МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ   
РОССИЙСКОЙ ФЕДЕРАЦИИ  
  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА(САМАРСКИЙ УНИВЕРСИТЕТ)

Институт информатики  
Кафедра программных систем

Дисциплина  
**Объектно-ориентированное программирование**

**ОТЧЕТ**по лабораторной работе № 1  
 **Объектно-ориентированное программирование обработки строк с использованием функций стандартной библиотеки языка С++ в стиле языка С.**

Вариант № 12

Студент: Мананников М.А.  
Группа: 6103-020302D  
  
Преподаватель: Баландин А.В.  
  
Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Самара 2022

**Текст задания**

Ввести с клавиатуры две символьные строки длиной не более 40 символов каждая. Строки должны быть введены посимвольно с использованием функции getchar(). Строки должны представлять собой произвольные абстрактные предложения, состоящие из слов. Слова отделяются друг от друга как минимум одним символом «пробел». Из исходных строк построить новую строку, в которой нечётные слова берутся из первого введённого предложения, а чётные – из второго. Когда слова в одном из исходных предложений закончатся раньше, чем в другом, его используют повторно с самого начала. Результирующая строка завершается, когда закончатся слова в самом длинном предложении.

**Объектно-ориентированный анализ (OOA)**

**Информационная модель предметной области**

Информационный объект «Предложение» (тип Сущность)

* Свойства:
* Строка;
* Набор слов;
* Процедуры:
* Создание предложения;
* Аннулирование предложения;

Информационный объект «Набор слов» (тип Связь)

* Свойства:
* Набор слов
* Процедуры:
* Построение нового предложения по набору слов

Информационный объект «Предложение» (тип Сущность)

* Свойства:
  + Символьная строка
* Процедуры:
  + Создание оформленного предложения
  + Построение набора слов по заданной символьной строке
  + Построение новой символьной строки по заданному набору слов
  + Вывод новой строки предложения
  + Аннулирование предложения

**Объектно**-**ориентированное** **проектирование (OOD)**

Информационный объект «Предложение» (тип Сущность)

* Свойства:
  + Строка - динамический массив символов;
* Процедуры:
  + Создание строки - конструктор с параметрами;
  + Аннулирование строки – деструктор;

Информационный объект «Набор слов» (тип Связь)

* Свойства:
  + Набор слов - динамический массив;
* Процедуры:
  + Добавление слова в набор слов – метод;
  + Формирование нового отсортированного предложения – метод;

Информационный объект «Предложение» (тип Сущность)

* Свойства:
  + Символьная строка - динамический массив символов;
  + Набор слов — динамический массив;
* Процедуры:
  + Создание оформленного предложения – конструктор с параметрами;
  + Формирование набора слов предложения – метод;
  + Сортировка набора слов в лексикографическом порядке – метод;
  + Формирование строки предложения по набору слов;
  + Вывод строки предложения – метод;
  + Аннулирование предложения – деструктор;

**Объектно-ориентированное программирование (OOP)**

**Структура алгоритма программы**

1. Создание первого исходного предложения;
2. Создание второго исходного предложения;
3. Вывод исходного предложения;
4. Построение набора слов из исходных предложений;
5. Выделение памяти под результирующее предложение;
6. Цикл перебора исходных предложений в поисках слов;
7. Вывод результирующего предложения;
8. Завершение программы;

**Текст программы**

#define \_CRT\_SECURE\_NO\_WARNINGS // устранение предупреждения о нерекомендуемых функциях для старых менее безопасных функций

#pragma once // директива, позволяющая открыть исходный файл строго один раз

#include <iostream>

#include <string.h> // библиотека для работы со си-строками

#include <stdio.h> // стандартная библиотека ввода\вывода

#include <stdlib.h> // поиск, сортировка и тд

#include <Windows.h> // использовать функионал ОС

#include "Sentence.h"

char\* sep(Sentence s1, Sentence s2) // процедура программы работы с предложением

{

char\* str1 = s1.str; // инициализация строки символов, через объект класса

char\* str2 = s2.str;

char\* res = (char\*)malloc(1); // выделение памяти для символа

char\* stroka1[40]; // массив для записи в него слов из первой строки

char\* stroka2[40]; //массив для записи в него слов из второй строки

int count\_first = 0; // счетчик слов в первой строке

int count\_second = 0; // счетчик слов во второй строке

int count\_add = 0; // счетчик добавленных слов для равного количества слов в первой и второй строках

int j = 0; // переменная для счетчика

int g1 = 0; // переменная для записи наибольшего кол-ва слов для выделения длины результирующей строки

int i = 0;

int h = 0;

char\* p1 = str1;

p1 = strtok(p1, " \t\n,.:-"); // разделение первой строки на слова

while (p1) //инициализация массива слов первой строки

{

stroka1[count\_first] = p1;

p1 = strtok(NULL, " \t\n,.:-"); // разделение первой строки на слова

count\_first++;

}

char\* p2 = str2;

p2 = strtok(p2, " \t\n,.:-"); // разделение второй строки на слова

while (p2) //инициализация массива слов второй строки

{

stroka2[count\_second] = p2;

p2 = strtok(NULL, " \t\n,.:-"); // разделение второй строки на лексемы

count\_second++;

}

int count\_second\_2 = count\_second;

int count\_first\_1 = count\_first;

if (count\_first >= count\_second) //уравнивание количества слов в первой и второй строке

{

while ((count\_second + count\_add) != count\_first)

{

if (j == count\_second) // обнуление счетчика в случае, если кол-во слов во 2 стороке равно кол-ву добавленных

{

j = 0;

}

stroka2[count\_second\_2] = stroka2[j]; // сортировка слов на четные

count\_add++;

count\_second\_2++;

j++;

}

g1 = count\_first; // присваивание переменной кол-во слов в первой строке

}

else

{

while ((count\_first + count\_add) != count\_second)

{

if (j == count\_first)

{

j = 0;

}

stroka1[count\_first\_1] = stroka1[j]; // сортировка на нечетные слова

count\_add++;

count\_first\_1++;

j++;

}

g1 = count\_second; // присваивание переменной кол-во слов во второй строке

}

count\_add = 0;

while (count\_add < 2 \* g1) // цикл посимвольной записи результирующей строки

{

char\* len = stroka1[i];

for (int i = 0; i < strlen(len); i++) // цикл посимвольной записи слова из первой строки в результирующую строку

{

res[h] = len[i];

h++;

res = (char\*)realloc(res, h + 1); // увеличивание памяти динамического массива для возможности записи символов

}

res[h] = ' '; // разделение между словами

h++;

res = (char\*)realloc(res, h + 1); // увеличивание памяти динамического массива для возможности записи символов

count\_add++;

len = stroka2[i];

for (int i = 0; i < strlen(len); i++) //цикл посимвольной записи слова из второй строки в результирующую строку

{

res[h] = len[i];

h++;

res = (char\*)realloc(res, h + 1); // увеличивание памяти динамического массива для возможности записи символов

}

res[h] = ' ';

h++;

res = (char\*)realloc(res, h + 1); // увеличивание памяти динамического массива для возможности записи символов

count\_add++;

i++;

}

res[h] = 0; // конец строки

return res;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Rus"); // подключение русского языка

puts("Лабораторная работа 1 (12 вариант)\n");

puts("Подготовил студент группы 6103 Мананников Максим\n");

Sentence s1; // создание объекта класса(первой строки)

Sentence s2; // создание объекта класса(второй строки)

Sentence ress(1);// результирующая строка

ress.str = sep(s1, s2); // формирование результирующей строки

ress.print(); // вывод строки

system("pause"); //задержка консоли на экране при завершении программы

return 0;

}

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string.h> //библиотека для работы со си-строками

#include <stdio.h> // стандартная библиотека ввода\вывода

#include <stdlib.h> // поиск, сортировка и тд

#include <Windows.h> // использовать функионал ОС

using namespace std;

class Sentence // объявление класса Sentence

{

public: char\* str = (char\*)malloc(1); // выделение памяти для 1 символа строки

public: Sentence() // конструктор без параметров, инициализирующий поле

{

char p = 1; // переменная для складывания в нее введенного символа

int i = 0; // счетчик символов в строке

puts("Введите строку:");

while ((p = getchar()) != '\n' && (i <= 40)) // цикл для проверки наличия слов и не превышения 40 символов

{

str[i] = p;

i++;

str = (char\*)realloc(str, i + 1); // увеличивание памяти динамического массива для возможности записи символов

}

str[i] = 0; // конец строки

}

public: Sentence(int k) // конструктор c параметром, создающий пустую строку

{

}

~Sentence() // вызов деструктора

{

free(str); // освобождение памяти

}

void print() // процедура печати строки

{

puts(str); // вывод

}

};

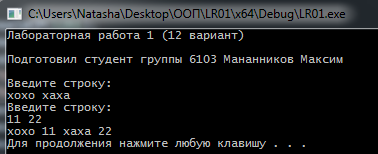


Рисунок 1 – Пример выполнения задания