Disciplina: Linguagem de Programação II
Professora: Caroline Mazetto Mendes

Aula 6 - 14/03/17

1) Funções (cont.)

Passagem de parâmetros por referência (Deitel 2011, cap 7.4)

As funções possibilitam a reutilização de trechos de código em várias partes de um programa. Para isso, as funções devem ser independentes, produzindo resultados a partir dos parâmetros de entrada. O uso de variáveis globais deve ser evitado, pois torna as funções dependentes destas variáveis.

A passagem de parâmetros por referência permite alterar (atualizar) o conteúdo de uma variável dentro de uma função sem o uso de variáveis globais. Na linguagem C, essa operação é implementada por meio de *Ponteiros.*

Ponteiros (Deitel 2011, cap 7)

Uma ponteiro é um tipo especial de variável que armazena endereços de memória. O valor armazenado pode ser o endereço de memória de uma outra variável. Assim, quando um ponteiro contém um determinado endereço, dizemos que ele aponta para este endereço (variável).

Para declarar um ponteiro, usa-se * antes do nome da variável, por exemplo:

int *pi;

Nesse exemplo, a variável *ptr* pode armazenar endereços de memória que contém valores do tipo inteiro. Em outras palavras, a variável ptr "aponta" para variáveis do tipo inteiro. Pode-se declarar ponteiros para diferentes tipo de dados, como ponteiros para char, ponteiros para float, entre outros.

Para o ponteiro armazenar o endereço de memória de uma variável, usa-se & antes do nome desta variável:

```
int *ptr;
int num;

ptr = &num;
printf("O valor da variável ptr eh %d\n", ptr);
```

Nesse exemplo, a variável ponteiro *ptr* armazena o endereço de memória da variável *num.* Ao imprimir o valor da variável *ptr*, será exibido um número inteiro que representa o endereço de memória da variável *num*.

Para acessar o valor de uma variável endereçada por um ponteiro, usa-se o operador dereferência * antes do nome da variável ponteiro:

```
int *ptr;
int num = 10;

ptr = &num;
printf("O valor da variável apontada por ptr eh %d\n", *ptr);
```

Nesse exemplo, será impresso o valor da variável apontada por *ptr (num)*.

Ponteiro como parâmetro de funções

Para alterar o conteúdo de uma variável passada por parâmetro para uma função, é necessário passar o endereço da variável (e não o seu valor). O seguintes passos permitem produzir a passagem de parâmetros por deferência:

1. Na chamada da função, deve-se usar o operador & antecedendo o nome da variável passada por parâmetro;
2. No protótipo da função, deve-se declarar o parâmetro correspondente como um ponteiro;
3. Dentro da função, deve-se usar o operador de derreferência * antecedendo o nome do parâmetro para alterar o conteúdo da variável apontada pelo parâmetro.

Exemplo: Creditar um valor na conta corrente e na conta poupança.

```
#include <stdio.h>

int creditar_conta (float *conta, float valor) {

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

        return 0;
}

int main(void) {
        float conta_corrente;
        float conta_poupanca;
        float valor;

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

        /* Inicializa a conta poupança */
        conta_poupanca = 100.0;

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

        /* Credita o valor na conta corrente */
        if (creditar_conta(&conta_corrente, valor) == 1) {
                printf("O valor foi creditado na conta corrente.\n");
        }
        eles {
                printf("Credito nao realizado.\n");
        }
```

```
        /* Credita o valor na conta poupança*/
        if (creditar_conta(&conta_poupanca, valor) == 1) {
                printf("O valor foi creditado na conta poupanca.\n");
        }
        else {
                printf("Credito nao realizado.\n");
        }

        printf("Saldo atual da conta corrente = %f\n", conta_corrente);
        printf("Saldo atual da conta poupança = %f\n", conta_poupanca);

        return 0;
}
```

Código 1: Programa que credita valores em uma conta corrente e conta poupança.

Esse exemplo exemplifica a **reutilização de código**. A função *creditar_conta()* realiza a operação de crédito para qualquer conta. A sua funcionalidade é creditar um valor em uma conta, seja qual for o tipo dessa conta (corrente ou poupança).

Primeiro, foram declaradas as variáveis do tipo float *conta_corrente* e *conta_poupanca*. Na chamada da função *creditar_conta()*, foi passado o endereço de memória da conta em questão utilizando o operador &. No protótipo da função, o parâmetro conta foi declarado como ponteiro, indicado pelo * antes do nome do parâmetro. Por fim, o operador dereferência * antecedendo a variável *conta* permite o acesso e modificação da variável passada por parâmetro (conta_corrente ou conta_poupança).

2) Exercícios

- Faça um programa que declare um ponteiro; faça esse ponteiro armazenar o endereço de outra variável; imprima o valor do ponteiro; e imprima o valor da variável apontada pelo ponteiro.
- Implementar e testar o Código 1.

3) Atividade Avaliativa 5

1. Considere o programa para realizar movimentações bancárias proposto na Atividade Avaliativa 4. Faça uma nova versão desse programa parar creditar/debitar valores em conta corrente e conta poupança. Após a escolha da operação, permita ao usuário escolher em qual conta deseja realizar a operação (1 – conta corrente; 2 – conta poupança). Use passagem de parâmetros por referência.

2. Considere o programa para simular uma votação eletrônica proposto na Atividade Avaliativa 3. Faça uma nova versão desse programa usando funções e passagem de parâmetros por referência.

3. Considere o programa para controlar o estoque de uma loja proposto na Atividade Avaliativa 3. Faça uma nova versão desse programa usando funções e passagem de parâmetros por referência.

Orientações:

(1) A nota da atividade está condicionada à apresentação dos programas.
(2) Faça os programas seguindo as dicas apresentadas na próxima página.

Dicas:

Para modularizar um programa, considere os trechos de código que realizam operações bem definidas e que podem sofrer futuras expansões:

Creditar/debitar uma conta

- Pode-se criar uma função para realizar a operação de crédito e outra função para realizar a operação de débito.
- Motivo: Cada uma das operações realiza uma tarefa bem definida. Assim, as funções poderão ser chamadas em qualquer parte do programa.

Imprimir saldos

- Pode-se criar uma função apenas para imprimir o saldo de todas as contas.
- Motivo: Se forem adicionadas novas contas? Essa operação vai começar a consumir muitas linhas de código, então é melhor isolar esta operação em uma função.

Imprimir candidatos

- Pode-se criar uma função apenas para exibir o nome e número dos candidatos que participam da votação.
- Motivo: Se forem adicionados novos candidatos à votação? Essa operação vai começar a consumir muitas linhas de código, então é melhor isolar esta operação em uma função.

Imprimir o resultado da votação

- Pode-se criar uma função apenas para imprimir o resultado da votação.
- Motivo: Se for preciso imprimir novas informações relacionadas ao resultado? Por exemplo, se vai ter segundo turno ou não, entre outras informações. Essa operação vai começar a consumir muitas linhas de código, então é melhor isolar esta operação em uma função.

Registrar um voto

- Pode-se criar uma função apenas para registrar um voto.
- Motivo: Esta operação realiza uma tarefa bem definida. Além disso, se forem adicionados novos candidatos à votação? Essa operação vai começar a consumir muitas linhas de código, então é melhor isolar esta operação em uma função.

Registrar uma venda

- Pode-se criar uma função apenas para registrar uma venda.
- Motivo: Se forem adicionados novos produtos em uma loja? Essa operação vai começar a consumir muitas linhas de código, então é melhor isolar esta operação em uma função.

Imprimir estoques

- Pode-se criar uma função apenas para exibir a quantidade em estoque de todos os produtos da loja.
- Motivo: Se forem adicionados novos produtos em uma loja? Essa operação vai começar a consumir muitas linhas de código, então é melhor isolar esta operação em uma função.

Imprimir relatório de vendas

- Pode-se criar uma função apenas para imprimir o relatório de vendas.
- Motivo: Se for preciso imprimir novas informações sobre as vendas? Por exemplo, qual produto esgotou antes do estoque. Essa operação vai começar a consumir muitas linhas de código, então é melhor isolar esta operação em uma função.

Para permitir a reutilização de código, observe os trechos que realizam operações similares. Em geral, o que muda nessas operações é a manipulação de uma ou outra variável, por exemplo:

Creditar/debitar uma conta

- As operações de crédito ou débito podem ser realizadas em uma conta corrente ou conta poupança.
- Por quê criar uma função para cada tipo de conta se uma única função pode realizar a operação usando passagem de parâmetros por referência?
- E se forem adicionados novos tipos de conta no banco?

Inicializar um estoque

- A operação pode ser realizada para o produto A ou para um produto B.
- Por quê criar uma função para inicializar o estoque de cada produto se uma única função pode realizar essa operação usando passagem de parâmetros por referência?
- E se forem adicionados novos produtos na loja?

Atualizar um estoque

- A operação (subtrair o estoque) pode ser realizada para o produto A ou para um produto B.
- Por quê criar uma função para atualizar o estoque de cada produto se uma única função pode realizar essa operação usando passagem de parâmetros por referência?
- E se forem adicionados novos produtos na loja?