lab3实验报告

侯新铭2021201651

1.需求分析

（1）输入

①载入一个合法的html文件

②检索部分：输入下面selector列表对应的任意有效检索字符串（输入exit退出整个检索过程，不合法字符串则重新请求输入）

选择器	例子	描述
.class	.intro	选择 class="intro" 的所有元素。
.class1.class2	.name1.name2	选择 class 属性中同时有 name1 和 name2 的所有元素。
.class1 .class2	.name1 .name2	选择作为类名 name1 元素后代的所有类名 name2 元素。
#id	#firstname	选择 id="firstname" 的元素。
*	*	选择所有元素。
element	p	选择所有 元素。
element.class	p.intro	选择 class="intro" 的所有 元素。
element,element	div, p	选择所有 元素和所有 元素。
element element	div p	选择 元素内的所有 元素。
element>element	div > p	选择父元素是 的所有 元素。

选择器	例子	描述
element+element	div + p	选择紧跟 元素的首个 元素。
element1~element2	p ~ ul	选择前面有 元素的每个 元素。

③进一步操作部分：输入字符串“text”,"html","href"之一（输入q退出对检索出的元素的操作，其他字符串则重新请求输入）

（2）输出形式

预输出：

正确载入后，会输出一行`Load successfully.`,若载入的html文件不合法，会提示不合法出现的位置及其详细不匹配信息。

把html文件建模为树后，会输出一行`totally ... nodes, root's tag is ....`,来提示用户。

检索后输出：

输出一行指示语句，例如，输入`.content_col_3_item .relatednews`后，会输出`开始执行classChild，选择类名为 content_col_3_item 的元素后代中所有类名为 relatednews 的元素 满足您要求的元素共 1 个。 在存储中的序号分别为：332 位的 div ；`

进一步操作后输出： `text`：各元素内部的所有TEXT结点的内容

`html`：直接输出各元素的html文本

`href`：对于a标签，获取其包含链接href的值

（3）程序所能达到的功能

实现上述列表中任意一行的选择功能，并对选出来的元素通过进一步操作选择性输出。

（4）测试数据

测试数据样例

`.content_col_3_item .relatednews` 开始执行classChild，选择类名为 content_col_3_item 的元素后代中所有类名为 relatednews 的元素 满足您要求的元素共 1 个。 在存储中的序号分别为：332 位的 div ; 请输入你要进行的操作（text,html,href,输入q退出） `html`

```
<div class="relatednews">
                                        <span class="title">相关新闻</span>
                                        <ul>

</ul>
                                </div>
```

text 相关新闻

2.概要设计

（1）抽象数据类型定义

首先，使用之前实现的线性表来存储好html文本，其定义为：

```
typedef struct SqList{
        ElemType* elem;
        int length;
        int listsize;
    }SqList;
```

其中ElemType即为char。顺序表的抽象操作包括：

- InitSqList(SqList &L)：初始化顺序表。
- inputFromFile(SqList &L, char *name, bool print)：读入 HTML 文档并存入顺序表。
- getTag(SqList l, int i, char str[])：获取 HTML 标签。
- typeSingle(char str[])：判断 HTML 标签是否为单标签。

而后借助同样是之前实现的栈来实现用新定义的树来对HTML DOM层次结构进行建模，栈的定义为：

```
typedef struct
{
    sElemType* base;
    sElemType* top;
    int stacksize;
}sqStack;
```

其中sElemType为自己定义的一个结构体：

```
typedef struct Tag{
    char s[30]="";
    char* p;
}Tag;
```

其中的s即该Tag的标签字符串，p指向Tag的开头在线性表中的位置。

新定义的树采用孩子链表的结构来实现，包括很多子定义。首先定义出孩子结点：

```
typedef struct cNode{
    int tNum=0;
    struct cNode *next=NULL;
}*cPtr;
```

再定义构成树的树结点：

```
typedef struct tNode{
    char tag[30]="";
    char* startPtr=NULL;
    char* endPtr=NULL;
    cPtr firstchild=NULL;
}tNode;
```

树的定义为：

```
typedef struct{
    tNode nodes[MAX_TREE_SIZE];
    int n=0;
    int r=0;
}tree;
```

树包含三个成员：`nodes` 数组、`n` 和 `r`。`nodes` 数组用来存储 HTML 文档的标签，`n` 用来存储 HTML 文档的标签数量，`r` 用来存储根节点的位置。

树的抽象操作包括：

- `buildNodes(SqList l, tree &t)`：构建树的节点。
- `buildChilden(SqList l, tree &t)`：构建树的子节点。

此外，为了方便后续函数相互调用，我定义了如下指示一个数组的结构体：

```
typedef struct inArray
{
    int indexs[MAX_TREE_SIZE];
    int n=0;
}inArray;
```

（2）主程序流程以及各程序模块之间的调用关系

我将程序整体模块化为下述几部分：

```
├── header.h
├── seqListForHtml.h
├── stack.h
├── tree.h
├── selector.h
├── main.cpp
```

主程序即main.cpp首先执行下述代码完成对html的存储：

```
    SqList L;
    InitSqList(L)
    char name[20]="lab3_news.html";
    // scanf("%s",name);
    inputFromFile(L, name, true);;
```

其中应用的函数均在头文件seqListForHtml.h中。

再执行下述两个函数完成把html建模为我们定义的树：

```
    tree t;
    buildNodes(L,t);
    buildChilden(L,t);
```

上述两个函数均在头文件tree.h中，其中使用了stack.h中的部分函数。

最后执行输入循环以及每轮中的selector函数：

```
    char s[50]="";
    while (true)
    {
        cout<<"请输入您的检索语句（输入exit退出）："<<endl;
        cin.getline(s, 50);
        if (strcmp(s, "exit") == 0)
        break;

        inArray l;
        l.n=0;

        selector(t,l,s);

        // 清空缓冲区
        // cin.ignore(numeric_limits<streamsize>::max(),'\n');
        getchar();
    }
```

其中，selector函数在头文件selector.h中，其作为整体接口函数，做的是对输入字符串进行识别，根据识别的结果调用不同的子函数。具体来说，首先在输入字符串中查找特殊字符（如'.'、'#'等），根据找到的特殊字符调用不同的子函数。如果没有找到特殊字符，则认为输入的是标签名，调用tag1()函数处理。进一步调用同样在selector.h中的下述函数：

```
// 检索函数
void class1(tree t, inArray &l, char*s);
void class2(tree t, inArray &l, char*s);
void class0(tree t, inArray &l, char*s);
void classChild(tree t, inArray &l, char*s);
void id(tree t, inArray &l, char*s);
void all(tree t, inArray &l, char*s);
void tag1(tree t, inArray &l, char*s);
void tagClass(tree t, inArray &l, char*s);
void tag2(tree t, inArray &l, char*s);
void tagSon(tree t, inArray &l, char*s);
void tagChild(tree t, inArray &l, char*s);
void tagHeel(tree t, inArray &l, char*s);
void tagAfter(tree t, inArray &l, char*s);

//操作函数
void innerText(tree t, inArray &l);
void outerHTML(tree t, inArray &l);
void href(tree t, inArray &l);
```

其中，class2会调用class0，其他检索函数成并列关系。各个操作函数之间也为并列关系，会被各个检索函数调用。

3.详细设计

buildNodes函数的作用是创建树的结点。主要流程为：

1. 创建一个栈来记录当前未匹配的标签。
2. 遍历输入的顺序表，搜索每个HTML标签。
3. 如果遇到单标签，则直接将它作为一个结点添加到树中。
4. 如果遇到非单标签，则将其入栈。
5. 如果遇到结束标签，则将其匹配到与之对应的开始标签，并将这对标签作为一个结点添加到树中。
6. 如果遇到文本内容，则将其作为一个结点添加到树中。

buildChilden函数的主要流程为：

1. 遍历树的所有节点。
2. 对于每一个节点，检查它是否有子节点。如果有，则让当前节点成为它的第一个子节点。
3. 如果当前节点没有子节点，则跳过它。
4. 继续遍历下一个节点，直到遍历完整棵树。

此外，buildChilden函数还可能包含排序等其他操作，以保证子节点的排列顺序与文档中原有的排列顺序相同。

4.调试分析

（1）在代码实现过程中遇到了很多很多问题，解决后收获颇丰，主要包括：

- 构建树时下标的选取十分细节，需要一恰当的循环来跳到下一个合适的下标处
- 获取class和tag标签时，参数的正确传递花费了不少时间和心思，我最终使用指向线性表的指针来传递，而非一个字符串指针，规避了字符串处理操作带来的诸多问题
- 是否包含class标签的问题中涉及多标签的问题，只实现getClass函数是不够的，因此我进一步实现了hasClass函数，也更好地进行了函数封装
- 有关child的选择函数中对于标签范围的确定比较细节，需要同时使用指针比较和孩子结点遍历来确定；还需解决去重的问题
- 孩子链表构建过程中对空指针的处理需要格外小心，要添加多个合适的判断语句，避免访问空指针而报错

（2）算法的时空分析：

对于函数 buildNodes，它在 for 循环中执行了一个比较操作，并在循环结束时执行了一次 sort 操作，因此其时间复杂度为 O(n * log(n))。由于它不会动态分配内存，因此其空间复杂度为 O(1)。

对于函数 buildChilden，它在外层 for 循环中执行了一个比较操作，内层 for 循环也是一个比较操作，因此其时间复杂度为 O(n^2)。由于它会在内层 for 循环中执行动态内存分配，因此其空间复杂度为 O(n^2)，但实际上只会在一部分轮次内分配来构建各个孩子链表。

而各个选择器函数时空复杂度为O（n）或O（n^2），其中n为构建的结点总个数。

（3）经验和体会：本次实验代码量比较大，加起来大概1500行，实验过程中我深刻体会到了函数式编程和代码封装的重要性，一方面使得自己实现起来的思路更清晰，一方面便于调试，此外增加了代码的可读性。

5.用户使用说明

结合前面的需求分析部分，在提示`请输入您的检索语句（输入exit退出）：`时输入检索字符串，在提示`请输入你要进行的操作（text,html,href;输入q退出）`时输入操作字符串即可。

6. 测试结果

测试数据1

`#header-mobile` 开始执行id，选择id为 header-mobile 的元素 满足您要求的元素共 1 个。 在存储中的序号分别为：61 位的 div ; 请输入你要进行的操作（text,html,href;输入q退出） `html`

```
<div id="header-mobile">
                        <a href="/"><img src="/wp-
content/themes/rucnews/images/rucnews-logo.png" /></a>
```

```
                        <a class="back" href="/">返回首页</a>
                </div>
```

`text` 返回首页

测试数据2

`.homepage.more` 开始执行class2，选择class属性同时包含 homepage 和 more 的元素 满足您要求的元素共 1 个。 在存储中的序号分别为：35 位的 a ; 请输入你要进行的操作（text,html,href;输入q退出） `html`

```
    <a href="/" class="homepage more">新闻网首页</a>
```

`text` 新闻网首页

测试数据3

`div>a` 开始执行tagSon，选择父元素是

的所有 元素 满足您要求的元素共 27 个。 在存储中的序号分别为：35 位的 a ; 37 位的 a ; 39 位的 a ; 41 位的 a ; 43 位的 a ; 45 位的 a ; 47 位的 a ; 49 位的 a ; 51 位的 a ; 53 位的 a ; 55 位的 a ; 57 位的 a ; 59 位的 a ; 62 位的 a ; 64 位的 a ; 69 位的 a ; 79 位的 a ; 82 位的 a ; 85 位的 a ; 88 位的 a ; 93 位的 a ; 98 位的 a ; 122 位的 a ; 132 位的 a ; 136 位的 a ; 283 位的 a ; 346 位的 a ; 请输入你要进行的操作（text,html,href;输入q退出） `html`

```
<a href="/" class="homepage more">新闻网首页</a>
<a href="http://www.ruc.edu.cn" target="_blank" class="more">人大主页</a>
<a href="http://portal.ruc.edu.cn" target="_blank" class="more">数字人大</a>
<a href="mailto:leader@ruc.edu.cn" class="more">校长信箱</a>
<a href="/archives/category/camp_news/view" class="cnews dist2">广角</a>
<a href="/archives/category/camp_news/department_news" class="cnews">部处</a>
<a href="/archives/category/camp_news/institute_news" class="cnews">院系</a>
<a href="/archives/category/important_news/campus" class="inews dist">校园</a>
<a href="/archives/category/important_news/affairs" class="inews">校务</a>
<a href="/archives/category/important_news/exchange" class="inews">交流</a>
<a href="/archives/category/important_news/scholars" class="inews">学者</a>
<a href="/archives/category/important_news/students" class="inews">学生</a>
<a href="/archives/category/important_news/academic" class="inews">学术</a>
<a href="/"><img src="/wp-content/themes/rucnews/images/rucnews-logo.png" /></a>
<a class="back" href="/">返回首页</a>
<a href="/"><img src="/wp-content/themes/rucnews/images/rucnews-logo.png" /></a>
<a href="/archives/category/important_news">人大要闻</a>
<a href="/archives/category/special_news">专题新闻</a>
<a href="/archives/category/camp_news">校园时讯</a>
<a href="#">Latest<br>News</a>
<a href="#">Special<br>Topics</a>
<a href="#">Campus<br>News</a>
<a href="mailto:news@ruc.edu.cn" class="contribution"></a>
<a href="http://news.ruc.edu.cn">人大新闻网</a>
<a href="https://news.ruc.edu.cn/archives/category/special_news/20th">二十大专题
```

```
</a>
<a href="/">返回首页</a>
<a href="#"><img src="/wp-content/themes/rucnews/images/ruc-logo.png" /></a>
```

text 新闻网首页 人大主页 数字人大 校长信箱 广角 部处 院系 校园 校务 交流 学者 学生 学术 返回首页 人大要闻 专题新闻 校园时讯 Latest News Special Topics Campus News 人大新闻网 二十大专题 返回首页 href / http://www.ruc.edu.cn http://portal.ruc.edu.cn mailto:leader@ruc.edu.cn /archives/category/camp_news/view /archives/category/camp_news/department_news /archives/category/camp_news/institute_news /archives/category/important_news/campus /archives/category/important_news/affairs /archives/category/important_news/exchange /archives/category/important_news/scholars /archives/category/important_news/students /archives/category/important_news/academic / / / /archives/category/important_news /archives/category/special_news /archives/category/camp_news

mailto:news@ruc.edu.cn http://news.ruc.edu.cn https://news.ruc.edu.cn/archives/category/special_news/20th /

测试数据4

script+link

开始执行tagHeel，选择紧跟