|  |  |
| --- | --- |
| 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 |  |  | И.А. Люляев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  |  |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

Лабораторная работа № 6.

**Лабораторная работа 6. Динамические структуры данных. Списки**

Дана символьная строка, состоящая из слов, разделенных пробелом. Составить программу, которая вводит строку, организует из ее слов два однонаправленных списка (в первый входят все нечетные слова, а во второй – четные), определяет в каждом списке количество слов, начинающихся на гласные буквы, и определяет, в каком списке таких слов больше. Выводит на экран оба списка и результаты подсчетов и сравнений.

Код программы:

#include <iostream>

#include <string>

typedef struct elements {

elements\* el; // next

std::string st;

} elements;

int main() {

elements\* last0; elements\* p0; elements\* q0;

elements\* last1; elements\* p1; elements\* q1;

last0 = new elements; last0->el = NULL; last0->st = "LAST"; p0 = last0;

last1 = new elements; last1->el = NULL; last1->st = "LAST"; p1 = last1;

std::string str;

std::getline(std::cin, str);

int i = 0;

std::string t;

while (str[i]) {

if ((str[i] == ' ') && (t.size() != 0)) {

if (t.size() % 2 == 0) { // четные слова

q0 = new elements;

q0->st = t;

q0->el = p0;

p0 = q0;

t = "";

} else { // нечетные слова

q1 = new elements;

q1->st = t;

q1->el = p1;

p1 = q1;

t = "";

}

} else {

t.push\_back(str[i]);

}

if ((i == str.size()-1) && (t.size() != 0)) {

if (t.size() % 2 == 0) { // четные слова

q0 = new elements;

q0->st = t;

q0->el = p0;

p0 = q0;

t = "";

} else { // нечетные слова

q1 = new elements;

q1->st = t;

q1->el = p1;

p1 = q1;

t = "";

}

} // для обработки последнего символа вводимой строки (push\_back делается в else)

i++;

}

elements\* first0 = p0;

elements\* first1 = p1;

std::cout << "First list:" << std::endl;

q1 = first1; int num1 = 0; int j = 1;

while (q1->el) {

if ((q1->st[0] == 'a') || (q1->st[0] == 'e') || (q1->st[0] == 'i') || (q1->st[0] == 'o') || (q1->st[0] == 'u')) {

num1++;

}

std::cout << j << ". " << q1->st << std::endl;

p0 = q1;

delete p0;

q1 = q1->el;

j++;

}

std::cout << std::endl << "Second list:" << std::endl;

q0 = first0; int num0 = 0; j = 1;

while (q0->el) {

if ((q0->st[0] == 'a') || (q0->st[0] == 'e') || (q0->st[0] == 'i') || (q0->st[0] == 'o') || (q0->st[0] == 'u')) {

num0++;

}

std::cout << j << ". " << q0->st << std::endl;

p0 = q0;

delete p0;

q0 = q0->el;

j++;

}

std::cout << std::endl << "Words count in first list (1): " << num1 << std::endl;

std::cout << "Words count in second list (0): " << num0 << std::endl;

if (num1 > num0) {

std::cout << "More in first" << std::endl;

} else if (num0 > num1) {

std::cout << "More in second" << std::endl;

} else {

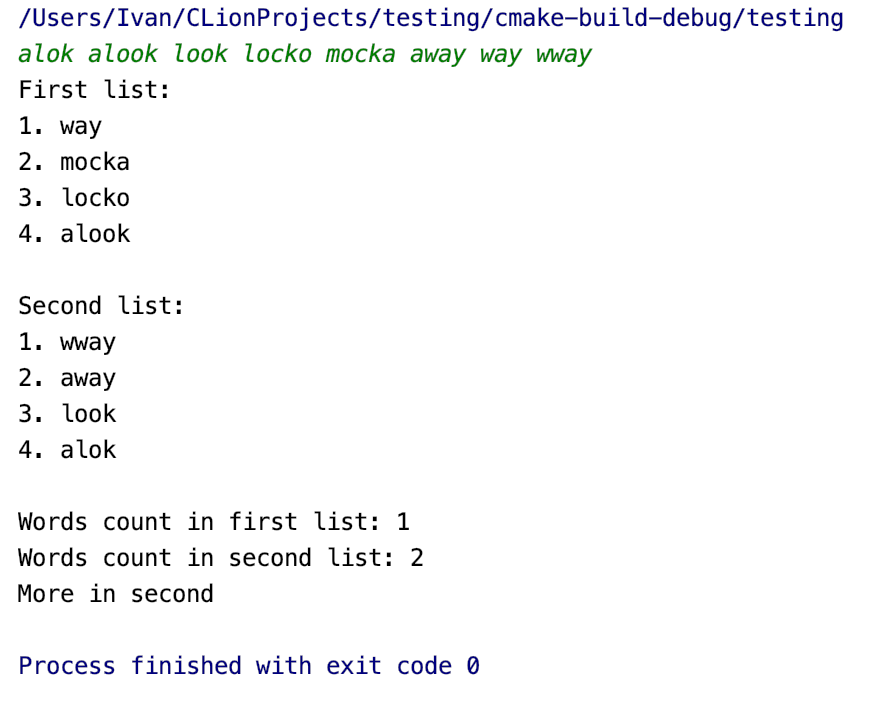
std::cout << "Equal number of this kind of words" << std::endl;

}

return 0;

}

Результат:



Вывод: Мы создали две структуры и произвели поиск среди элементов по заданным параметрам. При заданных исходных данных программа работает успешно. Мы научились работать со списками, с динамической памятью.

Схема алгоритма:

