ГУАП

КАФЕДРА № 14

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  | / |  | С.М. Пахарев |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 4 |
| Обработка символьных строк |
| по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 1842 |  | / |  | А.Д. Аничкин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2019

1. Цель работы

Изучение формата и правил описания символьных строк, а также методов обработки строковых данных.

1. Задание

Задача 4.2

В символьной строке удалить все слова, начинающиеся и заканчивающиеся на одну и ту же букву.

1. Формализация задачи

Для решения поставленной задачи мы используем функцию, объявленную нами, process\_line (обрабатывает массив в соответствии с условием задачи) для упрощения чтения программы. Также используется функция GETCHAR() и особенности таблицы ASCII, поэтому переменные c (текущий символ), prev\_c (предыдущий символ), E (первый символ слова) используют тип данных int. Переменные i, j, post\_i используют тип данных int, т.к. являются значениями отображающими позицию в слове. Переменные flag и found используют тип данных int, т.к. могут принимать только 2 целочисленных значения (константами YES, равной 1, и NO, равной 0). Массивы line и buffer имеют тип данных char, т.к. хранят в себе символы. Также они ограничены константой MAXLINE, равной 1000. Обработка массива ведётся в одном и том же массиве.

1. Тестовые примеры



1. Схема алгоритма
   1. Схема основного алгоритма

На лист 2

Рисунок 2 – cхема тела функции process\_line

A

j = 0

E = buffer[0]

flag = NO

found = NO

'Result:'

process\_line

Начало

Лист 1

5.2. Схема тела функции process\_line

Рисунок 1 – схема основного алгоритма

Конец

ВЫВОД line

ВВОД line

'Please, enter some words:'

Начало

MAXLINE = 1000

NO = 0

YES = 1

Лист 1

С листа 1

A

Лист 2

prev\_c = buffer[0]

i = 0

post\_i = -1

С листа 3

F

c = buffer[i]

and(c<'A')or(c>'Z'))and

(c<>' ')and(c<>'.')and

(c<>',')and(c<>'\n')and

(c<>'\0')and(c<>'\t')

((c<'a')or(c>'z')

На лист 3

На лист 3

На лист 3

На лист 3

Рисунок 2

E

D

C

B

inc post\_i

(buffer[i-1]= ' ')

j = i

flag = NO

E = c

Да

Да

Да

E-'A')or(prev\_c-'a'=E-'a')

or(prev\_c-'a'=E-'A')

or(prev\_c-'A'=E-'a'))

and(found<>YES)

Да

((prev\_c-'A'=

flag = YES

Да

flag = YES

buffer[post\_i] = c

inc post\_i

or(buffer[i-1]='.')

or(buffer[i-1]=',')

or(buffer[i-1]='\n')

or(buffer[i-1]='\0')

or(buffer[i-1]='\t')

or(c=',')or(c='\n')or

(c='\0')or(c='\t')

(c=' ')or(c='.')

found = YES

На лист 2

C листа 2

С листа 2

С листа 2

С листа 2

F

B

E

D

C

Конец

buffer[post\_i] = buffer[ j]

buffer[post\_i] = c

buffer[post\_i] = '\0'

inc post\_i

Да

c = '\0'

inc i

found = NO

inc j

inc post\_i

Да

j < i

Лист 3

1. Листинг программы
2. #pragma warning(disable : 4996) // отключение ошибки c4996
3. #include <stdio.h>
4. //
5. #define YES 1
6. #define NO 0
7. #define MAXLINE 1000
8. //
9. void process\_line(char buffer[]); // объявляем функцию
10. int main(void) //
11. {
12. char line[MAXLINE]; //
13. //
14. printf("Please, enter some words,\nWARNING: words that start and end on similar letter will be deleted\n");
15. gets(line); // считываем строку до конца файла и запихиваем в массив line
16. process\_line(line); // выполняем функцию над массивом line
17. printf("Result:\n");
18. puts(line); // выводим массив line
19. return 0;
20. }
21. void process\_line(char buffer[]) // тело функции
22. {
23. int c; // текущий символ
24. int prev\_c; // предыдущий символ
25. int flag; // признак разделения слов
26. int E; // первый символ
27. int i; // позиция символа
28. int post\_i; // позиция символа в финальном варианте
29. int j; // позиция начала слова
30. int found; // признак отсутствия слова
31. // ну понеслаааась
32. found = NO;
33. flag = NO;
34. E = buffer[0]; // присваиваем первый символ, чтобы не потерять его
35. j = 0;
36. prev\_c = buffer[0]; // та же причина
37. i = 0; // потому что массивы начинаются с нуля
38. post\_i = -1; // особенноесть for
39. // читаем строчки
40. do // цикл с постусловием
41. {
42. c = buffer[i];
43. if (((c < 'a' || c > 'z') && (c < 'A' || c > 'Z')) && (c != ' ' && c != '.' && c != ',' && c != '\n' && c != '\0' && c != '\t')) // проверка на не слово
44. {
45. found = YES;
46. }
47. if (c == ' ' || c == '.' || c == ',' || c == '\n' || c == '\0' || c == '\t') // проверка на конец слова
48. {
49. flag = YES; // нашли разделитель
50. if ((prev\_c - 'A' == E - 'A' || prev\_c - 'a' == E - 'a' || prev\_c - 'a' == E - 'A' || prev\_c - 'A' == E - 'a') && found != YES) // проверка на условие
51. {
53. }
54. else
55. {
56. if (buffer[i-1] == ' ' || buffer[i-1] == '.' || buffer[i-1] == ',' || buffer[i-1] == '\n' || buffer[i-1] == '\0' || buffer[i-1] == '\t') // если предыдущий символ разделитель, то...
57. {
58. ++post\_i;
59. buffer[post\_i] = c; // просто ставим разделитель
60. }
61. else
62. {
63. for (++post\_i; j < i; ++j) // перезапись слова в тот-же массив
64. {
65. buffer[post\_i] = buffer[j]; // присвоение
66. ++post\_i;
67. }
68. buffer[post\_i] = c; // устанавливаем разделитель между словами
69. }
70. }
71. found = NO;
72. }
73. else
74. {
75. if (flag == YES) // проверка на прошедший разделитель
76. {
77. E = c; // первый символ не разделитель
78. flag = NO; // обнуляем флаг
79. j = i; // запоминаем место первого символа
80. }
81. prev\_c = c; // запоминаем последний не разделитель
82. }
83. ++i;
84. }
85. while (c != '\0'); // цикл работает до конца файла
86. buffer[++post\_i] = '\0';// устанавливаем конец файла
87. }
88. Вывод; д-во правильности работы

 Программа работает, что подтверждают тестовые примеры и снимки экрана

