Universidade Federal do Espírito Santo
Centro Tecnológico
Departamento de Informática

Programação Básica de Computadores
(INF 09325) – 2013/1
Prof. Vítor E. Silva Souza

Simulado da Prova Parcial 2 (01/08/2013)

A prova consiste na elaboração de algoritmos para resolução dos problemas descritos a seguir. Você deve usar a linguagem C para escrever os algoritmos. Seus algoritmos devem estar corretos (sintática e semanticamente), além de organizados (nomes de variáveis, identação do código, etc.) e bem escritos. Não é necessário preocupar-se em construir interfaces amigáveis com o usuário.

Questão 1
(2 pontos)

Escreva um programa que leia uma matriz de reais A de tamanho 75 x 75 e imprima seus valores na tela somente se a matriz for simétrica, ou seja, se ela satisfaz a condição:

$$\forall i,j: 0 < i \leq 75, 0 < j \leq 75: A(i,j) = A(j,i)$$

Ao imprimir a matriz, seu programa deve imprimir os números com no máximo 2 casas decimais, cada linha da matriz em uma linha da tela e os elementos da mesma linha separados por um caractere de tabulação: `'\t'`.

Questão 2
(3 pontos)

Escreva um programa que leia um número inteiro N e verifique se o mesmo é par. Caso não seja, solicite ao usuário que escolha um outro valor para N. Em seguida, o programa deve ler N números inteiros positivos, armazenando-os em um vetor A.

Então, gere um vetor B de N/2 elementos que contenha o máximo divisor comum (MDC) entre elementos de A, dois a dois, sendo que na primeira posição de B deve estar o MDC entre o primeiro e último elementos de A, na segunda posição deve estar o MDC entre o segundo e penúltimo elementos de A e assim por diante.

Ao final, faça um laço que imprima os pares de números de A que são primos entre si (aqueles cujo MDC é igual a 1). Organize o código utilizando sub-rotinas para facilitar seu entendimento por outros programadores.

Questão 3
(5 pontos)

Escreva as seguintes sub-rotinas:

- `carregarCaixa`: recebe um tipo de cédula (R$ 100, R$ 50, R$ 20, etc.) e uma quantidade e grava no estado interno do caixa eletrônico que o mesmo foi recarregado com a determinada quantidade do determinado tipo de cédula;
- `verificarSaque`: recebe uma quantia que se deseja sacar do caixa eletrônico e retorna verdadeiro (`1`) se é possível sacar aquela quantia dadas as cédulas presentes no caixa, ou falso (`0`) se não é possível sacar aquela quantia;
- `efetuarSaque`: recebe uma quantia que se deseja sacar e retorna 0 se não é possível sacar aquela quantia ou o mesmo valor da quantia se o saque for possível. Quando o saque é possível, modificar o estado interno do caixa eletrônico de maneira apropriada (reduzindo a quantidade das cédulas sacadas).

Então, escreva um programa que utilize as sub-rotinas acima para responder aos seguintes comandos do usuário:

- `c100 <qtd>`: carrega o caixa com <qtd> notas de R$ 100;
- `c50`, `c20`, `c10` e `c5`: idem acima;
- `saque <quantia>`: efetua, se possível, o saque de <quantia> em R$. Se não for possível, imprima "Desculpe, não há cédulas disponíveis para este saque";
- `estado`: imprime o estado do caixa (quantas cédulas de cada nota);
- `sair`: sai do programa.

Finalmente, observe o seguinte:

- Não é permitido utilizar variáveis globais (passe as estruturas que forem necessárias como parâmetros às sub-rotinas, usando ponteiros quando adequado);
- Assuma que o cliente que está sacando o dinheiro sempre tem saldo para efetuar o saque (não é necessário fazer esse controle);
- Caso o usuário digite um comando inválido, seu programa não precisa responder (pode ignorar o comando e aguardar o próximo comando);
- Crie outras sub-rotinas (além das listadas acima) se achar necessário.

Simulado da Prova Parcial 2 – Correção

Questão 1

```
float matriz[TAM_MATRIZ][TAM_MATRIZ];
int i, j, simetrica;

for (i = 0; i < TAM_MATRIZ; i++)
    for (j = 0; j < TAM_MATRIZ; j++)
        scanf("%f", &matriz[i][j]);

simetrica = 1;
for (i = 0; i < TAM_MATRIZ; i++)
    for (j = i + 1; j < TAM_MATRIZ; j++)
        if (matriz[i][j] != matriz[j][i]) simetrica = 0;

if (simetrica)
    for (i = 0; i < TAM_MATRIZ; i++) {
        for (j = 0; j < TAM_MATRIZ; j++)
            printf("%.2f\t", matriz[i][j]);
        printf("\n");
    }
```

Questão 2

```
int lerN() {
    int n;
    scanf("%d", &n);

    while (n % 2 != 0) scanf("%d", &n);
    return n;
}

int *lerVetor(int n) {
    int i;
    int *vetor = (int *)malloc(n * sizeof(int));

    for (i = 0; i < n; i++) scanf("%d", &vetor[i]);
    return vetor;
}

int mdc(int x, int y) {
    int tmp;
    while (x != y) {
        if (x > y)
            x = x - y;
        else {
            tmp = x;
            x = y;
            y = tmp;
```

```
            }
        }
        return x;
}

int *gerarVetorMDC(int n, int *vetor) {
        int i;
        int *vetorMDC = (int *)malloc(n / 2 * sizeof(int));

        for (i = 0; i < n / 2; i++)
            vetorMDC[i] = mdc(vetor[i], vetor[n - i - 1]);
        return vetorMDC;
}

main() {
        int i, n, *vetorA, *vetorB;

        n = lerN();
        vetorA = lerVetor(n);
        vetorB = gerarVetorMDC(n, vetorA);

        for (i = 0; i < n / 2; i++)
            if (vetorB[i] == 1)
                printf("%d e %d\n", vetorA[i], vetorA[n - i - 1]);
}
```

Questão 3

```
void carregarCaixa(int *valores, int *quantidades, int tamanho,
        int valorCedula, int qtd) {

    int i, idx = -1;
    for (i = 0; i < tamanho && idx == -1; i++)
        if (valores[i] == valorCedula) idx = i;
    if (idx != -1) quantidades[idx] += qtd;
}

int quantidadeCedulas(int valorCedula, int qtdCedulas,
        int quantia) {

    int qtd = 0;
    while ((quantia >= valorCedula) && (qtdCedulas > 0)) {
        quantia -= valorCedula;
        qtdCedulas--;
        qtd++;
    }
    return qtd;
}

int verificarSaque(int *valores, int *quantidades, int tamanho,
        int quantia) {
```

```
    int i, qtd;

    for (i = 0; quantia > 0 && i < tamanho; i++) {
        qtd = quantidadeCedulas(valores[i], quantidades[i],
                quantia);

        quantia -= qtd * valores[i];
    }

    return (quantia == 0);
}

int efetuarSaque(int *valores, int *quantidades, int tamanho,
        int quantia) {

    int i, valor, qtd, quantiaOriginal = quantia;
    if (! verificarSaque(valores, quantidades, tamanho, quantia))
        return 0;

    for (i = 0; quantia > 0 && i < tamanho; i++) {
        qtd = quantidadeCedulas(valores[i], quantidades[i],
            quantia);

        quantia -= qtd * valores[i];
        quantidades[i] -= qtd;
    }
    return quantiaOriginal;
}

void imprimirEstado(int *valores, int *quantidades, int tamanho) {
    int i;
    for (i = 0; i < tamanho; i++)
        printf("%d de R$ %d; ", quantidades[i], valores[i]);
    printf("\n");
}

void questao3() {
    int valores[] = {100, 50, 20, 10, 5};
    int quantidades[] = {0, 0, 0, 0, 0};
    char comando[10];
    int param;

    scanf("%s", comando);
    while (strcmp(comando, "sair") != 0) {
        if (strcmp(comando, "c100") == 0) {
            scanf("%d", &param);
            carregarCaixa(valores, quantidades, 5, 100, param);
        }
        else if (strcmp(comando, "c50") == 0) {
            scanf("%d", &param);
            carregarCaixa(valores, quantidades, 5, 50, param);
        }
```

```
        else if (strcmp(comando, "c20") == 0) {
            scanf("%d", &param);
            carregarCaixa(valores, quantidades, 5, 20, param);
        }
        else if (strcmp(comando, "c10") == 0) {
            scanf("%d", &param);
            carregarCaixa(valores, quantidades, 5, 10, param);
        }
        else if (strcmp(comando, "c5") == 0) {
            scanf("%d", &param);
            carregarCaixa(valores, quantidades, 5, 5, param);
        }
        else if (strcmp(comando, "saque") == 0) {
            scanf("%d", &param);
            if (! efetuarSaque(valores, quantidades, 5, param))
                printf("Desculpe, não há cédulas disponíveis
                        para este saque.\n");

        }
        else if (strcmp(comando, "estado") == 0) {
            imprimirEstado(valores, quantidades, 5);
        }

        scanf("%s", comando);
    }
}
```