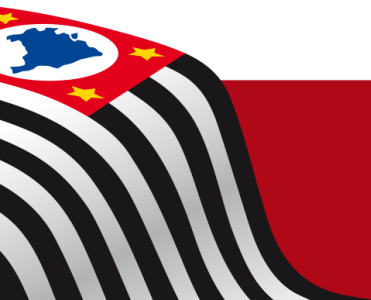
CENTRO PAULA SOUZA

Curso: ADS
Estrutura de Dados
Ponteiros

Profº Msc. Anderson L. Coan
anderson.coan@fatec.sp.gov.br



Centro Paula Souza

GOVERNO DO ESTADO SÃO PAULO

Alocação de Memória

- Da área de memória que é reservada ao programa, uma parte é usada para armazenar as instruções a serem executadas e a outra é destina ao armazenamento de dados;

- Quem determina quanto de memória será usado para as instruções é o compilador;

- Alocar área para armazenamento de dados, entretanto, é responsabilidade do programador;

Alocação de Memória

- Quando a quantidade de memória utilizada pelos dados é previamente conhecida e definida no próprio código-fonte do programa, trata-se de alocação estática;

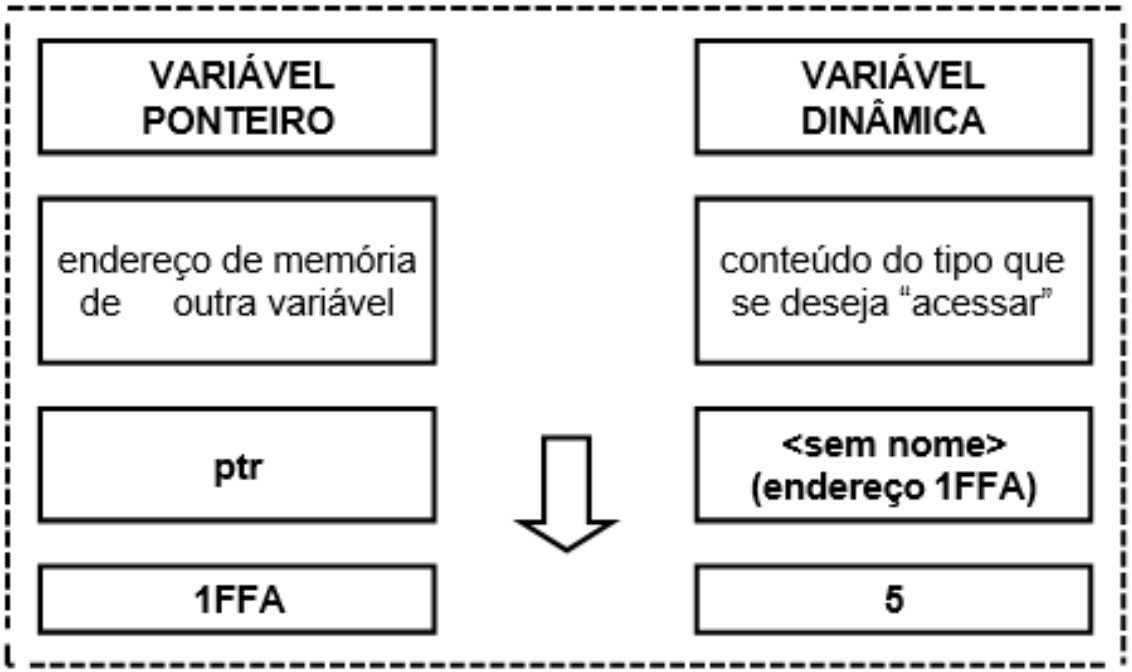
- Quando o programa é capaz de criar novas variáveis durante sua execução, dizemos que a alocação é dinâmica.

Ponteiros

- **PONTEIRO é ...**

- uma variável especial que contém o endereço de memória de outra variável;

- variável que contém endereços de posições de memória alocadas na memória.



Ponteiros – Comandos básicos

< tipo de dado > *< identificador > ;

- Declara uma variável do tipo ponteiro;

- Pode haver um ponteiro (ou apontador) para qualquer tipo de variável.

(void*) malloc(tamanho em bytes);

- Aloca dinamicamente, durante a execução do programa, células de memória;

- Esta função devolve um ponteiro void que deve ser ser convertido para o tipo desejado;

- O tamanho em bytes pode ser determinado utilizando a função sizeof() que retorna o tamanho em bytes que um tipo de dados ocupa.

Ponteiros – Comandos básicos

Exemplo:

```
int *ptr;
ptr = ( int * ) malloc ( sizeof ( int ) ) ;
```

free (ponteiro)

- Libera as células de memória alocadas dinamicamente.

Exemplo:

```
free ( ptr );
```

Para manipulação de valores utilizando ponteiros temos dois operadores:

- O operador * devolve o valor contido no endereço apontado pela variável ponteiro;

- O operador & devolve o endereço de memória alocado de uma variável.

Exemplo

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

int main ( ) {
        int *x, *y;
        x = ( int * ) malloc ( sizeof ( int ) ) ;
        y = ( int * ) malloc ( sizeof ( int ) ) ;
        *x = 27 ;
        *y = 43 ;
        printf ( "Valor: %d - %d \n\n", *x , *y ) ;
        *x = *y ;
        printf ( "Valor: %d - %d \n\n", *x , *y ) ;
        *x = 27 ; y = x ;
        printf ( "Valor: %d - %d \n\n", *x , *y ) ;
        free ( x ) ;
        system ( "pause" ) ;
        return 0 ;
}
```

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

int main ( ) {
        int *x, *y;
        int a, b;
        a = 27;
        b = 43;
        x = &a;
        y = &b;
        printf ( "Valor: %d - %d \n\n", *x , *y );
        *x = *y;
        printf ( "Valor: %d - %d \n\n", *x , *y );
        *x = 27;
        y = x;
        printf ( "Valor: %d - %d \n\n", *x , *y );
        system ( "pause");
        return 0;
}
```

Exemplo

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

int main ( ) {
        char *x, *y;
        x = ( char * ) malloc (sizeof ( char ) );
        y  =  (  char  *  )  malloc  (sizeof  (  char  ));
        strcpy ( x, "Estruturas" );
        strcpy ( y, "de Dados" );
        printf ( "%s %s\n\n", x, y );
        strcpy ( y, x );
        printf ( "%s %s\n\n", x, y );
        strcpy ( y, "de Dados" );
        printf ( "%s %s\n\n", x, y );
        x = y;
        printf ( "%s %s\n\n", x, y ) ;
        free( x );
        system ( "pause" );
        return 0;
}
```

DÚVIDAS

Exercícios - 1

1. Faça um programa que modifique as vogais de uma frase. O programa deve ler uma frase (max. 100 caracteres) e armazeá-la num vetor. Imprimir a frase lida trocando as vogais, isto é, trocar 'a' pelo 'u', 'e' pelo 'o', 'i' pelo 'u', 'o' pelo 'a' e o 'u' pelo 'e'. Usar uma função void (procedimento) para realizar a troca e uma função para realizar a impressão da frase trocada. A função deve ter como parâmetro um ponteiro char referente ao vetor. Dica: Use a função gets() da biblioteca string.h para realizar a leitura da frase. use o switch para realizar as trocas. Só considere as letras minúsculas.

Exercícios – 1 Resolução

```
#include<stdio.h>
#include<string.h>
void troca(char *vet) {
int i, tam;

tam = strlen(vet);
for (i=0; i < tam; i++) {
 switch(*vet) {
   case 'a':
   *vet = 'u';
    break;
   case 'e':
    *vet='o';
     break;
   case 'i':
     *vet='u';
     break;
   case 'o':
     *vet='a';
     break;
   case 'u':
     *vet='e';
     break;
}
vet++;
}

}
void imprime(char *vet) {
int i;
char *ptr;
ptr = vet;
printf("\n\n");
for (i=0; i < strlen(vet); i++) {
 printf("%c", *ptr);
 ptr++;
}
}
int main(){
char vet[100];

printf("\n\nDigite uma frase: ");
gets(vet);
troca(vet);
imprime(vet);
}
```

Exercícios - 2

1. Escreva um programa que declare uma matriz 10x10 de inteiros. Você criar uma função void (procedimento) para inicializar a matriz com zeros usando um ponteiro para a matriz. Faça outra função void para preencher depois a matriz com os números de 99 a 0, também usando ponteiro para matriz como parâmetro. Por fim, o programa deve imprimir a matriz.

Exercícios – 2 Resolução

```
#include<stdio.h>
#include<string.h>
void inicializa(int *mat) {
int i;

for(i=0; i < 100; i++) {
  *mat=0;
  mat++;
}
}
void preenche(int *mat) {
int i;

for(i=0; i < 100; i++) {
  *mat=99 - i;
  mat++;
}
}
int main() {
int matriz[10][10];
int i, j;

inicializa(*matriz);
preenche(*matriz);
for(i=0; i< 10; i++){
 printf("\n");
 for (j=0; j < 10; j++)
   printf("mat[%d][%d]= %d ", i,j,matriz[i][j]);
}
}
```

Exercícios - 3

1. Codifique um programa que contenha uma variável ponteiro que aponte para um vetor de 5 elementos do tipo inteiro, insira dados neste vetor e mostre os mesmos.

2. Codifique um programa que contenha um vetor de 5 elementos do tipo ponteiro para um char, insira dados neste vetor e mostre os mesmos.

3. O que será impresso no seguinte programa:

Exercícios - 3

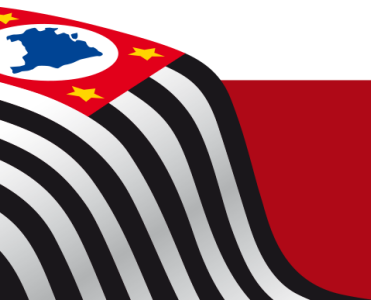
```
# include <stdio.h>
# include <stdlib.h>
# include <string.h>

int main ( )
{
        int *p1, *p2, p3 ;
        char *x1, *x2, x3 [ 7 ] ;
        p1 = ( int * ) malloc ( sizeof ( int ) ) ;
        p2 = ( int * ) malloc ( sizeof ( int ) ) ;
        x1 = ( char * ) malloc ( 7 * sizeof ( char ) ) ;
        x2 = ( char * ) malloc ( 7 * sizeof ( char ) ) ;
        *p1 = 7 ;
        strcpy ( x1,"teste" ) ;
        *p2 = 15 ;
        strcpy ( x2, "aula" ) ; p3 = 11 ;
        strcpy ( x3, "prova" ) ; p1 = p2 ;
        p2 = &p3 ; strcpy ( x2, x3 ) ; strcpy ( x1, x3 ) ;
        printf ( "%d \n", *p1 ) ;
        printf ( "%d \n", *p2 ) ;
        printf ( "%d \n", p3 ) ;
        printf ( "%s \n", x1 ) ;
        printf ( "%s \n", x2 ) ;
        printf ( "%s \n", x3 ) ;
        system ( "pause" ) ; return 0 ;
}
```

CENTRO PAULA SOUZA

Curso: ADS
Estrutura de Dados
Ponteiros

Profº Msc. Anderson L. Coan
anderson.coan@fatec.sp.gov.br



Centro Paula Souza

GOVERNO DO ESTADO SÃO PAULO