ESTRUTURA DE DADOS

Prof.ª Priscilla Abreu

priscilla.braz@rj.senac.br

Estrutura de dados

Roteiro de Aula

- Objetivo da aula
  - Alocação Dinâmica

Estrutura de dados

Objetivo da aula

Identificar situações em que são necessárias o uso de estruturas dinâmicas e encadeadas e compreender a manipulação dessas estruturas.

Competência:

Desenvolver estruturas de dados lineares e não lineares.

“Considere um programa para cadastro de clientes que utilize um vetor de 100 posições para armazenar as informações.

Em um dado momento da execução do programa, o vetor atinge seu limite de preenchimento. O que fazer? É possível modificar o tamanho do vetor nesse momento?”.

ALOCAÇÃO ESTÁTICA
X
ALOCAÇÃO DINÂMICA

Estrutura de dados

ALOCAÇÃO ESTÁTICA

Alocação estática: o espaço de memória para as variáveis é reservado no início da execução, não podendo ser alterado depois.

```
int a; float n;
char c;
int b[20];
```

Estrutura de dados

ALOCAÇÃO ESTÁTICA

A forma mais simples de estruturarmos um conjunto de dados é por meio de vetores.
Definimos um vetor em C da seguinte forma:

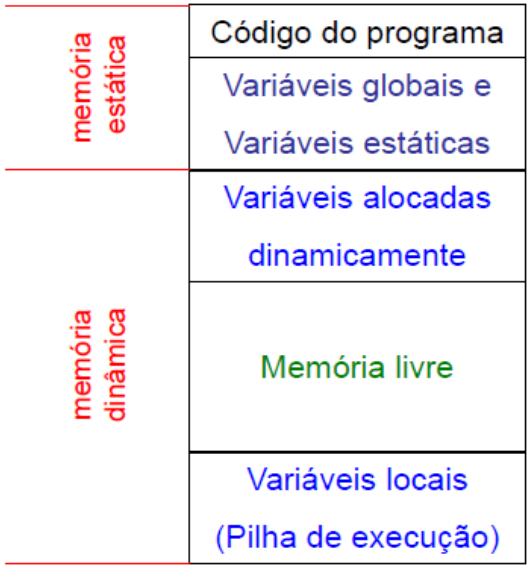
int v[10];

v é um vetor de inteiros dimensionado com 10 elementos, isto é, reservamos um espaço de memória contínuo para armazenar 10 valores inteiros.
Assim, se cada int ocupa 4 bytes, a declaração reserva um espaço de memória de 40 bytes.

Estrutura de dados

ALOCAÇÃO DE MEMÓRIA EM C

Uso da memória
(1) uso de variáveis globais.
(2) uso de variáveis locais.
(3) reserva de memória requisitando ao sistema, em tempo de execução, um espaço de um determinado tamanho.



Estrutura de dados

ALOCAÇÃO DINÂMICA

- Processo de solicitar e utilizar memória durante a **execução** de um programa.
- Visa que um programa utilize apenas a memória necessária pra sua execução, sem desperdícios.
- Deve ser utilizada quando não se sabe inicialmente quanto espaço de memória será necessário para o armazenamento de valores.
- Funções para solicitação e liberação de espaço:

  Aloca

  Libera

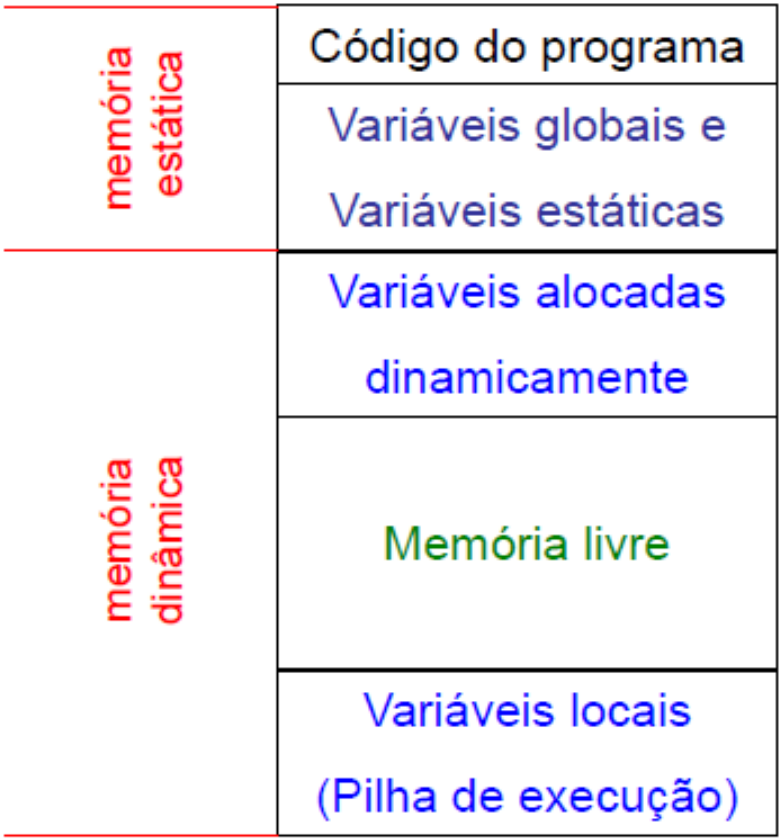
Estrutura de dados

ALOCAÇÃO DINÂMICA

- O espaço alocado dinamicamente permanece reservado até que explicitamente seja liberado pelo programa.
- A partir do momento que liberarmos o espaço, ele estará disponibilizado para outros usos e não poderemos mais acessá-lo.
- Se o programa não liberar um espaço alocado, este será automaticamente liberado quando a execução do programa terminar.

Estrutura de dados

USO DA MEMÓRIA



Estrutura de dados

ALOCAÇÃO DE MEMÓRIA EM C

Funções de alocação de memória em C:

- void * malloc(int qty_bytes_alloc);
- void * calloc(int qtd, int size);
- void * realloc(void * pointer, int new_size);
- free(void * pointer);

Estrutura de dados

ALOCAÇÃO DE MEMÓRIA EM C

Função “malloc”:

– recebe como parâmetro o número de bytes que se deseja alocar.
– retorna um ponteiro genérico para o endereço inicial da área de memória alocada, se houver espaço livre.
– retorna um endereço nulo (NULL), se não houver espaço livre.

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

Aloca um bloco de memória para 10 inteiros.

Estrutura de dados

ALOCAÇÃO DE MEMÓRIA EM C

Função “calloc”:

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

- Aloca uma quantidade de memória igual a num * size;
- Inicializa os espaços de memória com o valor zero;

Exemplo:

```
int *v;
v = (int *) calloc(10 , sizeof(int ));
```

Estrutura de dados

ALOCAÇÃO DE MEMÓRIA EM C

Função “sizeof”:

– retorna o número de bytes ocupado por um tipo

Função “free”:

– recebe como parâmetro o ponteiro da memória a ser liberada.

A função free deve receber um endereço de memória que tenha sido alocado dinamicamente.

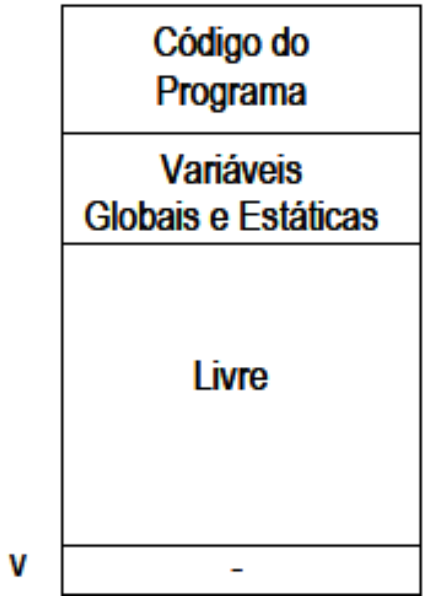
Estrutura de dados

ALOCAÇÃO DE MEMÓRIA EM C

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

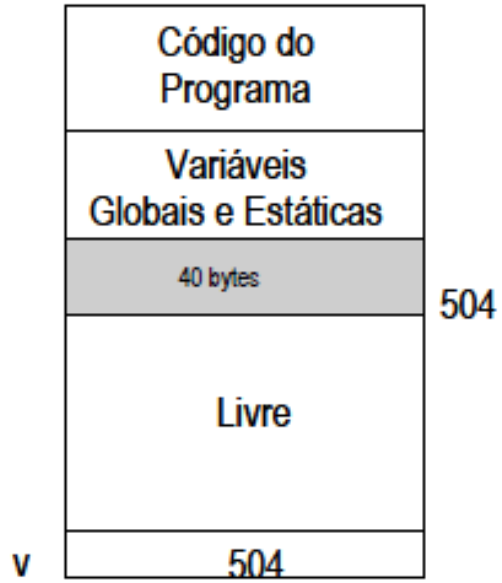
1 - Declaração: int *v

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



Estrutura de dados

ALOCAÇÃO DE MEMÓRIA EM C

- v armazena endereço inicial de uma área contínua de memória suficiente para armazenar 10 valores inteiros;

- v pode ser tratado como um vetor declarado estaticamente
  - v aponta para o inicio da área alocada
  - v[0] acessa o espaço para o primeiro elemento
  - v[1] acessa o segundo
  - .... até v[9].

Estrutura de dados

ALOCAÇÃO DE MEMÓRIA EM C

- tratamento de erro após chamada a malloc
  - imprime mensagem de erro
  - aborta o programa (com a função exit)

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

free(v);
```

Estrutura de dados

EXEMPLO - ALOCAÇÃO DE MEMÓRIA EM C

Escreva um programa em C que solicita ao usuário um número n e então lê um vetor de n notas e calcula a média aritmética.

- Usar alocação dinâmica do vetor
- Liberar a memória ao final

Estrutura de dados

EXEMPLO

```
#include <stdio.h>
int main (){
      float *v, media, soma=0.0;
      int qtde,i;
      printf("Informe quantas notas que deseja armazenar: ");
      scanf("%d", &qtde);
      v = (float*) malloc(qtde*sizeof(float));
```

Ponteiro v recebe o endereço da primeira posição do espaço alocado.

Estrutura de dados

EXEMPLO

```
    for (i=0; i<qtde;i++){
        printf("Informe a nota: ");
        scanf("%f", &v[i]);
        soma = soma + v[i];
    }
    media = soma / qtde;
    printf("Média: %.1f\n", media);
    free(v);
}
```

Libera o espaço de memória alocado em v.

E SE FOR PRECISO ACRESCENTAR MAIS NOTAS DURANTE A MESMA EXECUÇÃO?

Estrutura de dados

MODIFICANDO O ESPAÇO ALOCADO

Realloc

(tipo*) realloc(tipo *apontador, int novo_tamanho)

```
int *x, i;
x = (int *) malloc (200 * sizeof(int));
for (i = 0; i<200; i++){
        printf("Valor: "); scanf(&x[i]);
}
x = (int *) realloc (x, 400 * sizeof(int));
x = (int *) realloc (x, 100 * sizeof(int));
free(x);
```

Estrutura de dados

MODIFICANDO O ESPAÇO ALOCADO

Realloc

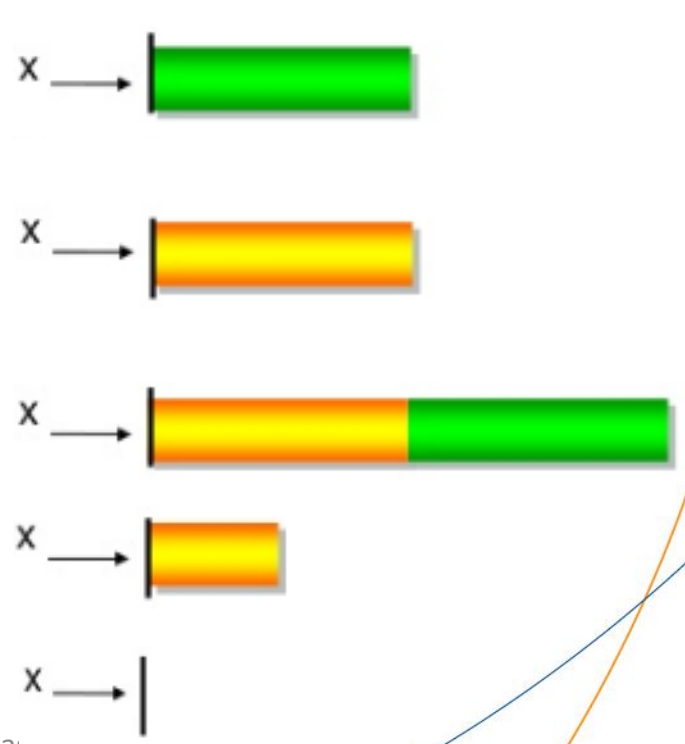
```
int *x, i;
x = (int *) malloc (200 * sizeof(int));

for (i = 0; i<200; i++){
     printf("Valor: "); scanf(&x[i]);
}

x = (int *) realloc (x, 400 * sizeof(int));

x = (int *) realloc (x, 100 * sizeof(int));

free(x);
```



Estrutura de dados

AUMENTANDO O ESPAÇO ALOCADO

```
#include <stdio.h>
int main (void){
    float *v;
    float soma=0.0, media;
    int qtde,i, qtdFinal;
    printf("Informe quantas notas deseja armazenar:");
    scanf("%d", &qtde);
    v = (float*) malloc(qtde*sizeof(float));
```

Estrutura de dados

AUMENTANDO O ESPAÇO ALOCADO

```
if (v != NULL){
   for (i=0; i<qtde;i++){
        printf("Informe a nota: ");
        scanf("%f", &v[i]);
        soma = soma + v[i];
    }
   media = soma / qtde;
   printf("Média: %.1f\n", media);
```

Estrutura de dados

AUMENTANDO O ESPAÇO ALOCADO

```
    printf(“Quantidade de notas que deseja
acrescentar - Digite 0 caso não deseje.");
    scanf("%d",&qtdFinal);
    if (qtdFinal>0){
        qtdFinal = qtdFinal + qtde;
        v = (float*) realloc(v,(qtdFinal)*sizeof(float));
```

Estrutura de dados

AUMENTANDO O ESPAÇO ALOCADO

```
            if (v!= NULL){
                for (i=qtde; i<qtdFinal; i++){
                    printf("Informe a nota: ");
                    scanf("%f", &v[i]);
                    soma=soma+v[i];
                }
                media = soma / (qtdeFinal);
                printf("Nova média: %.1f\n", media);
            }
        }
    }
}
```

Estrutura de dados

EXEMPLO ALOCAÇÃO DINÂMICA - STRUCT

```
#include <stdio.h>
struct ST_DADOS {
        char nome[40];
        float salario;
};
int main(){
        struct ST_DADOS * p;
        p = (struct ST_DADOS *)
                malloc(sizeof(struct ST_DADOS));
```

Estrutura de dados

EXEMPLO ALOCAÇÃO DINÂMICA

```
        if (p!=NULL){
                printf("\nEntre com o nome ->");
                scanf("%s", p->nome);
                printf("Entre com o salario ->");
                scanf("%f", &p->salario);
        }
        printf("\n===== Dados digitados ====");
        printf("\nNome = %s", p->nome);
        printf("\nSalario = %f", p->salario);
        free(p);
}
```

Estrutura de dados

ALOCAÇÃO DINÂMICA

Através desse modo de alocação dinâmica, conseguimos alocar espaços de memória em tempo de execução e modificar o espaço alocado, também em tempo de execução.

No entanto, todo o espaço alocado dinamicamente reservou um bloco de memória com endereços sequenciais, que foram manipulados como vetores.

DÚVIDAS?