

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра РАПС

КУРСОВАЯ РАБОТА
по дисциплине «Программирование и основы алгоритмизации»
Тема: «Перечень станков производственного участка»

Студент гр.3487

Артемьев А.С.

Преподаватель

Армашев А.А.

Санкт-Петербург

2024

СОДЕРЖАНИЕ

1. Введение.....	3
2. Разработка структуры приложения.....	3
3. Разработка структуры данных.....	4
4. Разработка интерфейса.....	5
5. Заключение.....	5
6. Приложение.....	6

ВВЕДЕНИЕ

Легенда курсового приложения: создание приложения для работы с перечнем станков производственного участка.

Разработка структуры приложения

Функции:

`void __fastcall RemoveMachineByNumber(TListView* listView,
bool isChecked1, bool isChecked2, bool isChecked3)` – Удаление строки по номеру;

`void __fastcall SearchFromMachineList(TListView* listView,
const std::list<std::unique_ptr<Machine>>& machineList,
bool& isChecked1, bool& isChecked2, bool& isChecked3)` – поиск

`void __fastcall FillListViewFromMachineList(TListView* listView,
const std::list<std::unique_ptr<Machine>>& machineList, bool&
isChecked1,
bool& isChecked2, bool& isChecked3)` – заполнение ListView

`void __fastcall
FillMachineList(std::list<std::unique_ptr<Machine>>& machineList,
const std::string& filePath)` – заполнение списка объектов класса из файла;

`void __fastcall SaveListViewToFile(TListView* listView)` – сохранение в файл;

`void __fastcall LoadDataFromMachineList(const
std::list<std::unique_ptr<Machine>>& machineList,
int rowNumber)` - загрузка данных из основной формы в форму редактирования;

метод:

```
virtual void AddToListView(TListView* listView, int rowNum)
```

const - заполнение строки ListView элементами класса;

Разработка структуры данных

class Machine – главный класс станков;

class CNC : public Machine – дочерний класс станков с ЧПУ;

class Manual : public Machine – дочерний класс станков с руч.
управлением;

Разработка интерфейса

Интерфейс главной формы:

№	Название	Тип упр.	обр. материал	Исправность	год выпуска	ФИО рабочего	система ЧПУ
1	Фрезерный станок	ЧПУ	универсал	исправный	2012	-	Mitsubishi M80A
2	Фрезерный станок	ручное	металл	исправный	1972	Иванов А. М.	-
3	Фрезерный станок	ЧПУ	дерево	неисправный	2002	-	SYNTEC
4	Токарный станок	ЧПУ	универсал	исправный	2018	-	Siemens
5	Токарный станок	ручное	дерево	исправный	1999	Петров С. И.	-
6	Ленточная пила	ручное	универсал	исправный	2015	Семенов И. И.	-
7	Маятниковая пила	ручное	универсал	неисправный	2017	Игнатов Г. М.	-
8	Пескоструйный станок	ручное	металл	исправный	2020	Кулинбин И. П.	-
9	Станок лазерной резки	ЧПУ	металл	исправный	2024	-	Lantek Expert Cut
10	Сверлильный станок	ручное	металл	исправный	1985	Сидоров В. П.	-
11	Шлифовальный станок	ручное	универсал	исправный	2005	Петрова А. Н.	-
12	Сверильнорасточной ...	ЧПУ	металл	исправный	2021	-	FANUC
13	Фрезерногравироваль...	ЧПУ	дерево	исправный	2019	-	Roland
14	Токарнорезьбонарез...	ручное	металл	неисправный	1968	Васильев Ф. И.	-
15	Лесопилка	ручное	дерево	исправный	2010	Иванова Е. А.	-
16	Циркулярная пила	ручное	универсал	исправный	1995	Соловьев М. Ю.	-
17	Гриндер	ручное	дерево	исправный	2008	Козлов А. Г.	-
18	Шлифовальный станок	ручное	металл	исправный	1978	Егоров А. В.	-
19	Пресс	ручное	металл	неисправный	1980	Петров И. И.	-
20	Гибочный станок	ЧПУ	металл	исправный	2023	-	Bystronic
21	Шлифовальнополиров...	ручное	металл	неисправный	1990	Смирнов П. П.	-
22	Электроэрозионная ма...	ЧПУ	металл	исправный	2012	-	Sodick

На этой форме находятся основные элементы управления программой и таблица для вывода самой базы данных.

Вторичная форма редактирования и добавления строк:

Добавление/редактирование данных

Введите данные:

Обрабатываемый материал: ☐ Металл ☐ Дерево ☒ Универсал

Название станка:

Год выпуска:

ПО / ФИО:

☐ Наличие ЧПУ

☒ Необходим ремонт

Вторичная форма выбора принципа сортировки списка;

Сортировка

☐ По названию

☐ По году выпуска

☐ В обратном порядке

ЗАКЛЮЧЕНИЕ

В рамках проекта была создана программа для комплексной работы с данными о станках. В процессе разработки я освоил работу с классами в среде C++ Builder, а также приобрел ценные навыки использования этой среды программирования.

ПРИЛОЖЕНИЕ

Текстовый документ:

Фрезерный станок, ЧПУ, универсал, исправный, 2012, Mitsubishi M80A
Фрезерный станок, ручное, металл, исправный, 1972, Иванов А. М.
Фрезерный станок, ЧПУ, дерево, неисправный, 2002, SYNTEC
Токарный станок, ЧПУ, универсал, исправный, 2018, Siemens
Токарный станок, ручное, дерево, исправный, 1999, Петров С. И.
Ленточная пила, ручное, универсал, исправный, 2015, Семёнов И. И.
Сверлильный станок, ручное, металл, исправный, 1985, Сидоров В. П.
Сверлильнорасточной станок, ЧПУ, металл, исправный, 2021, FANUC
Лесопилка, ручное, дерево, исправный, 2010, Иванова Е. А.
Гриндер, ручное, дерево, исправный, 2008, Козлов А. Г.
Шлифовальный станок, ручное, металл, исправный, 1978, Егоров А. В.
Пресс, ручное, металл, неисправный, 1980, Петров И. И.
Гибочный станок, ЧПУ, металл, исправный, 2023, Bystronic
Электроэрозионная машина, ЧПУ, металл, исправный, 2012, Sodick

Unit1.h:

```
#ifndef Unit1H
#define Unit1H

#include <System.Classes.hpp>
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
#include <Vcl.ComCtrls.hpp>
#include <Vcl.ExtCtrls.hpp>
#include <vcl.h>
#include <string>
#include <fstream>
#include <iostream>
#include <System.hpp>
#include <list>
#include <memory>
#include "Unit3.h"
```

```

class Machine {
private:
    UnicodeString name;
    UnicodeString type_of_control;
    UnicodeString material;
    UnicodeString repair;
    int year_of_release;

public:
    Machine() : name(""), type_of_control(""), material(""),
repair(""), year_of_release(0) {}

    Machine(const UnicodeString& name, const UnicodeString&
type_of_control, const UnicodeString& material,
            const UnicodeString& repair, int year_of_release)
        : name(name), type_of_control(type_of_control),
material(material), repair(repair),
year_of_release(year_of_release) {}

    virtual ~Machine() {}

    UnicodeString getName() const { return name; }
    UnicodeString getTypeOfControl() const { return
type_of_control; }
    UnicodeString getMaterial() const { return material; }
    UnicodeString getRepair() const { return repair; }
    int getYearOfRelease() const { return year_of_release; }

    void setName(const UnicodeString& newName) { name = newName;
}

    void setTypeOfControl(const UnicodeString& newTypeOfControl)
{ type_of_control = newTypeOfControl; }
    void setMaterial(const UnicodeString& newMaterial) { material
= newMaterial; }

```

```

        void setRepair(const UnicodeString& newRepair) { repair =
newRepair; }

        void setYearOfRelease(int newYearOfRelease) { year_of_release
= newYearOfRelease; }

        virtual void AddToListView(TListView* listView, int
rowNumber) const {
            TListItem* item = listView->Items->Add();
            item->Caption = IntToStr(rowNumber);
            item->SubItems->Add(name);
            item->SubItems->Add(type_of_control);
            item->SubItems->Add(material);
            item->SubItems->Add(repair);
            item->SubItems->Add(String(year_of_release));
            item->SubItems->Add("-");
            item->SubItems->Add("-");
        }
};

class CNC : public Machine {
private:
    UnicodeString software;

public:
    CNC() : software("") {}

    CNC(const UnicodeString& name, const UnicodeString&
type_of_control, const UnicodeString& material,
        const UnicodeString& repair, int year_of_release, const
UnicodeString& software)
        : Machine(name, type_of_control, material, repair,
year_of_release), software(software) {}

    UnicodeString getSoftware() const { return software; }

```



```

        void setSoftware(const UnicodeString& newSoftware) { software
= newSoftware; }

```

```

        void AddToListView(TListView* listView, int rowNumber) const
override{
            TListItem* item = listView->Items->Add();
            item->Caption = IntToStr(rowNumber);
            item->SubItems->Add(getName());
            item->SubItems->Add(getTypeOfControl());
            item->SubItems->Add(getMaterial());
            item->SubItems->Add(getRepair());
            item->SubItems->Add(String(getYearOfRelease()));
            item->SubItems->Add("-");
            item->SubItems->Add(software);
        }
};

```

```

class Manual : public Machine {
private:

```

```

    UnicodeString FCs;

```

```

public:

```

```

    Manual() : Machine(), FCs("") {}

```

```

    Manual(const UnicodeString& name, const UnicodeString&
type_of_control, const UnicodeString& material,
            const UnicodeString& repair, int year_of_release,
const UnicodeString& FCs)
        : Machine(name, type_of_control, material, repair,
year_of_release), FCs(FCs) {}

```

```

    UnicodeString getFCs() const { return FCs; }

```

```

    void setFCs(const UnicodeString& newFCs) { FCs = newFCs; }

```

```

        void AddToListView(TListView* listView, int rowNumber) const
override{
    TListItem* item = listView->Items->Add();
    item->Caption = IntToStr(rowNumber);
    item->SubItems->Add(getName());
    item->SubItems->Add(getTypeOfControl());
    item->SubItems->Add(getMaterial());
    item->SubItems->Add(getRepair());
    item->SubItems->Add(String(getYearOfRelease()));
    item->SubItems->Add(FCs);
    item->SubItems->Add("-");
}
};

```

```

class TForm1 : public TForm {
__published:
    TListView *ListView1;

    TButton *Button1;
    TButton *Button2;
    TButton *Button3;
    TButton *Button4;
    TButton *Button5;
    TButton *Button6;
    TButton *Button9;
    TButton *Button10;

    TPaintBox *PaintBox1;

    TLabel *Label1;

    TCheckBox *CheckBox1;
    TCheckBox *CheckBox2;
    TCheckBox *CheckBox3;

```

```

TEdit *Edit1;
TEdit *Edit2;
TEdit *Edit3;

void __fastcall Button1Click(TObject *Sender);
void __fastcall Button2Click(TObject *Sender);
void __fastcall Button3Click(TObject *Sender);
void __fastcall Button4Click(TObject *Sender);
void __fastcall Button6Click(TObject *Sender);
void __fastcall Button5Click(TObject *Sender);
void __fastcall Button9Click(TObject *Sender);
void __fastcall Button10Click(TObject *Sender);

void __fastcall PaintBox1Paint(TObject *Sender);

private:

    void __fastcall
FillMachineList(std::list<std::unique_ptr<Machine>>& machineList,
const std::string& filePath);

    void __fastcall FillListViewFromMachineList(TListView*
listView,
const std::list<std::unique_ptr<Machine>>& machineList, bool&
isChecked1,
bool& isChecked2, bool& isChecked3);

    void __fastcall SearchFromMachineList(TListView* listView,
const std::list<std::unique_ptr<Machine>>& machineList,
bool& isChecked1, bool& isChecked2, bool& isChecked3);

    void __fastcall RemoveMachineByNumber(TListView* listView,
bool isChecked1, bool isChecked2, bool isChecked3);

```

```

void __fastcall SaveListViewToFile(TListView* listView);

std::list<std::unique_ptr<Machine>> machineList;

bool isChecked1;
bool isChecked2;
bool isChecked3;
public:
    __fastcall TForm1(TComponent* Owner);
};

extern PACKAGE TForm1 *Form1;

#endif

```

Unit1.cpp:

```

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "Unit3.h"
#include "Unit4.h"

#include <cstdio>
#include <fstream>
#include <sstream>
#include <iostream>
#include <string>
#include <locale>
#include <codecvt>
#include <windows.h>
#include <SysUtils.hpp>
#include <memory>
#include <list>
#include <System.SysUtils.hpp>

```

```

#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;

//заполнение класса из файла
void __fastcall
TForm1::FillMachineList(std::list<std::unique_ptr<Machine>>&
machineList,
const std::string& filePath) {

    FILE* file = fopen(filePath.c_str(), "r");
    if (!file) {
        ShowMessage("Не удалось открыть файл");
        return;
    }

    char line[1024];
    while (fgets(line, sizeof(line), file)) {

        size_t len = strlen(line);
        if (len > 0 && (line[len - 1] == '\n' || line[len - 1] ==
'\r')) {
            line[len - 1] = '\0';
        }

        int wideCharSize = MultiByteToWideChar(CP_UTF8, 0, line,
-1, NULL, 0);
        if (wideCharSize <= 0) {
            continue;
        }
        std::wstring wideString(wideCharSize, L'\0');
        MultiByteToWideChar(CP_UTF8, 0, line, -1, &wideString[0],
wideCharSize);
    }
}

```

```

        std::wstring machineType, controlType, material, repair,
additional;
        int yearOfRelease = 0;

        wchar_t machineTypeBuffer[256], controlTypeBuffer[256],
materialBuffer[256],
            repairBuffer[256], additionalBuffer[256];

        if (swscanf(wideString.c_str(), L"%255[^,], %255[^,],
%255[^,], %255[^,], %d, %255[^\n]",
            machineTypeBuffer, controlTypeBuffer,
materialBuffer, repairBuffer,
            &yearOfRelease, additionalBuffer) < 6) {
            continue;
        }

        machineType = machineTypeBuffer;
        controlType = controlTypeBuffer;
        material = materialBuffer;
        repair = repairBuffer;
        additional = additionalBuffer;

        controlType.erase(0, controlType.find_first_not_of(L'
''));
        controlType.erase(controlType.find_last_not_of(L' ') +
1);

        if (controlType == L"ЧПУ") {
            machineList.push_back(std::make_unique<CNC>(
                UnicodeString(machineType.c_str()),
UnicodeString(controlType.c_str()),
                UnicodeString(material.c_str()),
UnicodeString(repair.c_str()), yearOfRelease,
                UnicodeString(additional.c_str())));

```

```

        } else if (controlType == L"ручное") {
            machineList.push_back(std::make_unique<Manual>(
                UnicodeString(machineType.c_str()),
UnicodeString(controlType.c_str()),
                UnicodeString(material.c_str()),
UnicodeString(repair.c_str()), yearOfRelease,
                UnicodeString(additional.c_str())));
        }
    }

    fclose(file);
}

//ВЫВОД В ЛИСТ ИЗ КЛАССА
void __fastcall TForm1::FillListViewFromMachineList(TListView*
listView,
    const std::list<std::unique_ptr<Machine>>& machineList,
    bool& isChecked1, bool& isChecked2, bool& isChecked3) {

    listView->Items->Clear();
    int rowNumber = 1;

    for (const auto& machine : machineList) {
        bool addItem = false;

        if (!isChecked1 && !isChecked2 && !isChecked3) {
            addItem = true;
        } else if (isChecked1 && !isChecked2 && !isChecked3) {
            addItem = machine->getTypeOfControl() == "ЧПУ";
        } else if (isChecked1 && !isChecked2 && isChecked3) {
            addItem = (machine->getTypeOfControl() == "ЧПУ" &&
machine->getRepair() == "неисправный");
        } else if (!isChecked1 && isChecked2 && isChecked3) {
            addItem = (machine->getTypeOfControl() == "ручное"
&& machine->getRepair() == "неисправный");

```

```

        } else if (!isChecked1 && isChecked2 && !isChecked3) {
            addItem = machine->getTypeOfControl() == "ручное";
        } else if (!isChecked1 && !isChecked2 && isChecked3) {
            addItem = machine->getRepair() == "неисправный";
        }

        if (addItem) {
            machine->AddToListView(listView, rowNumber);
            rowNumber++;
        }
    }

    for (int i = 0; i < ComponentCount; i++) {
        if (dynamic_cast<TEdit*>(Components[i])) {
            TEdit* edit = static_cast<TEdit*>(Components[i]);
            edit->Text = "";
        }
    }
}

//поиск
void __fastcall TForm1::SearchFromMachineList(TListView* listView,
const std::list<std::unique_ptr<Machine>>& machineList,
bool& isChecked1, bool& isChecked2, bool& isChecked3){

    String searchText = Edit2->Text.Trim();
    if (searchText.IsEmpty()) {
        ShowMessage(L"Введите текст для поиска!");
        FillListViewFromMachineList(ListView1, machineList,
isChecked1, isChecked2, isChecked3);
        return;
    }

    ListView1->Items->Clear();
    int rowNumber = 1;

```



```

for (const auto& machine : machineList) {
    bool addItem = false;

    if (!isChecked1 && !isChecked2 && !isChecked3) {
        addItem = true;
    } else if (isChecked1 && !isChecked2 && !isChecked3) {
        addItem = machine->getTypeOfControl() == "ЧПУ";
    } else if (isChecked1 && !isChecked2 && isChecked3) {
        addItem = (machine->getTypeOfControl() == "ЧПУ" &&
machine->getRepair() == "неисправный");
    } else if (!isChecked1 && isChecked2 && isChecked3) {
        addItem = (machine->getTypeOfControl() == "ручное"
&& machine->getRepair() == "неисправный");
    } else if (!isChecked1 && isChecked2 && !isChecked3) {
        addItem = machine->getTypeOfControl() == "ручное";
    } else if (!isChecked1 && !isChecked2 && isChecked3) {
        addItem = machine->getRepair() == "неисправный";
    }

    if (addItem) {
        if (machine->getName().Pos(searchText) > 0) {
            machine->AddToListView(ListView1, rowNumber);
            rowNumber++;
        }
    }
}

if (ListView1->Items->Count == 0) {
    ShowMessage(L"Совпадений не найдено.");
    FillListViewFromMachineList(ListView1, machineList,
isChecked1, isChecked2, isChecked3);
}
}

```

```

//сохранение в файл
void __fastcall TForm1::SaveListViewToFile(TListView* listView) {
    const wchar_t* filePath =
L"C:\\Users\\user\\Desktop\\ппора\\кypсач\\machines.txt";

    FILE* file;
    if (_wfopen_s(&file, filePath, L"wb") != 0) {
        ShowMessage(L"Не удалось открыть файл для записи");
        return;
    }

    for (int i = 0; i < listView->Items->Count; ++i) {
        TListItem* item = listView->Items->Item[i];
        if (item == nullptr) continue;

        bool isFirst = true;

        for (int j = 0; j < item->SubItems->Count; ++j) {
            UnicodeString subItem = item->SubItems->Strings[j];
            subItem = StringReplace(subItem, L"-", L"",
TReplaceFlags() << rfReplaceAll);

            if (subItem.IsEmpty()) continue;

            if (!isFirst) {
                fwrite(", ", sizeof(char), 2, file);
            } else {
                isFirst = false;
            }

            int utf8Size = WideCharToMultiByte(CP_UTF8, 0,
subItem.c_str(), -1, NULL, 0, NULL, NULL);
            if (utf8Size > 0) {
                char* utf8Buffer = new char[utf8Size];

```

```

        WideCharToMultiByte(CP_UTF8, 0,
subItem.c_str(), -1, utf8Buffer, utf8Size, NULL, NULL);
        fwrite(utf8Buffer, sizeof(char), utf8Size - 1,
file);

        delete[] utf8Buffer;
    }
}

    fwrite("\n", sizeof(char), 1, file);
}

fclose(file);
ShowMessage(L"Данные успешно сохранены в файл");
}

```

```

//сортировки
//по названию
void SortMachineListByName(std::list<std::unique_ptr<Machine>>&
machineList, bool isReverse) {
    if (machineList.empty()) return;

    if (isReverse) {
        bool swapped;
        do {
            swapped = false;
            for (auto it1 = machineList.begin(), it2 =
std::next(it1); it2 != machineList.end(); ++it1, ++it2) {
                if ((*it1)->getName() < (*it2)->getName()) {
                    std::swap(*it1, *it2);
                    swapped = true;
                }
            }
        } while (swapped);
    } else{

```

```

        bool swapped;
        do {
            swapped = false;
            for (auto it1 = machineList.begin(), it2 =
std::next(it1); it2 != machineList.end(); ++it1, ++it2) {
                if ((*it1)->getName() > (*it2)->getName()) {
                    std::swap(*it1, *it2);
                    swapped = true;
                }
            }
        } while (swapped);
    }

}

//по году
void SortMachineListByYear(std::list<std::unique_ptr<Machine>>&
machineList, bool isReverse) {
    if (machineList.empty()) return;

    if (isReverse) {
        bool swapped;
        do {
            swapped = false;
            for (auto it1 = machineList.begin(), it2 =
std::next(it1); it2 != machineList.end(); ++it1, ++it2) {
                if ((*it1)->getYearOfRelease() > (*it2)-
>getYearOfRelease()) {
                    std::swap(*it1, *it2);
                    swapped = true;
                }
            }
        } while (swapped);
    }else{
        bool swapped;
        do {

```

```

        swapped = false;
        for (auto it1 = machineList.begin(), it2 =
std::next(it1); it2 != machineList.end(); ++it1, ++it2) {
            if ((*it1)->getYearOfRelease() < (*it2)-
>getYearOfRelease()) {
                std::swap(*it1, *it2);
                swapped = true;
            }
        }
    } while (swapped);
}

//удаление
void __fastcall TForm1::RemoveMachineByNumber(TListView* listView,
bool isChecked1, bool isChecked2, bool isChecked3)
{
    String inputText = Edit3->Text.Trim();
    if (inputText.IsEmpty()) {
        ShowMessage(L"Введите номер станка для удаления!");
        return;
    }

    int machineNumber;
    if (!TryStrToInt(inputText, machineNumber) || machineNumber
<= 0) {
        ShowMessage(L"Введите корректный номер станка!");
        return;
    }

    auto it = machineList.begin();
    int currentIndex = 1;

    while (it != machineList.end()) {
        bool matchesFilter = false;

```

```

        if (!isChecked1 && !isChecked2 && !isChecked3) {
            matchesFilter = true;
        } else if (isChecked1 && !isChecked2 && !isChecked3) {
            matchesFilter = (*it)->getTypeOfControl() == "ЧПУ";
        } else if (isChecked1 && !isChecked2 && isChecked3) {
            matchesFilter = ((*it)->getTypeOfControl() == "ЧПУ"
&&
            (*it)->getRepair() == "неисправный");
        } else if (!isChecked1 && isChecked2 && isChecked3) {
            matchesFilter = ((*it)->getTypeOfControl() ==
"ручное" &&
            (*it)->getRepair() == "неисправный");
        } else if (!isChecked1 && isChecked2 && !isChecked3) {
            matchesFilter = (*it)->getTypeOfControl() == "ручное";
        } else if (!isChecked1 && !isChecked2 && isChecked3) {
            matchesFilter = (*it)->getRepair() == "неисправный";
        }

        if (matchesFilter && currentIndex == machineNumber) {
            it = machineList.erase(it);
            ShowMessage(L"Станок удалён успешно.");

            FillListViewFromMachineList(listView, machineList,
isChecked1, isChecked2, isChecked3);
            return;
        }

        if (matchesFilter) {
            ++currentIndex;
        }
        ++it;
    }

    ShowMessage(L"Станок с таким номером не найден.");

```

```

}

//графика
void __fastcall TForm1::PaintBox1Paint(TObject *Sender)
{
    //лупа
    PaintBox1->Canvas->Pen->Color = clBlue;
    PaintBox1->Canvas->Pen->Width = 2;
    PaintBox1->Canvas->Brush->Style = bsClear;

    int left = 60, top = 53, right = 80, bottom = 73;

    PaintBox1->Canvas->Ellipse(left, top, right, bottom);
    PaintBox1->Canvas->MoveTo(64, 69);
    PaintBox1->Canvas->LineTo(55, 78);

    //редактирование
    PaintBox1->Canvas->Pen->Color = clBlack;

    left = 60, top = 340, right = 80, bottom = 360;

    PaintBox1->Canvas->Rectangle(left, top, right, bottom);
    PaintBox1->Canvas->MoveTo(70, 350);
    PaintBox1->Canvas->LineTo(85, 335);

    //Сортировка
    PaintBox1->Canvas->Pen->Color = clBlack;
    PaintBox1->Canvas->Pen->Width = 2;

    PaintBox1->Canvas->MoveTo(60, 382);
    PaintBox1->Canvas->LineTo(60, 407);
    PaintBox1->Canvas->MoveTo(60, 382);
    PaintBox1->Canvas->LineTo(53, 395);
    PaintBox1->Canvas->MoveTo(59, 382);
    PaintBox1->Canvas->LineTo(66, 395);

```

```

PaintBox1->Canvas->MoveTo(75, 382);
PaintBox1->Canvas->LineTo(75, 407);
PaintBox1->Canvas->MoveTo(75, 407);
PaintBox1->Canvas->LineTo(68, 397);
PaintBox1->Canvas->MoveTo(75, 407);
PaintBox1->Canvas->LineTo(82, 397);

//выход
left = 110, top = 655, right = 130, bottom = 685;
PaintBox1->Canvas->Rectangle(left, top, right, bottom);

PaintBox1->Canvas->MoveTo(120, 670);
PaintBox1->Canvas->LineTo(90, 670);
PaintBox1->Canvas->MoveTo(90, 670);
PaintBox1->Canvas->LineTo(100, 678);
PaintBox1->Canvas->MoveTo(90, 670);
PaintBox1->Canvas->LineTo(100, 662);

//памка
left = 80, top = 100, right = 270, bottom = 300;
PaintBox1->Canvas->Rectangle(left, top, right, bottom);

//сохранить
PaintBox1->Canvas->MoveTo(95, 635);
PaintBox1->Canvas->LineTo(130, 635);
PaintBox1->Canvas->MoveTo(95, 635);
PaintBox1->Canvas->LineTo(100, 625);
PaintBox1->Canvas->MoveTo(125, 625);
PaintBox1->Canvas->LineTo(130, 635);

PaintBox1->Canvas->MoveTo(113, 625);
PaintBox1->Canvas->LineTo(113, 600);
PaintBox1->Canvas->MoveTo(113, 625);
PaintBox1->Canvas->LineTo(105, 615);
PaintBox1->Canvas->MoveTo(113, 625);

```



```

PaintBox1->Canvas->LineTo(121, 615);

//добавить/удалить
PaintBox1->Canvas->Pen->Width = 4;
PaintBox1->Canvas->Pen->Color = clRed;

PaintBox1->Canvas->MoveTo(62, 485);
PaintBox1->Canvas->LineTo(77, 503);
PaintBox1->Canvas->MoveTo(77, 485);
PaintBox1->Canvas->LineTo(62, 503);

PaintBox1->Canvas->Pen->Color = clLime;

PaintBox1->Canvas->MoveTo(70, 435);
PaintBox1->Canvas->LineTo(70, 455);
PaintBox1->Canvas->MoveTo(60, 445);
PaintBox1->Canvas->LineTo(80, 445);
}

//основа
__fastcall TForm1::TForm1(TComponent* Owner) : TForm(Owner) {
    isChecked1 = CheckBox1->Checked;
    isChecked2 = CheckBox2->Checked;
    isChecked3 = CheckBox3->Checked;
    FillMachineList(machineList,
"C:\\Users\\user\\Desktop\\пгора\\кypcay\\machines.txt");
    FillListViewFromMachineList(ListView1, machineList,
isChecked1, isChecked2, isChecked3);
    PaintBox1->Repaint();
}

//редактирование
void __fastcall TForm1::Button5Click(TObject *Sender)
{

```

```

if ((Edit1->Text.Trim()).IsEmpty()) {
    ShowMessage(L"Введите строку для редактирования!");
    return;
}

int rowNumber = StrToInt(Edit1->Text.Trim());

if (rowNumber < 1 || rowNumber > machineList.size()) {
    ShowMessage(L"Неверный номер строки.");
    return;
}

TForm3 *form3 = new TForm3(this);
auto it = machineList.begin();
std::advance(it, rowNumber - 1);

form3->LoadDataFromMachineList(machineList, rowNumber);

if (form3->ShowModal() == mrOk) {
    UnicodeString materialType;

    switch (form3->isMaterial) {
        case 1: materialType = "металл"; break;
        case 2: materialType = "дерево"; break;
        case 3: materialType = "универсал"; break;
        default: materialType = "неизвестно"; break;
    }

    if (form3->isCNC) {
        *it = std::make_unique<CNC>(form3->isName, "ЧПУ",
materialType,
form3->isRepair ?
"неисправный" : "исправный",
form3->isYear,
form3->isInfo);
    }
}

```

```

        } else {
            *it = std::make_unique<Manual>(form3->isName,
"ручное", materialType,
form3-
>isRepair ? "неисправный" : "исправный",
form3->isYear,
form3->isInfo);
        }

        FillListViewFromMachineList(ListView1, machineList,
isChecked1, isChecked2, isChecked3);
    }
    delete form3;
}

//добавление
void __fastcall TForm1::Button4Click(TObject *Sender)
{
    TForm3 *form3 = new TForm3(this);

    if (form3->ShowModal() == mrOk) {

        UnicodeString materialType;

        switch (form3->isMaterial) {
            case 1: materialType = "металл"; break;
            case 2: materialType = "дерево"; break;
            case 3: materialType = "универсал"; break;
            default: materialType = "неизвестно"; break;
        }

        if (form3->isCNC) {
            machineList.push_back(std::make_unique<CNC>(form3-
>isName, "ЧПУ", materialType,

```

```

        form3->isRepair ? "неисправный" : "исправный",
form3->isYear, form3->isInfo));
        } else {

        machineList.push_back(std::make_unique<Manual>(form3->isName,
"ручное", materialType,
        form3->isRepair ? "неисправный" : "исправный",
form3->isYear, form3->isInfo));
        }

        FillListViewFromMachineList(ListView1, machineList,
isChecked1, isChecked2, isChecked3);
    }

    delete form3;
}

//сортировка
void __fastcall TForm1::Button9Click(TObject *Sender) {
    TForm4 *form4 = new TForm4(this);

    if (form4->ShowModal() == mrOk) {
        int selected = form4->ChoiseSortMetod;
        bool Reverse = form4->Reverse;
        if (selected == 1) {
            SortMachineListByName(machineList, Reverse);
        } else if(selected == 2){
            SortMachineListByYear(machineList, Reverse);
        }
    }

    delete form4;

    FillListViewFromMachineList(ListView1, machineList,
isChecked1, isChecked2, isChecked3);
}

```

```

//выход
void __fastcall TForm1::Button10Click(TObject *Sender) {
    Form1->Close();
}

//удаление
void __fastcall TForm1::Button2Click(TObject *Sender) {
    RemoveMachineByNumber(ListView1, isChecked1, isChecked2,
isChecked3);
}

//поиск
void __fastcall TForm1::Button6Click(TObject *Sender) {
    SearchFromMachineList(ListView1, machineList, isChecked1,
isChecked2, isChecked3);
}

//вывод с условиями
void __fastcall TForm1::Button1Click(TObject *Sender){
    isChecked1 = CheckBox1->Checked;
    isChecked2 = CheckBox2->Checked;
    isChecked3 = CheckBox3->Checked;
    FillListViewFromMachineList(ListView1, machineList,
isChecked1, isChecked2, isChecked3);
}

//сохранение в файл
void __fastcall TForm1::Button3Click(TObject *Sender) {
    SaveListViewToFile(ListView1);
}

```

Unit3.h:

```

#ifndef Unit3H
#define Unit3H

```

```

#include <System.Classes.hpp>
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
#include <string>
#include "Unit1.h"

class Machine;
class CNC;
class Manual;

class TForm3 : public TForm
{
__published:    // IDE-managed Components
    TLabel *Label1;
    TEdit *Edit1;
    TLabel *Label3;
    TEdit *Edit2;
    TCheckBox *CheckBox1;
    TLabel *Label5;
    TEdit *Edit4;
    TCheckBox *CheckBox2;
    TButton *Button1;
    TButton *Button2;
    TLabel *Label6;
    TLabel *Label2;
    TRadioButton *RadioButton1;
    TRadioButton *RadioButton2;
    TRadioButton *RadioButton3;
    void __fastcall Button2Click(TObject *Sender);
    void __fastcall Button1Click(TObject *Sender);
private:    // User declarations
public:        // User declarations
    __fastcall TForm3(TComponent* Owner);

```

```

        int isMaterial, isYear;
        bool isCNC, isRepair;
        UnicodeString isName, isInfo;

        void __fastcall LoadDataFromMachineList(const
std::list<std::unique_ptr<Machine>>& machineList,
int rowNumber);
};

extern PACKAGE TForm3 *Form3;

#endif

```

Unti3.cpp:

```

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "Unit3.h"

#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;

__fastcall TForm3::TForm3(TComponent* Owner): TForm(Owner)
{

}

void __fastcall TForm3::Button2Click(TObject *Sender)
{
    ModalResult = mrOk;
}

```

```

void __fastcall TForm3::Button1Click(TObject *Sender)
{
    isName = Edit1->Text.Trim();
    UnicodeString YearOfRelease = Edit2->Text.Trim();
    isInfo = Edit4->Text.Trim();

    isCNC = CheckBox1->Checked;
    isRepair = CheckBox2->Checked;

    if (!TryStrToInt(YearOfRelease, isYear) || isYear <= 0) {
        ShowMessage(L"Введите корректный год!");
        return;
    }

    if (RadioButton1->Checked) {
        isMaterial = 1;
    } else if (RadioButton2->Checked){
        isMaterial = 2;
    } else if (RadioButton3->Checked){
        isMaterial = 3;
    }

    ModalResult = mrOk;
}

void __fastcall TForm3::LoadDataFromMachineList(const
std::list<std::unique_ptr<Machine>>& machineList,
int rowNumber) {
    if (rowNumber <= 0 || rowNumber > machineList.size()) {
        ShowMessage(L"Неверный номер строки!");
        return;
    }

    auto it = machineList.begin();
    std::advance(it, rowNumber - 1);
}

```



```

Machine* machine = it->get();

Edit1->Text = machine->getName();
Edit2->Text = UnicodeString(machine->getYearOfRelease());

if (machine->getTypeOfControl() == "ЧПУ") {
    const CNC* cncMachine = dynamic_cast<const
CNC*>(machine);
    CheckBox1->Checked = true;
    Edit4->Text = cncMachine->getSoftware();
} else {
    const Manual* manualMachine = dynamic_cast<const
Manual*>(machine);
    CheckBox1->Checked = false;
    Edit4->Text = manualMachine->getFCs();
}

if (machine->getRepair() == "неисправный") {
    CheckBox2->Checked = true;
} else {
    CheckBox2->Checked = false;
}

if (machine->getMaterial() == "металл") {
    RadioButton1->Checked = true;
} else if (machine->getMaterial() == "дерево") {
    RadioButton2->Checked = true;
} else if (machine->getMaterial() == "универсал") {
    RadioButton3->Checked = true;
}
}

```

Unit4.h:

```
#ifndef Unit4H
```

```

#define Unit4H

#include <System.Classes.hpp>
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>

class TForm4 : public TForm
{
__published:    // IDE-managed Components
    TButton *Button1;
    TButton *Button2;
    TRadioButton *RadioButton1;
    TRadioButton *RadioButton2;
    TCheckBox *CheckBox1;
    void __fastcall Button1Click(TObject *Sender);
    void __fastcall Button2Click(TObject *Sender);
private:    // User declarations
public:     // User declarations
    __fastcall TForm4(TComponent* Owner);
    int ChoiseSortMetod;
    bool Reverse;
};

extern PACKAGE TForm4 *Form4;
#endif

```

Unit4.cpp:

```

#include <vcl.h>
#pragma hdrstop

#include "Unit4.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm4 *Form4;

```

```

__fastcall TForm4::TForm4(TComponent* Owner)
    : TForm(Owner)
{
}

void __fastcall TForm4::Button1Click(TObject *Sender)
{
    Reverse = CheckBox1->Checked;
    if (RadioButton1->Checked) {
        ChoiseSortMetod = 1;
    } else if (RadioButton2->Checked){
        ChoiseSortMetod = 2;
    }
    ModalResult = mrOk;
}

void __fastcall TForm4::Button2Click(TObject *Sender)
{
    ModalResult = mrOk;
}

```