Московский Авиационный институт
(Национальный исследовательский университет)

Факультет №8
«Компьютерные науки и прикладная математика»

Кафедра 806
«Вычислительная математика и программирование»

Дисциплина
«Практикум программирования»

Курсовая работа №6

по теме

«Обработка последовательной файловой структуры на языке C++»

Студент:	Ефименко К.И.
Группа:	М8О-109Б-22
Преподаватель:	Сысоев М. А.
Подпись:	
Оценка:	

Москва, 2023

Постановка задачи

Разработать последовательную структуру данных для представления простейшей базы данных на файлах в СП Си. Составить программу для генерации внешнего нетекстового файла заданной структуры, содержащего представительный набор записей (15-20). Распечатать содержимое сгенерированного файла в виде таблицы и выполнить над ним заданное действие для 2-3 значений параметров запроса, распечатать результат.

Вариант 42

Содержимое и структура файла:

Общая информация о выпускниках школы студента: фамилия, инициалы, пол, номер класса, буква класса, в каком ВУЗе учится, где работает, в каком полку служит и т.п.

Задание:

Выяснить, имеются ли однофамильцы в каком-нибудь классе.

Серверы и рабочие станции

Серверы и рабочие станции от персональных компьютеров отличаются в основном более высокой мощностью и производительностью.

Рабочая станция часто используется для решения сложных прикладных, с чем не справится обычный компьютер, также она подключена к локальной сети.

Сервер может выполнять разные функции, хранить информацию, обрабатывать многочисленные запросы извне, обслуживать офисную технику и многое другое.

По комплектации сервер и рабочую станцию отличают типы процессоров (в них больше ядер, они надёжнее), для них устанавливается своя операционная система. Может быть больше одного процессора. В зависимости от назначения к ним может быть подключено много периферийных устройств.

Код программы

Row.hpp

```
#ifndef ROW_HPP_INCLUDED
#define ROW_HPP_INCLUDED

#include <iostream>
#include <string>

class DataBase;

class Row {
    friend class DataBase;

private:
    std::string surname;
    std::string initials;
    std::string sex;
    size_t class_number;
    std::string class_letter;
    std::string university;
    std::string job;

public:
    Row() = default;
    ~Row() = default;
    Row(
            const std::string& surname,
            const std::string& initials,
            const std::string& sex,
            const size_t class_number,
            const std::string& class_letter,
            const std::string& university,
            const std::string& job
        );

    friend std::ostream& operator<<(std::ostream& out, const Row& stud
ent);
    friend std::istream& operator>>(std::istream& in, Row& student);

    friend class Database;
};

#include "../src/Row.cpp"

#endif
```

Row.cpp

```
#include "../include/Row.hpp"

Row::Row(
            const std::string& surname,
            const std::string& initials,
            const std::string& sex,
            const size_t class_number,
            const std::string& class_letter,
            const std::string& university,
            const std::string& job
        ) {
```

```
            this->surname = surname;
            this->initials = initials;
            this->sex = sex;
            this->class_number = class_number;
            this->class_letter = class_letter;
            this->university = university;
            this->job = job;
        }

std::ostream& operator<<(std::ostream &out, const Row& students) {
    out.write(students.surname.c_str(), students.surname.length() + 1)
;
    out.write(students.initials.c_str(), students.initials.length() +
1);
    out.write(students.sex.c_str(), students.sex.length() + 1);
    out.write((char *) &students.class_number, sizeof(students.class_n
umber));
    out.write(students.class_letter.c_str(), students.class_letter.len
gth() + 1);
    out.write(students.university.c_str(), students.university.length(
) + 1);
    out.write((char *) &students.job, sizeof(students.job));

    return out;
}


std::istream& operator>>(std::istream &in, Row &students) {
    getline(in, students.surname, '\0');
    getline(in, students.initials, '\0');
    getline(in, students.sex, '\0');
    in.read((char *) &students.class_number, sizeof(students.class_num
ber));
    getline(in, students.class_letter, '\0');
    getline(in, students.university, '\0');
    getline(in, students.job, '\0');
    return in;
}
```

DataBase.hpp

```
#ifndef DATABASE_HPP_INCLUDED
#define DATABASE_HPP_INCLUDED

#include "Row.hpp"
#include <iostream>
#include <string>

class DataBase {
private:
    std::string path;

public:
    DataBase() = default;
    DataBase(const std::string& file_name);
    ~DataBase() = default;

    std::string create_file(const std::string& file_name = "db_1");
    void delete_file();
    void add_data(const Row& student);
    std::string make_dir(const std::string& dir_name);
    void task(const size_t class_number, const std::string& class_lett
er) const;
    void print() const;
```

```
};

#include "../src/DataBase.cpp"

#endif
```

DataBase.cpp

```
#include "../include/DataBase.hpp"
#include "../../KP7/include/Vector.hpp"
#include <iostream>
#include <fstream>
#include <string>
#include <filesystem>

DataBase::DataBase(const std::string& filename) {
    path = create_file(filename);
}

std::string DataBase::make_dir(const std::string& dir_name) {
    std::string dir_path = std::string("/Users/kirille/Desktop/Kolokvi
ym/MAI_109B_22/Efimenko/KP6/dirs/" + dir_name);

    if (!std::filesystem::is_directory(dir_path.c_str())) {
        std::filesystem::create_directories(dir_path.c_str());
    }

    return dir_path;
}

std::string DataBase::create_file(const std::string& filename) {
    std::fstream file;
    std::string curr_path = make_dir("Databases") + "/" + filename + "
.txt";
    file.open(curr_path, std::ios::app);
    if (!file) {
        throw std::runtime_error("Problems with file creation");
    }
    file.close();
    return curr_path;
}

void DataBase::delete_file() {
    std::fstream file;
    if (file) file.close();
    remove(path.data());
}

void DataBase::add_data(const Row& student) {
    std::fstream file;
    if (!file) {
        throw std::runtime_error("File doesn't exist");
    }
    file.open(path.data(), std::ios::out | std::ios::app);
    file << student;
    file.close();
}

bool find(const Vector<std::string>& vec, const std::string& value) {
    for (size_t i = 0; i < vec.get_size(); ++i) {
        if (vec[i] == value) {
            return true;
        }
    }
```

```
    return false;
}

void DataBase::task(const size_t class_number, const std::string& clas
s_letter) const {
    std::fstream file;
    Vector<std::string> families;
    file.open(path.data(), std::ios::in);
    Row student;
    while (file.peek() != EOF) {
        file >> student;
        if (find(families, student.surname)) {
            std::cout << "namesakes is exist\n";
            file.close();
            return;
        }
        families.push_back(student.surname);
    }
    std::cout << "namesakes doesn't exist\n";
    file.close();
}
```

main.cpp

```
#include "../include/DataBase.hpp"

int main() {
    DataBase db("students");
    Row r1 (
            "Kudryashov",
            "B.A.",
            "M",
            11,
            "A",
            "HSE",
            "RZD"
    );

    Row r2 (
            "Valentinov",
            "S.T.",
            "M",
            9,
            "D",
            "MAI",
            "Yandex"
    );
    Row r3(
           "Valentinov",
            "Y.T.",
            "M",
            9,
            "D",
            "Innopolis",
            "Tinkoff"
    );
    Row r4(
            "Samarskaya",
            "U.I.",
            "W",
            11,
            "A",
            "HSE",
```

```
            "RZD"
    );
    Row r5(
            "Petrov",
            "B.A.",
            "M",
            11,
            "C",
            "MFTI",
            "RZD"
    );

    size_t number_of_classes = 9;
    std::string letter_of_class = "D";

    db.add_data(r1);
    db.add_data(r2);
    db.add_data(r3);
    db.add_data(r4);
    db.add_data(r5);

    db.task(number_of_classes, letter_of_class);
}
```

Заключение

В задании №6 курсовой работы я научился работать с файлами и обрабатывать файловые структуры. Я построил простейшую СУБД на бинарных файлах с возможностью вывода всех данных в виде таблицы, добавления в неё записей, удаления данных, поиска данных в таблице по определенному параметру. Полученные знания работы с файлами и структурами, а также основы работы с базами данных обязательно пригодятся мне в будущем.

Список использованной литературы

1. http://cppstudio.com/post/1253/
2. https://www.avk-company.ru/articles/11/
3. http://sqlfiddle.com/#!9/f23fe0/3
4. https://itvdn.com/ru/blog/article/m-sql#5
5. https://sql-language.ru/select-where.html