第四章补充作业：

（1）

源代码：

// main.cpp

// 1

// Created by mac on 16/3/29.

// Copyright © 2016年 mac. All rights reserved.

#include <iostream>

using namespace std;

int main(int argc, const char \* argv[]) {

// insert code here...

//std::cout << "Hello, World!\n";

string str="Xi’an Institute of Posts and Telecommunications is co-designed and implemented by the People’s Government of Shaanxi Province and the Ministry of Industry and Information Technology.\nThe Institute is located in Xi’an, a historic city in Northwest China, famous for its magnificent ancient culture";

string str1="Institute";

string rstr="University";

string::size\_type pfd=str.find(str1);

while (pfd!=string::npos) {

str.replace(pfd, rstr.length()-1, rstr);

pfd=str.find(str1);

}

cout<<str<<endl;

return 0;

}

运行结果：

**Xi’an University of Posts and Telecommunications is co-designed and implemented by the People’s Government of Shaanxi Province and the Ministry of Industry and Information Technology.**

**The University is located in Xi’an, a historic city in Northwest China, famous for its magnificent ancient culture**

**Program ended with exit code: 0**

**（2）：**

**源代码：**

// main.cpp

// 2

// Created by mac on 16/3/29.

// Copyright © 2016年 mac. All rights reserved.

#include <iostream>

using namespace std;

int main(int argc, const char \* argv[]) {

// insert code here...

//std::cout << "Hello, World!\n";

string string1("The airplane landed on time!");

cout<<string1.substr(7,5)<<endl;

return 0;

}

运行结果：

**plane**

**Program ended with exit code: 0**

**（3）：**

**源代码：**

#include <iostream>

using std::cout;

using std::endl;

#include <string>

using std::string;

int main()

{

string first( "one" );

string second( "two" );

cout << "Before swap:\n first: " << first << "\nsecond: " << second;

first.swap( second );

cout << "\n\nAfter swap:\n first: " << first

<< "\nsecond: " << second << endl;

return 0;

}

运行结果：

**Before swap:**

**first: one**

**second: two**

**After swap:**

**first: two**

**second: one**

**Program ended with exit code: 0**

**（4）：**

**源代码：**

#include <iostream>

using std::cout;

using std::endl;

using std::cin;

using std::boolalpha;

#include <string>

using std::string;

void printStatistics( const string & );

int main()

{

string string1;

cout << "Statistics before input:\n" << boolalpha;

printStatistics( string1 );

cout << "\n\nEnter a string: ";

cin >> string1;

cout << "The string entered was: " << string1;

cout << "\nStatistics after input:\n";

printStatistics( string1 )；

cin >> string1;

cout << "\n\nThe remaining string is: " << string1 << endl;

printStatistics( string1 );

string1 += "1234567890abcdefghijklmnopqrstuvwxyz1234567890";

cout << "\n\nstring1 is now: " << string1 << endl;

printStatistics( string1 );

string1.resize( string1.length() + 10 );

cout << "\n\nStats after resizing by (length + 10):\n";

printStatistics( string1 );

cout << endl;

return 0;

}

void printStatistics( const string &stringRef )

{

cout << "capacity: " << stringRef.capacity() << "\nmax size: "

<< stringRef.max\_size() << "\nsize: " << stringRef.size()

<< "\nlength: " << stringRef.length()

<< "\nempty: " << stringRef.empty();

}

**运行结果：**

**Statistics before input:**

**capacity: 22**

**max size: 18446744073709551599**

**size: 0**

**length: 0**

**empty: true**

**Enter a string: Steve\_ke**

**The string entered was: STEVE\_KE**

**Statistics after input:**

**capacity: 22**

**max size: 18446744073709551599**

**size: 8**

**length: 8**

**empty: falsehuang\_ke**

**The remaining string is: HUANG\_KE**

**capacity: 22**

**max size: 18446744073709551599**

**size: 8**

**length: 8**

**empty: false**

**string1 is now: HUANG\_KE1234567890abcdefghijklmnopqrstuvwxyz1234567890**

**capacity: 63**

**max size: 18446744073709551599**

**size: 54**

**length: 54**

**empty: false**

**Stats after resizing by (length + 10):**

**capacity: 127**

**max size: 18446744073709551599**

**size: 64**

**length: 64**

**empty: false**

**Program ended with exit code: 0**

(5):

源代码：

#include <iostream>

using std::cout;

using std::endl;

#include <string>

using std::string;

int main()

{

string string1( "beginning end" );

string string2( "middle " );

string string3( "12345678" );

string string4( "xx" );

cout << "Initial strings:\nstring1: " << string1

<< "\nstring2: " << string2 << "\nstring3: " << string3

<< "\nstring4: " << string4 << "\n\n";

string1.insert(10, string2);

string3.insert(3, string4);

cout << "Strings after insert:\nstring1: " << string1

<< "\nstring2: " << string2 << "\nstring3: " << string3

<< "\nstring4: " << string4 << endl;

return 0;

}

运行结果：

**Initial strings:**

**string1: beginning end**

**string2: middle**

**string3: 12345678**

**string4: xx**

**Strings after insert:**

**string1: beginning middle end**

**string2: middle**

**string3: 123xx45678**

**string4: xx**

**Program ended with exit code: 0**

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;

}

 答：string类的实例中，字符串的长度为string类对象的实际长度，c语言中字符串通常以/0结尾，长度会比实际长度大，string的长度与实际串的运行长度相同，系统为string类对象分配存储空间时，往往比实际字符串长度更大的空间，以适应字符串的改变。

(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";

}

答：string类对象在完成insert()插入方法时，首先要得到需要插入的字符串，然后找到需要插入的位置，调用被插入对象的insert()方法将第一步得到的串插入到该串中。

(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;

}

答：与string类的insert()方法类似，首先需要得到需要被删除的串。找到需要删除的子串的起始位置。输入需删除子串的长度。调用string类对象的erase()方法，实现子串删除操作。

(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;

}

 答：字符串查找操作是模式匹配的经典内容。在数据结构与算法课堂中，我们学习过常用于字符串查找操作的算法有BF算法，KMP算法。BF算法时间复杂度较高，效率比较低下。KMP算法效率较高，可以用于字符串查找操作。

伪代码：

ADT{

char\* str = "There are two needles in this haystack with needles.";

  char\* str2 = "needle";

  int found = StrIndex(str,0,str2);

  if (found!=-1)

    printf( "first 'needle' found at: %d\n",found );

  found= StrIndex ("needles are small",found+1,str2);

  if (found!=-1)

printf( " second 'needle' found at: %d\n",found );

  found= StrIndex (str,0,"haystack");

  if (found!=-1)

    printf( "'haystack' also found at: %d\n" , found) ;

  found= StrIndex (str,0,'.');

  if (found!=-1)

  printf("Period found at: %d\n", found );

  StrReplace(str, "needle","preposition");

  printf(”%s”,str);

  return 0;

}

KMP算法实现StrIndex():

int StrIndex(SString S,int pos,SString T){

int i=pos,j=0;

while(i<S.len&&j<T.len){

if(T.ch[j]==S.ch[i]||j==0){i++,j++}

else j=nextval[j];

}

if(j=T.len) return i-j;

else return 0;

}

void Get\_next(SString T){

int j=1,k=0;

next[1]=0;

while(j<T.len){

if(k==0||T.ch[j]==T.ch[k]) next[++j]=++k;

else k=next[k];

}

}

void Get\_nextval(SString T){

int i=2,k;

Get\_next(T);

nextval[1]=0;

while(i<T.len){

k=next[j];

if(T.ch[i]==T.ch[k]) nextval[i]=naxtval[k];

else nextval[i]=k;

i++;

}

}

(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';

}

答：首先应该确定目标字符串，再确定替代串，在目标字符串中确定应该替换放入放入位置，确定替换长度，调用replace()方法实现字符串的替换。

(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;

}

答：在c语言中字符串copy函数实现这个字符串的拷贝，包括字符串结束标志’\0’。如给目标字符串分配空间不完整拷贝会丢失源串的内容。

C++的string类copy方法实现字符拷贝，若需给串末尾加’\0’。则需要手动添加。

(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;

}

 答:c++在实现字符串连接时非常方便,直接使用’+=’操作,便可实现字符串连接因为被连接的各个字符串末尾都没有字符串结束标识符.在c语言中连接两个字符串的时候，首先需要删除前一个字符串的结束标识符’/0’,接着开辟另外一个更大的存储空间用来存储相加之后的结果，将前字符串的除末尾元素’\0’之外的内容存到新开辟的空间，再将后一个穿的内容存到第一个串复制完成之后的地方。

(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;