ESTRUTURA DE DADOS

Curso de Licenciatura em Ciências da Computação

Lucas Sampaio Leite

Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

```c
#include <stdio.h>


int main() {

    int x, y;
    float c;
    char nome[50];

    return 0;
}
```

Alocação dinâmica de memória

- Precisamos desenvolver um programa que processe os salários de uma determinada empresa.

Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

- Precisamos desenvolver um programa que processe os salários de uma determinada empresa.
- Como ainda só sabemos trabalhar com alocação estática, uma solução simples seria armazenar os valores em um vetor de tamanho grande previamente definido.

```c
#include <stdio.h>


int main() {

    float salarios[1000];

    return 0;
}
```

Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

- Precisamos desenvolver um programa que processe os salários de uma determinada empresa.
- Como ainda só sabemos trabalhar com alocação estática, uma solução simples seria armazenar os valores em um vetor de tamanho grande previamente definido.

Declarar um vetor grande resolve o problema...
mas qual é o custo dessa decisão?

```c
#include <stdio.h>


int main() {

    float salarios[1000];

    return 0;
}
```

Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

- Sabemos que:
  - Vetores são estruturas que armazenam dados do mesmo tipo em posições sequenciais de memória.
  - Um ponteiro é uma variável que armazena o endereço de memória de outro dado.
  - O nome de um vetor pode ser utilizado como um ponteiro para o primeiro elemento do array.

Alocação dinâmica de memória

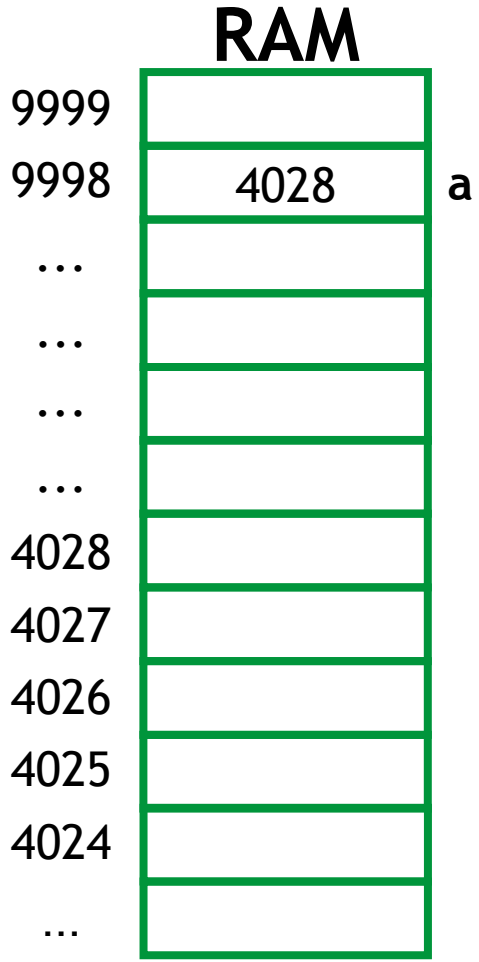
INSTITUTO FEDERAL
Baiano

- Sabemos que:
  - Vetores são estruturas que armazenam dados do mesmo tipo em posições sequenciais de memória.
  - Um ponteiro é uma variável que armazena o endereço de memória de outro dado.
  - O nome de um vetor pode ser utilizado como um ponteiro para o primeiro elemento do array.

Solução: solicitar dinamicamente um bloco de memória e guardar, em um ponteiro, o endereço da primeira posição desse bloco.

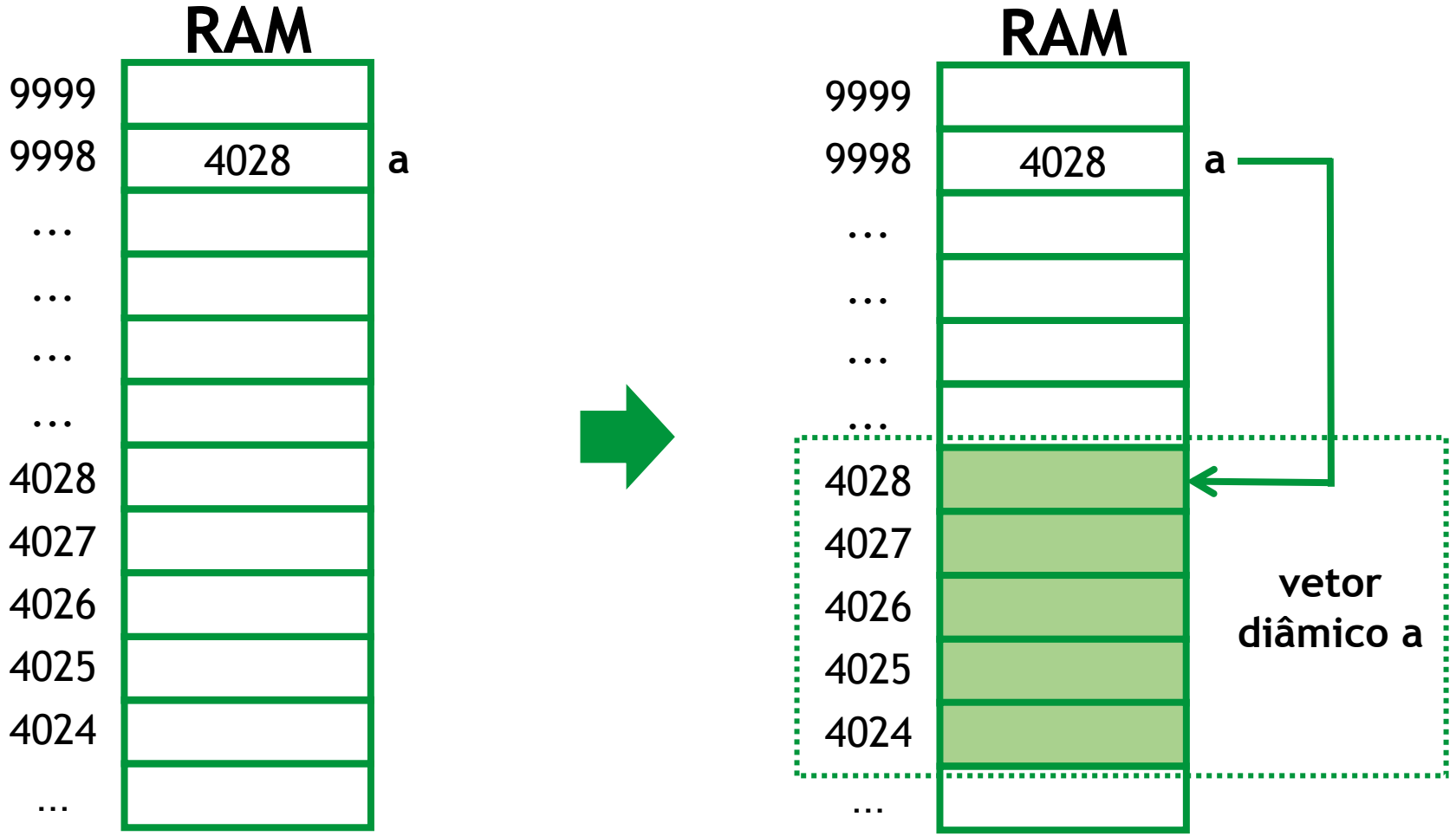
Alocação dinâmica de memória

- Solução: solicitar dinamicamente um bloco de memória e guardar, em um ponteiro, o endereço da primeira posição desse bloco.



Alocação dinâmica de memória

- Solução: solicitar dinamicamente um bloco de memória e guardar, em um ponteiro, o endereço da primeira posição desse bloco.



Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

- A linguagem C disponibiliza quatro funções para alocação dinâmica de memória, presentes na biblioteca <stdlib.h>:
  - malloc;
  - calloc;
  - realloc;
  - free.
- Além delas, utiliza-se com frequência o operador sizeof, que permite determinar a quantidade de memória necessária para um determinado tipo de dado.

Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

- Alocar memória para dados do tipo int é diferente de alocar memória para dados do tipo char.
- Isso ocorre porque tipos diferentes podem ocupar quantidades distintas de bytes na memória.

Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

- Alocar memória para dados do tipo int é diferente de alocar memória para dados do tipo char.
- Isso ocorre porque tipos diferentes podem ocupar quantidades distintas de bytes na memória.

Tipo de dado	nº de bytes
char	1 byte
int	4 bytes
float	4 bytes
double	8 bytes

Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

```c
#include <stdio.h>

struct ponto{
    int eixoX, eixoY;
};

int main() {
    printf("char: %ld bytes \n", sizeof(char));
    printf("int: %ld bytes \n", sizeof(int));
    printf("float: %ld bytes \n", sizeof(float));
    printf("double: %ld bytes \n", sizeof(double));
    printf("struct ponto: %ld bytes \n", sizeof(struct ponto));
    return 0;
}
```

Alocação dinâmica de memória

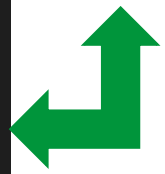
INSTITUTO FEDERAL
Baiano

```c
#include <stdio.h>

struct ponto{
    int eixoX, eixoY;
};

int main() {
    printf("char: %ld bytes \n", sizeof(char));
    printf("int: %ld bytes \n", sizeof(int));
    printf("float: %ld bytes \n", sizeof(float));
    printf("double: %ld bytes \n", sizeof(double));
    printf("struct ponto: %ld bytes \n", sizeof(struct ponto));
    return 0;
}
```

```
char: 1 bytes
int: 4 bytes
float: 4 bytes
double: 8 bytes
struct ponto: 8 bytes
```



Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

- O operador sizeof informa quantos bytes são necessários para armazenar um elemento de um determinado tipo (ou uma variável).
- Forma geral:
  - sizeof(nome_do_tipo)
  - sizeof(variável)

```c
int x = 10;
printf("valor de x: %d \n", x);
printf("memória alocada para x: %ld bytes\n", sizeof(x));
```

```
valor de x: 10
memória alocada para x: 4 bytes
```

Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

- Função malloc:
  - Pertence à biblioteca <stdlib.h>.
  - Aloca dinamicamente um bloco contíguo de memória e retorna o endereço do primeiro byte desse bloco.
  - Isso permite escrever programas mais flexíveis, pois a memória pode ser solicitada em tempo de execução.
  - Exemplo de uso: alocar um vetor cujo tamanho é definido pelo usuário.

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

Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

- A função malloc() recebe como parâmetro a quantidade de bytes a ser alocada e retorna:
  - Um ponteiro para o início do bloco de memória alocado;
  - NULL, caso ocorra falha na alocação.

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


int main() {

    int *v1 = malloc(200); //cria um vetor de 50 inteiros (200 bytes)
    char *v2 = malloc(200); //cria um vetor de 200 chars (200 bytes)

    return 0;
}
```

Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

- A função malloc() recebe como parâmetro a quantidade de bytes a ser alocada e retorna:
  - Um ponteiro para o início do bloco de memória alocado;
  - NULL, caso ocorra falha na alocação.

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


int main() {

    int *v1 = malloc(200); //cria um vetor de 50 inteiros (200 bytes)
    char *v2 = malloc(200); //cria um vetor de 200 chars (200 bytes)

    return 0;
}
```

A função malloc trabalha com bytes, não com "quantidade de elementos"

Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

- Na alocação dinâmica de memória, é necessário considerar o tamanho do tipo de dado que será armazenado.

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


int main() {

    int *v1 = malloc(50 * sizeof (int)); //cria um vetor de 50 inteiros (200 bytes)
    char *v2 = malloc(200 * sizeof(char)); //cria um vetor de 200 chars (200 bytes)

    return 0;
}
```

Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

- Se não houver memória suficiente disponível, a função malloc() retorna NULL.
- Para liberar um bloco de memória previamente alocado, utiliza-se a função free().

```c
int *vetor;
vetor = malloc (5 * sizeof(int));

if (vetor == NULL){
    printf("ERRO!!! Memória insuficiente.");
    exit(1);
}

for(int i = 0; i < 5; i++){
    printf("Digite vetor[%d]: ", i);
    scanf("%d", &vetor[i]);
}

free(vetor);
```

Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

- A função calloc() é usada para alocação dinâmica de memória em tempo de execução.
- Ela reserva espaço para múltiplos elementos e retorna um ponteiro para o início da área alocada.
- Diferentemente de malloc(), a memória alocada por calloc() é inicializada com zero.

```
int *vetor = calloc(5, sizeof(int));
```

Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

- A função calloc() recebe como parâmetros o número de elementos a serem alocados e o tamanho de cada elemento, e retorna:
  - Um ponteiro para o início do bloco de memória alocado;
  - NULL, em caso de falha na alocação.

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


int main() {

    int *v1 = calloc(50, sizeof(int));
    char *v2 = calloc(200, sizeof(char));

    return 0;
}
```

Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

- A função realloc() é utilizada para redimensionar um bloco de memória previamente alocado durante a execução do programa.
- A função realloc() recebe como parâmetros:
  - Um ponteiro para um bloco de memória já alocado (ou NULL);
  - A nova quantidade de bytes desejada.
- A função realloc() retorna:
  - Um ponteiro para o início do bloco de memória redimensionado;
  - NULL, em caso de falha na realocação.

```
int *v;
v = realloc(v, 50 * sizeof(int));
```

Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

```c
int main() {

    int *v;
    v = malloc(5 * sizeof(int));

    if(v == NULL)
        encerrar();
    v = realloc(v, 10 * sizeof(int));
    if (v == NULL)
        encerrar();
    free(v);

    return 0;
}
```

Alocação dinâmica de memória

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

void encerrar(){
    printf("ERRO!!! Memória insuficiente.");
    exit(1);
}

int main() {

    int *v;
    v = malloc(5 * sizeof(int));

    if(v == NULL)
        encerrar();
    v = realloc(v, 10 * sizeof(int));
    if (v == NULL)
        encerrar();
    free(v);

    return 0;
}
```

Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

- Ao realocar memória, não se deve atribuir o retorno de realloc() diretamente ao ponteiro original.
- Se a realocação falhar, o retorno será NULL e o endereço da memória previamente alocada será perdido, causando vazamento de memória.

```
int *v = malloc(5 * sizeof(int));

int *v1 = realloc(v, 15 * sizeof(int));
if(v1 == NULL){
    printf("ERRO!!! Memória insuficiente.");
}else{
    v = v1;
}
```

Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

- A função realloc() também pode funcionar como malloc().
- Quando o primeiro argumento passado para realloc() é NULL, ela simplesmente realiza uma nova alocação de memória com o tamanho especificado.

```
int *v;
v = realloc(NULL, 5 * sizeof(int));

if(v == NULL){
    printf("ERRO!!! Memória insuficiente.");
    exit(1);
}
```

Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

```c
int *v;
v = realloc(NULL, 5 * sizeof(int));

if(v == NULL){
    printf("ERRO!!! Memória insuficiente.");
    exit(1);
}
```

```c
int *v;
v = malloc(5 * sizeof(int));

if(v == NULL){
    printf("ERRO!!! Memória insuficiente.");
    exit(1);
}
```

Alocação dinâmica de memória

INSTITUTO FEDERAL
Baiano

- Se realloc() for chamada com tamanho igual a 0, ela pode liberar o bloco de memória apontado, funcionando de maneira semelhante à função free().

```
int *v;
v = malloc(5 * sizeof(int));

if(v == NULL){
    printf("ERRO!!! Memória insuficiente.");
    exit(1);
}

v = realloc(v, 0);
```

Exercícios

INSTITUTO FEDERAL
Baiano

1. Aloque dinamicamente memória para um vetor de 5 números inteiros. Em seguida, solicite ao usuário que digite os 5 valores e armazene-os no vetor. Exiba na tela os valores armazenados e, ao final, libere corretamente a memória alocada.

2. Leia do usuário o tamanho de um vetor de números inteiros e, em seguida, faça a alocação dinâmica de memória para armazená-lo. Depois, solicite que o usuário informe os valores do vetor, exiba todos os elementos na tela e libere a memória ao final do programa.

Exercícios

INSTITUTO FEDERAL
Baiano

3. Utilize a função calloc para alocar dinamicamente um vetor com 1500 elementos do tipo int. Em seguida, verifique por meio de um laço se todos os elementos foram inicializados com zero, contando quantos valores são iguais a zero. Depois, atribua a cada posição do vetor o valor correspondente ao seu índice, exiba na tela os 10 primeiros e os 10 últimos elementos do vetor e, ao final, libere a memória alocada.

4. Aloque dinamicamente memória para um vetor de 5 números inteiros e armazene valores informados pelo usuário. Em seguida, utilize a função realloc para aumentar o vetor, permitindo armazenar mais 5 números inteiros, e leia os novos valores. Depois, utilize novamente realloc para remover os 3 últimos elementos do vetor, exiba o conteúdo final do vetor e libere a memória ao término do programa.

Dúvidas

INSTITUTO FEDERAL
Baiano

ESTRUTURA DE DADOS

Curso de Licenciatura em Ciências da Computação

Lucas Sampaio Leite