

Paradigmas de Programação I

Teste Prático 2D

Nome: _____ N° Mec.: _____

1. Considere a classe **CStr** destinada a manipular strings em C++. O objectivo desta classe é poder tratar strings como objectos de forma a simplificar a sua manipulação. Por exemplo, se forem instanciados dois objectos da classe **CStr**, uma funcionalidade interessante desta é permitir fazer a atribuição e a justaposição (concatenação) de strings da seguinte forma:

```
CStr s1("uma string"), s2("outra string");
s1 = s2;      s1 = s1 + s2;      s1 += s2;      s1 += "mais outra string";
```

O interface pretendido para a classe **CStr** é o seguinte:

```
class CStr
{
public:
    CStr(); //Constroi uma string vazia
    CStr(const char* pStr); //Constroi uma string c/o conteúdo do array de caracteres "pStr"
    CStr(const CStr& str); //Constroi uma cópia de "str"
    virtual ~CStr(); //Destroi a string

public:
    void Empty(); //Esvazia a string
    bool IsEmpty(); //Verifica se a string está vazia
    unsigned int Length(); //Devolve o tamanho da string

    bool operator==(const CStr& str); //Verifica se a string "*this" é igual à string "str"
    CStr& operator=(const CStr& str); //Atribui à string "*this" a string "str"
    CStr& operator+=(const CStr& str); //Justapõe a string "*this" com a string "str"
    CStr operator+(const CStr& str); //Devolve uma string resultante da justaposição
    //da string "*this" com a string "str"
    char operator[](unsigned int index); //Devolve o carácter que está na posição "index"
    //da string "*this"
    friend ostream& operator<<(ostream& os, const CStr& str); //Escreve "str" na stream "os"

    int Insert(unsigned int index, const CStr& str); //Insere a string "str" a partir da
    //posição "index" da string "*this"
    int Find(const CStr& str); //Devolve o índice da primeira ocorrência
    //da string "str" na string "*this"
    void MakeUpper(); //Transforma as letras minúsculas da
    //string "*this" em maiúsculas
    void MakeLower(); //Transforma as letras maiúsculas da
    //string "*this" em minúsculas
    CStr Left(unsigned int count); //Devolve uma string com os primeiros
    //"count" caracteres da string "*this"
    CStr Right(unsigned int count); //Devolve uma string com os últimos
    //"count" caracteres da string "*this"

protected:
    char* m_pStorage; //Ponteiro para o bloco de memória onde é armazenada a string
    //terminada com o carácter nulo ('\0')
    static char* m_pEmptyStr; //Ponteiro para uma string vazia (com uma única posição para o
    //terminador) para simplificar a manipulação de strings vazias
};
```

Considere também o seguinte excerto da implementação da classe **CStr**:

```
CStr::CStr(const char* pStr)
{
    m_pStorage = new char[strlen(pStr) + 1];
    if (m_pStorage != NULL)
    {
        strcpy(m_pStorage, pStr);
    }
    else
    {
        m_pStorage = m_pEmptyStr;
    }
}

CStr::~CStr()
{
    if (m_pStorage != m_pEmptyStr)
    {
        delete[] m_pStorage;
    }
}

//Este atributo não deve ser alterado
char* CStr::m_pEmptyStr = "";
```

Escreva a implementação dos seguintes elementos da classe **CStr**:

1.1 O construtor **CStr ()** ;

1.2 O construtor **CStr(const CStr& str)** ;

1.3 O método **void Empty ()** ;

1.4 O método **bool operator==(const CStr& str)** ;

1.5 O operador **CStr& operator=(const CStr& str)** ;

1.6 O operador **CStr& operator+=(const CStr& str)** ;

1.7 O operador **CStr operator+(const CStr& str);**

1.8 O operador **ostream& operator<<(ostream& os, const CStr& str);**

1.9 O método **int Find(const CStr& str);**

1.10 O método **void MakeUpper();**

1.11 O método **CStr Right(unsigned int count);**

2. Assumindo que a classe **CStr** do problema anterior foi completamente implementada, considere o seguinte programa:

```
void main()
{
    CStr s1, s2(""), s3("PP1"), s4(s3), s5("Turmas P3 e P4");

    if (s1.IsEmpty())
        cout << "s1 está vazia!" << endl;

    s1 = s3;
    cout << s1 << endl;

    s2 += s3;
    cout << s2 << endl;

    s2 += " Teste P2";
    cout << s2 << endl;

    if (s3 == s4)
        cout << "s3 == s4" << endl;

    cout << s4 + s5 << endl;

    s2.MakeLower();
    cout << s2 << endl;

    cout << "Tamanho de s3 = " << s3.Length() << endl;
    cout << "s3[2] = " << s3[2] << endl;

    cout << s3.Left(2) << endl;

    s3 = s3 + s4;
    s3.Insert(1, " - ");
    cout << s3 << endl;
}
```

2.1 Qual a saída produzida no ecrã pelo programa quando executado?

3. Considere o seguinte programa com erros de sintaxe:

```
void main()
{
    char c = 'x';
    char s[] = "abc";
    char* p;
    char &r = c;

    s = "def";
    p = &c;
    *p = 'i';
    s[0] = 'j';
    &c = p;
    p = s;
    p[2] = 'k';
    *(p + 1) = r;
    s = p;
    r = s[0];
    p = &r;

    cout << c << endl << s << endl;
    cout << *p << endl << endl << r << endl;
}
```

3.1 Identifique e sublinhe as linhas com erros.

3.2 Suponha que as linhas com erros de sintaxe eram retiradas do programa. Qual a saída produzida pelo programa no ecrã se este fosse compilado com sucesso e executado.