Aula 11: Funções

Prof. Sérgio Montazzolli Silva
smsilva@uel.br

Introdução

- Até agora **utilizamos** várias funções diferentes, como printf(), scanf(), gets(), sin(), cos(), etc...
- E **criamos** apenas uma única função: **main**()
- Nesta aula aprenderemos a criar novas funções!
- Veremos que este conceito é extremamente útil e fundamental para a programação, uma vez que permite a:
  - eficiente reutilização de código
  - facilita a leitura para programas longos (melhor organização)

Funções

- Uma função é definida por:

```
tipo identificador(tipo param1, ..., tipo paramN) {
    // corpo (código)
}
```

- *Tipo*: refere-se ao tipo de dado que a função retornará como resposta
- *Identificador*: nome da função
- *Parâmetros*: lista de variáveis que são passadas a função
- *Corpo*: local onde ficam as instruções que serão executadas

Funções

- Note que a função **main** possui:
  - *Tipo* (retorno): int
  - *Identificador*: main
  - *Parâmetros*: nenhum
  - *Corpo*: tudo que estiver entre { }s
- Resumidamente, podemos dizer que a função **main** não possui parâmetros e retorna um valor inteiro

Exemplo

- Vamos analisar um exemplo de uma função simples de soma:
  - A função abaixo chama-se "soma". Ela recebe duas variáveis do tipo inteiro, soma, e retorna o resultado

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

int main() {
    int r = 10, s = 50, t;
    t = soma(r,s);
    printf("%d",t);
}
```

Parametrização

- Os parâmetros na declaração de uma função, quando existirem, precisam ser acompanhados de um tipo e um nome (identificação)
  - Na função **soma**(), temos "int a" e "int b"
  - Ou seja, "tipo identificação"

- Parâmetros devem ser separado por vírgula:
  - Na função **soma**(), temos "(int a, int b)"
  - Note que a vírgula separa as duplas tipo + identificação
  - Ao contrário da declaração de variáveis feitas, onde podemos utilizar "*int a, b*", isto **não** é permitido na parametrização!
    - Por exemplo:
      - "int **soma**(int a,b)" está errado! Uma vez que **b** não possui tipo

Parametrização

- Funções podem possuir quantos parâmetros forem necessários, apenas lembre-se de declara-los antes
- No exemplo abaixo, temos uma função com 4 parâmetros de tipos diferentes:

```c
int minha_funcao(char nome[], char sexo, int idade, float altura) {
    printf("Nome  : %s\n",nome);
    printf("Sexo  : %c\n",sexo);
    printf("Idade : %d\n",idade);
    printf("Altura: %f\n",altura);
    return 0;
}

int main() {
    char n[] = "João da Silva";
    char s   = 'M';
    int  i   = 31;
    float a  = 1.86;
    minha_funcao(n,s,i,a);
}
```

Escopo

- Basicamente escopo é o que a função consegue enxergar!

```c
int minha_funcao(char nome[], char sexo, int idade, float altura) {
    printf("Nome   : %s\n",nome);
    printf("Sexo   : %c\n",sexo);
    printf("Idade  : %d\n",idade);
    printf("Altura: %f\n",altura);
    return 0;
}

int main() {
    char n[] = "João da Silva";
    char s   = 'M';
    int  i   = 31;
    float a  = 1.86;
    minha_funcao(n,s,i,a);
}
```

Escopo de minha_função()
nome
sexo
idade
altura

Escopo de main()
n, s, i, a

Escopo

- Basicamente escopo é o que a função consegue enxergar!

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

int main() {
    int r = 10, s = 50, t;
    t = soma(r,s);
    printf("%d",t);
}
```

Escopo de soma()
a
b

Escopo de main()
r,s,t

Escopo

- Uma função apenas tem acesso a variáveis que foram passadas como parâmetros ou que foram criadas dentro da mesma função!
- No exemplo da soma, as variáveis **r**,**s** e **t** estavam dentro do escopo da função **main**(), e por isso não podem ser acessadas por **soma**()
- Da mesma forma, **a** e **b** fazem parte do escopo de **soma**(), e não podem ser acessadas pela função **main**()
- É importante saber que, ao chamar a função **soma**(), o conteúdo das variáveis **r** e **s** são copiados para **a** e **b**! Logo, alterar o conteúdo de **a** ou **b** não altera de **r** ou **s**.
  - Esta cópia não acontece quando uma variável é uma string, vetor, ou é passada por referência ou ponteiro
  - *Referência e ponteiro não estão no "escopo" deste curso*
  - Falaremos sobre a cópia mais adiante

Escopo

- Vamos ver alguns exemplos:
  - Se tentarmos incrementar o valor de **r** dentro de **soma**(), o compilador irá acusar um erro, uma vez que **r** não existe no escopo de **soma**(), apenas no escopo de **main**()

```
int soma(int a, int b) {
    r++;
    return a + b;
}
```

Erro!

Escopo

- Vamos ver alguns exemplos:
  - Da mesma forma, teremos um erro ao tentarmos alterar o valor de **a** dentro da função **main**(), uma vez que **a** não existe nela

```
int main() {
    int r = 10, s = 50, t;
    t = soma(r,s);
    a = r;
    printf("%d",t);
}
```

Erro!

Escopo (identificação de variáveis)

- Variáveis podem ter o mesmo nome se estiverem em escopos diferentes!
- Por exemplo, poderíamos reescrever nosso programa de soma da seguinte maneira:

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

int main() {
    int a = 10, b = 50, c;
    c = soma(a,b);
    printf("%d",c);
}
```

Observe que criamos variáveis de mesmo nome em ambas as funções! Isto é aceito pelo compilador uma vez que não há ambiguidade se o escopo for diferente.

Retorno

- Toda declaração de função se inicia com algum tipo, que indica como deve ser o valor de retorno desta função!
- Portanto, uma função deve sempre retornar um valor do seu tipo.
- O retorno é dado pela palavra "**return**"
- Toda função declarada como **int**, **char**, **float**, ou qualquer outro tipo, deve retornar um valor
- Esta regra não vale para funções de tipo **void** e para a função **main**()
  - No caso da função main, por ser uma função de entrada, isto não é exigido, porém é recomendável ao menos incluir um "return 0;" ao final
- Note no exemplo anterior que "**int** minha_funcao(...)" exige o retorno de um valor inteiro, e por isso foi adicionado "return 0;" ao final.
- No caso da função **soma**() apresentada no inicio, a utilização do retorno é evidente, uma vez que ele usado para atribuir o resultado

Retorno

- Funções que não retornam nenhum valor podem ser declaradas utilizando o tipo **void**. Reescrevendo um dos exemplos anteriores, temos:

```
void minha_funcao(char nome[], char sexo, int idade, float altura) {
    printf("Nome  : %s\n",nome);
    printf("Sexo  : %c\n",sexo);
    printf("Idade : %d\n",idade);
    printf("Altura: %f\n",altura);
}

int main() {
    char n[] = "João da Silva";
    char s   = 'M';
    int  i   = 31;
    float a  = 1.86;
    minha_funcao(n,s,i,a);
}
```

Retorno

- O retorno de uma função pode ser utilizado para atribuição de outra variável, ou até mesmo como parâmetro de uma outra função, veja o exemplo:

```c
int soma(int a, int b) {
    return a + b;
}
```

```c
int main() {
    int a = 10, b = 20, c;

    c = soma(a,soma(a,b));
    printf("%d\n",c); // a + (a+b)

    printf("%d\n",soma(c,soma(a,b))); // c + (a+b)
}
```

Multiplas funções

- É possível criar quantas funções forem necessárias

```c
int soma(int a, int b) {
    return a + b;
}

int subtracao(int a, int b) {
    return a - b;
}

int multiplicacao(int a, int b) {
    return a * b;
}

int main() {
    int a = 10, b = 20;

    printf("%d\n", soma(a,b));
    printf("%d\n", subtracao(a,b));
    printf("%d\n", multiplicacao(a,b));
}
```

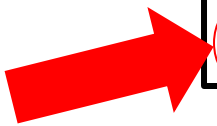
Cuidado!

- Apenas os chamados **tipos primitivos** (int, char e float) são copiados ao serem passados como parâmetros!
- Uma string não é um tipo primitivo, logo, se ela for alterada em uma função, o valor alterado permanecerá

```
void altera_string_e_int(char string[], int numero) {
    numero = numero + 10;
    string[0] = 'D';
}

int main() {
    char nome[] = "Joao da Silva";
    int  n = 100;

    printf("%s | %d\n",nome,n);
    altera_string_e_int(nome,n);
    printf("%s | %d\n",nome,n);
}
```



Saída:

```
Joao da Silva | 100
Doao da Silva | 100
```

Ordem

- Todas as funções devem ser escritas acima da função **main**()
- É também possível colocá-las abaixo, mas neste caso será necessário declara-las (sem corpo) acima da função **main**()
  - Veja o exemplo ao lado:

```c
int soma(int a, int b);
int subtracao(int a, int b);

int main() {
    int a = 10, b = 20;
    printf("%d\n", soma(a,b));
    printf("%d\n", subtracao(a,b));
}

int soma(int a, int b) {
    return a + b;
}

int subtracao(int a, int b) {
    return a - b;
}
```

Exercícios em Sala

- Crie um programa que possua uma função chamada **quadrado**, que retorne o quadrado de um número inteiro passado como parâmetro. Teste a função para os valores 2, 70, 324.

- Crie uma função que calcule $f(x, y, z) = x^2 + 3y + z$, onde x, y e z são números reais. Depois, na função **main**(), calcule $f(x, y, z)$ para x variando de 0 a 10 (de 1 em 1), e $y = 2, z = 80$. Note que y e z são fixos. Imprima na tela o resultado.

Exercícios em Sala

Crie uma função sem retorno, e sem parâmetros, chamada **main_alternativa**(), que resolva algum exercício visto anteriormente em qualquer aula anterior a esta.

A escolha do exercício fica a critério do aluno.

O programa deve funcionar normalmente, e o exercício deve ser resolvido completamente nesta nova função, mostrando na tela o resultado esperado (do exercício).

Exercícios em Sala

- Crie um programa que tenha uma função chamada **juntar_string**. Esta função deve receber como parâmetro duas strings lidas na função **main**(), e junta-las em uma terceira string, mostrando na tela.