实现字符串类

#include <iostream>

#include <CString>

#pragma warning(disable:4996)

using namespace std;

class AString {

public:

AString(const char\*str = NULL);

AString(const AString&other);

~AString(void);

AString &operator=(const AString &other);

private:

char \*m\_data;

};

AString::AString(const char\*str)

{

if (str != NULL)

{

//m\_data =(char\*) str;//不宜这样操作，因为str可能在外部会被释放或修改，从而影响到该对象的表达

//应该通过复制字符串的方式进行

int len = strlen(str);

m\_data = new char[len+1];

if (m\_data)

{

strcpy(m\_data, str);

}

m\_data[len] = '\0';

}

else

{

m\_data = new char[1];

\*m\_data = '\0';

}

}

AString& AString::operator=(const AString &other)

{

/\* 不能这样直接返回本类对象，要释放\*/

//\*this = other;//这样会重复调用operator=，产生无限循环，相当于在递归调用

//赋值应当是将other的m\_data内容复制到this指向的对象的m\_data;相当于重复构造函数的操作

if(m\_data)

delete[]m\_data;

if (other.m\_data)

{

int len = strlen(other.m\_data);

m\_data = new char[len + 1];

strcpy(m\_data, other.m\_data);

m\_data[len] = '\0';

}

else

{

m\_data = NULL;

}

return \*this;

}

AString::AString(const AString&other) {

//同样的道理不应该直接赋值字符串指针(浅拷贝)

//应当通过复制字符串的方式进行深拷贝

int len = strlen(other.m\_data);

m\_data = new char[len+1];

strcpy(m\_data, other.m\_data);

};

AString::~AString(void) {

delete[] m\_data;

};