项目设计：截止时间2022年5月20日

编程实现仿string类设计的字符串类MyString，并自行设计测试该类功能的示例程序。

**1. 构造函数**

MyString类有多个构造函数，用法示例如下：

MyString s1(); // si = ""

MyString s2("Hello"); // s2 = "Hello"

MyString s3(4, 'K'); // s3 = "KKKK"

MyString s4("12345", 1, 3); //s4 = "234"，即 "12345" 的从下标 1 开始，长度为 3 的子串

**2. 对MyString对象赋值**

可以用 char\* 类型的变量、常量，以及 char 类型的变量、常量对MyString对象进行赋值。例如：

MyString s1;

s1 = "Hello"; // s1 = "Hello"

s2 = 'K'; // s2 = "K”

MyString类还有 assign 成员函数，可以用来对MyString对象赋值。assign 成员函数返回对象自身的引用。例如：

MyString s1("12345"), s2;

s3.assign(s1); // s3 = s1

s2.assign(s1, 1, 2); // s2 = "23"，即 s1 的子串(1, 2)

s2.assign(4, 'K'); // s2 = "KKKK"

s2.assign("abcde", 2, 3); // s2 = "cde"，即 "abcde" 的子串(2, 3)

**3. 求字符串的长度**

length 成员函数返回字符串的长度。size 成员函数可以实现同样的功能。

**4. MyString对象中字符串的连接**

除了可以使用+和+=运算符对MyString对象执行字符串的连接操作外，MyString类还有 append 成员函数，可以用来向字符串后面添加内容。append 成员函数返回对象自身的引用。例如：

MyString s1("123"), s2("abc");

s1.append(s2); // s1 = "123abc"

s1.append(s2, 1, 2); // s1 = "123abcbc"

s1.append(3, 'K'); // s1 = "123abcbcKKK"

s1.append("ABCDE", 2, 3); // s1 = "123abcbcKKKCDE"，添加 "ABCDE" 的子串(2, 3)

**5. MyString对象的比较**

除了可以用<、<=、==、!=、>=、>运算符比较MyString对象外，MyString类还有 compare 成员函数，可用于比较字符串。compare 成员函数有以下返回值：

小于 0 表示当前的字符串小；

等于 0 表示两个字符串相等；

大于 0 表示另一个字符串小。

例如：

MyString s1("hello"), s2("hello, world");

int n = s1.compare(s2);

n = s1.compare(1, 2, s2, 0, 3); //比较s1的子串 (1,2) 和s2的子串 (0,3)

n = s1.compare(0, 2, s2); // 比较s1的子串 (0,2) 和 s2

n = s1.compare("Hello");

n = s1.compare(1, 2, "Hello"); //比较 s1 的子串(1,2)和"Hello”

n = s1.compare(1, 2, "Hello", 1, 2); //比较 s1 的子串(1,2)和 "Hello" 的子串(1,2)

**6. 求MyString对象的子串**

substr 成员函数可以用于求子串 (n, m)，原型如下：

MyString substr(int n = 0, int m = MyString::npos) const;

调用时，如果省略 m 或 m 超过了字符串的长度，则求出来的子串就是从下标 n 开始一直到字符串结束的部分。例如：

MyString s1 = "this is ok";

MyString s2 = s1.substr(2, 4); // s2 = "is i"

s2 = s1.substr(2); // s2 = "is is ok"

**7. 交换两个MyString对象的内容**

swap 成员函数可以交换两个MyString对象的内容。例如：

MyString s1("West”), s2("East");

s1.swap(s2); // s1 = "East"，s2 = "West"

**8. 查找子串和字符**

MyString类有一些查找子串和字符的成员函数，它们的返回值都是子串或字符在MyString对象字符串中的位置（即下标）。如果查不到，则返回MyString::npos。MyString: :npos 是在MyString类中定义的一个静态常量。这些函数如下：

find：从前往后查找子串或字符出现的位置。

rfind：从后往前查找子串或字符出现的位置。

find\_first\_of：从前往后查找何处出现另一个字符串中包含的字符。例如：

s1.find\_first\_of("abc");  //查找s1中第一次出现"abc"中任一字符的位置

find\_last\_of：从后往前查找何处出现另一个字符串中包含的字符。

find\_first\_not\_of：从前往后查找何处出现另一个字符串中没有包含的字符。

find\_last\_not\_of：从后往前查找何处出现另一个字符串中没有包含的字符。

下面是MyString类的查找成员函数的示例程序。

#include <iostream>

#include <MyString.h>

u[sin](http://c.biancheng.net/ref/sin.html)g namespace std;

int main()

{

MyString s1("Source Code");

int n;

if ((n = s1.find('u')) != MyString::npos) //查找 u 出现的位置

cout << "1) " << n << "," << s1.substr(n) << endl;

//输出 l)2,urce Code

if ((n = s1.find("Source", 3)) == MyString::npos)//从下标3开始查找"Source"，找不到

cout << "2) " << "Not Found" << endl; //输出 2) Not Found

if ((n = s1.find("Co")) != MyString::npos) //查找子串"Co"。能找到，返回"Co"的位置

cout << "3) " << n << ", " << s1.substr(n) << endl;

//输出 3) 7, Code

if ((n = s1.find\_first\_of("ceo")) != MyString::npos)//查找第一次出现或 'c'、'e'或'o'的位置

cout << "4) " << n << ", " << s1.substr(n) << endl;

//输出 4) l, ource Code

if ((n = s1.find\_last\_of('e')) != MyString::npos)//查找最后一个 'e' 的位置

cout << "5) " << n << ", " << s1.substr(n) << endl;

//输出 5) 10, e

if ((n = s1.find\_first\_not\_of("eou", 1)) != MyString::npos)

//从下标1开始查找第一次出现非 'e'、'o' 或 'u' 字符的位置

cout << "6) " << n << ", " << s1.substr(n) << endl;

//输出 6) 3, rce Code

return 0;

}

**9. 替换子串**

replace 成员函数可以对MyString对象中的子串进行替换，返回值为对象自身的引用。例如：

MyString s1("Real Steel");

s1.replace(1, 3, "123456", 2, 4); //用 "123456" 的子串(2,4) 替换 s1 的子串(1,3)

cout << s1 << endl; //输出 R3456 Steel

MyString s2("Harry Potter");

s2.replace(2, 3, 5, '0'); //用 5 个 '0' 替换子串(2,3)

cout << s2 << endl; //输出 HaOOOOO Potter

int n = s2.find("OOOOO"); //查找子串 "00000" 的位置，n=2

s2.replace(n, 5, "XXX"); //将子串(n,5)替换为"XXX"

cout << s2 << endl; //输出 HaXXX Potter

**10. 删除子串**

erase 成员函数可以删除MyString对象中的子串，返回值为对象自身的引用。例如：

MyString s1("Real Steel");

s1.erase(1, 3); //删除子串(1, 3)，此后 s1 = "R Steel"

s1.erase(5); //删除下标5及其后面的所有字符，此后 s1 = "R Ste"

**11. 插入字符串**

insert 成员函数可以在MyString对象中插入另一个字符串，返回值为对象自身的引用。例如：

MyString s1("Limitless"), s2("00");

s1.insert(2, "123"); //在下标 2 处插入字符串"123"，s1 = "Li123mitless"

s1.insert(3, s2); //在下标 2 处插入 s2 , s1 = "Li10023mitless"

s1.insert(3, 5, 'X'); //在下标 3 处插入 5 个 'X'，s1 = "Li1XXXXX0023mitless"

**12. 将MyString对象作为流处理(选做)**

编程实现类iMyStringstream 和 oMyStringstream，可以将MyString对象当作一个流进行输入输出。

示例程序如下：

#include <iostream>

#include <MyStringstream.h>

#include <MyString.h>

using namespace std;

int main()

{

MyString src("Avatar 123 5.2 Ti[tan](http://c.biancheng.net/ref/tan.html)ic K");

iMyStringstream istrStream(src); //建立src到istrStream的联系

MyString s1, s2;

int n; double d; char c;

istrStream >> s1 >> n >> d >> s2 >> c; //把src的内容当做输入流进行读取

oMyStringstream ostrStream;

ostrStream << s1 << endl << s2 << endl << n << endl << d << endl << c <<endl;

cout << ostrStream.str();

return 0;

}

程序的输出结果是：

Avatar

Titanic

123

5.2

K

第 11 行，从输入流 istrStream 进行读取，过程和从 cin 读取一样，只不过输入的来源由键盘变成了MyString对象 src。因此，"Avatar" 被读取到 s1，123 被读取到 n，5.2 被读取到 d，"Titanic" 被读取到s2，'K' 被读取到 c。

第 12 行，将变量的值输出到流 ostrStream。输出结果不会出现在屏幕上，而是被保存在 ostrStream 对象管理的某处。用 oMyStringstream 类的 str 成员函数能将输出到 oMyStringstream 对象中的内容提取出来。

附：测试程序

#include <iostream>

//#define \_SELECT

//如果完成了选做内容，请将此宏\_SELECT定义前的注释符号删除

#include <cstring>

#ifndef \_SELECT

#include <string>

#include <sstream>

#endif

using namespace std;

// 请将完成的所有代码粘贴在此处，包括类的声明和所有定义 ,并注释掉下面一行语句

typedef string MyString;

//测试时请注释掉上面一行语句

int main()

{

{ //测试1

MyString s1; // si = ""

MyString s2("Hello"); // s2 = "Hello"

MyString s3(4, 'K'); // s3 = "KKKK"

MyString s4("12345", 1, 3); //s4 = "234"，即 "12345" 的从下标 1 开始，长度为 3 的子串

cout<<s1<<s2<<s3<<s4 ;

}

{ //测试2

{

MyString s1,s2;

s1 = "Hello"; // s1 = "Hello"

s2 = 'K'; // s2 = "K”

cout<<s1<<s2 ;

}

{

MyString s1("12345"), s2,s3;

s3.assign(s1); // s3 = s1

cout<<s3;

s2.assign(s1, 1, 2); // s2 = "23"，即 s1 的子串(1, 2)

cout<<s2;

s2.assign(4, 'K'); // s2 = "KKKK"

cout<<s2;

s2.assign("abcde", 2, 3); // s2 = "cde"，即 "abcde" 的子串(2, 3)

cout<<s2 ;

}

}

{ //测试3

MyString s1("12345"), s2;

cout<<s1.length()<<s2.size() ;

}

{//测试4

MyString s1("123"), s2("abc");

s1.append(s2); // s1 = "123abc"

cout<<s1;

s1.append(s2, 1, 2); // s1 = "123abcbc"

cout<<s1;

s1.append(3, 'K'); // s1 = "123abcbcKKK"

cout<<s1;

s1.append("ABCDE", 2, 3); // s1 = "123abcbcKKKCDE"，添加 "ABCDE" 的子串(2, 3)

cout<<s1 ;

}

{ //测试5

MyString s1("hello"), s2("hello, world");

cout<<(s1>s2)<<(s1<s2)<<(s1>=s2)<<(s1<=s2)<<(s1==s2)<<(s1!=s2) ;

int n = s1.compare(s2);

cout<<n;

n = s1.compare(1, 2, s2, 0, 3); //比较s1的子串 (1,2) 和s2的子串 (0,3)

cout<<n;

n = s1.compare(0, 2, s2); // 比较s1的子串 (0,2) 和 s2

cout<<n;

n = s1.compare("Hello");

cout<<n;

n = s1.compare(1, 2, "Hello"); //比较 s1 的子串(1,2)和"Hello”

cout<<n;

n = s1.compare(1, 2, "Hello", 1, 2); //比较 s1 的子串(1,2)和 "Hello" 的子串(1,2)

cout<<n ;

}

{ //测试6

MyString s1 = "this is ok";

MyString s2 = s1.substr(2, 4); // s2 = "is i"

cout<<s1<<s2;

s2 = s1.substr(2); // s2 = "is is ok"

cout<<s2 ;

}

{ //测试7

MyString s1("West"), s2("East");

cout<<s1<<s2;

s1.swap(s2); // s1 = "East"，s2 = "West"

cout<<s1<<s2 ;

}

{ //测试8

MyString s1("Source Code");

size\_t n;

if ((n = s1.find('u')) != MyString::npos) //查找 u 出现的位置

cout << "1) " << n << "," << s1.substr(n) ;

//输出 l)2,urce Code

if ((n = s1.find("Source", 3)) == MyString::npos)//从下标3开始查找"Source"，找不到

cout << "2) " << "Not Found" ; //输出 2) Not Found

if ((n = s1.find("Co")) != MyString::npos) //查找子串"Co"。能找到，返回"Co"的位置

cout << "3) " << n << ", " << s1.substr(n) ;

//输出 3) 7, Code

if ((n = s1.find\_first\_of("ceo")) != MyString::npos)//查找第一次出现或 'c'、'e'或'o'的位置

cout << "4) " << n << ", " << s1.substr(n) ;

//输出 4) l, ource Code

if ((n = s1.find\_last\_of('e')) != MyString::npos)//查找最后一个 'e' 的位置

cout << "5) " << n << ", " << s1.substr(n) ;

//输出 5) 10, e

if ((n = s1.find\_first\_not\_of("eou", 1)) != MyString::npos) //从下标1开始查找第一次出现非 'e'、'o' 或 'u' 字符的位置

cout << "6) " << n << ", " << s1.substr(n) ;

//输出 6) 3, rce Code

}

{ //测试9

MyString s1("Real Steel");

cout<<s1;

s1.replace(1, 3, "123456", 2, 4); //用 "123456" 的子串(2,4) 替换 s1 的子串(1,3)

cout << s1 ; //输出 R3456 Steel

MyString s2("Harry Potter");

cout<<s2;

s2.replace(2, 3, 5, '0'); //用 5 个 '0' 替换子串(2,3)

cout << s2 ; //输出 HaOOOOO Potter

size\_t n = s2.find("00000"); //查找子串 "00000" 的位置，n=2

cout<<n; //输出2

s2.replace(n, 5, "XXX"); //将子串(n,5)替换为"XXX"

cout << s2 ; //输出 HaXXX Potter

}

{ //测试10

MyString s1("Real Steel");

cout<<s1;

s1.erase(1, 3); //删除子串(1, 3)，此后 s1 = "R Steel"

cout<<s1;

s1.erase(5); //删除下标5及其后面的所有字符，此后 s1 = "R Ste"

cout<<s1 ;

}

{ //测试11

MyString s1("Limitless"), s2("00");

cout<<s1<<s2;

s1.insert(2, "123"); //在下标 2 处插入字符串"123"，s1 = "Li123mitless"

cout<<s1;

s1.insert(3, s2); //在下标 2 处插入 s2 , s1 = "Li10023mitless"

cout<<s1;

s1.insert(3, 5, 'X'); //在下标 3 处插入 5 个 'X'，s1 = "Li1XXXXX0023mitless"

cout<<s1;

}

{ //测试12 选做部分

#ifndef \_SELECT

//没有完成选作部分则使用string类和istringstream类测试

string src("Avatar 123 5.2 Titanic K");

istringstream istrStream(src); //建立src到istrStream的联系

string s1, s2;

int n; double d; char c;

istrStream >> s1 >> n >> d >> s2 >> c; //把src的内容当做输入流进行读取

ostringstream ostrStream;

ostrStream << s1 << s2 << n << d << c ;

cout << ostrStream.str()<<endl;

#else

//如果完成选作部分则使用MyString类和iMyStringstream类测试

MyString src("Avatar 123 5.2 Titanic K");

iMyStringstream istrStream(src); //建立src到istrStream的联系

MyString s1, s2;

int n; double d; char c;

istrStream >> s1 >> n >> d >> s2 >> c; //把src的内容当做输入流进行读取

oMyStringstream ostrStream;

ostrStream << s1 << s2 << n << d << c ;

cout << ostrStream.str()<<endl;

#endif // \_SELECT

}

return 0;

}