

# LPG0001 – Linguagem de Programação

## Realocação de Memória

Prof. Rui Jorge Tramontin Junior  
Departamento de Ciência da Computação  
UDESC / Joinville

# Exemplo prático da aula passada: busca sequencial em vetor

- Dados um vetor **v**, com capacidade **n** e uma **chave** de busca;
- A função retorna um vetor (alocado dinamicamente) com os índices em que a chave se encontra (termina com -1);

```
int * busca( int v[], int n, int chave );
```

- Exemplo de entrada:

```
    v = {3, 6, 7, -1, 3, 12, 9, 8, 3, 17}  
    chave = 3
```

- Saída:

```
vetor resultante = {0, 4, 8, -1}
```

# Introdução

- A **alocação dinâmica de memória** é um recurso útil quando não se sabe previamente a quantidade de memória que será utilizada no programa;

# Introdução

- A **alocação dinâmica de memória** é um recurso útil quando não se sabe previamente a quantidade de memória que será utilizada no programa;
- A função *malloc()* aloca um novo bloco de memória sempre que é chamada;

# Introdução

- A **alocação dinâmica de memória** é um recurso útil quando não se sabe previamente a quantidade de memória que será utilizada no programa;
- A função *malloc()* aloca um novo bloco de memória sempre que é chamada;
- Todavia, às vezes é necessário aumentar (ou diminuir) uma área previamente alocada;

# Introdução

- A **alocação dinâmica de memória** é um recurso útil quando não se sabe previamente a quantidade de memória que será utilizada no programa;
- A função *malloc()* aloca um novo bloco de memória sempre que é chamada;
- Todavia, às vezes é necessário aumentar (ou diminuir) uma área previamente alocada;
  - **Realocação de memória!**

# Realocação de Memória

```
void * realloc ( void *p , size_t n_bytes )
```

# Realocação de Memória

```
void * realloc ( void *p , size_t n_bytes )
```

- **Entrada:**
  - Endereço da área já alocada (**p**);



# Realocação de Memória

```
void * realloc ( void *p , size_t n_bytes )
```

- **Entrada:**

- Endereço da área já alocada (**p**);
- quantos *bytes* a nova área realocada deve ter (**n\_bytes**);

# Realocação de Memória

```
void * realloc ( void *p , size_t n_bytes )
```

- **Entrada:**
  - Endereço da área já alocada (**p**);
  - quantos *bytes* a nova área realocada deve ter (**n\_bytes**);
- **Saída:** o endereço da área realocada;

# Realocação de Memória

```
void * realloc ( void *p , size_t n_bytes )
```

- **Entrada:**
  - Endereço da área já alocada (**p**);
  - quantos *bytes* a nova área realocada deve ter (**n\_bytes**);
- **Saída:** o endereço da área realocada;
- Caso não seja possível realizar a realocação, a função retorna a constante **NULL**.

# Realocação de Memória

- A realocação será feita de modo que o bloco de memória atual aumente cobrindo os endereços adjacentes à área já alocada;

# Realocação de Memória

- A realocação será feita de modo que o bloco de memória atual aumente cobrindo os endereços adjacentes à área já alocada;
- Caso não haja espaço suficiente “ao lado” da área já alocada, uma nova área é encontrada, de modo que realocação possa ser feita;

# Realocação de Memória

- A realocação será feita de modo que o bloco de memória atual aumente cobrindo os endereços adjacentes à área já alocada;
- Caso não haja espaço suficiente “ao lado” da área já alocada, uma nova área é encontrada, de modo que realocação possa ser feita;
  - Todos os dados ali armazenados são copiados;

# Realocação de Memória

- A realocação será feita de modo que o bloco de memória atual aumente cobrindo os endereços adjacentes à área já alocada;
- Caso não haja espaço suficiente “ao lado” da área já alocada, uma nova área é encontrada, de modo que realocação possa ser feita;
  - Todos os dados ali armazenados são copiados;
- Portanto, um bloco previamente alocado pode mudar de endereço após uma realocação.

# **EXEMPLO 1: REALOCANDO UM VETOR**



# Exemplo 1: realocando um vetor

- Neste exemplo, um vetor com capacidade  $n$  é alocado;

# Exemplo 1: realocando um vetor

- Neste exemplo, um vetor com capacidade  $n$  é alocado;
- Após a entrada e saída de dados, o usuário informa quantos valores a mais ele deseja ( $m$ );

# Exemplo 1: realocando um vetor

- Neste exemplo, um vetor com capacidade  $n$  é alocado;
- Após a entrada e saída de dados, o usuário informa quantos valores a mais ele deseja ( $m$ );
- O vetor então é realocado com capacidade  $n + m$ ;

# Exemplo 1: realocando um vetor

- Neste exemplo, um vetor com capacidade  $n$  é alocado;
- Após a entrada e saída de dados, o usuário informa quantos valores a mais ele deseja ( $m$ );
- O vetor então é realocado com capacidade  $n + m$ ;
- É feita a entrada na parte realocada, e a saída de dados;

# Exemplo 1: realocando um vetor

- Neste exemplo, um vetor com capacidade  $n$  é alocado;
- Após a entrada e saída de dados, o usuário informa quantos valores a mais ele deseja ( $m$ );
- O vetor então é realocado com capacidade  $n + m$ ;
- É feita a entrada na parte realocada, e a saída de dados;
- Finalmente, a memória alocada é liberada.

# Exemplo 1: realocando um vetor

```
int *p, n, m, i;
```

## Modelo da Memória

p		1000
n		1008
m		1012
i		1016
	•	
	•	
	•	
	•	

# Exemplo 1: realocando um vetor

```
int *p, n, m, i;
```

```
printf("Quantos valores? ");
```

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

## Modelo da Memória

p		1000
n		1008
m		1012
i		1016
	•	
	•	
	•	
	•	

# Exemplo 1: realocando um vetor

```
int *p, n, m, i;  
  
printf("Quantos valores? ");  
scanf("%d", &n); // usuário digitou 3
```

## Modelo da Memória

p		1000
n	3	1008
m		1012
i		1016
	•	
	•	
	•	
	•	



# Exemplo 1: realocando um vetor

```
int *p, n, m, i;  
  
printf("Quantos valores? ");  
scanf("%d", &n);  
  
p = malloc( sizeof(int) * n );
```

## Modelo da Memória

p		1000
n	3	1008
m		1012
i		1016
	•	
	•	
	•	
	•	

# Exemplo 1: realocando um vetor

```
int *p, n, m, i;
```

```
printf("Quantos valores? ");  
scanf("%d", &n);
```

```
p = malloc( sizeof(int) * n );
```

```
/* Aloca vetor com capacidade 3...  
   ... área ocupa 12 bytes  
*/
```

## Modelo da Memória

p	5000	1000
n	3	1008
m		1012
i		1016
	.	
	.	
	.	
	.	
p[0]		5000
p[1]		5004
p[2]		5008

# Exemplo 1: realocando um vetor

```
int *p, n, m, i;

printf("Quantos valores? ");
scanf("%d", &n);

p = malloc( sizeof(int) * n );

for( i = 0 ; i < n ; i++ )
    scanf("%d", p + i);
```

## Modelo da Memória

p	5000	1000
n	3	1008
m		1012
i		1016
	.	
	.	
	.	
	.	
p[0]		5000
p[1]		5004
p[2]		5008

# Exemplo 1: realocando um vetor

```
int *p, n, m, i;

printf("Quantos valores? ");
scanf("%d", &n);

p = malloc( sizeof(int) * n );
```

```
for( i = 0 ; i < n ; i++ )
    scanf("%d", p + i);
```

```
// Entrada de dados
```

## Modelo da Memória

p	5000	1000
n	3	1008
m		1012
i		1016
	.	
	.	
	.	
	.	
p[0]	3	5000
p[1]	6	5004
p[2]	9	5008

# Exemplo 1: realocando um vetor

```
int *p, n, m, i;

printf("Quantos valores? ");
scanf("%d", &n);

p = malloc( sizeof(int) * n );

for( i = 0 ; i < n ; i++ )
    scanf("%d", p + i);

for( i = 0 ; i < n ; i++ )
    printf("P[%d] : %d\n", i, p[i] );
```

## Modelo da Memória

p	5000	1000
n	3	1008
m		1012
i		1016
	.	
	.	
	.	
	.	
p[0]	3	5000
p[1]	6	5004
p[2]	9	5008

# Exemplo 1: realocando um vetor

```
int *p, n, m, i;

printf("Quantos valores? ");
scanf("%d", &n);

p = malloc( sizeof(int) * n );

for( i = 0 ; i < n ; i++ )
    scanf("%d", p + i);

for( i = 0 ; i < n ; i++ )
    printf("P[%d] : %d\n", i, p[i] );

// Imprime dados do vetor
```

## Modelo da Memória

p	5000	1000
n	3	1008
m		1012
i		1016
	.	
	.	
	.	
	.	
p[0]	3	5000
p[1]	6	5004
p[2]	9	5008

# Exemplo 1: realocando um vetor

```
int *p, n, m, i;

printf("Quantos valores? ");
scanf("%d", &n);

p = malloc( sizeof(int) * n );

for( i = 0 ; i < n ; i++ )
    scanf("%d", p + i);

for( i = 0 ; i < n ; i++ )
    printf("P[%d] : %d\n", i, p[i] );

// continua...
```

## Modelo da Memória

p	5000	1000
n	3	1008
m		1012
i		1016
	.	
	.	
	.	
	.	
p[0]	3	5000
p[1]	6	5004
p[2]	9	5008

# Exemplo 1: realocando um vetor

```
printf("Quantos valores a mais? ");  
scanf("%d", &m);
```

## Modelo da Memória

p	5000	1000
n	3	1008
m		1012
i		1016
	.	
	.	
	.	
	.	
p[0]	3	5000
p[1]	6	5004
p[2]	9	5008



# Exemplo 1: realocando um vetor

```
printf("Quantos valores a mais? ");  
scanf("%d", &m); // usuário digitou 2
```

## Modelo da Memória

p	5000	1000
n	3	1008
m	2	1012
i		1016
	•	
	•	
	•	
	•	
p[0]	3	5000
p[1]	6	5004
p[2]	9	5008

# Exemplo 1: realocando um vetor

```
printf("Quantos valores a mais? ");  
scanf("%d", &m);  
  
p = realloc( p , sizeof(int) * (n + m) );
```

## Modelo da Memória

p	5000	1000
n	3	1008
m	2	1012
i		1016
	.	
	.	
	.	
	.	
p[0]	3	5000
p[1]	6	5004
p[2]	9	5008

# Exemplo 1: realocando um vetor

```
printf("Quantos valores a mais? ");  
scanf("%d", &m);
```

```
p = realloc( p , sizeof(int) * (n + m) );
```

```
// Realocando área: n + m = 5: 20 bytes.
```

## Modelo da Memória

p	5000	1000
n	3	1008
m	2	1012
i		1016
	.	
	.	
	.	
	.	
p[0]	3	5000
p[1]	6	5004
p[2]	9	5008
p[3]		5012
p[4]		5016

# Exemplo 1: realocando um vetor

```
printf("Quantos valores a mais? ");  
scanf("%d", &m);  
  
p = realloc( p , sizeof(int) * (n + m) );  
  
for( i = n ; i < n + m ; i++ )  
    scanf("%d", p + i);
```

## Modelo da Memória

p	5000	1000
n	3	1008
m	2	1012
i		1016
	•	
	•	
	•	
	•	
p[0]	3	5000
p[1]	6	5004
p[2]	9	5008
p[3]	12	5012
p[4]	15	5016

# Exemplo 1: realocando um vetor

```
printf("Quantos valores a mais? ");  
scanf("%d", &m);  
  
p = realloc( p , sizeof(int) * (n + m) );
```

```
for( i = n ; i < n + m ; i++ )  
    scanf("%d", p + i);
```

```
// Entrada de dados...  
// ... somente na parte realocada.
```

## Modelo da Memória

p	5000	1000
n	3	1008
m	2	1012
i		1016
	·	
	·	
	·	
	·	
p[0]	3	5000
p[1]	6	5004
p[2]	9	5008
p[3]	12	5012
p[4]	15	5016

# Exemplo 1: realocando um vetor

```
printf("Quantos valores a mais? ");  
scanf("%d", &m);  
  
p = realloc( p , sizeof(int) * (n + m) );  
  
for( i = n ; i < n + m ; i++ )  
    scanf("%d", p + i);  
  
for( i = 0 ; i < n + m ; i++ )  
    printf("P[%d] : %d\n", i, p[i] );
```

## Modelo da Memória

p	5000	1000
n	3	1008
m	2	1012
i		1016
	.	
	.	
	.	
	.	
p[0]	3	5000
p[1]	6	5004
p[2]	9	5008
p[3]	12	5012
p[4]	15	5016

# Exemplo 1: realocando um vetor

```
printf("Quantos valores a mais? ");
scanf("%d", &m);

p = realloc( p , sizeof(int) * (n + m) );

for( i = n ; i < n + m ; i++ )
    scanf("%d", p + i);

for( i = 0 ; i < n + m ; i++ )
    printf("P[%d] : %d\n", i, p[i] );

// Imprime dados do vetor todo
```

## Modelo da Memória

p	5000	1000
n	3	1008
m	2	1012
i		1016
	.	
	.	
	.	
	.	
p[0]	3	5000
p[1]	6	5004
p[2]	9	5008
p[3]	12	5012
p[4]	15	5016

# Exemplo 1: realocando um vetor

```
printf("Quantos valores a mais? ");  
scanf("%d", &m);  
  
p = realloc( p , sizeof(int) * (n + m) );  
  
for( i = n ; i < n + m ; i++ )  
    scanf("%d", p + i);  
  
for( i = 0 ; i < n + m ; i++ )  
    printf("P[%d] : %d\n", i, p[i] );  
  
free(p); // libera a memória
```

## Modelo da Memória

p	5000	1000
n	3	1008
m	2	1012
i		1016
	•	
	•	
	•	
	•	
p[0]	3	5000
p[1]	6	5004
p[2]	9	5008
p[3]	12	5012
p[4]	15	5016



# Exemplo 1: realocando um vetor

```
printf("Quantos valores a mais? ");  
scanf("%d", &m);  
  
p = realloc( p , sizeof(int) * (n + m) );  
  
for( i = n ; i < n + m ; i++ )  
    scanf("%d", p + i);  
  
for( i = 0 ; i < n + m ; i++ )  
    printf("P[%d] : %d\n", i, p[i] );  
  
free(p); // libera a memória
```

## Modelo da Memória

p	5000	1000
n	3	1008
m	2	1012
i		1016
	.	
	.	
	.	
	.	
p[0]	3	5000
p[1]	6	5004
p[2]	9	5008
p[3]	12	5012
p[4]	15	5016

# **EXEMPLO PRÁTICO 1: BUSCA SEQUENCIAL EM VETOR**

# Exemplo prático 2:

## busca sequencial em vetor

- Dados um vetor **v**, com capacidade **n** e uma **chave** de busca;
- A função retorna um vetor (alocado dinamicamente) com os índices em que a chave se encontra (termina com -1);

```
int * busca( int v[], int n, int chave );
```

- Exemplo de entrada:

```
v = {3, 6, 7, -1, 3, 12, 9, 8, 3, 17}  
chave = 3
```

- Saída:

```
vetor resultante = {0, 4, 8, -1}
```

# **EXEMPLO PRÁTICO 2: INTERSECÇÃO**

# Exemplo Prático 2: Intersecção

- Dados dois vetores  $v1$  e  $v2$  (e suas capacidades), a função retorna:
  - O endereço de um vetor (alocado dinamicamente, contendo a intersecção entre  $v1$  e  $v2$ ;
  - A capacidade do novo vetor (parâmetro por referência);
- Protótipo da função:

```
int *intersecao(int *v1, int n1, int *v2, int n2, int *p3)
```

# Exemplo Prático 2: Intersecção

*main()*

```
int a[] = { 1, 2, 3, 4, 5 };
int b[] = { 3, 4, 5, 6, 7, 8 };

int n_a = sizeof( a ) / sizeof( int );
int n_b = sizeof( b ) / sizeof( int );
int n_c, i;

int *c = interseccao( a, n_a, b, n_b, &n_c );

for( i = 0 ; i < n_c ; i++ ){
    printf("%d : %d\n", i , c[i] );
}

free( c );
```

# Exemplo Prático 2: Intersecção

## *função*

```
int *interseccao(int *v1, int n1, int *v2, int n2, int *p3){
    int *p = NULL; // Ponteiro NULL pode ser realocado!
    int i, j;
    *p3 = 0; // n_c = 0;
    for( i = 0 ; i < n1 ; i++ )
        for( j = 0 ; j < n2 ; j++ )
            if( v1[i] == v2[j] ){
                (*p3)++; // n_c++;
                p = realloc( p, sizeof(int) * *p3 );
                p[*p3 - 1] = v1[i];
            }

    return p;
}
```

# Exercício

- Reescreva a função `interseccao()`, mas desta vez sem usar a realocação;
- A função deve percorrer *v1* e *v2* e contar quantos valores são iguais;
- Em seguida, deve alocar o vetor resultante, e percorrer *v1* e *v2* novamente para copiar os valores.