Lista 7 (EXERCÍCIOS)

1) Escreva um programa que calcule a sequência de Fibonacci em tempo de compilação.

2) Utilizando templates, escreva uma função imprime que receba dois números e imprima a soma na tela. Considere o caso do template especializado para valores booleanos tal que a soma seja equivalente ao **OU** lógico.

3) Deduza a saída do código abaixo:

```
#include <iostream>

using namespace std;

template <typename T>
void fun(const T & x)
{
    static int contador = 0;
    cout << "x = " << x << " contador = " << contador << endl;
    ++contador;
    return;
}

int main()
{
    fun<int> (1);
    cout << endl;
    fun<int>(1);
    cout << endl;
    fun<double>(1.1);
    cout << endl;
    return 0;
}
```

4) Deduza a saída do código abaixo:

```
#include <iostream>
using namespace std;

template <class T>
class Teste
{
private:
    T val;
public:
    static unsigned int contador;
    Teste()  {   contador++;   }
};
```

```
template <class T>
unsigned int Teste<T>::contador = 0;

int main()
{
    Teste<int> a;
    Teste<int> b;
    Teste<double> c;
    cout << Teste<int>::contador   << endl;
    cout << Teste<double>::contador << endl;
    return 0;
}
```

5) Estude o seguinte código:

```
#include <iostream>

using std::cout;
using std::endl;

template <typename T>
T max(T x, T y)
{
    return (x > y)? x : y;
}
int main()
{
    cout << max(3, 7) << endl;
    cout << max(3.0, 7.0) << endl;
    cout << max(3, 7.0) << endl;
    return 0;
}
```

Considerando o código acima:

a) Corrija o erro produzido pelo terceiro cout sem alterar o conteúdo de main, para isto, altere a função parametrizada.
b) Reescreva o código sem utilizar templates.

6) Considere o código abaixo:

```
void tarefa(bool escolha)
{
      for (int i = 0; i < 1000000; i++)
      {
        if(escolha)
            isso();
        else
            aquilo();
      }
}
```

```
tarefa(true);
tarefa(false);
```

Esse código realiza uma mesma tarefa um milhão de vezes, entretanto, percebe-se que o valor de escolha é fixo antes da iteração. Este código é extremamente ineficiente pois é produzida uma ramificação (o `if`) a qual é executada um milhão de vezes desnecessariamente, aumentando a quantidade de instruções e reduzindo a eficiência do mecanismo de *pipeline* do processador.

Re-escreva este código utilizando templates para que o mesmo fique mais eficiente.

7) Considere a função abaixo que a realiza a troca de conteúdo de duas variáveis:

```
void troca(int &a, int & b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```

Caso um usuário deseje utilizá-la para diversos tipos é necessário reescrevê-la para cada tipo. Implemente essa função utilizando template.

8) Re-escreva o código da lista encadeada visto em aulas anteriores utilizando templates. Teste a lista para tipos inteiros, doubles e strings.

9) Responda como templates permitem que um único trecho de código seja reutilizado para diferentes tipos. Exemplifique.

10) Templates são um exemplo de qual tipo de polimorfismo?

11) Pesquise pelo problema de *two-phase name look up*. Explique o erro do código abaixo e o motivo pelo qual as duas soluções funcionam.

```
// Super classe
template <class T>
class Vader
```

```
{
public:
    T iamyourfather;
};

// Subclasse
template <class T>
class Luke : public Vader<T>
{
public:
    T thatsimpossible;

    void nooooo()
    {
        // nao pode ser resolvido
        // precisa de this-> ou Pai<T>::
        cout << iamyourfather << endl;

        // pode ser resolvido
        cout << thatsimpossible << endl;
    }
};
```

Dica: Procure pelas seguintes entradas no C++ FAQ:

[35.19] *Why am I getting errors when my template-derived-class accesses something it inherited from its template-base-class?*

[35.20] *Can the previous problem hurt me silently? Is it possible that the compiler will silently generate the wrong code?*

12) O código abaixo apresenta um erro na definição de pvalor. Corrija este erro e explique.

```
template <class T>
class A
{
public:
    typedef unsigned int uint;
};

template <class T>
class B
{
public:
    B<T>::uint *pvalor;
};
```

DICA: Pesquise pela entrada [35.18] no C++ FAQ.

[35.18] *Why am I getting errors when my template-derived-class uses a nested type it inherits from its template-base-class?*