

# Linguagem C

# Funções

André Tavares da Silva

[atavares@joinville.udesc.br](mailto:atavares@joinville.udesc.br)

# Funções

- É uma coleção de comandos identificados por um nome (identificador);
- As funções executam ações e podem retornar valores;
- main, printf e scanf são exemplos de funções.

```
<tipo retornado> <nome>(<parâmetro 1>, . . . ,<parâmetro n >)
{
    <declaração das variáveis locais>
    <comandos>
}
```

# Funções

- É uma coleção de comandos identificados por um nome (identificador);
- As funções executam ações e podem retornar valores;
- main, printf e scanf são exemplos de funções.

```
void soma(int a, int b)
{
    int resultado = a + b;
    printf("soma de %d e %d é %d", a, b, resultado);
}
```

# Funções

- **Tipo retornado:** pode ser qualquer um dos tipos que já vimos (int, short int, long int, char, ...) ou então void, que indica que a função não retornará nenhuma informação.
- **Nome:** é o identificador da função, servindo para referenciar a mesma.
- **Parâmetros:** são as informações para trabalhar. Constituem-se de variáveis locais à função, que são declarados individualmente, separados por vírgula.

# Funções

- Quando uma função é definida com o tipo de retorno ***void***, ela também pode ser chamada de procedimento pois somente executa um conjunto de comandos e não retorna nenhum valor;

```
void soma(int a, int b)
{
    int resultado;
    resultado = a + b;
    printf("soma de %d e %d é %d", a, b, resultado);
}
```

# Funções

- Quando o tipo de retorno é diferente de *void*, precisamos retornar um valor para o trecho de código que chamou a função. Isto é feito por meio do comando *return*.

```
int soma(int a, int b)
{
    int resultado;
    Resultado = a + b;
    return resultado;
}
```

# Passagem de Parâmetros

- **Por valor:** São fornecidas cópias dos valores dos parâmetros na expressão que chama a função. A função pode alterar o valor dos parâmetros, mas esta mudança não é refletida às variáveis originais. É o tipo padrão na linguagem C.

```
void ImprimirValor (float v) {  
    printf("%.2f", v);  
}
```

# Passagem de Parâmetros

- **Por referência:** Este tipo permite que se altere o conteúdo das variáveis passadas como parâmetro. A função recebe um endereço de memória como parâmetro. É exatamente o que acontece quando utilizamos a função *scanf*. É informado que os dados lidos deverão ser armazenados em variáveis passadas por parâmetro.

```
scanf("%f", &valor); /* note o & */
```

# Variáveis Locais

- Variáveis declaradas dentro de uma função.
- Não são reconhecidas fora de seu próprio bloco de código.
- Existem apenas enquanto o bloco de código em que foram declaradas está sendo executado, ou seja, é criada na entrada de seu bloco e destruída na saída.

# Variáveis Globais

- São reconhecidas pelo programa inteiro e podem ser usadas por qualquer pedaço de código.
- Guardam seus valores durante toda a execução.
- Devem ser declaradas fora de qualquer função.
- Ocupam memória durante todo o tempo de execução do programa.
- Tornam as funções menos gerais.
- Podem levar a erros no programa devido a efeitos colaterais.

# Protótipo de Funções

- É a repetição da declaração da função, porém não necessita dos nomes das variáveis que recebem os parâmetros, encerrando com ponto e vírgula (;).
- Isto ajuda o compilador a verificar se os parâmetros que estão sendo passados nas funções estão de acordo com os tipos que elas esperam receber.

# Protótipo de Funções

- É feita extraindo-se:
  - Tipo retornado pela função
  - Nome da função;
  - Tipo dos parâmetros entre parênteses

```
void ImprimirValor(float);  
int soma(int, int);
```

```
#include <stdio.h>      /* inclusão de biblioteca - entrada e saída */

const float VALORHORA=10.0;      /* variável global e constante */

int main() {                      /* função principal */

    int horas = LeHoras();
    printf("A receber: %.2f\n", CalculaSalario(horas) );
    return 0;
}

float CalculaSalario(int h) {        /*calcula salário*/
    return (float)h*VALORHORA;
}

int LeHoras() {                    /* função para leitura de horas */
    int horas;                     /* variável local */
    printf("Entre numero de horas trabalhadas: ");
    scanf("%d", &horas );
    return horas;
}
```

```
#include <stdio.h>          /* inclusão de biblioteca - entrada e saída */
int LeHoras();                /* protótipo */
float CalculaSalario(int);    /* protótipo */

const float VALORHORA=10.0;    /* variável global e constante */

int main() {                   /* função principal */
    int horas = LeHoras();
    printf("A receber: %.2f\n", CalculaSalario(horas) );
    return 0;
}

float CalculaSalario(int h) {   /*calcula salário*/
    return (float)h*VALORHORA;
}

int LeHoras() {                /* função para leitura de horas */
    int horas;                  /* variável local */
    printf("Entre numero de horas trabalhadas: ");
    scanf("%d", &horas );
    return horas;
}
```

# Exercício

- Escreva um programa em C que executa conversões de temperatura entre os sistemas CELSIUS e FAHRENHEIT. Caso o usuário deseje converter de CELSIUS para FAHRENHEIT ele pressiona a tecla C; pressionando a tecla F o programa faz a conversão de FAHRENHEIT para CELSIUS. ( $F = C * (9/5) + 32$ ).

# Exercício

- Altere o programa anterior incluindo duas funções: uma para converter de graus Celsius para graus Fahrenheit (**float CelsiusParaFahr (float)**) e outra para converter de graus Fahrenheit para Celsius (**float FahrParaCelsius (float)**).