**实验四 串与数组**

实验日期：2021年 4月 14日 姓名：刘宇诺 序号：

|  |  |
| --- | --- |
| **教师评语：** | **得分** |
|  |  |

一、实验目的

1. 掌握串的相关概念
2. 掌握串的存储结构及基本运算的实现
3. 运用串的特点解决应用问题。
4. 掌握数组的应用
5. 掌握掌握稀疏矩阵的表示方法及其运算的实现
6. 实现稀疏矩阵在三元组表示下的各运算并分析其效率

二、 实验题目

【实验题目1】利用串的堆分配顺序存储结构实现串的基本运算算法，并设计一个测试主函数进行测试：

1. 串的赋值（将一个字符串常量赋值给一个串）；
2. 求串的实际长度；
3. 两个串的比较：串1>串1，返回值大于0，串1==串2，返回值等于0，串1<串2，返回值小于0；
4. 将串清为空串；
5. 两个串的联接；返回长的新串
6. 求子串：返回主串中第i个位置开始的m个字符；
7. 求子串的位置：返回主串中子串第一次出现的位置序号。
8. 删除串中第i个字符开始的j个字符。

【实验题目2】稀疏矩阵的三元组顺序表示方法及基本操作的实现（建立、输出、转置）并实现一个主菜单来实现。

三、源程序

// 实验题目一

#include <cstdio>

#include <cstring>

#include <iostream>

#include <algorithm>

using namespace std;

struct SString

{

char\* data;

int len;

};

void init(SString &s); // 清空串 也是初始化

void setstr(SString &s, char\* ch); // 给串赋值

int getlen(SString s); // 获得串的长度

int SStrcmp(SString s, SString t); // 比较串s和串七

void SStrcat(SString &str, SString s, SString t); // 拼接两个串 返回一个新串

void subStr(SString &t, SString s, int pos, int len); // 求子串

int index(SString s, SString t); // 求子串位置 朴素匹配

void delStr(SString &s, int pos, int len); // 删除串中的某个子串

int main()

{

cout << "需要保证每个输入都是正确的" << endl;

char ch1[100], ch2[100];

SString s, t;

// 初始化

init(s), init(t);

cout << "分别对串 s 和串 t 进行赋值: " << endl;

scanf("%s%s", ch1, ch2);

puts("");

// 赋值操作

setstr(s, ch1), setstr(t, ch2);

// 输出

cout << "test 输出" << endl;

printf("%s\n%s\n", s.data, t.data);

puts("");

// 测试 获取字符串长度 函数

cout << "test 获取字符串长度" << endl;

cout << "s len: " << getlen(s) << '\n' << "t len: " << getlen(t) << endl;

puts("");

// 测试 字符串比较 函数

cout << "test 字符串比较" << endl;

cout << SStrcmp(s, t) << endl;

puts("");

// 将串清空

cout << "test 清空串 t 并输出被清空的串的内容和长度" << endl;

init(t);

printf("%s\t%d\n", t.data, t.len);

puts("");

init(s);

// 拼接串

cout << "test 拼接串 请输入两个串" << endl;

SString str;

scanf("%s%s", ch1, ch2);

setstr(s, ch1), setstr(t, ch2);

SStrcat(str, s, t);

printf("s: %s, t: %s, s + t: %s\n", s.data, t.data, str.data);

puts("");

// 求子串

cout << "test 求子串 请输入一个串" << endl;

scanf("%s", ch1);

setstr(s, ch1);

cout << "请输入要求子串的位置和长度" << endl;

int pos, len;

cin >> pos >> len;

subStr(t, s, pos, len);

printf("substring: %s\n", t.data);

puts("");

// 字符串匹配

cout << "test 字符串匹配(输出-1 表示未匹配) 请输入要进行匹配的俩个串" << endl;

scanf("%s%s", ch1, ch2);

init(s), init(t);

setstr(s, ch1), setstr(t, ch2);

cout << index(s, t) << endl;

puts("");

// 删除某个子串

cout << "test 删除子串 请输入一个主串" << endl;

scanf("%s", ch1);

init(s), setstr(s, ch1);

cout << "请输入要删除的位置和长度" << endl;

cin >> pos >> len;

delStr(s, pos, len);

printf("%s\n", s.data);

return 0;

}

void init(SString &s)

{

s.data = NULL;

s.len = 0;

}

void setstr(SString &s, char\* ch)

{

int n = strlen(ch);

s.len = n;

s.data = (char\*)malloc(sizeof(char) \* (n + 1));

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

s.data[i] = ch[i];

s.data[n] = '\0';

}

int getlen(SString s) // 在给字符串赋值的函数中最后设置了一个 '\0'

{

return s.len;

}

int SStrcmp(SString s, SString t)

{

int n = s.len, m = t.len;

for (int i = 0; i < n && i < m; i ++ )

if (s.data[i] != t.data[i])

return s.data[i] - t.data[i];

if (n == m) return 0;

return n - m;

}

void SStrcat(SString& str, SString s, SString t)

{

int n = s.len + t.len;

str.data = (char\*)malloc(sizeof(char) \* (n + 1));

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

str.data[i] = s.data[i];

for (int i = s.len, j = 0; j < t.len; i ++ , j ++ )

str.data[i] = t.data[j];

str.data[n] = '\0';

}

void subStr(SString &t, SString s, int pos, int len)

{

pos = pos - 1;

init(t);

t.data = (char\*)malloc(sizeof(char) \* (len + 1));

int cnt = 0;

for (int i = pos; i <= pos + len - 1; i ++ )

t.data[cnt ++ ] = s.data[i];

t.data[cnt] = '\0';

t.len = cnt;

}

int index(SString s, SString t) // 朴素匹配 未找到返回 -1

{

int n = s.len, m = t.len;

for (int i = 0; i <= n - m; i ++ )

{

int j = 0, k = i;

for ( ; j < m; j ++ , k ++ )

{

if (t.data[j] != s.data[k])

break;

}

if (j == m)

return i + 1;

}

return -1;

}

void delStr(SString &s, int pos, int len)

{

char ch[100];

pos -= 1;

int n = s.len;

int cnt = n - len;

int i = 0;

for (i = 0; i < pos; i ++ )

ch[i] = s.data[i];

if (pos + len - 1 >= n - 1)

{

ch[i] = '\0';

init(s);

setstr(s, ch);

return;

}

for (int j = pos + len; j < n; j ++ )

ch[i ++ ] = s.data[j];

ch[i] = '\0';

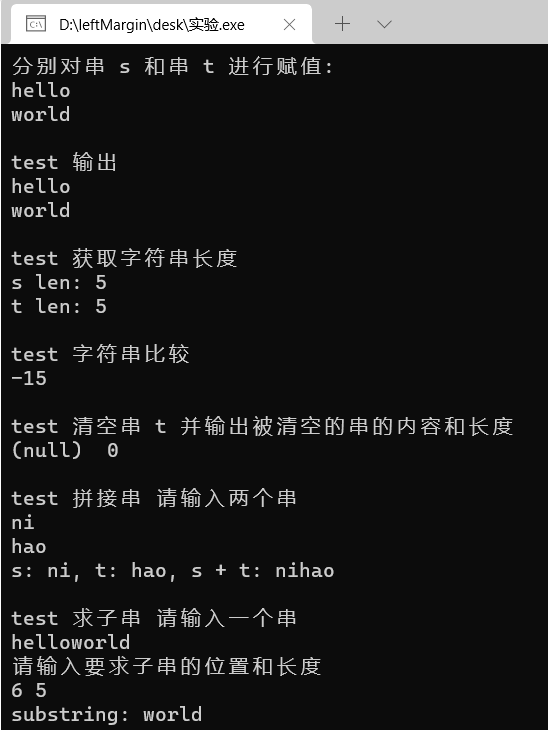
init(s);

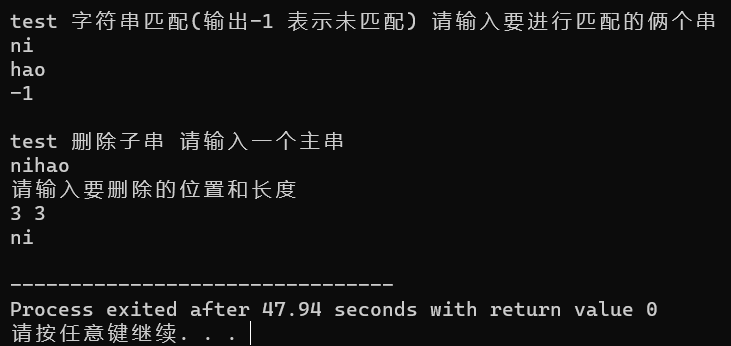
setstr(s, ch);

}

//

1. 实验结果(截图)





五、实验分析：

1.遇到的问题：

2.解决的方法：

3.收获：

4.实验体会及想对老师说的话