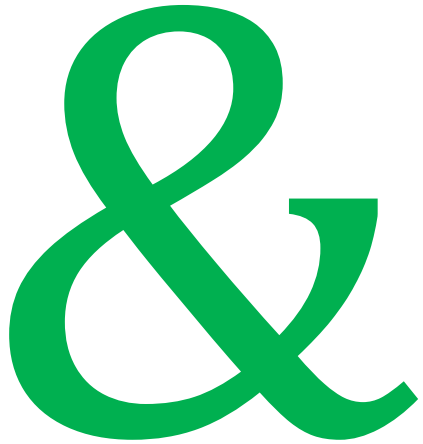
Ponteiros

Profa. Elloá B. Guedes

www.elloaguedes.com

UEA
UNIVERSIDADE
DO ESTADO DO
AMAZONAS



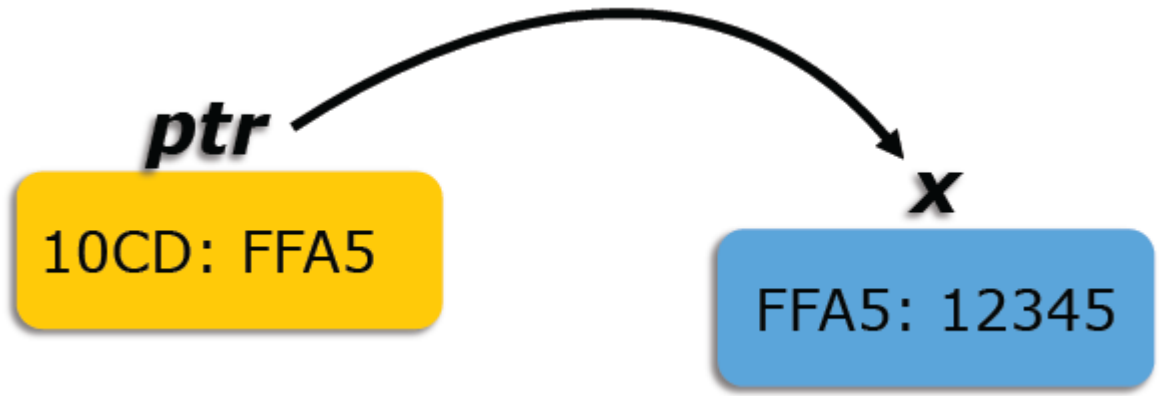
- Operador de endereço
- Fornece o endereço de memória de uma variável

- Operador de derreferência
- Declara uma variável do tipo de dado “ponteiro para algum tipo de dado”

- Uma variável do tipo ponteiro guarda o **endereço de memória** de uma outra variável.

- Quando um variável ***ptr*** é do tipo ponteiro e armazena o endereço de uma outra variável ***x*** dizemos que "***ptr*** aponta para ***x***".

- Quando um variável ***ptr*** é do tipo ponteiro e armazena o endereço de uma outra variável ***x*** dizemos que "***ptr*** aponta para ***x***".



- Quando um variável ***ptr*** é do tipo ponteiro e armazena o endereço de uma outra variável ***x*** dizemos que "***ptr*** aponta para ***x***".

Ponteiros

- Variáveis são armazenadas na memória
  - Possuem um tipo e um endereço no qual são armazenadas
- Ponteiros guardam endereços de memória
  - Primeira utilização: apontar para variáveis
- Um ponteiro também tem tipo
  - A depender de para quem ele aponta

Ponteiros

- Para declarar um ponteiro temos a seguinte forma geral:

<tipo> *<nome da variável>;

- Operador asterisco
  - Indica que a variável não vai guardar um valor, mas sim um endereço
- Exemplos:
  - int *pt;
  - char *temp, *pt2;

Utilização de Ponteiros

- Ponteiros não inicializados apontam para um lugar indefinido.

- Ponteiros precisam ser inicializados
  - Devem ser apontado para algum lugar conhecido antes de ser usados
  - Solução: igualá-lo a um valor de memória

- Operador &: Retorna o endereço de uma variável

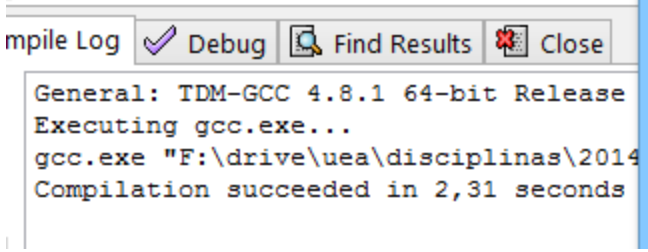
Exemplo

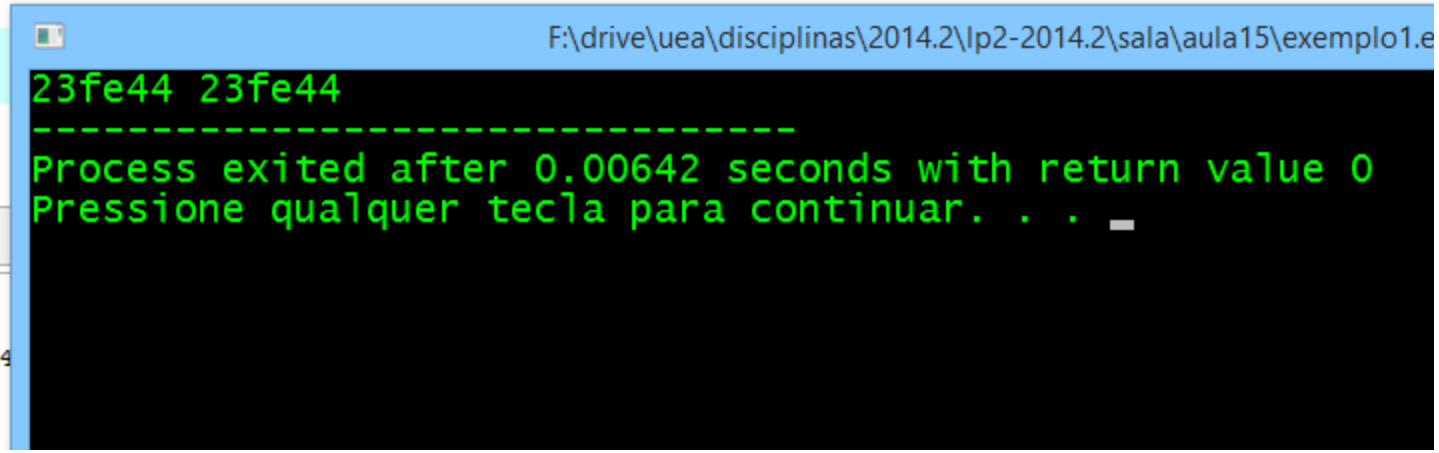
```c
#include <stdio.h>

int main(){
    int n = 10;
    int *pointer = &n;

    printf("%x %x",&n,pointer);

    return 0;
}
```





Utilização de Ponteiros

- Para alterar o valor de uma variável apontado por um ponteiro, basta usar o operador *

```
int count=10;
int *pt;
pt = &count;
*pt = 12;
```

- Resumindo,
- *pt: o conteúdo da posição de memória apontado por pt
- &count: o endereço onde a variável count está armazenada

Exemplo

```c
#include <stdio.h>

int main(){
    int a = 3, b = 2, c;
    int *p;
    int *q;
    p = &a;
    q = &b;
    c = *p + *q;

    printf("%d",c);

    return 0;
}
```

Exemplo

```
void main(){
    int num, valor;
    int *p;

    num = 55;
    p = &num;
    valor = *p;

    printf ("%d", valor);
    printf ("\nEndereco: %p\n", p);
    printf ("Valor da variavel apontada: %d\n", *p);

}
```

Funções com Ponteiros

```
void troca(int i, int j){
     int aux;
     aux = i;
     i = j;
     j = aux;
}

void main(){
     int a = 0, b = 1;
     printf("%d %d",a,b);
     troca(a,b);
     printf("\n%d %d",a,b);
}
```

Funções com Ponteiros

```c
#include <stdio.h>

void troca( int *i, int *j)
{
    int temp;
    temp = *i; *i = *j; *j = temp;
}

void main(){
     int a = 0, b = 1;
     printf("%d %d",a,b);
     troca(&a,&b);
     printf("\n%d %d",a,b);
}
```

Funções com Ponteiros

```c
#include <stdio.h>

void troca( int *i, int *j) {
    int *temp;
    *temp = *i; *i = *j; *j = *temp;
}

void main(){
     int a = 0, b = 1;
     printf("%d %d",a,b);
     troca(&a,&b);
     printf("\n%d %d",a,b);
}
```

Operações com Ponteiros

- Atribuição
  - p1 = p2; p1 aponte para o mesmo lugar que p2;

- Conteúdo:
- *p1 = *p2; a variável apontada por p1 tenha o mesmo conteúdo da variável apontada por p2;

- Incremento
  - p++; passa a apontar para o próximo valor do mesmo tipo para o qual o ponteiro aponta. Isto é, se temos um ponteiro para um inteiro e o incrementamos ele passa a apontar para o próximo inteiro

- Decremento:
  - p--; funciona semelhantemente

Operações Aritméticas com Ponteiros

- (*p)++;
  - Incrementar o conteúdo da variável apontada pelo ponteiro p;

- p = p+15; ou p+=15;
  - Incrementar um ponteiro de 15 posições de inteiros;

- *(p +15);
  - Usar o conteúdo do ponteiro 15 posições adiante;

Operações Relacionais com Ponteiros

- == e !=
  - para saber se dois ponteiros são iguais ou diferentes;

- >, <, >= e <=
  - Comparando dois ponteiros em relação à sua posição na memória

Passagem por Valor vs Passagem por Referência

- **Passagem de parâmetros por valor:** A função recebe uma cópia da variável que é fornecida quando é invocada. Todas as alterações feitas dentro da função não vão afetar os valores originais.

- **Passagem de parâmetros por referência:** Neste caso o que é enviado para a função é uma referência às variáveis utilizadas, e não uma simples cópia, pelo que as alterações realizadas dentro da função irão certamente alterar os valores contidos nessas variáveis.

Passagem por Valor vs Passagem por Referência

- **Em C não há passagem de parâmetros por referência**

- Uso de ponteiros é um “truque” para tornar isto possível na linguagem C

Recomendação de Leitura

- Damas – Capítulo 8