Aula Prática III

Prazo: 2 semanas

Pedro O.S. Vaz de Melo

April 3, 2017

1 EXERCÍCIO

Neste exercício, você deve implementar duas funções para trocar o valor de duas variáveis. Na primeira função, `troca1`, os valores serão trocados dentro da função e não se alterarão no programa principal. No entanto, na função `troca2`, através do uso de ponteiros será possível trocar os valores das variáveis dentro da função e também no programa principal.

1.1 TEORIA

Nesta prática veremos a primeira utilidade clara de ponteiros, ou seja, de variáveis que podem armazenar endereços de memória. Como vimos na sala de aula, ponteiros são declarados da seguinte maneira:

```
int *ponteiro_para_inteiro;
float *ponteiro_para_float;
//etc...
```

Para acessar o conteúdo de um endereço de memória, basta usar o operador * em qualquer variável de tipo ponteiro. Além disso, para descobrir e retornar o endereço de uma variável basta utilizar o operador &. Exemplos:

```
int conta_corrente = 1234;
//ponteiro para inteiro recebe o endereço da variavel conta_corrente
int *p = &conta_corrente;
printf("O endereco da variavel conta_corrente eh: %p \n", &conta_corrente);
printf("O conteudo do endereco armazenado pelo ponteiro p eh: %d \n", *p);
```

```
//podemos tambem alterar o valor da variavel apontada pelo ponteiro
*p = *p + 100;
printf("O endereco da variavel conta_corrente eh: %p \n", &conta_corrente);
//conta corrente = 1334
```

Note que o operador `*` é usando para declarar uma variável do tipo ponteiro **e** para acessar o conteúdo de um endereço de memória, que pode estar armazenado em um ponteiro. Para mais informações sobre este operador, consulte os slides 20 ao 41 da Aula 4 publicada no site da disciplina (www.dcc.ufmg.br/ olmo/AEDS1.html).

2 CRIAÇÃO DE UM MÓDULO

Criar um módulo `modtroca` com duas funções, uma de nome `troca1` e outra de nome `troca2`. Você deve criar um arquivo `modtroca.h` com o cabeçalho das funções, um arquivo `modtroca.c` com a implementação das funções. Feito isso, você deve gerar um programa objeto `modtroca.o`, que deve ser ligado na compilação arquivo `.c` que contém o `main` (exercício seguinte).

2.1 FUNÇÃO TROCA1

Implementar uma função de nome `troca1` que tem como parâmetros dois inteiros, `valor1` e `valor2`, e não retorna nada. Essa função deve trocar os valores dos seus dois parâmetros de entrada, `valor1` e `valor2`, e imprimir os seus valores trocados. Exemplo: se a função receber como parâmetros `valor1=32` e `valor2=99`, então a função deve imprimir a mensagem `fim da função: valor1=99 e valor2=32` ao final da sua execução (**ainda dentro da função**).

2.2 FUNÇÃO TROCA2

Implementar uma função de nome `troca2` que tem como parâmetros duas variáveis capazes de armazenar endereços de memória de inteiros (que tipo de variável é capaz de fazer isso?), `end_valor1` e `end_valor2`. Essa função deve trocar o conteúdo dos endereços armazenados nessas variáveis, ou seja, o conteúdo armazenado pelo primeiro parâmetro deve ser armazenado no endereço do segundo parâmetro e vice-versa. Assim como na função anterior, essa função deve imprimir o conteúdo dos endereços de `end_valor1` e `end_valor2` ao final da sua execução.

3 PROGRAMA PRINCIPAL

Implemente um programa para usar e testar as funções do módulo criado. Para isso, neste programa, crie duas variáveis inteiras: `x=1` e `y=100`. Depois disso, faça as seguintes operações:

1. Chame a função `troca1` passando as variáveis `x` e `y` como parâmetros e nesta ordem.
2. Imprima na tela os valores de `x` e `y`.
3. Chame a função `troca2` passando as variáveis `x` e `y` como parâmetros e nesta ordem. Note que a função `troca2` não recebe inteiros como parâmetros, mas endereços de memórias de inteiros. Então, o que devo fazer?
4. Imprima na tela os valores de `x` e `y`.

4 MAIS UM SOBRE PONTEIROS

Escreva um **procedimento** de nome `aumentaOsIguais` que recebe como parâmetro dois endereços de memória de variáveis inteiras `end_var1` e `end_var2`. A função deve verificar se esses endereços de memória têm o mesmo valor inteiro armazenado neles. Caso negativo, a função deve subtrair 1 de ambos conteúdos dos endereços de memória. Caso positivo, a função deve fazer a soma dos dois valores e armazenar essa soma em ambos endereços de memória.

5 MANIPULAÇÃO DE BITS

Neste exercício você deve usar operações de deslocamento de bits « e/ou ». Primeiro, faça uma função de nome `fast_pow_2` que recebe um inteiro `expoente` e retorna um `unsigned long long` contendo a potência de dois correspondente. Um número do tipo `unsigned long long` é um inteiro sem sinal de 64 bits. Você o trata exatamente como um `unsigned int`, mas os seu alcance é muito maior. Depois, faça um programa para responder a seguinte pergunta: qual é o maior número que um `unsigned long long` pode representar? Faça um programa que use a função `fast_pow_2` para imprimir esse limite. Para imprimir um `unsigned long long` você deve usar o especificador de formato `%llu`. Protótipo da função `fast_pow_2`:

```
unsigned long long fast_pow_2(int expoente);
```

6 FUNÇÕES SIMPLES SEM OPERADORES CONDICIONAIS

Implemente funções para realizar as operações abaixo sobre parâmetros recebidos como números inteiros sem sinal (`unsigned int`). Suas funções **não devem** usar condicionais (`if`). Dica: algumas delas podem requerer operações bit-a-bit. Abaixo um exemplo de uma função que retorna o negativo do parâmetro:

```
int neg(unsigned int number) {
   return -number;
}
```

DDD. Extrair código de área de números de telefone com 8 dígitos (e.g., para o telefone 3134095858 a sua função deve retorna 31). Protótipo:

```
int ddd(unsigned int number);
```

SOMA 1 SE FOR PAR. Transformar um número par no próximo número ímpar e manter um número ímpar inalterado (e.g., para o número 4 a sua função deve retornar 5 e para o número 5 a sua função deve retornar 5).Protótipo:

```
int soma1SePar(unsigned int number);
```

PAR OU ÍMPAR Retornar verdadeiro se o número for par ou falso caso contrário. Dica: lembre dos conceitos de verdadeiro e falso para a linguagem C. Protótipo:

```
int parOuImpar(unsigned int number);
```

7 PROGRAMA PRINCIPAL

Faça um programa para testar as funções `aumentaOsIguais`, `fast_pow_2`, `ddd`, `soma1SePar` e `parOuImpar`.