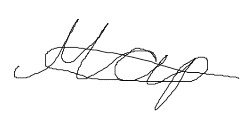
|  |  |
| --- | --- |
|  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |
| Институт комплексной безопасности и специального приборостроения | | |
| Кафедра КБ-3 «Управление и моделирование систем» | | |

**КУРСОВОЙ ПРОЕКТ (РАБОТА)**

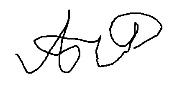
по дисциплине: **«Технологии и методы программирования»**

(наименование дисциплины)

**Тема курсового проекта (работы)**: Разработка программного средства работы с файловой базой данных студентов

**Студент группы** БИСО-05-18 Маркеев Арсений Сергеевич

(учебная группа, фамилия, имя, отчество студента) (подпись студента)



**Руководитель курсового проекта (работы)** старший преподаватель Русаков А.М.

должность, звание, ученая степень, (подпись руководителя)

**Рецензент** (при наличии)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

должность, звание, ученая степень подпись рецензента

Работа представлена к защите «1» июня 2020г.

Допущен к защите «1» июня 2020г.

|  |  |
| --- | --- |
|  | **Утверждаю** |
|  | и.о. Заведующий кафедрой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Пушкин П.Ю.  *Подпись ФИО* |
|  | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_2020 г. |
| **ЗАДАНИЕ** | |
| **на выполнение курсового проекта (работы)** по дисциплине | |
| «**Технологии и методы программирования**» | |

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |
| Институт комплексной безопасности и специального приборостроения |
| Кафедра КБ-3 «Управление и моделирование систем» |

Студент: Маркеев А.С. Группа: БИСО-05-18

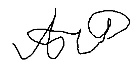
**Тема:** Разработка программного средства работы   
с файловой базой данных студентов

**Исходные данные:** Описание файловой базы данных студентов

**Перечень вопросов, подлежащих разработке, и обязательного графического материала:**

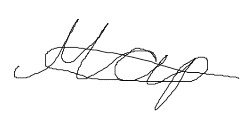
**Разработать** **программное средство работы с файловой базой данных студентов согласно персональному заданию по варианту 57**

**Срок представления к защите курсового проекта (работы)** **до** «1» июня 2020 г.



**Задание на выполнение курсовой проект (работу) выдал \_\_\_\_\_\_\_\_** (Русаков А.М.)

*Подпись руководителя Ф.И.О. руководителя*

** «20» апреля 2020 г. Задание на курсовой проект (работу) получил**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (**Маркеев А.С**)**

*Подпись обучающегося Ф.И.О. исполнителя*

**Оглавление**

[Задание 4](#_Toc41249107)

[Описание используемых методов, свойств, диалоговых окон, элементов управления 6](#_Toc41249108)

[Блок-схема 9](#_Toc41249109)

[Руководство-пользователя 10](#_Toc41249110)

[Листинг 12](#_Toc41249111)

[Распечатка данных 23](#_Toc41249112)

[Список литературы 26](#_Toc41249113)

# **Задание**

Требования к программе:

1) Организовать корректировку всех данных, т.е. иметь возможность изменять данные внутри без изменения остальной информации.

2) Программа должна быть разработана на основе методов объектно- ориентированного программирования с использованием визуальной среды Visual C++ и библиотеки классов MFC(диалоговые окна, стандартные элементы управления, архитектура документ-вид), а также иметь файловую структуру.

Исходными данными для проектирования является:

1. информация о группе студентов из N человек, где запись о студенте содержит следующие данные:

1) Ф.И.О. студента.

2) Число, месяц, год рождения.

3) Год поступления в институт.

4) Факультет, кафедра.

5) Группа.

6) Номер зачетной книжки.

7) Пол.

8) Названия предметов и оценки по каждому предмету в каждой сессии. (максимально 9 сессий и 10 предметов в каждом семестре, которые м.б. разные).

2. конкретное задание, вариант которого определяется 2-мя последними цифрами зачетной книжки.

Выходными данными проекта являются:

1. данные о студентах по всей группе в порядке ввода информация.

2. При выполнении конкретного задания необходимо выводить все данные по каждому студенту начиная от “Ф.И.О.” и заканчивая “Текущим семестром”

**Вариант 57.**

Распечатать всех студентов, у которых за все время обучения не более 25% оценок:

а) 3

б) 3 и 4

в) 5

г) 3 и 5

д) 4 и 5.

Варианты ад выбираются по желанию пользователя. Их можно выбрать как 1, так и несколько или все варианты, с указанием интервала года рождения.

# **Описание используемых методов, свойств, диалоговых окон, элементов управления**

**Класс первого окна(CKyrsov2020Dlg)**

Структура Session: содержит название десяти предметов и оценок к ним.

Структура Student: содержит Ф.И.О., Дату рождения, Пол, Год поступления, Факультет, Кафедру, Номер зачетной книжки и 9 структур типа Session.

Метод fillSession: Заполняет имя предметов во все сессиях элементом

“# “.

**Класс второго окна(CDlg1)**

Структура Session: содержит название десяти предметов и оценок к ним.

Структура Student: содержит Ф.И.О., Дату рождения, Пол, Год поступления, Факультет, Кафедру, Номер зачетной книжки и 9 структур типа Session.

Функция createTable(CListCtrl &ctrl): Задает столбцы в принятом ListCtrl.

**Класс Сryptographer**

Функция Crypting (CStringA str): Принимает строку, шифрует ее, записывает ключ шифрования в файл Key.txt и выводит зашифровыванию строку.

**Класс clStudent**

Потомок класса Сryptographer.

Структура Session: содержит название десяти предметов и оценок к ним.

Функция getElement(CString str): Принимает строку и возвращает кусок строки от последнего просмотренного элемента до первого пробела.

Функция AddStudent(CString str): Принимает на вход стоку и заполняет переменные класса данными из этой строки.

Функция AddData(CListCtrl &ctrl, int row, int col, CString str): Принимает на вход указатель на ListCtrl, номер строки, номер столбца, строку. Записывает данные в таблицу в заданную строчку и столбец.

Функция addToTable(CListCtrl &ctrl, int totalCount, int numSession): Принимает на вход указатель на ListCtrl, номер студента, номер сессии. Заполняет строчку таблицы данными студента.

Функция StudentToStr(bool DataView):

Принимает на вход bool значение. Формирует строку из данных студента. Если bool значение равно true, то строка формируется с одной заданной сессией и датой рождения видом дд.мм.гггг. Если bool значение равно false, то строка формируется со всеми сессиями.

**Исходный файл Kyrsov2020Dlg.cpp**

Функция valid(CString str): Принимает на вход строку и возвращает true, если в этой строке содержатся цифры или пробел.

Функция OnBnClickedButton2: Заполняет структуру Student данными из окна, открывает файл «bd.txt» и записывает структуру в файл.

Функция OnBnClickedButton3: Заполняет выбранную сессию данными из окна.

Функция OnBnClickedButton1: Открывает окно с таблицей студентов

Функция OnBnClickedButton5: Перезаписывает данные студента в файле.

**Исходный файл CDLG1.cpp**

Функция ReadFile(): Записывает данные из файла в лист объектов типа класса сlStudent.

Функция OnBnClickedButton1: Заполняет таблицу данными студентов.

Функция OnBnClickedButton2: Передает данные выбранного студента в первое окно.

Функция RewritingStudent(int curSession): Перезаписывает данные студента в листе студентов.

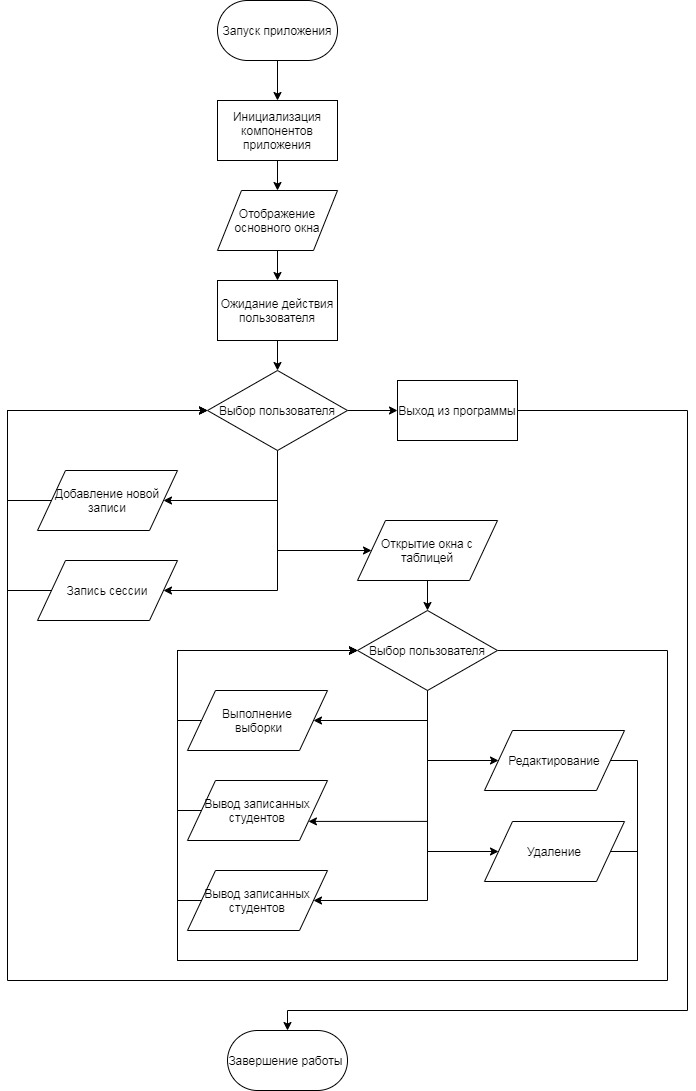
Функция Rewriting(): Перезаписывает файл «bd.txt» данными из листа студентов.

Функция OnBnClickedButton3(): Выводит студентов в таблицу согласно варианту задания.

Функция OnBnClickedButton4(): Перезаписывает файл «bd.txt» без выбранного студента.

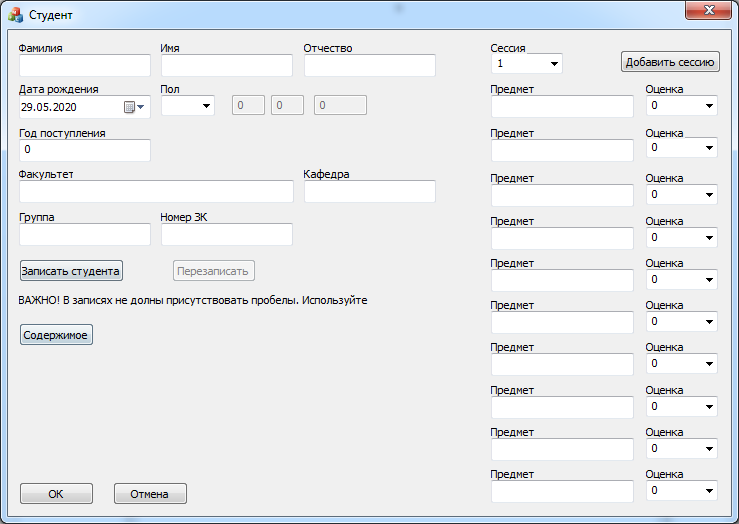
Функция OnBnClickedButton5(): записывает зашифровываный лист студентов в файл «crypt\_bd.txt».

# **Блок-схема**



# **Руководство-пользователя**

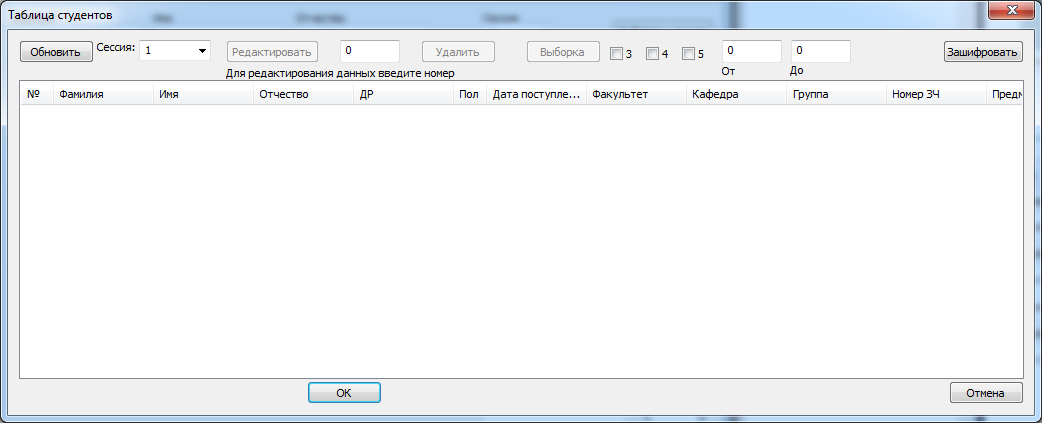
1. После запуска программы открывается главное окно программы.



1. Для добавления записи нужно заполнить поля Ф.И.О., Дату рождения, Год поступления, Факультет, Кафедру, Номер зачетной книжки (если пол не указать в записи будет Н.у.), если нужно добавить сессию, то выбираем номер сессии заполняем ее и нажимаем кнопку “Добавить сессию”. После нажимает кнопку “Записать”.

Важно! При заполнении полей не использовать пробел. (Можно заменить “ - ”)

1. Чтобы вывести таблицу записанных студентов нужно нажать кнопку “Содержимое”.



1. Кнопка «Обновить» очищает таблицу и записывает студентов в нее с данными текущей сессии .
2. Выпадающий список задает текущую сессию.
3. Кнопка «Редактировать» Заполняет поля основного поля данными выбранного студента для их редактирования.
4. Кнопка «Удалить» Удаляет выбранного студента.
5. Кнопка «Выборка» Заполняет таблицу студентами у которых меньше 25% выбранной оценки или оценок.
6. Кнопка «Зашифровать» Создает файл с зашифрованными данными студентов.

**Листинг**

**Класс первого окна(CKyrsov2020Dlg)**

struct Session

{

CStringA name\_subject[10];

int value[10];

};

public:

struct Student

{

CStringA name, surname, otc;

int birthday[3];

int year\_admission;

CStringA faculty, department, group, number\_book, gender;

Session progress[9];

};

void fillSession() {

for (int s = 0; s < 9; s++) {

for (int i = 0; i < 10; i++) {

ec\_student.progress[s].name\_subject[i] = "#";

ec\_student.progress[s].value[i] = 0;

}

}

}

**Класс второго окна(CDlg1)**

struct Session

{

CStringA name\_subject[10];

int value[10];

};

struct Student

{

CStringA name, surname, otc;

int birthday[3];

int year\_admission;

CStringA faculty, department, group, number\_book, gender;

Session progress[9];

};

public:

void createTable(CListCtrl &ctrl) {

CRect rect;

ctrl.GetClientRect(&rect);

int count;

count = rect.Width() / 10.;

ctrl.InsertColumn(0, \_T("№"), LVCFMT\_LEFT, count / 3);

ctrl.InsertColumn(1, \_T("Фамилия"), LVCFMT\_LEFT, count);

ctrl.InsertColumn(2, \_T("Имя"), LVCFMT\_LEFT, count);

ctrl.InsertColumn(3, \_T("Отчество"), LVCFMT\_LEFT, count);

ctrl.InsertColumn(4, \_T("ДР"), LVCFMT\_LEFT, count);

ctrl.InsertColumn(5, \_T("Пол"), LVCFMT\_LEFT, count / 3);

ctrl.InsertColumn(6, \_T("Дата поступления"), LVCFMT\_LEFT, count);

ctrl.InsertColumn(7, \_T("Факультет"), LVCFMT\_LEFT, count);

ctrl.InsertColumn(8, \_T("Кафедра"), LVCFMT\_LEFT, count);

ctrl.InsertColumn(9, \_T("Группа"), LVCFMT\_LEFT, count);

ctrl.InsertColumn(10, \_T("Номер ЗЧ"), LVCFMT\_LEFT, count);

ctrl.InsertColumn(11, \_T("Предметы"), LVCFMT\_LEFT, count);

ctrl.InsertColumn(12, \_T("Оценки"), LVCFMT\_LEFT, count / 3);

}

**Класс Сryptographer**

class Сryptographer {

public:

CStringA Crypting(CStringA str) {

HCRYPTPROV hProv;

HCRYPTKEY hSessionKey;

const char \*chrstr = str;

DWORD count = strlen(chrstr);

if (!CryptAcquireContext(&hProv, NULL, NULL, PROV\_RSA\_FULL, CRYPT\_VERIFYCONTEXT)) { MessageBoxA(NULL, "CryptAcquireContext", "Ошибка шифровщика", MB\_ICONERROR | MB\_OK); return NULL; }

if (!CryptGenKey(hProv, CALG\_RC4, CRYPT\_EXPORTABLE, &hSessionKey)) { MessageBoxA(NULL, "CryptGenKey", "Ошибка шифровщика", MB\_ICONERROR | MB\_OK); return NULL; }

if (!CryptEncrypt(hSessionKey, 0, true, 0, (BYTE\*)chrstr, &count, strlen(chrstr))) { MessageBoxA(NULL, "CryptEncrypt", "Ошибка шифровщика", MB\_ICONERROR | MB\_OK); return NULL; }

ofstream out("Key.txt");

if (!out.is\_open()) {

MessageBoxA(NULL, "Не удалось открыть файл: Key", "Fille not open", MB\_ICONERROR | MB\_OK); return NULL;

}

out << hSessionKey;

out.close();

str = chrstr;

return str;

}

};

**Класс clStudent**

class clStudent: public Сryptographer {

struct Session

{

CString name\_subject[10];

int value[10];

};

int pos = 0;

CString getElement(CString str) {

CString element = \_T("");

for (pos; str[pos] != \_T(' ') && pos != str.GetLength(); pos++) { //Проверка на конец строки

element += str[pos];

}

if (pos != str.GetLength()) { pos++; }

return element;

}

void AddStudent(CString str) {

surname = (LPCTSTR) getElement(str);

name = getElement(str);

otc = getElement(str);

CString buf;

buf = getElement(str);

CSTRbirthday = buf; birthday[0] = \_wtoi(buf.GetString());

buf = getElement(str);

CSTRbirthday += '.' + buf; birthday[1] = \_wtoi(buf.GetString());

buf = getElement(str);

CSTRbirthday += '.' + buf; birthday[2] = \_wtoi(buf.GetString());

gender = getElement(str);

CSTRyear\_admission = getElement(str); year\_admission = \_wtoi(CSTRyear\_admission.GetString());

faculty = getElement(str);

department = getElement(str);

group = getElement(str);

number\_book = getElement(str);

for (int i = 0; i < 9; i++) {

getElement(str);

for (int j = 0; j < 10; j++) {

progress[i].name\_subject[j] = getElement(str);

progress[i].value[j] = \_wtoi(getElement(str));

}

}

}

void AddData(CListCtrl &ctrl, int row, int col, CString str)

{

LVITEM lv;

lv.iItem = row;

lv.iSubItem = col;

lv.pszText = (LPTSTR)(LPCTSTR)str;

lv.mask = LVIF\_TEXT;

if (col == 0)

ctrl.InsertItem(&lv);

else

ctrl.SetItem(&lv);

}

public:

clStudent(CString str) { AddStudent(str); }

CString name, surname, otc;

CString CSTRbirthday;

int birthday[3];

CString CSTRyear\_admission;

int year\_admission;

CString faculty, department, group, number\_book, gender;

Session progress[9];

void addToTable(CListCtrl &ctrl, int totalCount, int numSession) {

CString number; number.Format(\_T("%d"), totalCount + 1);

AddData(ctrl, totalCount, 0, number);

ctrl.SetItemText(totalCount, 1, surname);

ctrl.SetItemText(totalCount, 2, name);

ctrl.SetItemText(totalCount, 3, otc);

ctrl.SetItemText(totalCount, 4, CSTRbirthday);

ctrl.SetItemText(totalCount, 5, gender);

ctrl.SetItemText(totalCount, 6, CSTRyear\_admission);

ctrl.SetItemText(totalCount, 7, faculty);

ctrl.SetItemText(totalCount, 8, department);

ctrl.SetItemText(totalCount, 9, group);

ctrl.SetItemText(totalCount, 10, number\_book);

CString sub = \_T(""); CString val = \_T("");

for (int i = 0; i < 10; i++)

{

if (progress[numSession].name\_subject[i] != \_T("#")) {

sub += progress[numSession].name\_subject[i] + \_T(";");

CString strVal; strVal.Format(\_T("%d"), progress[numSession].value[i]);

val += strVal + \_T(";");

}

}

if (sub != \_T("")) {

ctrl.SetItemText(totalCount, 11, sub);

ctrl.SetItemText(totalCount, 12, val);

}

else {

ctrl.SetItemText(totalCount, 11, \_T("Нет"));

ctrl.SetItemText(totalCount, 12, \_T("Нет"));

}

}

CStringA StudentToStr(bool DataView) {// DataView - вид даты ДР в строке: 0 - ДД.ММ.ГГГГ 1 - ДД ММ ГГГГ

CStringA str = "";

str += surname += " "; str += name += " "; str += otc += " ";

CStringA strDay, strMonth, strYear; strDay.Format("%d", birthday[0]); strMonth.Format("%d", birthday[1]); strYear.Format("%d", birthday[2]);

if (!DataView) {

str += strDay += " "; str += strMonth += " "; str += strYear += " ";

}

else { str += strDay += "."; str += strMonth += "."; str += strYear += " "; }

str += gender += " "; CStringA strYearAdm; strYearAdm.Format("%d", year\_admission); str += strYearAdm += " ";

str += faculty += " "; str += department += " "; str += group += " "; str += number\_book += " ";

for (int i = 0; i < 9; i++)

{

if (DataView) { str += "|"; }

for (int j = 0; j < 10; j++)

{

if (DataView) {

if (progress[i].name\_subject[j] != "-") {

str += progress[i].name\_subject[j] += " ";

CStringA strVal; strVal.Format("%d", progress[i].value[j]);

str += strVal += ";";

}

}

else

{

str += progress[i].name\_subject[j] += " ";

CStringA strVal; strVal.Format("%d", progress[i].value[j]);

str += strVal += " ";

}

}

}

return str;

}

};

**Исходный файл Kyrsov2020Dlg.cpp**

bool valid(CString str) {

string s = (CT2CA)str.GetString();

for (int i = 0; i < strlen(s.c\_str()); i++)

{

if (s[i] >= '0' && s[i] <= '9') {

MessageBoxA(NULL, "Цифры в поле", "Ошибка", MB\_ICONERROR | MB\_OK);

return true;

}

if (s[i] == ' ' ) {

MessageBoxA(NULL, "Пробел", "Ошибка", MB\_ICONERROR | MB\_OK);

return true;

}

}

return false;

}

void CKyrsov2020Dlg::OnBnClickedButton2()

{

UpdateData();

if (ec\_surname == "" || valid(ec\_surname) || ec\_name == "" || valid(ec\_name) || ec\_otc == "" || valid(ec\_otc)) { MessageBoxA(NULL, "Не заполненны поля: ФИО", "Ошибка", MB\_ICONERROR | MB\_OK); return; }

ec\_student.surname = ec\_surname; ec\_student.name = ec\_name; ec\_student.otc = ec\_otc;

if(ec\_year == 0 || ec\_day == 0 || ec\_month == 0 ){ MessageBoxA(NULL, "Не заполненны поля: Год рождения", "Ошибка", MB\_ICONERROR | MB\_OK); return; }

ec\_student.birthday[0] = ec\_day; ec\_student.birthday[1] = ec\_month; ec\_student.birthday[2] = ec\_year;

if (ec\_gender == "") { ec\_student.gender = "Н.у."; } else ec\_student.gender = ec\_gender;

if(ec\_faculty == "" || valid(ec\_faculty) || ec\_department == "" || ec\_year\_admission == 0) {MessageBoxA(NULL, "Не заполненны поля: Год поступления, Факультет, Кафедра", "Ошибка", MB\_ICONERROR | MB\_OK); return;}

ec\_student.faculty = ec\_faculty; ec\_student.department = ec\_department; ec\_student.year\_admission = ec\_year\_admission;

if (ec\_group == "" || ec\_number\_book == "") { MessageBoxA(NULL, "Не заполненны поля: Группа, Номер ЗК", "Ошибка", MB\_ICONERROR | MB\_OK); return;}

ec\_student.group = ec\_group; ec\_student.number\_book = ec\_number\_book;

UpdateData(false);

std::ofstream out("bd.txt", std::ios\_base::app);

if (!out.is\_open()) { MessageBoxA(NULL, "Не удалось открыть файл", "Fille not open", MB\_ICONERROR | MB\_OK); return; }

out << ec\_student.surname << " " << ec\_student.name << " " << ec\_student.otc << " " << ec\_student.birthday[0] << " " << ec\_student.birthday[1] << " " << ec\_student.birthday[2] << " "

<< ec\_student.gender << " " << ec\_student.year\_admission << " " << ec\_student.faculty << " " << ec\_student.department << " "

<< ec\_student.group << " " << ec\_student.number\_book << " ";

for (int i = 0; i < 9; i++) {

out << i << " ";

for (int j = 0; j < 10; j++)

{

out << ec\_student.progress[i].name\_subject[j] << " " << ec\_student.progress[i].value[j] << " ";

}

}

out << "\n";

out.close();

// TODO: добавьте свой код обработчика уведомлений

}

void CKyrsov2020Dlg::OnDtnDatetimechangeDatetimepicker1(NMHDR \*pNMHDR, LRESULT \*pResult)

{

LPNMDATETIMECHANGE pDTChange = reinterpret\_cast<LPNMDATETIMECHANGE>(pNMHDR);

// TODO: добавьте свой код обработчика уведомлений

UpdateData();

CTime m = ec\_birthday.GetTime();

ec\_year = m.GetYear();

ec\_day = m.GetDay();

ec\_month = m.GetMonth();

UpdateData(false);

\*pResult = 0;

}

void CKyrsov2020Dlg::OnBnClickedButton3()

{

UpdateData();

if (ec\_subject1 != "" || valid(ec\_subject1)) { ec\_student.progress[ec\_session].name\_subject[0] = ec\_subject1; ec\_student.progress[ec\_session].value[0] = ec\_value1; }

if (ec\_subject2 != "" || valid(ec\_subject2)) { ec\_student.progress[ec\_session].name\_subject[1] = ec\_subject2; ec\_student.progress[ec\_session].value[1] = ec\_value2; }

if (ec\_subject3 != "" || valid(ec\_subject3)) { ec\_student.progress[ec\_session].name\_subject[2] = ec\_subject3; ec\_student.progress[ec\_session].value[2] = ec\_value3; }

if (ec\_subject4 != "" || valid(ec\_subject4)) { ec\_student.progress[ec\_session].name\_subject[3] = ec\_subject4; ec\_student.progress[ec\_session].value[3] = ec\_value4; }

if (ec\_subject5 != "" || valid(ec\_subject5)) { ec\_student.progress[ec\_session].name\_subject[4] = ec\_subject5; ec\_student.progress[ec\_session].value[4] = ec\_value5; }

if (ec\_subject6 != "" || valid(ec\_subject6)) { ec\_student.progress[ec\_session].name\_subject[5] = ec\_subject6; ec\_student.progress[ec\_session].value[5] = ec\_value6; }

if (ec\_subject7 != "" || valid(ec\_subject7)) { ec\_student.progress[ec\_session].name\_subject[6] = ec\_subject7; ec\_student.progress[ec\_session].value[6] = ec\_value7; }

if (ec\_subject8 != "" || valid(ec\_subject8)) { ec\_student.progress[ec\_session].name\_subject[7] = ec\_subject8; ec\_student.progress[ec\_session].value[7] = ec\_value8; }

if (ec\_subject9 != "" || valid(ec\_subject9)) { ec\_student.progress[ec\_session].name\_subject[8] = ec\_subject9; ec\_student.progress[ec\_session].value[8] = ec\_value9; }

if (ec\_subject10 != "" || valid(ec\_subject10)) { ec\_student.progress[ec\_session].name\_subject[9] = ec\_subject10; ec\_student.progress[ec\_session].value[9] = ec\_value10; }

UpdateData(false);

// TODO: добавьте свой код обработчика уведомлений

}

void CKyrsov2020Dlg::OnBnClickedButton1()

{

// TODO: добавьте свой код обработчика уведомлений

if (m\_pDlg1 == NULL) {

m\_pDlg1 = new CDLG1(this);

m\_pDlg1->Create(IDD\_DIALOG1);

m\_pDlg1->createTable(m\_pDlg1->tableStudents);

}

else {

m\_pDlg1->SetActiveWindow();

}

m\_pDlg1->ShowWindow(SW\_SHOWNORMAL);

}

void CKyrsov2020Dlg::OnBnClickedButton4()

{

// TODO: добавьте свой код обработчика уведомлений

but\_Ref.EnableWindow(false); bt\_session.EnableWindow(true);

but\_write.EnableWindow(true); cb\_session.EnableWindow(true);

ct\_control.EnableWindow(true);

cbc\_day.EnableWindow(false); cbc\_month.EnableWindow(false); cbc\_year.EnableWindow(false);

UpdateData();

if (ec\_surname == "" || ec\_name == "" || ec\_otc == "") { MessageBoxA(NULL, "Не заполненны поля: ФИО", "Ошибка", MB\_ICONERROR | MB\_OK); return; }

m\_pDlg1->struct\_st.surname = ec\_surname; m\_pDlg1->struct\_st.name = ec\_name; m\_pDlg1->struct\_st.otc = ec\_otc;

if (ec\_year == 0 || ec\_day == 0 || ec\_month == 0) { MessageBoxA(NULL, "Не заполненны поля: Год рождения", "Ошибка", MB\_ICONERROR | MB\_OK); return; }

m\_pDlg1->struct\_st.birthday[0] = ec\_day; m\_pDlg1->struct\_st.birthday[1] = ec\_month; m\_pDlg1->struct\_st.birthday[2] = ec\_year;

if (ec\_gender == "") { m\_pDlg1->struct\_st.gender = "Не указан"; return;}

else m\_pDlg1->struct\_st.gender = ec\_gender;

if (ec\_faculty == "" || ec\_department == "" || ec\_year\_admission == 0) { MessageBoxA(NULL, "Не заполненны поля: Год поступления, Факультет, Кафедра", "Ошибка", MB\_ICONERROR | MB\_OK); return; }

m\_pDlg1->struct\_st.faculty = ec\_faculty; m\_pDlg1->struct\_st.department = ec\_department; m\_pDlg1->struct\_st.year\_admission = ec\_year\_admission;

if (ec\_group == "" || ec\_number\_book == "") { MessageBoxA(NULL, "Не заполненны поля: Группа, Номер ЗК", "Ошибка", MB\_ICONERROR | MB\_OK); return;}

m\_pDlg1->struct\_st.group = ec\_group; m\_pDlg1->struct\_st.number\_book = ec\_number\_book;

m\_pDlg1->struct\_st.progress[ec\_session].name\_subject[0] = ec\_subject1; m\_pDlg1->struct\_st.progress[ec\_session].value[0] = ec\_value1;

m\_pDlg1->struct\_st.progress[ec\_session].name\_subject[1] = ec\_subject2; m\_pDlg1->struct\_st.progress[ec\_session].value[1] = ec\_value2;

m\_pDlg1->struct\_st.progress[ec\_session].name\_subject[2] = ec\_subject3; m\_pDlg1->struct\_st.progress[ec\_session].value[2] = ec\_value3;

m\_pDlg1->struct\_st.progress[ec\_session].name\_subject[3] = ec\_subject4; m\_pDlg1->struct\_st.progress[ec\_session].value[3] = ec\_value4;

m\_pDlg1->struct\_st.progress[ec\_session].name\_subject[4] = ec\_subject5; m\_pDlg1->struct\_st.progress[ec\_session].value[4] = ec\_value5;

m\_pDlg1->struct\_st.progress[ec\_session].name\_subject[5] = ec\_subject6; m\_pDlg1->struct\_st.progress[ec\_session].value[5] = ec\_value6;

m\_pDlg1->struct\_st.progress[ec\_session].name\_subject[6] = ec\_subject7; m\_pDlg1->struct\_st.progress[ec\_session].value[6] = ec\_value7;

m\_pDlg1->struct\_st.progress[ec\_session].name\_subject[7] = ec\_subject8; m\_pDlg1->struct\_st.progress[ec\_session].value[7] = ec\_value8;

m\_pDlg1->struct\_st.progress[ec\_session].name\_subject[8] = ec\_subject9; m\_pDlg1->struct\_st.progress[ec\_session].value[8] = ec\_value9;

m\_pDlg1->struct\_st.progress[ec\_session].name\_subject[9] = ec\_subject10; m\_pDlg1->struct\_st.progress[ec\_session].value[9] = ec\_value10;

m\_pDlg1->RewritingStudent(ec\_session);

m\_pDlg1->Rewriting();

m\_pDlg1->b\_creat.EnableWindow(true);

m\_pDlg1->bt\_delete.EnableWindow(true);

m\_pDlg1->OnBnClickedButton1();

UpdateData(false);

}

**Исходный файл CDLG1.cpp**

list<clStudent> baseStudent;

void ReadFile() {

setlocale(LC\_ALL, "Rus");

CStdioFile reader;

CFileException fileEx;

CString s;

if (reader.Open(\_T("bd.txt"), CFile::modeRead, &fileEx)) {

while (reader.ReadString(s)) {

baseStudent.push\_back(clStudent(s));

}

}

reader.Close();

}

void CDLG1::OnBnClickedButton1()

{

// TODO: добавьте свой код обработчика уведомлени

UpdateData();

b\_creat.EnableWindow(true);

bt\_delete.EnableWindow(true);

tableStudents.DeleteAllItems();

baseStudent.clear();

ReadFile();

int row = 0;

for (list<clStudent>::iterator i = baseStudent.begin(); i != baseStudent.end(); i++) {

(\*i).addToTable(tableStudents, row, cb\_session); row++;

}

buttton\_Search.EnableWindow(true);

numberRedact = 0;

UpdateData(false);

}

void CDLG1::OnBnClickedButton2()

{

UpdateData();

if (numberRedact == 0 || numberRedact > (&baseStudent)->size()) { MessageBoxA(NULL, "Введенного номера не существует", "Редоктирование", MB\_ICONERROR | MB\_OK); }

else {

b\_creat.EnableWindow(false);

bt\_delete.EnableWindow(false);

list<clStudent>::iterator i = baseStudent.begin();

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

m\_pCKyrsov2020Dlg->cb\_session.EnableWindow(false);

m\_pCKyrsov2020Dlg->but\_Ref.EnableWindow(true);

m\_pCKyrsov2020Dlg->but\_write.EnableWindow(false);

m\_pCKyrsov2020Dlg->bt\_session.EnableWindow(false);

m\_pCKyrsov2020Dlg->ec\_surname = (\*i).surname;

m\_pCKyrsov2020Dlg->ec\_name = (\*i).name;

m\_pCKyrsov2020Dlg->ec\_otc = (\*i).otc;

m\_pCKyrsov2020Dlg->ct\_control.EnableWindow(false);

m\_pCKyrsov2020Dlg->cbc\_day.EnableWindow(true);m\_pCKyrsov2020Dlg->cbc\_month.EnableWindow(true);m\_pCKyrsov2020Dlg->cbc\_year.EnableWindow(true);

m\_pCKyrsov2020Dlg->ec\_day = (\*i).birthday[0]; m\_pCKyrsov2020Dlg->ec\_month = (\*i).birthday[1]; m\_pCKyrsov2020Dlg->ec\_year = (\*i).birthday[2];

m\_pCKyrsov2020Dlg->ec\_gender = (\*i).gender;

m\_pCKyrsov2020Dlg->ec\_year\_admission = (\*i).year\_admission;

m\_pCKyrsov2020Dlg->ec\_faculty = (\*i).faculty; m\_pCKyrsov2020Dlg->ec\_department = (\*i).department;

m\_pCKyrsov2020Dlg->ec\_group = (\*i).group; m\_pCKyrsov2020Dlg->ec\_number\_book = (\*i).number\_book;

m\_pCKyrsov2020Dlg->rewriting = true;

m\_pCKyrsov2020Dlg->ec\_session = cb\_session;

m\_pCKyrsov2020Dlg->fillSession();

m\_pCKyrsov2020Dlg->ec\_subject1 = (\*i).progress[cb\_session ].name\_subject[0]; m\_pCKyrsov2020Dlg->ec\_value1 = (\*i).progress[cb\_session].value[0];

m\_pCKyrsov2020Dlg->ec\_subject2 = (\*i).progress[cb\_session ].name\_subject[1]; m\_pCKyrsov2020Dlg->ec\_value2 = (\*i).progress[cb\_session].value[1];

m\_pCKyrsov2020Dlg->ec\_subject3 = (\*i).progress[cb\_session].name\_subject[2]; m\_pCKyrsov2020Dlg->ec\_value3 = (\*i).progress[cb\_session].value[2];

m\_pCKyrsov2020Dlg->ec\_subject4 = (\*i).progress[cb\_session].name\_subject[3]; m\_pCKyrsov2020Dlg->ec\_value4 = (\*i).progress[cb\_session].value[3];

m\_pCKyrsov2020Dlg->ec\_subject5 = (\*i).progress[cb\_session].name\_subject[4]; m\_pCKyrsov2020Dlg->ec\_value5 = (\*i).progress[cb\_session].value[4];

m\_pCKyrsov2020Dlg->ec\_subject6 = (\*i).progress[cb\_session].name\_subject[5]; m\_pCKyrsov2020Dlg->ec\_value6 = (\*i).progress[cb\_session].value[5];

m\_pCKyrsov2020Dlg->ec\_subject7 = (\*i).progress[cb\_session].name\_subject[6]; m\_pCKyrsov2020Dlg->ec\_value7 = (\*i).progress[cb\_session].value[6];

m\_pCKyrsov2020Dlg->ec\_subject8 = (\*i).progress[cb\_session].name\_subject[7]; m\_pCKyrsov2020Dlg->ec\_value8 = (\*i).progress[cb\_session].value[7];

m\_pCKyrsov2020Dlg->ec\_subject9 = (\*i).progress[cb\_session].name\_subject[8]; m\_pCKyrsov2020Dlg->ec\_value9 = (\*i).progress[cb\_session].value[8];

m\_pCKyrsov2020Dlg->ec\_subject10 = (\*i).progress[cb\_session].name\_subject[9]; m\_pCKyrsov2020Dlg->ec\_value10 = (\*i).progress[cb\_session].value[9];

m\_pCKyrsov2020Dlg->UpdateData(false);

}

}

void CDLG1::RewritingStudent(int curSession) {

list<clStudent>::iterator i = baseStudent.begin();

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

(\*i).surname = struct\_st.surname;

(\*i).name = struct\_st.name;

(\*i).otc = struct\_st.otc;

(\*i).birthday[0] = struct\_st.birthday[0]; (\*i).birthday[1] = struct\_st.birthday[1]; (\*i).birthday[2] = struct\_st.birthday[2];

(\*i).gender = struct\_st.gender;

(\*i).year\_admission = struct\_st.year\_admission;

(\*i).faculty = struct\_st.faculty; (\*i).department = struct\_st.department;

(\*i).group = struct\_st.group; (\*i).number\_book = struct\_st.number\_book;

for (int k = 0; k < 10; k++) {

(\*i).progress[curSession].name\_subject[k] = struct\_st.progress[curSession].name\_subject[k];

(\*i).progress[curSession].value[k] = struct\_st.progress[curSession].value[k];

}

}

void CDLG1::Rewriting()

{

ofstream out("bd.txt");

for (list<clStudent>::iterator i = baseStudent.begin(); i != baseStudent.end(); i++) {

out << (CT2CA) (\*i).surname << " " << (CT2CA)(\*i).name << " " << (CT2CA)(\*i).otc << " "

<< (\*i).birthday[0] << " " << (\*i).birthday[1] << " " << (\*i).birthday[2] << " "

<< (CT2CA)(\*i).gender << " " << (\*i).year\_admission << " " << (CT2CA)(\*i).faculty << " "

<< (CT2CA)(\*i).department << " " << (CT2CA)(\*i).group << " " << (CT2CA)(\*i).number\_book << " ";

for (int s = 0; s < 9; s++) {

out << s << " ";

for (int j = 0; j < 10; j++)

{

out << (CT2CA)(\*i).progress[s].name\_subject[j] << " " << (\*i).progress[s].value[j] << " ";

}

}

out << "\n";

}

out.close();

}

void CDLG1::OnBnClickedButton3()

{

UpdateData(true);

if (ec\_inter\_do == 0 || ec\_inter\_ot > ec\_inter\_do) { MessageBoxA(NULL, "Ошибка интервала", "Выборка", MB\_ICONERROR | MB\_OK); return; }

tableStudents.DeleteAllItems();

int row = 0;

// TODO: добавьте свой код обработчика уведомлений

for (list<clStudent>::iterator i = baseStudent.begin(); i != baseStudent.end(); i++) {

int value3 = 0, value4 = 0, value5 = 0;

for (int j = 0; j < 9; j++) {

for (int k = 0; k < 10; k++)

{

switch ((\*i).progress[j].value[k] && ec\_inter\_ot <= (\*i).birthday[2] && ec\_inter\_do > (\*i).birthday[2])

{

case 3: {value3++; break; }

case 4: {value4++; break; }

case 5: {value5++; break; }

default:break;

}

}

}

int result = 0;

if (cb\_value3)result += value3;

if (cb\_value4)result += value4;

if (cb\_value5)result += value5;

if (result <= (value3 + value4 + value5)\*0.25) {

(\*i).addToTable(tableStudents, row, cb\_session); row++;

}

}

UpdateData(false);

}

void CDLG1::OnBnClickedButton4()

{

if (numberRedact == 0 || numberRedact > (&baseStudent)->size()) { MessageBoxA(NULL, "Введенного номера не существует", "Редоктирование", MB\_ICONERROR | MB\_OK); return; }

ofstream out("bd.txt");

int k = 1;

for (list<clStudent>::iterator i = baseStudent.begin(); i != baseStudent.end(); i++,k++) {

if(k != numberRedact){

out << (CT2CA)(\*i).surname << " " << (CT2CA)(\*i).name.GetString() << " " << (CT2CA)(\*i).otc.GetString() << " "

<< (\*i).birthday[0] << " " << (\*i).birthday[1] << " " << (\*i).birthday[2] << " "

<< (CT2CA)(\*i).gender.GetString() << " " << (\*i).year\_admission << " " << (CT2CA)(\*i).faculty.GetString() << " "

<< (CT2CA)(\*i).department.GetString() << " " << (CT2CA)(\*i).group.GetString() << " " << (CT2CA)(\*i).number\_book.GetString() << " ";

for (int s = 0; s < 9; s++) {

out << s << " ";

for (int j = 0; j < 10; j++)

{

out << (CT2CA)(\*i).progress[s].name\_subject[j].GetString() << " " << (\*i).progress[s].value[j] << " ";

}

}

out << "\n";

}

}

out.close();

OnBnClickedButton1();

}

void CDLG1::OnBnClickedButton5()

{

ofstream out("crypt\_bd.txt");

for (list<clStudent>::iterator i = baseStudent.begin(); i != baseStudent.end(); i++) {

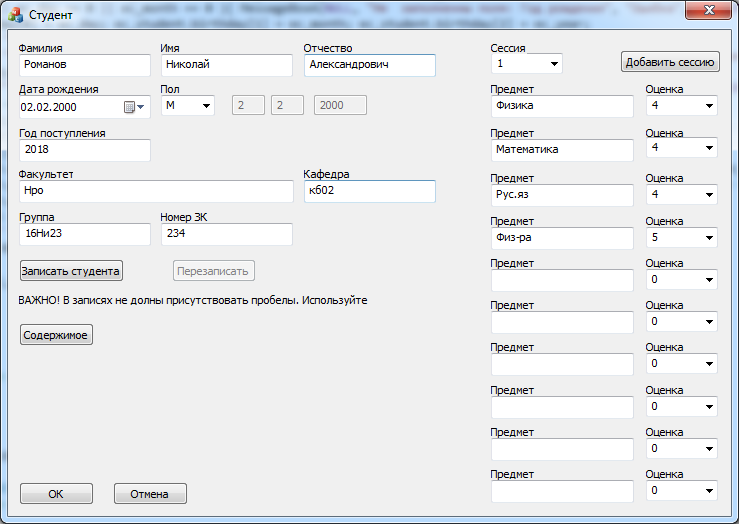
out << (\*i).Crypting((\*i).StudentToStr(0)) << "\n";

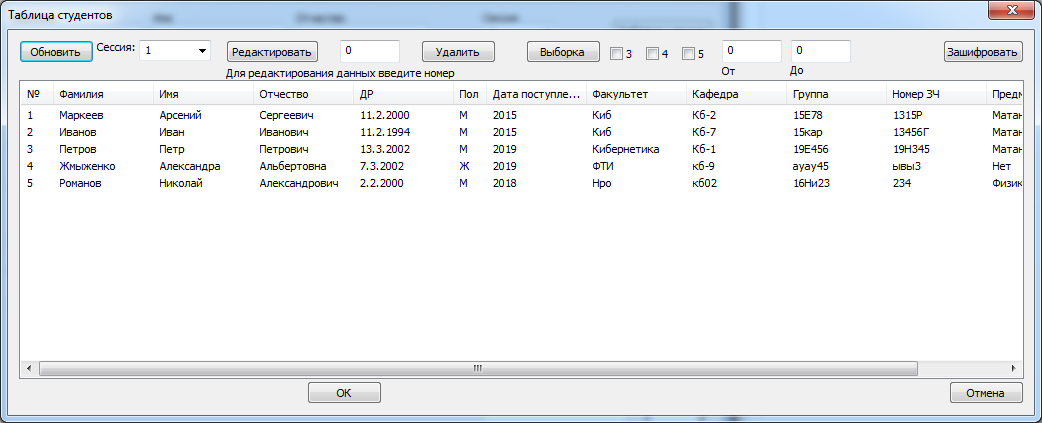
}

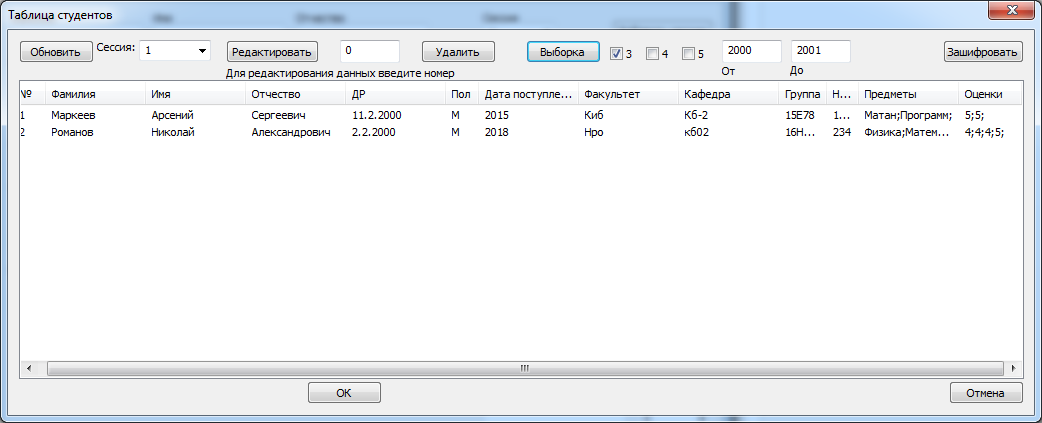
out.close();

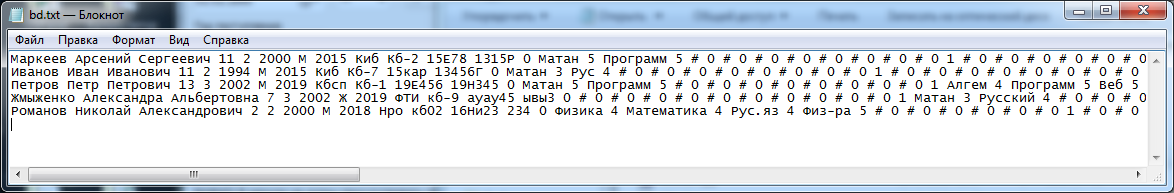
}

# **Распечатка данных**









**Вывод**

Я ознакомился библиотекой классов MFC, применение стандартных элементов управления. Также получил знания основ работы с визуальной средой программирования, как осуществляется работа в архитектуре “документ-представление” и механизм создания диалоговых окон.

Приобрел умения: создавать программное обеспечение с использованием различной архитектуры составления программ в среде Visual Studio, осуществлять проектирование программного продукта, обрабатывать сообщения, выполнять действия с различными элементами среды, разрабатывать прикладной и пользовательский интерфейс.

# **Список литературы**

* Бьёрн Страуструп. Язык программирования C++ = The C++ Programming Language / Пер. с англ. — 3-е изд. — СПб.; М.: Невский диалект — Бином, 1999.
* Сиддхартха Рао. Освой самостоятельно C++ за 21 день, 7-е издание (C++11) = Sams Teach Yourself C++ in One Hour a Day, 7th Edition. — М.: «Вильямс», 2013.
* Айвор Хортон. Visual C++ 2010: полный курс = Ivor Horton’s Beginning Visual C++ 2010. — М.: Диалектика, 2010.

**Использованные источники**

* Создание приложения MFC [Электронный ресурс] Режим доступа: <https://docs.microsoft.com/ru-ru/cpp/mfc/reference/creating-an-mfc-application?view=vs-2019>
* MFC – Краткое руководство [Электронный ресурс] Режим доступа: <https://coderlessons.com/tutorials/microsoft-technologies/izuchite-mfts/mfc-kratkoe-rukovodstvo>
* Введение в Visual C++. MFC [Электронный ресурс] Режим доступа: https://kvodo.ru/visual-c-mfc-vvedenie.html