### 第10题：URL映射

#### 数据结构：

字符串类型string，将字符串与流关联的对象stringstream，STL库中的vector类型

#### 算法设计思想：

每条规则分为两部分：规则本体和规则名字，故可封装在一起作为一个类Rule。获取规则和地址时用字符串存储，然后进行分割成多个可匹配项，由于需要便捷按“/”分割，故使用string和stringstream，由于有多个规则和地址且不定长，故使用vector来做为二维字符串类型数组，vector<string>存储地址，vector<Rule>存储规则。

思想很简单。获取当前地址时逐个与每条规则的每个可匹配项尝试匹配，匹配到其中一条规则根据该规则输出相应规则名字。

值得注意的是地址末尾若有分隔符，该分隔符不能当作分隔符，而是当作地址最后一个可匹配项的一部分。在之后的分割过程和比较判断匹配过程需特别注意到这种情况。

#### 源程序：

#include<cstdio>

#include<algorithm>

#include<iostream>

#include<iomanip>

#include<vector>

#include<string>

#include<string.h>

#include<sstream>

using namespace std;

//规则

class Rule{

public:

string name;

vector<string> content;

Rule(string in\_name,vector<string> in\_content) :name(in\_name),content(in\_content){}

};

//存储数条规则的变量

vector<Rule> rules;

bool isMatch;

//string与int相互转换的函数

//stringstream方便类型转换及字符串分割

int StringToInt(string s){

int answer;

stringstream ss(s);

ss >> answer;

return answer;

}

string IntToString(int answer){

stringstream ss;

ss << answer;

return ss.str();

}

bool JudgeIsNum(string s){

for (int i = 0; i < s.length(); i++)

{

if (s[i] > '9' || s[i] < '0') return false;

}

return true;

}

void HandleURLRule (string s,string name){

int spiltLoc;

string str;

vector<string> content;

bool isSpilt = true;

//URL最后一个字符为 / 是不算做分隔符的,之后也要加入进字符串组里

if (s[s.length()-1] != '/'){

isSpilt = false;

s = s + "/"; //该数据类型同样方便字符拼接

}

spiltLoc = s.find("/");

while (spiltLoc != -1){

str = s.substr(0,spiltLoc); //获取分割后的其中一个字符串

s = s.substr(spiltLoc + 1,s.length());

content.push\_back(str);

spiltLoc = s.find("/");

}

if (isSpilt){

content.push\_back("/");

}

rules.push\_back(Rule(name,content));

}

//判断符合规则函数，就一个一个逐个if-else地去判断

vector<string> Match (int k, string url){

int spiltLoc;

int i;

string str;

bool isSpilt = true;

vector<string> result;

vector<string> urlCur;

if (url[url.length() - 1] != '/')

{

isSpilt = false;

url = url + "/";

}

int urlIndex = 0;

spiltLoc = url.find("/");

while (spiltLoc != -1)

{

str = url.substr(0,spiltLoc);

url = url.substr(spiltLoc + 1,url.length());

urlCur.push\_back(str);

spiltLoc = url.find("/");

}

if (isSpilt)

urlCur.push\_back("/");

for (i = 0; i < rules[k].content.size() && urlIndex < urlCur.size(); i++){

if (rules[k].content[i] == "<path>" && urlCur[urlIndex].length() != 0){

str = urlCur[urlIndex];

for (int j = urlIndex + 1;j < urlCur.size();j++){

if (urlCur[j] == "/"){

str = str + "/";

}

else

str = str + "/" + urlCur[j];

}

result.push\_back(str);

urlIndex = urlCur.size();

i = rules[k].content.size();

}

else if (rules[k].content[i] == "<int>" && urlCur[urlIndex].length() != 0){

if (JudgeIsNum(urlCur[urlIndex])){

result.push\_back(IntToString(StringToInt(urlCur[urlIndex])));

urlIndex++;

}

else {

isMatch = false;

break;

}

}

else if (rules[k].content[i] == "<str>" && urlCur[urlIndex].length() != 0){

result.push\_back(urlCur[urlIndex]);

urlIndex++;

}

else {

if (rules[k].content[i] == urlCur[urlIndex])

urlIndex++;

else{

isMatch = false;

break;

}

}

}

//未匹配完整

if (urlIndex != urlCur.size() || i < rules[k].content.size()){

isMatch = false;

}

return result;

}

int main(){

string format,name,url;

int ruleNum,addressNum;

int error;

int j;

cin >> ruleNum >> addressNum;

for (int i = 0; i < ruleNum; i++)

{

cin >> format >> name;

format = format.substr(1, format.length());

HandleURLRule(format,name);

/\* code \*/

}

for (int i = 0; i < addressNum; i++)

{

cin >> url;

url = url.substr(1,url.length());

//开始对当前地址进行规则匹配

for ( j = 0; j < ruleNum; j++)

{

isMatch = true;

vector<string> vec = Match(j,url);

if (isMatch)

{

cout << rules[j].name;

for (int h = 0; h < vec.size(); h++)

{

cout << " "<< vec[h];

/\* code \*/

}

cout << endl;

break;

/\* code \*/

}

/\* code \*/

}

if (j == ruleNum) cout<<"404"<<endl;

/\* code \*/

}

return 0;

}

#### 测试数据和结果：

输入：

5 4

/articles/2003/ special\_case\_2003

/articles/<int>/ year\_archive

/articles/<int>/<int>/ month\_archive

/articles/<int>/<int>/<str>/ article\_detail

/static/<path> static\_serve

/articles/2004/

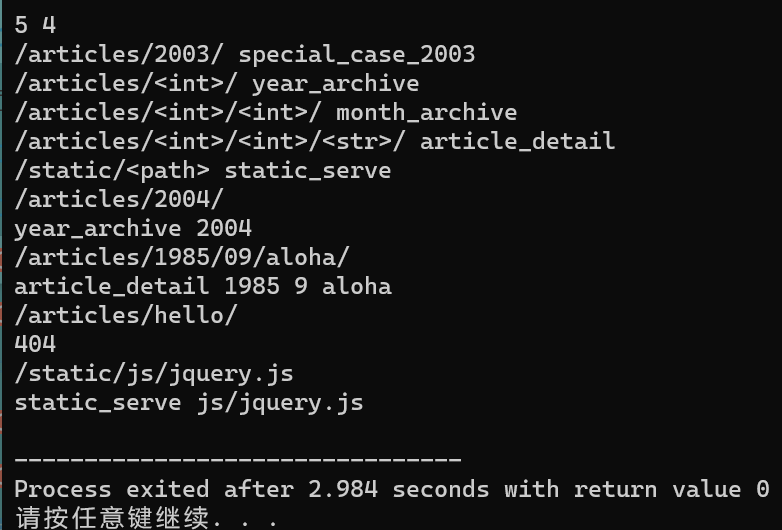
/articles/1985/09/aloha/

/articles/hello/

/static/js/jquery.js

输出：

year\_archive 2004  
article\_detail 1985 9 aloha  
404  
static\_serve js/jquery.js



#### 时间复杂度：

时间复杂度O(n\*m+(n\*m)\*j）

(n为规则数，m为地址数，j为分割后的字符串组中字符串的个数)

#### 该题代码行：

172行