МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ

КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

Курсовая работа

**«Модель работы принтера при его взаимодействии с компьютером»**

по дисциплине

**«Технологии программирования»**

Выполнил:

студент гр. БЭИ2201 Зудаев Б. Д.

Проверил:

доц. каф. «Информатика» Волков А. И.,

ст. преп. Загвоздкина А. В.

Москва 2024

Содержание

[Введение 3](#_Toc167866966)

[Задание 4](#_Toc167866966)

[Логическая структура программы 5](#_Toc167866967)

[Схемы алгоритмов 8](#_Toc167866968)

[1.1 Пространство имен SimulationObjects 8](#_Toc167866969)

[1.1.1 Printer 8](#_Toc167866970)

[1.1.2 InkjetPrinter 13](#_Toc167866971)

[1.1.3 Computer 17](#_Toc167866972)

[1.1.4 Internet 22](#_Toc167866973)

[1.1.5 Document 27](#_Toc167866974)

[1.1.6 Human 30](#_Toc167866975)

[1.2 Пространство имен Simulation 32](#_Toc167866976)

[1.2.1 DelayedImageChanger 32](#_Toc167866977)

[1.2.2 DelayedVisibilityChanger 34](#_Toc167866978)

[1.2.3 PrinterSimulation 36](#_Toc167866979)

[Текст программы 56](#_Toc167866980)

[2.1 Пространство имен SimulationObjects 56](#_Toc167866981)

[2.1.1 Document 56](#_Toc167866982)

[2.1.2 Internet 58](#_Toc167866983)

[2.1.3 Computer 60](#_Toc167866984)

[2.1.4 Printer 63](#_Toc167866985)

[2.1.5 InkjetPrinter 66](#_Toc167866986)

[2.1.6 Human 68](#_Toc167866987)

[2.2 Пространство имен Simulation 69](#_Toc167866988)

[2.2.1 DelayedImageChanger 69](#_Toc167866989)

[2.2.2 DelayedVisibilityChancher 70](#_Toc167866990)

[2.2.3 PrinterSimulation 71](#_Toc167866991)

[Результаты тестирования программы 9](#_Toc167866992)2

[Вывод 9](#_Toc167866993)7

[Список использованной литературы 98](#_Toc167866994)

ВВЕДЕНИЕ

Объектно-ориентированное программирование (ООП) — это парадигма программирования, основанная на концепции "объектов", которые могут содержать данные и код: данные представляют собой атрибуты (или свойства) объекта, а код — методы, которые описывают поведение объекта. ООП стало популярным подходом в разработке программного обеспечения благодаря своей способности моделировать сложные системы и упрощать процесс разработки, тестирования и сопровождения программ.

Ключевые принципы ООП включают инкапсуляцию, наследование и полиморфизм. Инкапсуляция объединяет данные и методы их обработки в один объект, скрывая внутренние детали и предоставляя только необходимый интерфейс. Наследование позволяет создавать новые классы на основе существующих, унаследуя их свойства и методы, а также расширяя их функциональность. Полиморфизм даёт возможность объектам одного типа использовать одинаковые методы, но с разной реализацией, что увеличивает гибкость и удобство программирования.

ООП находит широкое применение в различных областях разработки программного обеспечения, включая создание веб-приложений, мобильных приложений, игр и систем управления базами данных. Языки программирования, такие как Java, C++, Python и C#, предоставляют мощные инструменты для реализации объектно-ориентированных решений.

В данной работе будет разработано Windows Forms приложение, активно использующее объектно-ориентированное программирование. В процессе создания будут рассмотрены и применены все основные аспекты ООП для построения гибкой и эффективной архитектуры. Особое внимание будет уделено реализации событийного взаимодействия между объектами, что обеспечит интерактивность внутри приложения.

ЗАДАНИЕ

Разработать приложение для моделирования работы принтера при его взаимодействии с компьютером. В этом приложении:

1. Создать класс, содержащий данные о некотором объекте (базовом) из заданной предметной области. Эти данные должны быть спрятаны от непосредственного доступа (инкапсулированы). Членами класса должны быть не менее 4-х свойств, 2-х методов, 1-го конструктора и 2-х событий. Дополнительно создать классы, описывающие вспомогательные объекты, необходимые для моделирования;

В главной программе создать экземпляры созданных классов и продемонстрировать возможности изменения свойств объектов, использования их методов и обработки реализованных событий (разработав соответствующие методы обработки событий, объявленных в классах). При реализации событий необходимо обратить особое внимание на организацию событийного взаимодействия объектов;

1. Создать еще один класс путем наследования имеющегося класса. В новом классе переопределить хотя бы один из методов базового класса

В главной программе создать экземпляры базового и дочернего классов (а также вспомогательных классов) и продемонстрировать их возможности по изменению свойств, использованию методов и обработки событий. Организовать взаимодействие созданных объектов между собой или с другими вспомогательными объектами с использованием пользовательских событий;

Реализовать валидацию ввода, а также запретить взаимодействие с программой, приводящее к непредвиденному состоянию.

ЛОГИЧЕСКАЯ СТРУКТУРА ПРОГРАММЫ

Программа делится на две части: SimulationObjects — проект с основными классами; Simulation — основной проект с поддержкой Windows Forms, в котором также реализованы вспомогательные классы.

Основными классами являются принтер (Printer) и компьютер (Computer). Класс, наследующийся от принтера — струйный принтер (InkjetPrinter).

Printer обладает следующими свойствами:

* PaperTrayCount — количество бумаги в лотке для бумаги;
* PaperTrayCapacity — объём лотка для бумаги;
* CartridgeCapacity — объём картриджа.

Методами:

* Print — печатает страницу документа;
* GetCartridgeFillLevel — получает текущее количество краски в картридже;
* FillCartridge — заправляет картридж.

Событиями:

* Printing — вызывается при печати;
* StopPrinting — вызывается при окончании печати;
* CartridgeEmpty — вызывается при опустошении картриджа;
* PaperTrayEmpty — вызывается при опустошении лотка для бумаги.

InkjetPrinter обладает всеми свойствами, методами и событиями, которыми обладает Printer, так как наследуется от него, а также полями \_cartridgeFillLevels, хранящее уровни заполненности картриджей, и \_cartridgeColors, хранящее цвета картриджей, и переопределёнными методами Print для печати цветных документов, GetCartridgeLevel для получения уровня заполненности определённого картриджа и FillCartridge для заправки определённого картриджа.

Computer обладает следующими свойствами:

* AdapterIsEnabled — включен ли интернет.

Методами:

* RequestDocument — запрашивает документ.

Вспомогательными методами:

* ChangeTimerInterval — устанавливает интервал таймеров;
* SendDocumentToPrinter — вызывает события DocumentSentToInkjetPrinter и DocumentSentToPrinter;
* SubscribeToEvents — подписывает компьютер на события принтера;
* UnsubscribeFromEvents — отписывает компьютер от событий принтера;
* OnPaperTrayEmpty — обработчик события PaperTrayEmpty;
* OnCartridgeEmpty — обработчик события CartridgeEmpty.

Событиями:

* DocumentRequested — вызывается при запросе документа;
* DocumentSentToInkjetPrinter — вызывается при отправке документа на струйный принтер;
* DocumentSentToPrinter — вызывается при отправке документа на принтер;
* CartridgeEmpty — вызывается при опустошении картриджа;
* PaperTrayEmpty — вызывается при опустошении лотка для бумаги.

Internet обладает следующими методами:

* GenerateDocument — создает документ;
* TrySendDocument — при возможности отправляет документ;
* GetCurrentDocumentCount — возвращает текущее количество документов;
* RequestDocument — запрашивает документ.

Вспомогательными методами:

* StartGenerating — запускает создание документов;
* StopGenerating — останавливает создание документов;
* ChangeTimerInterval — устанавливает интервал таймера.

Событиями:

* DocumentSent — вызывается при отправке документа.

Document обладает следующими свойствами:

* PageCount — количество страниц.

Методами:

* IsColor — возвращает, цветной ли документ;
* GetPageColor — возвращает цвет страницы.

Human обладает следующими методами:

* RefillCartridge — заправляет картридж;
* RefillPaperTray — заполняет лоток для бумаги.

Событиями:

* CartridgeRefilled — вызывается при заправке картриджа;
* PaperTrayRefilled — вызывается при заполнении лотка для бумаги.

Также для визуализации программы реализованы дополнительные вспомогательные объекты, такие как DelayedImageChanger и DelayedVisibilityChanger, которые устанавливают интервал показа изображения.

СХЕМЫ АЛГОРИТМОВ

1.1 Пространство имен SimulationObjects

1.1.1 Printer

Схема алгоритма для метода CartridgeCapacity::get класса Printer представлена на рисунке 1.

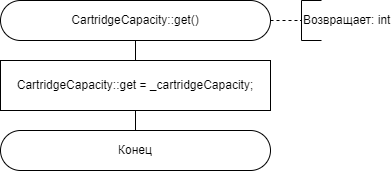


Рисунок 1 — Схема алгоритма для метода CartridgeCapacity::get класса Printer

Схема алгоритма для метода FillCartridge класса Printer представлена на рисунке 2.

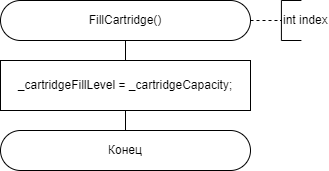


Рисунок 2 — Схема алгоритма для метода FillCartridge класса Printer

Схема алгоритма для метода GetCartridgeFillLevel класса Printer представлена на рисунке 3.

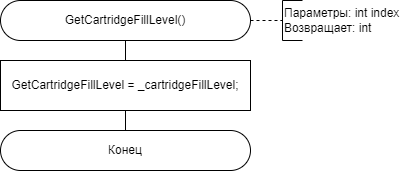


Рисунок 3 — Схема алгоритма для метода GetCartridgeFillLevel класса Printer

Схема алгоритма для метода PaperTrayCapacity::get класса Printer представлена на рисунке 4.

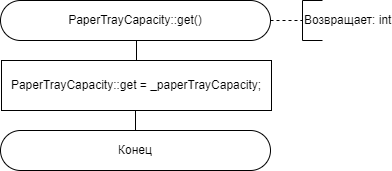


Рисунок 4 — Схема алгоритма для метода PaperTrayCapacity::get класса Printer

Схема алгоритма для метода PaperTrayCount::get класса Printer представлена на рисунке 5.

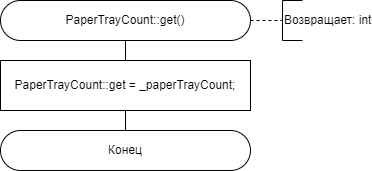


Рисунок 5 — Схема алгоритма для метода PaperTrayCount::get класса Printer

Схема алгоритма для метода PaperTrayCount::set класса Printer представлена на рисунке 6.

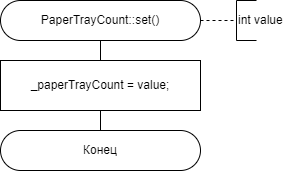


Рисунок 6 — Схема алгоритма для метода PaperTrayCount::set класса Printer

Схема алгоритма для метода Print класса Printer представлена на рисунке 7.

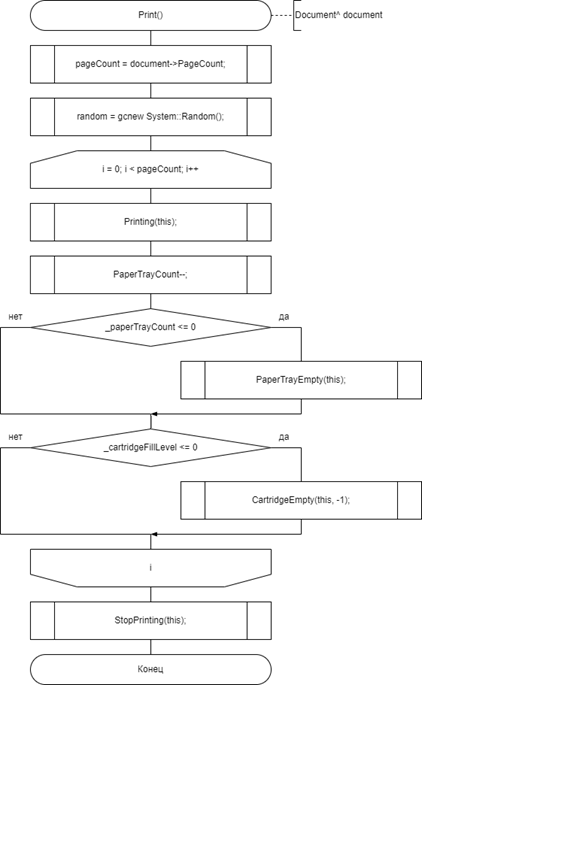


Рисунок 7 — Схема алгоритма для метода Print класса Printer

Схема алгоритма для метода Printer класса Printer представлена на рисунке 8.

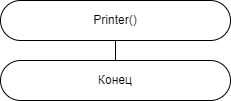


Рисунок 8 — Схема алгоритма для метода Printer класса Printer

Схема алгоритма для метода Printer класса Printer представлена на рисунке 9.

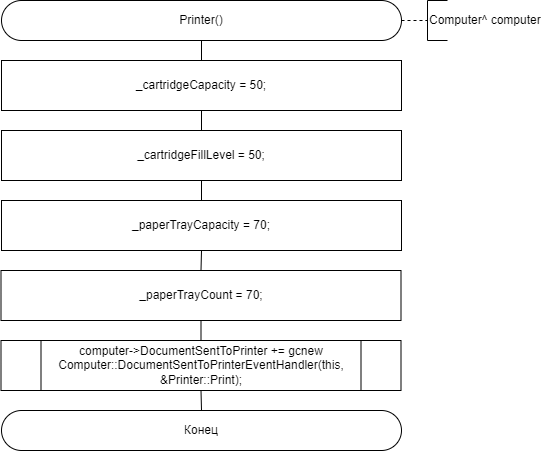


Рисунок 9 — Схема алгоритма для метода Printer класса Printer

1.1.2 inkjetPrinter

Схема алгоритма для метода FillCartridge класса InkjetPrinter представлена на рисунке 10.

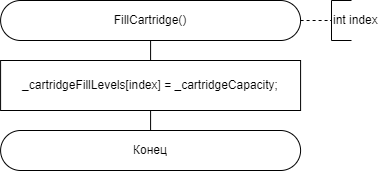


Рисунок 10 — Схема алгоритма для метода FillCartridge класса InkjetPrinter

Схема алгоритма для метода GetCartridgeFillLevel класса InkjetPrinter представлена на рисунке 11.

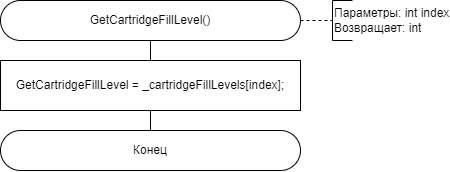


Рисунок 11 — Схема алгоритма для метода GetCartridgeFillLevel класса InkjetPrinter

Схема алгоритма для метода InkjetPrinter класса InkjetPrinter представлена на рисунке 12.

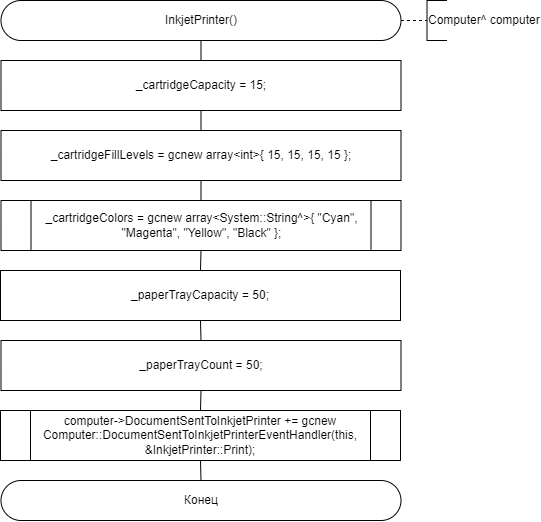


Рисунок 12 — Схема алгоритма для метода InkjetPrinter класса InkjetPrinter

Схема алгоритма для метода Print класса InkjetPrinter представлена на рисунках 13-14.

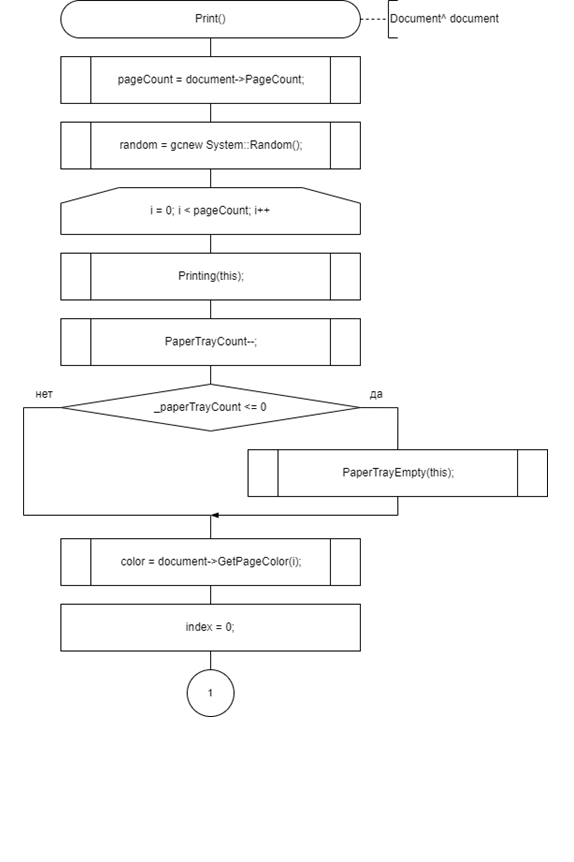


Рисунок 13 — Схема алгоритма для метода Print класса InkjetPrinter

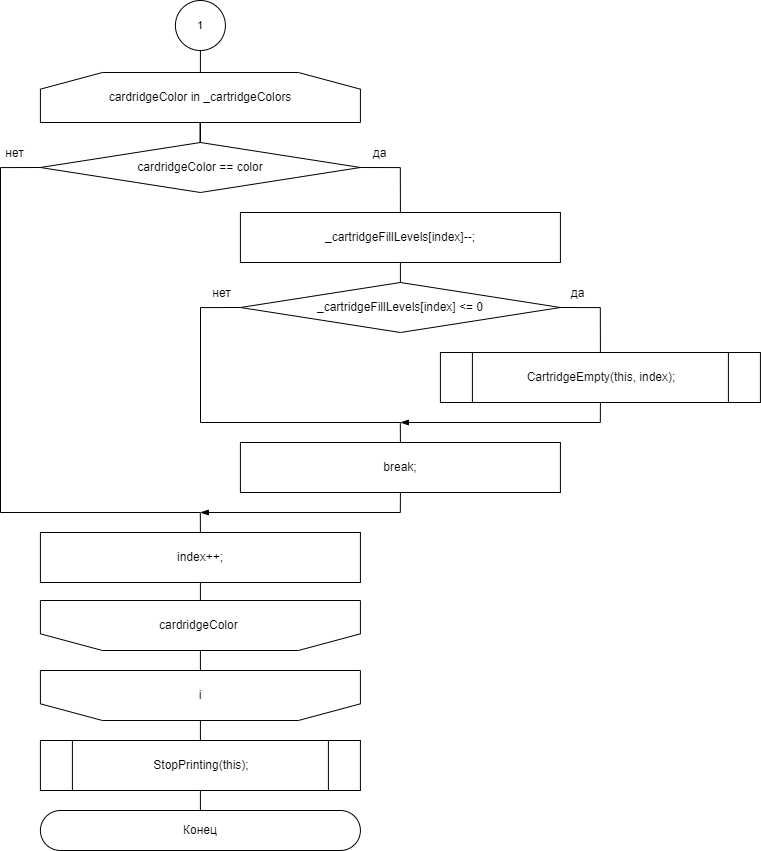


Рисунок 14 — Схема алгоритма для метода Print класса InkjetPrinter

1.1.3 Computer

Схема алгоритма для метода AdapterInEnabled::get класса Computer представлена на рисунке 15.

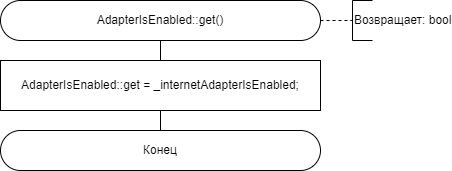


Рисунок 15 — Схема алгоритма для метода AdapterInEnabled::get класса Computer

Схема алгоритма для метода AdapterIsEnabled::set класса Computer представлена на рисунке 16.

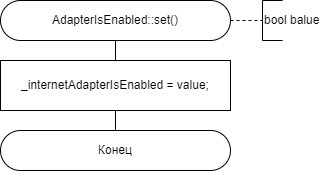


Рисунок 16 — Схема алгоритма для метода AdapterIsEnabled::set класса Computer

Схема алгоритма для метода ChangeTimerInterval класса Computer представлена на рисунке 17.

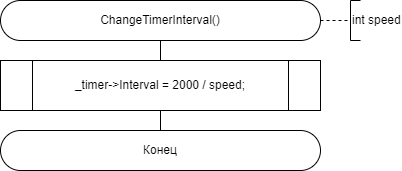


Рисунок 17 — Схема алгоритма для метода ChangeTimerInterval класса Computer

Схема алгоритма для метода Computer класса Computer представлена на рисунке 18.

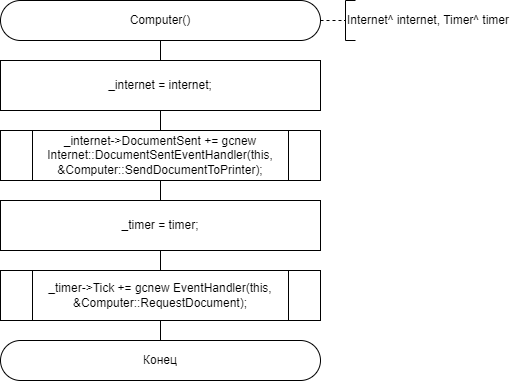


Рисунок 18 — Схема алгоритма для метода Computer класса Computer

Схема алгоритма для метода OnCartridgeEmpty класса Computer представлена на рисунке 19.

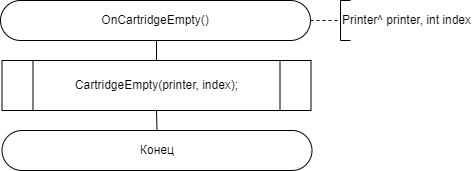


Рисунок 19 — Схема алгоритма для метода OnCartridgeEmpty класса Computer

Схема алгоритма для метода OnPaperTrayEmpty класса Computer представлена на рисунке 20.

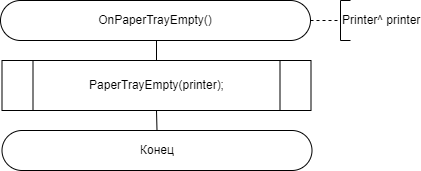


Рисунок 20 — Схема алгоритма для метода OnPaperTrayEmpty класса Computer

Схема алгоритма для метода RequestDocument класса Computer представлена на рисунке 21.

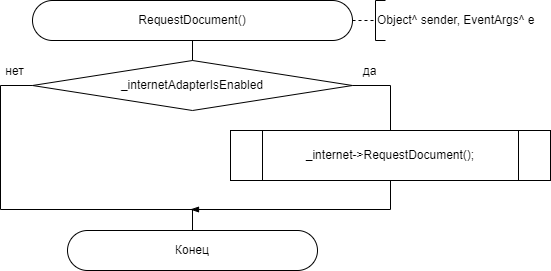


Рисунок 21 — Схема алгоритма для метода RequestDocument класса Computer

Схема алгоритма для метода SendDocumentToPrinter класса Computer представлена на рисунке 22.

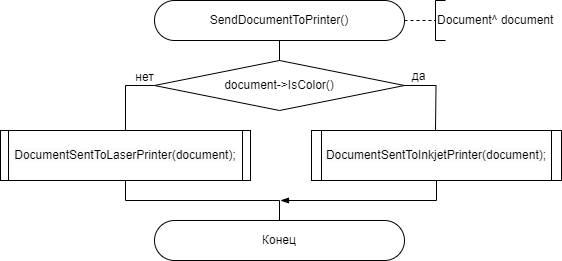


Рисунок 22 — Схема алгоритма для метода SendDocumentToPrinter класса Computer

Схема алгоритма для метода SubscribeToEvents класса Computer представлена на рисунке 23.

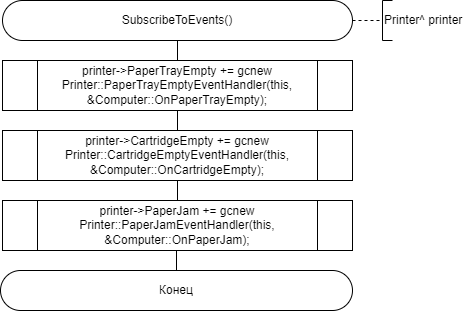


Рисунок 23 — Схема алгоритма для метода SubscribeToEvents класса Computer

Схема алгоритма для метода UnsubscribeFromEvents класса Computer представлена на рисунке 24.

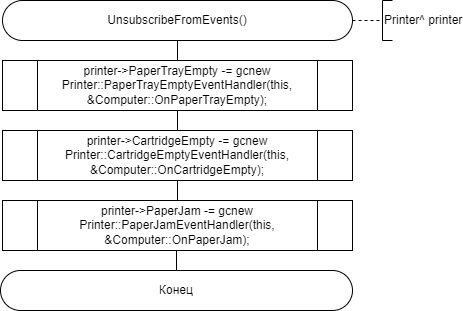


Рисунок 24 — Схема алгоритма для метода UnsubscribeFromEvents класса Computer

1.1.4 Internet

Схема алгоритма для метода ChangeTimerInterval класса Internet представлена на рисунке 25.

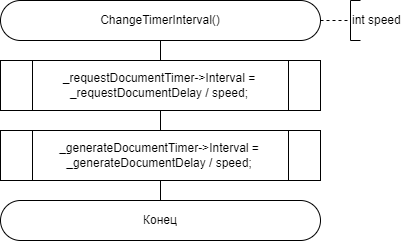


Рисунок 25 — Схема алгоритма для метода ChangeTimerInterval класса Internet

Схема алгоритма для метода GetCurrentDocumentCount класса Internet представлена на рисунке 26.

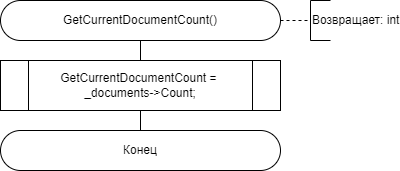


Рисунок 26 — Схема алгоритма для метода GetCurrentDocumentCount класса Internet

Схема алгоритма для метода GenerateDocument класса Internet представлена на рисунке 27.

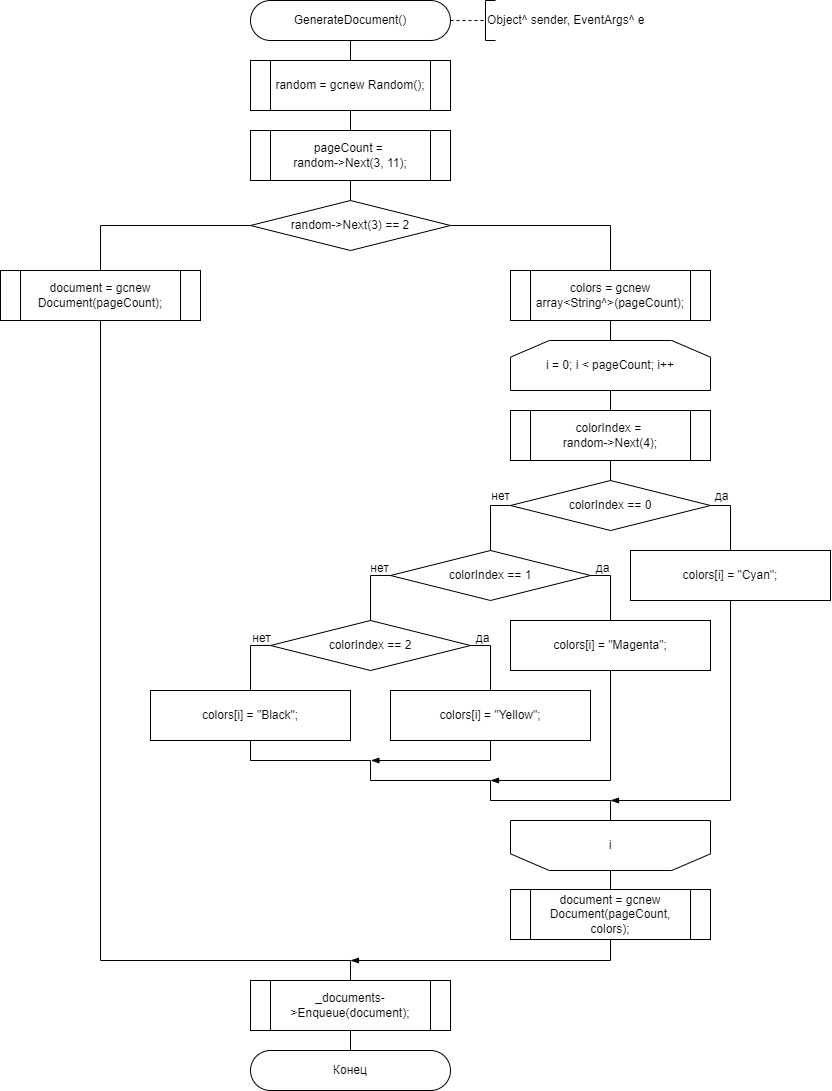


Рисунок 27 — Схема алгоритма для метода GenerateDocument класса Internet

Схема алгоритма для метода Internet класса Internet представлена на рисунке 28.

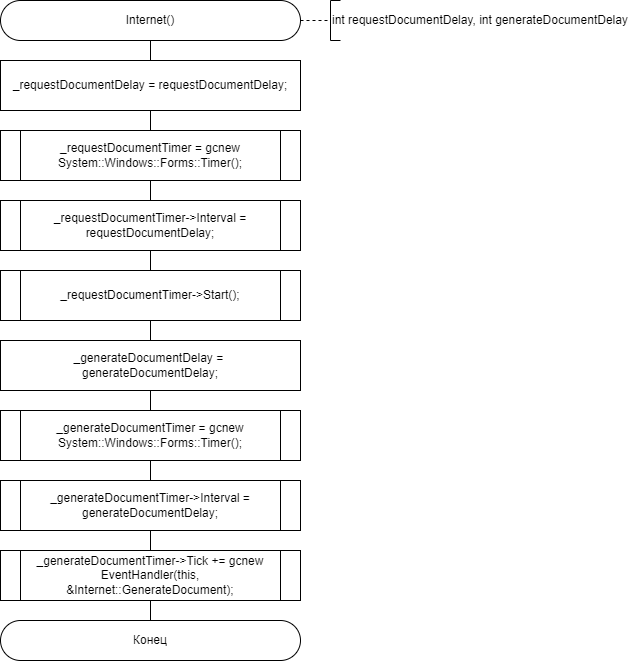


Рисунок 28 — Схема алгоритма для метода Internet класса Internet

Схема алгоритма для метода RequestDocument класса Internet представлена на рисунке 29.

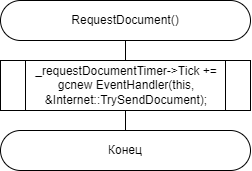


Рисунок 29 — Схема алгоритма для метода RequestDocument класса Internet

Схема алгоритма для метода StartGenerating класса Internet представлена на рисунке 30.

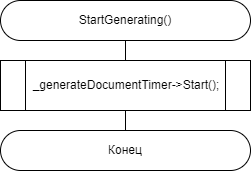


Рисунок 30 — Схема алгоритма для метода StartGenerating класса Internet

Схема алгоритма для метода StopGenerating класса Internet представлена на рисунке 31.

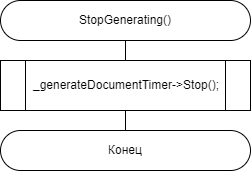


Рисунок 31 — Схема алгоритма для метода StopGenerating класса Internet

Схема алгоритма для метода TrySendDocument класса Internet представлена на рисунке 32.

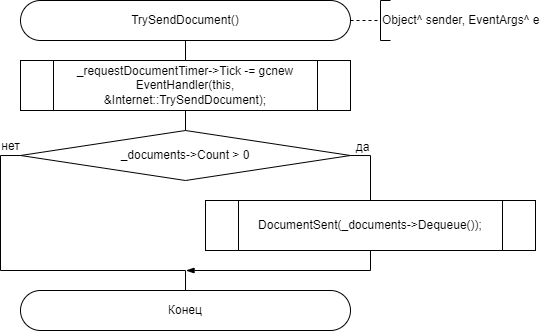


Рисунок 32 — Схема алгоритма для метода TrySendDocument класса Internet

1.1.5 Document

Схема алгоритма для метода Document класса Document представлена на рисунке 33.

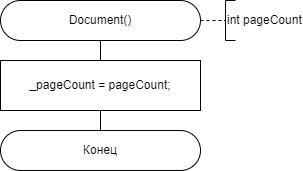


Рисунок 33 — Схема алгоритма для метода Document класса Document

Схема алгоритма для метода Document класса Document представлена на рисунке 34.

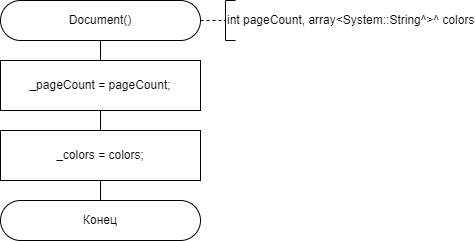


Рисунок 34 — Схема алгоритма для метода Document класса Document

Схема алгоритма для метода GetPageColor класса Document представлена на рисунке 35.

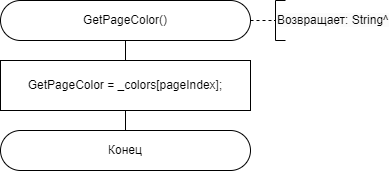


Рисунок 35 — Схема алгоритма для метода GetPageColor класса Document

Схема алгоритма для метода IsColor класса Document представлена на рисунке 36.

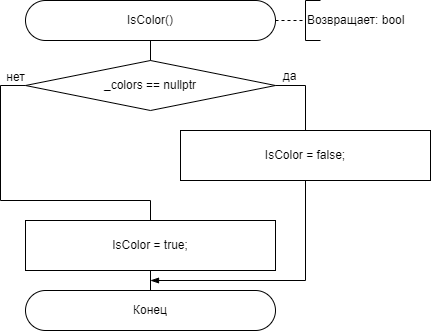


Рисунок 36— Схема алгоритма для метода IsColor класса Document

Схема алгоритма для метода PageCount::get класса Document представлена на рисунке 37.

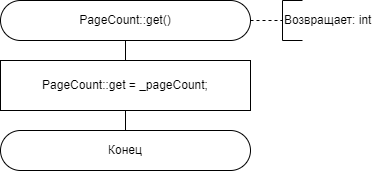


Рисунок 37 — Схема алгоритма для метода PageCount::get класса Document

Схема алгоритма для метода PageCount::set класса Document представлена на рисунке 38.

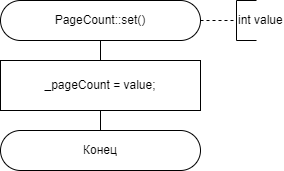


Рисунок 38 — Схема алгоритма для метода PageCount::set класса Document

1.1.6 Human

Схема алгоритма для метода Human класса Human представлена на рисунке 39.

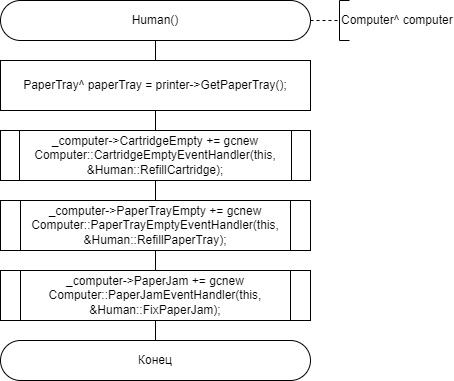


Рисунок 39 — Схема алгоритма для метода Human класса Human

Схема алгоритма для метода RefillCartridge класса Human представлена на рисунке 40.

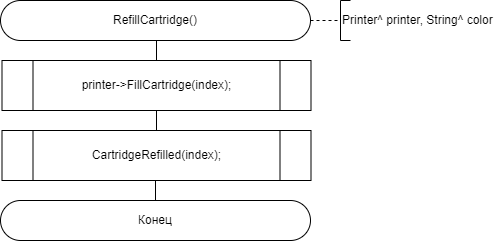


Рисунок 40 — Схема алгоритма для метода RefillCartridge класса Human

Схема алгоритма для метода RefillPaperTray класса Human представлена на рисунке 41.

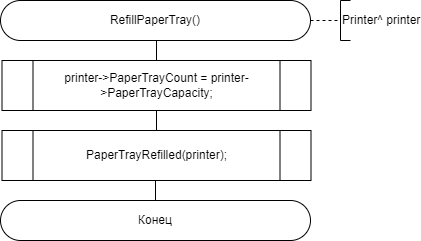


Рисунок 41 — Схема алгоритма для метода RefillPaperTray класса Human

1.2 Пространство имен Simulation

1.2.1 DelayedImageChanger

Схема алгоритма для метода класса OnTick представлена на рисунке 42.

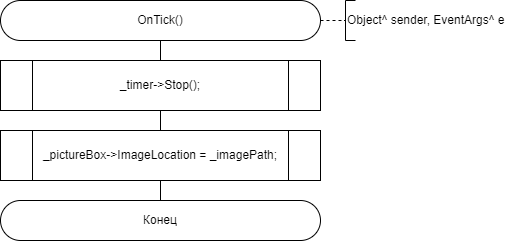


Рисунок 42 — Схема алгоритма для метода OnTick класса DelayedImageChanger

Схема алгоритма для метода DelayedImageChanger класса DelayedImageChanger представлена на рисунке 43.

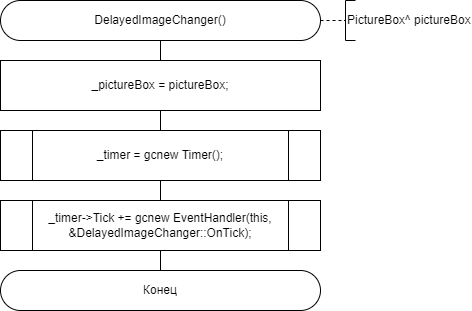


Рисунок 43 — Схема алгоритма для метода DelayedImageChanger класса DelayedImageChanger

Схема алгоритма для метода Change класса DelayedImageChanger представлена на рисунке 44.

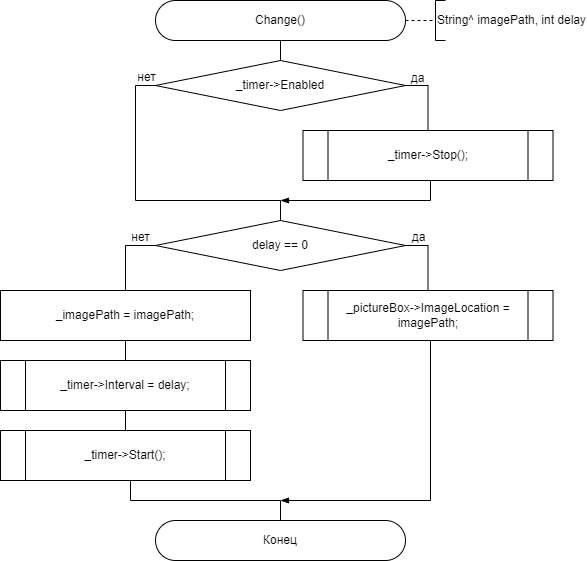


Рисунок 44 — Схема алгоритма для метода Change класса DelayedImageChanger

1.2.2 DelayedVisibilityChanger

Схема алгоритма для метода класса OnTick представлена на рисунке 45.

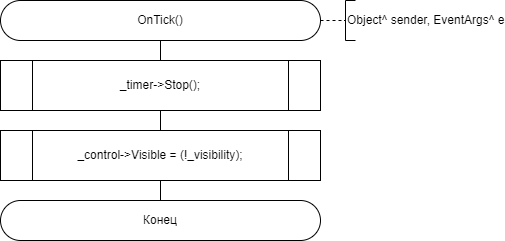


Рисунок 45 — Схема алгоритма для метода OnTick класса DelayedVisibilityChanger

Схема алгоритма для метода DelayedVisibilityChanger класса DelayedVisibilityChanger представлена на рисунке 46.

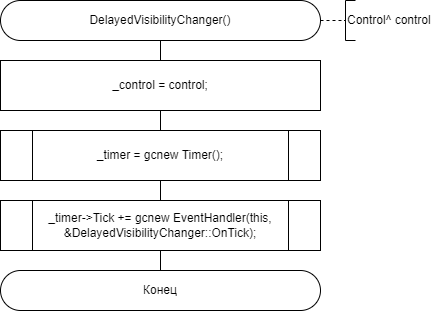


Рисунок 46 — Схема алгоритма для метода DelayedImageChanger класса DelayedImageChanger

Схема алгоритма для метода Change класса DelayedVisibilityChanger представлена на рисунке 47.

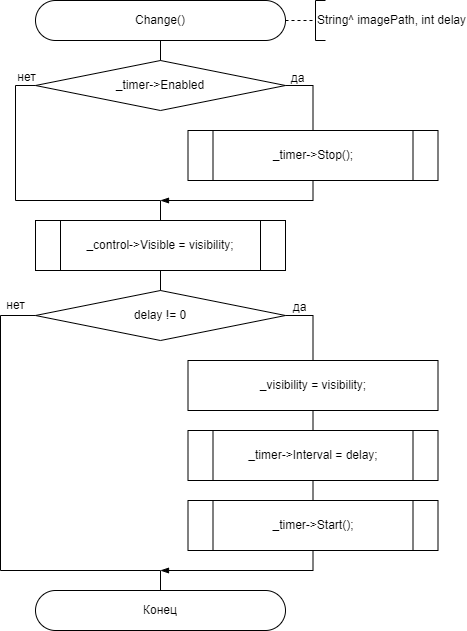


Рисунок 47 — Схема алгоритма для метода Change класса DelayedVisibilityChanger

1.2.3 PrinterSimulation

Схема алгоритма для метода класса UnsubscribeFromPrinterEvents представлена на рисунке 48.

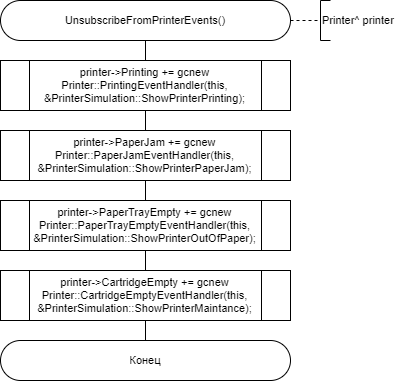


Рисунок 48 — Схема алгоритма для метода UnsubscribeFromPrinterEvents класса PrinterSimulation

Схема алгоритма для метода UnsubcribeFromHumanEvents класса PrinterSimulation представлена на рисунке 49.

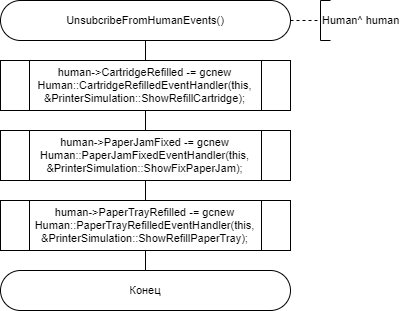


Рисунок 49 — Схема алгоритма для метода UnsubcribeFromHumanEvents класса PrinterSimulation

Схема алгоритма для метода timer\_Tick класса PrinterSimulation представлена на рисунке 50-52.

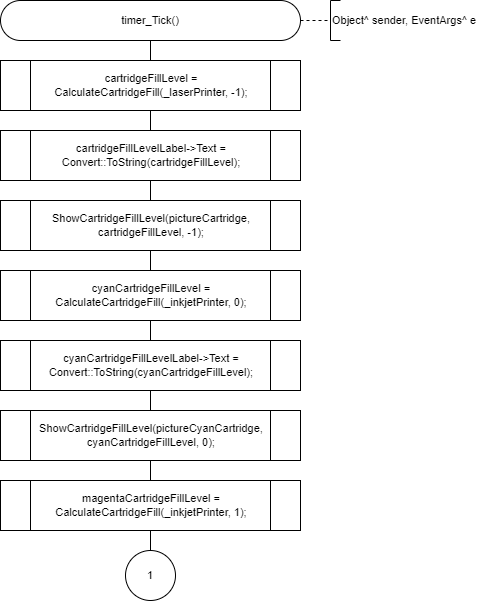


Рисунок 50 — Схема алгоритма для метода timer\_Tick класса PrinterSimulation

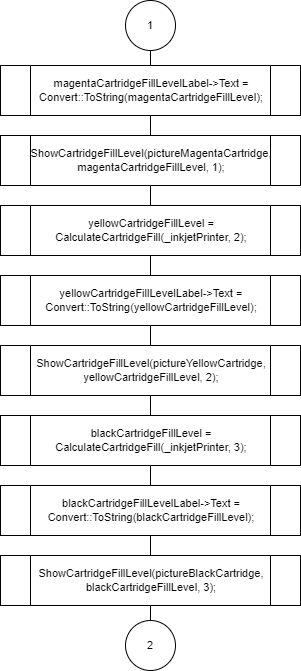


Рисунок 51 — Схема алгоритма для метода timer\_Tick класса PrinterSimulation

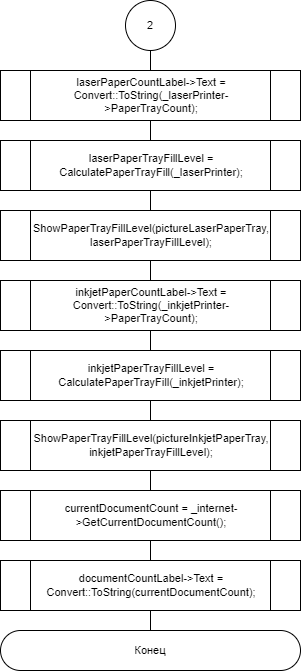


Рисунок 52 — Схема алгоритма для метода timer\_Tick класса PrinterSimulation

Схема алгоритма для метода класса SubscribeToPrinterEvents представлена на рисунке 53.

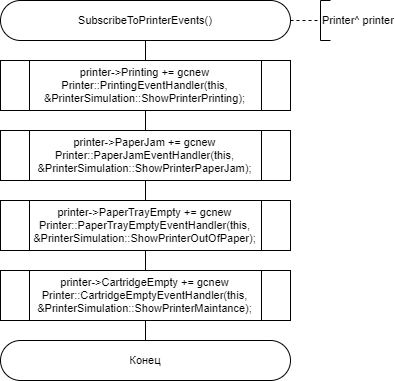


Рисунок 53 — Схема алгоритма для метода SubscribeToPrinterEvents класса PrinterSimulation

Схема алгоритма для метода SubscribeToHumanEvents класса PrinterSimulation представлена на рисунке 54.

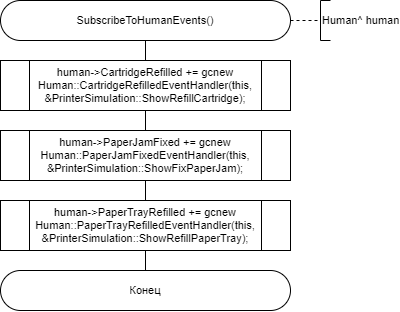


Рисунок 54 — Схема алгоритма для метода SubscribeToHumanEvents класса PrinterSimulation

Схема алгоритма для метода simulationSpeedTrackBar\_ValueChanged класса PrinterSimulation представлена на рисунке 55.

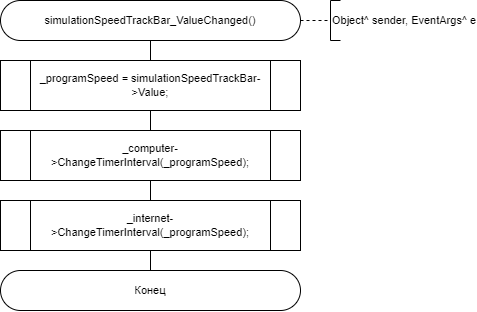


Рисунок 55 — Схема алгоритма для метода simulationSpeedTrackBar\_ValueChanged класса PrinterSimulation

Схема алгоритма для метода класса ShowRefillPaperTray представлена на рисунке 56.

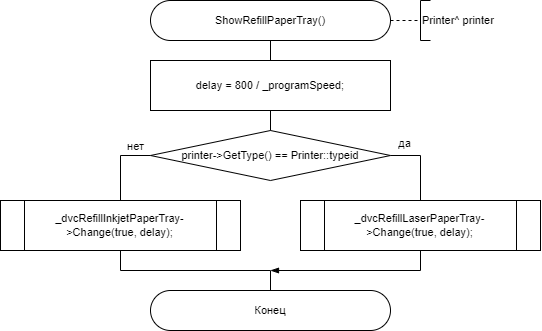


Рисунок 56 — Схема алгоритма для метода ShowRefillPaperTray класса PrinterSimulation

Схема алгоритма для метода класса ShowRefillCartridge представлена на рисунке 57.

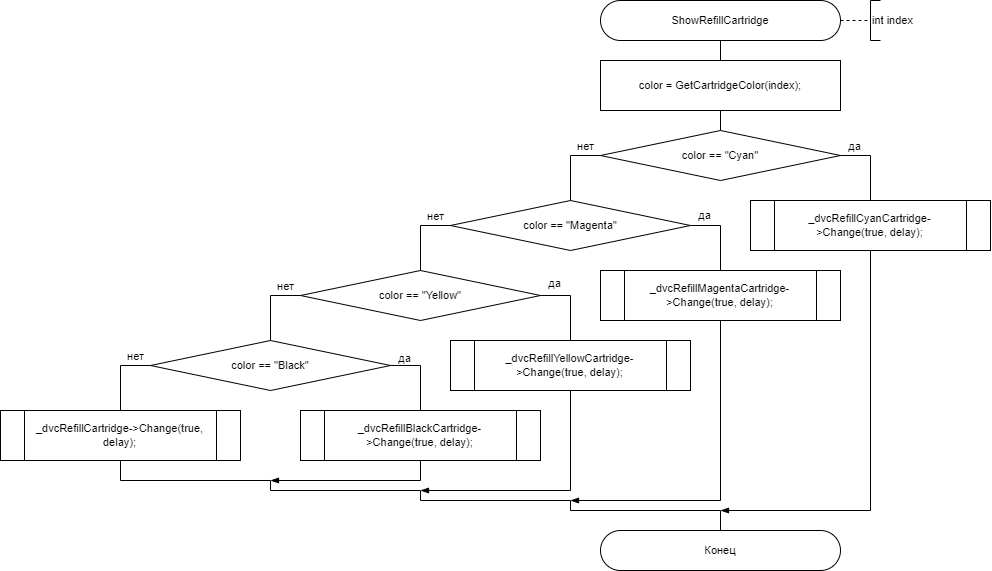


Рисунок 57 — Схема алгоритма для метода ShowRefillCartridge класса PrinterSimulation

Схема алгоритма для метода ShowPrinterPrinting класса PrinterSimulation представлена на рисунке 58.

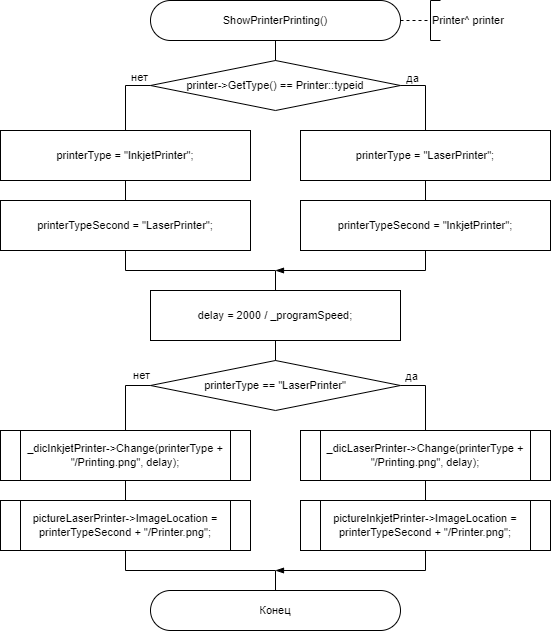


Рисунок 58 — Схема алгоритма для метода ShowPrinterPrinting класса PrinterSimulation

Схема алгоритма для метода ShowPrinterOutOfPaper класса PrinterSimulation представлена на рисунке 59.

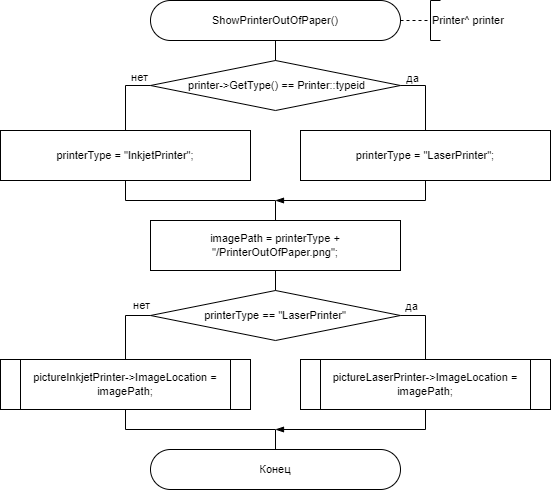


Рисунок 59 — Схема алгоритма для метода ShowPrinterOutOfPaper класса PrinterSimulation

Схема алгоритма для метода ShowPrinterMaintance класса PrinterSimulation представлена на рисунке 60.

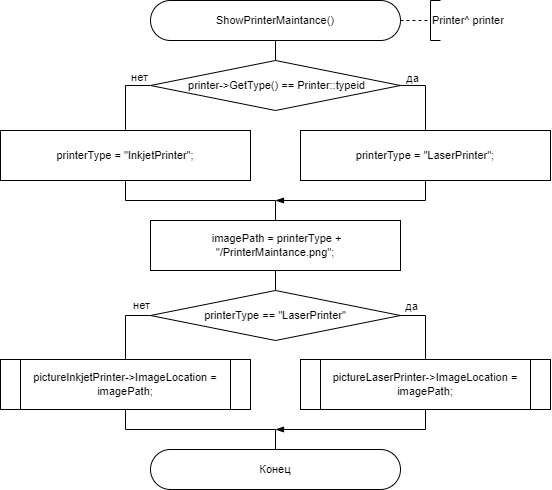


Рисунок 60 — Схема алгоритма для метода ShowPrinterMaintance класса PrinterSimulation

Схема алгоритма для метода ShowPaperTrayFillLevel класса PrinterSimulation представлена на рисунке 61.

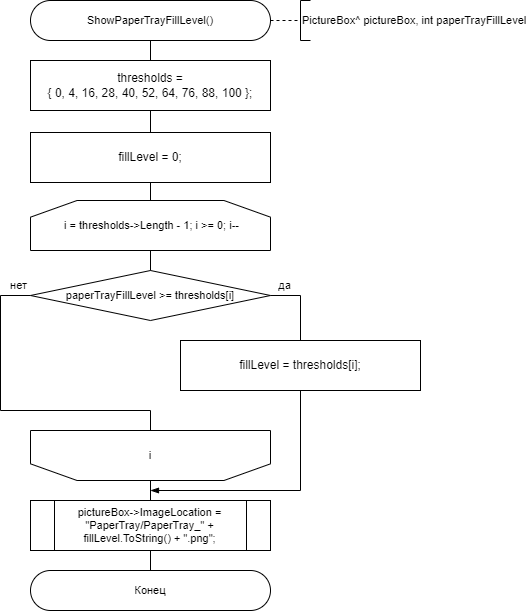


Рисунок 61 — Схема алгоритма для метода ShowPaperTrayFillLevel класса PrinterSimulation

Схема алгоритма для метода ShowCartridgeFillLevel класса PrinterSimulation представлена на рисунке 62.

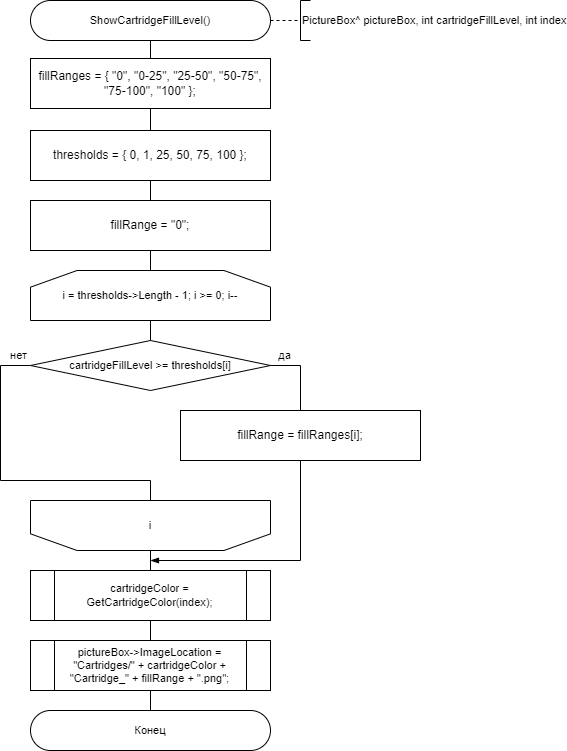


Рисунок 62 — Схема алгоритма для метода ShowCartridgeFillLevel класса PrinterSimulation

Схема алгоритма для метода PrinterSimulation класса PrinterSimulation представлена на рисунке 63-65.

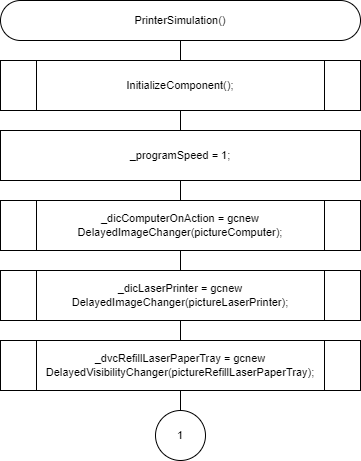


Рисунок 63 — Схема алгоритма для метода PrinterSimulation класса PrinterSimulation

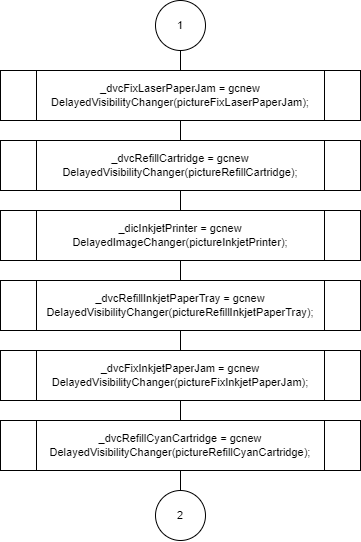


Рисунок 64 — Схема алгоритма для метода PrinterSimulation класса PrinterSimulation

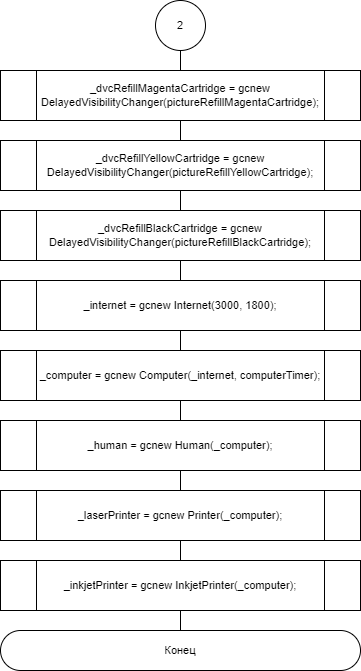


Рисунок 65 — Схема алгоритма для метода PrinterSimulation класса PrinterSimulation

Схема алгоритма для метода TurnOnDevices\_CheckedChanged класса PrinterSimulation представлена на рисунке 66.

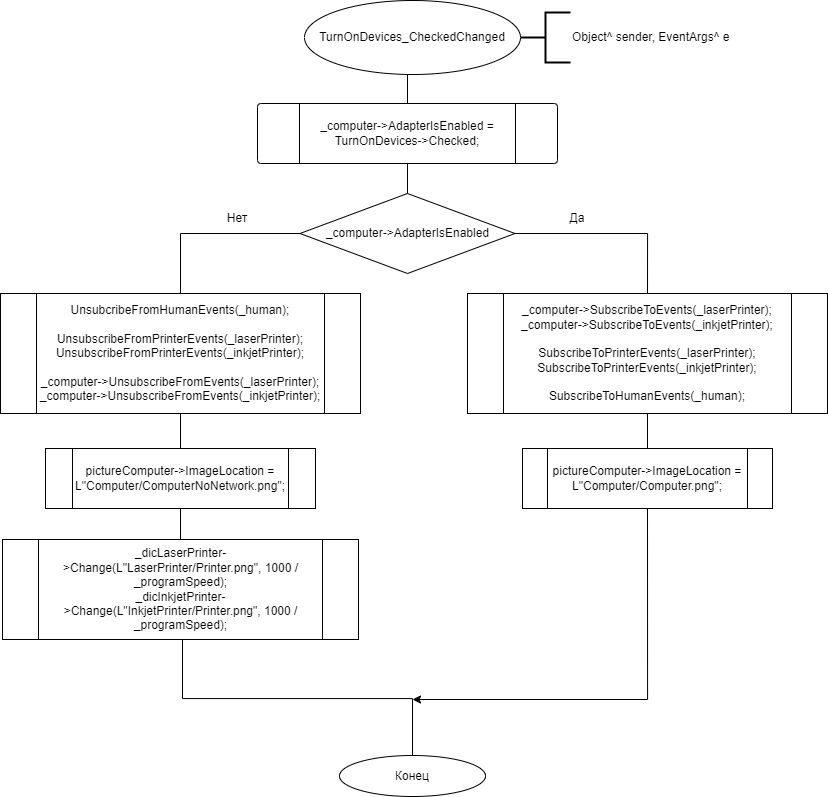


Рисунок 66 — Схема алгоритма для метода TurnOnDevices\_CheckedChanged класса PrinterSimulation

Схема алгоритма для метода GetCartridgeColor класса PrinterSimulation представлена на рисунке 67.

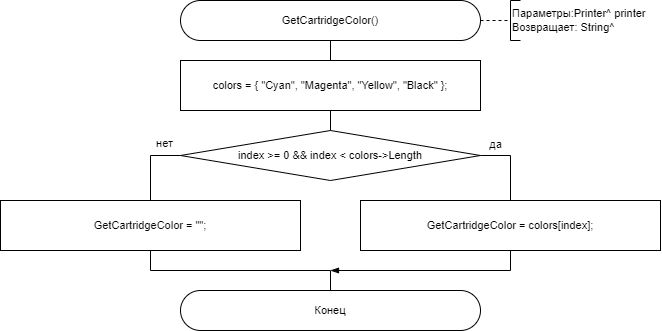


Рисунок 67 — Схема алгоритма для метода GetCartridgeColor класса PrinterSimulation

Схема алгоритма для метода generateDocumentSwitch\_CheckedChanged класса PrinterSimulation представлена на рисунке 68.

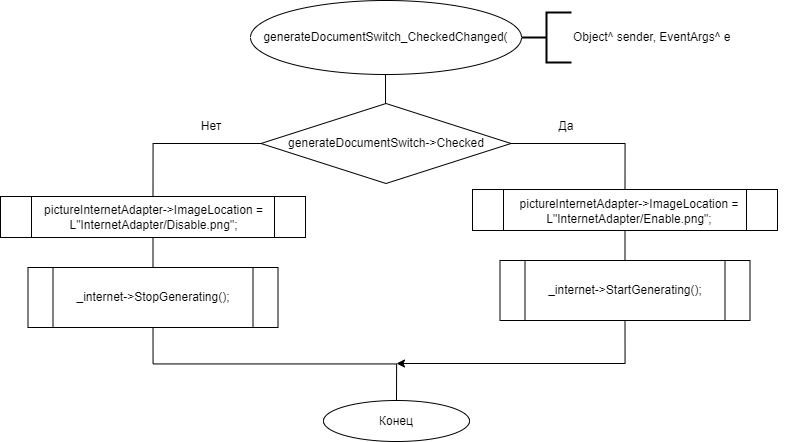


Рисунок 68 — Схема алгоритма для метода generateDocumentSwitch\_CheckedChanged класса PrinterSimulation

Схема алгоритма для метода ComputerTimer\_Tick класса PrinterSimulation представлена на рисунке 69.

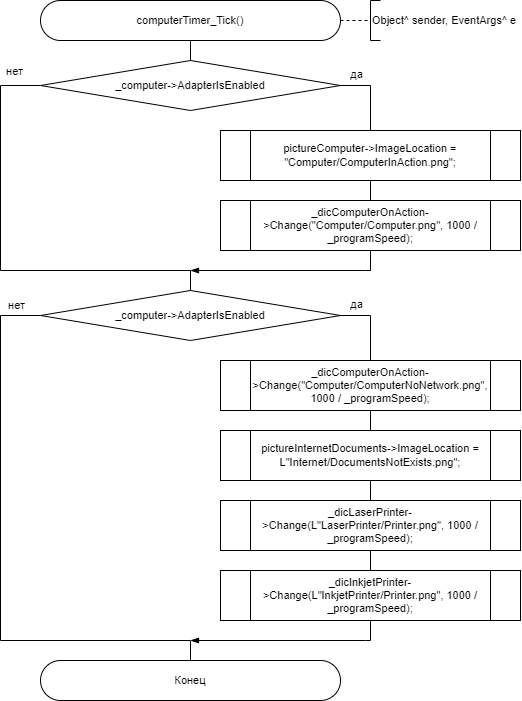


Рисунок 69 — Схема алгоритма для метода ComputerTimer\_Tick класса PrinterSimulation

Схема алгоритма для метода CalculatePaperTrayFill класса PrinterSimulation представлена на рисунке 70.

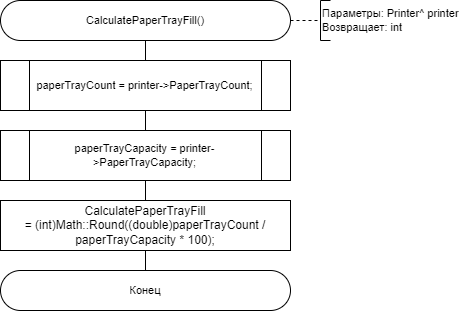


Рисунок 70 — Схема алгоритма для метода CalculatePaperTrayFill класса PrinterSimulation

Схема алгоритма для метода CalculateCartridgeFill класса PrinterSimulation представлена на рисунке 71.

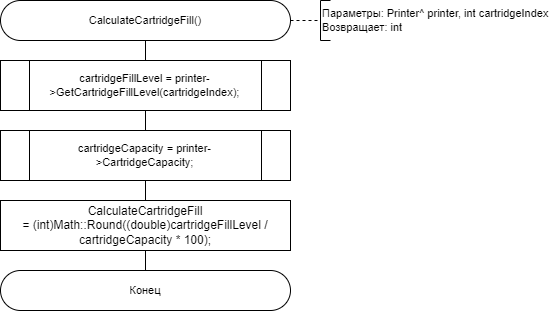


Рисунок 71 — Схема алгоритма для метода CalculateCartridgeFill класса PrinterSimulation

КОД ПРОГРАММЫ

2.1 Пространство имен SimulationObjects

2.1.1 Document

Код заголовочного файла:

#pragma once

namespace SimulationObjects

{

public ref class Document

{

private:

int \_pageCount;

array<System::String^>^ \_colors;

public:

Document(int pageCount);

Document(int pageCount, array<System::String^>^ colors);

property int PageCount {

int get();

void set(int value);

}

bool IsColor();

System::String^ GetPageColor(int pageIndex);

};

}

Код исходного файла:

#include "pch.h"

#include "Document.h"

namespace SimulationObjects

{

Document::Document(int pageCount)

{

\_pageCount = pageCount;

}

Document::Document(int pageCount, array<System::String^>^ colors)

{

\_pageCount = pageCount;

\_colors = colors;

}

int Document::PageCount::get()

{

return \_pageCount;

}

void Document::PageCount::set(int value)

{

\_pageCount = value;

}

bool Document::IsColor()

{

if (\_colors == nullptr)

return false;

return true;

}

System::String^ Document::GetPageColor(int pageIndex)

{

return \_colors[pageIndex];

}

}

2.1.2 Internet

Код заголовочного файла:

#pragma once

#include "Document.h"

namespace SimulationObjects

{

public ref class Internet

{

private:

System::Windows::Forms::Timer^ \_requestDocumentTimer;

System::Windows::Forms::Timer^ \_generateDocumentTimer;

int \_requestDocumentDelay;

int \_generateDocumentDelay;

private:

System::Collections::Generic::Queue<Document^>^ \_documents = gcnew System::Collections::Generic::Queue<Document^>();

void GenerateDocument(Object^ sender, System::EventArgs^ e);

void TrySendDocument(Object^ sender, System::EventArgs^ e);

public:

Internet(int requestDocumentDelay, int generateDocumentDelay);

void StartGenerating();

void StopGenerating();

void RequestDocument();

void ChangeTimerInterval(int speed);

int GetCurrentDocumentCount();

delegate void DocumentSentEventHandler(Document^ document);

event DocumentSentEventHandler^ DocumentSent;

};

}

Код исходного файла:

#include "pch.h"

#include "Internet.h"

#include "Document.h"

using namespace System;

namespace SimulationObjects

{

Internet::Internet(int requestDocumentDelay, int generateDocumentDelay)

{

\_requestDocumentDelay = requestDocumentDelay;

\_requestDocumentTimer = gcnew System::Windows::Forms::Timer();

\_requestDocumentTimer->Interval = requestDocumentDelay;

\_requestDocumentTimer->Start();

\_generateDocumentDelay = generateDocumentDelay;

\_generateDocumentTimer = gcnew System::Windows::Forms::Timer();

\_generateDocumentTimer->Interval = generateDocumentDelay;

\_generateDocumentTimer->Tick += gcnew EventHandler(this, &Internet::GenerateDocument);

}

void Internet::GenerateDocument(Object^ sender, System::EventArgs^ e)

{

Document^ document;

Random^ random = gcnew Random();

int pageCount = random->Next(3, 11);

if (random->Next(3) == 2)

{

array<String^>^ colors = gcnew array<String^>(pageCount);

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

{

int colorIndex = random->Next(4);

if (colorIndex == 0)

colors[i] = "Cyan";

else if (colorIndex == 1)

colors[i] = "Magenta";

else if (colorIndex == 2)

colors[i] = "Yellow";

else

colors[i] = "Black";

}

document = gcnew Document(pageCount, colors);

}

else

{

document = gcnew Document(pageCount);

}

\_documents->Enqueue(document);

}

void Internet::TrySendDocument(Object^ sender, System::EventArgs^ e)

{

\_requestDocumentTimer->Tick -= gcnew EventHandler(this, &Internet::TrySendDocument);

if (\_documents->Count > 0)

DocumentSent(\_documents->Dequeue());

}

void Internet::StartGenerating()

{

\_generateDocumentTimer->Start();

}

void Internet::StopGenerating()

{

\_generateDocumentTimer->Stop();

}

void Internet::RequestDocument()

{

\_requestDocumentTimer->Tick += gcnew EventHandler(this, &Internet::TrySendDocument);

}

void Internet::ChangeTimerInterval(int speed)

{

\_requestDocumentTimer->Interval = \_requestDocumentDelay / speed;

\_generateDocumentTimer->Interval = \_generateDocumentDelay / speed;

}

int Internet::GetCurrentDocumentCount()

{

return \_documents->Count;

}

}

2.1.3 Computer

Код заголовочного файла:

#pragma once

#include "Document.h"

#include "Internet.h"

#include "Printer.h"

using namespace System::Windows::Forms;

namespace SimulationObjects

{

ref class Printer;

public ref class Computer

{

// поля класса

private:

Timer^ \_timer;

Internet^ \_internet;

bool \_internetAdapterIsEnabled;

public:

Computer(Internet^ internet, Timer^ timer); // конструктор

//Свойство

property bool AdapterIsEnabled

{

bool get();

void set(bool value);

}

// Методы

void RequestDocument(System::Object^ sender, System::EventArgs^ e);

void ChangeTimerInterval(int speed);

void SendDocumentToPrinter(Document^ document);

void SubscribeToEvents(Printer^ printer);

void UnsubscribeFromEvents(Printer^ printer);

void OnPaperTrayEmpty(Printer^ printer);

void OnCartridgeEmpty(Printer^ printer, int index);

//// Делегаты и события

//delegate void DocumentRequestedEventHandler();

//event DocumentRequestedEventHandler^ DocumentRequested;

delegate void DocumentSentToInkjetPrinterEventHandler(Document^ document);

event DocumentSentToInkjetPrinterEventHandler^ DocumentSentToInkjetPrinter;

delegate void DocumentSentToPrinterEventHandler(Document^ document);

event DocumentSentToPrinterEventHandler^ DocumentSentToPrinter;

delegate void CartridgeEmptyEventHandler(Printer^ printer, int index);

event CartridgeEmptyEventHandler^ CartridgeEmpty;

delegate void PaperTrayEmptyEventHandler(Printer^ printer);

event PaperTrayEmptyEventHandler^ PaperTrayEmpty;

};

}

Код исходного файла:

#include "pch.h"

#include "Computer.h"

namespace SimulationObjects

{

Computer::Computer(Internet^ internet, Timer^ timer) // Конструктор с параметрами

{

\_internet = internet;

\_internet->DocumentSent += gcnew Internet::DocumentSentEventHandler(this, &Computer::SendDocumentToPrinter); // обработчик события

\_timer = timer;

\_timer->Tick += gcnew System::EventHandler(this, &Computer::RequestDocument);

}

// Свойства

bool Computer::AdapterIsEnabled::get() { // Возвращает состояние интернет-адаптера

return \_internetAdapterIsEnabled;

}

void Computer::AdapterIsEnabled::set(bool value) { // устанавливает состояние интернет-адаптера

\_internetAdapterIsEnabled = value;

}

// Методы

void Computer::ChangeTimerInterval(int speed) // Изменяет интервал таймера в зависимости от заданной скорости

{

\_timer->Interval = 2000 / speed;

}

void Computer::RequestDocument(System::Object^ sender, System::EventArgs^ e) // запрашивает докумень чз интернет, если адаптер включен

{

if (\_internetAdapterIsEnabled)

\_internet->RequestDocument();

}

void Computer::SendDocumentToPrinter(Document^ document) // отправляет докумень на принтер

{

if (document->IsColor())

DocumentSentToInkjetPrinter(document);

else

DocumentSentToPrinter(document);

}

// Слудеющие 2 метода отписываются и подписываются от событий принтера

void Computer::SubscribeToEvents(Printer^ printer)

{

printer->PaperTrayEmpty += gcnew Printer::PaperTrayEmptyEventHandler(this, &Computer::OnPaperTrayEmpty);

printer->CartridgeEmpty += gcnew Printer::CartridgeEmptyEventHandler(this, &Computer::OnCartridgeEmpty);

}

void Computer::UnsubscribeFromEvents(Printer^ printer)

{

printer->PaperTrayEmpty -= gcnew Printer::PaperTrayEmptyEventHandler(this, &Computer::OnPaperTrayEmpty);

printer->CartridgeEmpty -= gcnew Printer::CartridgeEmptyEventHandler(this, &Computer::OnCartridgeEmpty);

}

// обработчики событий

void Computer::OnPaperTrayEmpty(Printer^ printer)

{

PaperTrayEmpty(printer);

}

void Computer::OnCartridgeEmpty(Printer^ printer, int index)

{

CartridgeEmpty(printer, index);

}

}

2.1.4 Printer

Код заголовочного файла:

#pragma once

#include "Document.h"

#include "Computer.h"

namespace SimulationObjects

{

ref class Document;

ref class Computer;

public ref class Printer

{

protected: // Поля класса

int \_cartridgeFillLevel;

protected:

Printer();

int \_cartridgeCapacity;

int \_paperTrayCapacity;

int \_paperTrayCount;

public:

// Свойства

Printer(Computer^ computer);

property int PaperTrayCount

{

int get();

void set(int value);

}

property int PaperTrayCapacity // вместимость бумаги

{

int get();

}

property int CartridgeCapacity // вместимость картриджа

{

int get();

}

// Методы

virtual void Print(Document^ document);

virtual int GetCartridgeFillLevel(int index);

virtual void FillCartridge(int index);

// Делагаты и события

delegate void PrintingEventHandler(Printer^ printer);

event PrintingEventHandler^ Printing;

delegate void StopPrintingEventHandler(Printer^ printer);

event StopPrintingEventHandler^ StopPrinting;

delegate void CartridgeEmptyEventHandler(Printer^ printer, int index);

event CartridgeEmptyEventHandler^ CartridgeEmpty;

delegate void PaperTrayEmptyEventHandler(Printer^ printer);

event PaperTrayEmptyEventHandler^ PaperTrayEmpty;

};

}

Код исходного файла:

#include "pch.h"

#include "Printer.h"

namespace SimulationObjects

{

Printer::Printer() {} // Конструктор

Printer::Printer(Computer^ computer) // Конструктор с параметром

{

\_cartridgeCapacity = 50; // Емкость картриджа

\_cartridgeFillLevel = 50; // Текущий уровень чернил в картридже

\_paperTrayCapacity = 70; // Вместимость лотка для бумаги

\_paperTrayCount = 70; // Текущее кол-во бумаги

computer->DocumentSentToPrinter += gcnew Computer::DocumentSentToPrinterEventHandler(this, &Printer::Print); // Получение документов для печати

}

int Printer::CartridgeCapacity::get() // Возвращает емкость картриджа

{

return \_cartridgeCapacity;

}

int Printer::PaperTrayCapacity::get() // Возвращает вместимость лотка для бумаги

{

return \_paperTrayCapacity;

}

int Printer::PaperTrayCount::get() // Возвращает текущее кол-во бумаги в лотке

{

return \_paperTrayCount;

}

void Printer::PaperTrayCount::set(int value) // Устанавливает текущее кол-во бумаги в лотке

{

\_paperTrayCount = value;

}

void Printer::Print(Document^ document)

{

int pageCount = document->PageCount;

System::Random^ random = gcnew System::Random();

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

{

Printing(this); // Вызов события

PaperTrayCount--;

if (\_paperTrayCount <= 0)

PaperTrayEmpty(this); // Вызов события

\_cartridgeFillLevel--;

if (\_cartridgeFillLevel <= 0)

CartridgeEmpty(this, -1); // Вызов события

}

StopPrinting(this); // Вызов события

}

int Printer::GetCartridgeFillLevel(int index) // Возвращает тукцщий уровень чернил

{

return \_cartridgeFillLevel;

}

void Printer::FillCartridge(int index) // Заправляет картридж до полной емкости

{

\_cartridgeFillLevel = \_cartridgeCapacity;

}

}

2.1.5 InkjetPrinter

Код заголовочного файла:

#pragma once

#include "Printer.h"

#include "Computer.h"

namespace SimulationObjects

{

public ref class InkjetPrinter : Printer

{

protected:

// Поля и свойства

array<int>^ \_cartridgeFillLevels;

array<System::String^>^ \_cartridgeColors;

public:

InkjetPrinter(Computer^ computer); // конструктор с параметрами

// методы

void Print(Document^ document) override; // переопределяет виртуальный метод из базового класса `Printer`

int GetCartridgeFillLevel(int index) override; // возвращает уровень заполнения картриджа по указанному индексу

void FillCartridge(int index) override; // заполняет картридж по указанному индексу.

};

}

Код исходного файла:

#include "pch.h"

#include "InkjetPrinter.h"

namespace SimulationObjects

{

InkjetPrinter::InkjetPrinter(Computer^ computer) // конструктор с параметрами

{

\_cartridgeCapacity = 15;

\_cartridgeFillLevels = gcnew array<int>{ 15, 15, 15, 15 };

\_cartridgeColors = gcnew array<System::String^>{ "Cyan", "Magenta", "Yellow", "Black" };

\_paperTrayCapacity = 50;

\_paperTrayCount = 50;

// Подписка на событие `DocumentSentToInkjetPrinter`, которое вызывает метод `Print`, когда документ отправляется на принтер.

computer->DocumentSentToInkjetPrinter += gcnew Computer::DocumentSentToInkjetPrinterEventHandler(this, &InkjetPrinter::Print);

}

void InkjetPrinter::Print(Document^ document)

{

int pageCount = document->PageCount;

System::Random^ random = gcnew System::Random();

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

{

Printing(this); // вызов события

PaperTrayCount--;

if (\_paperTrayCount <= 0)

PaperTrayEmpty(this); // вызов события

System::String^ color = document->GetPageColor(i);

// заполнение картриджей

int index = 0;

for each (System::String^ cardridgeColor in \_cartridgeColors)

{

if (cardridgeColor == color)

{

\_cartridgeFillLevels[index]--;

if (\_cartridgeFillLevels[index] <= 0)

CartridgeEmpty(this, index); // вызов события

break;

}

index++;

}

}

StopPrinting(this); // вызов события

}

int InkjetPrinter::GetCartridgeFillLevel(int index) // возвращает уровень заполнения картриджа по заданному индексу

{

return \_cartridgeFillLevels[index];

}

void InkjetPrinter::FillCartridge(int index) // заполняет картридж до максимального уровня

{

\_cartridgeFillLevels[index] = \_cartridgeCapacity;

}

}

2.1.6 Human

Код заголовочного файла:

#pragma once

#include "Computer.h"

namespace SimulationObjects {

public ref class Human

{

private:

Computer^ \_computer;

public:

// конструктор с параметром

Human(Computer^ computer);

// методы

void RefillCartridge(Printer^ printer, int index); // уведомляет о том, что картридж был перезаполнен

void RefillPaperTray(Printer^ printer); // уведомляет о том, что лоток для бумаги был перезаполнен

// делегаты и события

delegate void CartridgeRefilledEventHandler(int index);

event CartridgeRefilledEventHandler^ CartridgeRefilled;

delegate void PaperTrayRefilledEventHandler(Printer^ printer);

event PaperTrayRefilledEventHandler^ PaperTrayRefilled;

};

}

Код исходного файла:

#include "pch.h"

#include "Human.h"

namespace SimulationObjects

{

Human::Human(Computer^ computer) // конструктор класса с параметрами

{

\_computer = computer;

// обработчики событий

\_computer->CartridgeEmpty += gcnew Computer::CartridgeEmptyEventHandler(this, &Human::RefillCartridge);

\_computer->PaperTrayEmpty += gcnew Computer::PaperTrayEmptyEventHandler(this, &Human::RefillPaperTray);

}

// методы обработчики

void Human::RefillCartridge(Printer^ printer, int index) // уведомляет о том, что картридж был перезаполнен

{

printer->FillCartridge(index);

CartridgeRefilled(index);

}

void Human::RefillPaperTray(Printer^ printer) // уведомляет о том, что лоток был перезаполнен

{

printer->PaperTrayCount = printer->PaperTrayCapacity;

PaperTrayRefilled(printer);

}

}

**2.2 Пространство имен Simulation**

2.2.1 DelayedImageChanger

Код заголовочного файла:

#pragma once

using namespace System::Windows::Forms;

namespace Simulation

{

public ref class DelayedImageChanger

{

private:

PictureBox^ \_pictureBox;

System::String^ \_imagePath;

Timer^ \_timer;

void OnTick(System::Object^ sender, System::EventArgs^ e);

public:

DelayedImageChanger(PictureBox^ pictureBox);

void Change(System::String^ imagePath, int delay);

};

}

Код исходного файла:

#include "DelayedImageChanger.h"

namespace Simulation

{

DelayedImageChanger::DelayedImageChanger(PictureBox^ pictureBox)

{

\_pictureBox = pictureBox;

\_timer = gcnew Timer();

\_timer->Tick += gcnew System::EventHandler(this, &DelayedImageChanger::OnTick);

}

void DelayedImageChanger::OnTick(System::Object^ sender, System::EventArgs^ e)

{

\_timer->Stop();

\_pictureBox->ImageLocation = \_imagePath;

}

void DelayedImageChanger::Change(System::String^ imagePath, int delay)

{

if (\_timer->Enabled)

\_timer->Stop();

if (delay == 0)

\_pictureBox->ImageLocation = imagePath;

else

{

\_imagePath = imagePath;

\_timer->Interval = delay;

\_timer->Start();

}

}

}

2.2.2 DelayedVisibilityChancher

Код заголовочного файла:

#pragma once

using namespace System::Windows::Forms;

namespace Simulation

{

public ref class DelayedVisibilityChanger

{

private:

Control^ \_control;

Timer^ \_timer;

bool \_visibility;

void OnTick(System::Object^ sender, System::EventArgs^ e);

public:

DelayedVisibilityChanger(Control^ control);

void Change(bool visibility, int delay);

};

}

Код исходного файла:

#include "DelayedVisibilityChanger.h"

namespace Simulation

{

DelayedVisibilityChanger::DelayedVisibilityChanger(Control^ control)

{

\_control = control;

\_timer = gcnew Timer();

\_timer->Tick += gcnew System::EventHandler(this, &DelayedVisibilityChanger::OnTick);

}

void DelayedVisibilityChanger::OnTick(System::Object^ sender, System::EventArgs^ e)

{

\_timer->Stop();

\_control->Visible = (!\_visibility);

}

void DelayedVisibilityChanger::Change(bool visibility, int delay)

{

if (\_timer->Enabled)

\_timer->Stop();

\_control->Visible = visibility;

if (delay != 0)

{

\_visibility = visibility;

\_timer->Interval = delay;

\_timer->Start();

}

}

}

2.2.3 PrinterSimulation

Код заголовочного файла:

#pragma once;

#include "DelayedImageChanger.h"

#include "DelayedVisibilityChanger.h"

namespace Simulation {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

using namespace SimulationObjects;

/// <summary>

/// Сводка для PrinterSimulation

/// </summary>

public ref class PrinterSimulation : public System::Windows::Forms::Form

{

private:

Human^ \_human;

Internet^ \_internet;

Computer^ \_computer;

Printer^ \_laserPrinter;

InkjetPrinter^ \_inkjetPrinter;

DelayedImageChanger^ \_dicLaserPrinter;

DelayedImageChanger^ \_dicInkjetPrinter;

DelayedImageChanger^ \_dicComputerOnAction;

DelayedVisibilityChanger^ \_dvcRefillLaserPaperTray;

DelayedVisibilityChanger^ \_dvcRefillCartridge;

DelayedVisibilityChanger^ \_dvcRefillInkjetPaperTray;

DelayedVisibilityChanger^ \_dvcRefillCyanCartridge;

DelayedVisibilityChanger^ \_dvcRefillMagentaCartridge;

DelayedVisibilityChanger^ \_dvcRefillYellowCartridge;

DelayedVisibilityChanger^ \_dvcRefillBlackCartridge;

private: System::Windows::Forms::GroupBox^ internetAdapterIllustrationGroupBox;

private: System::Windows::Forms::PictureBox^ pictureInternetAdapter;

private: System::Windows::Forms::Button^ slow;

private: System::Windows::Forms::Button^ fast;

private: System::Windows::Forms::PictureBox^ pictureRefillInkjetPaperTray;

private: System::Windows::Forms::PictureBox^ pictureRefillLaserPaperTray;

int \_programSpeed;

public:

PrinterSimulation(void)

{

InitializeComponent();

\_programSpeed = 1;

\_dicComputerOnAction = gcnew DelayedImageChanger(pictureComputer);

\_dicLaserPrinter = gcnew DelayedImageChanger(pictureLaserPrinter);

\_dvcRefillLaserPaperTray = gcnew DelayedVisibilityChanger(pictureRefillLaserPaperTray);

\_dvcRefillCartridge = gcnew DelayedVisibilityChanger(pictureRefillCartridge);

\_dicInkjetPrinter = gcnew DelayedImageChanger(pictureInkjetPrinter);

\_dvcRefillInkjetPaperTray = gcnew DelayedVisibilityChanger(pictureRefillInkjetPaperTray);

\_dvcRefillCyanCartridge = gcnew DelayedVisibilityChanger(pictureRefillCyanCartridge);

\_dvcRefillMagentaCartridge = gcnew DelayedVisibilityChanger(pictureRefillMagentaCartridge);

\_dvcRefillYellowCartridge = gcnew DelayedVisibilityChanger(pictureRefillYellowCartridge);

\_dvcRefillBlackCartridge = gcnew DelayedVisibilityChanger(pictureRefillBlackCartridge);

\_internet = gcnew Internet(3000, 1800);

\_computer = gcnew Computer(\_internet, computerTimer);

\_human = gcnew Human(\_computer);

\_laserPrinter = gcnew Printer(\_computer);

\_inkjetPrinter = gcnew InkjetPrinter(\_computer);

}

private: System::Windows::Forms::PictureBox^ pictureInkjetPrinter;

private: System::Windows::Forms::PictureBox^ pictureCyanCartridge;

private: System::Windows::Forms::PictureBox^ pictureMagentaCartridge;

private: System::Windows::Forms::PictureBox^ pictureYellowCartridge;

private: System::Windows::Forms::PictureBox^ pictureBlackCartridge;

private: System::Windows::Forms::PictureBox^ pictureRefillBlackCartridge;

private: System::Windows::Forms::PictureBox^ pictureInkjetPaperTray;

private: System::Windows::Forms::PictureBox^ pictureRefillYellowCartridge;

private: System::Windows::Forms::PictureBox^ pictureRefillCyanCartridge;

private: System::Windows::Forms::PictureBox^ pictureRefillMagentaCartridge;

private: System::Windows::Forms::PictureBox^ pictureRefillCartridge;

private: System::Windows::Forms::PictureBox^ pictureComputer;

private: System::Windows::Forms::PictureBox^ pictureCartridge;

private: System::Windows::Forms::PictureBox^ pictureLaserPaperTray;

private: System::Windows::Forms::PictureBox^ pictureLaserPrinter;

private: System::Windows::Forms::CheckBox^ generateDocumentSwitch;

private: System::Windows::Forms::CheckBox^ TurnOnDevices;

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::Label^ label5;

private: System::Windows::Forms::Label^ label6;

private: System::Windows::Forms::Label^ label7;

private: System::Windows::Forms::Label^ label13;

private: System::Windows::Forms::Label^ label12;

private: System::Windows::Forms::Label^ label10;

private: System::Windows::Forms::Label^ label11;

private: System::Windows::Forms::Label^ laserPaperCountLabel;

private: System::Windows::Forms::Label^ cartridgeFillLevelLabel;

private: System::Windows::Forms::Label^ inkjetPaperCountLabel;

private: System::Windows::Forms::Label^ blackCartridgeFillLevelLabel;

private: System::Windows::Forms::Label^ yellowCartridgeFillLevelLabel;

private: System::Windows::Forms::Label^ magentaCartridgeFillLevelLabel;

private: System::Windows::Forms::Label^ cyanCartridgeFillLevelLabel;

private: System::Windows::Forms::Label^ documentCountLabel;

private: System::Windows::Forms::GroupBox^ internetGroupBox;

private: System::Windows::Forms::GroupBox^ laserPrinterIllustrationGroupBox;

private: System::Windows::Forms::GroupBox^ inkjetPrinterIllustrationGroupBox;

private: System::Windows::Forms::GroupBox^ computerIllustrationGroupBox;

private: System::Windows::Forms::GroupBox^ controlPanelGroupBox;

private: System::Windows::Forms::GroupBox^ laserPrinterGroupBox;

private: System::Windows::Forms::GroupBox^ inkjetPrinterGroupBox;

private: System::Windows::Forms::GroupBox^ simulationGroupBox;

private: System::Windows::Forms::Timer^ timer;

private: System::Windows::Forms::Timer^ computerTimer;

private: System::Windows::Forms::TrackBar^ simulationSpeedTrackBar;

private: System::ComponentModel::IContainer^ components;

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~PrinterSimulation()

{

if (components)

{

delete components;

}

}

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

this->components = (gcnew System::ComponentModel::Container());

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(PrinterSimulation::typeid));

this->TurnOnDevices = (gcnew System::Windows::Forms::CheckBox());

this->timer = (gcnew System::Windows::Forms::Timer(this->components));

this->pictureComputer = (gcnew System::Windows::Forms::PictureBox());

this->pictureLaserPrinter = (gcnew System::Windows::Forms::PictureBox());

this->generateDocumentSwitch = (gcnew System::Windows::Forms::CheckBox());

this->label1 = (gcnew System::Windows::Forms::Label());

this->label2 = (gcnew System::Windows::Forms::Label());

this->label3 = (gcnew System::Windows::Forms::Label());

this->label4 = (gcnew System::Windows::Forms::Label());

this->label5 = (gcnew System::Windows::Forms::Label());

this->label6 = (gcnew System::Windows::Forms::Label());

this->label7 = (gcnew System::Windows::Forms::Label());

this->computerTimer = (gcnew System::Windows::Forms::Timer(this->components));

this->controlPanelGroupBox = (gcnew System::Windows::Forms::GroupBox());

this->slow = (gcnew System::Windows::Forms::Button());

this->fast = (gcnew System::Windows::Forms::Button());

this->label13 = (gcnew System::Windows::Forms::Label());

this->label12 = (gcnew System::Windows::Forms::Label());

this->label11 = (gcnew System::Windows::Forms::Label());

this->simulationSpeedTrackBar = (gcnew System::Windows::Forms::TrackBar());

this->laserPrinterGroupBox = (gcnew System::Windows::Forms::GroupBox());

this->laserPaperCountLabel = (gcnew System::Windows::Forms::Label());

this->cartridgeFillLevelLabel = (gcnew System::Windows::Forms::Label());

this->inkjetPrinterGroupBox = (gcnew System::Windows::Forms::GroupBox());

this->inkjetPaperCountLabel = (gcnew System::Windows::Forms::Label());

this->blackCartridgeFillLevelLabel = (gcnew System::Windows::Forms::Label());

this->yellowCartridgeFillLevelLabel = (gcnew System::Windows::Forms::Label());

this->magentaCartridgeFillLevelLabel = (gcnew System::Windows::Forms::Label());

this->cyanCartridgeFillLevelLabel = (gcnew System::Windows::Forms::Label());

this->pictureCartridge = (gcnew System::Windows::Forms::PictureBox());

this->pictureLaserPaperTray = (gcnew System::Windows::Forms::PictureBox());

this->internetGroupBox = (gcnew System::Windows::Forms::GroupBox());

this->documentCountLabel = (gcnew System::Windows::Forms::Label());

this->label10 = (gcnew System::Windows::Forms::Label());

this->simulationGroupBox = (gcnew System::Windows::Forms::GroupBox());

this->inkjetPrinterIllustrationGroupBox = (gcnew System::Windows::Forms::GroupBox());

this->pictureCyanCartridge = (gcnew System::Windows::Forms::PictureBox());

this->pictureRefillInkjetPaperTray = (gcnew System::Windows::Forms::PictureBox());

this->pictureMagentaCartridge = (gcnew System::Windows::Forms::PictureBox());

this->pictureYellowCartridge = (gcnew System::Windows::Forms::PictureBox());

this->pictureBlackCartridge = (gcnew System::Windows::Forms::PictureBox());

this->pictureRefillBlackCartridge = (gcnew System::Windows::Forms::PictureBox());

this->pictureRefillYellowCartridge = (gcnew System::Windows::Forms::PictureBox());

this->pictureRefillCyanCartridge = (gcnew System::Windows::Forms::PictureBox());

this->pictureRefillMagentaCartridge = (gcnew System::Windows::Forms::PictureBox());

this->pictureInkjetPrinter = (gcnew System::Windows::Forms::PictureBox());

this->pictureInkjetPaperTray = (gcnew System::Windows::Forms::PictureBox());

this->internetAdapterIllustrationGroupBox = (gcnew System::Windows::Forms::GroupBox());

this->pictureInternetAdapter = (gcnew System::Windows::Forms::PictureBox());

this->laserPrinterIllustrationGroupBox = (gcnew System::Windows::Forms::GroupBox());

this->pictureRefillLaserPaperTray = (gcnew System::Windows::Forms::PictureBox());

this->pictureRefillCartridge = (gcnew System::Windows::Forms::PictureBox());

this->computerIllustrationGroupBox = (gcnew System::Windows::Forms::GroupBox());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureComputer))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureLaserPrinter))->BeginInit();

this->controlPanelGroupBox->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->simulationSpeedTrackBar))->BeginInit();

this->laserPrinterGroupBox->SuspendLayout();

this->inkjetPrinterGroupBox->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureCartridge))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureLaserPaperTray))->BeginInit();

this->internetGroupBox->SuspendLayout();

this->simulationGroupBox->SuspendLayout();

this->inkjetPrinterIllustrationGroupBox->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureCyanCartridge))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureRefillInkjetPaperTray))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureMagentaCartridge))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureYellowCartridge))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBlackCartridge))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureRefillBlackCartridge))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureRefillYellowCartridge))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureRefillCyanCartridge))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureRefillMagentaCartridge))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureInkjetPrinter))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureInkjetPaperTray))->BeginInit();

this->internetAdapterIllustrationGroupBox->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureInternetAdapter))->BeginInit();

this->laserPrinterIllustrationGroupBox->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureRefillLaserPaperTray))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureRefillCartridge))->BeginInit();

this->computerIllustrationGroupBox->SuspendLayout();

this->SuspendLayout();

//

// TurnOnDevices

//

this->TurnOnDevices->AutoSize = true;

this->TurnOnDevices->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->TurnOnDevices->Location = System::Drawing::Point(8, 58);

this->TurnOnDevices->Margin = System::Windows::Forms::Padding(4);

this->TurnOnDevices->Name = L"TurnOnDevices";

this->TurnOnDevices->Size = System::Drawing::Size(240, 24);

this->TurnOnDevices->TabIndex = 2;

this->TurnOnDevices->Text = L"Включить оборудование";

this->TurnOnDevices->UseVisualStyleBackColor = true;

this->TurnOnDevices->CheckedChanged += gcnew System::EventHandler(this, &PrinterSimulation::TurnOnDevices\_CheckedChanged);

//

// timer

//

this->timer->Enabled = true;

this->timer->Tick += gcnew System::EventHandler(this, &PrinterSimulation::timer\_Tick);

//

// pictureComputer

//

this->pictureComputer->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureComputer.Image")));

this->pictureComputer->Location = System::Drawing::Point(77, 28);

this->pictureComputer->Margin = System::Windows::Forms::Padding(4);

this->pictureComputer->Name = L"pictureComputer";

this->pictureComputer->Size = System::Drawing::Size(192, 192);

this->pictureComputer->SizeMode = System::Windows::Forms::PictureBoxSizeMode::AutoSize;

this->pictureComputer->TabIndex = 3;

this->pictureComputer->TabStop = false;

this->pictureComputer->Click += gcnew System::EventHandler(this, &PrinterSimulation::pictureComputer\_Click);

//

// pictureLaserPrinter

//

this->pictureLaserPrinter->BackColor = System::Drawing::SystemColors::ButtonHighlight;

this->pictureLaserPrinter->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureLaserPrinter.Image")));

this->pictureLaserPrinter->ImageLocation = L"";

this->pictureLaserPrinter->Location = System::Drawing::Point(278, 97);

this->pictureLaserPrinter->Margin = System::Windows::Forms::Padding(4);

this->pictureLaserPrinter->Name = L"pictureLaserPrinter";

this->pictureLaserPrinter->Size = System::Drawing::Size(212, 180);

this->pictureLaserPrinter->SizeMode = System::Windows::Forms::PictureBoxSizeMode::AutoSize;

this->pictureLaserPrinter->TabIndex = 7;

this->pictureLaserPrinter->TabStop = false;

//

// generateDocumentSwitch

//

this->generateDocumentSwitch->AutoSize = true;

this->generateDocumentSwitch->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->generateDocumentSwitch->Location = System::Drawing::Point(8, 26);

this->generateDocumentSwitch->Margin = System::Windows::Forms::Padding(4);

this->generateDocumentSwitch->Name = L"generateDocumentSwitch";

this->generateDocumentSwitch->Size = System::Drawing::Size(200, 24);

this->generateDocumentSwitch->TabIndex = 8;

this->generateDocumentSwitch->Text = L"Включить интернет";

this->generateDocumentSwitch->UseVisualStyleBackColor = true;

this->generateDocumentSwitch->CheckedChanged += gcnew System::EventHandler(this, &PrinterSimulation::generateDocumentSwitch\_CheckedChanged);

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label1->Location = System::Drawing::Point(8, 27);

this->label1->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(267, 20);

this->label1->TabIndex = 10;

this->label1->Text = L"Заполненность картриджа, %:";

//

// label2

//

this->label2->AutoSize = true;

this->label2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label2->Location = System::Drawing::Point(8, 28);

this->label2->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(285, 20);

this->label2->TabIndex = 15;

this->label2->Text = L"Заполненность C-картриджа, %:";

//

// label3

//

this->label3->AutoSize = true;

this->label3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label3->Location = System::Drawing::Point(8, 60);

this->label3->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(287, 20);

this->label3->TabIndex = 16;

this->label3->Text = L"Заполненность M-картриджа, %:";

//

// label4

//

this->label4->AutoSize = true;

this->label4->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label4->Location = System::Drawing::Point(8, 95);

this->label4->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(283, 20);

this->label4->TabIndex = 17;

this->label4->Text = L"Заполненность Y-картриджа, %:";

//

// label5

//

this->label5->AutoSize = true;

this->label5->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label5->Location = System::Drawing::Point(8, 129);

this->label5->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(284, 20);

this->label5->TabIndex = 18;

this->label5->Text = L"Заполненность K-картриджа, %:";

//

// label6

//

this->label6->AutoSize = true;

this->label6->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label6->Location = System::Drawing::Point(8, 58);

this->label6->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label6->Name = L"label6";

this->label6->Size = System::Drawing::Size(177, 20);

this->label6->TabIndex = 21;

this->label6->Text = L"Количество бумаги:";

//

// label7

//

this->label7->AutoSize = true;

this->label7->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label7->Location = System::Drawing::Point(8, 164);

this->label7->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label7->Name = L"label7";

this->label7->Size = System::Drawing::Size(177, 20);

this->label7->TabIndex = 22;

this->label7->Text = L"Количество бумаги:";

//

// computerTimer

//

this->computerTimer->Enabled = true;

this->computerTimer->Interval = 3000;

this->computerTimer->Tick += gcnew System::EventHandler(this, &PrinterSimulation::computerTimer\_Tick);

//

// controlPanelGroupBox

//

this->controlPanelGroupBox->Controls->Add(this->slow);

this->controlPanelGroupBox->Controls->Add(this->fast);

this->controlPanelGroupBox->Controls->Add(this->label13);

this->controlPanelGroupBox->Controls->Add(this->label12);

this->controlPanelGroupBox->Controls->Add(this->label11);

this->controlPanelGroupBox->Controls->Add(this->simulationSpeedTrackBar);

this->controlPanelGroupBox->Controls->Add(this->generateDocumentSwitch);

this->controlPanelGroupBox->Controls->Add(this->TurnOnDevices);

this->controlPanelGroupBox->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->controlPanelGroupBox->Location = System::Drawing::Point(16, 15);

this->controlPanelGroupBox->Margin = System::Windows::Forms::Padding(4);

this->controlPanelGroupBox->Name = L"controlPanelGroupBox";

this->controlPanelGroupBox->Padding = System::Windows::Forms::Padding(4);

this->controlPanelGroupBox->Size = System::Drawing::Size(365, 217);

this->controlPanelGroupBox->TabIndex = 23;

this->controlPanelGroupBox->TabStop = false;

this->controlPanelGroupBox->Text = L"Панель управления";

//

// slow

//

this->slow->Location = System::Drawing::Point(101, 161);

this->slow->Name = L"slow";

this->slow->Size = System::Drawing::Size(43, 40);

this->slow->TabIndex = 35;

this->slow->Text = L"<";

this->slow->UseVisualStyleBackColor = true;

this->slow->Click += gcnew System::EventHandler(this, &PrinterSimulation::slow\_Click);

//

// fast

//

this->fast->Location = System::Drawing::Point(219, 161);

this->fast->Name = L"fast";

this->fast->Size = System::Drawing::Size(44, 40);

this->fast->TabIndex = 34;

this->fast->Text = L">";

this->fast->UseVisualStyleBackColor = true;

this->fast->Click += gcnew System::EventHandler(this, &PrinterSimulation::fast\_Click);

//

// label13

//

this->label13->AutoSize = true;

this->label13->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label13->Location = System::Drawing::Point(325, 114);

this->label13->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label13->Name = L"label13";

this->label13->Size = System::Drawing::Size(18, 20);

this->label13->TabIndex = 33;

this->label13->Text = L"5";

//

// label12

//

this->label12->AutoSize = true;

this->label12->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label12->Location = System::Drawing::Point(8, 114);

this->label12->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label12->Name = L"label12";

this->label12->Size = System::Drawing::Size(18, 20);

this->label12->TabIndex = 33;

this->label12->Text = L"1";

//

// label11

//

this->label11->AutoSize = true;

this->label11->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label11->Location = System::Drawing::Point(8, 86);

this->label11->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label11->Name = L"label11";

this->label11->Size = System::Drawing::Size(94, 20);

this->label11->TabIndex = 28;

this->label11->Text = L"Скорость:";

//

// simulationSpeedTrackBar

//

this->simulationSpeedTrackBar->LargeChange = 2;

this->simulationSpeedTrackBar->Location = System::Drawing::Point(35, 114);

this->simulationSpeedTrackBar->Margin = System::Windows::Forms::Padding(4);

this->simulationSpeedTrackBar->Maximum = 5;

this->simulationSpeedTrackBar->Minimum = 1;

this->simulationSpeedTrackBar->Name = L"simulationSpeedTrackBar";

this->simulationSpeedTrackBar->Size = System::Drawing::Size(283, 56);

this->simulationSpeedTrackBar->TabIndex = 32;

this->simulationSpeedTrackBar->Value = 1;

this->simulationSpeedTrackBar->Scroll += gcnew System::EventHandler(this, &PrinterSimulation::simulationSpeedTrackBar\_ValueChanged);

this->simulationSpeedTrackBar->ValueChanged += gcnew System::EventHandler(this, &PrinterSimulation::simulationSpeedTrackBar\_ValueChanged);

//

// laserPrinterGroupBox

//

this->laserPrinterGroupBox->Controls->Add(this->laserPaperCountLabel);

this->laserPrinterGroupBox->Controls->Add(this->cartridgeFillLevelLabel);

this->laserPrinterGroupBox->Controls->Add(this->label1);

this->laserPrinterGroupBox->Controls->Add(this->label6);

this->laserPrinterGroupBox->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->laserPrinterGroupBox->Location = System::Drawing::Point(16, 258);

this->laserPrinterGroupBox->Margin = System::Windows::Forms::Padding(4);

this->laserPrinterGroupBox->Name = L"laserPrinterGroupBox";

this->laserPrinterGroupBox->Padding = System::Windows::Forms::Padding(4);

this->laserPrinterGroupBox->Size = System::Drawing::Size(365, 98);

this->laserPrinterGroupBox->TabIndex = 24;

this->laserPrinterGroupBox->TabStop = false;

this->laserPrinterGroupBox->Text = L"Лазерный принтер";

//

// laserPaperCountLabel

//

this->laserPaperCountLabel->AutoSize = true;

this->laserPaperCountLabel->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->laserPaperCountLabel->Location = System::Drawing::Point(307, 58);

this->laserPaperCountLabel->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->laserPaperCountLabel->Name = L"laserPaperCountLabel";

this->laserPaperCountLabel->Size = System::Drawing::Size(27, 20);

this->laserPaperCountLabel->TabIndex = 22;

this->laserPaperCountLabel->Text = L"70";

//

// cartridgeFillLevelLabel

//

this->cartridgeFillLevelLabel->AutoSize = true;

this->cartridgeFillLevelLabel->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->cartridgeFillLevelLabel->Location = System::Drawing::Point(307, 27);

this->cartridgeFillLevelLabel->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->cartridgeFillLevelLabel->Name = L"cartridgeFillLevelLabel";

this->cartridgeFillLevelLabel->Size = System::Drawing::Size(36, 20);

this->cartridgeFillLevelLabel->TabIndex = 22;

this->cartridgeFillLevelLabel->Text = L"100";

//

// inkjetPrinterGroupBox

//

this->inkjetPrinterGroupBox->Controls->Add(this->inkjetPaperCountLabel);

this->inkjetPrinterGroupBox->Controls->Add(this->blackCartridgeFillLevelLabel);

this->inkjetPrinterGroupBox->Controls->Add(this->yellowCartridgeFillLevelLabel);

this->inkjetPrinterGroupBox->Controls->Add(this->magentaCartridgeFillLevelLabel);

this->inkjetPrinterGroupBox->Controls->Add(this->cyanCartridgeFillLevelLabel);

this->inkjetPrinterGroupBox->Controls->Add(this->label2);

this->inkjetPrinterGroupBox->Controls->Add(this->label3);

this->inkjetPrinterGroupBox->Controls->Add(this->label7);

this->inkjetPrinterGroupBox->Controls->Add(this->label4);

this->inkjetPrinterGroupBox->Controls->Add(this->label5);

this->inkjetPrinterGroupBox->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->inkjetPrinterGroupBox->Location = System::Drawing::Point(16, 373);

this->inkjetPrinterGroupBox->Margin = System::Windows::Forms::Padding(4);

this->inkjetPrinterGroupBox->Name = L"inkjetPrinterGroupBox";

this->inkjetPrinterGroupBox->Padding = System::Windows::Forms::Padding(4);

this->inkjetPrinterGroupBox->Size = System::Drawing::Size(365, 196);

this->inkjetPrinterGroupBox->TabIndex = 25;

this->inkjetPrinterGroupBox->TabStop = false;

this->inkjetPrinterGroupBox->Text = L"Струйный принтер";

//

// inkjetPaperCountLabel

//

this->inkjetPaperCountLabel->AutoSize = true;

this->inkjetPaperCountLabel->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->inkjetPaperCountLabel->Location = System::Drawing::Point(307, 164);

this->inkjetPaperCountLabel->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->inkjetPaperCountLabel->Name = L"inkjetPaperCountLabel";

this->inkjetPaperCountLabel->Size = System::Drawing::Size(27, 20);

this->inkjetPaperCountLabel->TabIndex = 22;

this->inkjetPaperCountLabel->Text = L"50";

//

// blackCartridgeFillLevelLabel

//

this->blackCartridgeFillLevelLabel->AutoSize = true;

this->blackCartridgeFillLevelLabel->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->blackCartridgeFillLevelLabel->Location = System::Drawing::Point(307, 129);

this->blackCartridgeFillLevelLabel->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->blackCartridgeFillLevelLabel->Name = L"blackCartridgeFillLevelLabel";

this->blackCartridgeFillLevelLabel->Size = System::Drawing::Size(36, 20);

this->blackCartridgeFillLevelLabel->TabIndex = 22;

this->blackCartridgeFillLevelLabel->Text = L"100";

//

// yellowCartridgeFillLevelLabel

//

this->yellowCartridgeFillLevelLabel->AutoSize = true;

this->yellowCartridgeFillLevelLabel->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->yellowCartridgeFillLevelLabel->Location = System::Drawing::Point(307, 95);

this->yellowCartridgeFillLevelLabel->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->yellowCartridgeFillLevelLabel->Name = L"yellowCartridgeFillLevelLabel";

this->yellowCartridgeFillLevelLabel->Size = System::Drawing::Size(36, 20);

this->yellowCartridgeFillLevelLabel->TabIndex = 22;

this->yellowCartridgeFillLevelLabel->Text = L"100";

//

// magentaCartridgeFillLevelLabel

//

this->magentaCartridgeFillLevelLabel->AutoSize = true;

this->magentaCartridgeFillLevelLabel->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->magentaCartridgeFillLevelLabel->Location = System::Drawing::Point(307, 60);

this->magentaCartridgeFillLevelLabel->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->magentaCartridgeFillLevelLabel->Name = L"magentaCartridgeFillLevelLabel";

this->magentaCartridgeFillLevelLabel->Size = System::Drawing::Size(36, 20);

this->magentaCartridgeFillLevelLabel->TabIndex = 22;

this->magentaCartridgeFillLevelLabel->Text = L"100";

//

// cyanCartridgeFillLevelLabel

//

this->cyanCartridgeFillLevelLabel->AutoSize = true;

this->cyanCartridgeFillLevelLabel->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->cyanCartridgeFillLevelLabel->Location = System::Drawing::Point(307, 28);

this->cyanCartridgeFillLevelLabel->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->cyanCartridgeFillLevelLabel->Name = L"cyanCartridgeFillLevelLabel";

this->cyanCartridgeFillLevelLabel->Size = System::Drawing::Size(36, 20);

this->cyanCartridgeFillLevelLabel->TabIndex = 22;

this->cyanCartridgeFillLevelLabel->Text = L"100";

//

// pictureCartridge

//

this->pictureCartridge->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureCartridge.Image")));

this->pictureCartridge->ImageLocation = L"";

this->pictureCartridge->Location = System::Drawing::Point(589, 274);

this->pictureCartridge->Margin = System::Windows::Forms::Padding(4);

this->pictureCartridge->Name = L"pictureCartridge";

this->pictureCartridge->Size = System::Drawing::Size(121, 128);

this->pictureCartridge->SizeMode = System::Windows::Forms::PictureBoxSizeMode::StretchImage;

this->pictureCartridge->TabIndex = 26;

this->pictureCartridge->TabStop = false;

//

// pictureLaserPaperTray

//

this->pictureLaserPaperTray->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureLaserPaperTray.Image")));

this->pictureLaserPaperTray->Location = System::Drawing::Point(8, 172);

this->pictureLaserPaperTray->Margin = System::Windows::Forms::Padding(4);

this->pictureLaserPaperTray->Name = L"pictureLaserPaperTray";

this->pictureLaserPaperTray->Size = System::Drawing::Size(212, 180);

this->pictureLaserPaperTray->SizeMode = System::Windows::Forms::PictureBoxSizeMode::AutoSize;

this->pictureLaserPaperTray->TabIndex = 30;

this->pictureLaserPaperTray->TabStop = false;

this->pictureLaserPaperTray->Click += gcnew System::EventHandler(this, &PrinterSimulation::pictureLaserPaperTray\_Click);

//

// internetGroupBox

//

this->internetGroupBox->Controls->Add(this->documentCountLabel);

this->internetGroupBox->Controls->Add(this->label10);

this->internetGroupBox->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->internetGroupBox->Location = System::Drawing::Point(16, 577);

this->internetGroupBox->Margin = System::Windows::Forms::Padding(4);

this->internetGroupBox->Name = L"internetGroupBox";

this->internetGroupBox->Padding = System::Windows::Forms::Padding(4);

this->internetGroupBox->Size = System::Drawing::Size(365, 71);

this->internetGroupBox->TabIndex = 31;

this->internetGroupBox->TabStop = false;

this->internetGroupBox->Text = L" ";

//

// documentCountLabel

//

this->documentCountLabel->AutoSize = true;

this->documentCountLabel->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->documentCountLabel->Location = System::Drawing::Point(307, 33);

this->documentCountLabel->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->documentCountLabel->Name = L"documentCountLabel";

this->documentCountLabel->Size = System::Drawing::Size(18, 20);

this->documentCountLabel->TabIndex = 22;

this->documentCountLabel->Text = L"0";

//

// label10

//

this->label10->AutoSize = true;

this->label10->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label10->Location = System::Drawing::Point(8, 33);

this->label10->Margin = System::Windows::Forms::Padding(4, 0, 4, 0);

this->label10->Name = L"label10";

this->label10->Size = System::Drawing::Size(220, 20);

this->label10->TabIndex = 23;

this->label10->Text = L"Количество документов:";

//

// simulationGroupBox

//

this->simulationGroupBox->BackColor = System::Drawing::SystemColors::ButtonHighlight;

this->simulationGroupBox->Controls->Add(this->inkjetPrinterIllustrationGroupBox);

this->simulationGroupBox->Controls->Add(this->internetAdapterIllustrationGroupBox);

this->simulationGroupBox->Controls->Add(this->laserPrinterIllustrationGroupBox);

this->simulationGroupBox->Controls->Add(this->computerIllustrationGroupBox);

this->simulationGroupBox->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F, System::Drawing::FontStyle::Regular,

System::Drawing::GraphicsUnit::Point, static\_cast<System::Byte>(204)));

this->simulationGroupBox->Location = System::Drawing::Point(389, 15);

this->simulationGroupBox->Margin = System::Windows::Forms::Padding(4);

this->simulationGroupBox->Name = L"simulationGroupBox";

this->simulationGroupBox->Padding = System::Windows::Forms::Padding(4);

this->simulationGroupBox->Size = System::Drawing::Size(2133, 1252);

this->simulationGroupBox->TabIndex = 32;

this->simulationGroupBox->TabStop = false;

this->simulationGroupBox->Text = L" ";

//

// inkjetPrinterIllustrationGroupBox

//

this->inkjetPrinterIllustrationGroupBox->Controls->Add(this->pictureCyanCartridge);

this->inkjetPrinterIllustrationGroupBox->Controls->Add(this->pictureRefillInkjetPaperTray);

this->inkjetPrinterIllustrationGroupBox->Controls->Add(this->pictureMagentaCartridge);

this->inkjetPrinterIllustrationGroupBox->Controls->Add(this->pictureYellowCartridge);

this->inkjetPrinterIllustrationGroupBox->Controls->Add(this->pictureBlackCartridge);

this->inkjetPrinterIllustrationGroupBox->Controls->Add(this->pictureRefillBlackCartridge);

this->inkjetPrinterIllustrationGroupBox->Controls->Add(this->pictureRefillYellowCartridge);

this->inkjetPrinterIllustrationGroupBox->Controls->Add(this->pictureRefillCyanCartridge);

this->inkjetPrinterIllustrationGroupBox->Controls->Add(this->pictureRefillMagentaCartridge);

this->inkjetPrinterIllustrationGroupBox->Controls->Add(this->pictureInkjetPrinter);

this->inkjetPrinterIllustrationGroupBox->Controls->Add(this->pictureInkjetPaperTray);

this->inkjetPrinterIllustrationGroupBox->Location = System::Drawing::Point(456, 26);

this->inkjetPrinterIllustrationGroupBox->Margin = System::Windows::Forms::Padding(4);

this->inkjetPrinterIllustrationGroupBox->Name = L"inkjetPrinterIllustrationGroupBox";

this->inkjetPrinterIllustrationGroupBox->Padding = System::Windows::Forms::Padding(4);

this->inkjetPrinterIllustrationGroupBox->Size = System::Drawing::Size(1033, 442);

this->inkjetPrinterIllustrationGroupBox->TabIndex = 42;

this->inkjetPrinterIllustrationGroupBox->TabStop = false;

this->inkjetPrinterIllustrationGroupBox->Text = L"Струйный принтер";

//

// pictureCyanCartridge

//

this->pictureCyanCartridge->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureCyanCartridge.Image")));

this->pictureCyanCartridge->ImageLocation = L"";

this->pictureCyanCartridge->Location = System::Drawing::Point(589, 276);

this->pictureCyanCartridge->Margin = System::Windows::Forms::Padding(4);

this->pictureCyanCartridge->Name = L"pictureCyanCartridge";

this->pictureCyanCartridge->Size = System::Drawing::Size(92, 128);

this->pictureCyanCartridge->SizeMode = System::Windows::Forms::PictureBoxSizeMode::StretchImage;

this->pictureCyanCartridge->TabIndex = 27;

this->pictureCyanCartridge->TabStop = false;

//

// pictureRefillInkjetPaperTray

//

this->pictureRefillInkjetPaperTray->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureRefillInkjetPaperTray.Image")));

this->pictureRefillInkjetPaperTray->Location = System::Drawing::Point(17, 27);

this->pictureRefillInkjetPaperTray->Margin = System::Windows::Forms::Padding(4);

this->pictureRefillInkjetPaperTray->Name = L"pictureRefillInkjetPaperTray";

this->pictureRefillInkjetPaperTray->Size = System::Drawing::Size(212, 180);

this->pictureRefillInkjetPaperTray->SizeMode = System::Windows::Forms::PictureBoxSizeMode::AutoSize;

this->pictureRefillInkjetPaperTray->TabIndex = 31;

this->pictureRefillInkjetPaperTray->TabStop = false;

this->pictureRefillInkjetPaperTray->Visible = false;

//

// pictureMagentaCartridge

//

this->pictureMagentaCartridge->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureMagentaCartridge.Image")));

this->pictureMagentaCartridge->ImageLocation = L"";

this->pictureMagentaCartridge->Location = System::Drawing::Point(699, 276);

this->pictureMagentaCartridge->Margin = System::Windows::Forms::Padding(4);

this->pictureMagentaCartridge->Name = L"pictureMagentaCartridge";

this->pictureMagentaCartridge->Size = System::Drawing::Size(93, 128);

this->pictureMagentaCartridge->SizeMode = System::Windows::Forms::PictureBoxSizeMode::StretchImage;

this->pictureMagentaCartridge->TabIndex = 28;

this->pictureMagentaCartridge->TabStop = false;

this->pictureMagentaCartridge->Click += gcnew System::EventHandler(this, &PrinterSimulation::pictureMagentaCartridge\_Click);

//

// pictureYellowCartridge

//

this->pictureYellowCartridge->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureYellowCartridge.Image")));

this->pictureYellowCartridge->ImageLocation = L"";

this->pictureYellowCartridge->Location = System::Drawing::Point(800, 276);

this->pictureYellowCartridge->Margin = System::Windows::Forms::Padding(4);

this->pictureYellowCartridge->Name = L"pictureYellowCartridge";

this->pictureYellowCartridge->Size = System::Drawing::Size(91, 128);

this->pictureYellowCartridge->SizeMode = System::Windows::Forms::PictureBoxSizeMode::StretchImage;

this->pictureYellowCartridge->TabIndex = 28;

this->pictureYellowCartridge->TabStop = false;

//

// pictureBlackCartridge

//

this->pictureBlackCartridge->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureBlackCartridge.Image")));

this->pictureBlackCartridge->ImageLocation = L"";

this->pictureBlackCartridge->Location = System::Drawing::Point(910, 276);

this->pictureBlackCartridge->Margin = System::Windows::Forms::Padding(4);

this->pictureBlackCartridge->Name = L"pictureBlackCartridge";

this->pictureBlackCartridge->Size = System::Drawing::Size(93, 128);

this->pictureBlackCartridge->SizeMode = System::Windows::Forms::PictureBoxSizeMode::StretchImage;

this->pictureBlackCartridge->TabIndex = 28;

this->pictureBlackCartridge->TabStop = false;

//

// pictureRefillBlackCartridge

//

this->pictureRefillBlackCartridge->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureRefillBlackCartridge.Image")));

this->pictureRefillBlackCartridge->ImageLocation = L"";

this->pictureRefillBlackCartridge->Location = System::Drawing::Point(939, 14);

this->pictureRefillBlackCartridge->Margin = System::Windows::Forms::Padding(4);

this->pictureRefillBlackCartridge->Name = L"pictureRefillBlackCartridge";

this->pictureRefillBlackCartridge->Size = System::Drawing::Size(36, 222);

this->pictureRefillBlackCartridge->SizeMode = System::Windows::Forms::PictureBoxSizeMode::AutoSize;

this->pictureRefillBlackCartridge->TabIndex = 36;

this->pictureRefillBlackCartridge->TabStop = false;

this->pictureRefillBlackCartridge->Visible = false;

//

// pictureRefillYellowCartridge

//

this->pictureRefillYellowCartridge->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureRefillYellowCartridge.Image")));

this->pictureRefillYellowCartridge->ImageLocation = L"";

this->pictureRefillYellowCartridge->Location = System::Drawing::Point(826, 14);

this->pictureRefillYellowCartridge->Margin = System::Windows::Forms::Padding(4);

this->pictureRefillYellowCartridge->Name = L"pictureRefillYellowCartridge";

this->pictureRefillYellowCartridge->Size = System::Drawing::Size(36, 222);

this->pictureRefillYellowCartridge->SizeMode = System::Windows::Forms::PictureBoxSizeMode::AutoSize;

this->pictureRefillYellowCartridge->TabIndex = 35;

this->pictureRefillYellowCartridge->TabStop = false;

this->pictureRefillYellowCartridge->Visible = false;

//

// pictureRefillCyanCartridge

//

this->pictureRefillCyanCartridge->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureRefillCyanCartridge.Image")));

this->pictureRefillCyanCartridge->ImageLocation = L"";

this->pictureRefillCyanCartridge->Location = System::Drawing::Point(615, 14);

this->pictureRefillCyanCartridge->Margin = System::Windows::Forms::Padding(4);

this->pictureRefillCyanCartridge->Name = L"pictureRefillCyanCartridge";

this->pictureRefillCyanCartridge->Size = System::Drawing::Size(36, 224);

this->pictureRefillCyanCartridge->SizeMode = System::Windows::Forms::PictureBoxSizeMode::AutoSize;

this->pictureRefillCyanCartridge->TabIndex = 33;

this->pictureRefillCyanCartridge->TabStop = false;

this->pictureRefillCyanCartridge->Visible = false;

//

// pictureRefillMagentaCartridge

//

this->pictureRefillMagentaCartridge->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureRefillMagentaCartridge.Image")));

this->pictureRefillMagentaCartridge->ImageLocation = L"";

this->pictureRefillMagentaCartridge->Location = System::Drawing::Point(727, 14);

this->pictureRefillMagentaCartridge->Margin = System::Windows::Forms::Padding(4);

this->pictureRefillMagentaCartridge->Name = L"pictureRefillMagentaCartridge";

this->pictureRefillMagentaCartridge->Size = System::Drawing::Size(36, 222);

this->pictureRefillMagentaCartridge->SizeMode = System::Windows::Forms::PictureBoxSizeMode::AutoSize;

this->pictureRefillMagentaCartridge->TabIndex = 34;

this->pictureRefillMagentaCartridge->TabStop = false;

this->pictureRefillMagentaCartridge->Visible = false;

//

// pictureInkjetPrinter

//

this->pictureInkjetPrinter->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureInkjetPrinter.Image")));

this->pictureInkjetPrinter->Location = System::Drawing::Point(292, 135);

this->pictureInkjetPrinter->Margin = System::Windows::Forms::Padding(4);

this->pictureInkjetPrinter->Name = L"pictureInkjetPrinter";

this->pictureInkjetPrinter->Size = System::Drawing::Size(212, 180);

this->pictureInkjetPrinter->SizeMode = System::Windows::Forms::PictureBoxSizeMode::AutoSize;

this->pictureInkjetPrinter->TabIndex = 6;

this->pictureInkjetPrinter->TabStop = false;

//

// pictureInkjetPaperTray

//

this->pictureInkjetPaperTray->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureInkjetPaperTray.Image")));

this->pictureInkjetPaperTray->Location = System::Drawing::Point(17, 174);

this->pictureInkjetPaperTray->Margin = System::Windows::Forms::Padding(4);

this->pictureInkjetPaperTray->Name = L"pictureInkjetPaperTray";

this->pictureInkjetPaperTray->Size = System::Drawing::Size(212, 180);

this->pictureInkjetPaperTray->SizeMode = System::Windows::Forms::PictureBoxSizeMode::AutoSize;

this->pictureInkjetPaperTray->TabIndex = 29;

this->pictureInkjetPaperTray->TabStop = false;

//

// internetAdapterIllustrationGroupBox

//

this->internetAdapterIllustrationGroupBox->Controls->Add(this->pictureInternetAdapter);

this->internetAdapterIllustrationGroupBox->Location = System::Drawing::Point(31, 503);

this->internetAdapterIllustrationGroupBox->Margin = System::Windows::Forms::Padding(4);

this->internetAdapterIllustrationGroupBox->Name = L"internetAdapterIllustrationGroupBox";

this->internetAdapterIllustrationGroupBox->Padding = System::Windows::Forms::Padding(4);

this->internetAdapterIllustrationGroupBox->Size = System::Drawing::Size(281, 313);

this->internetAdapterIllustrationGroupBox->TabIndex = 43;

this->internetAdapterIllustrationGroupBox->TabStop = false;

this->internetAdapterIllustrationGroupBox->Text = L"Интернет-адаптер";

//

// pictureInternetAdapter

//

this->pictureInternetAdapter->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureInternetAdapter.Image")));

this->pictureInternetAdapter->Location = System::Drawing::Point(34, 84);

this->pictureInternetAdapter->Margin = System::Windows::Forms::Padding(4);

this->pictureInternetAdapter->Name = L"pictureInternetAdapter";

this->pictureInternetAdapter->Size = System::Drawing::Size(212, 180);

this->pictureInternetAdapter->SizeMode = System::Windows::Forms::PictureBoxSizeMode::StretchImage;

this->pictureInternetAdapter->TabIndex = 4;

this->pictureInternetAdapter->TabStop = false;

//

// laserPrinterIllustrationGroupBox

//

this->laserPrinterIllustrationGroupBox->Controls->Add(this->pictureRefillLaserPaperTray);

this->laserPrinterIllustrationGroupBox->Controls->Add(this->pictureRefillCartridge);

this->laserPrinterIllustrationGroupBox->Controls->Add(this->pictureCartridge);

this->laserPrinterIllustrationGroupBox->Controls->Add(this->pictureLaserPrinter);

this->laserPrinterIllustrationGroupBox->Controls->Add(this->pictureLaserPaperTray);

this->laserPrinterIllustrationGroupBox->Location = System::Drawing::Point(456, 503);

this->laserPrinterIllustrationGroupBox->Margin = System::Windows::Forms::Padding(4);

this->laserPrinterIllustrationGroupBox->Name = L"laserPrinterIllustrationGroupBox";

this->laserPrinterIllustrationGroupBox->Padding = System::Windows::Forms::Padding(4);

this->laserPrinterIllustrationGroupBox->Size = System::Drawing::Size(854, 410);

this->laserPrinterIllustrationGroupBox->TabIndex = 41;

this->laserPrinterIllustrationGroupBox->TabStop = false;

this->laserPrinterIllustrationGroupBox->Text = L"Лазерный принтер";

//

// pictureRefillLaserPaperTray

//

this->pictureRefillLaserPaperTray->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureRefillLaserPaperTray.Image")));

this->pictureRefillLaserPaperTray->Location = System::Drawing::Point(8, 27);

this->pictureRefillLaserPaperTray->Margin = System::Windows::Forms::Padding(4);

this->pictureRefillLaserPaperTray->Name = L"pictureRefillLaserPaperTray";

this->pictureRefillLaserPaperTray->Size = System::Drawing::Size(212, 180);

this->pictureRefillLaserPaperTray->SizeMode = System::Windows::Forms::PictureBoxSizeMode::AutoSize;

this->pictureRefillLaserPaperTray->TabIndex = 32;

this->pictureRefillLaserPaperTray->TabStop = false;

this->pictureRefillLaserPaperTray->Visible = false;

//

// pictureRefillCartridge

//

this->pictureRefillCartridge->Image = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureRefillCartridge.Image")));

this->pictureRefillCartridge->ImageLocation = L"";

this->pictureRefillCartridge->Location = System::Drawing::Point(631, 27);

this->pictureRefillCartridge->Margin = System::Windows::Forms::Padding(4);

this->pictureRefillCartridge->Name = L"pictureRefillCartridge";

this->pictureRefillCartridge->Size = System::Drawing::Size(36, 222);

this->pictureRefillCartridge->SizeMode = System::Windows::Forms::PictureBoxSizeMode::AutoSize;

this->pictureRefillCartridge->TabIndex = 37;

this->pictureRefillCartridge->TabStop = false;

this->pictureRefillCartridge->Visible = false;

//

// computerIllustrationGroupBox

//

this->computerIllustrationGroupBox->Controls->Add(this->pictureComputer);

this->computerIllustrationGroupBox->Location = System::Drawing::Point(31, 28);

this->computerIllustrationGroupBox->Margin = System::Windows::Forms::Padding(4);

this->computerIllustrationGroupBox->Name = L"computerIllustrationGroupBox";

this->computerIllustrationGroupBox->Padding = System::Windows::Forms::Padding(4);

this->computerIllustrationGroupBox->Size = System::Drawing::Size(395, 313);

this->computerIllustrationGroupBox->TabIndex = 42;

this->computerIllustrationGroupBox->TabStop = false;

this->computerIllustrationGroupBox->Text = L"Компьютер";

//

// PrinterSimulation

//

this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(1924, 1055);

this->Controls->Add(this->simulationGroupBox);

this->Controls->Add(this->internetGroupBox);

this->Controls->Add(this->inkjetPrinterGroupBox);

this->Controls->Add(this->laserPrinterGroupBox);

this->Controls->Add(this->controlPanelGroupBox);

this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedSingle;

this->Margin = System::Windows::Forms::Padding(4);

this->MaximizeBox = false;

this->Name = L"PrinterSimulation";

this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;

this->Text = L"Модель работы принтера при его взаимодействии с компьютером";

this->WindowState = System::Windows::Forms::FormWindowState::Maximized;

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureComputer))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureLaserPrinter))->EndInit();

this->controlPanelGroupBox->ResumeLayout(false);

this->controlPanelGroupBox->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->simulationSpeedTrackBar))->EndInit();

this->laserPrinterGroupBox->ResumeLayout(false);

this->laserPrinterGroupBox->PerformLayout();

this->inkjetPrinterGroupBox->ResumeLayout(false);

this->inkjetPrinterGroupBox->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureCartridge))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureLaserPaperTray))->EndInit();

this->internetGroupBox->ResumeLayout(false);

this->internetGroupBox->PerformLayout();

this->simulationGroupBox->ResumeLayout(false);

this->inkjetPrinterIllustrationGroupBox->ResumeLayout(false);

this->inkjetPrinterIllustrationGroupBox->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureCyanCartridge))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureRefillInkjetPaperTray))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureMagentaCartridge))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureYellowCartridge))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBlackCartridge))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureRefillBlackCartridge))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureRefillYellowCartridge))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureRefillCyanCartridge))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureRefillMagentaCartridge))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureInkjetPrinter))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureInkjetPaperTray))->EndInit();

this->internetAdapterIllustrationGroupBox->ResumeLayout(false);

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureInternetAdapter))->EndInit();

this->laserPrinterIllustrationGroupBox->ResumeLayout(false);

this->laserPrinterIllustrationGroupBox->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureRefillLaserPaperTray))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureRefillCartridge))->EndInit();

this->computerIllustrationGroupBox->ResumeLayout(false);

this->computerIllustrationGroupBox->PerformLayout();

this->ResumeLayout(false);

}

#pragma endregion

private:

void ShowPrinterPrinting(Printer^ printer)

{

String^ printerType;

String^ printerTypeSecond;

if (printer->GetType() == Printer::typeid)

{

printerType = "LaserPrinter";

printerTypeSecond = "InkjetPrinter";

}

else

{

printerType = "InkjetPrinter";

printerTypeSecond = "LaserPrinter";

}

int delay = 2000 / \_programSpeed;

if (printerType == "LaserPrinter")

{

\_dicLaserPrinter->Change(printerType + "/Printing.png", delay);

pictureInkjetPrinter->ImageLocation = printerTypeSecond + "/Printer.png";

}

else

{

\_dicInkjetPrinter->Change(printerType + "/Printing.png", delay);

pictureLaserPrinter->ImageLocation = printerTypeSecond + "/Printer.png";

}

}

void ShowPrinterOutOfPaper(Printer^ printer)

{

String^ printerType = printer->GetType() == Printer::typeid ? "LaserPrinter" : "InkjetPrinter";

String^ imagePath = printerType + "/PrinterOutOfPaper.png";

if (printerType == "LaserPrinter")

pictureLaserPrinter->ImageLocation = imagePath;

else

pictureInkjetPrinter->ImageLocation = imagePath;

}

void ShowPrinterMaintance(Printer^ printer, int index)

{

String^ printerType = printer->GetType() == Printer::typeid ? "LaserPrinter" : "InkjetPrinter";

String^ imagePath = printerType + "/PrinterMaintance.png";

if (printerType == "LaserPrinter")

pictureLaserPrinter->ImageLocation = imagePath;

else

pictureInkjetPrinter->ImageLocation = imagePath;

}

String^ GetCartridgeColor(int index)

{

array<String^>^ colors = { "Cyan", "Magenta", "Yellow", "Black" };

if (index >= 0 && index < colors->Length)

return colors[index];

else

return "";

}

void ShowCartridgeFillLevel(PictureBox^ pictureBox, int cartridgeFillLevel, int index)

{

array<String^>^ fillRanges = { "0", "0-25", "25-50", "50-75", "75-100", "100" };

array<int>^ thresholds = { 0, 1, 25, 50, 75, 100 };

String^ fillRange = "0";

for (int i = thresholds->Length - 1; i >= 0; i--)

{

if (cartridgeFillLevel >= thresholds[i])

{

fillRange = fillRanges[i];

break;

}

}

String^ cartridgeColor = GetCartridgeColor(index);

pictureBox->ImageLocation = "Cartridges/" + cartridgeColor + "Cartridge\_" + fillRange + ".png";

}

void ShowPaperTrayFillLevel(PictureBox^ pictureBox, int paperTrayFillLevel)

{

array<int>^ thresholds = { 0, 4, 16, 28, 40, 52, 64, 76, 88, 100 };

int fillLevel = 0; // default value

for (int i = thresholds->Length - 1; i >= 0; i--)

{

if (paperTrayFillLevel >= thresholds[i])

{

fillLevel = thresholds[i];

break;

}

}

pictureBox->ImageLocation = "PaperTray/PaperTray\_" + fillLevel.ToString() + ".png";

}

void ShowRefillPaperTray(Printer^ printer)

{

int delay = 800 / \_programSpeed;

if (printer->GetType() == Printer::typeid)

\_dvcRefillLaserPaperTray->Change(true, delay);

else

\_dvcRefillInkjetPaperTray->Change(true, delay);

}

void ShowRefillCartridge(int index) //

{

String^ color = GetCartridgeColor(index);

int delay = 800 / \_programSpeed;

if (color == "Cyan")

\_dvcRefillCyanCartridge->Change(true, delay);

else if (color == "Magenta")

\_dvcRefillMagentaCartridge->Change(true, delay);

else if (color == "Yellow")

\_dvcRefillYellowCartridge->Change(true, delay);

else if (color == "Black")

\_dvcRefillBlackCartridge->Change(true, delay);

else

\_dvcRefillCartridge->Change(true, delay);

}

int CalculateCartridgeFill(Printer^ printer, int cartridgeIndex) // возвращает текущий уровень чернил

{

int cartridgeFillLevel = printer->GetCartridgeFillLevel(cartridgeIndex);

int cartridgeCapacity = printer->CartridgeCapacity;

return (int)Math::Round((double)cartridgeFillLevel / cartridgeCapacity \* 100);

}

int CalculatePaperTrayFill(Printer^ printer)

{

int paperTrayCount = printer->PaperTrayCount;

int paperTrayCapacity = printer->PaperTrayCapacity;

return (int)Math::Round((double)paperTrayCount / paperTrayCapacity \* 100);

}

void SubscribeToPrinterEvents(Printer^ printer)

{

printer->Printing += gcnew Printer::PrintingEventHandler(this, &PrinterSimulation::ShowPrinterPrinting);

printer->PaperTrayEmpty += gcnew Printer::PaperTrayEmptyEventHandler(this, &PrinterSimulation::ShowPrinterOutOfPaper);

printer->CartridgeEmpty += gcnew Printer::CartridgeEmptyEventHandler(this, &PrinterSimulation::ShowPrinterMaintance);

}

void SubscribeToHumanEvents(Human^ human)

{

human->CartridgeRefilled += gcnew Human::CartridgeRefilledEventHandler(this, &PrinterSimulation::ShowRefillCartridge);

human->PaperTrayRefilled += gcnew Human::PaperTrayRefilledEventHandler(this, &PrinterSimulation::ShowRefillPaperTray);

}

void UnsubscribeFromPrinterEvents(Printer^ printer)

{

printer->Printing -= gcnew Printer::PrintingEventHandler(this, &PrinterSimulation::ShowPrinterPrinting);

printer->PaperTrayEmpty -= gcnew Printer::PaperTrayEmptyEventHandler(this, &PrinterSimulation::ShowPrinterOutOfPaper);

printer->CartridgeEmpty -= gcnew Printer::CartridgeEmptyEventHandler(this, &PrinterSimulation::ShowPrinterMaintance);

}

void UnsubcribeFromHumanEvents(Human^ human)

{

human->CartridgeRefilled -= gcnew Human::CartridgeRefilledEventHandler(this, &PrinterSimulation::ShowRefillCartridge);

human->PaperTrayRefilled -= gcnew Human::PaperTrayRefilledEventHandler(this, &PrinterSimulation::ShowRefillPaperTray);

}

private: System::Void TurnOnDevices\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

\_computer->AdapterIsEnabled = TurnOnDevices->Checked;

if (\_computer->AdapterIsEnabled)

{

\_computer->SubscribeToEvents(\_laserPrinter);

\_computer->SubscribeToEvents(\_inkjetPrinter);

SubscribeToPrinterEvents(\_laserPrinter);

SubscribeToPrinterEvents(\_inkjetPrinter);

SubscribeToHumanEvents(\_human);

pictureComputer->ImageLocation = L"Computer/Computer.png";

}

else

{

UnsubcribeFromHumanEvents(\_human);

UnsubscribeFromPrinterEvents(\_laserPrinter);

UnsubscribeFromPrinterEvents(\_inkjetPrinter);

\_computer->UnsubscribeFromEvents(\_laserPrinter);

\_computer->UnsubscribeFromEvents(\_inkjetPrinter);

pictureComputer->ImageLocation = L"Computer/ComputerNoNetwork.png";

\_dicLaserPrinter->Change(L"LaserPrinter/Printer.png", 1000 / \_programSpeed);

\_dicInkjetPrinter->Change(L"InkjetPrinter/Printer.png", 1000 / \_programSpeed);

}

}

private: System::Void fast\_Click(System::Object^ sender, System::EventArgs^ e) {

if ((\_programSpeed >= 1) && (\_programSpeed < 5))

\_programSpeed++;

simulationSpeedTrackBar->Value = \_programSpeed;

\_computer->ChangeTimerInterval(\_programSpeed);

\_internet->ChangeTimerInterval(\_programSpeed);

}

private: System::Void slow\_Click(System::Object^ sender, System::EventArgs^ e) {

if ((\_programSpeed > 1) && (\_programSpeed <= 5))

\_programSpeed--;

simulationSpeedTrackBar->Value = \_programSpeed;

\_computer->ChangeTimerInterval(\_programSpeed);

\_internet->ChangeTimerInterval(\_programSpeed);

}

private: System::Void generateDocumentSwitch\_CheckedChanged(System::Object^ sender, System::EventArgs^ e)

{

if (generateDocumentSwitch->Checked)

{

pictureInternetAdapter->ImageLocation = L"InternetAdapter/Enable.png";

\_internet->StartGenerating();

}

else

{

pictureInternetAdapter->ImageLocation = L"InternetAdapter/Disable.png";

\_internet->StopGenerating();

}

}

private: System::Void computerTimer\_Tick(System::Object^ sender, System::EventArgs^ e)

{

if (\_computer->AdapterIsEnabled)

{

pictureComputer->ImageLocation = "Computer/ComputerInAction.png";

\_dicComputerOnAction->Change("Computer/Computer.png", 1000 / \_programSpeed);

}

if (\_internet->GetCurrentDocumentCount() == 0 && (!generateDocumentSwitch->Checked))

{

\_dicComputerOnAction->Change("Computer/ComputerNoNetwork.png", 1000 / \_programSpeed);

\_dicLaserPrinter->Change(L"LaserPrinter/Printer.png", 1000 / \_programSpeed);

\_dicInkjetPrinter->Change(L"InkjetPrinter/Printer.png", 1000 / \_programSpeed);

}

}

private: System::Void timer\_Tick(System::Object^ sender, System::EventArgs^ e)

{

int cartridgeFillLevel = CalculateCartridgeFill(\_laserPrinter, -1);

cartridgeFillLevelLabel->Text = Convert::ToString(cartridgeFillLevel);

ShowCartridgeFillLevel(pictureCartridge, cartridgeFillLevel, -1);

int cyanCartridgeFillLevel = CalculateCartridgeFill(\_inkjetPrinter, 0);

cyanCartridgeFillLevelLabel->Text = Convert::ToString(cyanCartridgeFillLevel);

ShowCartridgeFillLevel(pictureCyanCartridge, cyanCartridgeFillLevel, 0);

int magentaCartridgeFillLevel = CalculateCartridgeFill(\_inkjetPrinter, 1);

magentaCartridgeFillLevelLabel->Text = Convert::ToString(magentaCartridgeFillLevel);

ShowCartridgeFillLevel(pictureMagentaCartridge, magentaCartridgeFillLevel, 1);

int yellowCartridgeFillLevel = CalculateCartridgeFill(\_inkjetPrinter, 2);

yellowCartridgeFillLevelLabel->Text = Convert::ToString(yellowCartridgeFillLevel);

ShowCartridgeFillLevel(pictureYellowCartridge, yellowCartridgeFillLevel, 2);

int blackCartridgeFillLevel = CalculateCartridgeFill(\_inkjetPrinter, 3);

blackCartridgeFillLevelLabel->Text = Convert::ToString(blackCartridgeFillLevel);

ShowCartridgeFillLevel(pictureBlackCartridge, blackCartridgeFillLevel, 3);

laserPaperCountLabel->Text = Convert::ToString(\_laserPrinter->PaperTrayCount);

int laserPaperTrayFillLevel = CalculatePaperTrayFill(\_laserPrinter);

ShowPaperTrayFillLevel(pictureLaserPaperTray, laserPaperTrayFillLevel);

inkjetPaperCountLabel->Text = Convert::ToString(\_inkjetPrinter->PaperTrayCount);

int inkjetPaperTrayFillLevel = CalculatePaperTrayFill(\_inkjetPrinter);

ShowPaperTrayFillLevel(pictureInkjetPaperTray, inkjetPaperTrayFillLevel);

int currentDocumentCount = \_internet->GetCurrentDocumentCount();

documentCountLabel->Text = Convert::ToString(currentDocumentCount);

}

private: System::Void simulationSpeedTrackBar\_ValueChanged(System::Object^ sender, System::EventArgs^ e)

{

\_programSpeed = simulationSpeedTrackBar->Value;

\_computer->ChangeTimerInterval(\_programSpeed);

\_internet->ChangeTimerInterval(\_programSpeed);

}

private: System::Void pictureComputer\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void pictureLaserPaperTray\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void simulationSpeedTrackBar\_Scroll(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void pictureMagentaCartridge\_Click(System::Object^ sender, System::EventArgs^ e) {

}

};

}

Код исходного файла:

#include "PrinterSimulation.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThread]

void Main(array<String^>^ args)

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Simulation::PrinterSimulation form;

Application::Run(% form);

}

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ ПРОГРАММЫ

Программа в исходном состоянии представлена на рисунке 72.

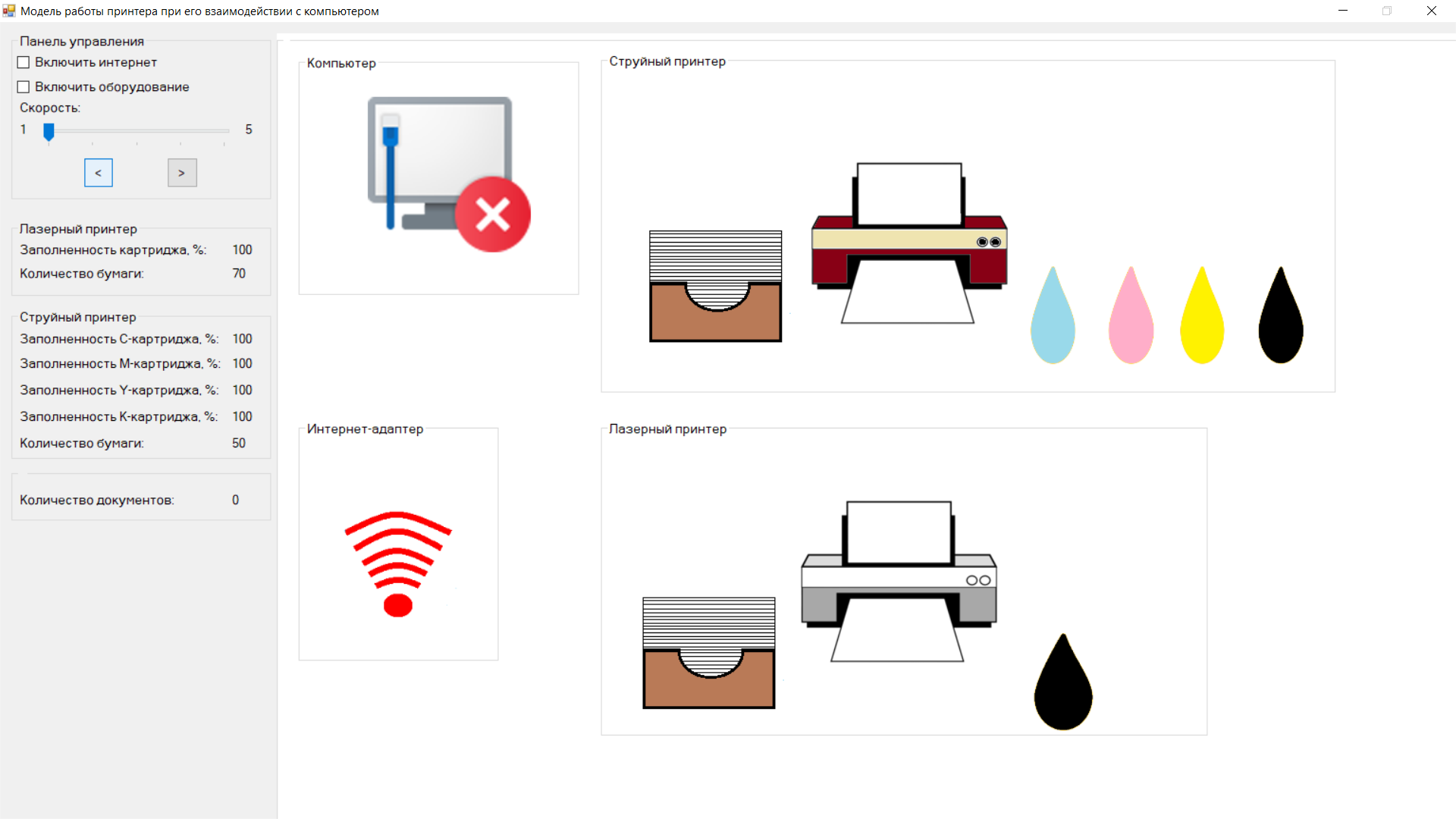


Рисунок 72 — Программа в исходном состоянии

После отметки поля «Включить интернет» начинают создаваться документы, увеличивается счётчик количества документов, что представлено на рисунке 73.

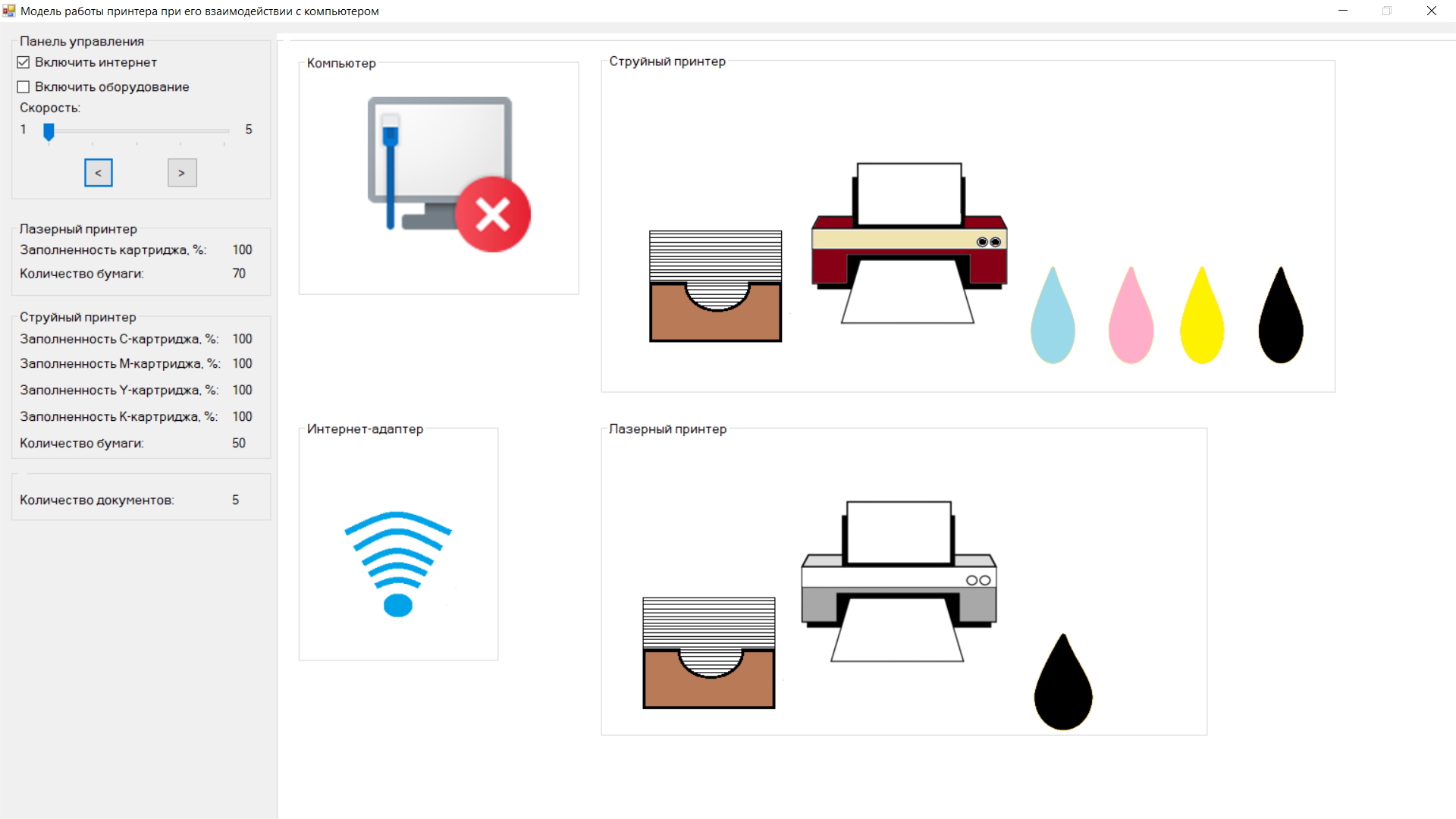


Рисунок 73 — Отмеченное поле «Создание документов»

После отметки поля «Включить оборудование» меняется изображение компьютера и он начинает запрашивать документы, что представлено на рисунках 74-75.

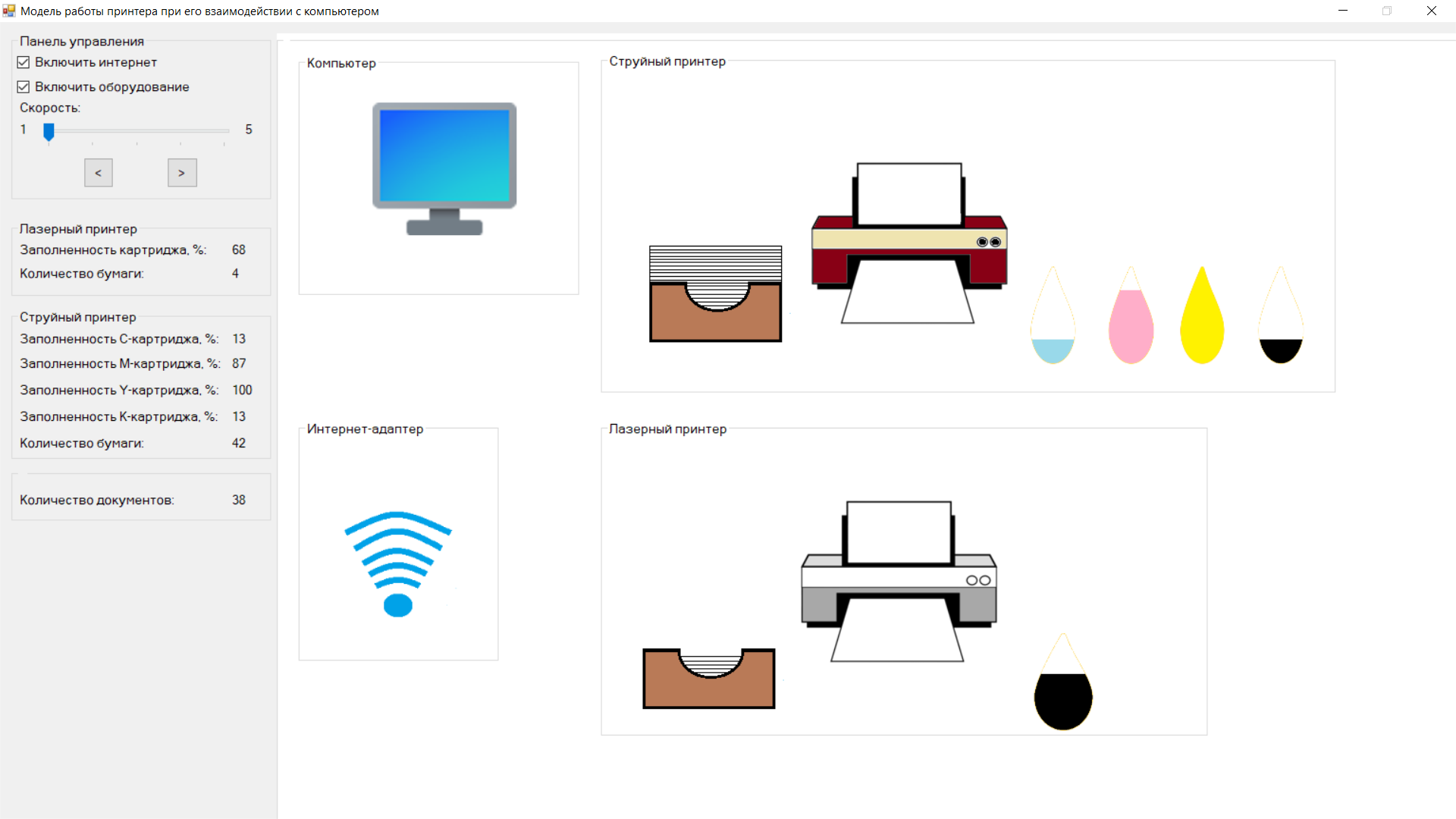


Рисунок 74— Отмеченное поле «Включить оборудование»

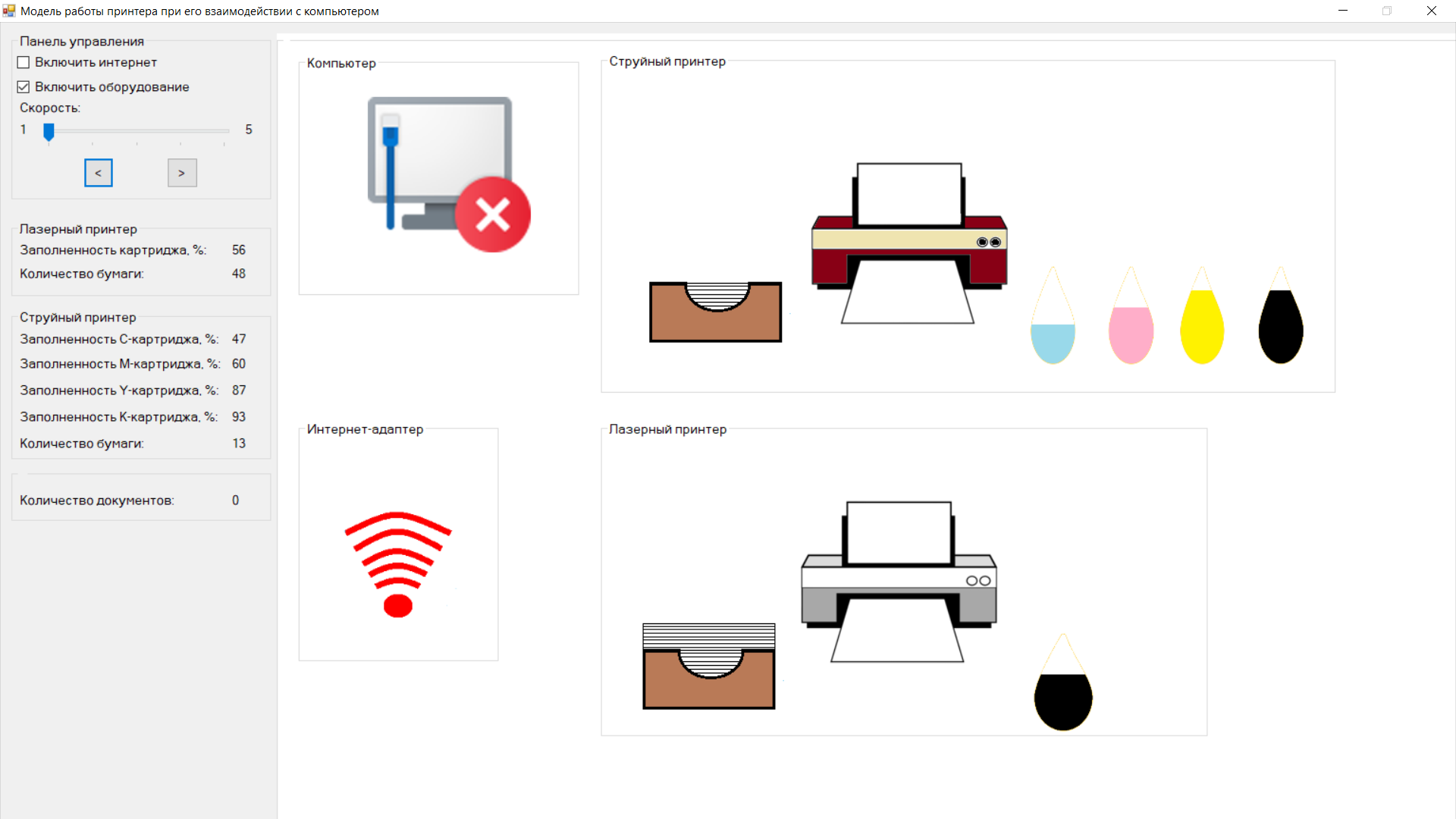


Рисунок 75 — Запрос документов

Когда отмечены оба поля, происходит печать. Возможные события при печати на двух принтерах одинаковы и представлены на рисунках 76-78.

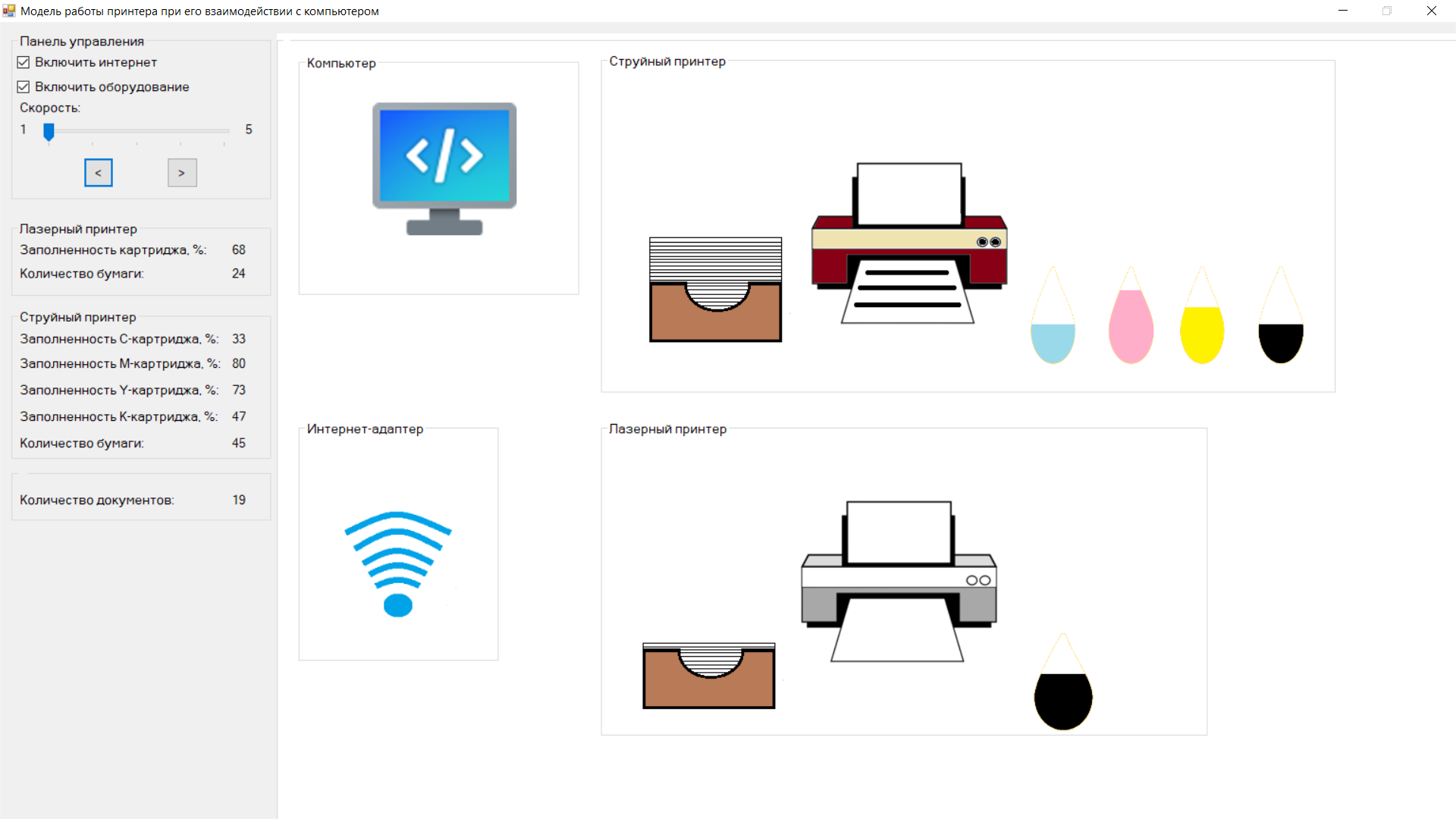


Рисунок 76 — Принтер печатает

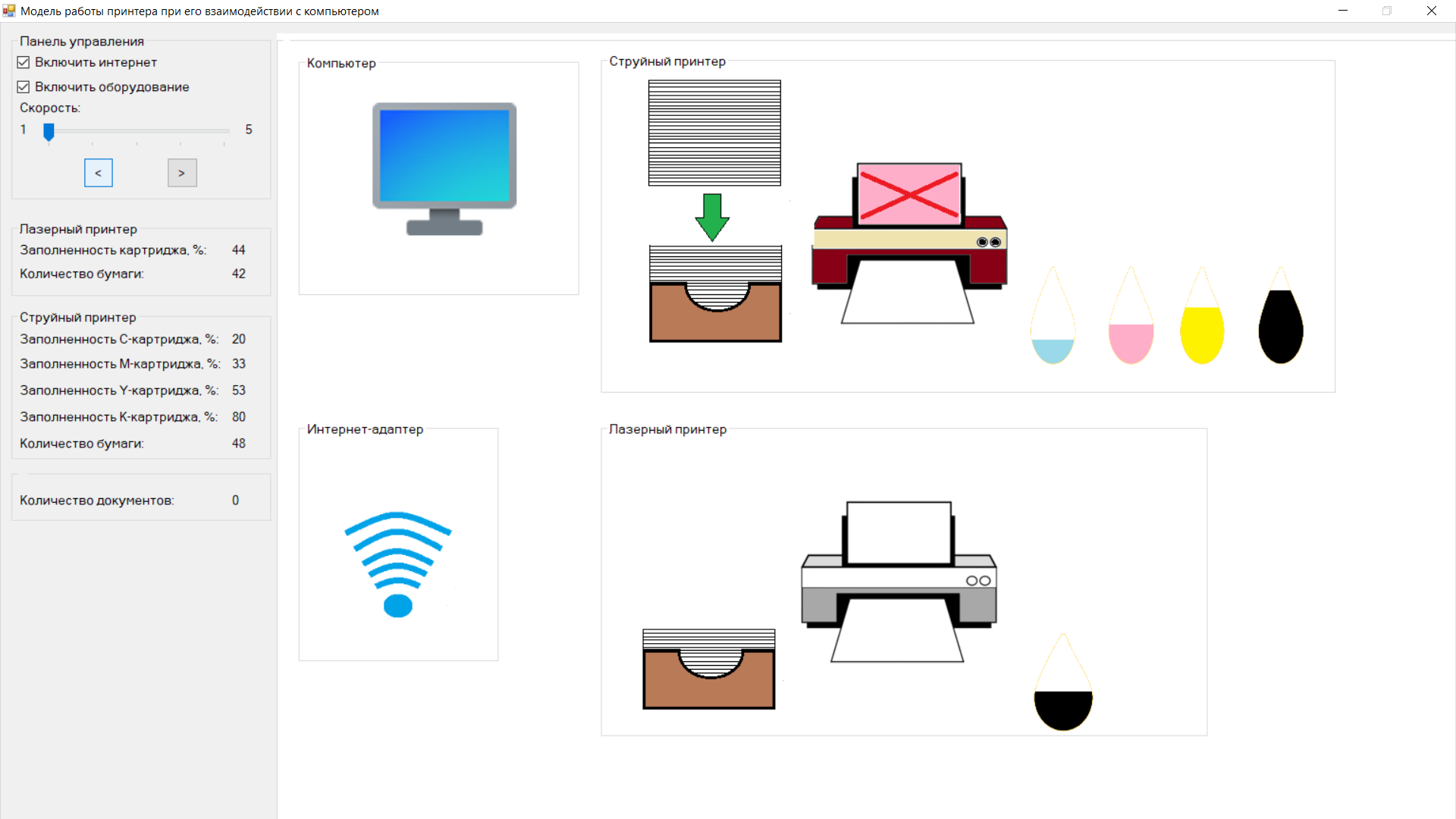


Рисунок 77 — В принтере закончилась бумага

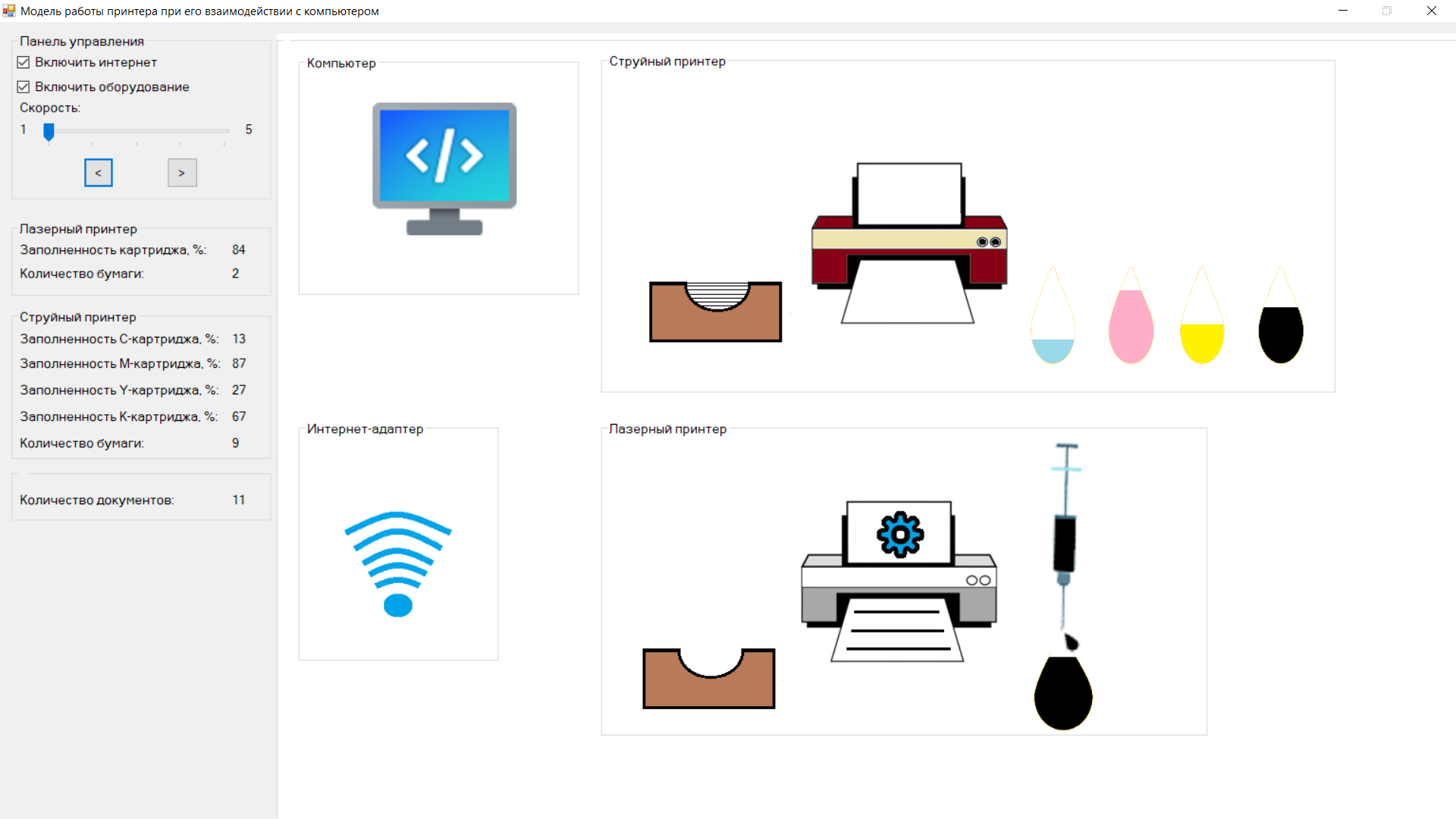


Рисунок 78 — В принтере закончился картридж

На форме также приведены текущие данные и ползунок регулировки скорости симуляции, которые представлены на рисунке 79.

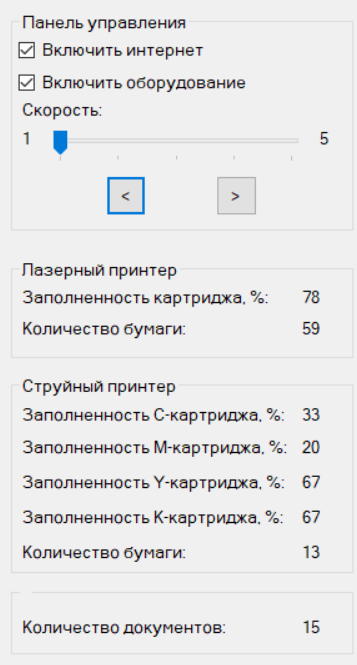


Рисунок 79 — Панель управления симуляции

ВЫВОД

В результате выполнения работы была разработана программа симуляции работы принтера при взаимодействии с компьютером с использованием объектно-ориентированного подхода на языке C++/CLI с использованием Windows Forms.

При разработке программы были применены ключевые принципы объектно-ориентированного программирования, такие как инкапсуляция, наследование и полиморфизм. Классы спроектированы и реализованы с учётом этих принципов, обеспечивая модульность, повторное использование кода и возможность дальнейшего расширения функциональности приложения.

Таким образом, созданное приложение успешно решает поставленную задачу по визуализации модели работы принтера при его взаимодействии с компьютером и демонстрирует практическое применение объектно-ориентированного подхода в программировании на C++/CLI с использованием C++/CLI.

Список использованной литературы

1 ГОСТ 7.32-2017. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления" (введен в действие Приказом Росстандарта от 24.10.2017 N 1494-ст) — М., Стандартинформ, 2017.