*Disciplina: Linguagem de Programação II*
*Professora: Caroline Mazetto Mendes*

Aula 5 - 07/03/17

**1) Funções (Deitel 2011, cap 5)**

Um programa em C é composto por um conjunto de funções. Além da função *main()*, outras funções podem ser definidas pelo programador para realizar tarefas específicas. Essas funções podem ser acionadas (chamadas) dentro da função *main()* ou em outras funções.

Funções podem receber parâmetros de entrada e retornar valores de saída. Três tipo de funções são geralmente utilizadas:

- Funções que não retornam valor e não recebem parâmetros
- Funções que não retornam valor, porém recebem parâmetros
- Funções que retornam valor e recebem parâmetros

**Exemplo:** Função que não retorna valor e não recebe parâmetros.

```
#include <stdio.h>

void exibe_menu(void) {
	printf(“Cadastro de Estudantes\n”);
	printf(“1 - Cadastrar\n”);
	printf(“2 - Listar\n”);
	printf(“3 - Pesquisar\n”);
	printf(“4 - Atualizar\n”);
	printf(“5 - Atualizar\n”);
	printf(“6 - Sair\n”);
}

int main(void) {
	int opcao;

	exibe_menu();
	scanf(“%d”, &opcao);

	return 0;
}
```

**Código 1**: Exemplo de função que não retorna valor e não recebe parâmetros.

A palavra *void* é usada para indicar que a função não retorna valor e não recebe parâmetros. Nesse caso, basta chamar a função no local desejado, como mostra o exemplo.

Funções são declaradas após as diretivas e antes da função *main()*. Por regra, toda função deve ser declarada antes de ser chamada, ou seja, deve estar localizada antes da função chamadora. Existe a possibilidade de definir as funções em qualquer posição do código. Para isso, deve-se declarar o protótipo da função antes da sua definição.

**Exemplo:** Função que não retorna valor, porém recebe parâmetros.

```
#include <stdio.h>

/* Protótipo da função */
void imprime_data(int dia, int mes, int ano);

int main(void) {
	int dia, mes, ano;

	scanf(“%d %d %d”, &dia, &mes, &ano);
	imprime_data(dia, mes, ano);
	return 0;
}

void imprime_data(int dia, int mes, int ano) {

	switch (mes) {
		case 1:
			printf(“Curitiba, %d de janeiro de %d”, dia, ano);
			break;
		case 2:
			printf(“Curitiba, %d de fevereiro de %d”, dia, ano);
			break;
		case 3:
			printf(“Curitiba, %d de marco de %d”, dia, ano);
			break;
		// …
	}
}
```

**Código 2**: Exemplo de função que não retorna valor, porém recebe parâmetros.

Nesse caso, a palavra *void* é utilizada para indicar que a função não retorna valor. Contudo, os parâmetros de entrada devem ser especificados, informando o tipo e nome do parâmetro. Na chamada da função, variáveis são “passadas” à função conforme especificado no protótipo.

**Exemplo:** Função que retorna valor e recebe parâmetros.

```
#include <stdio.h>

int retorna_maior(int a, int b) {
	int maior;

	if (a > b) {
		maior = a;
	}
	else {
		maior = b;
	}

	return maior;
}
```

```
int main(void) {

	int num1, num2, resultado;

	scanf(“%d %d”, &num1, &num2);

	resultado = retorna_menor(num1, num2);

	printf(“Maior = %d”, resultado);

	return 0;
}
```

**Código 3:** Exemplo de função que retorna valor e recebe parâmetros.

Nesse exemplo, a função retorna um valor do tipo inteiro e recebe parâmetros de entrada. O comando *return* (obrigatório) é utilizado para retornar um valor à função chamadora. Por fim, o valor de retorno é atribuído a uma variável na chamada da função.

**Obs.:** O comando *return* pode ser utilizado em qualquer parte do corpo da função. No entanto, a sua execução determina o retorno do valor e encerramento imediato da função. Instruções subsequentes não serão executadas.

**Passagem de parâmetros por valor**

Os parâmetros de uma função são variáveis dessa função. Assim, os valores passados na chamada na função são atribuídos aos parâmetros correspondentes. Essa operação é conhecida como *passagem de parâmetros por valor*, pois ocorre a cópia de um valor para um parâmetro.

Há situações em que deseja-se que uma função possa retornar mais de um valor à função chamadora. Em outras palavras, deseja-se alterar o conteúdo de uma variável passada por parâmetro dentro da função. Na linguagem C, essa operação é implementada por meio de *Ponteiros*, que será estudado nas próximas aulas.

**Variáveis globais e variáveis locais**

Variáveis globais são variáveis que podem ser acessadas diretamente por qualquer função do programa. Na linguagem C, variáveis globais são declaradas logo após as diretivas (#include e #define), antes da declaração das funções. A Figura 1 mostra o local onde são declaradas as variáveis globais.

```
#include <stdio.h>

int numero;

/* Declaração de Funções */
…

int main(void) {
	…
}
```

**Figura 1.** Declaração de uma variável global.

As variáveis que pertencem a uma função são consideradas variáveis locais. Elas podem ser acessadas apenas pela função que a declarou. As variáveis de uma função podem ser declaradas dentro do corpo da função ou na declaração da função (protótipo). Assim, os parâmetros de uma função são variáveis locais dessa função.

**Exemplo**: Creditar (adicionar) um valor em uma conta corrente.

```
#include <stdio.h>

float conta;

int creditar_conta (float valor) {

	if (valor > 0) {
		conta = conta + valor;
		return 1;
	}

	return 0;
}

int main(void) {
	float valor;

	/* Inicializa a conta corrente */
	conta = 100.0;

	printf(“Entre com o valor:”);
	scanf(“%f”, &valor);

	if (creditar_conta(valor) == 1) {
		printf(“O valor foi creditado na conta corrente.\n”);
	}
	eles {
		printf(“Credito nao realizado.\n”);
	}

	printf(“Saldo atual = %f\n”, conta);
	return 0;
}
```

**Código 4**: Programa que credita valores em uma conta.

**2) Exercícios**

- Implementar e testar os códigos 1, 2, 3 e 4.

**3) Atividade Avaliativa 4**

1. Faça uma função em C chamada *calcula_imc()* que recebe como parâmetros a altura e o peso de uma pessoa e retorna o índice de massa corporal de acordo com a fórmula:

$$IMC = peso \div altura^2$$

O programa principal deve imprimir a classificação da pessoa de acordo com o IMC calculado:

IMC < 18.5 – abaixo do peso ideal
18.5 <= IMC < 25 – peso ideal
25 <= IMC < 30 – sobrepeso
IMC >= 30 – obesidade

2. Faça um programa em C para simular movimentações em uma conta bancária. Duas operações são permitidas: *creditar* e *debitar*. A operação *creditar* permite adicionar um valor à uma conta corrente, enquanto a operação *debitar* permite subtrair um valor de uma conta corrente.

**Requisitos**

a) O programa deverá exibir para o usuário as operações disponíveis, conforme a Figura 1.

b) O programa deve permitir a realização iterativa das operações, enquanto não for digitado 3 para Sair. A opção Sair deve encerrar a execução do programa.

c) Deverão ser exibidas mensagens informando o estado das operações, por exemplo: O valor foi debitado da conta corrente" e "O valor foi creditado na conta corrente". Possíveis falhas nas operações também devem ser informadas ao usuário.

d) O programa deverá informar o saldo atual da conta corrente logo após a realização das operações.

e) Escreva uma função para realizar cada operação.

| |
|---|
| Banco Eletrônico<br>1 – Creditar<br>2 – Debitar<br>3 – Sair |

**Figura 1**. Opções disponíveis no banco eletrônico.