Departamento
de Ciência da
COMPUTAÇÃO

UDESC
Joinville

Linguagem C
Ponteiro para Funções

André Tavares da Silva

andre.silva@udesc.br

Ponteiro para funções

- O C permite que acessemos não só variáveis através de ponteiros. Podemos também acessar **funções** através de ponteiros.
- Pode-se desta forma passar uma função como argumento de uma outra função.

Ponteiro para funções

- Sintaxe:

```
<tipo> (*<identificador>) ( <parametros> );
```

- Exemplo:

```
float (*func) ( float, float );
```

- Os parênteses em torno de *func são essenciais, pois sem eles o compilador entenderia o argumento como uma função que retorna um ponteiro para um inteiro.
- Pode-se também fazer um ponteiro para uma função que retorna um ponteiro:

```
float* (*func) ( float, float );
```

Exemplo


```
#include <stdio.h>

float plus( float a, float b)      { return a + b; }
float minus( float a, float b)     { return a - b; }
float times( float a, float b)     { return a * b; }
float division( float a, float b) { return a / b; }

int main()
{
  float x, y;
  float (*operation)(float, float);

  scanf("%f %f", &x, &y);

  operation = plus;
  printf("%f + %f: %f\n", x, y, operation(x,y));
  operation = minus;
  printf("%f - %f: %f\n", x, y, operation(x,y));
  operation = times;
  printf("%f * %f: %f\n", x, y, operation(x,y));
  operation = division;
  printf("%f / %f: %f\n", x, y, operation(x,y));

  return 0;
}
```

Passando uma função como argumento a outra função

```
void ExecutaFuncao(float (*ptrFunc)(float, float))
{
  // execução da função apontada por ptrFunc
  float result = (*ptrFunc)(3.5, 2.7);
  printf("resultado da funcao: %.2f", result);
}

void ExecutaFuncoes()
{
  // passando por parâmetro a função
  ExecutaFuncao(&plus);
  ExecutaFuncao(&minus);
}
```

Exercício

- Construa um programa que receba da linha de comando, com a qual o programa é executado, dois inteiros e a descrição da operação que será efetuada sobre eles (soma, subtração, multiplicação, divisão e resto). O programa deve possuir uma função para efetuar cada uma das operações mencionadas, devendo se utilizar do conceito de "ponteiro para função" na seleção da função adequada a ser executada.

Exercício

- Construa duas funções que recebem como parâmetro dois valores quaisquer (*float*):
  - `int maior(float, float)` : retorna 1 (um) se o primeiro argumento for maior que o segundo e 0 (zero) caso contrário;
  - `int menor(float, float)` : retorna 1 (um) se o primeiro argumento for menor que o segundo e 0 (zero) caso contrário;
- Faça uma outra função que ordene um vetor de números reais, recebendo como parâmetro o vetor (ponteiro) e a uma das funções acima para ordenar de forma crescente ou decrescente.