**1.** 实验目的

练习string类的使用。掌握string类的insert，erase，find，replace等常用方法。

学习类库的应用方法。通常我们需要阅读联机帮助或类库参考手册，通过联机帮助或类库参考手册（类库工作手册）了解：

⒈　软件开发系统提供的类库的类层次结构。

⒉　可能需要用到的类及其父类的属性和方法。

⒊　通过联机文档提供的例子学习有关类的使用方法。

下列网址提供了标准类库中string类的在线帮助：

http://www.cplusplus.com/reference/string/string/

http://en.cppreference.com/w/cpp/string

同学们可以通过在线帮助中的示例来学习使用string类。

如果在阅读英文的在线帮助时感到太困难，可以直接阅读并运行本次实验内容中例题代码。

**2.** 实验要求

(1) 学习string类对象的创建和访问。

(2) 熟悉string类的insert，erase，find，replace等常用方法。

(3) 学习string类对象的复制、比较和连接等方法。

**3.** 实验内容

(1) 通过下面的示例程序我们可以比较与string类对象的尺寸相关的方法，编译运行该程序，观察其运行输出的结果，你认为string类对象中采用何种方式存储字符串？

// comparing size, length, capacity and max\_size

#include <iostream>

#include <string>

using namespace std;

int main ()

{

  string str ("Test string");

  cout << "size: " << str.size() << "\n";

  cout << "length: " << str.length() << "\n";

  cout << "capacity: " << str.capacity() << "\n";

  cout << "max\_size: " << str.max\_size() << "\n";

  return 0;

}

(2) 下面的程序演示了string类对象的插入insert方法，编译、运行该程序。想一想如果要你实现字符串的插入，需要完成哪些操作步骤？

//  Demonstrate insert().

#include <iostream>

#include <string>

using namespace std;

int main()

{

  string str1("The website address of our university is:");

  string str2("www.xupt.edu.cn");

  cout << "Initial strings:\n";

  cout << "str1: " << str1 << endl;

  cout << "str2: " << str2 << "\n\n";

// demonstrate insert()

  cout << "Insert str1 into str2:\n";

  str2.insert(0,str1);

  cout << str2 << "\n\n";

}

(3) 下面的程序演示了string类对象的删除erase方法，编译、运行该程序。想一想如果要你实现字符串中的子串的删除，需要完成哪些操作步骤？

// string::erase

#include <iostream>

#include <string>

using namespace std;

int main ()

{

  std::string str ("This is an example sentence.");

  std::cout << str << '\n';

                                           // "This is an example sentence."

  str.erase (10,8);                        //            ^^^^^^^^

  std::cout << str << '\n';

                                           // "This is an sentence."

  str.erase (str.begin()+9);               //        ^

  std::cout << str << '\n';

                                           // "This is a sentence."

  str.erase (str.begin()+5, str.end()-9);  // ^^^^^

  std::cout << str << '\n';

                                           // "This sentence."

  return 0;

}

(4) 下面的程序演示了string类对象的查找find方法，阅读并编译、运行该程序，观察输出结果。利用你在数据结构课程中学习过的字符串查找算法完成本题类似的查找操作。

// string::find

#include <iostream>

#include <string>

using namespace std;

int main ()

{

  string str ("There are two needles in this haystack with needles.");

  string str2 ("needle");

  // different member versions of find in the same order as above:

  int found = str.find(str2);

  if (found!=-1)

    cout << "first 'needle' found at: " << found << '\n';

  found=str.find("needles are small",found+1,6);

  if (found!=-1)

    cout << "second 'needle' found at: " << found << '\n';

  found=str.find("haystack");

  if (found!=-1)

    cout << "'haystack' also found at: " << found << '\n';

  found=str.find('.');

  if (found!=-1)

    cout << "Period found at: " << found << '\n';

  // let's replace the first needle:

  str.replace(str.find(str2),str2.length(),"preposition");

  cout << str << '\n';

  return 0;

}

(5) 下面的程序演示了string类对象的替换replace方法，阅读并编译、运行该程序。想一想如果要你实现字符串中的子串的替换，需要完成哪些操作步骤？

// replacing in a string

#include <iostream>

#include <string>

using namespace std;

int main ()

{

    string base="this is a test string.";

    string str2="n example";

    string str3="sample phrase";

    string str4="useful.";

    // replace signatures used in the same order as described above:

    // Using positions:                 0123456789\*123456789\*12345

    string str=base;           // "this is a test string."

    str.replace(9,5,str2);          // "this is an example string." (1)

    str.replace(19,6,str3,7,6);     // "this is an example phrase." (2)

    str.replace(8,10,"just a");     // "this is just a phrase."     (3)

    str.replace(8,6,"a shorty",7);  // "this is a short phrase."    (4)

    str.replace(22,1,3,'!');        // "this is a short phrase!!!"  (5)

    cout << str << '\n';

}

(6) 下面的程序演示了string类对象的复制copy方法，阅读并编译、运行该程序。请你将copy方法与C语言中的strcpy( )函数进行比较。

// string::copy

#include <iostream>

#include <string>

using namespace std;

//int string::copy(char\* s, int len, int pos = 0) const 函数原型

int main ()

{

  char ChangAn[20];

  string str ("Xi'an University of Post & Telecommunications");

  int length = str.copy(ChangAn,10,6);

  ChangAn[length]='\0';

  cout << "ChangAn contains: " << ChangAn << '\n';

  return 0;

}

(7) 下面的程序演示了string类的连接方法，采用运算符“+”表示连接两个字符串。阅读并编译、运行该程序。请与C语言中的strcat( )函数进行比较。

// concatenating strings

#include <iostream>

#include <string>

main ()

{

  std::string firstlevel ("cn");

  std::string secondlevel ("edu");

  std::string thirdlevel("xupt");

  std::string scheme ("http://");

  std::string hostname;

  std::string url;

  hostname = "www." + thirdlevel+ '.'+secondlevel + '.' + firstlevel;

  url = scheme + hostname;

  std::cout << url << '\n';

  return 0;

}

(8) 下面的程序演示了string类的比较compare方法，阅读并编译、运行该程序。

// comparing apples with apples

#include <iostream>

#include <string>

int main ()

{

  std::string str1 ("green apple");

  std::string str2 ("red apple");

  if (str1.compare(str2) != 0)

    std::cout << str1 << " is not " << str2 << '\n';

  if (str1.compare(6,5,"apple") == 0)

    std::cout << "still, " << str1 << " is an apple\n";

  if (str2.compare(str2.size()-5,5,"apple") == 0)

    std::cout << "and " << str2 << " is also an apple\n";

  if (str1.compare(6,5,str2,4,5) == 0)

    std::cout << "therefore, both are apples\n";

  return 0;

}

string类的比较compare方法包括6个重载函数，函数原型如下：

string (1)

int compare (const string& str) const;

substrings (2)

int compare (size\_t pos, size\_t len, const string& str) const;

int compare (size\_t pos, size\_t len, const string& str,

             size\_t subpos, size\_t sublen) const;

c-string (3)

int compare (const char\* s) const;

int compare (size\_t pos, size\_t len, const char\* s) const;

buffer (4)

int compare (size\_t pos, size\_t len, const char\* s, size\_t n) const;