|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 6 |

**Название:**

Динамические структуры данных. Списки

**Дисциплина:** Объектно-ориентированное программирование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-22Б |  |  | Баканов Роман |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  |  |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

**Задание:**

Дана символьная строка длиной не более 80 символов, состоящая из слов, разделенных пробелом. Составить программу, которая вводит строку, организует из слов строки двунаправленный список, упорядоченный по алфавиту, выводит на экран список в прямом и обратном направлении. Определить количество слов в списке, длина которых меньше 5 символов. Вывести на экран найденные слова и их количество.

**Текст программы:**

#include <iostream>

struct str { char word[80]; str\* next; str\* prev; };

int main()

{

char s[80], \*l;

gets\_s(s);

printf("Original string:\n%s\n", s);

str \*op, \*ed, \* cur, \* nex;

cur = new str;

op = cur;

cur->prev = NULL;

l = strtok(s, " ");

while (l != NULL)

{

strcpy(cur->word, l);

nex = new str;

cur->next = nex;

nex->prev = cur;

cur = nex;

l = strtok(NULL, " ");

}

cur->next = NULL;

ed = cur->prev;

ed->next = NULL;

printf("String in alphabetical order:\n");

cur = op;

str\* ri = ed;

bool dot = false;

while ((!dot)&&(ri!=NULL))

{

dot = true;

while (cur != ri)

{

nex = cur->next;

bool ex = false;

int i = 0, l1, l2, ma = 0;

while (!ex)

{

((int)cur->word[i] < 97) ? (l1 = (int)cur->word[i] + 32) : (l1 = (int)cur->word[i]);

((int)nex->word[i] < 97) ? (l2 = (int)nex->word[i] + 32) : (l2 = (int)nex->word[i]);

if ((cur->word[i + 1] != '\0') && (nex->word[i + 1] != '\0'))

i++;

if (l1 != l2)

{

ma = l1 - l2;

ex = true;

}

if ((cur->word[i + 1] == '\0') || (nex->word[i + 1] == '\0'))

ex = true;

}

if (ma == 0)

ma = strcmp(cur->word, nex->word);

if (ma >= 0)

{

str\* tem = new str;

strcpy(tem->word, cur->word);

strcpy(cur->word, nex->word);

strcpy(nex->word, tem->word);

dot = false;

delete tem;

}

cur = cur->next;

}

ri = ed->prev;

cur = op;

}

cur = op;

while (cur!=NULL)

{

printf("%s ", cur->word);

cur = cur->next;

}

cur = ed;

printf("\nString in reverse alphabetical order:\n");

while (cur != NULL)

{

printf("%s ", cur->word);

cur = cur->prev;

}

cur = op;

printf("\nWord(s) with lenght less than 5:\n");

int co = 0;

while (cur != NULL)

{

if (strlen(cur->word) < 5)

{

printf("%s ", cur->word);

co++;

}

cur = cur->next;

}

printf("\nThere are %d word(s) with length less than 5", co);

cur = op;

delete cur->prev;

while (cur != NULL)

{

nex = cur->next;

delete cur;

cur = nex;

}

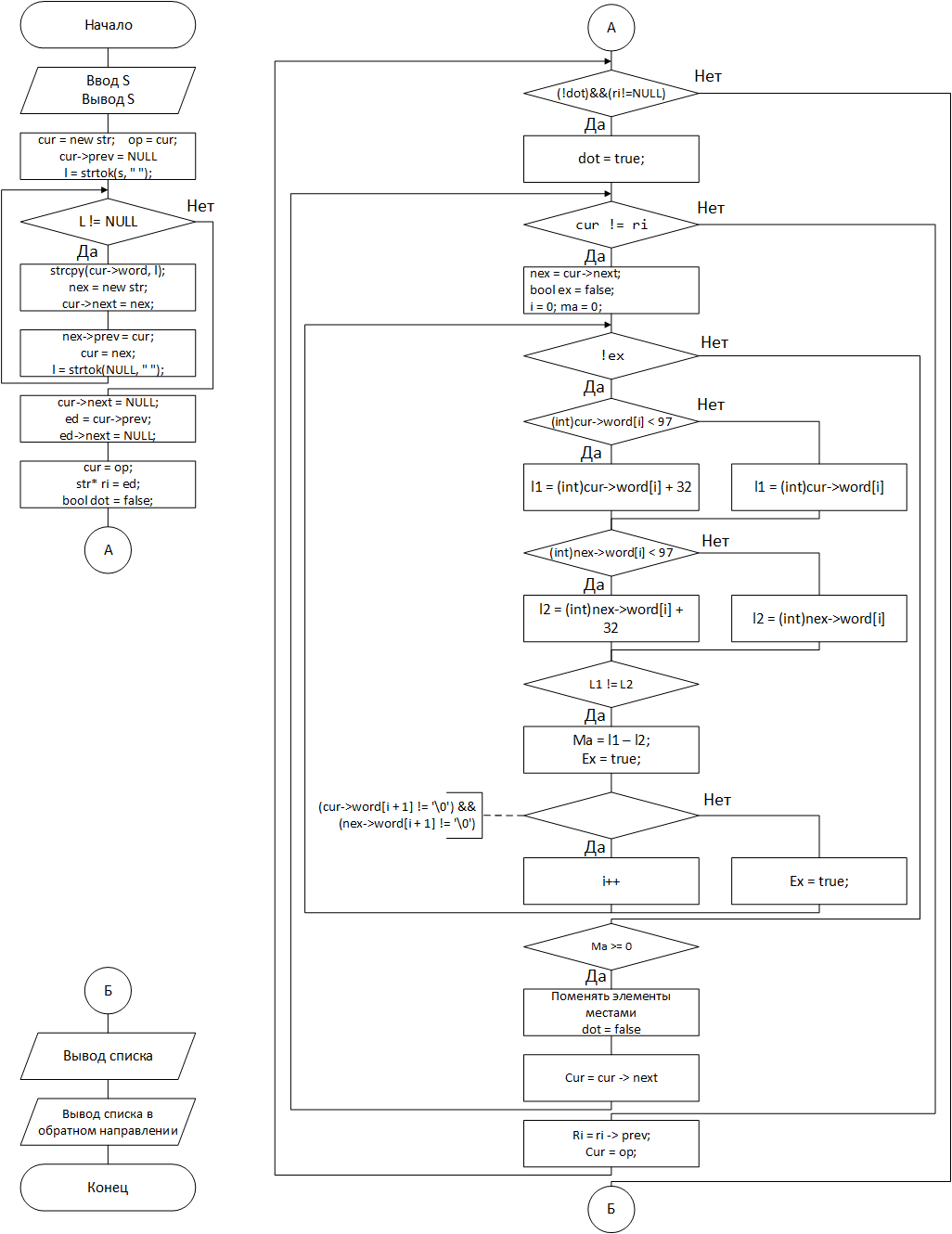
delete cur;

}

**Тесты:**

|  |  |  |
| --- | --- | --- |
| Исходные данные | Ожидаемый результат | Полученный результат |
| Its just a simple test nothing difficult | Original string:  Its just a simple test nothing difficult  String in alphabetical order:  a difficult Its just nothing simple test  String in reverse alphabetical order:  test simple nothing just Its difficult a  Word(s) with length less than 5:  a Its just test  There are 4 word(s) with length less than 5 | Original string:  Its just a simple test nothing difficult  String in alphabetical order:  a difficult Its just nothing simple test  String in reverse alphabetical order:  test simple nothing just Its difficult a  Word(s) with length less than 5:  a Its just test  There are 4 word(s) with length less than 5 |
| A bit Harder test but its going well so far | Original string:  A bit Harder test but its going well so far  String in alphabetical order:  A bit but far going Harder its so test well  String in reverse alphabetical order:  well test so its Harder going far but bit A  Word(s) with length less than 5:  A bit but far its so test well  There are 8 word(s) with length less than 5 | Original string:  A bit Harder test but its going well so far  String in alphabetical order:  A bit but far going Harder its so test well  String in reverse alphabetical order:  well test so its Harder going far but bit A  Word(s) with length less than 5:  A bit but far its so test well  There are 8 word(s) with length less than 5 |

**Схема алгоритма:**

****

**Выводы:**

Получены навыки работы с динамическими структурами данных.