Linguagens de Programação
Departamento de Engenharia Eletrônica e de Computação - DEL-Poli/UFRJ
Prof.: Miguel Elias Mitre Campista

== Laboratório 2 ==

1. **Escreva um programa em C++ para calcular o n-ésimo elemento da série de Fibonacci. O programa deve possuir um arquivo principal que solicita o índice n do elemento ao usuário e depois exibe na tela o resultado. Para isso, é necessária uma função para calcular o n-ésimo elemento que recebe a ordem e retorna o elemento calculado. O protótipo dessa função deve ser disponibilizado no arquivo `fibonacci-sc.h` e o código fonte da função, no arquivo `fibonacci-sc.cpp`. Não há o uso de classes.**

2. **Repetir o exemplo anterior criando a classe Fibonacci. A função `computeFibonacci`, entretanto, recebe o índice da série, mas não retorna o resultado. Assim, é necessária a criação de uma função de acesso `private` que calcule o elemento da série. O resultado é armazenado como um atributo privado da classe que só é acessado por meio de uma função do tipo `get`.**

3. **Escreva uma classe Agenda para receber e armazenar 3 nomes distintos. Os três nomes devem ser passados pelo usuário através da execução do método `setNames`, que interage com o usuário solicitando cada um dos nomes. Os três nomes devem ser armazenados em um atributo privado do tipo array de strings. Antes de inserir os nomes no array, um método para validação deve ser usado. Nesse teste, assegura-se que cada um dos nomes não tenha mais do que 10 caracteres. Caso possua, uma mensagem deve ser exibida ao usuário e o nome deve ser truncado para caber no limite máximo definido. A classe Agenda ainda deve oferecer um método público para que os usuários possam visualizar os nomes inseridos. Use os métodos `substr` (`string substr (size_t pos = 0, size_t n = npos) const;`) e `length` (`int length ()`) definidas na biblioteca `string`.**

4. **Escreva um programa que calcule o volume de um paralelepípedo reto. Para isso, o usuário deverá passar as dimensões do paralelepípedo para um objeto da classe `Paralelepipedo` através do construtor da classe. Esse construtor inicializa as dimensões do paralelepípedo com argumentos passados pelo usuário e calcula o volume invocando o método `setVolume`. Esse método pode ser também usado pelo usuário, caso ele queira alterar as dimensões. A classe `Paralelepípedo` ainda oferece um método do tipo get para recuperar o volume calculado. Assim, é necessária a definição de atributos privados que armazenem o valor de cada uma das dimensões do paralelepípedo e outro para armazenar o volume.**

5. **Escreva um programa para calcular a distância de dois pontos no espaço. O programa deve oferecer a classe Distancia e a classe Ponto. A primeira classe possui como atributos privados dois objetos da classe Ponto e um double para armazenar a distância entre os pontos. Ainda, a classe Distancia possui métodos set e get para atribuir e recuperar as coordenadas dos pontos, um método get para recuperar a distância entre os pontos e um método set para calcular a distância que só precisa ser invocada de dentro da própria classe Distancia. Por fim, a classe Distancia ainda oferece um construtor que inicializa o atributo**

distância com zero. A classe Ponto possui métodos do tipo set e get para cada uma das coordenadas. Além disso, a classe Ponto ainda oferece um construtor que inicializa o valor de todas as dimensões (x, y, z) com 0, que é definido na origem do espaço.

== Respostas ==

1.

```
/*************************************************************************/
/************************** Programa Principal ***************************/

#include <iostream>
#include "fibonacci-sc.h"

using namespace std;

int main () {
    int idx;

    cout << "Entre com o índice da série de Fibonacci: ";
    cin >> idx;

    cout << "\n\nO número eh: " << computeFibonacci (idx) << endl;

    return 0;
}

/*************************************************************************/
/************************** Arquivo fibonacci-sc.h ***********************/

int computeFibonacci (int);

/*************************************************************************/
/************************** Arquivo fibonacc-sc.cpp **********************/

#include "fibonacci-sc.h"

int computeFibonacci (int i) {
    if (i == 1)
            return 1;
    else if (i == 2)
            return 1;
    else
            return computeFibonacci (i - 1) + computeFibonacci (i - 2);
}

/*************************************************************************/
```

2.

```
/*************************************************************************/
/************************** Programa Principal ***************************/

#include "fibonacci-cc.h"

using namespace std;

int main () {
    Fibonacci fib;
    int idx;

    cout << "Entre com o índice da série de Fibonacci: ";
    cin >> idx;

    fib.computeFibonacci (idx);

    cout << "\n\nO número eh: " << fib.getFibonacci () << endl;
```

```
    return 0;
}

/*****************************************************************************/
/************************** Arquivo fibonacci-cc.h **************************/

#include <iostream>

class Fibonacci {
    public:
            void computeFibonacci (int);
            int getFibonacci ();
    private:
            int result;

            int setResult (int);
};

/*****************************************************************************/
/************************** Arquivo fibonacc-cc.cpp *************************/

#include "fibonacci-cc.h"

void Fibonacci::computeFibonacci (int i) {
    result = setResult (i);
}

int Fibonacci::getFibonacci () {
    return result;
}

int Fibonacci::setResult (int i) {
    if (i == 1)
            return 1;
    else if (i == 2)
            return 1;
    else
            return setResult (i - 1) + setResult (i - 2);
}

/*****************************************************************************/
```

3.

```
/*****************************************************************************/
/************************** Programa Principal ******************************/

#include <iostream>
#include <string>
#include "agenda.h"
usung namespace std;

int main () {
    Agenda agenda;
    agenda.setNames ();
    agenda.getNames ();
    return 0;
}

/*****************************************************************************/
/************************** Arquivo agenda.h ********************************/

#include <iostream>
#include <string>

using namespace std;

class Agenda {
    public:
    void setNames ();
    void getNames ();

    private:
            string names [3];
```

```
            string checkName (string);
};

/*****************************************************************************/
/************************* Arquivo agenda.cpp *************************/

#include "agenda.h"

void Agenda::setNames () {
    string name;

    for (int i = 0; i < 3; i++) {
            cout << "Entre com o novo nome: " << endl;
            getline (cin, name);
            names [i] = checkName (name);
    }
}

void Agenda::getNames () {
    for (int i = 0; i < 3; i++)
            cout << names [i] << endl;
}

string Agenda::checkName (string name) {
    if (name.length () > 10) {
            name = name.substr (0, 10);
            cout << "Nome truncado para: " << name << endl;
    }
    return name;
}

/*****************************************************************************/
```

4.

```
/*****************************************************************************/
/************************** Programa Principal **************************/

#include <iostream>
#include "paralelepipedo.h"

using namespace std;

int main () {
    Paralelepipedo p (1, 1, 1);
    cout << p.getVolume () << endl;
    p.setVolume (1, 2, 3);
    cout << p.getVolume () << endl;

    return 0;
}

/*****************************************************************************/
/************************* Arquivo paralelepipedo.h *************************/

#include <iostream>
#include <string>

using namespace std;

class Paralelepipedo {
    public:
    Paralelepipedo (double, double, double);
    void setVolume (double, double, double);
    double getVolume ();

    private:
            double altura, largura, comprimento;
            double volume;
};

/*****************************************************************************/
/************************ Arquivo paralelepipedo.cpp ************************/

#include <iostream>
```

```
#include <string>
#include "paralelepipedo.h"

using namespace std;

Paralelepipedo::Paralelepipedo (double a, double l, double c) {
    setVolume (a, l, c);
}

void Paralelepipedo::setVolume (double a, double l, double c) {
    altura = a;
    largura = l;
    comprimento = c;

    volume = altura*largura*comprimento;
}

double Paralelepipedo::getVolume () {
    return volume;
}

/*****************************************************************************/
```

5.

```
/*****************************************************************************/
/************************** Programa Principal ***************************/

#include <iostream>
#include "distancia.h"

using namespace std;

int main () {
    Distancia distancia;
    distancia.getPontos ();
    distancia.setPontos (1, 0, 0, 3, 0, 0);
    distancia.getPontos ();
    cout << distancia.getDistancia ();
    return 0;
}

/*****************************************************************************/
/************************** Arquivo ponto.h ******************************/

#include <iostream>

using namespace std;

class Ponto {
    public:
    Ponto ();

    void setX (double)
    void setY (double);
    void setZ (double);

    double getX ();
    double getY ();
    double getZ ();

    private:
    double coord_x, coord_y, coord_z;
};

/*****************************************************************************/
/************************** Arquivo ponto.cpp ****************************/

#include "ponto.h"

Ponto::Ponto () {
    setX (0); setY (0); setZ (0);
}
```

```
void Ponto::setX (double x) {coord_x = x;}
void Ponto::setY (double y) {coord_y = y;}
void Ponto::setZ (double z) {coord_z = z;}

double Ponto::getX () {return coord_x;}
double Ponto::getY () {return coord_y;}
double Ponto::getZ () {return coord_z;}

/*************************************************************************/
/************************* Arquivo distancia.h ************************/

#include <iostream>
#include <cmath>

using namespace std;

class Distancia {
    public:
    Distancia ();

    void setPontos (double, double, double, double, double, double);
    void getPontos ();
    double getDistancia ();

    private:
    Ponto pontoA, pontoB;
    double distancia;

    void setDistancia ();
};

/*************************************************************************/
/************************* Arquivo distancia.cpp ************************/

#include "distancia.h"

Distancia::Distancia () {distancia = 0;}

void Distancia::setPontos (double xA, double yA, double zA,
                           double xB, double yB, double zB) {
    pontoA.setX (xA); pontoA.setY (yA); pontoA.setZ (zA);
    pontoB.setX (xB); pontoB.setY (yB); pontoB.setZ (zB);
}

void Distancia::getPontos () {
    cout << "Ponto A ("
           << pontoA.getX () << ","
           << pontoA.getY () << ","
           << pontoA.getZ ()
           << ")" << endl;
    cout << "Ponto B ("
           << pontoB.getX () << ","
           << pontoB.getY () << ","
           << pontoB.getZ ()
           << ")" << endl;
}

double Distancia::getDistancia () {
    setDistancia ();
    return distancia;
}

void Distancia::setDistancia () {
    distancia = sqrt (pow (pontoA.getX ()-pontoB.getX (), 2) +
                      pow (pontoA.getY ()-pontoB.getY (), 2) +
                      pow (pontoA.getZ ()-pontoB.getZ (), 2));
}

/*************************************************************************/
```