FICHA 2: PONTEIROS

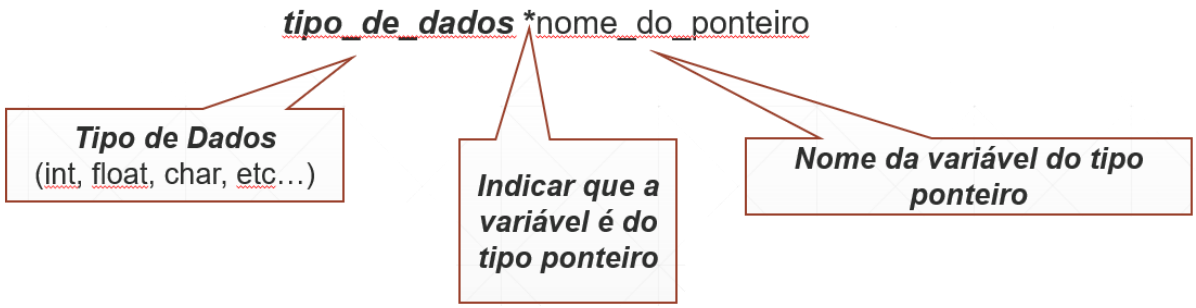
MODALIDADE:	Educação e Formação de Adultos (EFA)	EFA NS (Profissional)	
CURSO:	Programador/a Informático/a		
UFCD:	Programação em C/C++ - avançado	CÓDIGO UFCD:	0810
FORMADOR/A:	Bruno Silva	DATA:	

OBJETIVOS

- Saber como trabalhar com ponteiros e referências na memória

Os ponteiros são um mecanismo particularmente flexível de manipulação dos dados, ***pois permitem manipular diretamente os dados contidos nos endereços específicos da memória***. Um ponteiro é uma variável como outra qualquer. O seu objetivo é armazenar no seu valor, o endereço de outra variável, ou seja, está a apontar para algo.

A sintaxe de declaração de um ponteiro é o seguinte:



Exemplo de ponteiros:

- ***char*** a, *p, c, *b; *(p e b são ponteiros de tipo caracteres)*
- ***int*** idade, *p_idade; *(p_idade é ponteiro de tipo inteiro)*

Carga inicial automática de ponteiros

- Uma boa regra de programação para evitar problemas de programação é sempre que possível, iniciar os ponteiros num programa.
- No entanto podem existir situações em que declaramos um ponteiro e não queremos que ele aponte para nenhuma variável.
- Nestes casos, podemos apontar o mesmo para um valor Nulo (*NULL*).

FICHA 2: PONTEIROS

Um apontador é uma variável que contém o endereço de uma variável ou outro objecto de memória (elemento de array, campo de estrutura, etc.)

- ***Operador de endereço: &***
  - **&x = endereço do objecto x**
- ***Operador indireto ou ponteiro: ***
  - ***p = objecto apontado por p (o endereço do objecto é o valor de p)**
- ***Exemplo de um apontador para inteiro*: int *p; *(*p é do tipo int)***

Exercício 1 – Vamos ver qual o resultado das operações (com x = 1 e y =2):

int x = 1, y = 2;

int *p; /* p é um apontador para int */

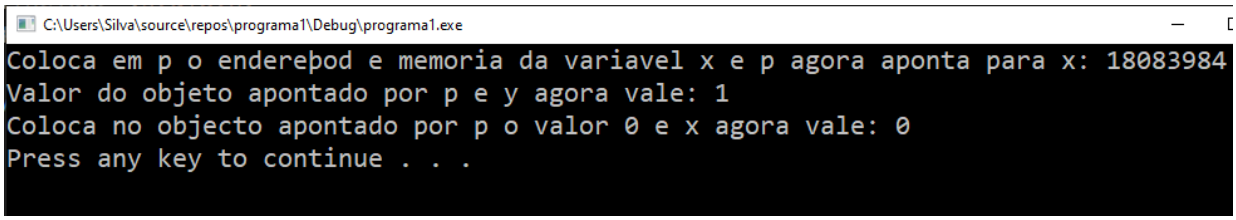
p = &x; /* coloca em p o endereço de x e p agora aponta para x */

y = *p; /* coloca em y o valor do objeto apontado por p e y agora vale 1 */

***p = 0; /* coloca no objecto apontado por p o valor 0 e x agora vale 0 */**

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

void main() {

    int x = 1, y = 2;
    int* p;    /* p é um apontador para int */

    p = &x;    /* coloca em p o endereço de x e p agora aponta para x */
    printf("Coloca em p o endereçod e memoria da variavel x e p agora aponta para x: %d \n", p);

    y = *p;    /* coloca em y o valor do objeto apontado por p e y agora vale 1 */
    printf("Valor do objeto apontado por p e y agora vale: %d \n", y);

    *p = 0;       /* coloca no objecto apontado por p o valor 0 e x agora vale 0 */
    printf("Coloca no objecto apontado por p o valor 0 e x agora vale: %d \n", *p);

    system("PAUSE");
}
```

Exercício 2 – Teste o programa e verifique quais foram os novos valores.



Como visto no UFCD 0809 - Programação em C - C++ (Fundamentos), houve uma seção que falava sobre a passagem de valores numa função por parâmetros.

Mas existe outro método para passar os parâmetros para uma função!

1. ***Passagem por parâmetros - os valores que são passados para as funções não são alteráveis***, ou seja, ***trabalhamos sobre uma cópia dos valores que foram fornecidos***. Qualquer alteração estará confinada à cópia efetuada, e nunca ao argumento original. Se for necessário retornar algo, temos de utilizar o ***return***!

Exercício 3 –Teste o programa a seguir e verifique quais foram os novos valores, utilizando o princípio da passagem por parâmetros;

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

/*Passagem de dados por parametros*/
/*Esta função vai trabalhar com uma cópia do valor que foi fornecido pelo main()*/
void multiplica_2x(int valor) {
    int resultado = valor * 2;
    printf("O dobro de %d e: %d\n", valor, resultado);
}

void main() {
    int soma = 2;
    printf("O valor da variavel soma tem o valor %d!\n", soma);

    /*Passagem de dados por a função multiplica_2x e leva 1 valor/parametro*/
    multiplica_2x(soma);

    printf("O valor da variavel soma continua igual desde o inicio!\n");

    system("PAUSE");
}
```

Mas, existe outra forma de enviar os parâmetros para as funções:

2. ***Passagem por referência - <u>os valores que são passados para as funções são alteráveis</u>***. As alterações realizadas nessa referência irão ter reflexo nos parâmetros passados **<u>e</u>** no programa que os invocar, ou seja, afeta os valores em todo o programa.

   A passagem por referência permite alterar o conteúdo das variáveis invocadas da função.

   Mas, na linguagem C, só existe passagem de parâmetros por valor.

   Ou seja, em C nunca é possível alterar os argumentos enviados para uma função. ***Como podemos fazer passagem por referência?***

   A solução está na utilização dos ponteiros! (Continua a saga dos ponteiros)

   Exercício 4 –Teste o programa a seguir e verifique quais foram os novos valores, utilizando o princípio da passagem por referência;

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

/*Função troca dois numeros através da passagem de valores por referência
  Através do uso de ponteiros para indicar os valores dos endereços
  que passamos na função troca_numeros invocada na funcao do main*/
void troca_numeros(int *n1, int *n2) { /*Indicar os tipos de ponteiros que vamos receber*/
    int temp;
    temp = *n1;
    *n1 = *n2;
    *n2 = temp;

    /*reparem que não fizemos nenhum retorno de valores*/
}

void main() {
    int a = 5, b = 8; /*Valores originais de a e b*/

    /*Temos de indicar qual os endereços na memoria das variaveis a (&a) e b (&b)
      & indica os endereços na memoria das variaveis
      *<nome_variavel> indica os valores dos endereços na memoria das variaveis */
    troca_numeros(&a, &b);

    printf("As variaveis originais a e b foram mdificadas pois:\n");
    printf("Valor original da Variavel a: %d e agora tem o valor: %d!\n", 5, a);
    printf("Valor original da Variavel b: %d e agora tem o valor: %d!\n", 8, b);

    system("PAUSE");
}
```

FICHA 2: PONTEIROS

EXERCICIOS

Exercício 1 – Crie um programa para criar duas variáveis inteiras (uma variável normal com um valor a sua escolha e outra do tipo ponteiro). De seguida, deve utilizar a variável do apontador para apontar e colocar o valor da variável inteira. No final mostre na consola as informações:

- Conteúdo da variável inteira;
- Endereço da variável valor;
- Conteúdo da variável ponteiro;

Exercício 2 – Escreva um programa que declare um inteiro, um decimal e um char, e ponteiros para inteiro, real, e char. Associe as variáveis aos ponteiros (use &). Modifique os valores de cada variável usando os ponteiros. Imprima os valores das variáveis antes e após a modificação.

Exercício 3 – Escreva um programa que contenha duas variáveis inteiras. Compare seus endereços e exiba o maior endereço.

Exercício 4 – Escreva um programa que leia 2 valores inteiros e chame uma função que receba 2 variáveis (com passagem de valores por referência), trocando os valores e deve mostrar no programa que invocou a função quais os valores finais.

Exercício 5 – Faça um programa que leia dois valores inteiros e chame um procedimento que receba 2 valores inteiros (passagem por parâmetros) de entrada e indique na consola, qual dos dois números é o maior.

Exercício 6 – Crie um programa que contenha uma função que permita passar por parâmetro dois números inteiros A e B. A função deverá calcular a soma entre estes dois números e armazenar o resultado na variável A. Esta função não deverá possuir retorno, mas deverá modificar o valor do primeiro parâmetro. Imprima os valores de A e B na função principal.