**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: | Концебалов О.С. |
| 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 1**

**Содержимое и структура файла:** Сведения о составе комплектующих

личных ПЭВМ в студенческой группе: фамилия владельца, число и тип процессоров, объём памяти, тип видеоконтроллера (встроенный, внешний, AGP, PCI) и объём видеопамяти, тип (SCSI/IDE, ATA/SATA), число и ёмкость винчестеров, количество интегрированных контроллеров и внешних (периферийных) устройств, операционная система.

**Задание:** найти всех владельцев двухпроцессорных компьютеров, имеющих не более p внешних устройств

**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**

**RowTable.hpp**

#ifndef COMPUTER\_HPP

#define COMPUTER\_HPP

#include <iostream>

#include <string>

class DataBase;

class RowTable{

    friend class DataBase;

private:

    std::string owner\_surname;

    size\_t num\_processors;

    std::string processor\_type;

    size\_t memory\_sz;

    std::string video\_controller\_type;

    size\_t video\_memory\_sz;

    std::string hard\_driver\_type;

    size\_t num\_hard\_drivers;

    size\_t hard\_driver\_memory;

    size\_t num\_integrated\_controllers;

    size\_t num\_external\_devices;

    std::string operating\_system;

public:

    RowTable() = default;

    RowTable(const std::string& owner\_surname, const size\_t& num\_processors, const std::string& processor\_type, const size\_t& memory\_sz,

            const std::string& video\_controller\_type, const size\_t& video\_memory\_sz, const std::string& hard\_driver\_type,

            const size\_t& num\_hard\_drivers, const size\_t& hard\_driver\_memory, const size\_t& num\_integrated\_controllers,

            const size\_t& num\_external\_devices, const std::string& operating\_system);

    ~RowTable() = default;

    void print() const;

};

#endif

**RowTable.cpp**

#include "RowTable.hpp"

#include <iostream>

#include <iomanip>

RowTable::RowTable(const std::string& owner\_surname, const size\_t& num\_processors, const std::string& processor\_type,

                const size\_t& memory\_sz, const std::string& video\_controller\_type, const size\_t& video\_memory\_sz,

                const std::string& hard\_driver\_type, const size\_t& num\_hard\_drivers, const size\_t& hard\_driver\_memory,

                const size\_t& num\_integrated\_controllers, const size\_t& num\_external\_devices, const std::string& operating\_system):

                owner\_surname(owner\_surname), num\_processors(num\_processors), processor\_type(processor\_type), memory\_sz(memory\_sz),

                video\_controller\_type(video\_controller\_type), video\_memory\_sz(video\_memory\_sz), hard\_driver\_type(hard\_driver\_type),

                num\_hard\_drivers(num\_hard\_drivers), hard\_driver\_memory(hard\_driver\_memory),

                num\_integrated\_controllers(num\_integrated\_controllers), num\_external\_devices(num\_external\_devices),

                operating\_system(operating\_system) {}

void RowTable::print() const{

    std::cout << "| " << std::setw(14) << std::left << owner\_surname << "| " << std::setw(18) << std::left << num\_processors <<

                "| " << std::setw(16) << std::left << processor\_type << "| " << std::setw(12) << std::left << memory\_sz <<

                "| " << std::setw(22) << std::left << video\_controller\_type << "| " << std::setw(13) << std::left << video\_memory\_sz <<

                "| " << std::setw(17) << std::left << hard\_driver\_type << "| " << std::setw(20) << std::left << num\_hard\_drivers <<

                "| " << std::setw(19) << std::left << hard\_driver\_memory << "| " << std::setw(30) << std::left <<

                num\_integrated\_controllers << "| " << std::setw(24) << std::left << num\_external\_devices << "| " <<

                std::setw(17) << std::left << operating\_system << "|\n";

}

**DataBase.hpp**

#ifndef DATABASE\_HPP

#define DATABASE\_HPP

#include "RowTable.hpp"

#include "RowTable.cpp"

#include <iostream>

#include <string>

class DataBase{

private:

    FILE\* file = nullptr;

    std::string\_view file\_path;

public:

    DataBase() = default;

    DataBase(const std::string& file\_name);

    ~DataBase();

    std::string createFile(const std::string& file\_name = "DBFile");

    void deleteFile();

    void addData(const RowTable& computer);

    void print() const;

    void findOwners(const uint64\_t& num\_of\_external\_devices) const;

};

#endif

**DataBase.cpp**

#include "RowTable.hpp"

#include "DataBase.hpp"

#include <direct.h>

#include <io.h>

#include <iostream>

#include <iomanip>

#include <string>

DataBase::DataBase(const std::string& file\_name){

    file\_path = createFile(file\_name);

    file = fopen(file\_path.data(), "wb+");

}

DataBase::~DataBase(){

    if (file != nullptr) fclose(file);

}

std::string make\_dir(const std::string& dir\_name){

    std::string dir\_path = std::string(getenv("USERPROFILE")) + "\\Desktop\\" + dir\_name;

    if (access(dir\_path.c\_str(), 0) != 0){

        mkdir(dir\_path.c\_str());

    }

    return dir\_path;

}

std::string DataBase::createFile(const std::string& file\_name){

    std::string path = make\_dir("DataBase") + "\\" + file\_name + ".bin";

    file = fopen(path.c\_str(), "ab+");

    if (!file){

        throw std::runtime\_error("Failed to create the file");

    }

    std::string header = "My Data Base";

    fwrite(header.c\_str(), header.size(), 1, file);

    fclose(file);

    return path;

}

void DataBase::deleteFile(){

    if (file != nullptr) fclose(file);

    remove(file\_path.data());

}

void DataBase::addData(const RowTable& computer){

    if (file == nullptr){

        throw std::runtime\_error("File doesn't exist");

    }

    fseek(file, 0, SEEK\_END);

    fwrite(&computer, sizeof(computer), 1, file);

}

void DataBase::print() const{

    std::cout << "\n| " << "Owner Surname" << " | " << "Number Processors" << " | " << "Processors Type" << " | " <<

                "Memory Size" << " | " << "Video Controller Type" << " | " << "Video Memory" << " | " <<

                "Hard Driver Type" << " | " << "Number Hard Drivers" << " | " << "Hard Driver Memory" << " | " <<

                "Number Integrated Controllers" << " | " << "Number External Devices" << " | " << "Operating System" << " |\n";

    std::cout << "+---------------+-------------------+-----------------+-------------+-----------------------+" <<

                "--------------+------------------+---------------------+--------------------+-------------------------------+" <<

                "-------------------------+------------------+\n";

    fseek(file, 0, SEEK\_SET);

    RowTable computer;

    while (fread(&computer, sizeof(RowTable), 1, file) == 1){

        computer.print();

        std::cout << "+---------------+-------------------+-----------------+-------------+-----------------------+" <<

                "--------------+------------------+---------------------+--------------------+-------------------------------+" <<

                "-------------------------+------------------+\n";

    }

}

void DataBase::findOwners(const uint64\_t& num\_of\_external\_devices) const{

    std::cout << "Owners dual processor PC, whose has no more than " << num\_of\_external\_devices << " external devices:\n";

    fseek(file, 0, SEEK\_SET);

    RowTable computer;

    size\_t counter = 1;

    while (fread(&computer, sizeof(RowTable), 1, file) == 1){

        if (computer.num\_processors == 2 && computer.num\_external\_devices <= num\_of\_external\_devices){

            std::cout << counter << ". " << computer.owner\_surname << "\n";

            ++counter;

        }

    }

}

**Run.cpp**

#include "RowTable.hpp"

#include "DataBase.hpp"

#include "DataBase.cpp"

#include <iostream>

#include <iomanip>

#include <string>

int main(int argc, char\* argv[]){

    DataBase db("MyBDFile");

    RowTable comp1("Alekseev", 2, "Pentium", 1400, "PCI", 16020, "SCSI", 1, 14751, 5, 4, "Ubuntu");

    RowTable comp2("Garibyan", 1, "AMD", 600, "AGP", 4096, "IDE", 2, 14789, 2, 1, "Windows");

    RowTable comp3("Nefedov", 3, "Baikal", 256, "Embedded", 14855, "ATA", 2, 18662, 3, 3, "Arch");

    RowTable comp4("Smerchinskaya", 4, "Intel", 1024, "PCI", 4665, "SATA", 2, 25641, 1, 6, "Windows");

    RowTable comp5("Bitukov", 1, "AMD", 2048, "PCI", 7888, "IDE", 3, 26465, 6, 7, "MacOS");

    RowTable comp6("Bartakovski", 2, "Intel", 1240, "Embedded", 14096, "SATA", 3, 14751, 3, 5, "Windows");

    RowTable comp7("Martushova", 1, "Intel", 1349, "Embedded", 13096, "IDE", 1, 18462, 5, 3, "Ubuntu");

    RowTable comp8("Pegachkova", 1, "AMD", 1491, "External", 2048, "SCSI", 1, 85616, 2, 9, "MacOS");

    RowTable comp9("Zaitsev", 1, "Intel", 41941, "PCI", 7895, "ATA", 2, 15445, 4, 7, "Ubuntu");

    RowTable comp10("Burdin", 4, "AMD", 14141, "AGP", 4096, "SCSI", 2, 26462, 5, 3, "Arch");

    RowTable comp11("Bobkov", 3, "AMD", 41414, "Embedded", 7889, "SCSI", 2, 15452, 1, 1, "MacOS");

    RowTable comp12("Krylov", 1, "AMD", 1656, "PCI", 1532, "IDE", 1, 78232, 2, 0, "Ubuntu");

    RowTable comp13("Bulakina", 1, "Intel", 784, "External", 1514, "ATA", 2, 7812, 4, 5, "Arch");

    RowTable comp14("Kondaratsev", 2, "Intel", 4151, "AGP", 7851, "SCSI", 1, 13228, 4, 4, "Windows");

    RowTable comp15("Soshnikov", 2, "AMD", 17885, "External", 3214, "SATA", 2, 79512, 2, 6, "Arch");

    db.addData(comp1);

    db.addData(comp2);

    db.addData(comp3);

    db.addData(comp4);

    db.addData(comp5);

    db.addData(comp6);

    db.addData(comp7);

    db.addData(comp8);

    db.addData(comp9);

    db.addData(comp10);

    db.addData(comp11);

    db.addData(comp12);

    db.addData(comp13);

    db.addData(comp14);

    db.addData(comp15);

    int p\_val = 0;

    for (size\_t i = 1; i < argc; ++i){

        if (argv[i] == "-f"){

            db.print();

        } else if (argv[i] == "-p" && (i + 1) < argc){

            p\_val = std::atoi(argv[i + 1]);

        }

    }

    db.findOwners(p\_val);

    return 0;

}

**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](http://sqlfiddle.com/%23!9/f23fe0/3)
4. <https://itvdn.com/ru/blog/article/m-sql#5>
5. <https://sql-language.ru/select-where.html>