INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS – CAMPUS GOIÂNIA – GO.
ENGENHARIA DE CONTROLE E AUTOMAÇÃO
ESTRUTURAS DE DADOS

Prof. MSc. Raphael Gomes (raphael@ifg.edu.br)

Roteiro de aula – Ponteiros em C

- **Seção 1.1.3 (LAUREANO, 2008):**
  - **Definição de Ponteiro**
  - **Operador & e ***

Exemplo:

```
#include <stdio.h>

int main() {
      int *piValor; /* ponteiro para inteiro */
      int iVariavel = 27121975;

      piValor = &iVariavel; /* pegando o endereço de memória da variável */

      printf("Endereco: %d\n", piValor);
      printf ("Valor : %d\n", *piValor);
      *piValor = 18011982;
      printf ("Valor alterado: %d\n", iVariavel);
      printf ("Endereco : %d\n", piValor);

      return 0;
}
```

O que será impresso?

```
int x;
int *i;
x = 23;
i = &x;
*i = 19;
printf("x =  %d",x);
```

```
int x;
int *p1, *p2;
x = 23;
p1 = &x;
p2 = p1;
printf("p2:  %p \n",p2);
printf("p1:  %p \n",p1);
printf("&x:  %p \n",&x);
```

```
int x;
int *i;
x = 23;
*i = x;
printf("i =  %d",*i);
```

```
int x;
int *p1, *p2;
x = 23;
p1 = &x;
p2 = p1;
x = 15;
printf("p2:  %p \n",p2);
printf("p1:  %p \n",p1);
printf("&x:  %p \n",&x);
```

  - **Aritmética com Ponteiros**

```
#include <stdio.h>

int main() {
      int *piValor;
      int iValor;
      char *pcValor;
      char cValor;
```

```
	piValor = &iValor;
	pcValor = &cValor;
	printf("Endereco de piValor = %d\n", piValor);
	printf("Endereco de pcValor = %d\n", pcValor);

	piValor++; /* somando uma unidade (4 bytes) na memória */
	pcValor++; /* somando uma unidade (1 byte) na memória */

	printf("Endereco de piValor = %d\n", piValor);
	printf("Endereco de pcValor = %d\n", pcValor);

	return 0;
}
```

- **Capítulo 2 (LAUREANO, 2008)**
  - **Alocação de memória Estática x Dinâmica**
  - **Função malloc**

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

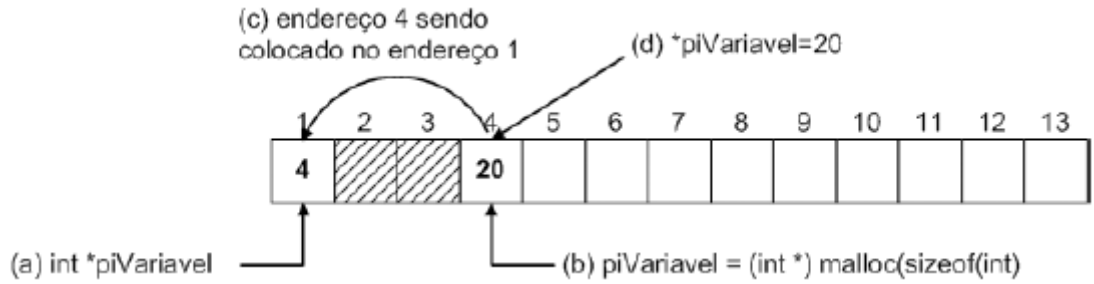
  - **Função calloc**

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

  - **Função realloc**

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

int main() {
	int *p;
	int i,k, n;
	printf("Digite a quantidade de numeros que serao digitados ->");
	scanf("%d", &i);

	/* A função malloc reserva espaço suficiente para um vetor de inteiros.
	Caso sejam digitados 5 elementos, serão reservados 20 bytes, pois cada
	inteiro ocupa 4 bytes na memória */
	p = (int *)(malloc(i*sizeof(int)));
	if (p == NULL) {
		printf("\nErro de alocacao de memoria");
		exit(1);
	}

	printf("\nDigite quantos elementos quer adicionar ao vetor ->");
	scanf("%d", &n);

	/* A função realloc aumenta ou diminui o tamanho do vetor dinamicamente.
	Ela recebe o ponteiro para o vetor anterior e retorna o novo espaço
	alocado. */
	p = (int*)(realloc(p, (i+n)*sizeof(int)));
	if(p == NULL) {
		printf("\nErro de re-alocacao de memoria");
		exit(1);
	}

	free(p);

	return 0;
}
```

- **Função free**
- **Ponteiros para ponteiros**



Operação	**Resultado**
printf("%d\n",&piVariavel)	1
printf("%d\n",piVariavel)	4
printf("%d\n",*piVariavel)	20

Referências

- LAUREANO, Marcos., Estrutura de Dados com Algoritmos e C . Rio de Janeiro: Brasport Livros e Multimídia Ltda , 2008.