**实验一 String-字符串**

一、实验目的

字符串匹配

二、实验要求

至少实现两种算法完成字符串匹配，并比较算法之间的性能差异。

处理字符串操作相关问题时，常见的做法是从字符串尾部开始编辑，从后往前逆向操作。这么做的原因是因为字符串的尾部往往有足够空间，可以直接修改而不用担心覆盖字符串前面的数据。

三、算法思想

1.双重for循环算法

class Solution

{

public:

int strStr(string haystack, string needle)

{

if (haystack.empty() && needle.empty()) return 0;

if (haystack.empty()) return -1;

if (needle.empty()) return 0;

// in case of overflow for negative

if (haystack.size() < needle.size()) return -1;

for (int i = 0; i < haystack.size() - needle.size() + 1; i++)

{

string::size\_type j = 0;

for (; j < needle.size(); j++)

{

if (haystack[i + j] != needle[j]) break;

}

if (j == needle.size()) return i;

}

return -1;

}

2.KMP算法

int kmp(char s1[],char s2[])

{

int i,j;

i=j=0;

int len1=strlen(s1);

int len2=strlen(s2);

while(i<len1&&j<len2)

{

if(j==-1||s1[i]==s2[j])

{

i++;

j++;

}

else j=next[j];

}

if(j==len2)

return i-len2+1;

else return -1;

}

四、程序清单

1.双重for循环算法

#include<iostream>

#include<string>

using namespace std;

class Solution

{

public:

int strStr(string haystack, string needle)

{

if (haystack.empty() && needle.empty()) return 0;

if (haystack.empty()) return -1;

if (needle.empty()) return 0;

// in case of overflow for negative

if (haystack.size() < needle.size()) return -1;

for (int i = 0; i < haystack.size() - needle.size() + 1; i++)

{

string::size\_type j = 0;

for (; j < needle.size(); j++)

{

if (haystack[i + j] != needle[j]) break;

}

if (j == needle.size()) return i;

}

return -1;

}

};

int main()

{

string haystack;

string needle;

Solution A;

cout<<"输入源字符串："<<endl;

cin>>haystack;

cout<<"输入目的字符串："<<endl;

cin>>needle;

cout<<A.strStr(haystack,needle)<<endl;

return 0;

}

2.KMP算法

#include <iostream>

#include <algorithm>

#include <string>

#include <cstring>

using namespace std;

int next[101];

void getnext(char s[])

{

int i=0,j=-1;

next[0]=-1;

int len=strlen(s);

while(i<len)

{

if(j==-1||s[i]==s[j])

{

i++;

j++;

if(s[i]!=s[j]) next[i]=j;

else next[i]=next[j];

}

else j=next[j];

}

}

int kmp(char s1[],char s2[])

{

int i,j;

i=j=0;

int len1=strlen(s1);

int len2=strlen(s2);

while(i<len1&&j<len2)

{

if(j==-1||s1[i]==s2[j])

{

i++;

j++;

}

else j=next[j];

}

if(j==len2)

return i-len2+1;

else return -1;

}

int main()

{

char s1[101],s2[101];

while(cout<<"输入源字符串:"<< endl <<"输入目的字符串:"<<endl,

cin>>s1>>s2)

{

getnext(s2);

int x=kmp(s1,s2);

if(x!=-1) cout<<"1\n";

else cout<<"-1\n";

}

return 0;

}

运行结果

1.双重for循环算法



2.KMP算法

