Universidade Tecnológica Federal do Paraná
Departamento Acadêmico de Informática
CSF13 – Fundamentos de Programação 1
Prof. Bogdan Tomoyuki Nassu
Profa. Leyza Baldo Dorini

1.

```
/* Solução 1: O problema aqui é achar uma maneira de contemplar todos os casos
  possíveis. Os números poderiam ser mostrados um a um, mas a estrutura não
  seria muito mais simples que isso. Este exercício já adianta uma visão de como
  o problema fica mais complicado conforme o número de itens aumenta: para n
  itens, existem n! permutações. Seria inviável tratar os casos individualmente
  para um número arbitrário de itens, do jeito que aparece abaixo. A solução
  para o caso geral envolveria um tipo de dados mais complexo do que as
  variáveis que temos usado. Uma solução com lógica mais sofisticada para o caso
  com 3 valores já apareceu em uma lista anterior, mas ela modifica os valores
  das variáveis. */

#include <stdio.h>

int main ()
{
    int n1, n2, n3;

    scanf ("%d %d %d", &n1, &n2, &n3);

    if ((n1 < n2) && (n2 < n3))
        printf ("%d, %d, %d\n", n1, n2, n3);
    else if ((n1 < n3) && (n3 < n2))
        printf ("%d, %d, %d\n", n1, n3, n2);
    else if ((n2 < n1) && (n1 < n3))
        printf ("%d, %d, %d\n", n2, n1, n3);
    else if ((n2 < n3) && (n3 < n1))
        printf ("%d, %d, %d\n", n2, n3, n1);
    else if ((n3 < n1) && (n1 < n2))
        printf ("%d, %d, %d\n", n3, n1, n2);
    else if ((n3 < n2) && (n2 < n1))
        printf ("%d, %d, %d\n", n3, n2, n1);

    return (0);
}
```

```c
/* Solução 2: explorando condicionais aninhadas, fica mais fácil compreender que
   caso cada trecho do programa está trabalhando. Se o erro for para o caso em
   que n1 é menor, podemos focar apenas em um pequeno trecho do problema. */

#include <stdio.h>

int main ()
{
    int n1, n2, n3;

    scanf ("%d %d %d", &n1, &n2, &n3);

    if (n1 < n2 && n1 < n3) // Se n1 é o menor...
    {
        if (n2 < n3) // Compara n2 e n3.
            printf("%d, %d, %d\n", n1, n2, n3);
        else
            printf("%d, %d, %d\n", n1, n3, n2);
    }
    else if (n2 < n1 && n2 < n3) // Se n2 é o menor...
    {
        if (n1 < n3)
            printf("%d, %d, %d\n", n2, n1, n3);
        else
            printf("%d, %d, %d\n", n2, n3, n1);
    }
    else // Neste caso, n3 é o menor.
    {
        if (n1 < n2)
            printf ("%d, %d, %d\n", n3, n1, n2);
        else
            printf ("%d, %d, %d\n", n3, n2, n1);
    }

    return (0);
}
```

2.

```
#include <stdio.h>
#include <math.h>

int main ()
{
    int dia1, mes1, ano1; // Primeira data.
    int dia2, mes2, ano2; // Segunda data.
    int anos;

    scanf ("%d/%d/%d", &dia1, &mes1, &ano1);
    scanf ("%d/%d/%d", &dia2, &mes2, &ano2);

    // Somente olhar a diferença de anos não é o suficiente!
    //recisamos ver também se o aniversário já passou!
    if (mes1 > mes2 || (mes1 == mes2 && dia1 > dia2))
        anos = ano2 - ano1 - 1;
    else
        anos = ano2 - ano1;

    printf ("%d anos\n", anos);

    return 0;
}
```

3.

```
#include <stdio.h>

int main()
{
    float dist;
    int pontos = 0; // Inicia com um número inválido de pontos.

    scanf ("%f", &dist);

    /* Note aqui que não preciso testar os intervalos inteiros.
       Você consegue ver o porquê? */
    if (dist <= 800)
        pontos = 1;
    else if (dist <= 1400)
        pontos = 2;
    else if (dist <= 2000)
        pontos = 3;

    /* Poderia colocar o printf dentro dos ifs, mas usar uma variável facilita
       se tivéssemos mudanças, como mostrar outra mensagem ou usar esta
       pontuação para algum propósito posterior. */
    if (pontos > 0)
        printf ("%d\n", pontos);

    return 0;
}
```

4.

```
/* Problema clássico, e muito parecido com aquele das horas. A maior diferença
   aqui é que precisamos quebrar o número inicial em muitos sub-pedaços, então
   fazer todo o cálculo para cada tipo de nota em uma única linha é um pouco
   inconveniente. Em vez disso, podemos ir subtraindo o dinheiro que já "temos"
   do valor total - este é o processo que muita gente usa mentalmente quando faz
   este tipo de coisa na vida real. A solução genérica para um número arbitrário
   de sub-pedaços com pesos variados seria mais complexa, e exigiria uma
   estrutura de dados mais sofisticada, que ainda não vimos. */

#include <stdio.h>

int main ()
{
    int n, n100, n50, n20, n10, n5, n2, n1;

    scanf ("%d", &n);

    n100 = n / 100;
    n = n - (n100 * 100);
    n50 = n / 50;
    n = n - (n50 * 50);
    n20 =  n / 20;
    n = n - (n20 * 20);
    n10 = n / 10;
    n = n - (n10 * 10);
    n5 = n / 5;
    n = n - (n5 * 5);
    n2 = n / 2;
    n = n - (n2 * 2);
    n1 = n;

    // Note que eu posso pular uma linha depois da string sem problemas.
    printf ("100: %d\n50: %d\n20: %d\n10: %d\n5: %d\n2: %d\n1: %d\n",
      n100, n50, n20, n10, n5, n2, n1);
    return (0);
}
```

5.

```
#include <stdio.h>

int main ()
{
    float l1, l2, l3;

    scanf ("%f %f %f", &l1, &l2, &l3);

    /* O comprimento de um lado do triângulo é sempre menor do que a soma dos
       outros dois. */
    if ((l1 < l2 + l3) && (l2 < l1 + l3) && (l3 < l1 + l2))
    {
        if (l1 == l2 && l2 == l3)
            printf ("Equilatero\n"); // 3 lados iguais.
        else if ((l1 == l2) || (l1 == l3) || (l2 == l3))
            printf ("Isoceles\n"); // 2 lados iguais.
        else
            printf ("Escaleno\n");
    }
    else
        printf ("Nao eh triangulo.\n");

    return (0);
}
```

6.

```
#include <stdio.h>

int main ()
{
    int n; // O número da conta.
    int n_invertido; // O número da conta, com os dígitos invertidos.
    int centenas, dezenas, unidades; // Nomes mais mnemônicos que c, d e u.
    int soma_n_n_invertido, final, digito_verificador;

    scanf ("%d", &n);

    // Inverte os dígitos de n.
    centenas = n/100;
    dezenas = (n%100)/10;
    unidades = n%10;
    n_invertido = centenas + dezenas*10 + unidades*100;

    // Soma n com o n invertido.
    soma_n_n_invertido = n + n_invertido;

    /* Separa de novo os dígitos. Se o resultado for > 999, desconsidera
       o milhar.*/
    soma_n_n_invertido = soma_n_n_invertido % 1000;
    centenas = soma_n_n_invertido/100;
    dezenas = (soma_n_n_invertido %100)/10;
    unidades = soma_n_n_invertido%10;

    /* Multiplica pela ordem posicional e soma tudo. Note que não precisamos
       realmente multiplicar a centena por 1, porque esta multiplicação não
       teria efeito! */
    final = centenas + (dezenas*2) + (unidades*3);

    // Pegamos o último dígito.
    digito_verificador = final % 10;

    printf ("%d\n", digito_verificador);

    return (0);
}
```