



Vetores e Alocação Dinâmica

Profa. Dra. Rosana Rego
rosana.rego@ufersa.edu.br



Agenda

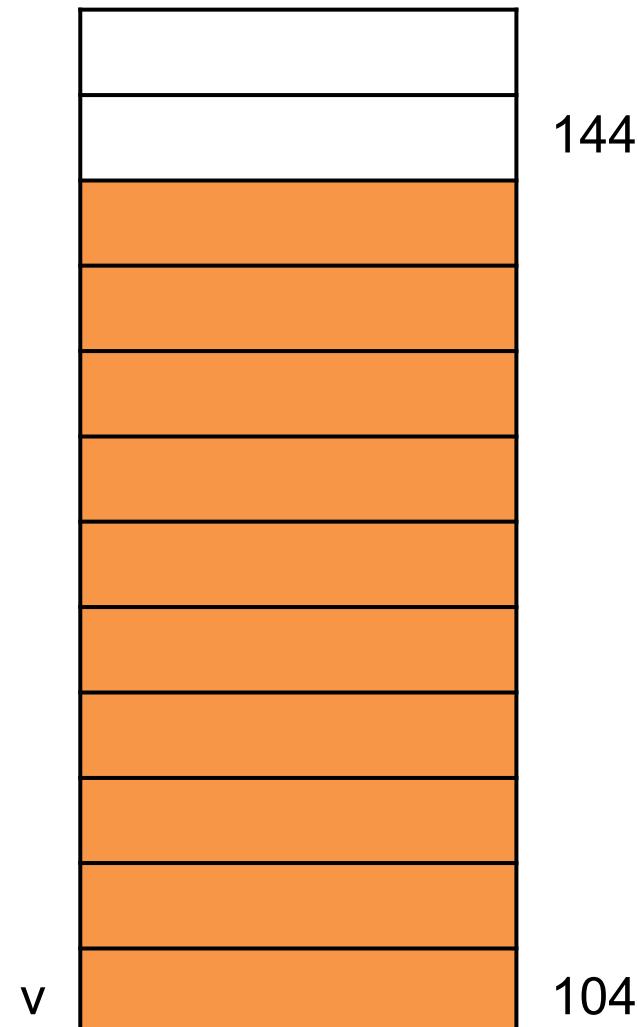
- Introdução
- Vetores
- Alocação dinâmica
- Vetores locais a funções

Vetores

- Definimos um vetor em C da seguinte forma:

```
int v[10];
```

- Reservamos um espaço de memória contínuo para armazenar o vetor.



Vetores

- O acesso a cada elemento é feito por meio de uma indexação de v:
 - Em C, varia de 0 a n-1, sendo n a dimensão do vetor.

$v[0]$ → acessa o primeiro elemento de v

$v[1]$ → acessa o segundo elemento de v

...

$v[9]$ → acessa o último elemento de v

$v[10]$ → está ERRADO (invasão de memória)



Vetores

- Existe uma forte associação entre vetores e ponteiros:
 - `int v[10];`
 - `v` representa o endereço inicial do vetor.



Vetores

- A linguagem C também suporta aritmética de ponteiros:

$v + 0 \rightarrow$ aponta para o primeiro elemento do vetor

$v + 1 \rightarrow$ aponta para o segundo elemento do vetor

$v + 2 \rightarrow$ aponta para o terceiro elemento do vetor

...

$v + 9 \rightarrow$ aponta para o décimo elemento do vetor



Vetores

- $\&v[i]$ é equivalente a $(v + i)$.
- $v[i]$ é equivalente a $*(v + i)$.
- Os vetores também podem ser inicializados na declaração:

`int v[5] = { 5, 10, 15, 20, 25};`

ou simplesmente:

`int v[] = { 5, 10, 15, 20, 25};`



Vetores

- Passagem de vetores para funções:
 - Passamos o endereço da primeira posição do vetor;
 - A função chamada deve ter um parâmetro do tipo ponteiro;
 - “passar um vetor para uma função” = “passar o endereço inicial do vetor”;
 - Os elementos do vetor não são copiados para a função.

Vetores

- Passagem de vetores para funções (Exemplo – parte I):

```
/* Cálculo da média e da variância
 * de 10 reais (segunda versão) */
#include <stdio.h>

/*Função para cálculo da média*/
float media (int n, float* v) {
    int i;
    float s = 0.0f;
    for(i = 0; i < n; i++)
        s += v[i];
    return s/n;
}
```

Vetores

- Passagem de vetores para funções (Exemplo – parte II):

```
/*Função para cálculo da variância*/
float variancia (int n, float* v, float m) {
    int i;
    float s = 0.0f;
    for(i = 0; i < n; i++)
        s += (v[i]-m) * (v[i]-m);
    return s/n;
}
```

Vetores

- Passagem de vetores para funções (Exemplo – parte III):

```
int main (void) {
    float v[10];
    float med, var;
    int i;
    /*leitura dos valores*/
    for(i = 0; i < 10; i++)
        scanf("%f", &v[i]);
    med = media(10, v);
    var = variancia(10, v, med);
    printf("Media = %f Variancia = %f\n", med, var);
    return 0;
}
```

Vetores

- Passagem de vetores para funções:
 - Podemos alterar os valores dos elementos do vetor dentro da função:

```
/*Incrementa elementos de um vetor*/
#include <stdio.h>
void incr_vetor (int n, int *v) {
    int i;
    for(i = 0; i < n; i++)
        v[i]++;
}

int main (void) {
    int a[] = {1, 3, 5};
    incr_vetor(3, a);
    printf("%d %d %d\n", a[0], a[1], a[2]);
    return 0;
}
```

Fonte: CELES;CERQUEIRA;RANGEL, 2004



Alocação dinâmica



Alocação dinâmica

- A linguagem C oferece meios de requisitar espaços de memória em tempo de execução.
- Exemplo da média e variância:
 - Consultar o número de alunos em tempo de execução;
 - Alocar o vetor dinamicamente.

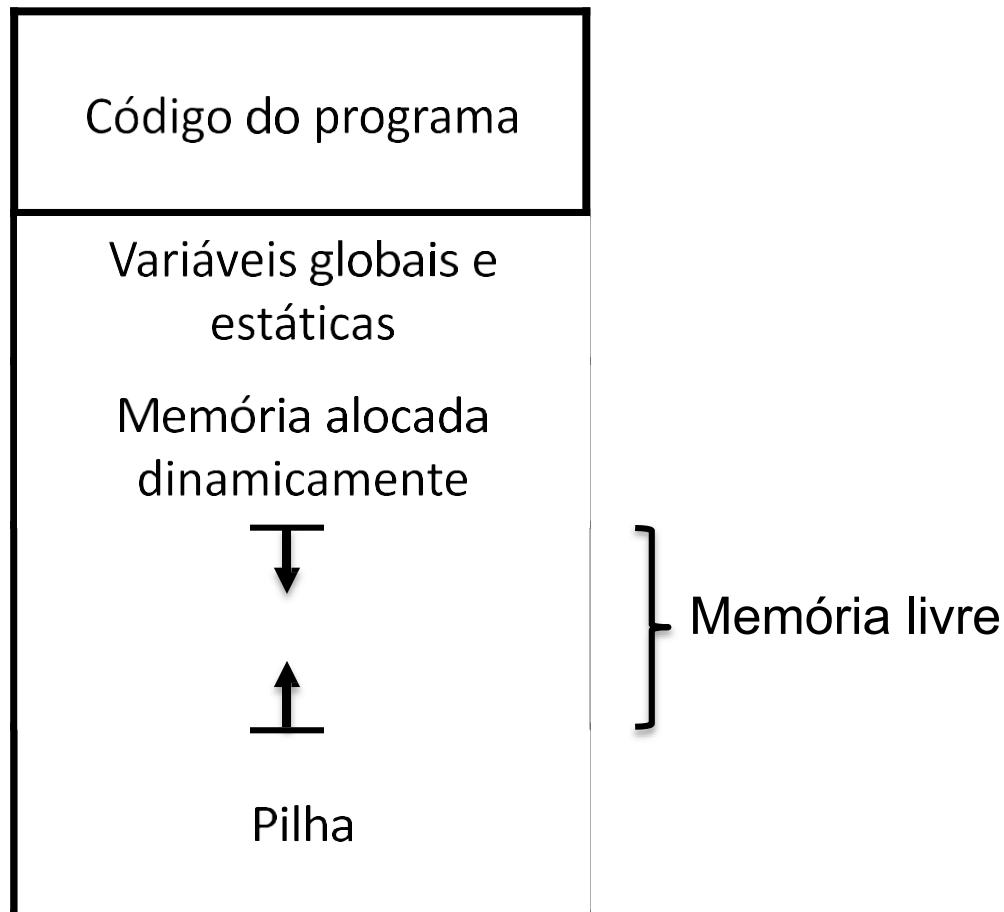


Alocação dinâmica

- Uso da memória:
 - Informalmente, há três maneiras de reservar espaço de memória:
 1. Usar variáveis globais (e estáticas);
 2. Usar variáveis locais;
 3. Requisitar ao sistema, em tempo de execução, um espaço de um determinado tamanho.
 - O espaço alocado dinamicamente permanece reservado até que seja explicitamente liberado.

Alocação dinâmica

- Uso da memória:



Fonte: CELES;CERQUEIRA;RANGEL, 2004 – Alocação esquemática de memória.

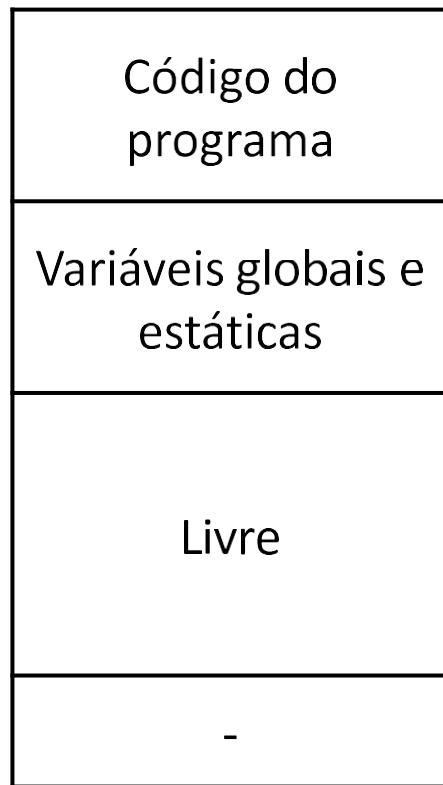
Alocação dinâmica

- Funções da biblioteca padrão (*stdlib*)
 - Função *malloc*:
 - Exemplo – alocação dinâmica de um vetor de inteiros com 10 elementos.

```
int *v;  
v = malloc(10*4);  
  
v = malloc(10*sizeof(int));  
  
v = (int*) malloc(10*sizeof(int));
```

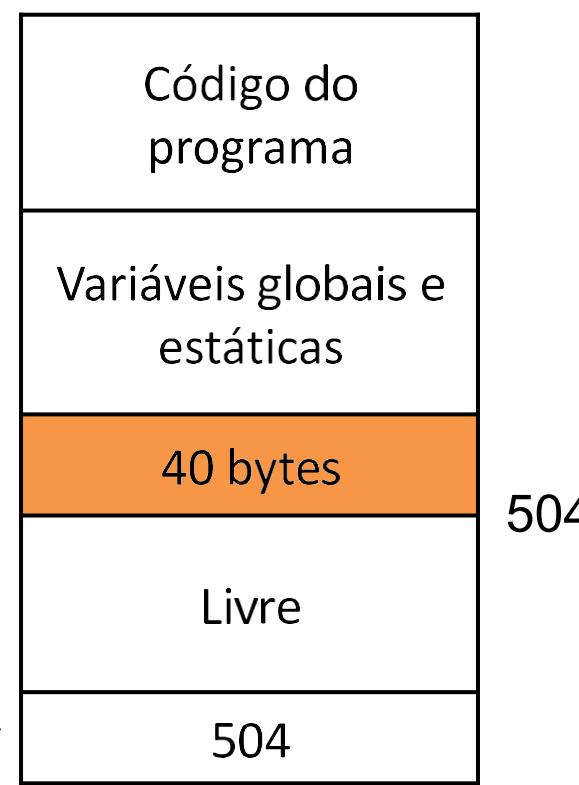
Alocação dinâmica

1 – Declaração: int *v;
Abre-se espaço na pilha para o
ponteiro (variável local)



2 – Comando:
`v = (int*) malloc(10*sizeof(int));`

Reserva espaço de memória da área livre e atribui endereço à variável.



Alocação dinâmica

- Funções da biblioteca padrão (*stdlib*)
 - Função *malloc*:
 - Se não houver espaço suficiente para a alocação, a função retorna um endereço nulo (NULL).

```
...
v = (int*) malloc(10*sizeof(int));
if(v == NULL) {
    printf("Memoria insuficiente.\n");
    exit(1); /* aborta o programa e retorna 1 para o sistema operacional */
}
...
```

Alocação dinâmica

- Funções da biblioteca padrão (*stdlib*)
 - Função *free*:
 - Libera um espaço de memória alocado dinamicamente.
 - Só podemos passar um endereço de memória alocado dinamicamente.

```
free(v);
```

Alocação dinâmica

- Funções da biblioteca padrão (*stdlib*)
 - Exercício: Alterar o programa da média e da variância, alocando o vetor dinamicamente.



Vetores locais a funções

Vetores locais a funções

- Cuidado com o uso de vetores locais dentro de funções.
- Exemplo: função que calcula o produto vetorial entre dois vetores 3D.

$$\mathbf{u} \times \mathbf{v} = \{u_y v_z - v_y u_z, u_z v_x - v_z u_x, u_x v_y - v_x u_y\}$$

Vetores locais a funções

- Função prod_vetorial (INCORRETA):

```
float* prod_vetorial(float* u, float* v) {  
    float p[3];  
    p[0] = u[1]*v[2] - v[1]*u[2];  
    p[1] = u[2]*v[0] - v[2]*u[0];  
    p[2] = u[0]*v[1] - v[0]*u[1];  
    return p; /* Erro: não podemos retornar endereço de área local */  
}
```

Vetores locais a funções

- Função prod_vetorial (CORRETA):

```
float* prod_vetorial(float* u, float* v) {
    float *p = (float*) malloc(3*sizeof(float));
    p[0] = u[1]*v[2] - v[1]*u[2];
    p[1] = u[2]*v[0] - v[2]*u[0];
    p[2] = u[0]*v[1] - v[0]*u[1];
    return p;
}
```

Vetores locais a funções

- Função prod_vetorial (CORRETA – outra solução):

```
void prod_vetorial(float* u, float* v, float* p) {  
    p[0] = u[1]*v[2] - v[1]*u[2];  
    p[1] = u[2]*v[0] - v[2]*u[0];  
    p[2] = u[0]*v[1] - v[0]*u[1];  
}
```



Referências

- CELES, W.; CERQUEIRA, R.; RANGEL, J. L. **Introdução a Estruturas de Dados: com técnicas de programação em C.** Rio de Janeiro: Ed. Campus-Elsevier, 2004.
- OLIVEIRA, U. **Programando em C – Volume I – Fundamentos.** Ciência Moderna, 2008.