
Universidade Federal Rural de Pernambuco
(UFRPE)
- Unidade Acadêmica de Garanhuns-

**Vetores e Alocação Dinâmica**

# Introdução a Linguagem C Parte VI

**Prof. Priscilla Kelly Machado Vieira**

**2019.1 – Aula 7**

# Apresentação do Capítulo



- Vetores
- Alocação Dinâmica
- Definição
- Malloc
- Free
- Exercícios

# Vetores



- Forma mais simples de estrutura para um conjunto de dados
  - int v[10];
  - Indexação inicia em 0 e vai até n-1

```
…
int v[10];
float med = 0.0;
for (int i = 0; i<10;i++)
   scanf (“%f”, &v[i]); //endereço de cada elemento
for (int i = 0; i<10;i++)
   med = med+v[i];
med = med/10;
```

# Vetores e Ponteiros



- Armazenados de forma continua na memória
- Inicializados na declaração
  - int v[5] = {1,2,3,4,5};
  - int v[ ] = {1,2,3,4,5};
  - v aponta para o primeiro elemento do array
    - v+0
    - v+1
    - v+2
    - …

$$\&v[i] = v+i$$

$$v[i] = *(v+i)$$

# Passagem de Vetores para Funções



- Consiste em passar o endereço da primeira posição do vetor (passagem de parâmetro por referência)
- Se passamos um endereço, o parâmetro da função deve ser um ponteiro

```
...
float media (int *v) {
int soma = 0;
for (int i = 0; i<10;i++)
   soma += v[i];
return soma/10;
}
```

```
int main (void){
int v [10] = {1,2,3,4,5,6,7,8,9,10}
media (v);
return 0;
}
```

# Alocação Dinâmica

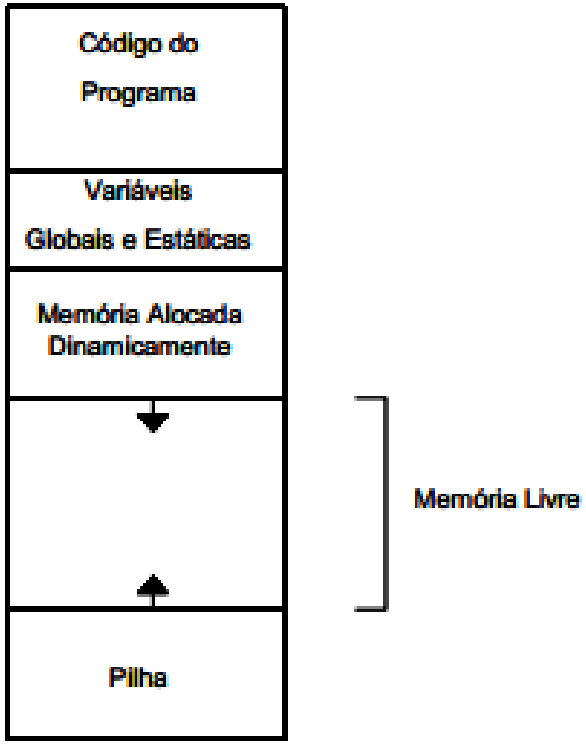


- Declarações alocam memória para diversas variáveis
- Alocação é estática, pois acontece antes que o programa comece a ser executado:
  - char c; int i; int v[10];
- Em alguns casos, a quantidade de memória a alocar só se torna conhecida durante a execução do programa.

O que fazer?

# Introdução




Código do
Programa
Variáveis
Globais e Estáticas
Memória Alocada
Dinamicamente
Memória Livre
Pilha

# Introdução



- Uso de memória
  - Dessa forma evita-se o desperdício de memória.
- Para usar esta biblioteca, é preciso:
  - #include <stdlib.h>
  - malloc
  - free

# malloc



- A função malloc (abreviação de *memory allocation*) aloca um bloco de bytes consecutivos na memória do computador e devolve o endereço desse bloco
- O número de bytes é especificado no argumento da função
- A sintaxe da função malloc() é dada por:

```
void *malloc(size_t size);
```

# malloc



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

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

n

```c
#include <stdio.h>
#include<stdlib.h> /* para utilizar o system("Pause") */

void main(void)
{
   int QuantElem, i;
   int vet[99];

   printf("Digite a quantidade de valores que deseja inserir no vetor:");
   scanf("%d",&QuantElem);

   /* Colhendo os dados a serem colocados no vetor:*/
   printf("\nDigite os valores a serem inseridos no vetor vet:\n");
   for (i=0;i<QuantElem;i++)
   {
     printf("Digite o valor de vet[%d]:",i);
     scanf("%d", &vet[i]);
   }

   /* Imprimindo os dados do vetor na tela:*/
   printf("\nImprimindo os valores do vetor vet:\n");
   for (i=0;i<QuantElem;i++)
  printf("vet[%d]:%d\n",i,vet[i]);

   system("Pause");
}
```

# mallo



```c
#include <stdio.h>
#include<stdlib.h>
#include <alloc.h> /* Para o uso da alocação dinâmica (malloc() e free()) */
#include<conio.h> /* Para o uso do getch() */

void main(void)
{
   int QuantElem, i;
   int *vet; /* uso de ponteiro */

   printf("Digite a quantidade de valores que deseja inserir no vetor:");
   scanf("%d",&QuantElem);

  /*Alocando memoria para o vetor vet do tamanho determinado pelo usuário,
  desta forma não terá diperdicio de memoria".*/
  if ((vet = (int *) malloc(QuantElem)) == NULL)
   {
     printf("Memoria insuficiente");
     exit(1);
   }

   /* Colhendo os dados a serem colocados no vetor:*/
   printf("\nDigite os valores a serem inseridos no vetor vet:\n");
   for (i=0;i<QuantElem;i++)
   {
     printf("Digite o valor de vet[%d]:",i);
     scanf("%d", &vet[i]);
   }

   /* Imprimindo os dados do vetor na tela:*/
   printf("\nImprimindo os valores do vetor vet:\n");
   for (i=0;i<QuantElem;i++)
      printf("vet[%d]:%d\n",i,vet[i]);

   free(vet);

   getch();
}
```

# Exemplo



*main.c

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

int main(){
    printf("system on!\n");
    int i,x;
    int *vetor;

    printf("Digite o tam do vetor\n");
    scanf("%d",&x); //tamanho do vetor

    vetor = malloc(x * sizeof(int));
    for ( i = 0 ; i < x ; i++ ){
        printf("Digite a info do vetor\n");
        scanf("%d",&vetor[i]);
    }
    printf("tam da memoria de int %d\n", sizeof(int));
    free(vetor); //liberando a memoria alocada.
    printf("system on!\n");
    return 0;
}
```

# Exemplo



- Verificando se a memória é suficiente

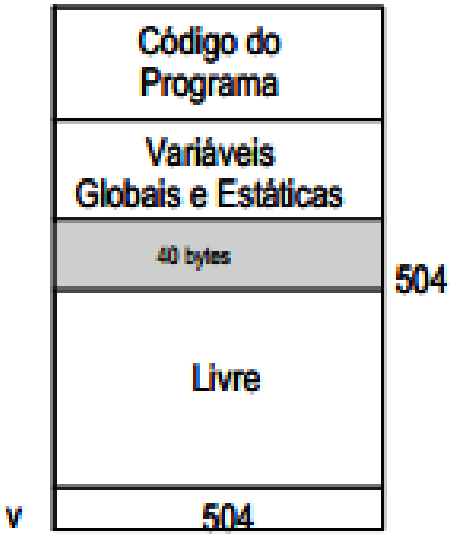
```
if ((vet = (int *) malloc(QuantElem)) == NULL)
 {
   printf("Memoria insuficiente");
   exit(1);
 }
```

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



# Exercício



- Lab 06
  - Escreva um programa que leia um número inteiro positivo n seguido de n números inteiros e imprima esses n números em ordem invertida
    - Por exemplo, ao receber 5 222 333 444 555 666
    - O seu programa deve imprimir 666 555 444 333 222
  - O seu programa não deve impor limitações sobre o valor de n.