Ponteiros

Declaração:

<tipo> *<nome_da_variavel ponteiro>;

operador especial, para indicar que a variável armazena um endereço de memória.

qualquer tipo válido em C

Exemplos:

```
float *f; // f é um ponteiro para variáveis do tipo float
int *p;  // p é um ponteiro para variáveis do tipo inteiro
```


Operadores de Ponteiros

& operador unário, que devolve o endereço de memória de seu operando

***** operador unário que devolve o valor da variável localizada no endereço que o segue

Exemplo:

```
int count, q, *m;
m = &count;  // m recebe  endereço de memória da variável count
q = *m;  // q recebe o valor contido no endereço apontado por m
```


Aritmética de Ponteiros

- Adição e Subtração de ponteiros
  - Podemos somar ou subtrair inteiros de ponteiros.
  - O valor do ponteiro irá aumentar ou diminuir, dependendo do número de bytes que o tipo base ocupa, isto é, o endereço de memória apontado será deslocado em tantos bytes quanto forem os reservados para o tipo associado ao ponteiro.
  - Supondo um ponteiro inteiro p1, apontando para o endereço de memória 2000:
    - ✓p1++ valor de p1 fica 2004. *
    - ✓p1-- valor de p1 fica 1996.
    - ✓p1= p1 + 5 p1 passa a apontar para 2000 + (5 X 4)

* Cada inteiro usa 4 bytes.

Semelhanças entre Ponteiros e Vetores

- Vetores são ponteiros: o nome do vetor corresponde ao ponteiro para o primeiro elemento.
- Todas as operações válidas para ponteiros podem ser executadas com vetores.
- Assim, as colunas abaixo são equivalentes:

com vetores	equivale a	com ponteiros
int v[100];		int *v;
v[i]		*(v+i)
&v[i]		v+i

declaração de vetor de inteiros de 100 posições indica que v é um ponteiro que aponta para o 1º elemento do vetor (v[0])

Diferença entre Ponteiros e Vetores

- Declaração de vetor:
  - o compilador automaticamente reserva um bloco de memória para que o vetor seja armazenado, do tamanho especificado na declaração (mas não controla se o uso é limitado à área reservada).
- Declaração de ponteiro:
  - o compilador aloca um ponteiro para apontar para a memória, sem reservar espaço de memória (usado para alocação dinâmica de memória).

Ponteiros e *Strings*

- *Strings* são vetores de caracteres.
- Assim:

```
char str[80], *p1;
p1 = str; //atribui à p1 o end. do 1º. elemento de str
// str[4] ou *(p1+4) acessam o 5º caractere (elemento) de str
```

Inicialização de Ponteiros

Ponteiro não inicializado:

- Após ser declarado, e antes de receber um valor, o valor inicial de um ponteiro é desconhecido.
- Para que um ponteiro não aponte para um endereço de memória desconhecido, deve receber o valor nulo (NULL).

Problemas com Ponteiros

Ponteiro não inicializado:

- Após ser declarado, e antes de receber um valor, o valor inicial de um ponteiro é desconhecido.
- Para que um ponteiro não aponte para um endereço de memória desconhecido, deve receber o valor nulo (NULL).

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  int x, *p;

  x = 10;
  *p = x;

  system("PAUSE");
  return 0;
}
```

Problema:
O ponteiro p nunca recebeu valor, por isso contém lixo. O programa atribui o valor 10 a uma posição de memória desconhecida.

Solução:
Certifique-se de que o ponteiro está apontando para algo válido antes de usá-lo.

Problemas com Ponteiros

- Endereço vs. Conteúdo

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

int main()
{
  int x, *p;

  x = 10;
  p = x;
  system("PAUSE");
  return 0;
}
```

Problema: // não compila!!!!
O programa está tentando atribuir o valor de x (10) a um endereço de memória.

Solução:
p = &x; //p recebe o endereço de x

Exercício 1

- Modifique o programa abaixo para imprimir o conteúdo do vetor usando ponteiros.

```
// Programando como vetor
#include <stdio.h>
#include <stdlib.h>

int main()
{
   float v[] = {1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0};
   int i;

   for (i=0; i<9; i++)
       printf("%.1f ", v[i]); // imprime com 1 casa decimal

  system("pause");
  return 0;
}
```


Solução

```
// Programando como ponteiro:
#include <stdio.h>
#include <stdlib.h>

int main()
{
   float v[] = {1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0};
   int i;
   //for (i=0; i<9; i++)
   //        printf("%.1f ", v[i]);
  // trabalhando como  ponteiro:
  for (i=0; i<9; i++)
   {
      printf("%.1f ", *(v+i)); // imprime com 1 casa decimal
      printf("%d ", (v+i));     // endereço apontado
   }
  system("pause");
  return 0;
}
```

```
1.0 2293568
2.0 2293572
3.0 2293576
4.0 2293580
5.0 2293584
6.0 2293588
7.0 2293592
8.0 2293596
9.0 2293600
```


Exercício 2

- Faça um programa que leia uma string e, utilizando ponteiros, conte quantos caracteres a string possui (sem utilizar *strlen*).

Solução

```
...

int main()
{
   char str[100], *p;
   int tam=0;
   puts("digite a string: ");
   gets(str);
   p = str; // p aponta para o primeiro caractere de str
   while (*p != '\0')
   {
     tam++;
     p++; // desloca ponteiro
   }
    printf ("A string tem %d letras.\n\n", tam);
   system("pause");
   return 0;
}
```

ou // sem usar variável *p:
while (*(str + tam) != '\0')
tam++;

ou while (*(p + tam) != '\0')
tam++;

O ponteiro não muda de posição

```
C:\Cora\Dis...
digite a string:
abacaxi
A string tem 7 letras.
```


Formas de passagem de valores de parâmetros

- Por valor:
  - passa somente o valor.
  - para cada parâmetro, é alocada uma área de uso local da função e o valor do argumento é copiado para esta área (valor inicial).
  - alterações deste valor, durante o processamento da função, não alteram o valor da variável enviada pelo programa que chamou a função (argumento ou parâmetro real), apenas a cópia local.

Funções void: parâmetros

- Por referência:
  - passa um endereço de memória, o qual é associado ao parâmetro, localmente.
  - alterações feitas no parâmetro afetam o valor da variável (é a própria variável) referida na chamada (argumento ou parâmetro real).
  - para passar parâmetros por referência precisamos usar ponteiros.
  - ATENÇÃO:
    - ✓ Vetores são SEMPRE passados por referência! (são ponteiros...)
    - ✓ É possível que o argumento de uma string seja enviado no formato de constante literal: neste caso, qualquer tentativa de modificação do conteúdo do argumento recebido irá causar erro de execução.

Exemplo 1: passagem por valor

- Fazer um procedimento que receba 2 inteiros x e y e troque seus valores.

```
void troca(int x, int y)
{
    int temp;

    temp = x;
    x = y;
    y = temp;
}
```

```
int main()
{

    int a=5, b=10;
    printf("a=%d b=%d\n",a,b);
    troca(a,b);
    printf("a=%d b=%d\n",a,b);
    .....
}
```

O que será impresso na tela?


```
C:\Documents and Set
a=5 b=10
a=5 b=10
```

- Os valores não foram trocados, porque a passagem de parâmetros foi feita por valor.
- Dentro do subprograma, os valores de x e y foram trocados, mas esta mudança foi feita sobre as cópias locais dos valores e não foi propagada para fora do subprograma.

Exemplo 2: passagem por referência

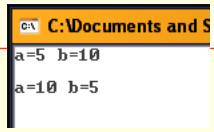
- Fazer um procedimento que receba 2 inteiros x e y e troque seus valores.

```
void troca(int *x, int *y)
{
    int temp;

    temp=*x;
    *x = *y;
    *y = temp;
}
```

```
int main()
{
    int a=5, b=10;
    printf("a=%d b=%d\n",a,b);
    troca(&a,&b);
    printf("a=%d b=%d\n",a,b);
    getch();
}
```

x e y agora recebem os endereços de a e b



- Os valores foram trocados, pois a passagem de parâmetros foi feita por referência.
- Dentro do subprograma, x e y receberam os endereços de memória de a e b.

Faça um programa que gere 4 matrizes inteiras, 4 x 4 (a, b, c e d) de números aleatórios, com números inteiros entre 1 e 10.
Faça uma função void que **recebe** duas matrizes 4 x 4, calcula e imprime a matriz-soma e **retorna ao programa principal** o valor da **soma** dos elementos da **matriz soma**.

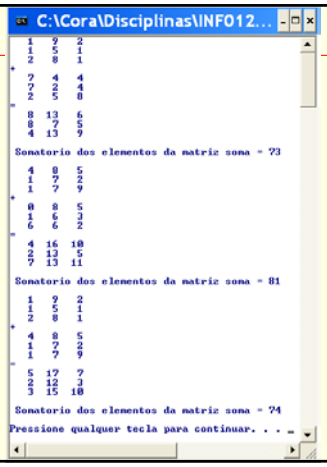
```
//Gera matrizes, soma matrizes e retorna somatorio:
#include <stdio.h>
#include <stdlib.h>
#include <time.h> // para usar função time
#define MAX 4
/* função que imprime uma matriz de MAX x MAX elementos*/
void imprmat (int mat [ ][MAX] ) // é ponteiro! 1ª dimensão não precisa ser definida
{
    int lin, col;
    //imprimindo as matrizes de entrada
    for (lin=0; lin<MAX; lin++)
    {
      for (col=0; col<MAX; col++)
         printf("%4d",mat [lin][col]);
      printf("\n");
    }
}
.... // próximo slide
```


Faça um programa que gere 4 matrizes inteiras, 4 x 4 (a, b, c e d) de números aleatórios, com números inteiros entre 1 e 10.
Faça uma função void que **recebe** duas matrizes 4 x 4, calcula e imprime a matriz-soma e **retorna ao programa principal** o valor da **soma** dos elementos da **matriz soma**.

```
.....
/* recebe 2 matrizes (são ponteiros, mesmo que não precise retornar nada) e
   retorna um valor inteiro (tem que definir ponteiro!!!) */
void somamat (int mat1[MAX][MAX], int mat2[MAX][MAX],int *somatorio )
{
   int lin, col; // indices das matrizes
   int matsoma[MAX][MAX]; // matriz local, que recebe a soma dos elementos
  //imprimindo as matrizes de entrada
   imprmat (mat1) ; // imprime através de função
   printf("+\n");      // imprime linha com símbolo + e quebra de linha
   imprmat (mat2) ;
   printf("=\n");

  //preenchendo matriz soma, calculando somatório e imprimindo matriz soma
   *somatorio = 0; // zera somatorio
   for (lin=0; lin<MAX; lin++)
       for (col=0; col<MAX; col++)
       {
           matsoma[ lin][col]= mat1[lin][col] + mat2[lin][col];
           *somatorio += matsoma[lin][col];  // incrementa somatório
       }
    imprmat (matsoma) ;
}
```

```
#include <time.h> // para usar função time
void imprmat (int mat [][MAX] )
{..... }
void somamat (int mat1[MAX][MAX], int mat2[MAX][MAX],int *somatorio )
{ ..... }
//Programa principal:
int main ( )
{
   int l,c, ma[MAX][MAX], mb[MAX][MAX], mc[MAX][MAX], md[MAX][MAX];
   int soma; // variável que contém somatório da matriz soma
   system("color f1");
   srand(time(NULL)); // valor inicial para o gerador de nros randômicos
   //criando as matrizes com valores randômicos entre 0 e 9:
   for (l=0; l<MAX; l++)
       for (c=0; c<MAX; c++)
       {
          ma[ l ][c] = rand( )%10;          mb[ l ][c] = rand( )%10;
          mc[ l ][c] = rand( )%10;          md[ l ][c] = rand( )%10;
       }
   //chamada da função para somar as matrizes:
   somamat(ma,mb,&soma); // manda o endereço da variável soma
   printf("\n Somatorio dos elementos da matriz soma = %d\n\n", soma);
   somamat(mc,md,&soma);
   printf("\n Somatorio dos elementos da matriz soma = %d\n\n", soma);
   somamat(ma,mc,&soma);
   printf("\n Somatorio dos elementos da matriz soma = %d\n\n", soma);
   system("pause");
   return 0;
}
```



Exercício:

Escrever um programa (a partir da tela a seguir e do conteúdo definido no slide 26) que executa o seguinte algoritmo:

1. preenche a matriz mat (LxC) por leitura (valores fornecidos por coluna à coluna);
2. imprime os valores da matriz no formato linha à linha (alinhado!);
3. calcula a soma de cada coluna da matriz e armazena esta soma em um vetor V;
4. imprime o vetor;
5. procura o maior elemento da matriz e sua posição;
6. informa o valor e esta posição (não precisa prever duplicidade).

Os passos 1 a 5 deverão ser executados por funções, chamadas a partir do programa principal.
O passo 6 deverá ser executado no programa principal.
Testar o programa com L = 3 e C = 4.

Não usar variáveis globais

Tela da execução:

```
Forneca valores da coluna 1:
    Linha 1: 1
    Linha 2: 3
    Linha 3: 5
Forneca valores da coluna 2:
    Linha 1: 4
    Linha 2: 18
    Linha 3: 11
Forneca valores da coluna 3:
    Linha 1: 2
    Linha 2: 6
    Linha 3: 8
Forneca valores da coluna 4:
    Linha 1: 12
    Linha 2: 13
    Linha 3: 14

Matriz lida:
    1    4    2   12
    3   18    6   13
    5   11    8   14

Soma das colunas da matriz:
    9   33   16   39

Maior valor:18
Posicao: [2,2]

Pressione qualquer tecla para continuar.
```



```c
// Trabalho da aula 19:
#include <stdio.h>
#include <stdlib.h>
#define L 3
#define C 4
// protótipo das funções a serem desenvolvidas:
void le_matriz(int [ ][C]);
void imprime_matriz(char [ ],int [ ][C]); // texto a ser impresso + matriz
void soma_col_mat(int [L][C], int [C]);
void imprime_vetor(char[ ],int [ ],int); // texto a ser impr,vetor,num elem do vet
void maior_elemento(int mat[L][C], int *pos_lin_maior, int *pos_col_maior)
// programa principal:
int main( )
 {
    int mat[L][C], vet[C]; // matriz e vetor somatório das colunas
    int lin_maior, col_maior; // posição do maior valor
    system("color f1");
    le_matriz(mat);
    imprime_matriz("Matriz lida",mat);
    soma_col_mat(mat, vet);
    imprime_vetor("Soma das colunas da matriz",vet, C);
    maior_elemento(mat, &lin_maior, &col_maior); //obtém e imprime a seguir!
    printf("\nMaior valor:%d \nPosicao: [%d,%d]\n",mat[lin_maior][col_maior],
                                                   lin_maior + 1, col_maior + 1);
    printf("\n");
    system("pause");
    return 0;
 }
```