**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА (САМАРСКИЙ УНИВЕРСИТЕТ)»**

ИНСТИТУТ ИНФОРМАТИКИ И КИБЕРНЕТИКИ

Кафедра программных систем

Дисциплина

**Объектно-ориентированное программирование**

**ОТЧЕТ**

по лабораторной работе № 2

**.** **<Объектно-ориентированное программирование последовательного доступа к текстовым файлам с использованием функций стандартной библиотеки С++ в стиле языка C>**

Вариант № 3

Студент: Боряков Н.С.

Группа: 6102-020302D

Преподаватель: Баландин А.В.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата:

Самара 2022

***Цель лабораторной работы:*** изучить и практически освоить при разработке программы использование указателей на функции, а также стандартные функции языка С++ в стиле языка С для работы с файлами в текстовом режиме, стандартные функции поиска и сортировки элементов в массивах произвольных типов, а также правила преобразования в языке С++ стандартных типов при использовании в выражениях разнотипных данных.

Порядок выполнения

1. Изучить следующие стандартные функции языка С++ в стиле языка С2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fopen() | fputc() | fgetc() | fputs() | fgets() |
| printf() | scanf() | fprintf() | fscanf() | sprintf() |
| qsort() | lfind() | lsearch() | bsearch() | sscanf() |
| feof() | ferror() |  |  |  |

2. Осуществить *объектно-ориентированный анализ* (**ООА**) структуры, представленной в задании предметной области программы - «*Текстовый\_файл*», и построить её информационную структуру, моделирующую предметную область посредством информационных объектов-"сущностей" и объектов-"связей". Представить и обосновать построенную информационную модель преподавателю. Пример оформления результата объектно-ориентированного анализа и проектирования программы приведён в Приложении 2;

3. Осуществить *объектно-ориентированное проектирование* (**ООD**). Определить абстрактные типы для информационных объектов: типы свойств, типы процедур;

4. Написать объектно-ориентированную программу, реализующую полученный вариант задания в виде консольного приложения в среде Visual Studio или любой другой среде программирования на языке С++.

*ЗАДАНИЕ*

Написать программу, которая создаёт в текстовом режиме файл в текущем каталоге с именем, вводимым с клавиатуры, и заполняет его произвольными строками, которые также вводятся с клавиатуры. После ввода последней строки файл закрывается. Построить в оперативной памяти массив записей (структур) вида, анализируя строки файла. Отсортировать полученный массив по возрастанию значений поля "количество\_различных\_символов", результирующий массив вывести на экран. Для сортировки элементов массива использовать стандартную функцию языка С qsort().

**ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ АНАЛИЗ**

**Объектная модель предметной области**

**1**.**Информационный объект – "Слово" (тип Сущность)**

*Свойства:*

* Строка
* Длина\_Строки

*Процедуры:*

* Создать
* Аннулировать

**2**.**Информационный объект – "БуферСлов" (тип Сущность)**

*Свойства:*

* Буфер памяти
* Размер буфера

*Процедуры:*

* Создать БуферСлов заданного размера
* Отсортировать слова в БуферСлов
* Аннулировать БуферСлов Длина\_Строки

**3**.**Информационный объект – "ФайлСлов" (тип Сущность)**

*Свойства:*

* Имя файла
* Расширение имени
* Расположение файла в файловой системе (Путь)
* Буфер ввода/вывода слов

*Процедуры:*

* Создать ФайлСлов
* Открыть для записи
* Открыть для чтения
* Записать Слово
* Прочитать Слово Длина\_Строки

**ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОЕКТИРОВАНИЕ**

**Спецификация классов**

**1**. **Класс СЛОВО**

*Свойства:*

* Символьная\_строка\n – Массив символов с \0.
* Размер\_буфера\_строки – Целое число.

*Процедуры:*

* Создать\_слово – Конструктор
* Аннулировать\_слово – Деструктор. Аннулировать

**2**. **Класс –БуферСлов**

*Свойства:*

* Буфер памяти - Массив символов.
* Размер буфера – Целое число.

*Процедуры:*

* Создать БуферСлов заданного размера – Конструктор.
* Отсортировать слова в БуферСлов – Метод.
* Аннулировать БуферСлов – Деструктор.

**3**. **Класс – ФайлСлов**

*Свойства:*

* ИмяФайла – Массив символов с \0.
* БуферСлов – Массив слов.

*Процедуры:*

* Создать ФайлСлов – Конструктор.
* ОткрытьПотокДляЗаписиСлов – Метод.
* ОткрытьПотокДляЧтенияСлов – Метод.
* ЗаписатьСлово – Метод.
* ПрочитатьСлово – Метод.
* СкопироватьСловаФайла\_ в\_БуферСлов.
* СкопироватьБуферСлов\_в\_Файл.
* ЗакрытьФайлСлов.
* АннулироватьФайлСлов.

**Спецификация алгоритма программы**

1. Создание Файла\_Слов;
2. Ввод количество вводимых строк в Файле\_Слов;
3. Заполнение строк Файла\_Слов Буфером\_Слов;
4. Чтение строк из Файла\_Слов с последующим заполнением в Исходный\_массив стрктур;
5. Вывод Исходного\_массива структур;
6. Чтение и сортировка строк из Файла\_Слов с последующим заполнением в Конечный\_массив стрктур;
7. Вывод Конечного\_массива структур;
8. Завершение работы.

**ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ**

#define \_CRT\_SECURE\_NO\_WARNINGS

#pragma once

#include <iostream>

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include <Windows.h>

using namespace std;

void Add(char \*str)

{

int index = 0; // счётчик символов

char element; // определение пременной для символа

while ((element = getchar()) != '\n') // введенный элемент не равен символу перехода

{

if (index < 80) // заполнение массива символов при кол-ве элементов в нем меньше 80

{

str[index] = element; // добавление элемента в массив

index++; // обновление счётчика для перехода к новому элементу

}

}

str[index] = '\0'; // последний элемент

}

int comp(const void\* i, const void\* j) // по возрастанию

{

return \*(int\*)i - \*(int\*)j; // преобразование указателей

}

struct Bear

{

int \_data;

int \_numer;

};

class Sentences

{

char\* name;

FILE\* fp;

int\* arrdata;

int k;

struct Bear structur[80];

public:

Sentences()

{

name = (char\*)malloc(81);

Add(name);

fp = fopen(name, "wt");

Strok();

Quantity();

}

~Sentences()

{

if (name != NULL)

{

free(name);

name = NULL;

}

if (fp != NULL)

{

free(fp);

fp = NULL;

}

}

void Qsort()

{

int comp(const void\*, const void\*); // прототип функции

qsort((int\*)arrdata, k, sizeof(int), comp);

for (int i = 0; i < k; i++)

{

structur[i].\_data = arrdata[i];

structur[i].\_numer = i + 1;

}

}

void Print()

{

for (int i = 0; i < k; i++)

{

printf("Номер\_строки:%d | \t | Количество\_различных\_символов: %d | \n", structur[i].\_numer, structur[i].\_data);

}

}

private:

void Strok()

{

printf("Введите количество строк: ");

int count = 0;

scanf("%d", &count);

cin.ignore(); // чистка буфера

for (int i = 0; i < count; i++)

{

printf("%d: ", i + 1);

char\* strok = (char\*)malloc(81);

Add(strok);

fprintf(fp, "%s\n", strok);

free(strok);

}

fclose(fp);

}

void Quantity()

{

fp = fopen(name, "rt");

arrdata = (int\*)malloc(81);

char\* slenght = (char\*)malloc(81);

k = 0;

int\* index = &k;

\*index = 0;

while (!feof(fp))

{

if (fgets(slenght, 81, fp) != NULL)

{

printf("%s ", slenght);

int data = strlen(slenght);

arrdata[k] = data - 1;

printf("%d\n", arrdata[k]);

k++;

}

}

char\* word = (char\*)malloc(81);

for (int i = 0; i < k; i++)

{

structur[i].\_data = arrdata[i];

structur[i].\_numer = i + 1;

}

fclose(fp);

}

};

int main()

{

printf("Hallo world\n");

system("chcp 1251"); // Использовать кириллицу для консольного ввода/вывода

printf("Введите имя файла\n");

Sentences s;

s.Print();

s.Qsort();

printf("\nОтсортированный массив: \n");

s.Print();

system("pause");//Задержать консоль на экране перед завершением программы

return 0;

}

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 – Пример работы программы