**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №1**

**по дисциплине «Программирование»**

Тема: **типы данных, определяемые пользователем. Структуры**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9894 |  | Алексеев В.А. |
| Преподаватель |  | Глущенко А.Г. |

Санкт-Петербург

2020

**Цель работы.**

Цель работы: изучение и организация структур. Получение практических навыков работы со структурами. Определение преимуществ и недостатков использования структур.

**Основные теоретические положения.**

1. Переименование типов (typedef)

Чтобы программы была более ясной, можно задать типу новое имя с помощью ключевого имя typedef: typedef тип имя [размерность]; Квадратные скобки являются элементом синтаксиса. Размерность может отсутствовать. Кроме задания типам с длинными описаниями более коротких псевдонимов, typedef используют для облегчения переносимости программ: если машинно-зависимые типы объявить с помощью оператора typedef, при переносе программы потребуется внести изменения только в эти операторы.

2. Понятие структуры (struct)

Структуры представляют собой группы связанных между собой, как правило, разнотипных переменных, объединенных в единый объект, в отличие от массива, все элементы которого однотипны. В языке C++ структура является видом класса и обладает всеми его свойствами. Чаще всего ограничиваются тем, как структуры представлены в языке С:

struct [имя\_типа]

{

тип\_1 элемент\_1;

тип \_2 элемент\_2;

…

тип\_k элемент\_k;

}

[ список\_описателей ];

Описание структуры начинается ключевым словом struct. Каждая входящая в структуру переменная называется членом (полем, элементом) структуры и описывается типом данных и именем. Поля структуры могут быть любого типа данных. Их количество не лимитировано.

Вся эта конструкция является инструкцией языка программирования, поэтому после нее всегда должен ставиться символ ‘;’.

При описании структуры память для размещения данных не выделяется. Работать с описанной структурой можно только после того, как будет определена переменная (переменные) этого типа данных, только при этом компилятор выделит необходимую память.

Для инициализации структуры значения ее элементов перечисляют в фигурных скобках в порядке их описания:

struct complex

{

float real, im;

} data [2][2] = { {{1,1}, {2,2}}, {{3,3}, {4,4}} };

Все поля структурных переменных располагаются в непрерывной области памяти одно за другим. Общий объем памяти, занимаемый структурой, равен сумме размеров всех полей структуры. Для определения размера структуры следует использовать инструкцию sizeof().

3. Доступ к полям структур

Чтобы записать данные в структурную переменную, необходимо каждому полю структуры присвоить определенное значение. Для этого необходимо использовать оператор “.” (“точка”):

struct Stack

{

// Cтек

float arr[100];

short topIndex;

}; … Stack stack;

// Объявляем переменную типа

Stack Stack.arr[0] = 1;

При доступе к определенному полю его следует рассматривать как обычную переменную, тип данных которой соответствует типу этого поля. Поля структур могут участвовать в качестве операндов любых выражений, допускающих использование операндов соответствующего типа данных.

Копирование данных из одной структурной переменной в другую осуществляется простой операцией присваивания не зависимо от количества полей и размера структуры (это можно делать только в том случае, когда обе переменные одного и того же типа).

В программировании очень часто используются такие конструкции, как массивы структур. Например, сведения о студентах некоторой учебной группы можно хранить в массиве студентов:

t\_Student Gruppa\_N [30];

Был определен 30-ти элементный массив, каждый элемент которого предназначен для хранения данных одного студента. Получение доступа к данным некоторого студента из группы N осуществляется обычной индексацией переменной массива. Поскольку поля структуры могут быть любого типа данных, то они могут в свою очередь быть другой структурой или массивом других структур:

struct Stud

{

char FN[100];

short listNumber;

};

struct Group

{

int groupNumber;

short students;

Stud stud[30];

};

Но в структуре поля нельзя использовать элемент, тип которого совпадает с типом самой структуры, так как рекурсивное использование структур запрещено.

4. Указатели на структуры

Любая структурная переменная занимает в памяти определенное положение, характеризующееся конкретным адресом. Для работы с адресами структурных переменных (как и для простых переменных) можно использовать указатели. Указатели на структурные переменные определяются точно так же, как и для обычных переменных. Разыменование указателя (обращение к данным по адресу, хранящемуся в указателе) осуществляется так же обычным образом.

Через указатели можно работать с отдельными полями структур. Для доступа к полю структуры через указатель используется оператор “→” (“стрелка”), а не “точка”.

7.5. Структурные параметры функции

Структуры можно использовать в качестве параметров функций, как и обычные переменные. Для структур поддерживаются все три механизма передачи данных – по значению, через указатели и по ссылке.

Передачу структур в функции по значению необходимо использовать аккуратно:

void WriteStudent ( t\_Student S )

{

cout << "Фамилия: " << S.Fam << endl;

cout << "Имя: " << S.Name << endl;

cout << "Год рождения: " << S.Year << endl;

if ( S.Sex ) cout << "Пол: " << "М\n";

else

cout << "Пол: " << "Ж\n"; cout << "Средний балл: " << S.Grade << endl;

}

Вызов такой функции сопровождается дополнительным расходом памяти для создания локальной переменной S и дополнительными затратами времени на физическое копирование данных из аргумента в параметр S. Учитывая то, что объем структур может быть очень большим, то эти дополнительные затраты вычислительных ресурсов могут быть чрезмерными.

Предпочтительно использование передачи структуры по указателю или ссылке:

void WriteStudent ( t\_Student \*S )

{

cout << "Фамилия: " << S -> Fam << endl;

cout << "Имя: " << S -> Name << endl;

cout << "Год рождения: " << S -> Year << endl;

if ( S -> Sex ) cout << "Пол: " << "М\n"; else

cout << "Пол: " << "Ж\n"; cout << "Средний балл: " << S -> Grade << endl;

}

Фактической передачи данных в функцию не осуществляется. Дополнительные затраты памяти для создания локальной переменной небольшие – это адрес памяти (4 байта, независимо от размера самой структуры). Вызов такой функции будет осуществляться быстрее, а расход памяти существенно меньше, чем при передачи данных по значению.

Передача по ссылке по своей эффективности эквивалентна передачи данных через указатель. Однако, поскольку при передаче данных по ссылке все адресные преобразования берет на себя компилятор, существенно упрощается программирование действий со структурами. При использовании ссылочных параметров структурных типов доступ к членам структуры осуществляется обычным способом – с помощью оператора “точка”.

Недостатком этих способов является то, что случайные изменения значений полей структуры внутри функции отразятся на значении аргумента после окончания работы функции. Если необходимо предотвратить изменения переданных по адресу аргументов, можно при определении соответствующего параметра объявить его константой (использовать спецификатор const).

**Постановка задачи.**

Необходимо создать массив структур, содержащий информацию о студентах: ФИО, пол, номер группы, номер в списке группы, оценки за прошедшую сессию (всего 3 экзамена и 5 дифф. зачётов), форма обучения, отметка времени о внесении или изменении данных. Если студент имеет оценку 2, то запись о нем нужно удалить. Написать функции, реализующие операции со структурами:

1) Создание новой записи о студенте.

2) Внесение изменений в уже имеющуюся запись.

3) Вывод всех данных о студентах.

4) Вывод информации о всех студентов группы N. N – инициализируется пользователем.

5) Вывод топа самых умных студентов с наивысшим по рейтингу среднем баллом за прошедшую сессию.

6) Вывод количества студентов мужского и женского пола.

7) Определение количества студентов, которые будут получать стипендию (стипендия начисляется если у студента нет 3 и форма обучения - очная);

8) Вывод данных о студентах, которые не получают стипендию; учатся только на «хор»; учатся на «хор» и «отл»; учатся только на «отл».

9) Вывод всех записей, сделанных в день, который введёт пользователь. Вывод всех записей, сделанных после полудня. Вывод всех записей, сделанных до полудня.

10) Вывод данных о студентах, имеющий номер в списке – k.

11) Вывод данных о всех студентах в файл students

**Выполнение работы.**

На изобр. 1 (см. приложение Ж) демонстрируется выполнение первого пункта работы – создание новой записи о студенте.

На изобр.Таблица 2 (см. приложение З) демонстрируется выполнение второго пункта работы – внесение изменений в уже имеющуюся запись.

На изобр.Таблица 3 (см. приложение И) демонстрируется выполнение третьего пункта работы – вывод всех данных о студентах.

На изобр.Таблица 4 (см. приложение К) демонстрируется выполнение четвертого пункта работы – вывод информации о всех студентов группы N.

На изобр.Таблица 5 (см. приложение Л) демонстрируется выполнение пятого пункта работы – вывод топа самых умных студентов с наивысшим по рейтингу среднем баллом за прошедшую сессию.

На изобр.Таблица 6 (см. приложение М) демонстрируется выполнение шестого пункта работы – вывод количества студентов мужского и женского пола

На изобр.Таблица 7 (см. приложение Н) демонстрируется выполнение седьмого пункта работы – определение количества студентов, которые будут получать стипендию.

На изобр.Таблица 8 (см. приложение О) демонстрируется выполнение восьмого пункта работы – вывод данных о студентах, которые не получают стипендию; учатся только на «хор»; учатся на «хор» и «отл»; учатся только на «отл».

На изобр.Таблица 10 (см. приложение П) демонстрируется выполнение десятого пункта работы – вывод данных о студентах, имеющий номер в списке – k.

На изобр.Таблица 11 (см. приложение Р) демонстрируется выполнение одиннадцатого пункта работы – вывод данных о всех студентах в файл students.

**Выводы.**

В результате получена программа, удовлетворяющая требованиям практического задания.

Подобная программа может быть использована для учета обучающихся в образовательном учреждении. Также при определенных изменениях программа может быть использована и в других областях.

Приложение А

Заголовочный Файл

#include "CommandInterpreter.h"

int main() {

// Запуск интерпретатора команд

CommandInterpreter shell;

shell.Start();

return 0;

}

Приложение Б

Интерпретатор команд

#pragma once

#include <iostream>

#include <string>

#include <vector>

#include "Strategies.h"

class CommandInterpreter

{

private:

// Предопределенные значения команд

const char\* \_loadStudent = "load\_student";

const char\* \_addNode = "add\_student";

const char\* \_help = "help";

const char\* \_exit = "exit";

const char\* \_showStudent = "show\_student";

const char\* \_showAllStudents = "show\_all\_students";

const char\* \_showStudentsByGroup = "show\_students\_by\_group";

const char\* \_showSexCount = "show\_sex\_count";

const char\* \_showWithScholarship = "show\_with\_scholarship";

const char\* \_showWOScholarship = "show\_wo\_scholarship";

const char\* \_allowableMarks = "show\_with\_marks";

const char\* \_showWithNumList = "show\_with\_num\_in\_list";

const char\* \_showTop = "show\_top";

const char\* \_loadFromFile = "load\_from\_file";

const char\* \_showChangedToday = "show\_changed\_today";

const char\* \_changeGroupNum = "change\_group\_num";

const char\* \_changeListNum = "change\_list\_num";

const char\* \_changeEduForm = "change\_edu\_form";

// Инициализация объекта для обработки последующих вариаций

Strategies\* \_strategies = new Strategies();

// Инициализация объекта для работы с пользовательским интерфейсом

UIController\* \_ui = new UIController();

// Перечисление возможных команд

static enum Commands

{

LoadStudent,

AddStudent,

ShowStudent,

ShowAllStudents,

ShowStudentsByGroup,

ShowSexCount,

ShowWithScholarship,

ShowWOScholarship,

ShowWithMarks,

ShowWithNumList,

ShowChangedToday,

ShowTop,

ChangeListNum,

ChangeEduForm,

ChangeGroupNum,

LoadFromFile,

Unrecognized,

Help,

Exit

};

// Явное сопоставление команд с их представлением в программе

int InterpreteCommand(std::string command)

{

if (command == \_loadStudent) return LoadStudent;

if (command == \_addNode) return AddStudent;

if (command == \_help) return Help;

if (command == \_exit) return Exit;

if (command == \_showStudent) return ShowStudent;

if (command == \_showAllStudents) return ShowAllStudents;

if (command == \_showStudentsByGroup) return ShowStudentsByGroup;

if (command == \_showSexCount) return ShowSexCount;

if (command == \_showWithScholarship) return ShowWithScholarship;

if (command == \_showWOScholarship) return ShowWOScholarship;

if (command == \_allowableMarks) return ShowWithMarks;

if (command == \_showWithNumList) return ShowWithNumList;

if (command == \_showTop) return ShowTop;

if (command == \_loadFromFile) return LoadFromFile;

if (command == \_changeGroupNum) return ChangeGroupNum;

if (command == \_changeListNum) return ChangeListNum;

if (command == \_changeEduForm) return ChangeEduForm;

if (command == \_showChangedToday) return ShowChangedToday;

return Unrecognized;

}

// Метод для определения поведения программы

// command contains commandName[0] and arguments [1], [2], [3], ...

bool DefineStrategy(std::vector<std::string> command)

{

int commandNum = InterpreteCommand(command[0]);

switch (commandNum)

{

case LoadStudent:

{

if(command.size() == 1)

\_strategies->LoadStudent();

break;

}

case AddStudent:

{

\_strategies->AddStudent();

break;

}

case Help:

{

std::cout << "Im nearby, ask me plz ^^\n";

break;

}

case ShowStudent:

{

if (command.size() != 2)

{

\_ui->ErrorMessage("ID is required");

return false;

}

int id = std::atoi(command[1].c\_str());

\_strategies->ShowStudent(id);

break;

}

case ShowAllStudents:

{

\_strategies->ShowAllStudents();

break;

}

case ShowStudentsByGroup:

{

if (command.size() != 2)

{

\_ui->ErrorMessage("Group num is required");

return false;

}

int id = std::atoi(command[1].c\_str());

\_strategies->ShowStudentsByGroup(id);

break;

}

case ShowSexCount:

{

\_strategies->ShowSexCount();

break;

}

case ShowWithScholarship:

{

\_strategies->ShowWithScholarship();

break;

}

case ShowWOScholarship:

{

\_strategies->ShowWOScholarship();

break;

}

case ShowWithMarks:

{

if (command.size() < 2)

{

\_ui->ErrorMessage("Marks are required");

return false;

}

int\* marks = new int[command.size() - 1];

for (int i = 0; i < command.size() - 1; ++i)

marks[i] = atoi(command[i + 1].c\_str());

\_strategies->ShowWithMarks(marks, command.size() - 1);

break;

}

case ShowWithNumList:

{

if (command.size() != 2)

{

\_ui->ErrorMessage("Num in list is required");

return false;

}

int num = std::atoi(command[1].c\_str());

\_strategies->ShowWithNumList(num);

break;

}

case ShowTop:

{

if (command.size() != 2)

{

\_ui->ErrorMessage("Top size is required");

return false;

}

int size = std::atoi(command[1].c\_str());

\_strategies->ShowTop(size);

break;

}

case LoadFromFile:

{

if (command.size() != 2)

{

\_ui->ErrorMessage("File name is required");

return false;

}

\_strategies->LoadFromFile(command[1]);

break;

}

case ChangeGroupNum:

{

if (command.size() != 2)

{

\_ui->ErrorMessage("Group num is required");

return false;

}

\_strategies->ChangeGroup(std::atoi(command[1].c\_str()));

break;

}

case ChangeListNum:

{

if (command.size() != 2)

{

\_ui->ErrorMessage("List num is required");

return false;

}

\_strategies->ChangeListNum(std::atoi(command[1].c\_str()));

break;

}

case ChangeEduForm:

{

if (command.size() != 2)

{

\_ui->ErrorMessage("Edu from is required");

return false;

}

\_strategies->LoadFromFile(command[1]);

break;

}

case Unrecognized:

{

std::cout << "Command unrecognized. Type \"help\" to see all available commands\n";

break;

}

case ShowChangedToday:

{

\_strategies->ShowChangedToday();

break;

}

case Exit:

{

Dispose();

return true;

}

}

return false;

}

public:

// Запуск интерпретатора

void Start()

{

std::vector<std::string> fullCommand;

while (true)

{

std::cout << ">>> ";

std::string currCommand;

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

if (currCommand.size() == 0) std::getline(std::cin, currCommand);

std::string commandPart;

for (int i = 0; i < currCommand.length(); ++i)

{

if (currCommand[i] == ' ' && !commandPart.empty())

{

fullCommand.push\_back(commandPart);

commandPart.clear();

}

else

{

commandPart.push\_back(currCommand[i]);

}

}

fullCommand.push\_back(commandPart);

if (DefineStrategy(fullCommand)) break;

fullCommand.clear();

}

}

// Освобождение памяти

void Dispose()

{

delete(\_strategies);

delete(\_ui);

}

};

Приложение В

Стратегии

#pragma once

#include "DataBaseController.h"

#include "UIController.h"

#include <ctime>

class Strategies

{

private:

// Указатель на контроллер базы данных

DataBaseController\* \_db;

// Указатель на контроллер пользовательского интерфейса

UIController\* \_ui;

// Указатель на загруженный узел БД

Student\* \_loadedStudent;

void DefineParameter(std::string parameter)

{

}

public:

Strategies()

{

\_ui = new UIController();

\_db = new DataBaseController();

}

void AddStudent()

{

if (\_loadedStudent == nullptr)

{

\_ui->ErrorMessage("Student not loaded");

return;

}

time\_t seconds = time(NULL);

tm timeInfo; char str[20];

if (localtime\_s(&timeInfo, &seconds) == 0) strftime(str, sizeof(str), "%e/%m/%Y", &timeInfo);

\_loadedStudent->LastChangeDate = str;

\_db->AddNode(\_loadedStudent);

}

void LoadStudent()

{

\_loadedStudent = \_ui->InputStudent();

}

void ShowStudent(int id)

{

Student\* student = \_db->GetById(id);

\_loadedStudent = student;

if (student == nullptr)

{

\_ui->ErrorMessage("There is no student with such id");

return;

}

\_ui->PrintStudent(student);

}

void ShowAllStudents()

{

\_db->GetStorage()->PrintAllList();

}

void ShowStudentsByGroup(int group)

{

\_db->GetStorage()->PrintNodesByGroup(group);

}

void ShowSexCount()

{

int\* sexCounted = \_db->GetSexCount();

\_ui->PrintSexCount(sexCounted);

}

void ShowWithScholarship()

{

\_db->GetStorage()->PrintWithScholarship();

}

void ShowWOScholarship()

{

\_db->GetStorage()->PrintWOScholarship();

}

void ChangeStudentInfo(int id, std::string paramenter)

{

Student\* student = \_db->GetById(id);

}

void ShowWithMarks(int\* marks, int size)

{

\_db->GetStorage()->PrintWithMarks(marks, size);

}

void ShowWithNumList(int num)

{

\_db->GetStorage()->PrintWithNumList(num);

}

void ShowTop(int sizeTop)

{

\_db->GetStorage()->PrintTop(sizeTop);

}

void LoadFromFile(std::string fileName)

{

Student\*\* fromFile = \_ui->LoadFile(fileName);

\_db->AddRange(fromFile, \_ui->GetNodesLoaded());

}

void ChangeGroup(int group)

{

time\_t seconds = time(NULL);

tm timeInfo; char str[20];

if (localtime\_s(&timeInfo, &seconds) == 0) strftime(str, sizeof(str), "%e/%m/%Y", &timeInfo);

\_loadedStudent->LastChangeDate = str;

if (\_loadedStudent == nullptr)

{

\_ui->ErrorMessage("Student not loaded");

return;

}

\_loadedStudent->GroupNum = group;

}

void ChangeListNum(int num)

{

time\_t seconds = time(NULL);

tm timeInfo; char str[20];

if (localtime\_s(&timeInfo, &seconds) == 0) strftime(str, sizeof(str), "%e/%m/%Y", &timeInfo);

\_loadedStudent->LastChangeDate = str;

if (\_loadedStudent == nullptr)

{

\_ui->ErrorMessage("Student not loaded");

return;

}

\_loadedStudent->ListNum = num;

}

void ChangeEduForm(std::string form)

{

time\_t seconds = time(NULL);

tm timeInfo; char str[20];

if (localtime\_s(&timeInfo, &seconds) == 0) strftime(str, sizeof(str), "%e/%m/%Y", &timeInfo);

\_loadedStudent->LastChangeDate = str;

if (\_loadedStudent == nullptr)

{

\_ui->ErrorMessage("Student not loaded");

return;

}

\_loadedStudent->EduFormat = \_ui->DefineEduForm(form);

}

void ShowChangedToday()

{

time\_t seconds = time(NULL);

tm timeInfo; char str[20];

if (localtime\_s(&timeInfo, &seconds) == 0) strftime(str, sizeof(str), "%e/%m/%Y", &timeInfo);

\_db->GetStorage()->PrintChangedToday(str);

}

~Strategies()

{

delete(\_ui);

delete(\_db);

}

};

Приложение г

Контроллер пользовательского интерфейса

#pragma once

#include "DataBaseController.h"

#include <iostream>

#include "List.h"

#include <fstream>

#include <iomanip>

class UIController

{

private:

const char\* \_consoleChars = ">>> ";

int \_gotNodes;

public:

const char\* DefineSex(Sex sex)

{

switch (sex)

{

case Male: return "Male";

case Female: return "Female";

}

}

const char\* DefineEduForm(EduFormat form)

{

switch (form)

{

case Fulltime: return "Fulltime";

case Evening: return "Evening";

case Distance: return "Distance";

}

}

Sex DefineSex(std::string sex)

{

if (sex == "male" || sex == "Male") return Male;

return Female;

}

EduFormat DefineEduForm(std::string form)

{

if (form == "Fulltime") return Fulltime;

else if (form == "Evening") return Evening;

return Distance;

}

int GetNodesLoaded()

{

return \_gotNodes;

}

void ErrorMessage(const char\* message)

{

std::cout << \_consoleChars << "Error: " << message << std::endl;

}

void InfoMessage(const char\* message1, const char\* message2)

{

std::cout << \_consoleChars << message1 << "\t" << message2 << std::endl;

}

void InfoMessage(const char\* message1, int message2)

{

std::cout << \_consoleChars << message1 << "\t" << message2 << std::endl;

}

int InputStudentId()

{

int id;

std::cout << \_consoleChars << "Enter student id..." << std::endl;

std::cin >> id;

return id;

}

void PrintSexCount(int\* sexCount)

{

InfoMessage("Male count ", sexCount[0]);

InfoMessage("Female count", sexCount[1]);

}

void PrintStudent(Student\* student)

{

std::string sex = DefineSex(student->Sex);

std::string eduForm = DefineEduForm(student->EduFormat);

std::cout << "==================================================================" << std::endl;

std::cout << "|| First name: " << student->FirstName << std::endl;

std::cout << "|| Last name: " << student->LastName << std::endl;

std::cout << "|| Ext name: " << student->ExtName << std::endl;

std::cout << "|| Sex: " << sex << std::endl;

std::cout << "|| Group #: " << student->GroupNum << std::endl;

std::cout << "|| List #: " << student->ListNum << std::endl;

std::cout << "|| Education format: " << eduForm << std::endl;

std::cout << "||" << std::endl;

for (int i = 0; i < 3; ++i)

{

std::cout << "|| "<< i + 1 << " exam: " << student->SessionResults[i] << std::endl;

}

for (int i = 3; i < 8; ++i)

{

std::cout << "|| " << i - 2 << " diff: " << student->SessionResults[i] << std::endl;

}

std::cout << "|| Last modify: " << student->LastChangeDate << std::endl;

std::cout << "==================================================================" << std::endl;

}

Student\* InputStudent()

{

Student\* toSend = new Student();

std::cout << "Enter first name:\t";

std::cin >> toSend->FirstName;

std::cout << "Enter last name:\t";

std::cin >> toSend->LastName;

std::cout << "Enter ext name:\t\t";

std::cin >> toSend->ExtName;

std::cout << "Enter sex:\t";

std::string sex;

std::cin >> sex;

toSend->Sex = DefineSex(sex);

std::cout << "Enter group num:\t";

std::cin >> toSend->GroupNum;

std::cout << "Enter list num:\t";

std::cin >> toSend->ListNum;

std::cout << "Enter education format:\t";

std::string eduFormat;

std::cin >> eduFormat;

toSend->EduFormat = DefineEduForm(eduFormat);

int\* sessionResults = new int[8];

for (int i = 0; i < 3; ++i)

{

std::cout << "Enter the mark for " << i + 1 << " exam:\t";

int mark;

std::cin >> mark;

sessionResults[i] = mark;

}

for (int i = 3; i < 8; ++i)

{

std::cout << "Enter the mark for " << i - 2 << " diff:\t";

int mark;

std::cin >> mark;

sessionResults[i] = mark;

}

toSend->SessionResults = sessionResults;

return toSend;

}

// Метод для загрузки узлов из файла

Student\*\* LoadFile(std::string file)

{

Student\*\* result = new Student\*[100];

int counter = 0;

std::ifstream fileStream;

fileStream.open(file);

std::string buff;

while (!fileStream.eof())

{

Student\* node = new Student();

std::string buff;

std::getline(fileStream, buff);

node->LastName = buff; buff.clear(); std::getline(fileStream, buff);

node->FirstName = buff; buff.clear(); std::getline(fileStream, buff);

node->ExtName = buff; buff.clear(); std::getline(fileStream, buff);

node->Sex = DefineSex(buff); buff.clear(); std::getline(fileStream, buff);

node->GroupNum = std::atoi(buff.c\_str()); buff.clear(); std::getline(fileStream, buff);

node->ListNum = std::atoi(buff.c\_str()); buff.clear();

int\* sessionMarks = new int[8];

for (int i = 0; i < 8; ++i)

{

std::getline(fileStream, buff);

sessionMarks[i] = std::atoi(buff.c\_str());

buff.clear();

} std::getline(fileStream, buff);

node->SessionResults = sessionMarks;

node->EduFormat = DefineEduForm(buff); buff.clear(); std::getline(fileStream, buff);

node->LastChangeDate = buff;

result[counter] = node;

counter++;

}

\_gotNodes = counter;

return result;

}

};

Приложение г

Хранилище (Двусвязный список)

#pragma once

#include "StudentModel.h"

#include "UIController.h"

struct Node

{

Student\* Value = nullptr;

Node\* Prev = nullptr;

Node\* Next = nullptr;

};

// Описание двусвязного списка

class List

{

private:

Node\* \_begin = nullptr;

Node\* \_end = nullptr;

UIController\* \_ui = new UIController();

int \_size;

void Swap(Student\* a, Student\* b)

{

Student\* buff = a;

a = b;

b = buff;

}

int Partition(Student\*\* arr, int low, int high)

{

Student\* pivot = arr[high];

int i = -1;

for (int j = 0; j <= high - 1; j++)

{

if (arr[j]->Smart <= pivot->Smart)

{

i++;

Swap(arr[i], arr[j]);

}

}

Swap(arr[i + 1], arr[high]);

return (i + 1);

}

void QuickSort(Student\*\* arr, int low, int high)

{

if (low < high)

{

int pi = Partition(arr, low, high);

QuickSort(arr, low, pi - 1);

QuickSort(arr, pi + 1, high);

}

}

public:

void AddRange(Student\*\* nodes, int size)

{

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

AddBack(nodes[i]);

delete(nodes);

}

void AddBack(Student\* value)

{

Node\* node = new Node();

node->Value = value;

if (\_begin == nullptr && \_end == nullptr)

{

\_end = \_begin = node;

\_size = 1;

}

else

{

\_end->Next = node;

node->Prev = \_end;

\_end = node;

\_size++;

}

}

void RemoveBack()

{

if (\_begin == nullptr && \_end == nullptr)

return;

if (\_begin == \_end && \_begin != nullptr)

{

delete(\_begin);

\_begin = \_end = nullptr;

}

else

{

\_end->Prev->Next = nullptr;

Node\* toDelete = \_end;

\_end = \_end->Prev;

delete(toDelete);

}

\_size--;

}

Student\* FindById(int id)

{

Node\* curr = \_begin;

while (curr != nullptr)

{

if (curr->Value->id == id)

return curr->Value;

curr = curr->Next;

}

return nullptr;

}

int\* CountSex()

{

int\* sexCount = new int[2];

sexCount[0] = sexCount[1] = 0;

Node\* curr = \_begin;

while (curr != nullptr)

{

if (curr->Value->Sex == Male)

sexCount[0]++;

else

sexCount[1]++;

curr = curr->Next;

}

return sexCount;

}

void PrintNodesByGroup(int group)

{

Node\* curr = \_begin;

while (curr != nullptr)

{

if (curr->Value->GroupNum == group)

\_ui->PrintStudent(curr->Value);

curr = curr->Next;

}

}

void PrintAllList()

{

Node\* curr = \_begin;

while (curr != nullptr)

{

\_ui->PrintStudent(curr->Value);

curr = curr->Next;

}

}

void PrintWithScholarship()

{

Node\* curr = \_begin;

bool scholarshop = true;

while (curr != nullptr)

{

if (curr->Value->EduFormat != Fulltime)

{

curr = curr->Next;

continue;

}

for (int i = 0; i < 8; ++i)

{

if (curr->Value->SessionResults[i] < 4)

scholarshop = false;

}

if(scholarshop) \_ui->PrintStudent(curr->Value);

curr = curr->Next;

scholarshop = true;

}

}

void PrintWOScholarship()

{

Node\* curr = \_begin;

bool scholarshop = true;

while (curr != nullptr)

{

if (curr->Value->EduFormat != Fulltime)

{

\_ui->PrintStudent(curr->Value);

curr = curr->Next;

continue;

}

for (int i = 0; i < 8; ++i)

{

if (curr->Value->SessionResults[i] < 4)

scholarshop = false;

}

if (!scholarshop) \_ui->PrintStudent(curr->Value);

curr = curr->Next;

scholarshop = true;

}

}

void PrintWithMarks(int\* marks, int size)

{

Node\* curr = \_begin;

while (curr != nullptr)

{

bool falseMark = false;

for (int i = 0; i < 8; ++i)

{

bool hitMark = false;

for (int j = 0; j < size; ++j)

{

if (curr->Value->SessionResults[i] == marks[j]) hitMark = true;

}

if (!hitMark) falseMark = true;

}

if(!falseMark) \_ui->PrintStudent(curr->Value);

curr = curr->Next;

}

}

void PrintWithNumList(int num)

{

Node\* curr = \_begin;

while (curr != nullptr)

{

if (curr->Value->ListNum == num)

\_ui->PrintStudent(curr->Value);

curr = curr->Next;

}

}

void PrintChangedToday(char\* date)

{

Node\* curr = \_begin;

while (curr != nullptr)

{

if (curr->Value->LastChangeDate == date)

\_ui->PrintStudent(curr->Value);

curr = curr->Next;

}

}

void PrintTop(int topSize)

{

Student\*\* arr = ToArray();

Student\* point = arr[\_size - 1];

Student\* point1 = arr[\_size - 2];

Student\* point2 = arr[\_size - 3];

QuickSort(arr, 0, \_size - 1);

if (\_size <= topSize) PrintAllList();

else for (int i = 0; i < topSize; ++i)

\_ui->PrintStudent(arr[i]);

delete(arr);

}

Student\*\* ToArray()

{

Student\*\* arr = new Student\*[\_size];

int counter = 0;

Node\* curr = \_begin;

while (curr != nullptr)

{

arr[counter] = curr->Value;

curr = curr->Next;

counter++;

}

return arr;

}

~List()

{

Node\* curr = \_begin;

while (curr != nullptr)

{

Node\* tmp = curr->Next;

delete(curr);

curr = tmp;

}

delete(\_ui);

}

};

Приложение Д

Контроллер базы данных

#pragma once

#include <string>

#include "List.h"

#include "StudentModel.h"

#include <ctime>

class DataBaseController

{

private:

List\* \_storage = new List();

int \_currId;

public:

DataBaseController()

{

\_currId = 0;

}

~DataBaseController()

{

delete(\_storage);

}

void AddNode(Student\* student)

{

\_currId++;

student->id = \_currId;

student->Smart = 0;

for (int i = 0; i < 8; ++i)

student->Smart += student->SessionResults[i];

\_storage->AddBack(student);

}

void AddRange(Student\*\* students, int size)

{

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

{

\_currId++;

students[i]->id = \_currId;

}

\_storage->AddRange(students, size);

}

Student\* GetById(int id)

{

return \_storage->FindById(id);

}

int\* GetSexCount()

{

return \_storage->CountSex();

}

List\* GetStorage()

{

return \_storage;

}

};

Приложение Е

представление студента как структуры

#pragma once

#include <string>

static enum Sex

{

Male,

Female

};

static enum EduFormat

{

Fulltime,

Evening,

Distance

};

class Student

{

public:

int id;

std::string LastName;

std::string FirstName;

std::string ExtName;

Sex Sex;

int GroupNum;

int ListNum;

int\* SessionResults;

EduFormat EduFormat;

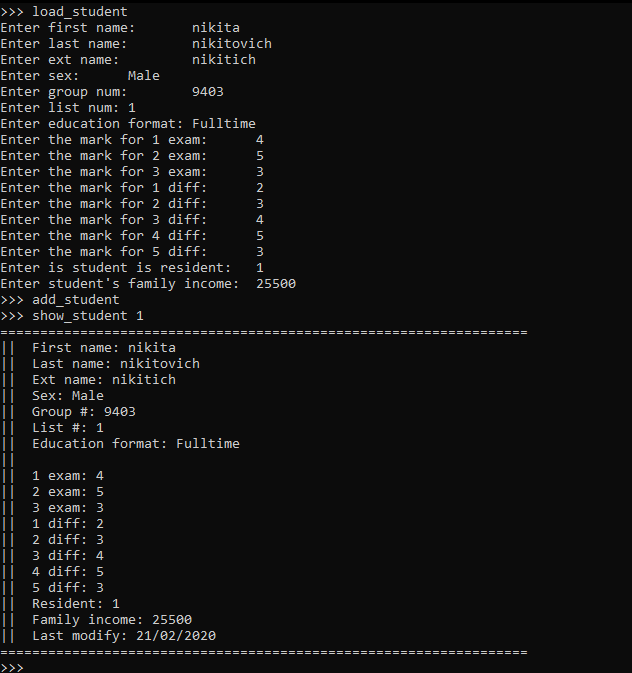
std::string LastChangeDate;

int Smart;

};

Приложение ж

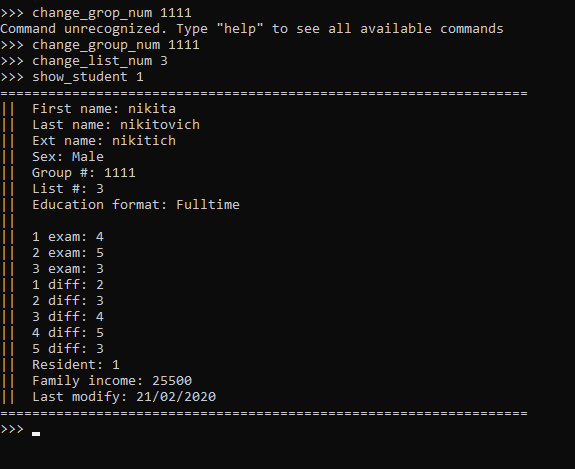
изображение 1



Изображение 1 – демонстрация работы первого пункта задания

Приложение з

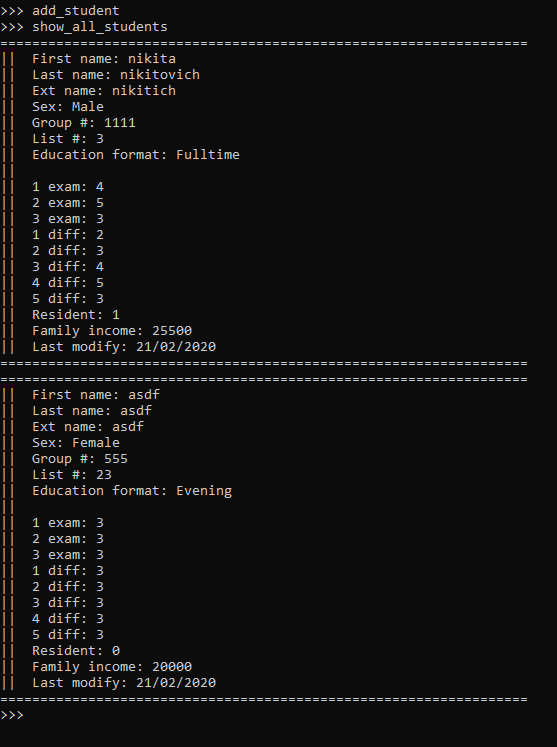
изображение 2



Изображение 2 – демонстрация работы второго пункта задания

Приложение И

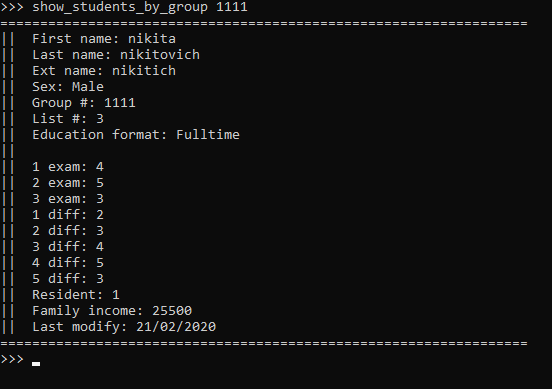
изображение 3



Изображение 3 – демонстрация работы третьего пункта задания

Приложение К

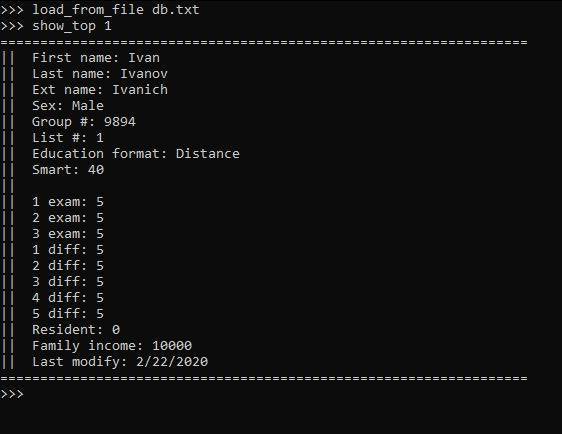
изображение 4



Изображение 4 – демонстрация работы четвертого пункта задания

Приложение Л

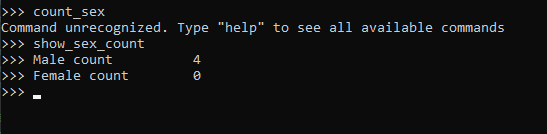
изображение 5



Изображение 5 – демонстрация работы пятого пункта задания

Приложение м

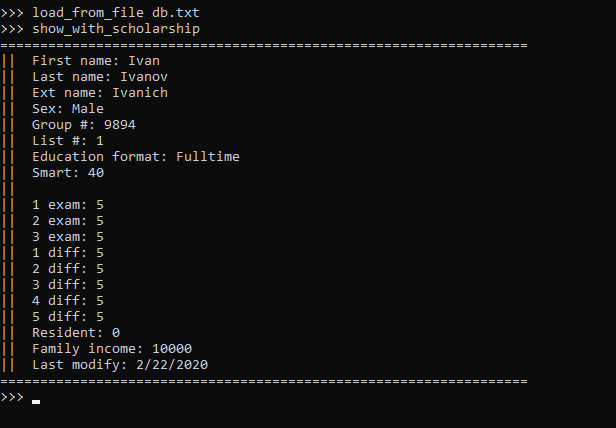
изображение 6



Изображение 6 – демонстрация работы шестого пункта задания

Приложение н

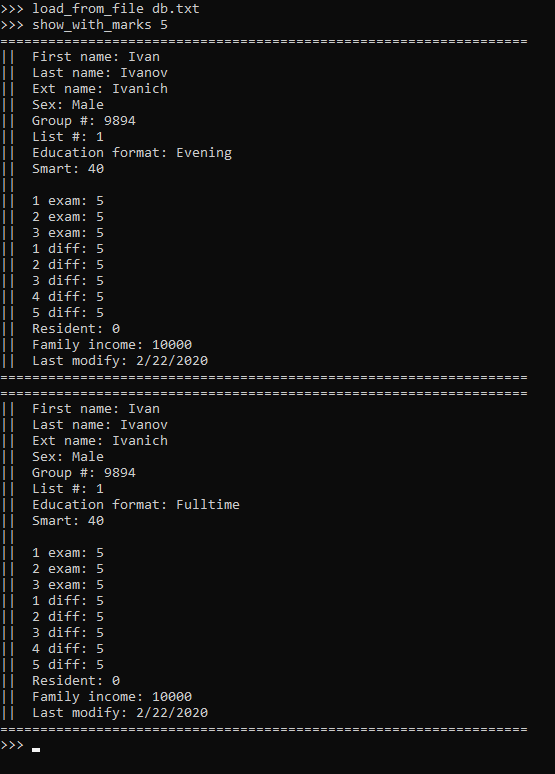
изображение 7



Изображение 7 – демонстрация работы седьмого пункта задания

Приложение о

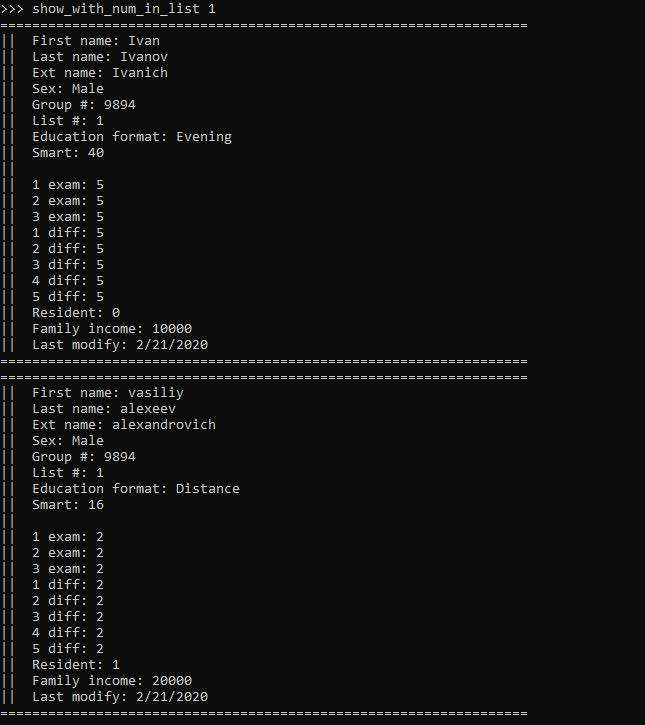
изображение 8



Изображение 8 – демонстрация работы восьмого пункта задания

Приложение п

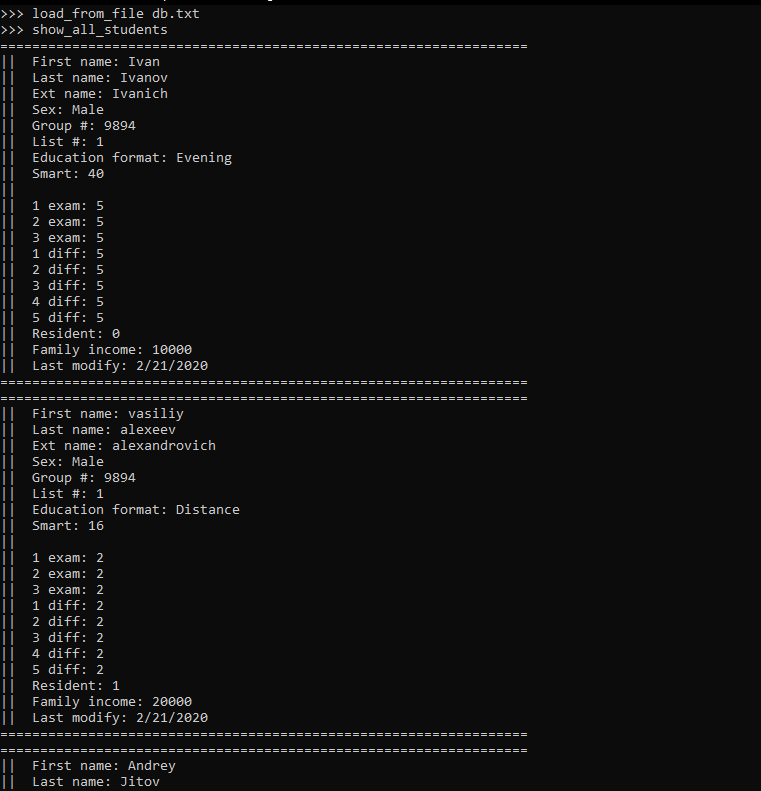
изображение 9



Изображение 9 – демонстрация девятого пункта задания

Приложение р

изображение 10



Изображение 10 – демонстрация работы десятого пункта задания