

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и
автоматизированных систем

Лабораторная работа №2

по дисциплине: Алгоритмы и структуры данных

тема: «Производные структуры данных.

Структура данных типа "строка" (Pascal/C)»

Выполнил: ст. группы ПВ-212
Гринченко Алина Сергеевна

Проверил:
Черников Сергей Викторович
Синюк Василий Григорьевич

Белгород 2022 г.

тема: «Производные структуры данных. Структура данных типа "строка" (Pascal/C)»

Вариант 4

Цель работы: изучение встроенной структуры данных типа «строка», разработка и использование производных структур данных строкового типа.

Задания

1. Для СД типа строка определить:

1.1 Абстрактный уровень представления СД:

1.1.1 Характер организованности и изменчивости: динамическая линейная последовательность..

1.1.2 Набор допустимых операций: операции доступа, операции инициализации, операции присваивания, операции сравнения, операция конкатенация.

1.2 Физический уровень представления СД:

1.2.1 Схему хранения: последовательная.

1.2.2 Объём памяти, занимаемый экземпляром СД: $V_{\text{стр}} = K + 1$, где K - — максимальное количество символов в строке.

1.2.3 Формат внутреннего представления СД и способ его интерпретации: последовательность из n однобайтовых значений кодов символов и числа 0 — признака окончания строки .

1.2.4 Характеристику допустимых значений: $CAR(string) = 1 + 256 + 256^2 + \dots + 256^K$, где K — максимальное количество элементов в строке.

1.2.5 Тип доступа к элементам: прямой.

1.3 Логический уровень представления СД.

1.3.1 Способ описания СД и экземпляра СД на языке программирования:

```
1 char *str;  
2 char str[]
```

2. Реализовать СД строкового типа в соответствии с вариантом индивидуального задания (см. табл.8) в виде модуля. Определить и обработать исключительные ситуации.

3. Разработать программу для решения задачи в соответствии с вариантом индивидуального задания (см. табл.8) с использованием модуля, полученного в результате выполнения пункта 2.

Заголовок: *function LastPost(s1,s2:string):word/ unsigned LastPost(string1 s1, string1 s2).*

Назначение: поиск последнего вхождения подстроки $s2$ в строку $s1$.

Входные параметры: $s1, s2$.

Выходные параметры: нет.

Формат 4 Реализация на языке C:

```

1  #if !defined(__FORM4_H)
2  #define __FORM4_H
3  const ...; // Определение исключительных ситуаций
4  typedef struct str
5  {
6      char s[1024];
7      unsigned N; // Динамическая длина строки
8  };
9  typedef str *string1;
10 void WriteToStr(string1 st, char *s);
11 void WriteFromStr(char *s, string1 st);
12 void InputStr(string1 st);
13 void OutputStr(string1 st);
14 int Comp(string1 s1, string1 s2);
15 void Delete(string1 s, unsigned Index, unsigned Count);
16 void Insert(string1 Subs, string1 s, unsigned Index);
17 void Concat(string1 s1, string1 s2, string1 srez);
18 void Copy(string1 s, unsigned Index, unsigned Count, string1 Subs);
19 unsigned Length(string1 s);
20 unsigned Pos(string1 SubS, string1 s);
21 int StrError; // Переменная ошибок
22 //...
23 #endif

```

Программа:

```

1  #include <stdio.h>
2  #include <assert.h>
3  #include <string.h>
4
5  //Операция прошла успешно
6  static const int STR_SUCCESSFUL = 0;
7
8  //Выход за границу максимально разрешенного размера строки
9  //при вводе в нее данных
10 static const int STRING_INPUT_ERROR = -1;
11
12 //Выход за границу максимально разрешенного размера строки
13 //при вставке данных из одной строки в другую
14 static const int STRING_INSERT_ERROR = -2;
15
16 //Попытка вставить элемент на место которое не существует
17 static const int STRING_NO_PLACE = -3;
18
19 //Выход за границу максимально разрешенного размера строки
20 //при склеивании в нее данных
21 static const int STRING_CONCATEN_ERROR = -4;
22
23 //Ошибка поиска в меньшей строки большей подстроки
24 static const int STRING_POS_ERROR = -5;
25
26 int STRING_ERROR;
27
28 #define MAX_L 1024
29
30 typedef struct {
31     char s[MAX_L];
32     unsigned N; // Динамическая (текущая) длина строки
33 } str;
34
35

```

```

36 typedef str string1[2];
37
38 //Запись данных в строку st из строки s
39 //Строка s заканчивается нулевым символом
40 void WriteToStr(string1 st, const char *s) {
41     int i = 0;
42     while (s[i] != '\0') {
43         st->s[i] = s[i];
44         i++;
45     }
46
47     st->N = i;
48     st->s[i++] = '\0';
49
50     if (i == MAX_L) {
51         STRING_ERROR = STRING_INPUT_ERROR;
52         assert(STRING_ERROR == -1);
53     }
54 }
55
56 //Запись данных в строку s из строки st
57 //Строка s заканчивается нулевым символом
58 void WriteFromStr(char *s, string1 st) {
59     int i = 0;
60     while (st->s[i] != '\0') {
61         s[i] = st->s[i];
62         i++;
63     }
64
65     st->N = i;
66     s[i] = '\0';
67 }
68
69 //Ввод строки s с клавиатуры
70 void InputStr(string1 st) {
71     char k = getchar();
72     unsigned short i = 0;
73
74     while ((k != EOF && k != '\n') && (i < MAX_L)) {
75         st->s[i] = k;
76         ++i;
77         k = getchar();
78     }
79
80     st->N = i;
81     st->s[i++] = '\0';
82
83     if ((i == MAX_L) && (k != EOF && k != '\n')) {
84         STRING_ERROR = STRING_INPUT_ERROR;
85         assert(STRING_ERROR == -1);
86     } else
87         STRING_ERROR = STR_SUCCESSFUL;
88
89 }
90
91 //Вывод строки s на экран монитора
92 void OutputStr(string1 st) {
93     int i = 0;
94     while (st->N != i) {
95         putchar(st->s[i]);

```

```

96         i++;
97     }
98 }
99
100 //Сравнивает строки s1 и s2 возвращает 0 если
101 //s1 == s2; 1 если s1 > s2; -1 если s1 < s2
102 int Comp(string1 s1, string1 s2) {
103     if (s1->N > s2->N)
104         return 1;
105     else if (s1->N < s2->N)
106         return -1;
107     else {
108         int i = 0;
109         while ((s1->s[i] == s2->s[i]) && (i < s1->N)) {
110             i++;
111         }
112
113         if ((i == s1->N) && (i == s2->N)) {
114             STRING_ERROR = STR_SUCCESSFUL;
115             return 0;
116         }
117
118         if (s1->s[i] > s2->s[i]) {
119             STRING_ERROR = STR_SUCCESSFUL;
120             return 1;
121         }
122     }
123 }
124 }
125
126 //Удаляет count символов из строки s
127 //начиная с позиции index
128 void Delete(string1 st, unsigned index, unsigned count) {
129     for (int i = index; i < st->N; i++) {
130         st->s[i] = st->s[i + count];
131     }
132
133     st->s[st->N - count] = '\0';
134     st->N = st->N - count;
135 }
136
137 //Вставляет подстроку subS в строку st
138 //начиная с позиции index
139 void Insert(string1 subS, string1 st, unsigned index) {
140     if (index > st->N)
141         STRING_ERROR = STRING_NO_PLACE;
142     else if (subS->N + st->N > MAX_L)
143         STRING_ERROR = STRING_INSERT_ERROR;
144     else {
145         unsigned i = st->N + 1;
146
147         while (i >= index) {
148             st->s[i + subS->N] = st->s[i];
149             --i;
150         }
151
152         i = index;
153         unsigned j = 0;
154
155         while (j < subS->N) {

```

```

156         st->s[i] = subS->s[j];
157         ++i;
158         ++j;
159     }
160
161     st->N = st->N + subS->N;
162 }
163
164     assert(String_Error == (-3 || -2));
165 }
166
167 //Выполняет конкатенацию строк s1 и s2 результат помещает в sRez
168 void Concat(string1 s1, string1 s2, string1 sRez) {
169     if (s1->N + s2->N > sRez->N)
170         String_Error = String_Concaten_Error;
171
172     //Скопируем в sRez первую строку
173     for (int i = 0; i < s1->N; i++) {
174         sRez->s[i] = s1->s[i];
175     }
176
177     unsigned i = s1->N;
178
179     //Соединим sRez со второй строкой
180     for (int j = 0; j < s2->N; j++) {
181         sRez->s[j + i] = s2->s[j];
182     }
183
184     sRez->N = s1->N + s2->N;
185     sRez->s[sRez->N] = '\0';
186
187     assert(String_Error == -4);
188 }
189
190 //Записывает count символов в строку subS из строки s
191 //начиная с позиции index
192 void Copy(string1 s, unsigned index, unsigned count, string1 subS) {
193     if (index + count > s->N || count > MAX_L)
194         String_Error = String_No_Place;
195     else {
196         unsigned i = 0;
197         unsigned rBord = index + count;
198
199         while (index < rBord) {
200             subS->s[i] = s->s[index];
201             ++i;
202             ++index;
203         }
204         subS->N = i;
205     }
206     assert(String_Error == -3);
207 }
208
209 //Возвращает текущую длину строки s
210 unsigned Length(string1 s) {
211     return s->N;
212 }
213
214 void reverse(string1 s) {
215     unsigned length = strlen(s->s);

```

```

216     char c;
217     int i, j;
218
219     for (i = 0, j = length - 1; i < j; i++, j--) {
220         c = s->s[i];
221         s->s[i] = s->s[j];
222         s->s[j] = c;
223     }
224 }
225
226 //Возвращает позицию начиная с которой в строке s
227 //располагается строка subS
228 unsigned Pos(string1 subS, string1 s) {
229     unsigned short j;
230     unsigned short lens = s->N;
231     unsigned short lensubS = subS->N;
232     unsigned short len = lens - lensubS;
233
234     if (lens < lensubS) {
235         STRING_ERROR = STRING_POS_ERROR;
236         assert(STRING_ERROR == -5);
237     }
238
239     for (unsigned short i = 0; i <= len; ++i) {
240         j = 0;
241         while ((j < lensubS) && (s->s[i + j] == subS->s[j]))
242             ++j;
243         if (j == subS->N)
244             return i + 1;
245     }
246
247     return 0;
248 }
249
250 //поиск последнего вхождения подстроки subS в строку s
251 unsigned LastPos(string1 subS, string1 s) {
252     reverse(subS);
253     reverse(s);
254
255     unsigned short j;
256     unsigned short lens = s->N;
257     unsigned short lensubS = subS->N;
258     unsigned short len = lens - lensubS;
259
260     if (lens < lensubS) {
261         STRING_ERROR = STRING_POS_ERROR;
262         assert(STRING_ERROR == -5);
263     }
264
265     for (unsigned short i = 0; i <= len; ++i) {
266         j = 0;
267         while ((j < lensubS) && (s->s[i + j] == subS->s[j]))
268             ++j;
269         if (j == subS->N)
270             return lens - i - lensubS;
271     }
272
273     return 0;
274 }
275

```

```

276
277 int main() {
278     string1 string_1, string_2;
279
280     InputStr(string_1);
281     InputStr(string_2);
282
283     printf("LastPos %d\n", LastPos(string_1, string_2));
284
285     return 0;
286 }

```

Код тестов:

```

1 void test_WriteToStr1() {
2     string1 st;
3     char *s = "abcd";
4     WriteToStr(st, s);
5     string1 expected = {
6         {"abcd", 4}
7     };
8
9     assert(Comp(st, expected) == 0);
10 }
11
12 void test_WriteToStr2() {
13     string1 st = {
14         {"errtudi", 7}
15     };
16     char *s = "abcd";
17     WriteToStr(st, s);
18     string1 expected = {
19         {"abcd", 4}
20     };
21
22     assert(Comp(st, expected) == 0);
23 }
24
25 void test_WriteToStr() {
26     test_WriteToStr1();
27     test_WriteToStr2();
28
29     printf("test_WriteToStr passed.\n");
30 }
31
32 void test_WriteFromStr1() {
33     string1 st = {
34         {"abcd", 4}
35     };
36
37     char s[10];
38     WriteFromStr(s, st);
39     char expected[] = "abcd";
40
41     assert(strcmp(s, expected) == 0);
42 }
43
44 void test_WriteFromStr2() {
45     string1 st = {
46         {"abcd", 4}
47     };

```



```

48
49     char s[2] = "rf";
50     WriteFromStr(s, st);
51     char expected[] = "abcd";
52
53     assert(strcmp(s, expected) == 0);
54 }
55
56 void test_WriteFromStr() {
57     test_WriteFromStr1();
58     test_WriteFromStr2();
59
60     printf("test_WriteFromStr passed.\n");
61 }
62
63 void test_Delete1() {
64     string1 st = {
65         {"abcdjddf", 8}
66     };
67
68     Delete(st, 1, 3);
69
70     string1 expected = {
71         {"addf", 4}
72     };
73
74     assert(Comp(st, expected));
75 }
76
77 void test_Delete2() {
78     string1 st = {
79         {"abcdjddf", 8}
80     };
81
82     Delete(st, 0, 2);
83
84     string1 expected = {
85         {"djddf", 5}
86     };
87
88     assert(Comp(st, expected));
89 }
90
91 void test_Delete() {
92     test_Delete1();
93     test_Delete2();
94
95     printf("test_Delete passed.\n");
96 }
97
98 void test_Insert1() {
99     string1 st = {
100         {"abcdjddf", 8}
101     };
102     string1 subS = {
103         {"LJ", 2}
104     };
105     string1 expected = {
106         {"aLJbcdjddf", 10}
107     };

```

```

108     Insert(subS, st, 0);
109
110     assert(Comp(expected, st));
111 }
112
113
114 void test_Insert2() {
115     string1 st = {
116         {"abcdjddf", 8}
117     };
118     string1 subS = {
119         {"LJ", 2}
120     };
121     string1 expected = {
122         {"abcLJdjddf", 10}
123     };
124
125     Insert(subS, st, 2);
126
127     assert(Comp(expected, st));
128 }
129
130 void test_Insert() {
131     test_Insert1();
132     test_Insert2();
133
134     printf("test_Insert passed.\n");
135 }
136
137 void test_Concat1() {
138     string1 s1 = {
139         {"abc", 3}
140     };
141     string1 s2 = {
142         {"LJ", 2}
143     };
144     string1 expected = {
145         {"abcLJ", 5}
146     };
147     string1 s3;
148     Concat(s1, s2, s3);
149
150     assert(Comp(s3, expected) == 0);
151 }
152
153 void test_Concat2() {
154     string1 s1 = {
155         {"abcgdeis", 8}
156     };
157     string1 s2 = {
158         {"s657n", 5}
159     };
160     string1 expected = {
161         {"abcgdeiss657n", 13}
162     };
163     string1 s3;
164     Concat(s1, s2, s3);
165
166     assert(Comp(s3, expected) == 0);
167 }

```

```

168
169 void test_Concat() {
170     test_Concat1();
171     test_Concat2();
172
173     printf("test_Concat passed.\n");
174 }
175
176 void test_Copy1() {
177     string1 s = {
178         {"abcgdeis", 8}
179     };
180     string1 subS = {
181         {"s657n", 5}
182     };
183     string1 expected = {
184         {"bc", 2}
185     };
186     Copy(s, 1, 2, subS);
187
188     assert(Comp(subS, expected) == 0);
189 }
190
191 void test_Copy2() {
192     string1 s = {
193         {"abcgdeis", 8}
194     };
195     string1 subS = {
196         {"s657n", 5}
197     };
198     string1 expected = {
199         {"cgde", 4}
200     };
201     Copy(s, 2, 4, subS);
202
203     assert(Comp(subS, expected) == 0);
204 }
205
206 void test_Copy() {
207     test_Copy1();
208     test_Copy2();
209
210     printf("test_Copy passed.\n");
211 }
212
213
214 void test_reverse1() {
215     string1 s = {
216         {"abcgdeis", 8}
217     };
218     string1 expected = {
219         {"siedgcba", 8}
220     };
221     reverse(s);
222
223     assert(Comp(s, expected) == 0);
224 }
225
226 void test_reverse2() {
227     string1 s = {

```

```

228     {"Alina", 5}
229 };
230 string1 expected = {
231     {"anilA", 5}
232 };
233 reverse(s);
234
235 assert(Comp(s, expected) == 0);
236 }
237
238 void test_reverse() {
239     test_reverse1();
240     test_reverse2();
241
242     printf("test_revers passed.\n");
243 }
244
245 void test_Pos1() {
246     string1 s1 = {
247         {"Alina girl", 10}
248     };
249     string1 s2 = {
250         {"Alina", 5}
251     };
252     unsigned i = 1;
253
254     assert(Pos(s2, s1) == i);
255 }
256
257 void test_Pos2() {
258     string1 s1 = {
259         {"girl Alina gtrsk", 16}
260     };
261     string1 s2 = {
262         {"Alina", 5}
263     };
264     unsigned i = 6;
265
266     assert(Pos(s2, s1) == i);
267 }
268
269
270 void test_Pos() {
271     test_Pos1();
272     test_Pos2();
273
274     printf("test_Pos passed.\n");
275 }
276
277 void test_LastPos1() {
278     string1 s1 = {
279         {"girl Alina gtrsk", 16}
280     };
281     string1 s2 = {
282         {"Alina", 5}
283     };
284     unsigned i = 5;
285
286     assert>LastPos(s2, s1) == i);
287 }

```

```

288
289 void test_LastPos2() {
290     string1 s1 = {
291         {"girl Alina Alina", 16}
292     };
293     string1 s2 = {
294         {"Alina", 5}
295     };
296     unsigned i = 11;
297
298     assert>LastPos(s2, s1) == i);
299 }
300
301 void test_LastPos() {
302     test_LastPos1();
303     test_LastPos2();
304
305     printf("test_LastPos passed.\n");
306 }
307
308 void TEST() {
309     test_WriteToStr();
310     test_WriteFromStr();
311     test_Delete();
312     test_Insert();
313     test_Concat();
314     test_Copy();
315     test_reverse();
316     test_Pos();
317     test_LastPos();
318 }

```

Результат работы программы и тестов:

```

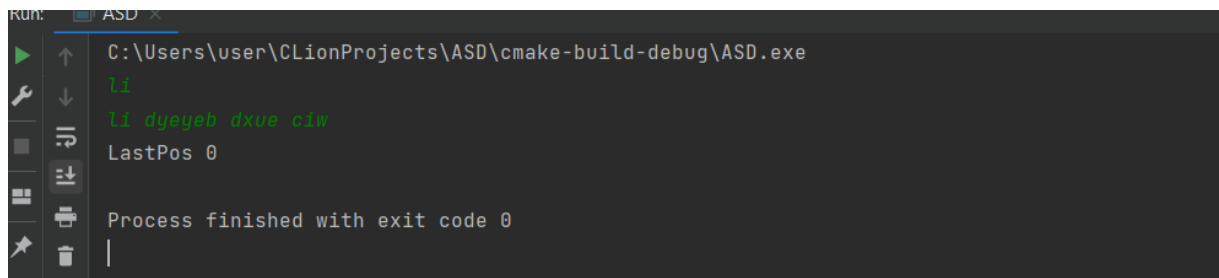
ASD x
C:\Users\user\CLionProjects\ASD\cmake-build-debug\ASD.exe
11
sed 11 fur
LastPos 4
Process finished with exit code 0

```

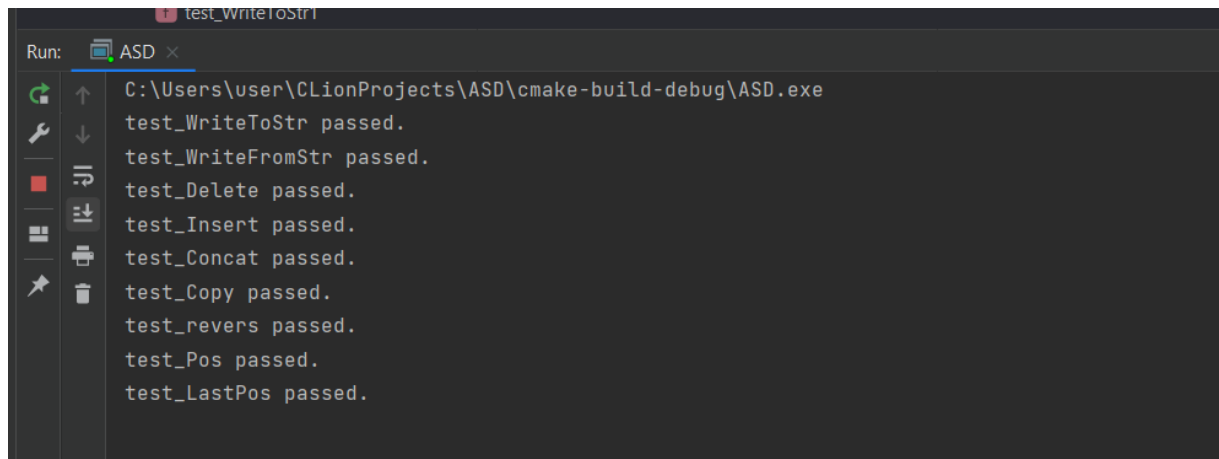
```

LastPos
Run: ASD x
C:\Users\user\CLionProjects\ASD\cmake-build-debug\ASD.exe
11
11 sude 11
LastPos 8
Process finished with exit code 0

```



```
Run: ASD x
C:\Users\user\CLionProjects\ASD\cmake-build-debug\ASD.exe
11
11 dyqyeb dxue cin
LastPos 0
Process finished with exit code 0
```



```
Run: ASD x
C:\Users\user\CLionProjects\ASD\cmake-build-debug\ASD.exe
test_WriteToStr passed.
test_WriteFromStr passed.
test_Delete passed.
test_Insert passed.
test_Concat passed.
test_Copy passed.
test_revers passed.
test_Pos passed.
test_LastPos passed.
```

Вывод: в ходе выполнения лабораторной работы мы изучили встроенные структуры данных типа «строка», разработали и использовали производные структур данных строкового типа.