

Paradigmas de Programação I

Teste Prático 2A

Nome: _____ N° Mec.: _____

1. Considere a classe chamada CString para manipulação de um array de caracteres. Para resolver os exercícios propostos é-lhe permitido, sempre que achar necessário, usar apenas as seguintes funções da biblioteca string.h:

```
int strcmp( const char *string1, const char *string2 );
unsigned int strlen( const char *string );
char *strcpy( char *strDestination, const char *strSource );
```

```
class CString
{
public:
    CString();
    CString(const char* str);
    virtual ~CString();

    int Find(char c);
    void ToggleChar();
    void operator-(char c);
    int operator[](int index);

    char* m_pString;
};

CString::CString()
{
    m_pString = NULL;
}
```

Dado o construtor por defeito, implemente os seguintes métodos:

- 1.1. O construtor **CString(const char* str)** para copiar a string str para o atributo m_pString e o destrutor **~CString()**.

```
CString::CString( const char* str)                CString::~~CString()
{
    if(str)
    {
        m_pString = new char[strlen(str) + 1];
        if(m_pString)
            strcpy(m_pString, str);
    }
}
{
    delete[] m_pString;
}
```

- 1.2. O método **int Find(char c)** para devolver o índice da primeira ocorrência do caracter c. Caso o caracter não pertença à string, a função deve devolver -1.

```
int CString::Find( char c)
{
    if(!m_pString)
        return -1;
    for( unsigned int i = 0; i < strlen(m_pString); i++)
        if(m_pString[i] == c)
            return i;
    return -1;
}
```

1.3. O método `void ToggleChar()` para alterar todas as minúsculas para maiúsculas e vice-versa. Pode utilizar as funções `bool islower(char c)` e `bool isupper(char c)` que devolvem verdadeiro quando o caracter é minúsculo ou maiúsculo, respectivamente, e as funções `char tolower(char c)` e `char toupper(char c)` para devolver o caracter `c` convertido em minúscula ou maiúscula, respectivamente.

```
void CString::ToggleChar()
{
    if (!m_pString)
        return;
    for (unsigned int i = 0; i < strlen(m_pString); i++)
    {
        if (islower(m_pString[i]))
            m_pString[i] = toupper(m_pString[i]);
        if (isupper(m_pString[i]))
            m_pString[i] = tolower(m_pString[i]);
    }
}
```

1.4. O método `void operator-(char c)` para apagar da string todas as ocorrências do caracter `c`.

```
void CString::operator-(char c)
{
    if (!m_pString)
        return;
    int newSize = 0, j = 0;
    for (unsigned int i = 0; i < strlen(m_pString); i++)
        if (m_pString[i] != c)
            newSize++;
    char *str = new char[newSize + 1];
    if (!str)
        return;
    for (i = 0; i < strlen(m_pString); i++)
        if (m_pString[i] != c)
            str[j++] = m_pString[i];
    str[j] = '\0';
    delete[] m_pString;
    m_pString = new char[newSize + 1];
    if (!m_pString)
    {
        delete[] str;
        return;
    }
    strcpy(m_pString, str);
    delete[] str;
}
```

1.5. O método `int operator[](int index)` para devolver o caracter armazenado na string `m_pString` na posição `index`. Em caso de erro a função deverá devolver o valor `-1`.

```
int CString::operator[](int index)
{
    if (!m_pString)
        return -1;
    if ((index >= (int)strlen(m_pString)) || (index < 0))
        return -1;
    return m_pString[index];
}
```

2. Considere a lista ligada CLinkedList semelhante à que implementou nas aulas práticas. Dado o método InsertHead para inserção de um elemento à cabeça da lista, implemente um método, chamado InsertTail para inserir um elemento no final da lista.

```

class CLinkedList
{
...
public:
    CLinkedList();
    virtual ~CLinkedList();

public:
    int InsertHead(const void* pData,
                  unsigned int dataSize);
    int RemoveHead(void* pData,
                  unsigned int dataSize);
    int InsertTail(const void* pData,
                  unsigned int dataSize);
    const void* GetHead();
    const void* GetNext();
    unsigned int Count();
...
protected:
    unsigned int m_count;
    SNode* m_pHead;
    SNode* m_pCurrent;
};

int CLinkedList::InsertHead(const void* pData,
                           unsigned int dataSize)
{
    SNode* pNode;

    if (pData == NULL)
        return NULL_PTR;
    pNode = new SNode;
    if (pNode == NULL)
        return OUT_OF_MEMORY;
    pNode->m_pData = new char[dataSize];
    if (pNode->m_pData == NULL)
    {
        delete pNode;
        return OUT_OF_MEMORY;
    }
    memcpy(pNode->m_pData, pData, dataSize);
    pNode->m_pNext = m_pHead;
    m_pHead = pNode;
    ++m_count;
    return OK;
}

int CLinkedList::InsertTail(const void* pData, unsigned int dataSize)
{
    if (!m_pHead)
        return InsertHead(pData, dataSize);
    if (!pData)
        return NULL_PTR;
    SNode *pNode = new SNode;
    if (!pNode)
        return OUT_OF_MEMORY;
    pNode->m_pData = new char[dataSize];
    if (!(pNode->m_pData))
    {
        delete pNode;
        return OUT_OF_MEMORY;
    }
    memcpy(pNode->m_pData, pData, dataSize);

    for (SNode* p = m_pHead; p->m_pNext != NULL; p = p->m_pNext);
    p->m_pNext = pNode;
    pNode->m_pNext = NULL;

    ++m_count;
    return OK;
}

```

3. Dado o programa, responda às seguintes questões:

```
int main(int argc, char* argv[])
{
    CString str1, str2("nome"), str3("ha,ha!");
    CLinkedList list;

    int error1 = list.InsertHead(str1.m_pString, strlen(str1.m_pString)+1);
    int error2 = list.InsertHead(str2.m_pString, strlen(str2.m_pString)+1);

    cout << list.Count() << endl;
    cout << (char*)list.GetHead() << endl;
    cout << ((char*)list.GetHead())[0] << endl;
    cout << *(char*)list.GetHead() << endl;
    cout << *((char*)list.GetHead()+2) << endl;

    list.InsertTail(str3.m_pString, strlen(str3.m_pString)+1);
    list.RemoveHead(str2.m_pString, strlen(str2.m_pString)+1);
    cout << (char*)list.GetHead() << endl;
    return 0;
}
```

3.1. Qual o valor de **error1** e **error2** assumindo que as operações de alocação de memória são sempre bem sucedidas.

$error1 = \mathcal{NULL_PTR}$
 $error2 = OK$

3.2. Escreva o resultado deste programa tal como apareceria no ecrã do seu computador.

1
nome
n
n
m
ha,ha!