

1) **Lista de exercícios:** Resolva os exercícios abaixo como se pede.

- a) Escreva um programa que utilize um objeto da classe `Agenda`. Para isso, divida o programa em três arquivos: `main.cpp`, `agenda.h` e `agenda.cpp`. A classe `Agenda` possui um objeto da classe `vector` como atributo privado, que é utilizado para armazenar `N` nomes inseridos pelos usuários. A classe `Agenda` deve então oferecer um método para inserção de nomes, chamado `insereNome`. Note que o método `insereNome` insere um nome por vez na agenda, sabendo que cada nome deve ter no máximo 10 caracteres. Dessa forma, caso o nome possua mais que 10 caracteres, uma mensagem deve ser exibida ao usuário e o nome deve ser truncado para caber no limite máximo definido. Todos os nomes inseridos na agenda devem ser exibidos na tela através da execução do método `mostraNomes`, também definido na classe `Agenda`.

Para truncar os nomes, utilize o método `substr` da classe `string`, como visto a seguir:

```
string substr (size_t pos = 0, size_t n = npos) const;
```

O número `N` de nomes deve ser passado para o construtor da classe `Agenda` como um argumento padrão. Esse valor é usado para inicializar o atributo privado que controla o número máximo de nomes na agenda. Use o método `push_back` para inserir novos nomes no `vector` privado e o método `size` para saber o número de nomes no `vector`. Esses dois métodos pertencem à classe `vector`.

- b) Reescreva o programa anterior, Questão 1(a), substituindo o `vector` privado por um `array` de `strings` privado. Utilize um atributo constante estático, chamado `maxNomes`, para definir o tamanho máximo do `array` e um atributo estático, chamado `numeroNomes`, para controlar o número de elementos já inseridos no `array`.

Com o uso do atributo `maxNomes`, veja se ainda é possível receber o número máximo de elementos do `array` pelo construtor. Ainda, com o uso do atributo `numeroNomes`, veja se é possível saber o número de nomes armazenados sem o uso do objeto da classe `Agenda`.

- c) Reescreva novamente o programa da Questão 1(b), substituindo o `array` de `strings` por um `array` de objetos da classe `Cadastro`. A classe `Cadastro` possui como atributos privados o nome, a profissão e a idade de cada uma das pessoas registradas. Além dos atributos, a classe `Cadastro` possui métodos do tipo “get” e “set” para cada um dos seus atributos. Realize a verificação do tamanho máximo do nome no método `set` correspondente.

O método `insereNome` da classe `Agenda` é modificado para `insereCadastro`, já que recebe uma referência a um objeto da classe `Cadastro`, no lugar de apenas um nome. Tal objeto da classe `Cadastro` foi previamente inicializado através de seus próprios métodos `set`. O método `mostraNomes` foi também alterado para exibir na tela todos os atributos de um objeto da classe `Cadastro`, dando lugar ao método `mostraCadastros`.

d) Reescreva o programa da Questão 1(c), substituindo o `array` de objetos da classe `Cadastro` por um `vector` de objetos da classe `Cadastro`. Realize todas as alterações necessárias para que o programa funcione como o da questão anterior.

2) Programa para entrega dia 13/08/2021: A entrega do programa será através do Google Classroom e consiste da devolução de todos os arquivos referentes ao código-fonte, um Makefile e um arquivo README que explique brevemente a compilação e a utilização do programa. Todos os arquivos serão avaliados. Esta atividade é individual.

O programa deverá apresentar dados referentes ao avanço da COVID-19 no Brasil e nos estados individualmente. Para isso, um menu contendo as seguintes opções deve ser disponibilizado:

1. Exibir a evolução do número de óbitos no Brasil, e nos estados individualmente, usando a estratégia de média móvel dos últimos N dias. A média móvel (MM) deverá ser calculada como se segue:

$$MM = \frac{1}{N} \sum_{i=0}^{N-1} d_{a-i},$$

onde d_a é o dia atual, d_{a-1} é o dia anterior ao atual e assim por diante. Calcule a média móvel ao menos para os últimos três dias, ou seja, usando $N=3$. Assumir que o programa tem acesso a dados referentes ao número de óbitos diários por estado por um intervalo maior que $N+1$ e que N é no máximo igual a 7;

2. Exibir de forma agrupada os estados em alta, estabilidade e baixa segundo a razão entre a média móvel como calculada no dia atual (MM calculada no item 1) e a média móvel como calculada no dia anterior (para calcular a média móvel do dia anterior faça: $d_a \leftarrow d_{a-1}$). Utilize como limiares de definição que estados com aumento superior a 10% são considerados em alta, redução superior a 10% em baixa e, caso contrário, estabilidade;
3. Repetir o mesmo dado do item 2, se em alta, baixa ou estabilidade, para o Brasil;
4. Exibir o estado com maior alta e maior baixa segundo a média móvel do dia atual calculada no item 1;
5. Exibir os dados acumulados de número de óbitos (soma desde o início do período de observação) no Brasil e nos estados individualmente.

O programa deve prever a implementação de uma classe `Nacional` e `Estadual`, na qual a classe `Nacional` é composta por M objetos da classe `Estadual`. Cada objeto da classe `Estadual` deve representar um estado brasileiro e, como tal, deve gerenciar os seus dados individuais. Note que os dados da série histórica podem ser arbitrários, ou seja, não necessariamente precisam representar a realidade. Os dados da série podem ser inicializados junto com os objetos de cada estado, sem necessidade de inserção ou remoção de dados diários através de opção do menu.

Dica: Verifique a possibilidade do uso de um array ou de um objeto da classe `vector` para armazenamento dos objetos da classe `Estadual` e para armazenamento dos dados da série histórica pelos estados.

== Respostas da Lista de Exercícios

1)

a)

```

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

#include <iostream>
#include <string>

#include "agenda.h"

/* Programa do Laboratório 3:
   Uma agenda...
   Autor: Miguel Campista */

using namespace std;

int main () {
    Agenda agenda;
    string nome;

    cout << "Entre com três nomes: " << endl;
    cout << endl;

    for (int i = 0; i < 3; i++) {
        cout << "Nome [" << i << "]: ";
        getline (cin, nome);
        agenda.inseraNome (nome);
    }

    // Mostrar os nomes
    cout << "\nOs nomes da agenda são:" << endl;
    agenda.mostraNomes ();

    return 0;
}
/*****
/***** Arquivo agenda.h *****/
#include <iostream>
#include <string>
#include <vector>

using namespace std;

class Agenda {
public:
    Agenda (int = 3);

    void inseraNome (string);
    void mostraNomes ();

private:
    vector <string> v;
    int maxNomes;

    string verificaNome (string);
};
/*****
/***** Arquivo agenda.cpp *****/

#include "agenda.h"

Agenda::Agenda (int n) {
    maxNomes = n;
}

void Agenda::inserirNome (string str) {
    str = verificaNome (str);
}
```

```

        if (v.size () > maxNomes)
            cout << "Agenda cheia!" << endl;
        else
            v.push_back (str);
    }

    void Agenda::mostraNomes () {
        for (int i = 0; i < v.size (); i++) {
            cout << "Nome [" << i << "] "
                << v.at (i) << endl;
        }
    }

    string Agenda::verificaNome (string str) {
        if (str.length () > 10) {
            cout << "Nome com mais de 10 caracteres, reduzindo para 10...";
            cout << endl;
            // Trunca o nome com método substr
            str = str.substr (0, 10);
        }

        return str;
    }
}
/*****

```

b)

```

/*****
/***** Programa Principal *****/
#include <iostream>
#include <string>

#include "agenda.h"

/* Programa do Laboratório 3:
   Uma agenda... agora com array privado
   Autor: Miguel Campista */

using namespace std;

int main () {
    Agenda agenda;
    string nome;

    cout << "Entre com três nomes: " << endl;
    cout << endl;

    for (int i = 0; i < 3; i++) {
        cout << "Nome [" << i << "]: ";
        getline (cin, nome);
        agenda.inseraNome (nome);
    }

    // Mostrar os nomes
    cout << "\nA agenda possui "
        << Agenda::numeroNomes // chamada sem objeto
        << " nomes...\nOs nomes da agenda são:"
        << endl;
    agenda.mostraNomes ();

    return 0;
}

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

#include <iostream>
#include <string>

using namespace std;

class Agenda {
public:
    void inseraNome (string);
    void mostraNomes ();

```

```

        static int numeroNomes;

    private:
        const static int maxNomes = 3;

        string nomes [maxNomes];

        string verificaNome (string);
};

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

#include "agenda.h"

int Agenda::numeroNomes = 0;

void Agenda::insereNome (string str) {
    str = verificaNome (str);

    if (numeroNomes >= maxNomes)
        cout << "Agenda cheia!" << endl;
    else
        nomes [numeroNomes++] = str;
}

void Agenda::mostraNomes () {
    for (int i = 0; i < numeroNomes; i++) {
        cout << "Nome [" << i << "] "
              << nomes [i] << endl;
    }
}

string Agenda::verificaNome (string str) {
    if (str.length () > 10) {
        cout << "Nome com mais de 10 caracteres, reduzindo para 10...";
        cout << endl;
        // Trunca o nome com método substr
        str = str.substr (0, 10);
    }

    return str;
}
/*****/

```

c)

```

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

#include <iostream>
#include <string>

#include "agenda.h"

/* Programa do Laboratório 3:
   Uma agenda... agora com array de objetos
   Autor: Miguel Campista */

using namespace std;

int main () {
    Agenda agenda;
    string nome, profissao;
    int idade;

    cout << "Entre com três cadastros: " << endl;
    cout << endl;

    for (int i = 0; i < 3; i++) {
        Cadastro cadastro;

        cout << "Nome [" << i << "]: ";
        getline (cin, nome);
        cadastro.setNome (nome);

        cout << "Profissao [" << i << "]: ";
    }
}

```

```

        getline (cin, profissao);
        cadastro.setProfissao (profissao);

        cout << "Idade [" << i << "]: ";
        cin >> idade;
        cin.ignore ();
        cadastro.setIdade (idade);

        agenda.insereCadastro (cadastro);

        if (i < 2) cout << "--- proximo ---" << endl;
    }

    // Mostrar os atributos dos cadastros
    cout << "\nOs cadastros da agenda são:" << endl;
    agenda.mostraCadastros ();

    return 0;
}

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

#include <iostream>
#include <string>

#include "cadastro.h"

using namespace std;

class Agenda {
public:
    void insereCadastro (Cadastro &c);
    void mostraCadastros ();

    static int numeroNomes;

private:
    const static int maxNomes = 3;

    Cadastro nomes [maxNomes];
};

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

#include "agenda.h"

int Agenda::numeroNomes = 0;

void Agenda::insereCadastro (Cadastro &c) {
    if (numeroNomes >= maxNomes)
        cout << "Agenda cheia!" << endl;
    else
        nomes [numeroNomes++] = c;
}

void Agenda::mostraCadastros () {
    for (int i = 0; i < numeroNomes; i++) {
        cout << "Nome [" << i << "] " << nomes [i].getNome () << endl
            << "Prof [" << i << "] " << nomes [i].getProfissao () << endl
            << "Idade [" << i << "] " << nomes [i].getIdade () << endl;
    }
}

/*****
**** Arquivo cadastro.h *****/

#include <iostream>
#include <string>
#include <vector>

using namespace std;

class Cadastro {
public:
    void setNome (string);
    string getNome ();

```

```

        void setProfissao (string);
        string getProfissao ();

        void setIdade (int);
        int getIdade ();

    private:
        string nome, profissao;
        int idade;

        string verificaNome (string);
};

/*****
**** Arquivo cadastro.cpp *****/

#include "cadastro.h"

void Cadastro::setNome (string n) {
    nome = verificaNome (n);
}

string Cadastro::getNome () {
    return nome;
}

void Cadastro::setProfissao (string p) {
    profissao = p;
}

string Cadastro::getProfissao () {
    return profissao;
}

void Cadastro::setIdade (int a) {
    idade = a;
}

int Cadastro::getIdade () {
    return idade;
}

string Cadastro::verificaNome (string str) {
    if (str.length () > 10) {
        cout << "Nome com mais de 10 caracteres, reduzindo para 10...";
        cout << endl;

        str = str.substr (0, 10);
    }

    return str;
}

/*****/

```

d)

```

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

#include <iostream>
#include <string>

#include "agenda.h"

/* Programa do Laboratório 3:
   Uma agenda... agora com vector de objetos
   Autor: Miguel Campista */

using namespace std;

int main () {
    Agenda agenda;
    string nome, profissao;
    int idade;
}

```

```

        cout << "Entre com três cadastros: " << endl;
        cout << endl;

        for (int i = 0; i < 3; i++) {
            Cadastro cadastro;

            cout << "Nome [" << i << "]: ";
            getline (cin, nome);
            cadastro.setNome (nome);

            cout << "Profissao [" << i << "]: ";
            getline (cin, profissao);
            cadastro.setProfissao (profissao);

            cout << "Idade [" << i << "]: ";
            cin >> idade;
            cin.ignore ();
            cadastro.setIdade (idade);

            agenda.insereCadastro (cadastro);

            if (i < 2) cout << "--- proximo ---" << endl;
        }

        // Mostrar os atributos dos cadastros
        cout << "\nOs cadastros da agenda são:" << endl;
        agenda.mostraCadastros ();

        return 0;
    }

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

#include <iostream>
#include <string>
#include <vector>

#include "cadastro.h"

using namespace std;

class Agenda {
public:
    void insereCadastro (Cadastro &);
    void mostraCadastros ();

private:
    const static int maxNomes = 3;
    // Vector para armazenamento de cadastros
    vector <Cadastro> v;
};

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

#include "agenda.h"

void Agenda::insereCadastro (Cadastro &c) {
    if (v.size () > maxNomes)
        cout << "Agenda cheia!" << endl;
    else
        v.push_back (c);
}

void Agenda::mostraCadastros () {
    for (int i = 0; i < v.size (); i++) {
        cout << "Nome [" << i << "] " << v [i].getNome () << endl
            << "Prof [" << i << "] " << v [i].getProfissao () << endl
            << "Idade [" << i << "] " << v [i].getIdade () << endl;
    }
}

/*****
***** Arquivo cadastro.h *****/

#include <iostream>

```



```

#include <string>
#include <vector>

using namespace std;

class Cadastro {
public:
    void setNome (string);
    string getNome ();

    void setProfissao (string);
    string getProfissao ();

    void setIdade (int);
    int getIdade ();

private:
    string nome, profissao;
    int idade;

    string verificaNome (string);
};

/*****
*****/
Arquivo cadastro.cpp *****/

#include "cadastro.h"

void Cadastro::setNome (string n) {
    nome = verificaNome (n);
}

string Cadastro::getNome () {
    return nome;
}

void Cadastro::setProfissao (string p) {
    profissao = p;
}

string Cadastro::getProfissao () {
    return profissao;
}

void Cadastro::setIdade (int a) {
    idade = a;
}

int Cadastro::getIdade () {
    return idade;
}

string Cadastro::verificaNome (string str) {
    if (str.length () > 10) {
        cout << "Nome com mais de 10 caracteres, reduzindo para 10...";
        cout << endl;

        str = str.substr (0, 10);
    }

    return str;
}

/*****/

```