EEL670 - Linguagens de Programação
Departamento de Engenharia Eletrônica e de Computação - DEL-Poli/UFRJ
Prof.: Miguel Elias Mitre Campista
== Laboratório 9

---

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

a) Escreva um programa de um jogo de Forca. Para isso, implemente a classe `Forca`, cujo construtor lê palavras de um arquivo texto presente no mesmo diretório do programa. Ainda, o construtor inicializa um atributo que armazena o número máximo de rodadas possível do jogo. O número máximo de rodadas caracteriza o número máximo de testes de caracteres que podem ser realizados até que o jogador seja "enforcado".

O jogo é executado somente após a chamada ao método `configure`. Este método escolhe aleatoriamente uma das palavras lidas, para que esta represente o segredo do jogo. Além do segredo, o método `configure` imprime na tela uma dica sobre a palavra sorteada e o número total de caracteres que a mesma possui. Observe que a dica pode estar também presente no mesmo arquivo de palavras ou em um arquivo separado. Por exemplo, o arquivo de palavras pode conter os seguintes pares (palavra, dica):

```
banana, fruta
papagaio, animal
carro, veiculo
linguagens, disciplina
niteroi, cidade
```

Após a configuração do jogo, o programa segue para o teste de caracteres, realizado através do método `teste`. Este método não possui parâmetros. Ao invés disso, ele interage com o jogador para solicitar um caractere para, em seguida, verificar se este mesmo caractere é encontrado no segredo. Caso todos os caracteres tenham sido encontrados ou o número máximo de rodadas seja atingido, o método teste retorna `false`. Caso contrário, retorna `true`. Antes de cada teste, o programa indica na tela os caracteres existentes e os ainda ocultos além do número de testes restantes. Supondo, por exemplo, que o segredo seja "banana", que o caractere "b" já tenha sido testado e que o número de testes remanescentes seja 2, a seguinte mensagem é impressa na tela:

```
b _ _ _ _ _
Vc ainda tem 2 chances.
```

Caso o jogador teste o caractere 'a', todos os caracteres 'a' pertences ao segredo são considerados adivinhados. Nesse caso, a impressão na tela seria a seguinte:

```
b a _ a _ a
Vc ainda tem 1 chance.
```

Note que métodos set e get devem ser propostos para todos os atributos privados da classe `Forca`.

b) A questão anterior assume que todas as ocorrências de um caractere repetido são reveladas, caso o jogador o utilize para teste. Essa premissa pode ser alterada, fazendo que o jogador precise repetir o teste, caso um dado caractere pertença à palavra na forca. Dessa forma, usando o mesmo exemplo da questão anterior, o resultado seria:

```
b a _ _ _ _
Vc ainda tem 1 chance.
```

Note que apenas a primeira ocorrência do caractere ‘a’ foi revelada, o que deve ser o padrão em caso de caracteres repetidos. A alteração no teste obriga os jogadores a repetir caracteres já testados, dificultando a forca.

Além da modificação do método teste, pense também na inserção de um método que gere uma mensagem aleatória de congratulações caso o jogador vença a forca.

Utilize o conceito de herança na implementação do programa. A implementação de métodos adicionais na classe base é uma possibilidade, caso seja necessário. Mantenha o funcionamento geral do programa sob o ponto de vista do jogador.

c) Implemente a classe `Pilha` para armazenar somente inteiros. Para isso, é necessário criar pelo menos dois métodos públicos: um para empilhar (`bool push (int)`) e outro desempilhar (`bool pop (int &)`) inteiros. O método `push` recebe o inteiro a ser inserido na pilha e retorna o resultado da operação. De forma semelhante, o método `pop` recebe uma referência que será inicializada com o inteiro removido da pilha e retorna o resultado da operação. Note que o `push` pode retornar `false` caso a pilha esteja cheia e o `pop`, caso a pilha esteja vazia. Em qualquer outra situação, o retorno do `push` e do `pop` é `true`.

Sabendo que a classe `vector <int>` já oferece métodos que podem ser aproveitados para implementação da classe Pilha, use herança. Faça uma função principal que ilustre todas as operações possíveis com a pilha de inteiros criada.

== Respostas da Lista de Exercícios

1)

a)

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

#include "forca.h"

/* Programa do Laboratório 9:
   Programa de um jogo da forca

   Autor: Miguel Campista */

using namespace std;

int main() {
    Forca forca (5);

    forca.configure ();

    while (forca.teste());

    return 0;
}

/*************************************************************************/
/***************************** Arquivo forca.h ***************************/

#include <iostream>
#include <string>
#include <fstream>
#include <vector>
#include <cstdlib>
#include <ctime>

using namespace std;

#ifndef FORCA_H
#define FORCA_H

class Forca {
    public:
            Forca (int);

            void configure ();

            bool teste ();

            string getPalavra();
            string getDica();

            vector <char> &getSegredo ();

            vector <string> getPalavras();
            vector <string> getDicas();

            int &getCaracteresFaltantes ();
            int &getTestesRestantes ();
            int getMaxTestes();

            void setPalavra (string);
            void setDica (string);

            void setCaracteresFaltantes (int);

            void setSegredo(char);

            void imprimeSegredo ();
```

```
    private:
            string palavra, dica;
            vector <char> segredo;
            vector <string> palavras, dicas;

            int testesRestantes, maxTestes, caracteresFaltantes;

};

#endif

/*****************************************************************************/
/***************************** Arquivo forca.cpp *****************************/

#include "forca.h"

Forca::Forca (int n) : maxTestes(n), testesRestantes(n) {
    fstream file;
    file.open("palavras.txt", fstream::in);

    if (!file.is_open()) {
            cerr << "Arquivo não encontrado" << endl;
            exit (0);
    }

    while (file.good()) {
            string str;

            getline (file, str, ',');
            palavras.push_back(str);

            getline (file, str);
            dicas.push_back(str);
    }

    file.close();

    srand (time(0));
}

void Forca::configure () {
    int idx = rand() % getPalavras().size ();

    setPalavra (getPalavras().at (idx));
    setDica (getDicas().at (idx));
    setCaracteresFaltantes (getPalavra().length());

    cout << "A dica eh: " << getDica()
            << "\nA palavra tem " << getCaracteresFaltantes()
            << " caracteres" <<endl;

    setSegredo('_');

    cout << endl;
    system("pause");

    imprimeSegredo();
}

bool Forca::teste () {
    char in;
    bool achou = false;

    getTestesRestantes()--;
    if (getCaracteresFaltantes() == 0) {
            cout << "\nVoce encontrou o segredo... parabens!" << endl;
            return false;
    }
    if (getTestesRestantes() < 0) {
            cout << "\nPoxa... acabaram as chances..." << endl;
            cout << "A palavra era: " << getPalavra() << endl;
            return false;
    }

    if (getTestesRestantes() == getMaxTestes() - 1) cout << "\n\n\n";
    else cout << "\n";
```

```
    cout << "Vc ainda tem " << (getTestesRestantes() + 1)
         << ((getTestesRestantes() > 0) ? " chances" : " chance");
    cout << "\nQual caractere deseja: "; cin >> in; cin.ignore();

    for (unsigned i = 0; i < getSegredo().size(); i++) {
           if (in == getPalavra()[i]) {
                  getSegredo().at(i) = in;
                  achou = true;
                  getCaracteresFaltantes()--;
           }
    }

    imprimeSegredo ();

    cout << endl;
    cout << (achou ? "Letra encontrada!" : "Hum... Letra nao foi encontrada...")
            << endl;

    return true;
}

void Forca::imprimeSegredo () {
    system("cls");
    for (unsigned i = 0; i < segredo.size(); i++) {
           cout << segredo.at(i) << ' ';
    }
    cout << endl;
}

string Forca::getPalavra() {
    return palavra;
}
string Forca::getDica() {
    return dica;
}

vector <char> &Forca::getSegredo () {
    return segredo;
}

vector <string> Forca::getPalavras() {
    return palavras;
}
vector <string> Forca::getDicas() {
    return dicas;
}

int & Forca::getCaracteresFaltantes () {
    return caracteresFaltantes;
}

int & Forca::getTestesRestantes () {
    return testesRestantes;
}

int Forca::getMaxTestes() {
    return maxTestes;
}

void Forca::setPalavra (string str) {
    palavra = str;
}
void Forca::setDica (string str) {
    dica = str;
}
void Forca::setCaracteresFaltantes (int i) {
    caracteresFaltantes = i;
}

void Forca::setSegredo(char delim) {
    for (unsigned i = 0; i < palavra.length(); i++) {
           segredo.push_back(delim);
    }
}
/*****************************************************************************/
```

b)

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

#include "forca-derivada.h"

/* Programa do Laboratório 9:
   Programa de um jogo da forca

   Autor: Miguel Campista */

using namespace std;

int main() {
    ForcaDerivada forca (5);

    forca.configure ();

    while (forca.teste());

    return 0;
}

/*****************************************************************************/
/***************************** Arquivo forca.h ******************************/

#include <iostream>
#include <string>
#include <fstream>
#include <vector>
#include <cstdlib>
#include <ctime>

using namespace std;

#ifndef FORCA_H
#define FORCA_H

class Forca {
    public:
            Forca (int);

            void configure ();

            bool teste ();

            string getPalavra();
            string getDica();

            vector <char> &getSegredo ();

            vector <string> getPalavras();
            vector <string> getDicas();

            int &getCaracteresFaltantes ();
            int &getTestesRestantes ();
            int getMaxTestes();

            void setPalavra (string);
            void setDica (string);

            void setCaracteresFaltantes (int);

            void setSegredo(char);

            void imprimeSegredo ();

    private:
            string palavra, dica;
            vector <char> segredo;
            vector <string> palavras, dicas;

            int testesRestantes, maxTestes, caracteresFaltantes;
```

```
};

#endif

/*****************************************************************************/
/***************************** Arquivo forca.cpp *****************************/

#include "forca.h"

Forca::Forca (int n) : maxTestes(n), testesRestantes(n) {
    fstream file;
    file.open("palavras.txt", fstream::in);

    if (!file.is_open()) {
            cerr << "Arquivo não encontrado" << endl;
            exit (0);
    }

    while (file.good()) {
            string str;

            getline (file, str, ',');
            palavras.push_back(str);

            getline (file, str);
            dicas.push_back(str);
    }

    file.close();

    srand (time(0));
}

void Forca::configure () {
    int idx = rand() % getPalavras().size ();

    setPalavra (getPalavras().at (idx));
    setDica (getDicas().at (idx));
    setCaracteresFaltantes (getPalavra().length());

    cout << "A dica eh: " << getDica()
             << "\nA palavra tem " << getCaracteresFaltantes()
             << " caracteres" <<endl;

    setSegredo('_');

    cout << endl;
    system("pause");

    imprimeSegredo();
}

bool Forca::teste () {
    char in;
    bool achou = false;

    getTestesRestantes()--;
    if (getCaracteresFaltantes() == 0) {
            cout << "\nVoce encontrou o segredo... parabens!" << endl;
            return false;
    }
    if (getTestesRestantes() < 0) {
            cout << "\nPoxa... acabaram as chances..." << endl;
            cout << "A palavra era: " << getPalavra() << endl;
            return false;
    }

    if (getTestesRestantes() == getMaxTestes() - 1) cout << "\n\n\n";
    else cout << "\n";

    cout << "Vc ainda tem " << (getTestesRestantes() + 1)
         << ((getTestesRestantes() > 0) ? " chances" : " chance");
    cout << "\nQual caractere deseja: "; cin >> in; cin.ignore();

    for (unsigned i = 0; i < getSegredo().size(); i++) {
            if (in == getPalavra()[i]) {
```

```
                    getSegredo().at(i) = in;
                    achou = true;
                    getCaracteresFaltantes()--;
            }
    }

    imprimeSegredo ();

    cout << endl;
    cout << (achou ? "Letra encontrada!" : "Hum... Letra nao foi encontrada...")
         << endl;

    return true;
}

void Forca::imprimeSegredo () {
    system("cls");
    for (unsigned i = 0; i < segredo.size(); i++) {
            cout << segredo.at(i) << ' ';
    }
    cout << endl;
}

string Forca::getPalavra() {
    return palavra;
}
string Forca::getDica() {
    return dica;
}

vector <char> &Forca::getSegredo () {
    return segredo;
}

vector <string> Forca::getPalavras() {
    return palavras;
}
vector <string> Forca::getDicas() {
    return dicas;
}

int & Forca::getCaracteresFaltantes () {
    return caracteresFaltantes;
}

int & Forca::getTestesRestantes () {
    return testesRestantes;
}

int Forca::getMaxTestes() {
    return maxTestes;
}

void Forca::setPalavra (string str) {
    palavra = str;
}
void Forca::setDica (string str) {
    dica = str;
}
void Forca::setCaracteresFaltantes (int i) {
    caracteresFaltantes = i;
}

void Forca::setSegredo(char delim) {
    for (unsigned i = 0; i < palavra.length(); i++) {
            segredo.push_back(delim);
    }
}

/*****************************************************************************/
/************************** Arquivo forca-derivada.h *************************/

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

#include "forca.h"
```

```
using namespace std;

#ifndef FORCADERIVADA_H
#define FORCADERIVADA_H

class ForcaDerivada: public Forca {
    public:
            ForcaDerivada (int);

            bool teste ();

            string felicitacoes ();
};

#endif

/*****************************************************************************/
/*********************** Arquivo forca-derivada.cpp ************************/

#include "forca-derivada.h"

ForcaDerivada::ForcaDerivada (int n): Forca (n) {}

bool ForcaDerivada::teste () {
    char in;
    bool achou = false;

    getTestesRestantes()--;
    if (getCaracteresFaltantes() == 0) {
            cout << "\n" << felicitacoes() << endl;
            cout << "Voce encontrou o segredo... parabens!" << endl;
            return false;
    }
    if (getTestesRestantes() < 0) {
            cout << "\nPoxa... acabaram as chances..." << endl;
            cout << "A palavra era: " << getPalavra() << endl;
            return false;
    }

    if (getTestesRestantes() == getMaxTestes() - 1) cout << "\n\n\n";
    else cout << "\n";

    cout << "Vc ainda tem " << (getTestesRestantes() + 1)
         << ((getTestesRestantes() > 0) ? " chances" : " chance");
    cout << "\nQual caractere deseja: "; cin >> in; cin.ignore();

    for (unsigned i = 0; i < getSegredo().size(); i++) {
            if ((in == getPalavra()[i]) && (getSegredo().at(i) == '_')) {
                    getSegredo().at(i) = in;
                    achou = true;
                    getCaracteresFaltantes()--;
                    break;
            }
    }

    imprimeSegredo ();

    cout << endl;
    cout << (achou ? "Letra encontrada!" : "Hum... Letra nao foi encontrada...")
         << endl;

    return true;
}

string ForcaDerivada::felicitacoes() {
    vector <string> mensagens ({"Parabens!",
                                "Muito bem!",
                                "Ótimo jogo.",
                                "Excepcional!"});

    return mensagens.at(rand() % mensagens.size());
}

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

c)

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

#include "pilha.h"

/* Programa do Laboratório 9:
   Programa de uma Estrutura em Pilha

   Autor: Miguel Campista */

using namespace std;

int main() {
    Pilha pilha (5);
    int valor = 0;

    while (pilha.push (++valor)) {
            cout << "Valor: " << valor << " inserido" << endl;
    }

    cout << endl;

    while (pilha.pop (valor)) {
            cout << "Valor: " << valor << " removido" << endl;
    }

    return 0;
}

/*****************************************************************************/
/***************************** Arquivo pilha.h *******************************/

#include <iostream>
#include <vector>

using namespace std;

#ifndef PILHA_H
#define PILHA_H

class Pilha: public vector <int> {
    public:
            Pilha (int);

            bool push (int);
            bool pop (int &);

            bool cheia();
            bool vazia();

    private:
            int topo;
};

#endif

/*****************************************************************************/
/***************************** Arquivo pilha.cpp *****************************/

#include "pilha.h"

using namespace std;

Pilha::Pilha (int t): topo (-1), vector <int> (t) {}

bool Pilha::push(int i) {
    if (cheia())
            return false;

    at (++topo) = i;

    return true;
```

```
}

bool Pilha::pop(int &i) {
    if (vazia())
            return false;

    i = at (topo--);

    return true;
}

bool Pilha::cheia() {
    return topo == (size() - 1);
}

bool Pilha::vazia() {
    return topo == -1;
}

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