

```
    // Carrega os elementos do vetor A
```

```
    printf("\nVetor A: ");
```

```
    for (i = 0; i < m; i++)
```

```
        scanf("%d", &A[i]);
```

```
    // Carrega os elementos do vetor B
```

```
    printf("\nVetor B: ");
```

```
    for (i = 0; i < m; i++)
```

```
        scanf("%d", &B[i]);
```

```
    // Determina e apresenta os elementos do vetor C
```

```
    printf("\nVetor C: ");
```

```
    for (i = 0; i < m; i++) {
```

```
        C[i] = A[i] > B[i] ? A[i] : B[i];
```

```
        printf("%d ", C[i]);
```

```
    }
```

```
    return 0;
```

```
}
```

Exemplo

Slide adaptado do material do
Prof. Paulo Coelho

Ponteiros podem ser
**manipulados como
variáveis indexadas**

Alocação de Estruturas

- Estruturas são alocadas de modo similar aos tipos primitivos

Exemplo:

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

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

```
struct st {
```

```
    int a;
```

```
    float b;
```

```
};
```

```
typedef struct st st; // Define um "apelido" para o tipo estruturado
```

```
...
```

```
int main() {
```

```
    st *p; // Cria uma variável ponteiro p que aponta para um dado do tipo st
```

```
    p = (st *) malloc(sizeof(st)); // Aloca um dado do tipo st
```

```
    ...
```

```
}
```


Alocação de Estruturas

- Existem 2 formas de manipular os campos de uma variável estruturada alocada dinamicamente:
 - Sintaxe 1: ***(*ponteiro).campo***
 - Sintaxe 2: ***ponteiro->campo***
- **Exemplo:** considerando o programa anterior, podemos acessar os campos endereçados por ***p*** das seguintes formas:

(*p).a = 15; (*p).b = 3.12;

// sintaxe 1

printf("a = %d, b = %f\n", p->a, p->b);

// sintaxe 2

Liberação de Memória

- Variáveis que não serão utilizadas DEVEM ser liberadas
 - Recurso de memória é limitado
 - Permite reutilização de posições de memória
- Existem 2 formas de liberar uma variável dinâmica:
 - **Liberação implícita:**
 - Liberação ocorre quando **o programa é finalizado**.
 - TODO o espaço alocado para o programa é liberado.
 - **Liberação explícita:**
 - Liberação ocorre através da função **free()**
 - **Sintaxe:** `free(Ponteiro);`
 - **Ponteiro:** variável que aponta para a área da memória a ser liberada
 - Boa prática: atribuir **NULL** ao ponteiro após a liberação

Exercícios

1. *Faça um programa que leia o tamanho de um vetor de inteiros e reserve dinamicamente o espaço na memória para esse vetor. Em seguida, leia os elementos do vetor, calcule e mostre o resultado da soma dos números ímpares presentes no vetor e, por fim, libere o espaço alocado.*
2. *Faça um programa que leia o tamanho de uma string (vetor de caracteres) e chame uma função para alocar dinamicamente essa string. Em seguida, o usuário deverá informar o conteúdo da string e o programa mostrará a string digitada sem as vogais.*
3. *Faça um programa que leia as dimensões de uma matriz e aloque dinamicamente a memória para esta variável. Em seguida, o programa deve ler os elementos da matriz e imprimir somente os elementos que estão na parte superior da diagonal principal (inclusive ela).*

Exercícios

4. *Faça um programa que mantenha uma tabela (na memória) para cadastro de bebidas, onde cada dado tem a seguinte estrutura:*

Nome	Volume (ml)	Preço
char[20]	int	float

A tabela deve ser representada por um vetor de ponteiros de bebidas, o qual deve ser devidamente inicializado com NULL (esse vetor deve comportar no máximo 20 bebidas). O programa deve apresentar um menu com as seguintes opções:

[1] Inserir registro

[2] Apagar último registro

[3] Imprimir tabela

[4] Sair

Referências

- *Coelho, Paulo R. S. L., Linguagem C: Variáveis do Tipo Ponteiro, material didático da disciplina de Introdução a Programação, UFU.*
- *Backes, André, Linguagem C Descomplicada, portal de vídeo-aulas, <https://programacaodescomplicada.wordpress.com/>, acessado em 09/03/2016.*
- *Celes, W., Cerqueira, R. e Rangel, J. L. Introdução a estruturas de dados. Ed. Campus Elsevier, 2004.*