Министерство образования и науки Российской Федерации

федеральное государственное бюджетное образовательное

учреждение высшего образования

«Алтайский государственный технический университет

им. И.И. Ползунова»

Факультет \_\_информационных технологий\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра \_\_\_прикладной математики\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Специальность (направление, профиль) \_\_ПИ\_\_\_\_\_\_\_\_\_\_\_\_\_

Курсовой проект

защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.С. Троицкий

(подпись руководителя проекта) (инициалы, фамилия)

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Курсовой проект

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Разработка компьютерной игры \_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_«Управление деревней»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(тема курсового проекта)

Пояснительная записка

по дисциплине \_\_Программирование\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_КП 09.03.04.37.000 ПЗ\_\_\_\_\_\_\_

(обозначение документа)



Студент группы ПИ92 Гулин А.Н.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_21.01.2021

(фамилия, имя, отчество) (подпись) (дата)

Руководитель

проекта доцент, к.ф.-м.н.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.С. Троицкий

(должность, ученое звание) (подпись) (инициалы, фамилия)

БАРНАУЛ 2021

Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Алтайский государственный технический университет

имени И.И. Ползунова»

Факультет информационных технологий

Кафедра «Прикладная математика»

З А Д А Н И Е

на курсовой проект по дисциплине «Программирование»

студенту группы ПИ-92 Гулину Алексею Николаевичу

Тема курсового проекта: «Разработка компьютерной игры “Управление деревней”».

Календарный план работы:

|  |  |  |
| --- | --- | --- |
| № этапа | Содержание этапа | Недели семестра |
| 1 | Получение задания | 1 |
| 2 | Постановка задачи. Работа с документацией | 2 |
| 3 | Проектирование программы | 3-4 |
| 4 | Реализация программы | 5-13 |
| 5 | Оформление пояснительной записки | 14 |
| 6 | Защита курсового проекта | 15-16 |

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Троицкий В.С., доцент, к.ф.-м.н.

подпись

Дата выдачи задания «14» сентября 2020 г.

число месяц год

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Гулин А.Н.

подпись

Содержание

[Введение 2](#_Toc62097022)

[1 Обзор предметной области и постановка задачи 3](#_Toc62097023)

[1.1 Обзор предметной области 3](#_Toc62097024)

[1.2 Постановка задачи 4](#_Toc62097025)

[2 Проектирование 6](#_Toc62097026)

[2.1 Структура данных 6](#_Toc62097027)

[2.2 Алгоритм решения 8](#_Toc62097028)

[3 Реализация 12](#_Toc62097029)

[3.1 Выбор средств реализации 12](#_Toc62097030)

[3.2 Структура программы 13](#_Toc62097031)

[3.3 Описание реализации 17](#_Toc62097032)

[Заключение 25](#_Toc62097033)

[Список использованных источников 27](#_Toc62097034)

[Приложение А Снимки экранных форм пользовательского интерфейса 28](#_Toc62097035)

[Приложение Б Исходный код программы 33](#_Toc62097036)

[Б.1 Исходный код реализации на C++ 33](#_Toc62097037)

[Б.2 Исходный код реализации на C# 90](#_Toc62097038)

[Приложение В Руководство системного администратора 114](#_Toc62097039)

[Приложение Г Руководство пользователя 115](#_Toc62097040)

# Введение

Значительный скачок в развитии вычислительной техники во второй половине XX века оказал заметное влияние на жизнь людей. Наступала эра потоков информации, с которыми неразрывно связаны компьютеры. Они позволяют значительно ускорить и упростить работу по структуризации и анализу данных. Однако в процессе совершенствования технологий людьми было достаточно быстро придумано альтернативное применение ЭВМ – в сфере развлечений. Программное обеспечение для работы с документами и таблицами развивалось параллельно с компьютерными играми, и увеличение вычислительных мощностей сказывалось на возможностях обоих направлений. Игры были разнообразны. От текста в консоли с вариантами выбора, бегающих по экрану пикселей размером с пшеничное зерно до самой настоящей виртуальной реальности с проработанными текстурами, моделями и анимацией движений. Сейчас методология объектно-ориентированного программирования является наиболее популярной и востребованной при разработке программных продуктов: как игр, так и рабочих программ. Изучение теоретической части, подкреплённое практическим опытом, значительно повышает профессиональную подготовку специалистов сферы IT.

Разработка компьютерной игры «Управление деревней» в данном проекте была выбрана не случайно. Игра построена на нескольких простых, но при этом основополагающих механиках: управлении юнитами (игровыми персонажами) и распределении ресурсов (игровых денежных и хозяйственных единиц). Произведённый программный продукт будет соответствовать всем требованиям современных реалий, за исключением красивой графики и анимации. Ведь даже самые высокобюджетные игры построены на простых и повторяющихся действиях: менеджменте, тактическом планировании и многом другом. В дальнейшем, разработанная игра может стать основой для более масштабных проектов. Кроме того, навыки работы с объектно-ориентированными языками программирования будут полезны как при разработке игр, так и при разработке обычных функционально-прикладных программ. Стоит заметить, что данная игра может стать довольно успешной при адаптации на мобильные платформы из-за простого геймплея (процесса игры) и отсутствия сюжета в игре. Игра проста и неспешна, чем может вызвать интерес даже самых непритязательных пользователей, но при этом может предложить увеличивающуюся в процессе игры сложность, дающую вызов более требовательным игрокам.

Цель работы: изучение принципов объектно-ориентированного программирования, получение навыков разработки приложений с графическим интерфейсом на объектно-ориентированных языках, таких как C++ и C#.

Главная задача работы: создание программного продукта на двух разных языках программирования, обладающего понятным интерфейсом и интересными механиками, а также функциями для комфортного пользования, такими как сохранение игры.

# 1 Обзор предметной области и постановка задачи

## 1.1 Обзор предметной области

Процесс создания игры – очень трудоемкое дело, отнимающее огромное количество времени. Чаще всего программисты-одиночки или коллективы единомышленников создают небольшие, так называемые казуальные игры, стоимость которых составляет не более пары десятков долларов. Набив руку на небольших проектах, эти самые коллективы либо перерастают в более мощные игровые студии, либо сотрудники фирмы находят себе хорошую работу в стане сильных   
компаний. [1]

Объектно-ориентированное программирование (ООП) — это технология, основанная на представлении программ в виде совокупности объектов, каждый из которых является реализацией собственного класса, которые в свою очередь образуют иерархию на принципах наследования.   
ООП является наиболее распространённой методологией программирования, включая создание компьютерных игр. Основное достоинство ООП — сокращение количества межмодульных вызовов и уменьшение объемов информации, передаваемой между модулями, по сравнению с модульным программированием. Это достигается посредством более полной локализации данных и интегрирования их с подпрограммами обработки, что позволяет вести практически независимую разработку отдельных частей (объектов) программы. При построении программных комплексов по технологии ООП структуризация информации осуществляется при помощи баз данных. [2]

База данных (database) – это организованная совокупность совместно используемых логически связанных данных и описаний этих данных, относящаяся к определённой предметной области, предназначенная для удовлетворения информационных потребностей организации. [3]

База данных представляет собой долговременное хранилище данных, структура которого определяется на этапе проектирования этой базы данных. Находящиеся в базе данных данные являются общекорпоративными ресурсом, в котором вместо отдельных файлов, разбросанных по отделам предприятия, данные собраны вместе с минимальной избыточностью. В базе данных хранятся не только рабочие данные, описывающие какую-то предметную область, но и метаданные, предназначенные для описания структуры самих данных. [3]

На сегодняшний день базы данных практически не используются сами по себе, а являются хранилищем для приложений. Обычно пользователь работает с приложением через графический интерфейс, специально разработанный для решения задач предметной области. Интерфейс скрывает детали реализации системы. Методы разработки приложений относятся к группе дисциплин, составляющих программную инженерию. [4]

Основными композиционными средствами, используемыми при разработке дизайна интерфейса, являются цельность, пропорциональность, симметрия, ритм, контраст. [5]

## 1.2 Постановка задачи

Реализовать компьютерную игру «Управление деревней», основной механикой которой является экономический менеджмент, с использованием двух разных объектно-ориентированных языков программирования. Эта игра схожа с СУБД из-за работы с базой данных, но ключевой особенностью является развлекательная направленность. Игрок должен взять на себя роль управляющего деревни и помочь крестьянами продержаться как можно дольше.

При запуске программы должно открываться главное меню, откуда можно перейти к новой игре, загрузить незаконченную игру, открыть справку, посмотреть список рекордов или же выйти на рабочий стол. Игра встречает пользователя окном, в центральной части которого располагается изображение деревни (отражает её текущее состояние), в левой отображается текущее число ресурсов, в правой находится несколько кнопок для управления деревней, а внизу есть окно с фразами вашего помощника о произошедших событиях. В верхнем ленточном меню доступны для взаимодействия кнопки сохранения, справки, а также выхода.

Отображаемыми показателями деревни являются число крестьян, текущий множитель «Скорость Жизни», «Дни Процветания», а также количество денег, хлеба и домашнего скота. Начало игры встречает игрока с небольшой деревней, обладающей малым количеством ресурсов. Каждый крестьянин ускоряет прирост количества хлеба и скота (скот добавляется медленнее), но при этом ускоряет и убыль денег (незначительно). Множитель (так называемая «Скорость Жизни») показывает, во сколько раз увеличилась трата денег отдельным крестьянином и его «прожорливость». Множитель увеличивается при завершении одного цикла сезонов, изначально равен единице. Показатель «Дни Процветания» отображает прошедшее время от начала игры, в итоге будет являться финальным счётом.

Для управления деревней используется несколько окон, открываемых с помощью кнопок:   
список крестьян и товарная биржа. Список крестьян представляет собой таблицу, в которой отображается фамилия и имя крестьянина, количество производимого хлеба и скота за Час Процветания (3 реальные секунды), «прожорливость», а также количество денег, которые он   
тратит за Час. Прожорливость – это два числа, обозначающих расход хлеба и скота на этого крестьянина за Час Процветания Зимой. Есть кнопка для изгнания какого-либо крестьянина, а также кнопка для открытия списка рынка труда. Список рынка труда является схожим по структуре, но добавляется информация о стоимости найма. На рынке труда всегда доступно несколько работников со случайными характеристиками. Изгнание крестьянина является бесплатным, наём – платным.

Товарная биржа показывает текущую стоимость единицы хлеба, а также домашнего скота. Здесь же можно и продать выбранное количество ресурсов. Особенностью игры является то, что хлеб и скот нельзя покупать, можно только продавать.

После начала игры игрок наблюдает за количеством ресурсов деревни (увеличиваются за счёт крестьян), продаёт их, а также нанимает и изгоняет крестьян. Есть два игровых сезона: Лето и Зима (в зависимости от них меняется изображение деревни). Лето длится 4 Дня Процветания (2 минуты реального времени), Зима – 2 Дня (одна реальная минута). Зима отличается от Лета тем, что крестьяне перестают производить хлеб и скот, и включается «прожорливость», но деньги не тратятся. Наём и изгнание крестьян становятся недоступными до Лета. Биржа товаров остаётся доступной (но покупать хлеб и скот всё также нельзя). В середине Зимы происходит событие – бесплатный приход нового крестьянина или нападение волков (съедают небольшую часть скота). Окно с информацией от помощника игрока отображает оставшееся время до завершения текущего сезона, а также произошедшее событие («Из лесу вышел потерявшийся крестьянин и присоединился к Вам», или «Из лесу вышла стая волков и напала на Ваш скот. Вы потеряли Xголов домашнего скота», или «Крестьянин X ушёл из деревни из-за недостатка пищи»). Зимой игрок может только наблюдать за изменением количества ресурсов (точнее, их убылью), а также продавать часть этих ресурсов (Зимой это довольно бесполезно). После окончания Зимы снова начинается Лето, а множитель «Скорость Жизни» увеличивается на единицу, и цены на бирже товаров случайно немного изменяются (растут или падают).

Если количество денег стало равно нулю, то крестьяне перестают производить ресурсы Летом (Зимой они всегда не могут производить). Если количество хлеба или скота стало равно нулю, то работник, которому для питания требуется этот ресурс, начинает отсчёт и через пять Часов Процветания (15 реальных секунд) уходит из деревни. Если до окончания отсчёта успел появиться нужный ресурс (наступило Лето и крестьяне успели произвести ресурс), то крестьянин остаётся. Если количество крестьян стало равно нулю, то начинается отсчёт в два Часа Процветания   
(6 реальных секунд), после окончания которых при сохранившемся отсутствии крестьян игра заканчивается со счётом Дней Процветания и предлагается ввести имя для таблицы рекордов.

Основная концепция игры заключается в тренировке умения производить быстрые и верные расчёты, а также в обучении ожиданию. Особенно сильно пониманию и осознанию ожидания способствует сезон Зимы, когда игрок никак не может повлиять на ход событий. Зимой игрок вместе со своими крестьянами трепетно и терпеливо ждёт наступления Лета. Он борется со скукой и тревожностью, как и крестьяне.

# 2 Проектирование

## 2.1 Структура данных

Условные обозначения:

Час Процветания – это 3 секунды реального времени,   
День Процветания – это 30 секунд реального времени.

Класс «Крестьянин» содержит следующие поля:

1. Имя (строка)
2. Фамилия (строка)
3. Кол-во производимого хлеба за Час Процветания (целое число) [варьируется от 1 до 10]
4. Кол-во производимого домашнего скота за Час Процветания (целое число)   
   [варьируется от 0 до 2]
5. Прожорливость: хлеб (расходуемое крестьянином кол-во   
   за Час Процветания Зимой) (целое число) [варьируется от 1 до 3]
6. Прожорливость: домашний скот (расходуемое крестьянином кол-во   
   за Час Процветания Зимой) (целое число) [варьируется от 0 до 1]
7. Денежные расходы за один Час Процветания в рублях (целое число)   
   [варьируется от 10 до 100]
8. Стоимость найма в рублях (целое число) [варьируется от 5 до 50]
9. Отсчёт времени в Часах Процветания с момента, когда закончилась еда для этого крестьянина (целое число) [изначально равно 0]

Класс «Деревня» содержит следующие поля:

1. Кол-во крестьян в деревне (целое число) [изначально равно 3]
2. Кол-во Часов Процветания с момента начала игры (целое число) [изначально равно 1]
3. Кол-во Дней Процветания с момента начала игры (равно финальному счёту   
   при завершении игры) (целое число) [изначально равно 0]
4. Текущий множитель «Скорость Жизни» (увеличивает денежные расходы и прожорливость крестьян, эти показатели умножаются на этот множитель. Он возрастает в конце каждого цикла сезонов) (целое число) [изначально равен 1]
5. Флаг текущего сезона (равен нулю Летом, равен единице Зимой) (целое число) [изначально равен 0]
6. Общее кол-во хлеба в деревне (целое число) [изначально равно 200]
7. Общее кол-во голов домашнего скота в деревне (целое число) [изначально равно 20]
8. Бюджет деревни в рублях (целое число) [изначально равен 1000]
9. Крестьяне (массив объектов класса «Крестьянин»)
10. Кол-во Часов Процветания в момент, когда текущее время года только началось (целое число) [изначально равно 0]
11. Отсчёт времени в Часах Процветания с момента, когда ушли все крестьяне (целое число) [изначально равно 0]
12. Текущая цена за ед. хлеба на Торговой бирже в рублях [изначально равна 30]
13. Текущая цена за одну голову домашнего скота на Торговой бирже в рублях   
    [изначально равна 200]

Объект класса «Деревня» хранится в файле «деревня.txt», массив объектов класса «Крестьянин» хранится в отдельном файле «крестьяне.txt». В файле «деревня.txt» значения полей отделяются переносом строки. В случае с файлом «крестьяне.txt» после записи значений всех полей через точку с запятой для одного крестьянина ставится символ переноса строки (enter), и идут значения этих же полей для другого крестьянина. В файле «имена.txt» хранится список возможных имён для крестьян, из которого имена выбираются случайно. В файле «фамилии.txt» хранится список возможных фамилий, которые тоже выбираются случайно. Как имена, так и фамилии отделяются друг от друга символом переноса строки.

В файле «рекорды.txt» хранятся соответственно рекорды игроков. Принцип хранения рекордов в файле: имя\_игрока, точка с запятой, значение\_рекорда, символ переноса строки. Затем идёт запись для следующего игрока. Значение рекорда является целым числом, имя игрока – строкой.

Классы «Крестьянин» и «Деревня» содержат в себе методы установления и получения значений отдельных полей (так называемые методы «set» и «get»). По ходу реализации программы могут быть добавлены и другие методы, такие как инициализация (установление значения полей) всего класса целиком.

На протяжении всей игры основным хранилищем всей информации о деревне является объект класса «Деревня», доступный из любого окна игры (является глобальным), внутри него же хранится массив с объектами класса «Крестьянин».

## 2.2 Алгоритм решения

Исходя из поставленной задачи, реализуемая игра должна работать в диалоговом режиме. При запуске игры открывается главное меню, содержащее кнопки:

* Новая игра
* Загрузка незаконченной игры
* Справка
* Список рекордов
* Выход из игры

Кнопка «Новая игра» открывает окно с игрой, при этом все показатели деревни имеют начальное значение по умолчанию. Кнопка «Загрузка незаконченной игры» также открывает окно с игрой, но показатели деревни и список крестьян предварительно загружаются из сохранённых файлов «крестьяне.txt» и «деревня.txt». «Справка» открывает окно со справочной информацией (правила игры, условные обозначения и т. д.). «Список рекордов» открывает окно с таблицей из двух столбцов: «имя игрока» и «значение рекорда», рекорды загружаются из файла «рекорды.txt».   
«Выход из игры» завершает работу игры.

Начатая игра встречает пользователя окном, в центральной части которого располагается изображение деревни (показывает, Лето сейчас или Зима), в левой части отображается текущее число ресурсов, в правой находится несколько кнопок для управления деревней, а внизу есть окно с фразами вашего помощника о произошедших событиях. В верхнем ленточном меню доступны для взаимодействия кнопки сохранения, справки, а также выхода. Текущее состояние игры не сохраняется автоматически, игрок должен самостоятельно сохранить игру перед выходом, если не хочет потерять свой прогресс.

Кнопками управления деревней являются:

* Список крестьян
* Товарная биржа

Кнопка «Список крестьян» открывает окно с таблицей, содержащей информацию о крестьянах деревни. В таблице содержится: порядковый номер крестьянина (простая нумерация строк таблицы), имя и фамилия, количество производимого хлеба и скота за Час Процветания, прожорливость, а также количество денег, которые он тратит за Час. Разные показатели производительности (хлеб и скот) записываются в одну ячейку таблицы, но через косую черту “/”. Также записывается и прожорливость в кол-ве хлеба и скота. В верхней части окна имеется кнопка «Биржа труда» для открытия окна с биржей, а также небольшое поле для ввода номера крестьянина, которого игрок желает изгнать. Правее расположена кнопка «Изгнать», при нажатии которой крестьянин с выбранным номером удаляется из таблицы и покидает деревню. Кнопки «Биржа труда» и «Изгнать» недоступны Зимой.

«Биржа труда» является окном, в котором отображается похожая на «Список крестьян» таблица. В таблице содержится: порядковый номер крестьянина (простая нумерация строк таблицы), имя и фамилия, количество производимого хлеба и скота за Час Процветания, прожорливость, денежные расходы за один Час Процветания, а также стоимость найма. В верхней части окна есть небольшое поле для ввода номера крестьянина, которого игрок желает нанять. Правее расположена кнопка «Нанять», которая позволяет добавить выбранного крестьянина в список крестьян своей деревни. Поля для ввода номеров крестьян в «Списке крестьян» и на «Бирже труда» защищены от ввода несуществующего номера крестьянина, также нельзя нанять крестьянина при недостаточном количестве денег. Биржа труда при каждом запуске отображает пять крестьян со случайными характеристиками (при повторном запуске Биржи труда будут показаны совершенно другие случайные крестьяне).

Окно «Товарная биржа» открывается из основного окна игры, в нём отображаются текущие цены единицы хлеба и одной головы домашнего скота. Есть поле для ввода кол-ва хлеба или скота, которое игрок желает продать. Правее этих полей расположены радиокнопки «Хлеб» и «Домашний скот», позволяющие выбрать нужный тип. Кнопка «Продать» осуществляет работу в соответствии с введённым игроком числом и выбранным типом ресурсов. Если введено некорректное число (ноль или число большее, чем есть у игрока), то кнопка «Продать» становится неактивной, пока игрок не введёт допустимое число.

В левой части основного окна игры отображается: число крестьян, текущий множитель «Скорость Жизни», «Дни Процветания», а также количество денег, хлеба и домашнего скота. В нижней части окна отображается информация от «Помощника», а именно: оставшееся время в Часах Процветания до завершения текущего времени года и последнее произошедшее событие («Из лесу вышел потерявшийся крестьянин и присоединился к Вам», или «Из лесу вышла стая волков и напала на Ваш скот. Вы потеряли X голов домашнего скота», или «Крестьянин X ушёл из деревни из-за недостатка пищи»).

Получение текущих показателей деревни, связанных с течением времени, происходит по таймеру. **Каждые 3 секунды реального времени (один Час Процветания) выполняется следующий алгоритм**:

Кол-во Часов Процветания увеличивается на 1.

Если кол-во крестьян равно 0, число «Отсчёта времени без крестьян» увеличивается на 1. Если это число равно 2 (прошло 2 Часа П. или же 6 реальных секунд), то игра заканчивается (сначала устанавливается спец. флаг). Если же при проверке хотя бы один крестьянин нашёлся, отсчёт аннулируется.

Если флаг сезона равен 0 (Лето) и бюджет не равен 0, к значению кол-ва хлеба прибавляется значение производительности хлеба каждого крестьянина в отдельности, аналогично с кол-вом скота. При этом же флаге из бюджета вычитается значение расходов каждого крестьянина в отдельности, умноженное на «Скорость Жизни».

Если флаг сезона равен 1 (Зима), из значения кол-ва хлеба вычитается значение прожорливости в хлебе каждого крестьянина в отдельности, умноженной на «Скорость Жизни», аналогично с кол-вом скота. Если кол-во хлеба или скота стало равно нулю или отрицательному числу, то оно становится равным нулю (отрицательных чисел быть не должно) и крестьянин, которому не досталось еды, увеличивает своё число «Отсчёта кол-ва Часов П. без еды» на 1. Если это число стало равно 5 (прошло 5 Часов П.), крестьянин уходит из деревни, и об этом пишет «Помощник». Если еда нашлась до достижения числа 5, то отсчёт аннулируется.

Получение Дней Процветания: если кол-во Часов Процветания делится без остатка на 10 (в одном Дне Процветания 10 Часов Процветания, или же в половине реальной минуты 10 раз по 3 секунды), то кол-во Дней Процветания увеличивается на 1.

Разница между «*Длительностью этого времени года в Часах Процветания*» и суммой *«Текущего кол-ва Часов Процветания»* и «*Кол-ва Часов в момент, когда текущее время года только началось»* выводится в нижней части окна игры. Она выступает в качестве числа X во фразе «До конца этого времени года осталось Х Часов Процветания».

Если флаг сезона равен 0 (так как Лето) и *«Текущее кол-во Часов Процветания»* вычесть «*Кол-во Часов в момент, когда текущее время года только началось»* равно 40 (то есть от начала времени года уже прошло 4 Дня П. или 40 Часов П., что соответствует длительности Лета), то флаг сезона равен 1 (настала Зима), изображение деревни меняется на зимнее, а *«Кол-во Часов Процветания в момент, когда текущее время года только началось*» станет равно *«Текущему кол-ву Часов».*

Если флаг равен 0 (так как Зима) и *«Текущее кол-во Часов Процветания»* вычесть «*Кол-во Часов в момент, когда текущее время года только началось»* равно 20 (то есть от начала времени года уже прошло 2 Дня П. или 20 Часов П., что соответствует длительности Зимы), то флаг сезона равен 0 (настало Лето), изображение деревни меняется на летнее, а *«Кол-во Часов Процветания в момент, когда текущее время года только началось*» станет равно *«Текущему кол-ву Часов».* Ещё после окончания Зимы множитель «Скорость Жизни» увеличивается на 1, а также случайно немного изменяются цены на продажу (каждая случайно увеличивается или уменьшается на величину в диапазоне от 5% до 30% относительно текущего значения, итоговые числа округляются до целых).

Если флаг равен 0 (так как Зима) и *«Текущее кол-во Часов Процветания»* вычесть «*Кол-во Часов в момент, когда текущее время года только началось»* равно 10 (то есть от начала времени года уже прошёл 1 День П. или 10 Часов П., что соответствует половине Зимы), то случайно выбирается одно из двух событий: в деревню бесплатно добавляется новый крестьянин, информация об этом пишется внизу в информации от «Помощника». Или же волки съедают часть скота (пропадает от 5% до 25% от текущего кол-ва, итоговые числа округляются до целых) и также пишется в информации от «Помощника».

Если спец. флаг окончания игры установлен (то есть равен 1), то игра заканчивается. Появляется небольшое окно, в котором написано кол-во набранных очков (Дней Процветания), а также есть поле, чтобы вписать имя игрока. Правее поля находится кнопка «ОК», которая позволяет записать имя игрока и его рекорд в список рекордов, после чего вновь открывается Главное меню.

# 3 Реализация

## 3.1 Выбор средств реализации

Программная реализация данного проекта была выполнена на языке C++ (первая реализация) и на языке C# (вторая реализация) с использованием интегрированной среды разработки программного обеспечения Microsoft Visual Studio 2010. В частности, для разработки интерфейса использовались Windows Forms.

Язык С++ – компилируемый, статически типизированный язык программирования общего назначения.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов, делегаты, атрибуты, события и т. д.

Системные требования программы:

* Операционная система Windows XP SP3 или выше;
* Оперативная память 50МБ или выше;
* Жесткий диск: 12МБ для хранения программы и вспомогательных файлов, дополнительное место для хранения сохранения игры;
* .NET Framework версии 2.0 или выше;
* Наличие мыши, клавиатуры, монитора;
* Разрешение экрана 1024 x 768 или выше.

Для проверки введенных данных на корректность использовались элементы MaskedTextBox и обработчики событий изменения текстового поля, а также таймер.

При разработке игры также использовалась система контроля версий Git и   
веб-сервис GitHub. Благодаря системе контроля версий можно откатить изменения, а также проследить процесс создания программы. Ветки не использовались из-за ненадобности параллельной разработки.

Ссылки на GitHub

Реализация на C++: <https://github.com/GulinAlexey/kurs2020>

Реализация на C#: <https://github.com/GulinAlexey/kurs2020_sharp>

## 3.2 Структура программы

В программе реализованы следующие модули и взаимодействие между ними, изображенные на рисунке 1.

Игра

(новая или загруженная)

Главное меню

Список рекордов

Справка

Список крестьян

Товарная биржа

Биржа труда

Конец игры

Рисунок 1 – Модульная структура, отражающая связь форм

В файле «names.txt» хранится список возможных имён для крестьян, из которого имена выбираются случайно. В файле «surnames.txt» хранится список возможных фамилий, которые тоже выбираются случайно. Как имена, так и фамилии отделяются друг от друга символом переноса строки.

Рекорды хранятся в файле «records.txt» в следующем формате: имя\_игрока, точка с запятой, значение\_рекорда, символ переноса строки. Затем идёт запись для следующего игрока.

Сохранение игры разделено на два файла: «village.txt» и «krests.txt». В первом файле хранятся значения полей объекта класса Village (деревня), отделяются друг от друга символом переноса строки. В файле «krests.txt» хранится информация о крестьянах. В реализации на C++ значения полей одного крестьянина отделяются точкой с запятой, затем ставится символ переноса строки, далее идут значения полей для другого крестьянина. В реализации на C# значения полей одного крестьянина отделяются символом переноса строки, затем идёт строка с символами “///////” (предназначена для разграничения информации о разных крестьянах) и символом переноса строки, далее идут значения полей для другого крестьянина.

Для изменения фона в главном окне игры при смене сезона используются изображения «leto.jpg» и «zima.jpg».

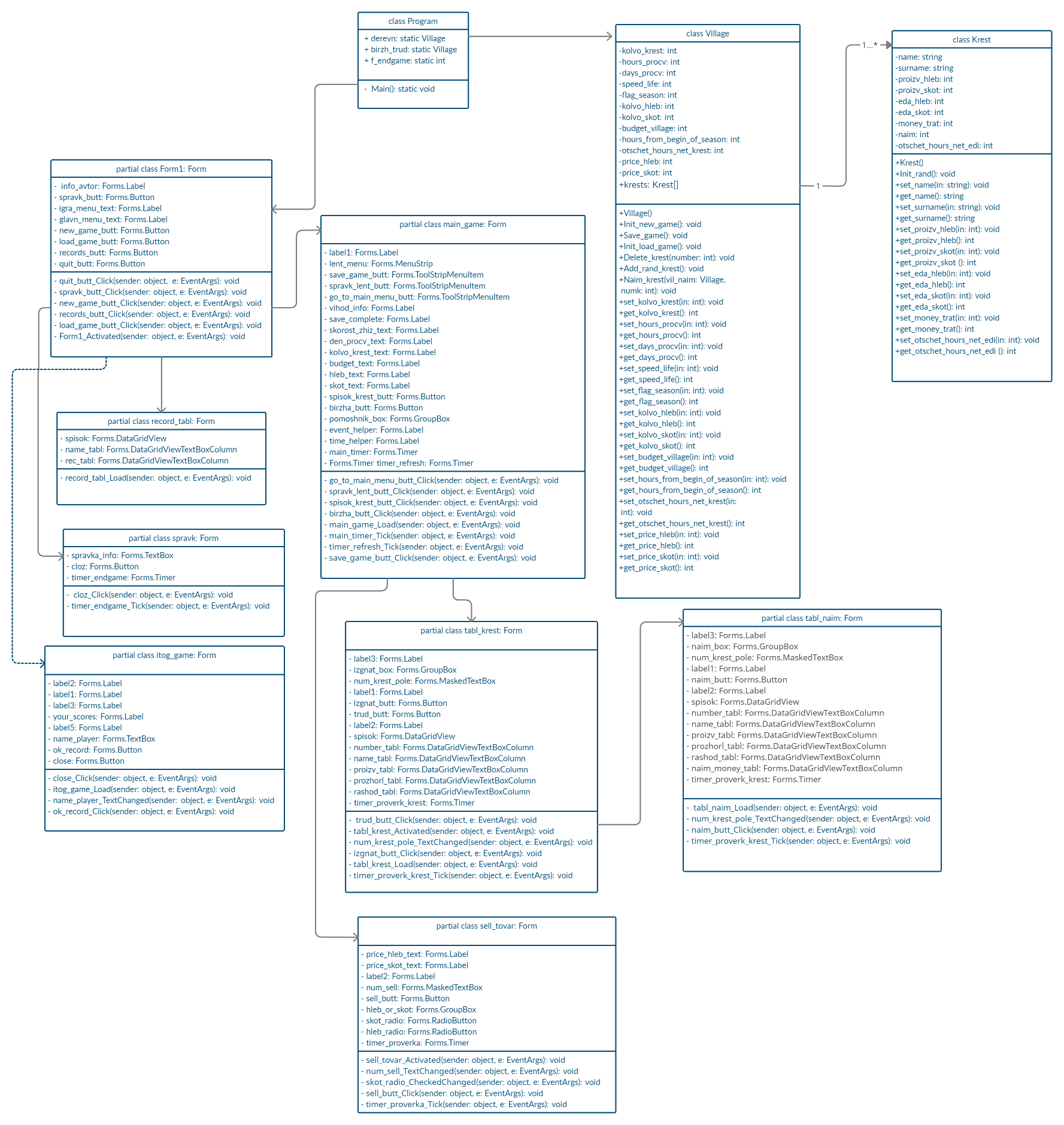


Рисунок 2 – Диаграмма классов

На рисунке 2 изображена диаграмма классов. Объект “derevn” класса Village содержит в себе всю информацию о деревне в процессе игры, в том числе массив крестьян (класс Krest). Этот объект является глобальным и может быть использован в нужной форме. В реализации на C++ объект деревни объявлен в файле «krest\_and\_village.h» (там же объявлен объект «birzh\_trud» класса Village, который используется в окне биржи труда). При необходимости работы с глобальным объектом этот файл подключается в нужную форму с помощью include. Отдельная глобальная переменная «f\_endgame» типа int используется, чтобы сигнализировать о том, что игра окончена. В реализации на C# соответствующие глобальные объекты объявлены один раз в файле «Program.cs».

Методы класса Krest

* Krest() – конструктор без параметров. Инициализирует объект. Числовые переменные приравниваются к нулю, имя равно «name», фамилия – «surname».
* void Init\_rand() – случайная инициализация объекта. Имя и фамилия берутся под случайными номерами из соответствующих файлов с возможными вариантами. Числовые поля объекта заполняются случайно в установленном диапазоне (см. пункт 2.1 Структура данных).
* Методы “void set\_xxx(string in)”, “string get\_xxx()”, “void set\_xxx(int in)” и “int get\_xxx()”   
  (где xxx – имя поля) используются для работы с соответствующими полями.   
  Метод set позволяет установить значение нужного поля, а метод get – получить значение поля.   
  В реализации на C# вместо переменной «in», являющейся аргументом функций set, используется переменная «inn» из-за особенностей языка (ключевое слово «in» зарезервировано).

Методы класса Village

* Village() – конструктор без параметров. Инициализирует объект. Полям присваиваются нулевые значения.
* void Init\_new\_game() – инициализация объекта перед новой игрой. Полям объекта присваиваются начальные значения (см. пункт 2.1 Структура данных). Массив крестьян очищается при необходимости, затем создаётся массив из трёх крестьян (объектов класса Krest), которые инициализируются с помощью Init\_rand().
* void Save\_game() – сохранение информации об объекте в файл. Данный метод используется при сохранении игры. Значения полей объекта записываются в файл «village.txt» (отделяются символом переноса строки), а значения полей объектов массива крестьян записываются в файл «krests.txt» (в реализации на C++ поля отделяются точкой с запятой, а отдельные крестьяне – переносом строки; на C# поля записываются на отдельных строках, а крестьяне отделяются строкой “///////”).
* void Init\_load\_game() – инициализация объекта из файлов. Этот метод используется при загрузке игры. Значения полей приравниваются к соответствующим значениям из файла. Массив крестьян предварительно удаляется, если нужно, а затем создаётся с помощью операции «new» с заданным количеством элементов. Значения полей крестьян также читаются из файла.
* void Delete\_krest(int number) – удаление крестьянина под выбранным номером. Крестьянин с выбранным номером удаляется из массива krests[] объектов класса Krest. Затем значение поля «kolvo\_krest» (кол-во крестьян в деревне) уменьшается на единицу.
* void Naim\_krest(Village vil\_naim, int numk) – найм крестьянина из другого объекта класса Village в данный. Используется при найме крестьян (крестьяне для найма хранятся в отдельном объекте класса Village). Выбранный крестьянин из другого массива добавляется в данный. Затем значение поля «kolvo\_krest» (кол-во крестьян в деревне) увеличивается на единицу.
* Методы “void set\_xxx(string in)”, “string get\_xxx()”, “void set\_xxx(int in)” и “int get\_xxx()”   
  (где xxx – имя поля) используются для работы с соответствующими полями.   
  Метод set позволяет установить значение нужного поля, а метод get – получить значение поля.   
  В реализации на C# вместо переменной «in», являющейся аргументом функций set, используется переменная «inn» из-за особенностей языка (ключевое слово «in» зарезервировано).

## 3.3 Описание реализации

При активации основной формы «Form1» открывается окно главного меню, содержащее несколько элементов Label (надписей) с названием игры и её автором, а также несколько элементов Button (кнопок) для открытия других окон или выхода из игры.

Примечание: запись “(object sender, EventArgs e)” на языке C# аналогична записи   
“(System::Object^ sender, System::EventArgs^ e)” на языке C++.

События формы «Form1»

* void quit\_butt\_Click(object sender, EventArgs e)  
  При нажатии на кнопку «quit\_butt» программа закрывается с использованием   
  метода exit(0) на С++ или Application.Exit() на C#.
* void spravk\_butt\_Click(object sender, EventArgs e)  
  При нажатии на кнопку «spravk\_butt» (справка) открывается форма «spravk» с использованием метода ShowDialog(). Пока не закрыто вызванное окно, взаимодействовать с предыдущим окном нельзя.
* void new\_game\_butt\_Click(object sender, EventArgs e)  
  При нажатии на кнопку «new\_game\_butt» (новая игра) флаг «f\_endgame» становится равным нулю, вызывается метод Init\_new\_game() для инициализации объекта derevn (деревня) класса Village начальными значениями. Затем открывается форма «main\_game» с использованием метода ShowDialog().
* void records\_butt\_Click(object sender, EventArgs e)  
  При нажатии на кнопку «records\_butt» (список рекордов) открывается форма «record\_tabl» с использованием метода ShowDialog().
* void load\_game\_butt\_Click(object sender, EventArgs e)  
  При нажатии на кнопку «load\_game\_butt» (загрузка незаконченной игры) флаг f\_endgame становится равным нулю, вызывается метод Init\_load\_game() для инициализации объекта derevn (деревня) класса Village с использованием информации из файлов сохранений. Затем открывается форма «main\_game» с использованием метода ShowDialog().
* void Form1\_Activated(object sender, EventArgs e)  
  При каждой активации данной формы проверяется, равен ли флаг «f\_endgame» единице. Если да, то вызывается форма «itog\_game» методом Show() (вызванная форма не мешает работе других форм). Затем проверяется, существуют ли файлы с сохранениями («village.txt» и «krests.txt»). Если хотя бы одного файла нет, кнопка «load\_game\_butt» становится неактивной (Enabled=false), в противном случае она становится активной.

При активации формы «spravk» открывается окно со справкой. Элемент TextBox «spravka\_info» содержит текст справки и имеет ползунок сбоку для прокрутки. Пользователь не может изменять текст. Ниже расположена кнопка «cloz» (закрыть), позволяющая закрыть данное окно. Также есть таймер «timer\_endgame», срабатывающий каждые 1,003 секунды.

События формы «spravk»

* void cloz\_Click(object sender, EventArgs e)  
  Данная форма закрывается с использованием метода Close().
* void timer\_endgame\_Tick(object sender, EventArgs e)  
  Каждый раз при срабатывании таймера через равные временные интервалы проверяется, равен ли флаг «f\_endgame» единице. Если да, то данная форма закрывается. Проверка по таймеру позволяет закрыть все открытые окна при окончании игры.

При активации формы «record\_tabl» открывается окно со списком рекордов. Оно содержит элемент DataGridView (таблицу) «spisok».

События формы «record\_tabl»

* void record\_tabl\_Load(object sender, EventArgs e)  
  При загрузке данной формы проверяется, существует ли файл с рекордами «records.txt». Если да, то таблица заполняется информацией из файла, пока он весь не будет прочитан. Имя игрока записывается в первую колонку, значение рекорда – во вторую.

При активации формы «main\_game» открывается основное окно с игрой. Оно содержит несколько элементов Label (надписей) слева, содержащих информацию о текущем состоянии деревни (kolvo\_krest\_text, budget\_text и т. д.). Пара надписей сверху содержит напоминания о том, чему равны игровые единицы времени, а также о том, что игра не сохраняется автоматически. Надпись «save\_complete» отображается на несколько секунд после сохранения игры. Ещё наверху расположен элемент MenuStrip (ленточное меню) «lent\_menu» с несколькими элементами ToolStripMenuItem (кнопки ленточного меню). В правой части окна есть две кнопки «spisok\_krest\_butt» и «birzha\_butt». Внизу есть GroupBox, содержащий информацию от «Помощника»: две надписи, первая из которых показывает оставшееся до конца сезона время, а вторая – последнее произошедшее событие. Также есть два таймера: «main\_timer» срабатывает каждые 3 секунды, «timer\_refresh» срабатывает каждые 0,512 секунды.

События формы «main\_game»

* void go\_to\_main\_menu\_butt\_Click(object sender, EventArgs e)  
  При нажатии в ленточном меню кнопки «выход в главное меню» данная форма закрывается.
* void spravk\_lent\_butt\_Click(object sender, EventArgs e)  
  При нажатии в ленточном меню кнопки «справка» открывается форма «spravk» с использованием метода ShowDialog().
* void spisok\_krest\_butt\_Click(object sender, EventArgs e)  
  При нажатии на кнопку «spisok\_krest\_butt» (список крестьян) открывается форма «tabl\_krest» с использованием метода ShowDialog().
* void birzha\_butt\_Click(object sender, EventArgs e)  
  При нажатии на кнопку «birzha\_butt» (товарная биржа) открывается форма «sell\_tovar» с использованием метода ShowDialog().
* void main\_game\_Load(object sender, EventArgs e)  
  При загрузке данной формы надписи с показателями деревни (kolvo\_krest\_text,   
  budget\_text и т. д.) заполняются с использованием методов get объекта derevn. Также заполняется надпись «time\_helper» в GroupBox «pomoshnik\_box», показывающая оставшееся до конца сезона время.
* void main\_timer\_Tick(object sender, EventArgs e)  
  При срабатывании основного таймера игры происходит изменение показателей деревни. Например, крестьяне производят ресурсы и тратят деньги Летом или же едят Зимой. Происходит проверка прошедшего времени и при необходимости сменяется сезон или наступает случайное событие. Проверяются условия ухода крестьян, а также условия конца игры. Если условия конца игры выполнены, флаг «f\_endgame» становится равным единице и форма закрывается (подробнее алгоритм таймера описан в пункте 2.2 Алгоритм решения). Затем выводится информация о деревне (аналогично событию “main\_game\_Load”)
* void timer\_refresh\_Tick(object sender, EventArgs e)  
  При срабатывании таймера аналогично событию “main\_game\_Load” в соответствующие надписи выводится обновлённая информация о деревне. Данный таймер нужен, чтобы отображать актуальную информацию о том, что игрок кого-то изгнал, нанял или же что-то продал в те моменты, когда основной таймер ещё не сработал.
* void save\_game\_butt\_Click(object sender, EventArgs e)  
  При нажатии в ленточном меню кнопки «сохранить игру» вызывается метод Save\_game() объекта derevn для сохранения игры. Затем надпись «save\_complete» (игра успешно сохранена) становится видимой (Visible = true). Эта надпись снова станет невидимой при срабатывании основного таймера.

При активации формы «tabl\_krest» открывается окно со списком крестьян. В нём расположено несколько вспомогательных надписей (примечание, введите номер и т. д.), кнопка «trud\_butt», таблица «spisok» с информацией о крестьянах и GroupBox «izgnat\_box» с элементом MaskedTextBox (защищённое поле ввода) «num\_krest\_pole» и кнопкой «izgnat\_butt». В данное защищённое поле можно вводить только цифры. Ещё есть таймер «timer\_proverk\_krest», срабатывающий каждые 1,003 секунды.

События формы «tabl\_krest»

* void trud\_butt\_Click(object sender, EventArgs e)  
  При нажатии на кнопку «trud\_butt» (открыть биржу труда) открывается форма «tabl\_naim» с использованием метода ShowDialog().
* void tabl\_krest\_Activated(object sender, EventArgs e)  
  При активации формы проверяется флаг «f\_endgame», если он установлен, то форма закрывается. После проверки заполняется таблица «spisok» с использованием методов get объектов массива krests[] объекта derevn. Перед заполнением таблица очищается, так как форма может быть активирована несколько раз.
* void num\_krest\_pole\_TextChanged(object sender, EventArgs e)  
  При изменении текста в защищённом поле проверяется его состояние. Если поле пустое, введён ноль, или введённый номер больше, чем есть в таблице, то кнопка изгнания становится неактивной (Enabled = false). В противном случае она становится активной. В этом же событии проверяется состояние флага сезона. Если он равен единице (Зима), то кнопка открытия биржи труда и изгнания в юбом случае становятся неактивными. Если флаг равен нулю (Лето), то кнопка биржи труда становится вновь активной.
* void izgnat\_butt\_Click(object sender, EventArgs e)  
  При нажатии на кнопку «izgnat\_butt» (изгнать) вызывается метод Delete\_krest() объекта derevn, которому передаётся введённое в «num\_krest\_pole» число. После выполнения метода удаления крестьянина вызываются события “num\_krest\_pole\_TextChanged(sender, e)” и “tabl\_krest\_Activated(sender,e)” для повторной проверки ввода и обновления данных в таблице.
* void tabl\_krest\_Load(object sender, EventArgs e)  
  При загрузке данной формы вызывается событие “num\_krest\_pole\_TextChanged(sender, e)” для проверки ввода. Точнее говоря, используется одна из проверок в этом событии: проверка на флаг сезона. Если сейчас Зима, кнопка биржи труда и кнопка изгнания становятся сразу неактивными.
* void timer\_proverk\_krest\_Tick(object sender, EventArgs e)  
  При срабатывании таймера проверяется флаг «f\_endgame», если он равен единице, то данное окно закрывается. Ещё вызываются события “num\_krest\_pole\_TextChanged(sender, e)” и “tabl\_krest\_Activated(sender,e)” для повторной проверки ввода и обновления данных в таблице.

При активации формы «tabl\_naim» открывается окно биржи труда. В нём расположено несколько вспомогательных надписей (примечание, введите номер и т. д.), таблица «spisok» с информацией о крестьянах на бирже и GroupBox «naim\_box» с элементом MaskedTextBox (защищённое поле ввода) «num\_krest\_pole» и кнопкой «naim\_butt». В данное защищённое поле можно вводить только цифры. Ещё есть таймер «timer\_proverk\_krest», срабатывающий каждые 1,003 секунды.

События формы «tabl\_naim»

* void tabl\_naim\_Load(object sender, EventArgs e)   
  При загрузке данной формы проверяется флаг сезона. Если он равен единице (Зима), то форма сразу закрывается. Если массив с нанимаемыми крестьянами krests[] объекта birzh\_trud (класс Village) не является пустым, то он очищается (для новых случайных крестьян). Затем создаётся новый массив из пяти крестьян (объектов класса Krest), каждый крестьянин инициализируется методом Init\_rand().
* void num\_krest\_pole\_TextChanged(object sender, EventArgs e)   
  При изменении текста в защищённом поле проверяется его состояние. Если денег для найма недостаточно, поле пустое, введён ноль, или введённый номер больше, чем есть в таблице, то кнопка найма становится неактивной (Enabled = false). В противном случае она становится активной. В этом же событии проверяется состояние флага сезона. Если он равен единице (Зима), то форма закрывается (нанимать крестьян Зимой по правилам игры нельзя).
* void naim\_butt\_Click(object sender, EventArgs e)  
  При нажатии на кнопку «naim\_butt» вызывается метод Naim\_krest() объекта derevn, которому передаётся введённое в «num\_krest\_pole» число и объект birzh\_trud класса Village с массивом крестьян для найма. Затем используется метод set\_budget\_village() объекта derevn, чтобы вычесть из данного бюджета стоимость найма. После чего вызывается метод Delete\_krest() объекта birzh\_trud, чтобы удалить крестьянина, который был только что нанят. В конце вызывается событие “num\_krest\_pole\_TextChanged(sender,e)” для обновления таблицы.
* void timer\_proverk\_krest\_Tick(object sender, EventArgs e)   
  При срабатывании таймера проверяется флаг «f\_endgame», если он равен единице, то данное окно закрывается. Ещё вызываются события “num\_krest\_pole\_TextChanged(sender, e)” и “tabl\_naim\_Activated(sender,e)” для повторной проверки ввода и обновления данных в таблице.

При активации формы «sell\_tovar» открывается окно товарной биржи. В нём расположены надписи «price\_hleb» и «price\_skot», защищённое поле ввода «num\_sell», кнопка «sell\_butt», а также GroupBox «heb\_or\_skot» с радиокнопками (RadioButton) «hleb\_radio» и «skot\_radio». Ещё есть таймер «timer\_proverka», срабатывающий каждые 1,003 секунды.

События формы «sell\_tovar»

* void sell\_tovar\_Activated(object sender, EventArgs e)  
  При активации данной формы в надписи «price\_hleb» и «price\_skot» выводится соответствующая информация из объекта derevn (используются методы “get\_price\_hleb()” и “get\_price\_skot()”).
* void num\_sell\_TextChanged(object sender, EventArgs e)  
  При изменении текста в защищённом поле проверяется его состояние. Если поле пустое, введён ноль, или введено число, превышающее кол-во имеющегося у игрока данного типа ресурсов, то кнопка продажи становится неактивной. В противном случае она становится активной.
* void skot\_radio\_CheckedChanged(object sender, EventArgs e)   
  При переключении радиокнопок выбора типа ресурсов вызывается событие “sell\_TextChanged(object sender, EventArgs e)” для повторной проверки ввода.
* void sell\_butt\_Click(object sender, EventArgs e)  
  Если нажата кнопка «sell\_butt» (продать), то выбранный тип ресурсов в выбранном количестве вычитается из ресурсов игрока (используются методы set), а затем ему начисляется соответствующее цене количество денег (используется метод set\_budget\_village()). После чего вызывается событие “sell\_TextChanged(object sender, EventArgs e)” для повторной проверки ввода.
* void timer\_proverka\_Tick(object sender, EventArgs e)   
  При срабатывании таймера проверяется флаг «f\_endgame», если он равен единице, то данное окно закрывается. Ещё вызываются события “sell\_TextChanged(object sender, EventArgs e)” и “sell\_tovar\_Activated(sender, e)” для повторной проверки ввода и обновления информации о текущих ценах.

При активации формы «itog\_game» открывается окно конца игры. Оно содержит несколько вспомогательных надписей (игра окончена, введите имя и т. д.), а также надпись «your\_scores» с финальным счётом. Ещё есть поле ввода (TextBox) «name\_player» и две кнопки: «ok\_record» и «close».

События формы «itog\_game»

* void close\_Click(object sender, EventArgs e)  
  При нажатии кнопки «close» (закрыть без сохранения) форма закрывается.
* void itog\_game\_Load(object sender, EventArgs e)  
  При загрузке данной формы в надпись «your\_scores» выводится финальный счёт игрока (используется метод get\_days\_procv() объекта derevn).
* void name\_player\_TextChanged(object sender, EventArgs e)  
  При вводе в поле «name\_player» проверяется, пустое ли поле и содержит ли оно точку с запятой (проверка на точку с запятой только в реализации на C++). Если да, то кнопка записи рекорда «ok\_record» становится неактивной. В противном случае она становится активной.
* void ok\_record\_Click(object sender, EventArgs e)  
  При нажатии на кнопку «ok\_record» (записать и закрыть) в файл с рекордами записывается имя игрока и его рекорд (в начало пустого файла или в конец существующего). Затем данная форма закрывается.

# Заключение

В процессе выполнения работы были изучены принципы объектно-ориентированного программирования, получены навыки разработки приложений с графическим интерфейсом на объектно-ориентированных языках C++ и C#. Кроме того, был получен опыт в проектировании игры с авторской концепцией. Полученная игра обладает рядом базовых механик (методов взаимодействия игры с игроком) и уже сейчас может предложить увлекательный игровой процесс. Поля ввода в данной игре защищены от некорректного ввода, поэтому игра стабильна и ошибки маловероятны.

Достоинства игры:

* Реиграбельность вследствие случайной генерации характеристик крестьян;
* Возможность сохранения игрового процесса;
* Список рекордов, сортируемый в порядке убывания;
* Постепенно увеличивающаяся сложность игры;
* Биржа труда крестьян обновляется при каждом запуске, поэтому игрок может легко подобрать крестьян с нужными характеристиками;
* Случайные события добавляют разнообразия в геймплей;
* Малый размер программы, её неприхотливость к характеристикам компьютера.

Недостатки игры:

* Малое количество анимаций, игра выглядит слишком статичной;
* Из-за монотонного темпа игры она может не подойти для активных игроков, желающих насыщенных и ярких событий;
* Можно хранить только одно сохранение игры;
* Файлы с именами, фамилиями, сохранениями и рекордами никак не защищены и не зашифрованы. Любой из файлов можно удалить и вызвать критические ошибки в работе программы. Кроме того, возможность редактирования сохранений и рекордов может привести к жульничеству;
* Отсутствует обучение.

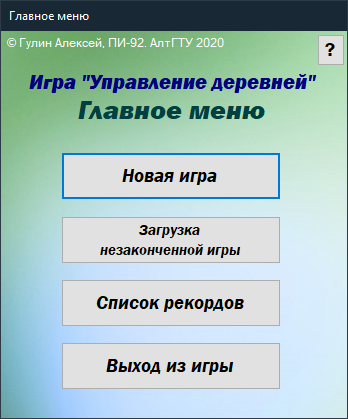
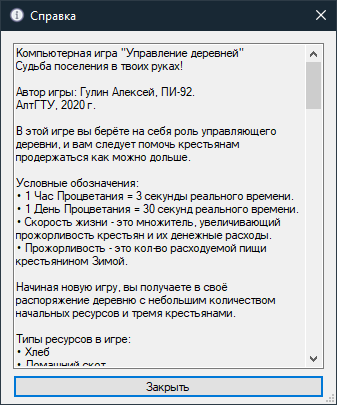
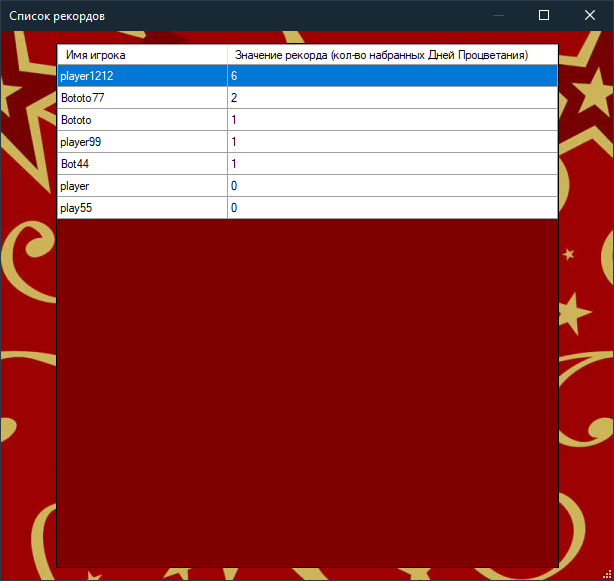
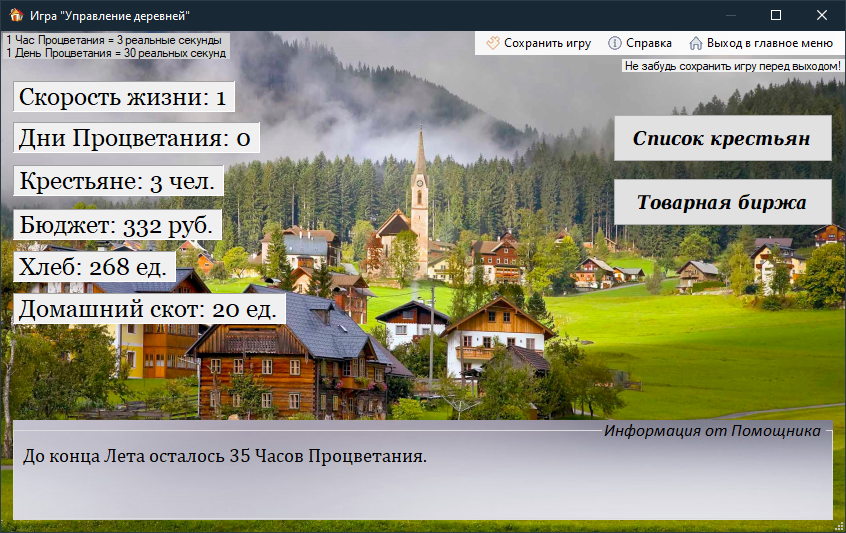
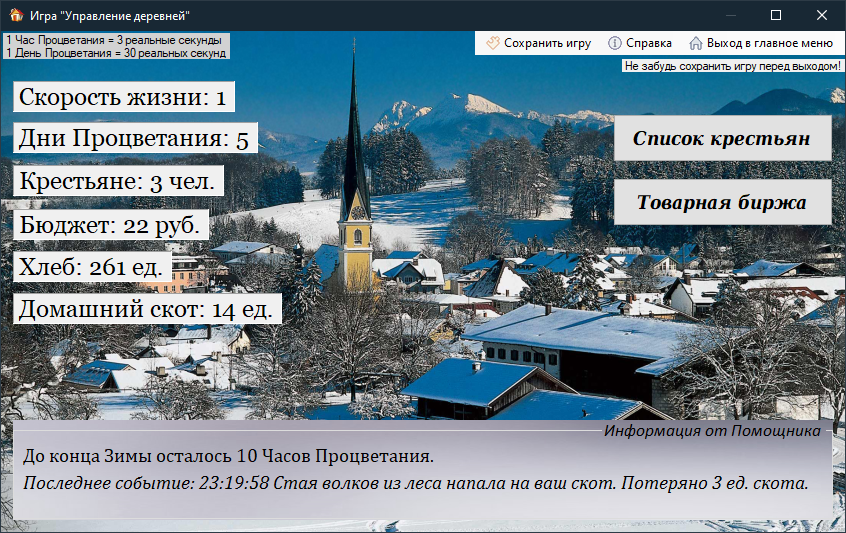
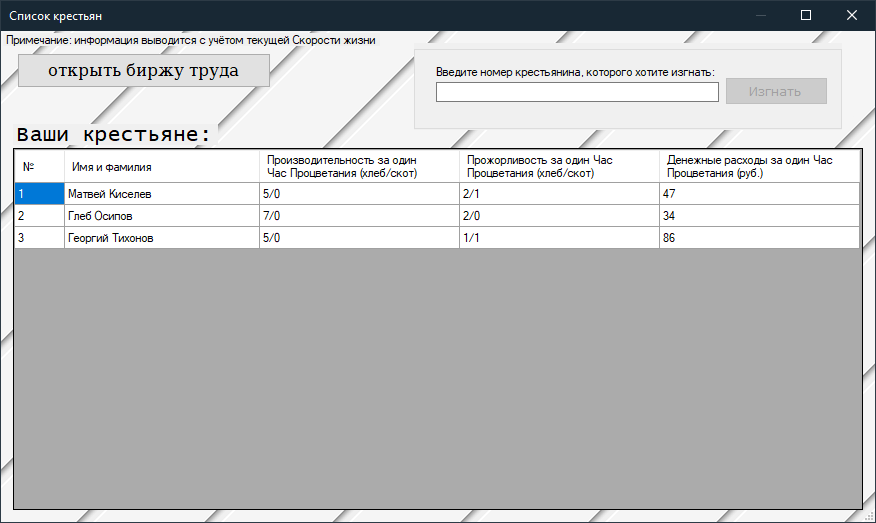
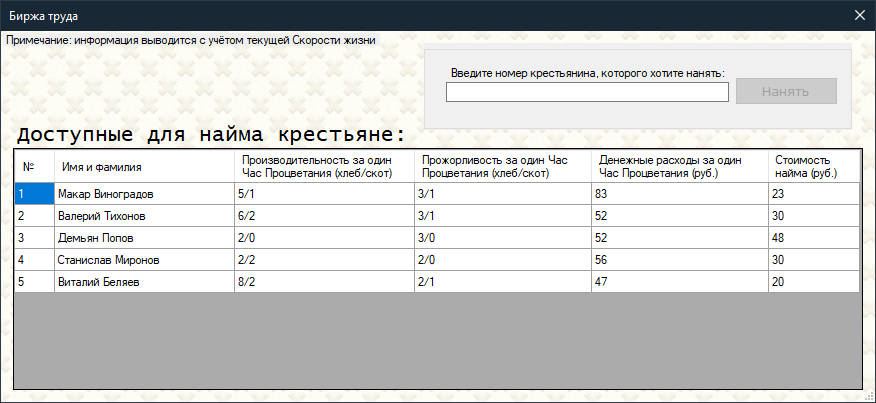
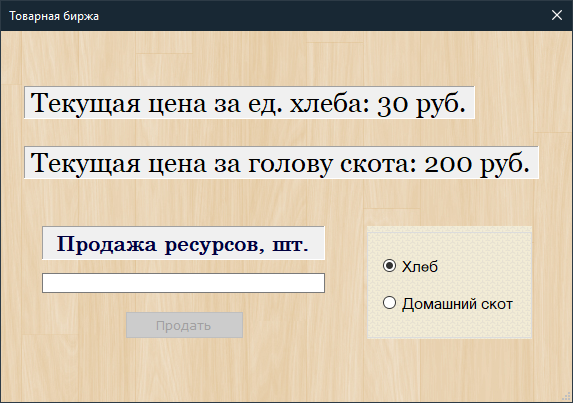
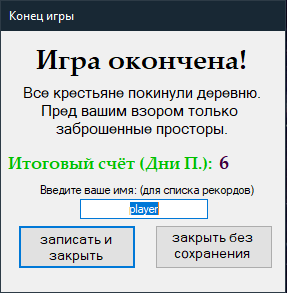
Перспективы дальнейшего развития работы:

* Улучшение интерфейса и качества графики в целом;
* Окно управления несколькими сохранениями игры;
* Адаптация для мобильных устройств;
* Звуковое сопровождение;
* Добавление новых механик и мини-игр;
* Возможность управления скоростью течения игры (ускорение или замедление) для удобства игрока.

# Список использованных источников

1. Горнаков, С. Г. Программирование компьютерных игр под Windows в XNA Game Studio Express / С. Г. Горнаков. — Москва : ДМК Пресс, 2008. — 384 с. — ISBN 978-5-94074-386-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/1166 (дата обращения: 25.10.2020). — Режим доступа: для авториз. пользователей.
2. Бабушкина, И. А. Практикум по объектно-ориентированному программированию : учебное пособие / И. А. Бабушкина, С. М. Окулов. — 5-е изд. — Москва : Лаборатория знаний, 2020. — 369 с. — ISBN 978-5-00101-780-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/135561 (дата обращения: 25.10.2020). — Режим доступа: для авториз. пользователей.
3. Осипов, Д. Л. Технологии проектирования баз данных / Д. Л. Осипов. — Москва : ДМК Пресс, 2019. — 498 с. — ISBN 978-5-97060-737-4. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131692 (дата обращения: 25.10.2020). — Режим доступа: для авториз. пользователей.
4. Новиков, Б. А. Основы технологий баз данных / Б. А. Новиков ; под редакцией Е. В. Рогова. — Москва : ДМК Пресс, 2019. — 240 с. — ISBN 978-5-94074-820-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/123699 (дата обращения: 25.10.2020). — Режим доступа: для авториз. пользователей.
5. Баканов, А. С. Эргономика пользовательского интерфейса. От проектирования к моделированию человеко-компьютерного взаимодействия / А. С. Баканов, А. А. Обознов. — Москва : Институт психологии РАН, 2011. — 176 c. — ISBN 978-5-9270-0191-0. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/15677.html (дата обращения: 25.10.2020). — Режим доступа: для авторизир. пользователей.
6. “MSDN” – информационный сервис для разработчиков [Электронный ресурс] – Режим доступа http://msdn.microsoft.com/ru-ru/ms348103/library, свободный.

# Приложение А Снимки экранных форм пользовательского интерфейса

1. Главное меню  
   
2. Справка  
   
3. Список рекордов  
   
4. Основное окно игры (Лето)  
   
5. Основное окно игры (Зима)  
   
6. Список крестьян  
   
7. Биржа труда  
   
8. Товарная биржа  
   
9. Конец игры  
   

# Приложение Б Исходный код программы

## Б.1 Исходный код реализации на C++

Form1.h

#pragma once

#include <cstdlib>

#include <ctime>

#include <fstream>

#include "krest\_and\_village.h"

#include "main\_game.h"

#include "spravk.h"

#include "itog\_game.h"

#include "record\_tabl.h"

int f\_endgame=0; //флаг о том, что игра окончена (игрок проиграл) (глобальн.)

namespace kurs2020 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для Form1

/// </summary>

public ref class Form1 : public System::Windows::Forms::Form

{

public:

Form1(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~Form1()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Label^ igra\_menu\_text;

private: System::Windows::Forms::Label^ glavn\_menu\_text;

private: System::Windows::Forms::Button^ new\_game\_butt;

private: System::Windows::Forms::Button^ load\_game\_butt;

private: System::Windows::Forms::Button^ records\_butt;

private: System::Windows::Forms::Button^ quit\_butt;

private: System::Windows::Forms::Button^ spravk\_butt;

private: System::Windows::Forms::Label^ info\_avtor;

protected:

protected:

protected:

private:

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

void InitializeComponent(void)

{

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(Form1::typeid));

this->igra\_menu\_text = (gcnew System::Windows::Forms::Label());

this->glavn\_menu\_text = (gcnew System::Windows::Forms::Label());

this->new\_game\_butt = (gcnew System::Windows::Forms::Button());

this->load\_game\_butt = (gcnew System::Windows::Forms::Button());

this->records\_butt = (gcnew System::Windows::Forms::Button());

this->quit\_butt = (gcnew System::Windows::Forms::Button());

this->spravk\_butt = (gcnew System::Windows::Forms::Button());

this->info\_avtor = (gcnew System::Windows::Forms::Label());

this->SuspendLayout();

//

// igra\_menu\_text

//

this->igra\_menu\_text->BackColor = System::Drawing::Color::Transparent;

this->igra\_menu\_text->Font = (gcnew System::Drawing::Font(L"Franklin Gothic Heavy", 15.75F, System::Drawing::FontStyle::Italic, System::Drawing::GraphicsUnit::Point,

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

this->igra\_menu\_text->ForeColor = System::Drawing::Color::Navy;

this->igra\_menu\_text->Location = System::Drawing::Point(1, 38);

this->igra\_menu\_text->Name = L"igra\_menu\_text";

this->igra\_menu\_text->Size = System::Drawing::Size(343, 29);

this->igra\_menu\_text->TabIndex = 0;

this->igra\_menu\_text->Text = L"Игра \"Управление деревней\"";

this->igra\_menu\_text->TextAlign = System::Drawing::ContentAlignment::TopCenter;

this->igra\_menu\_text->Click += gcnew System::EventHandler(this, &Form1::label1\_Click);

//

// glavn\_menu\_text

//

this->glavn\_menu\_text->BackColor = System::Drawing::Color::Transparent;

this->glavn\_menu\_text->Font = (gcnew System::Drawing::Font(L"Franklin Gothic Heavy", 20.25F, System::Drawing::FontStyle::Italic,

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

this->glavn\_menu\_text->ForeColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(0)), static\_cast<System::Int32>(static\_cast<System::Byte>(64)),

static\_cast<System::Int32>(static\_cast<System::Byte>(64)));

this->glavn\_menu\_text->Location = System::Drawing::Point(0, 62);

this->glavn\_menu\_text->Name = L"glavn\_menu\_text";

this->glavn\_menu\_text->Size = System::Drawing::Size(344, 44);

this->glavn\_menu\_text->TabIndex = 1;

this->glavn\_menu\_text->Text = L"Главное меню";

this->glavn\_menu\_text->TextAlign = System::Drawing::ContentAlignment::TopCenter;

this->glavn\_menu\_text->Click += gcnew System::EventHandler(this, &Form1::label1\_Click\_1);

//

// new\_game\_butt

//

this->new\_game\_butt->Font = (gcnew System::Drawing::Font(L"Franklin Gothic Demi", 14.25F, System::Drawing::FontStyle::Italic, System::Drawing::GraphicsUnit::Point,

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

this->new\_game\_butt->Location = System::Drawing::Point(60, 121);

this->new\_game\_butt->Name = L"new\_game\_butt";

this->new\_game\_butt->Size = System::Drawing::Size(220, 48);

this->new\_game\_butt->TabIndex = 2;

this->new\_game\_butt->Text = L"Новая игра";

this->new\_game\_butt->UseVisualStyleBackColor = true;

this->new\_game\_butt->Click += gcnew System::EventHandler(this, &Form1::button1\_Click);

//

// load\_game\_butt

//

this->load\_game\_butt->Font = (gcnew System::Drawing::Font(L"Franklin Gothic Demi", 11.25F, System::Drawing::FontStyle::Italic, System::Drawing::GraphicsUnit::Point,

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

this->load\_game\_butt->Location = System::Drawing::Point(60, 185);

this->load\_game\_butt->Name = L"load\_game\_butt";

this->load\_game\_butt->Size = System::Drawing::Size(220, 48);

this->load\_game\_butt->TabIndex = 3;

this->load\_game\_butt->Text = L"Загрузка\r\nнезаконченной игры";

this->load\_game\_butt->UseVisualStyleBackColor = true;

this->load\_game\_butt->Click += gcnew System::EventHandler(this, &Form1::load\_Click);

//

// records\_butt

//

this->records\_butt->Font = (gcnew System::Drawing::Font(L"Franklin Gothic Demi", 14.25F, System::Drawing::FontStyle::Italic, System::Drawing::GraphicsUnit::Point,

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

this->records\_butt->Location = System::Drawing::Point(60, 248);

this->records\_butt->Name = L"records\_butt";

this->records\_butt->Size = System::Drawing::Size(220, 48);

this->records\_butt->TabIndex = 4;

this->records\_butt->Text = L"Список рекордов";

this->records\_butt->UseVisualStyleBackColor = true;

this->records\_butt->Click += gcnew System::EventHandler(this, &Form1::record\_Click);

//

// quit\_butt

//

this->quit\_butt->DialogResult = System::Windows::Forms::DialogResult::Cancel;

this->quit\_butt->Font = (gcnew System::Drawing::Font(L"Franklin Gothic Demi", 14.25F, System::Drawing::FontStyle::Italic, System::Drawing::GraphicsUnit::Point,

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

this->quit\_butt->Location = System::Drawing::Point(60, 311);

this->quit\_butt->Name = L"quit\_butt";

this->quit\_butt->Size = System::Drawing::Size(220, 48);

this->quit\_butt->TabIndex = 5;

this->quit\_butt->Text = L"Выход из игры";

this->quit\_butt->UseVisualStyleBackColor = true;

this->quit\_butt->Click += gcnew System::EventHandler(this, &Form1::quit\_butt\_Click);

//

// spravk\_butt

//

this->spravk\_butt->Font = (gcnew System::Drawing::Font(L"Franklin Gothic Demi", 14.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->spravk\_butt->Location = System::Drawing::Point(316, 3);

this->spravk\_butt->Name = L"spravk\_butt";

this->spravk\_butt->Size = System::Drawing::Size(28, 32);

this->spravk\_butt->TabIndex = 6;

this->spravk\_butt->Text = L"\?";

this->spravk\_butt->UseVisualStyleBackColor = true;

this->spravk\_butt->Click += gcnew System::EventHandler(this, &Form1::spravk\_butt\_Click);

//

// info\_avtor

//

this->info\_avtor->AutoSize = true;

this->info\_avtor->BackColor = System::Drawing::Color::Transparent;

this->info\_avtor->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->info\_avtor->ForeColor = System::Drawing::SystemColors::ControlLightLight;

this->info\_avtor->Location = System::Drawing::Point(3, 3);

this->info\_avtor->Name = L"info\_avtor";

this->info\_avtor->Size = System::Drawing::Size(224, 15);

this->info\_avtor->TabIndex = 7;

this->info\_avtor->Text = L"© Гулин Алексей, ПИ-92. АлтГТУ 2020";

//

// Form1

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

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

this->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(224)), static\_cast<System::Int32>(static\_cast<System::Byte>(224)),

static\_cast<System::Int32>(static\_cast<System::Byte>(224)));

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

this->CancelButton = this->quit\_butt;

this->ClientSize = System::Drawing::Size(346, 387);

this->ControlBox = false;

this->Controls->Add(this->info\_avtor);

this->Controls->Add(this->spravk\_butt);

this->Controls->Add(this->quit\_butt);

this->Controls->Add(this->records\_butt);

this->Controls->Add(this->load\_game\_butt);

this->Controls->Add(this->new\_game\_butt);

this->Controls->Add(this->glavn\_menu\_text);

this->Controls->Add(this->igra\_menu\_text);

this->HelpButton = true;

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

this->MaximizeBox = false;

this->MaximumSize = System::Drawing::Size(362, 426);

this->MinimizeBox = false;

this->MinimumSize = System::Drawing::Size(362, 426);

this->Name = L"Form1";

this->RightToLeftLayout = true;

this->ShowIcon = false;

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

this->Text = L"Главное меню";

this->Activated += gcnew System::EventHandler(this, &Form1::Form1\_Activated);

this->Load += gcnew System::EventHandler(this, &Form1::Form1\_Load);

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void Form1\_Load(System::Object^ sender, System::EventArgs^ e) {

srand(time(NULL)); //автоматическая рандомизация генератора случайных чисел в зависимости от времени (чтобы всегда давал разный результат)

}

private: System::Void label1\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void label1\_Click\_1(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

f\_endgame=0; //флаг конца игры (будет равен 1, когда игрок проиграет)

derevn.Init\_new\_game(); //инициализация деревни перед началом новой игры

main\_game^ main\_game\_p = gcnew main\_game(); //указатель на форму

main\_game\_p -> ShowDialog(); //открыть форму

delete(main\_game\_p);

}

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

f\_endgame=0; //флаг конца игры (будет равен 1, когда игрок проиграет)

derevn.Init\_load\_game(); //инициализация деревни из файла перед продолжением сохранённой игры

main\_game^ main\_game\_p = gcnew main\_game(); //указатель на форму

main\_game\_p -> ShowDialog(); //открыть форму

delete(main\_game\_p);

}

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

record\_tabl^ record\_tabl\_p = gcnew record\_tabl(); //указатель на форму

record\_tabl\_p -> ShowDialog(); //открыть форму

delete(record\_tabl\_p);

}

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

exit(0); //выход из программы

}

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

spravk^ spravk\_p = gcnew spravk(); //указатель на форму

spravk\_p -> ShowDialog(); //открыть форму

delete(spravk\_p);

}

private: System::Void Form1\_Activated(System::Object^ sender, System::EventArgs^ e) {

if(f\_endgame==1)

{

f\_endgame=0;

itog\_game^ itog\_game\_p = gcnew itog\_game(); //указатель на форму

itog\_game\_p -> Show(); //открыть форму

}

ifstream iff("village.txt");

ifstream iff2("krests.txt");

if(!iff.is\_open() || !iff2.is\_open()) //проверка, есть ли файлы сохранения. Если хотя бы одного нет, то кнопка загрузки игры отключается

{

this->load\_game\_butt->Enabled=false;

}

else

{

this->load\_game\_butt->Enabled=true;

}

iff.close();

iff2.close();

}

};

}

main\_game.h

#pragma once

#include <ctime>

#include "tabl\_krest.h"

#include "krest\_and\_village.h"

#include "sell\_tovar.h"

#include "spravk.h"

extern int f\_endgame; //флаг о том, что игра окончена (игрок проиграл) (глобальн.)

namespace kurs2020 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для main\_game

/// </summary>

public ref class main\_game : public System::Windows::Forms::Form

{

public:

main\_game(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~main\_game()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::MenuStrip^ lent\_menu;

protected:

private: System::Windows::Forms::ToolStripMenuItem^ save\_game\_butt;

private: System::Windows::Forms::ToolStripMenuItem^ spravk\_lent\_butt;

private: System::Windows::Forms::ToolStripMenuItem^ go\_to\_main\_menu\_butt;

private: System::Windows::Forms::Label^ skorost\_zhiz\_text;

private: System::Windows::Forms::Label^ den\_procv\_text;

private: System::Windows::Forms::Label^ kolvo\_krest\_text;

private: System::Windows::Forms::Label^ budget\_text;

private: System::Windows::Forms::Label^ hleb\_text;

private: System::Windows::Forms::Label^ skot\_text;

private: System::Windows::Forms::GroupBox^ pomoshnik\_box;

private: System::Windows::Forms::Button^ spisok\_krest\_butt;

private: System::Windows::Forms::Button^ birzha\_butt;

private: System::Windows::Forms::Label^ vihod\_info;

private: System::Windows::Forms::Label^ event\_helper;

private: System::Windows::Forms::Label^ time\_helper;

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

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

private: System::Windows::Forms::Label^ save\_complete;

private: System::Windows::Forms::Timer^ main\_timer;

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

private:

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

#pragma region Windows Form Designer generated code

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

void InitializeComponent(void)

{

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

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(main\_game::typeid));

this->lent\_menu = (gcnew System::Windows::Forms::MenuStrip());

this->save\_game\_butt = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->spravk\_lent\_butt = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->go\_to\_main\_menu\_butt = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->skorost\_zhiz\_text = (gcnew System::Windows::Forms::Label());

this->den\_procv\_text = (gcnew System::Windows::Forms::Label());

this->kolvo\_krest\_text = (gcnew System::Windows::Forms::Label());

this->budget\_text = (gcnew System::Windows::Forms::Label());

this->hleb\_text = (gcnew System::Windows::Forms::Label());

this->skot\_text = (gcnew System::Windows::Forms::Label());

this->pomoshnik\_box = (gcnew System::Windows::Forms::GroupBox());

this->event\_helper = (gcnew System::Windows::Forms::Label());

this->time\_helper = (gcnew System::Windows::Forms::Label());

this->spisok\_krest\_butt = (gcnew System::Windows::Forms::Button());

this->birzha\_butt = (gcnew System::Windows::Forms::Button());

this->vihod\_info = (gcnew System::Windows::Forms::Label());

this->main\_timer = (gcnew System::Windows::Forms::Timer(this->components));

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

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

this->save\_complete = (gcnew System::Windows::Forms::Label());

this->lent\_menu->SuspendLayout();

this->pomoshnik\_box->SuspendLayout();

this->SuspendLayout();

//

// lent\_menu

//

this->lent\_menu->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::Top | System::Windows::Forms::AnchorStyles::Right));

this->lent\_menu->Dock = System::Windows::Forms::DockStyle::None;

this->lent\_menu->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(3) {this->save\_game\_butt, this->spravk\_lent\_butt,

this->go\_to\_main\_menu\_butt});

this->lent\_menu->Location = System::Drawing::Point(472, 0);

this->lent\_menu->Name = L"lent\_menu";

this->lent\_menu->RightToLeft = System::Windows::Forms::RightToLeft::No;

this->lent\_menu->Size = System::Drawing::Size(372, 24);

this->lent\_menu->TabIndex = 0;

this->lent\_menu->Text = L"lent\_menu";

//

// save\_game\_butt

//

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

this->save\_game\_butt->Name = L"save\_game\_butt";

this->save\_game\_butt->Size = System::Drawing::Size(122, 20);

this->save\_game\_butt->Text = L"Сохранить игру";

this->save\_game\_butt->Click += gcnew System::EventHandler(this, &main\_game::save\_game\_butt\_Click);

//

// spravk\_lent\_butt

//

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

this->spravk\_lent\_butt->Name = L"spravk\_lent\_butt";

this->spravk\_lent\_butt->Size = System::Drawing::Size(81, 20);

this->spravk\_lent\_butt->Text = L"Справка";

this->spravk\_lent\_butt->Click += gcnew System::EventHandler(this, &main\_game::spravk\_lent\_butt\_Click);

//

// go\_to\_main\_menu\_butt

//

this->go\_to\_main\_menu\_butt->Image = (cli::safe\_cast<System::Drawing::Image^ >(resources->GetObject(L"go\_to\_main\_menu\_butt.Image")));

this->go\_to\_main\_menu\_butt->Name = L"go\_to\_main\_menu\_butt";

this->go\_to\_main\_menu\_butt->Size = System::Drawing::Size(161, 20);

this->go\_to\_main\_menu\_butt->Text = L"Выход в главное меню";

this->go\_to\_main\_menu\_butt->Click += gcnew System::EventHandler(this, &main\_game::go\_to\_main\_menu\_butt\_Click);

//

// skorost\_zhiz\_text

//

this->skorost\_zhiz\_text->AutoSize = true;

this->skorost\_zhiz\_text->BorderStyle = System::Windows::Forms::BorderStyle::Fixed3D;

this->skorost\_zhiz\_text->Font = (gcnew System::Drawing::Font(L"Georgia", 18, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->skorost\_zhiz\_text->Location = System::Drawing::Point(12, 50);

this->skorost\_zhiz\_text->Name = L"skorost\_zhiz\_text";

this->skorost\_zhiz\_text->Size = System::Drawing::Size(212, 31);

this->skorost\_zhiz\_text->TabIndex = 1;

this->skorost\_zhiz\_text->Text = L"Скорость жизни: ";

//

// den\_procv\_text

//

this->den\_procv\_text->AutoSize = true;

this->den\_procv\_text->BorderStyle = System::Windows::Forms::BorderStyle::Fixed3D;

this->den\_procv\_text->Font = (gcnew System::Drawing::Font(L"Georgia", 18, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->den\_procv\_text->Location = System::Drawing::Point(12, 91);

this->den\_procv\_text->Name = L"den\_procv\_text";

this->den\_procv\_text->Size = System::Drawing::Size(232, 31);

this->den\_procv\_text->TabIndex = 2;

this->den\_procv\_text->Text = L"Дни Процветания: ";

//

// kolvo\_krest\_text

//

this->kolvo\_krest\_text->AutoSize = true;

this->kolvo\_krest\_text->BorderStyle = System::Windows::Forms::BorderStyle::Fixed3D;

this->kolvo\_krest\_text->Font = (gcnew System::Drawing::Font(L"Georgia", 18, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->kolvo\_krest\_text->Location = System::Drawing::Point(12, 134);

this->kolvo\_krest\_text->Name = L"kolvo\_krest\_text";

this->kolvo\_krest\_text->Size = System::Drawing::Size(146, 31);

this->kolvo\_krest\_text->TabIndex = 3;

this->kolvo\_krest\_text->Text = L"Крестьяне: ";

this->kolvo\_krest\_text->Click += gcnew System::EventHandler(this, &main\_game::label1\_Click);

//

// budget\_text

//

this->budget\_text->AutoSize = true;

this->budget\_text->BorderStyle = System::Windows::Forms::BorderStyle::Fixed3D;

this->budget\_text->Font = (gcnew System::Drawing::Font(L"Georgia", 18, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->budget\_text->Location = System::Drawing::Point(12, 178);

this->budget\_text->Name = L"budget\_text";

this->budget\_text->Size = System::Drawing::Size(164, 31);

this->budget\_text->TabIndex = 4;

this->budget\_text->Text = L"Бюджет: руб.";

//

// hleb\_text

//

this->hleb\_text->AutoSize = true;

this->hleb\_text->BorderStyle = System::Windows::Forms::BorderStyle::Fixed3D;

this->hleb\_text->Font = (gcnew System::Drawing::Font(L"Georgia", 18, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->hleb\_text->Location = System::Drawing::Point(12, 220);

this->hleb\_text->Name = L"hleb\_text";

this->hleb\_text->Size = System::Drawing::Size(85, 31);

this->hleb\_text->TabIndex = 5;

this->hleb\_text->Text = L"Хлеб: ";

//

// skot\_text

//

this->skot\_text->AutoSize = true;

this->skot\_text->BorderStyle = System::Windows::Forms::BorderStyle::Fixed3D;

this->skot\_text->Font = (gcnew System::Drawing::Font(L"Georgia", 18, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->skot\_text->Location = System::Drawing::Point(12, 262);

this->skot\_text->Name = L"skot\_text";

this->skot\_text->Size = System::Drawing::Size(208, 31);

this->skot\_text->TabIndex = 6;

this->skot\_text->Text = L"Домашний скот: ";

//

// pomoshnik\_box

//

this->pomoshnik\_box->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>(((System::Windows::Forms::AnchorStyles::Bottom | System::Windows::Forms::AnchorStyles::Left)

| System::Windows::Forms::AnchorStyles::Right));

this->pomoshnik\_box->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(255)), static\_cast<System::Int32>(static\_cast<System::Byte>(255)),

static\_cast<System::Int32>(static\_cast<System::Byte>(200)));

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

this->pomoshnik\_box->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Stretch;

this->pomoshnik\_box->Controls->Add(this->event\_helper);

this->pomoshnik\_box->Controls->Add(this->time\_helper);

this->pomoshnik\_box->Font = (gcnew System::Drawing::Font(L"Calibri", 12.75F, System::Drawing::FontStyle::Italic, System::Drawing::GraphicsUnit::Point,

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

this->pomoshnik\_box->Location = System::Drawing::Point(12, 389);

this->pomoshnik\_box->Name = L"pomoshnik\_box";

this->pomoshnik\_box->RightToLeft = System::Windows::Forms::RightToLeft::Yes;

this->pomoshnik\_box->Size = System::Drawing::Size(820, 100);

this->pomoshnik\_box->TabIndex = 7;

this->pomoshnik\_box->TabStop = false;

this->pomoshnik\_box->Text = L"Информация от Помощника";

//

// event\_helper

//

this->event\_helper->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>(((System::Windows::Forms::AnchorStyles::Bottom | System::Windows::Forms::AnchorStyles::Left)

| System::Windows::Forms::AnchorStyles::Right));

this->event\_helper->BackColor = System::Drawing::Color::Transparent;

this->event\_helper->Cursor = System::Windows::Forms::Cursors::Default;

this->event\_helper->Font = (gcnew System::Drawing::Font(L"Cambria", 14.25F, System::Drawing::FontStyle::Italic, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->event\_helper->Location = System::Drawing::Point(6, 51);

this->event\_helper->Name = L"event\_helper";

this->event\_helper->RightToLeft = System::Windows::Forms::RightToLeft::No;

this->event\_helper->Size = System::Drawing::Size(808, 46);

this->event\_helper->TabIndex = 1;

this->event\_helper->Text = L" ";

//

// time\_helper

//

this->time\_helper->AutoSize = true;

this->time\_helper->BackColor = System::Drawing::Color::Transparent;

this->time\_helper->Font = (gcnew System::Drawing::Font(L"Cambria", 14.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->time\_helper->Location = System::Drawing::Point(6, 24);

this->time\_helper->Name = L"time\_helper";

this->time\_helper->RightToLeft = System::Windows::Forms::RightToLeft::No;

this->time\_helper->Size = System::Drawing::Size(540, 22);

this->time\_helper->TabIndex = 0;

this->time\_helper->Text = L"До конца этого времени года осталось 11 Часов Процветания.";

//

// spisok\_krest\_butt

//

this->spisok\_krest\_butt->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::Top | System::Windows::Forms::AnchorStyles::Right));

this->spisok\_krest\_butt->Font = (gcnew System::Drawing::Font(L"DejaVu Serif Condensed", 14.25F, static\_cast<System::Drawing::FontStyle>((System::Drawing::FontStyle::Bold | System::Drawing::FontStyle::Italic)),

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

this->spisok\_krest\_butt->Location = System::Drawing::Point(612, 83);

this->spisok\_krest\_butt->Name = L"spisok\_krest\_butt";

this->spisok\_krest\_butt->Size = System::Drawing::Size(220, 48);

this->spisok\_krest\_butt->TabIndex = 3;

this->spisok\_krest\_butt->TabStop = false;

this->spisok\_krest\_butt->Text = L"Список крестьян";

this->spisok\_krest\_butt->UseVisualStyleBackColor = true;

this->spisok\_krest\_butt->Click += gcnew System::EventHandler(this, &main\_game::spisok\_krest\_butt\_Click);

//

// birzha\_butt

//

this->birzha\_butt->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::Top | System::Windows::Forms::AnchorStyles::Right));

this->birzha\_butt->Font = (gcnew System::Drawing::Font(L"DejaVu Serif Condensed", 14.25F, static\_cast<System::Drawing::FontStyle>((System::Drawing::FontStyle::Bold | System::Drawing::FontStyle::Italic)),

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

this->birzha\_butt->Location = System::Drawing::Point(612, 147);

this->birzha\_butt->Name = L"birzha\_butt";

this->birzha\_butt->Size = System::Drawing::Size(220, 48);

this->birzha\_butt->TabIndex = 8;

this->birzha\_butt->TabStop = false;

this->birzha\_butt->Text = L"Товарная биржа";

this->birzha\_butt->UseVisualStyleBackColor = true;

this->birzha\_butt->Click += gcnew System::EventHandler(this, &main\_game::birzha\_butt\_Click);

//

// vihod\_info

//

this->vihod\_info->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::Top | System::Windows::Forms::AnchorStyles::Right));

this->vihod\_info->AutoSize = true;

this->vihod\_info->Location = System::Drawing::Point(621, 28);

this->vihod\_info->Name = L"vihod\_info";

this->vihod\_info->Size = System::Drawing::Size(223, 13);

this->vihod\_info->TabIndex = 9;

this->vihod\_info->Text = L"Не забудь сохранить игру перед выходом!";

//

// main\_timer

//

this->main\_timer->Enabled = true;

this->main\_timer->Interval = 3000;

this->main\_timer->Tick += gcnew System::EventHandler(this, &main\_game::main\_timer\_Tick);

//

// label1

//

this->label1->AutoSize = true;

this->label1->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(210)), static\_cast<System::Int32>(static\_cast<System::Byte>(210)),

static\_cast<System::Int32>(static\_cast<System::Byte>(210)));

this->label1->Location = System::Drawing::Point(2, 2);

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

this->label1->Size = System::Drawing::Size(227, 26);

this->label1->TabIndex = 10;

this->label1->Text = L"1 Час Процветания = 3 реальные секунды\r\n1 День Процветания = 30 реальных секунд";

//

// timer\_refresh

//

this->timer\_refresh->Enabled = true;

this->timer\_refresh->Interval = 512;

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

//

// save\_complete

//

this->save\_complete->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::Top | System::Windows::Forms::AnchorStyles::Right));

this->save\_complete->AutoSize = true;

this->save\_complete->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(192)), static\_cast<System::Int32>(static\_cast<System::Byte>(255)),

static\_cast<System::Int32>(static\_cast<System::Byte>(192)));

this->save\_complete->Location = System::Drawing::Point(474, 28);

this->save\_complete->Name = L"save\_complete";

this->save\_complete->Size = System::Drawing::Size(137, 13);

this->save\_complete->TabIndex = 11;

this->save\_complete->Text = L"Игра успешно сохранена!";

this->save\_complete->Visible = false;

//

// main\_game

//

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::None;

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

this->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Stretch;

this->ClientSize = System::Drawing::Size(844, 501);

this->Controls->Add(this->save\_complete);

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

this->Controls->Add(this->vihod\_info);

this->Controls->Add(this->birzha\_butt);

this->Controls->Add(this->spisok\_krest\_butt);

this->Controls->Add(this->pomoshnik\_box);

this->Controls->Add(this->skot\_text);

this->Controls->Add(this->hleb\_text);

this->Controls->Add(this->budget\_text);

this->Controls->Add(this->kolvo\_krest\_text);

this->Controls->Add(this->den\_procv\_text);

this->Controls->Add(this->skorost\_zhiz\_text);

this->Controls->Add(this->lent\_menu);

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

this->MainMenuStrip = this->lent\_menu;

this->MinimizeBox = false;

this->MinimumSize = System::Drawing::Size(860, 540);

this->Name = L"main\_game";

this->Text = L"Игра \"Управление деревней\"";

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

this->Load += gcnew System::EventHandler(this, &main\_game::main\_game\_Load);

this->lent\_menu->ResumeLayout(false);

this->lent\_menu->PerformLayout();

this->pomoshnik\_box->ResumeLayout(false);

this->pomoshnik\_box->PerformLayout();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void go\_to\_main\_menu\_butt\_Click(System::Object^ sender, System::EventArgs^ e) {

Close();

}

private: System::Void label1\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void spisok\_krest\_butt\_Click(System::Object^ sender, System::EventArgs^ e) {

tabl\_krest^ tabl\_krest\_p = gcnew tabl\_krest(); //указатель на форму

tabl\_krest\_p -> ShowDialog(); //открыть форму

delete(tabl\_krest\_p);

}

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

sell\_tovar^ sell\_tovar\_p = gcnew sell\_tovar(); //указатель на форму

sell\_tovar\_p -> ShowDialog(); //открыть форму

delete(sell\_tovar\_p);

}

private: System::Void spravk\_lent\_butt\_Click(System::Object^ sender, System::EventArgs^ e) {

spravk^ spravk\_p = gcnew spravk(); //указатель на форму

spravk\_p -> ShowDialog(); //открыть форму

delete(spravk\_p);

}

private: System::Void main\_game\_Load(System::Object^ sender, System::EventArgs^ e) {

this->kolvo\_krest\_text->Text = L"Крестьяне: " + Convert::ToString(derevn.get\_kolvo\_krest())+ L" чел.";;

this->skorost\_zhiz\_text->Text = L"Скорость жизни: " + Convert::ToString(derevn.get\_speed\_life());

this->den\_procv\_text->Text = L"Дни Процветания: " + Convert::ToString(derevn.get\_days\_procv());

this->budget\_text->Text = L"Бюджет: " + Convert::ToString(derevn.get\_budget\_village()) + L" руб.";

this->hleb\_text->Text = L"Хлеб: " + Convert::ToString(derevn.get\_kolvo\_hleb()) + L" ед.";

this->skot\_text->Text = L"Домашний скот: " + Convert::ToString(derevn.get\_kolvo\_skot()) + L" ед.";

if(derevn.get\_flag\_season()==0)

this->time\_helper->Text = L"До конца Лета осталось " + Convert::ToString(40 -(derevn.get\_hours\_procv() - derevn.get\_hours\_from\_begin\_of\_season())) + L" Часов Процветания.";

if(derevn.get\_flag\_season()==1)

{

this->time\_helper->Text = L"До конца Зимы осталось " + Convert::ToString(20 -(derevn.get\_hours\_procv() - derevn.get\_hours\_from\_begin\_of\_season())) + L" Часов Процветания.";

this->BackgroundImage = Image::FromFile("zima.jpg"); //изменить фон на зимний

}

/\* //вывод текущего времени в строку события (час:мин:сек). Понадобится в другой части кода

time\_t now = time(0);

tm \*ltm = localtime(&now);

this->event\_helper->Text = L"Последнее событие: " + Convert::ToString(ltm->tm\_hour)+ L":" + Convert::ToString(ltm->tm\_min)+ L":" + Convert::ToString(ltm->tm\_sec) + L" ";

\*/

}

private: System::Void main\_timer\_Tick(System::Object^ sender, System::EventArgs^ e) {

this->save\_complete->Visible = false;

if(f\_endgame==1) //игра закончилась, закрыть все формы

Close();

derevn.set\_hours\_procv(derevn.get\_hours\_procv()+1); //кол-во часов процветания увеличилось на 1

if((derevn.get\_hours\_procv()%10)==0) //если прошло 10 часов процветания

{

derevn.set\_days\_procv(derevn.get\_days\_procv()+1);

}

if(derevn.get\_kolvo\_krest()==0) //если нет крестьян

{

derevn.set\_otschet\_hours\_net\_krest(derevn.get\_otschet\_hours\_net\_krest()+1);

}

if(derevn.get\_otschet\_hours\_net\_krest()==2) //если закончилось время отсчёта без крестьян

{

f\_endgame=1;

Close();

}

if(derevn.get\_kolvo\_krest()!=0) //если хотя бы один крестьянин нашёлся

{

derevn.set\_otschet\_hours\_net\_krest(0);

}

if(derevn.get\_flag\_season()==0 && derevn.get\_budget\_village()!=0) //если лето и бюджет не нулевой

{

int kolv\_k=derevn.get\_kolvo\_krest();

for(int ko=0; ko<kolv\_k; ko++)

{

if((derevn.get\_budget\_village()-(derevn.krests[ko].get\_money\_trat()\*derevn.get\_speed\_life()))>=0)

{

derevn.set\_kolvo\_hleb(derevn.get\_kolvo\_hleb()+derevn.krests[ko].get\_proizv\_hleb());

derevn.set\_kolvo\_skot(derevn.get\_kolvo\_skot()+derevn.krests[ko].get\_proizv\_skot());

derevn.set\_budget\_village(derevn.get\_budget\_village()-(derevn.krests[ko].get\_money\_trat()\*derevn.get\_speed\_life()));

}

else

break;

}

}

if(derevn.get\_flag\_season()==1) //если зима

{

int kolv\_k=derevn.get\_kolvo\_krest();

for(int ko=0; ko<kolv\_k; ko++)

{

derevn.set\_kolvo\_hleb(derevn.get\_kolvo\_hleb()-(derevn.krests[ko].get\_eda\_hleb()\*derevn.get\_speed\_life()));

derevn.set\_kolvo\_skot(derevn.get\_kolvo\_skot()-(derevn.krests[ko].get\_eda\_skot()\*derevn.get\_speed\_life()));

if(derevn.get\_kolvo\_hleb()>=0 && derevn.get\_kolvo\_skot()>=0)

derevn.krests[ko].set\_otschet\_hours\_net\_edi(0);

if(derevn.get\_kolvo\_hleb()<0 || derevn.get\_kolvo\_skot()<0)

{

if(derevn.get\_kolvo\_hleb()<0)

{

derevn.set\_kolvo\_hleb(0);

}

if(derevn.get\_kolvo\_skot()<0)

{

derevn.set\_kolvo\_skot(0);

}

derevn.krests[ko].set\_otschet\_hours\_net\_edi(derevn.krests[ko].get\_otschet\_hours\_net\_edi()+1);

if(derevn.krests[ko].get\_otschet\_hours\_net\_edi()>=5)

{

time\_t now = time(0); //для вывода времени

tm \*ltm = localtime(&now);

string str\_name= derevn.krests[ko].get\_name() + " " + derevn.krests[ko].get\_surname(); //для вывода имени и фамилии

this->event\_helper->Text = L"Последнее событие: " + Convert::ToString(ltm->tm\_hour)+ L":" + Convert::ToString(ltm->tm\_min)+ L":" + Convert::ToString(ltm->tm\_sec) + L" " + L"Крестьянин " + gcnew System::String(str\_name.c\_str()) + L" ушёл из деревни из-за недостатка пищи.";

derevn.Delete\_krest(ko+1); //удалить крестьянина (методу передаётся нумерация от единицы, а цикл нумеруется от нуля)

kolv\_k=kolv\_k-1; //крестьян стало на 1 меньше

if(kolv\_k==ko)

break;

}

}

}

}

if(derevn.get\_flag\_season()==0 && (derevn.get\_hours\_procv() - derevn.get\_hours\_from\_begin\_of\_season())==40) //если сейчас лето и прошло 40 часов П. от начала лета (4 дня П. или 2 минуты)

{

derevn.set\_flag\_season(1); //теперь флаг соответствует зиме

this->BackgroundImage = Image::FromFile("zima.jpg"); //изменить фон на зимний

derevn.set\_hours\_from\_begin\_of\_season(derevn.get\_hours\_procv());

}

if(derevn.get\_flag\_season()==1 && (derevn.get\_hours\_procv() - derevn.get\_hours\_from\_begin\_of\_season())==20) //если сейчас зима и прошло 20 часов П. от начала зимы (2 дня П. или 1 минута)

{

derevn.set\_flag\_season(0); //теперь флаг соответствует лету

this->BackgroundImage = Image::FromFile("leto.jpg"); //изменить фон на летний

derevn.set\_hours\_from\_begin\_of\_season(derevn.get\_hours\_procv());

derevn.set\_speed\_life( derevn.get\_speed\_life()+1); //множитель Скорость жизни увеличился на 1

int f\_izmen\_price; //флаг изменения цены на товар (0, если уменьшить; 1, если увеличить)

double proc\_izm; //процент от начального значения цены, на которое она изменится

//ДЛЯ ЦЕНЫ НА ХЛЕБ

f\_izmen\_price =(1 + rand() % 2) -1; //случайное число в интервале от 0 до 1 включительно

proc\_izm = 5 + rand() % 25; //случайное число в интервале от 5 до 30 включительно

proc\_izm = proc\_izm / 100; //перевод процентного значения в десятичное

if(f\_izmen\_price==0)

{

derevn.set\_price\_hleb(derevn.get\_price\_hleb() - derevn.get\_price\_hleb() \* proc\_izm);

}

if(f\_izmen\_price==1)

{

derevn.set\_price\_hleb(derevn.get\_price\_hleb() + derevn.get\_price\_hleb() \* proc\_izm);

}

//ДЛЯ ЦЕНЫ НА СКОТ

f\_izmen\_price =(1 + rand() % 2) -1; //случайное число в интервале от 0 до 1 включительно

proc\_izm = 5 + rand() % 25; //случайное число в интервале от 5 до 30 включительно

proc\_izm = proc\_izm / 100; //перевод процентного значения в десятичное

if(f\_izmen\_price==0)

{

derevn.set\_price\_skot(derevn.get\_price\_skot() - derevn.get\_price\_skot() \* proc\_izm);

}

if(f\_izmen\_price==1)

{

derevn.set\_price\_skot(derevn.get\_price\_skot() + derevn.get\_price\_skot() \* proc\_izm);

}

}

if(derevn.get\_flag\_season()==1 && (derevn.get\_hours\_procv() - derevn.get\_hours\_from\_begin\_of\_season())==10) //если сейчас зима и прошло 10 часов П. от начала зимы (середина зимы) (1 день П. или 0,5 минуты)

{

int f\_event; //флаг случайного события (0 - приход нового крестьянина, 1 - нападение волков на скот)

f\_event =(1 + rand() % 2) -1; //случайное число в интервале от 0 до 1 включительно

double proc\_izm; //процент от начального значения кол-ва скота, на которое оно изменится

proc\_izm = 5 + rand() % 25; //случайное число в интервале от 5 до 30 включительно

proc\_izm = proc\_izm / 100; //перевод процентного значения в десятичное

int poteri = derevn.get\_kolvo\_skot() \* proc\_izm; //отдельная переменная для кол-ва потерянного скота

if(f\_event==1 && poteri>0) //нападение волков

{

derevn.set\_kolvo\_skot(derevn.get\_kolvo\_skot() - poteri);

time\_t now = time(0); //для вывода времени

tm \*ltm = localtime(&now);

this->event\_helper->Text = L"Последнее событие: " + Convert::ToString(ltm->tm\_hour)+ L":" + Convert::ToString(ltm->tm\_min)+ L":" + Convert::ToString(ltm->tm\_sec) + L" " + L"Стая волков из леса напала на ваш скот. Потеряно " + Convert::ToString(poteri) + L" ед. скота.";

}

else //приход крестьянина

{

derevn.Add\_rand\_krest(); //добавление нового случайного крестьянина

time\_t now = time(0); //для вывода времени

tm \*ltm = localtime(&now);

int koo = derevn.get\_kolvo\_krest() - 1; //получить номер последнего крестьянина

string str\_name= derevn.krests[koo].get\_name() + " " + derevn.krests[koo].get\_surname(); //для вывода имени и фамилии

this->event\_helper->Text = L"Последнее событие: " + Convert::ToString(ltm->tm\_hour)+ L":" + Convert::ToString(ltm->tm\_min)+ L":" + Convert::ToString(ltm->tm\_sec) + L" " + L"Крестьянин " + gcnew System::String(str\_name.c\_str()) + L" вышел из лесу и присоединился к вам.";

}

}

//ВЫВОД ИНФОРМАЦИИ О ДЕРЕВНЕ

this->kolvo\_krest\_text->Text = L"Крестьяне: " + Convert::ToString(derevn.get\_kolvo\_krest())+ L" чел.";;

this->skorost\_zhiz\_text->Text = L"Скорость жизни: " + Convert::ToString(derevn.get\_speed\_life());

this->den\_procv\_text->Text = L"Дни Процветания: " + Convert::ToString(derevn.get\_days\_procv());

this->budget\_text->Text = L"Бюджет: " + Convert::ToString(derevn.get\_budget\_village()) + L" руб.";

this->hleb\_text->Text = L"Хлеб: " + Convert::ToString(derevn.get\_kolvo\_hleb()) + L" ед.";

this->skot\_text->Text = L"Домашний скот: " + Convert::ToString(derevn.get\_kolvo\_skot()) + L" ед.";

if(derevn.get\_flag\_season()==0)

this->time\_helper->Text = L"До конца Лета осталось " + Convert::ToString(40 -(derevn.get\_hours\_procv() - derevn.get\_hours\_from\_begin\_of\_season())) + L" Часов Процветания.";

if(derevn.get\_flag\_season()==1)

this->time\_helper->Text = L"До конца Зимы осталось " + Convert::ToString(20 -(derevn.get\_hours\_procv() - derevn.get\_hours\_from\_begin\_of\_season())) + L" Часов Процветания.";

}

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

//ВЫВОД ИНФОРМАЦИИ О ДЕРЕВНЕ

this->kolvo\_krest\_text->Text = L"Крестьяне: " + Convert::ToString(derevn.get\_kolvo\_krest())+ L" чел.";;

this->skorost\_zhiz\_text->Text = L"Скорость жизни: " + Convert::ToString(derevn.get\_speed\_life());

this->den\_procv\_text->Text = L"Дни Процветания: " + Convert::ToString(derevn.get\_days\_procv());

this->budget\_text->Text = L"Бюджет: " + Convert::ToString(derevn.get\_budget\_village()) + L" руб.";

this->hleb\_text->Text = L"Хлеб: " + Convert::ToString(derevn.get\_kolvo\_hleb()) + L" ед.";

this->skot\_text->Text = L"Домашний скот: " + Convert::ToString(derevn.get\_kolvo\_skot()) + L" ед.";

if(derevn.get\_flag\_season()==0)

this->time\_helper->Text = L"До конца Лета осталось " + Convert::ToString(40 -(derevn.get\_hours\_procv() - derevn.get\_hours\_from\_begin\_of\_season())) + L" Часов Процветания.";

if(derevn.get\_flag\_season()==1)

this->time\_helper->Text = L"До конца Зимы осталось " + Convert::ToString(20 -(derevn.get\_hours\_procv() - derevn.get\_hours\_from\_begin\_of\_season())) + L" Часов Процветания.";

}

private: System::Void save\_game\_butt\_Click(System::Object^ sender, System::EventArgs^ e) {

derevn.Save\_game(); //сохранить игру в файл

this->save\_complete->Visible = true; //показать надпись об успешном сохранении

}

};

}

krest\_and\_village.h

// препроцессорная обёртка (чтобы не возникало ошибок «переопределение типа "class"»)

#ifndef KREST\_AND\_VILLAGE\_H // если имя KREST\_AND\_VILLAGE\_H ещё не определено

#define KREST\_AND\_VILLAGE\_H // определить имя KREST\_AND\_VILLAGE\_H

#include <string>

using namespace std;

public class Krest //класс крестьянина

{

private:

string name; //Имя

string surname; //Фамилия

int proizv\_hleb; //Кол-во производимого хлеба за Час Процветания

int proizv\_skot; //Кол-во производимого домашнего скота за Час Процветания

int eda\_hleb; //Прожорливость: хлеб

int eda\_skot; //Прожорливость: домашний скот

int money\_trat; //Денежные расходы за один Час Процветания

int naim; //Стоимость найма в рублях

int otschet\_hours\_net\_edi; //Отсчёт времени в Часах Процветания с момента, когда закончилась еда для этого крестьянина

public:

Krest(); //конструктор без параметров

void Init\_rand(); //инициализация полей случайно (в фиксированном диапазоне) для новых крестьян

//Получение и установление соответствующих полей для крестьянина

void set\_name(string in);

string get\_name();

/////

void set\_surname(string in);

string get\_surname();

/////

void set\_proizv\_hleb(int in);

int get\_proizv\_hleb();

/////

void set\_proizv\_skot(int in);

int get\_proizv\_skot();

/////

void set\_eda\_hleb(int in);

int get\_eda\_hleb();

/////

void set\_eda\_skot(int in);

int get\_eda\_skot();

/////

void set\_money\_trat(int in);

int get\_money\_trat();

/////

void set\_naim(int in);

int get\_naim();

/////

void set\_otschet\_hours\_net\_edi(int in);

int get\_otschet\_hours\_net\_edi();

};

public class Village //класс деревни

{

private:

int kolvo\_krest; //Кол-во крестьян в деревне

int hours\_procv; //Кол-во Часов Процветания с момента начала игры

int days\_procv; //Кол-во Дней Процветания с момента начала игры (и финальный счёт)

int speed\_life; //Текущий множитель «Скорость Жизни»

int flag\_season; //Флаг текущего сезона (равен нулю Летом, равен единице Зимой)

int kolvo\_hleb; //Общее кол-во хлеба в деревне

int kolvo\_skot; //Общее кол-во голов домашнего скота в деревне

int budget\_village; //Бюджет деревни в рублях

int hours\_from\_begin\_of\_season; //Кол-во Часов Процветания в момент, когда текущее время года только началось

int otschet\_hours\_net\_krest; //Отсчёт времени в Часах Процветания с момента, когда ушли все крестьяне

int price\_hleb; //Текущая цена за ед. хлеба на Торговой бирже в рублях

int price\_skot; //Текущая цена за одну голову домашнего скота на Торговой бирже в рублях

public:

Krest\* krests; //крестьяне

Village(); //конструктор без параметров

void Init\_new\_game(); //инициализация при начале новой игры

void Save\_game(); //сохранение игры (запись информации о деревне в файл)

void Init\_load\_game(); //инициализация при загрузке сохранённой игры

void Delete\_krest(int number); //удаление крестьянина с выбранным номером (нумерация от единицы!)

void Add\_rand\_krest(); //добавление случайного крестьянина к имеющимся (особое событие, возможное зимой)

void Naim\_krest(Village vil\_naim, int numk); //найм крестьянина с выбранным номером из биржи труда (нумерация от единицы!)

//Получение и установление соответствующих полей для деревни

void set\_kolvo\_krest(int in);

int get\_kolvo\_krest();

/////

void set\_hours\_procv(int in);

int get\_hours\_procv();

/////

void set\_days\_procv(int in);

int get\_days\_procv();

/////

void set\_speed\_life(int in);

int get\_speed\_life();

/////

void set\_flag\_season(int in);

int get\_flag\_season();

/////

void set\_kolvo\_hleb(int in);

int get\_kolvo\_hleb();

/////

void set\_kolvo\_skot(int in);

int get\_kolvo\_skot();

/////

void set\_budget\_village(int in);

int get\_budget\_village();

/////

void set\_hours\_from\_begin\_of\_season(int in);

int get\_hours\_from\_begin\_of\_season();

/////

void set\_otschet\_hours\_net\_krest(int in);

int get\_otschet\_hours\_net\_krest();

/////

void set\_price\_hleb(int in);

int get\_price\_hleb();

/////

void set\_price\_skot(int in);

int get\_price\_skot();

};

extern Village derevn; //объект, в котором хранится вся информация о деревне (глобальн.)

extern Village birzh\_trud; //объект, в котором хранятся крестьяне для найма (глобальн.)

#endif KREST\_AND\_VILLAGE\_H // если имя KREST\_AND\_VILLAGE\_H уже определено, повторно не определять

krest\_and\_village.cpp

#include "stdafx.h"

#include "krest\_and\_village.h"

#include <fstream>

#include <iostream>

//РЕАЛИЗАЦИИ МЕТОДОВ КЛАССОВ (В ФАЙЛЕ .h ТОЛЬКО ИХ ПРОТОТИПЫ)

//ДЛЯ КЛАССА KREST

Krest::Krest() //конструктор без параметров

{

name="name";

surname="surname";

proizv\_hleb=0;

proizv\_skot=0;

eda\_hleb=0;

eda\_skot=0;

money\_trat=0;

naim=0;

otschet\_hours\_net\_edi=0;

}

void Krest::Init\_rand() //инициализация полей случайно (в фиксированном диапазоне) для новых крестьян

{

ifstream fin; //создать объект класса ifstream

//ПОЛУЧИТЬ СЛУЧАЙНОЕ ИМЯ ИЗ ФАЙЛА

fin.open("names.txt", ios::in); //открыть файл

int kolvo\_name=0; //начальное значение кол-ва слов в файле

while(!fin.eof()) //пока не прочитает весь файл

{

getline(fin, name, '\n'); //получить слово из файла

kolvo\_name+=1; //увеличить кол-во слов на 1

}

fin.close(); //закрыть файл

int num\_name = (1 + rand() % kolvo\_name) -2; //получить случайный номер слова из общего кол-ва

fin.open("names.txt", ios::in); //открыть файл

for (int i\_n=0; ; i\_n++)

{

getline(fin, name, '\n'); //получить слово из файла

if (i\_n==num\_name+1) //если это искомое слово, остановить цикл

break;

}

fin.close(); //закрыть файл

//ПОЛУЧИТЬ СЛУЧАЙНУЮ ФАМИЛИЮ ИЗ ФАЙЛА

fin.open("surnames.txt", ios::in); //открыть файл

int kolvo\_sur=0; //начальное значение кол-ва слов в файле

while(!fin.eof()) //пока не прочитает весь файл

{

getline(fin, surname, '\n'); //получить слово из файла

kolvo\_sur+=1; //увеличить кол-во слов на 1

}

fin.close(); //закрыть файл

int num\_sur = (1 + rand() % kolvo\_sur) -2; //получить случайный номер слова из общего кол-ва

fin.open("surnames.txt", ios::in); //открыть файл

for (int i\_nn=0; ; i\_nn++)

{

getline(fin, surname, '\n'); //получить слово из файла

if (i\_nn==num\_sur+1) //если это искомое слово, остановить цикл

break;

}

fin.close(); //закрыть файл

proizv\_hleb = 1 + rand() % 10; //случайное число в интервале от 1 до 10 включительно

proizv\_skot = (1 + rand() % 3) -1; //случайное число в интервале от 0 до 2 включительно

eda\_hleb = 1 + rand() % 3; //случайное число в интервале от 1 до 3 включительно

eda\_skot =(1 + rand() % 2) -1; //случайное число в интервале от 0 до 1 включительно

money\_trat = 10 + rand() % 90; //случайное число в интервале от 10 до 100 включительно

naim = 5 + rand() % 50; //случайное число в интервале от 5 до 50 включительно

otschet\_hours\_net\_edi=0;

}

//Получение и установление соответствующих полей для крестьянина

void Krest::set\_name(string in)

{

this->name=in;

}

string Krest::get\_name()

{

return name;

}

/////

void Krest::set\_surname(string in)

{

this->surname=in;

}

string Krest::get\_surname()

{

return surname;

}

/////

void Krest::set\_proizv\_hleb(int in)

{

this->proizv\_hleb=in;

}

int Krest::get\_proizv\_hleb()

{

return proizv\_hleb;

}

/////

void Krest::set\_proizv\_skot(int in)

{

this->proizv\_skot=in;

}

int Krest::get\_proizv\_skot()

{

return proizv\_skot;

}

/////

void Krest::set\_eda\_hleb(int in)

{

this->eda\_hleb=in;

}

int Krest::get\_eda\_hleb()

{

return eda\_hleb;

}

/////

void Krest::set\_eda\_skot(int in)

{

this->eda\_skot=in;

}

int Krest::get\_eda\_skot()

{

return eda\_skot;

}

/////

void Krest::set\_money\_trat(int in)

{

this->money\_trat=in;

}

int Krest::get\_money\_trat()

{

return money\_trat;

}

/////

void Krest::set\_naim(int in)

{

this->naim=in;

}

int Krest::get\_naim()

{

return naim;

}

/////

void Krest::set\_otschet\_hours\_net\_edi(int in)

{

this->otschet\_hours\_net\_edi=in;

}

int Krest::get\_otschet\_hours\_net\_edi()

{

return otschet\_hours\_net\_edi;

}

//ДЛЯ КЛАССА VILLAGE

Village::Village() //конструктор без параметров

{

kolvo\_krest=0;

hours\_procv=0;

days\_procv=0;

speed\_life=0;

flag\_season=0;

kolvo\_hleb=0;

kolvo\_skot=0;

budget\_village=0;

hours\_from\_begin\_of\_season=0;

otschet\_hours\_net\_krest=0;

price\_hleb=0;

price\_skot=0;

krests=0;

}

void Village::Init\_new\_game() //инициализация при начале новой игры

{

if(krests!=0)

{

delete [] krests;

}

kolvo\_krest=3;

hours\_procv=1;

days\_procv=0;

speed\_life=1;

flag\_season=0;

kolvo\_hleb=200;

kolvo\_skot=20;

budget\_village=1000;

hours\_from\_begin\_of\_season=0;

otschet\_hours\_net\_krest=0;

price\_hleb=30;

price\_skot=200;

krests = new Krest [3];

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

{

krests[i].Init\_rand();

}

}

void Village::Save\_game() //сохранение игры (запись информации о деревне в файл)

{

ofstream fof; //создать объект класса ofstream (для записи в файл информации о деревне) (крестьяне отдельно)

fof.open("village.txt", ios::out); //открыть файл для записи

fof << kolvo\_krest; //записать в файл

fof << '\n';

fof << hours\_procv; //записать в файл

fof << '\n';

fof << days\_procv; //записать в файл

fof << '\n';

fof << speed\_life; //записать в файл

fof << '\n';

fof << flag\_season; //записать в файл

fof << '\n';

fof << kolvo\_hleb; //записать в файл

fof << '\n';

fof << kolvo\_skot; //записать в файл

fof << '\n';

fof << budget\_village; //записать в файл

fof << '\n';

fof << hours\_from\_begin\_of\_season; //записать в файл

fof << '\n';

fof << otschet\_hours\_net\_krest; //записать в файл

fof << '\n';

fof << price\_hleb; //записать в файл

fof << '\n';

fof << price\_skot; //записать в файл

fof << '\n';

fof.close(); //закрыть файл

fof.open("krests.txt", ios::out); //открыть файл для записи

for(int m=0; m<kolvo\_krest; m++)

{

fof << krests[m].get\_name(); //записать в файл

fof << ';';

fof << krests[m].get\_surname(); //записать в файл

fof << ';';

fof << krests[m].get\_proizv\_hleb(); //записать в файл

fof << ';';

fof << krests[m].get\_proizv\_skot(); //записать в файл

fof << ';';

fof << krests[m].get\_eda\_hleb(); //записать в файл

fof << ';';

fof << krests[m].get\_eda\_skot(); //записать в файл

fof << ';';

fof << krests[m].get\_money\_trat(); //записать в файл

fof << ';';

fof << krests[m].get\_naim(); //записать в файл

fof << ';';

fof << krests[m].get\_otschet\_hours\_net\_edi(); //записать в файл

fof << '\n';

}

fof.close(); //закрыть файл

}

void Village::Init\_load\_game() //инициализация при загрузке сохранённой игры

{

if(krests!=0)

{

delete [] krests;

}

ifstream fin; //создать объект класса ifstream (для чтения информации о деревне из файла) (крестьяне отдельно)

string strread; //строка для чтения из файла

fin.open("village.txt", ios::in); //открыть файл

getline(fin, strread, '\n'); //прочитать из файла

kolvo\_krest=atoi(strread.c\_str()); //записать в нужное поле

getline(fin, strread, '\n'); //прочитать из файла

hours\_procv=atoi(strread.c\_str()); //записать в нужное поле

getline(fin, strread, '\n'); //прочитать из файла

days\_procv=atoi(strread.c\_str()); //записать в нужное поле

getline(fin, strread, '\n'); //прочитать из файла

speed\_life=atoi(strread.c\_str()); //записать в нужное поле

getline(fin, strread, '\n'); //прочитать из файла

flag\_season=atoi(strread.c\_str()); //записать в нужное поле

getline(fin, strread, '\n'); //прочитать из файла

kolvo\_hleb=atoi(strread.c\_str()); //записать в нужное поле

getline(fin, strread, '\n'); //прочитать из файла

kolvo\_skot=atoi(strread.c\_str()); //записать в нужное поле

getline(fin, strread, '\n'); //прочитать из файла

budget\_village=atoi(strread.c\_str()); //записать в нужное поле

getline(fin, strread, '\n'); //прочитать из файла

hours\_from\_begin\_of\_season=atoi(strread.c\_str()); //записать в нужное поле

getline(fin, strread, '\n'); //прочитать из файла

otschet\_hours\_net\_krest=atoi(strread.c\_str()); //записать в нужное поле

getline(fin, strread, '\n'); //прочитать из файла

price\_hleb=atoi(strread.c\_str()); //записать в нужное поле

getline(fin, strread, '\n'); //прочитать из файла

price\_skot=atoi(strread.c\_str()); //записать в нужное поле

fin.close(); //закрыть файл

krests = new Krest [kolvo\_krest];

fin.open("krests.txt", ios::in); //открыть файл

for(int m=0; m<kolvo\_krest; m++)

{

getline(fin, strread, ';'); //прочитать из файла

krests[m].set\_name(strread);

getline(fin, strread, ';'); //прочитать из файла

krests[m].set\_surname(strread);

getline(fin, strread, ';'); //прочитать из файла

krests[m].set\_proizv\_hleb(atoi(strread.c\_str())); //записать в нужное поле

getline(fin, strread, ';'); //прочитать из файла

krests[m].set\_proizv\_skot(atoi(strread.c\_str())); //записать в нужное поле

getline(fin, strread, ';'); //прочитать из файла

krests[m].set\_eda\_hleb(atoi(strread.c\_str())); //записать в нужное поле

getline(fin, strread, ';'); //прочитать из файла

krests[m].set\_eda\_skot(atoi(strread.c\_str())); //записать в нужное поле

getline(fin, strread, ';'); //прочитать из файла

krests[m].set\_money\_trat(atoi(strread.c\_str())); //записать в нужное поле

getline(fin, strread, ';'); //прочитать из файла

krests[m].set\_naim(atoi(strread.c\_str())); //записать в нужное поле

getline(fin, strread, '\n'); //прочитать из файла

krests[m].set\_otschet\_hours\_net\_edi(atoi(strread.c\_str())); //записать в нужное поле

}

fin.close(); //закрыть файл

}

void Village::Delete\_krest(int number) //удаление крестьянина с выбранным номером (нумерация от единицы!)

{

Krest\* kre\_change = new Krest[kolvo\_krest-1]; //временный массив крестьян

for(int j=0, k1=0; j<kolvo\_krest; j++) //перебрать весь массив крестьян в деревне

{

if(j!=(number-1)) //если номер крестьянина не равен искомому, то скопировать его в новый массив

{

kre\_change[k1]=krests[j];

k1++;

}

}

delete krests; //удалить старый массив

krests = kre\_change; //теперь указатель указывает на новый массив

kolvo\_krest=kolvo\_krest-1; //теперь на 1 крестьянина меньше

}

void Village::Add\_rand\_krest() //добавление случайного крестьянина к имеющимся (особое событие, возможное зимой)

{

Krest\* kre\_change = new Krest[kolvo\_krest+1]; //временный массив крестьян

int j;

for(j=0; j<kolvo\_krest; j++) //перебрать весь массив крестьян в деревне

{

kre\_change[j]=krests[j];

}

kre\_change[j].Init\_rand(); //новый крестьянин генерируется случайно

delete krests; //удалить старый массив

krests = kre\_change; //теперь указатель указывает на новый массив

kolvo\_krest=kolvo\_krest+1; //теперь на 1 крестьянина больше

}

void Village::Naim\_krest(Village vil\_naim, int numk) //найм крестьянина с выбранным номером из биржи труда (нумерация от единицы!)

{

Krest\* kre\_change = new Krest[kolvo\_krest+1]; //временный массив крестьян

int j;

for(j=0; j<kolvo\_krest; j++) //перебрать весь массив крестьян в деревне

{

kre\_change[j]=krests[j];

}

kre\_change[j]=vil\_naim.krests[numk-1]; //новый крестьянин берётся из массива найма под нужным номером (нумерация от единицы, а не от нуля!)

delete krests; //удалить старый массив

krests = kre\_change; //теперь указатель указывает на новый массив

kolvo\_krest=kolvo\_krest+1; //теперь на 1 крестьянина больше

}

//Получение и установление соответствующих полей для деревни

void Village::set\_kolvo\_krest(int in)

{

this->kolvo\_krest=in;

}

int Village::get\_kolvo\_krest()

{

return kolvo\_krest;

}

/////

void Village::set\_hours\_procv(int in)

{

this->hours\_procv=in;

}

int Village::get\_hours\_procv()

{

return hours\_procv;

}

/////

void Village::set\_days\_procv(int in)

{

this->days\_procv=in;

}

int Village::get\_days\_procv()

{

return days\_procv;

}

/////

void Village::set\_speed\_life(int in)

{

this->speed\_life=in;

}

int Village::get\_speed\_life()

{

return speed\_life;

}

/////

void Village::set\_flag\_season(int in)

{

this->flag\_season=in;

}

int Village::get\_flag\_season()

{

return flag\_season;

}

/////

void Village::set\_kolvo\_hleb(int in)

{

this->kolvo\_hleb=in;

}

int Village::get\_kolvo\_hleb()

{

return kolvo\_hleb;

}

/////

void Village::set\_kolvo\_skot(int in)

{

this->kolvo\_skot=in;

}

int Village::get\_kolvo\_skot()

{

return kolvo\_skot;

}

/////

void Village::set\_budget\_village(int in)

{

this->budget\_village=in;

}

int Village::get\_budget\_village()

{

return budget\_village;

}

/////

void Village::set\_hours\_from\_begin\_of\_season(int in)

{

this->hours\_from\_begin\_of\_season=in;

}

int Village::get\_hours\_from\_begin\_of\_season()

{

return hours\_from\_begin\_of\_season;

}

/////

void Village::set\_otschet\_hours\_net\_krest(int in)

{

this->otschet\_hours\_net\_krest=in;

}

int Village::get\_otschet\_hours\_net\_krest()

{

return otschet\_hours\_net\_krest;

}

/////

void Village::set\_price\_hleb(int in)

{

this->price\_hleb=in;

}

int Village::get\_price\_hleb()

{

return price\_hleb;

}

/////

void Village::set\_price\_skot(int in)

{

this->price\_skot=in;

}

int Village::get\_price\_skot()

{

return price\_skot;

}

//Обявления переменных

Village derevn; //объект, в котором хранится вся информация о деревне (глобальн.)

Village birzh\_trud; //объект, в котором хранятся крестьяне для найма (глобальн.)

itog\_game.h

#pragma once

#include "krest\_and\_village.h"

#include <fstream>

namespace kurs2020 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для itog\_game

/// </summary>

public ref class itog\_game : public System::Windows::Forms::Form

{

public:

itog\_game(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~itog\_game()

{

if (components)

{

delete components;

}

}

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

protected:

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

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

private: System::Windows::Forms::Label^ your\_scores;

private: System::Windows::Forms::TextBox^ name\_player;

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

private: System::Windows::Forms::Button^ ok\_record;

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

private:

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

void InitializeComponent(void)

{

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

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

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

this->your\_scores = (gcnew System::Windows::Forms::Label());

this->name\_player = (gcnew System::Windows::Forms::TextBox());

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

this->ok\_record = (gcnew System::Windows::Forms::Button());

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

this->SuspendLayout();

//

// label1

//

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->label1->Location = System::Drawing::Point(0, 48);

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

this->label1->Size = System::Drawing::Size(284, 62);

this->label1->TabIndex = 0;

this->label1->Text = L"Все крестьяне покинули деревню.\r\nПред вашим взором только \r\nзаброшенные просторы."

L"";

this->label1->TextAlign = System::Drawing::ContentAlignment::MiddleCenter;

//

// label2

//

this->label2->Font = (gcnew System::Drawing::Font(L"Palatino Linotype", 20.25F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

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

this->label2->ForeColor = System::Drawing::SystemColors::ControlText;

this->label2->Location = System::Drawing::Point(0, 9);

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

this->label2->Size = System::Drawing::Size(284, 39);

this->label2->TabIndex = 1;

this->label2->Text = L"Игра окончена!";

this->label2->TextAlign = System::Drawing::ContentAlignment::MiddleCenter;

//

// label3

//

this->label3->Font = (gcnew System::Drawing::Font(L"Palatino Linotype", 12.75F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

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

this->label3->ForeColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(0)), static\_cast<System::Int32>(static\_cast<System::Byte>(192)),

static\_cast<System::Int32>(static\_cast<System::Byte>(0)));

this->label3->Location = System::Drawing::Point(-1, 112);

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

this->label3->Size = System::Drawing::Size(223, 39);

this->label3->TabIndex = 2;

this->label3->Text = L"Итоговый счёт (Дни П.):";

this->label3->TextAlign = System::Drawing::ContentAlignment::MiddleCenter;

//

// your\_scores

//

this->your\_scores->Font = (gcnew System::Drawing::Font(L"Palatino Linotype", 14.25F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

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

this->your\_scores->ForeColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(64)), static\_cast<System::Int32>(static\_cast<System::Byte>(0)),

static\_cast<System::Int32>(static\_cast<System::Byte>(64)));

this->your\_scores->Location = System::Drawing::Point(213, 118);

this->your\_scores->Name = L"your\_scores";

this->your\_scores->Size = System::Drawing::Size(69, 60);

this->your\_scores->TabIndex = 3;

this->your\_scores->Text = L"12345";

//

// name\_player

//

this->name\_player->Location = System::Drawing::Point(79, 168);

this->name\_player->MaxLength = 20;

this->name\_player->Name = L"name\_player";

this->name\_player->Size = System::Drawing::Size(128, 20);

this->name\_player->TabIndex = 1;

this->name\_player->Text = L"player";

this->name\_player->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;

this->name\_player->TextChanged += gcnew System::EventHandler(this, &itog\_game::name\_player\_TextChanged);

//

// label5

//

this->label5->AutoSize = true;

this->label5->Location = System::Drawing::Point(36, 152);

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

this->label5->Size = System::Drawing::Size(221, 13);

this->label5->TabIndex = 5;

this->label5->Text = L"Введите ваше имя: (для списка рекордов)";

//

// ok\_record

//

this->ok\_record->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->ok\_record->Location = System::Drawing::Point(17, 194);

this->ok\_record->Name = L"ok\_record";

this->ok\_record->Size = System::Drawing::Size(118, 44);

this->ok\_record->TabIndex = 2;

this->ok\_record->Text = L"записать и закрыть";

this->ok\_record->UseVisualStyleBackColor = true;

this->ok\_record->Click += gcnew System::EventHandler(this, &itog\_game::ok\_record\_Click);

//

// close

//

this->close->DialogResult = System::Windows::Forms::DialogResult::Cancel;

this->close->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->close->Location = System::Drawing::Point(154, 194);

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

this->close->Size = System::Drawing::Size(118, 44);

this->close->TabIndex = 3;

this->close->Text = L"закрыть без сохранения рекорда";

this->close->UseVisualStyleBackColor = true;

this->close->Click += gcnew System::EventHandler(this, &itog\_game::close\_Click);

//

// itog\_game

//

this->AcceptButton = this->ok\_record;

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

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

this->CancelButton = this->close;

this->ClientSize = System::Drawing::Size(284, 261);

this->ControlBox = false;

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

this->Controls->Add(this->ok\_record);

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

this->Controls->Add(this->name\_player);

this->Controls->Add(this->your\_scores);

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

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

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

this->MaximizeBox = false;

this->MaximumSize = System::Drawing::Size(300, 300);

this->MinimizeBox = false;

this->MinimumSize = System::Drawing::Size(300, 300);

this->Name = L"itog\_game";

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

this->Text = L"Конец игры";

this->TopMost = true;

this->Load += gcnew System::EventHandler(this, &itog\_game::itog\_game\_Load);

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

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

Close();

}

private: System::Void itog\_game\_Load(System::Object^ sender, System::EventArgs^ e) {

this->your\_scores->Text = Convert::ToString(derevn.get\_days\_procv());

}

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

ifstream iff("records.txt");

ofstream fof; //создать объект класса ofstream (для записи в файл информации о деревне) (крестьяне отдельно)

if(!iff.is\_open())

{

fof.open("records.txt", ios::out); //открыть файл для новой записи

}

else

{

fof.open("records.txt", ios::app); //открыть файл для добавления записи

}

iff.close();

using namespace System::Runtime::InteropServices;

IntPtr ptr = Marshal::StringToHGlobalAnsi(this->name\_player->Text); //преобразовать информацию из TextBox в массив char

char\* char\_str = (char\*)ptr.ToPointer();

fof << char\_str; //записать в файл

fof << ';';

ptr = Marshal::StringToHGlobalAnsi(this->your\_scores->Text); //преобразовать информацию из Label в массив char (реализовано так, чтобы рекорд нельзя было изменить, начав новую игру, не закрыв окно с финальным счётом)

char\_str = (char\*)ptr.ToPointer();

fof << char\_str; //записать в файл

fof << '\n';

fof.close();

Close();

}

private: System::Void name\_player\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

if(this->name\_player->Text==L"") //если поле с именем игрока пустое

{

this->ok\_record->Enabled=false;

}

else

{

using namespace System::Runtime::InteropServices;

IntPtr ptr = Marshal::StringToHGlobalAnsi(this->name\_player->Text); //преобразовать информацию из TextBox в массив char

char\* char\_str = (char\*)ptr.ToPointer();

if(strchr( char\_str, ';')!=NULL) //проверить, есть ли в поле с именем игрока запрещённый символ ';'

{

this->ok\_record->Enabled=false;

}

else

{

this->ok\_record->Enabled=true;

}

}

}

};

}

tabl\_krest.h

#pragma once

#include "krest\_and\_village.h"

#include "tabl\_naim.h"

#include <string>

extern int f\_endgame; //флаг о том, что игра окончена (игрок проиграл) (глобальн.)

namespace kurs2020 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для tabl\_krest

/// </summary>

public ref class tabl\_krest : public System::Windows::Forms::Form

{

public:

tabl\_krest(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~tabl\_krest()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Button^ trud\_butt;

private: System::Windows::Forms::Button^ izgnat\_butt;

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

private: System::Windows::Forms::GroupBox^ izgnat\_box;

private: System::Windows::Forms::DataGridView^ spisok;

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

private: System::Windows::Forms::DataGridViewTextBoxColumn^ number\_tabl;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ name\_tabl;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ proizv\_tabl;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ prozhorl\_tabl;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ rashod\_tabl;

private: System::Windows::Forms::MaskedTextBox^ num\_krest\_pole;

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

private: System::Windows::Forms::Timer^ timer\_proverk\_krest;

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

protected:

private:

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

#pragma region Windows Form Designer generated code

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

void InitializeComponent(void)

{

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

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(tabl\_krest::typeid));

this->trud\_butt = (gcnew System::Windows::Forms::Button());

this->izgnat\_butt = (gcnew System::Windows::Forms::Button());

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

this->izgnat\_box = (gcnew System::Windows::Forms::GroupBox());

this->num\_krest\_pole = (gcnew System::Windows::Forms::MaskedTextBox());

this->spisok = (gcnew System::Windows::Forms::DataGridView());

this->number\_tabl = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->name\_tabl = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->proizv\_tabl = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->prozhorl\_tabl = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->rashod\_tabl = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

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

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

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

this->izgnat\_box->SuspendLayout();

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

this->SuspendLayout();

//

// trud\_butt

//

this->trud\_butt->Anchor = System::Windows::Forms::AnchorStyles::Top;

this->trud\_butt->Font = (gcnew System::Drawing::Font(L"DejaVu Serif", 12.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->trud\_butt->Location = System::Drawing::Point(16, 22);

this->trud\_butt->Name = L"trud\_butt";

this->trud\_butt->Size = System::Drawing::Size(254, 35);

this->trud\_butt->TabIndex = 9;

this->trud\_butt->TabStop = false;

this->trud\_butt->Text = L"открыть биржу труда";

this->trud\_butt->UseVisualStyleBackColor = true;

this->trud\_butt->Click += gcnew System::EventHandler(this, &tabl\_krest::trud\_butt\_Click);

//

// izgnat\_butt

//

this->izgnat\_butt->Anchor = System::Windows::Forms::AnchorStyles::Top;

this->izgnat\_butt->Enabled = false;

this->izgnat\_butt->Font = (gcnew System::Drawing::Font(L"Verdana", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->izgnat\_butt->Location = System::Drawing::Point(311, 34);

this->izgnat\_butt->Name = L"izgnat\_butt";

this->izgnat\_butt->Size = System::Drawing::Size(103, 28);

this->izgnat\_butt->TabIndex = 10;

this->izgnat\_butt->TabStop = false;

this->izgnat\_butt->Text = L"Изгнать";

this->izgnat\_butt->UseVisualStyleBackColor = true;

this->izgnat\_butt->Click += gcnew System::EventHandler(this, &tabl\_krest::izgnat\_butt\_Click);

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(19, 22);

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

this->label1->Size = System::Drawing::Size(286, 13);

this->label1->TabIndex = 12;

this->label1->Text = L"Введите номер крестьянина, которого хотите изгнать:";

//

// izgnat\_box

//

this->izgnat\_box->Anchor = System::Windows::Forms::AnchorStyles::Top;

this->izgnat\_box->Controls->Add(this->num\_krest\_pole);

this->izgnat\_box->Controls->Add(this->label1);

this->izgnat\_box->Controls->Add(this->izgnat\_butt);

this->izgnat\_box->Location = System::Drawing::Point(423, 12);

this->izgnat\_box->Name = L"izgnat\_box";

this->izgnat\_box->Size = System::Drawing::Size(428, 87);

this->izgnat\_box->TabIndex = 13;

this->izgnat\_box->TabStop = false;

//

// num\_krest\_pole

//

this->num\_krest\_pole->Location = System::Drawing::Point(22, 39);

this->num\_krest\_pole->Mask = L"000000000000000000000000000000000000000000000";

this->num\_krest\_pole->Name = L"num\_krest\_pole";

this->num\_krest\_pole->PromptChar = ' ';

this->num\_krest\_pole->Size = System::Drawing::Size(283, 20);

this->num\_krest\_pole->TabIndex = 16;

this->num\_krest\_pole->TabStop = false;

this->num\_krest\_pole->TextAlign = System::Windows::Forms::HorizontalAlignment::Right;

this->num\_krest\_pole->TextChanged += gcnew System::EventHandler(this, &tabl\_krest::num\_krest\_pole1\_TextChanged);

//

// spisok

//

this->spisok->AllowUserToAddRows = false;

this->spisok->AllowUserToDeleteRows = false;

this->spisok->AllowUserToResizeColumns = false;

this->spisok->AllowUserToResizeRows = false;

this->spisok->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::Top | System::Windows::Forms::AnchorStyles::Bottom));

this->spisok->ClipboardCopyMode = System::Windows::Forms::DataGridViewClipboardCopyMode::Disable;

this->spisok->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->spisok->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(5) {this->number\_tabl, this->name\_tabl,

this->proizv\_tabl, this->prozhorl\_tabl, this->rashod\_tabl});

this->spisok->EditMode = System::Windows::Forms::DataGridViewEditMode::EditProgrammatically;

this->spisok->Location = System::Drawing::Point(12, 117);

this->spisok->MultiSelect = false;

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

this->spisok->ReadOnly = true;

this->spisok->RowHeadersVisible = false;

this->spisok->RowHeadersWidthSizeMode = System::Windows::Forms::DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders;

this->spisok->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->spisok->SelectionMode = System::Windows::Forms::DataGridViewSelectionMode::CellSelect;

this->spisok->ShowEditingIcon = false;

this->spisok->Size = System::Drawing::Size(850, 362);

this->spisok->TabIndex = 14;

this->spisok->TabStop = false;

//

// number\_tabl

//

this->number\_tabl->AutoSizeMode = System::Windows::Forms::DataGridViewAutoSizeColumnMode::None;

this->number\_tabl->Frozen = true;

this->number\_tabl->HeaderText = L"№";

this->number\_tabl->Name = L"number\_tabl";

this->number\_tabl->ReadOnly = true;

this->number\_tabl->SortMode = System::Windows::Forms::DataGridViewColumnSortMode::Programmatic;

this->number\_tabl->Width = 50;

//

// name\_tabl

//

this->name\_tabl->AutoSizeMode = System::Windows::Forms::DataGridViewAutoSizeColumnMode::None;

this->name\_tabl->Frozen = true;

this->name\_tabl->HeaderText = L"Имя и фамилия";

this->name\_tabl->Name = L"name\_tabl";

this->name\_tabl->ReadOnly = true;

this->name\_tabl->SortMode = System::Windows::Forms::DataGridViewColumnSortMode::Programmatic;

this->name\_tabl->Width = 195;

//

// proizv\_tabl

//

this->proizv\_tabl->AutoSizeMode = System::Windows::Forms::DataGridViewAutoSizeColumnMode::None;

this->proizv\_tabl->Frozen = true;

this->proizv\_tabl->HeaderText = L"Производительность за один Час Процветания (хлеб/скот)";

this->proizv\_tabl->Name = L"proizv\_tabl";

this->proizv\_tabl->ReadOnly = true;

this->proizv\_tabl->SortMode = System::Windows::Forms::DataGridViewColumnSortMode::Programmatic;

this->proizv\_tabl->Width = 200;

//

// prozhorl\_tabl

//

this->prozhorl\_tabl->AutoSizeMode = System::Windows::Forms::DataGridViewAutoSizeColumnMode::None;

this->prozhorl\_tabl->Frozen = true;

this->prozhorl\_tabl->HeaderText = L"Прожорливость за один Час Процветания (хлеб/скот)";

this->prozhorl\_tabl->Name = L"prozhorl\_tabl";

this->prozhorl\_tabl->ReadOnly = true;

this->prozhorl\_tabl->SortMode = System::Windows::Forms::DataGridViewColumnSortMode::Programmatic;

this->prozhorl\_tabl->Width = 200;

//

// rashod\_tabl

//

this->rashod\_tabl->AutoSizeMode = System::Windows::Forms::DataGridViewAutoSizeColumnMode::None;

this->rashod\_tabl->Frozen = true;

this->rashod\_tabl->HeaderText = L"Денежные расходы за один Час Процветания (руб.)";

this->rashod\_tabl->Name = L"rashod\_tabl";

this->rashod\_tabl->ReadOnly = true;

this->rashod\_tabl->SortMode = System::Windows::Forms::DataGridViewColumnSortMode::Programmatic;

this->rashod\_tabl->Width = 200;

//

// label2

//

this->label2->Anchor = System::Windows::Forms::AnchorStyles::Top;

this->label2->AutoSize = true;

this->label2->Font = (gcnew System::Drawing::Font(L"Lucida Console", 15.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label2->Location = System::Drawing::Point(12, 93);

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

this->label2->Size = System::Drawing::Size(205, 21);

this->label2->TabIndex = 15;

this->label2->Text = L"Ваши крестьяне:";

//

// label3

//

this->label3->AutoSize = true;

this->label3->Location = System::Drawing::Point(2, 2);

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

this->label3->Size = System::Drawing::Size(377, 13);

this->label3->TabIndex = 17;

this->label3->Text = L"Примечание: информация выводится с учётом текущей Скорости жизни";

//

// timer\_proverk\_krest

//

this->timer\_proverk\_krest->Enabled = true;

this->timer\_proverk\_krest->Interval = 1003;

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

//

// tabl\_krest

//

this->AcceptButton = this->izgnat\_butt;

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

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

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

this->ClientSize = System::Drawing::Size(874, 491);

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

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

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

this->Controls->Add(this->izgnat\_box);

this->Controls->Add(this->trud\_butt);

this->DoubleBuffered = true;

this->MinimizeBox = false;

this->MinimumSize = System::Drawing::Size(890, 530);

this->Name = L"tabl\_krest";

this->ShowIcon = false;

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

this->Text = L"Список крестьян";

this->Activated += gcnew System::EventHandler(this, &tabl\_krest::tabl\_krest\_Activated);

this->Load += gcnew System::EventHandler(this, &tabl\_krest::tabl\_krest\_Load);

this->izgnat\_box->ResumeLayout(false);

this->izgnat\_box->PerformLayout();

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

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void tabl\_krest\_Activated(System::Object^ sender, System::EventArgs^ e) {

if(f\_endgame==1) //игра закончилась, закрыть все формы

Close();

else

{

//очистить все строки перед выводом

while (0 != spisok->RowCount)

spisok->Rows->RemoveAt(0);

int kolvo\_str=derevn.get\_kolvo\_krest(); //получить кол-во крестьян в списке

for(int num=0; num<kolvo\_str; num++)

{

spisok->Rows->Add(); //добавить новую строку в таблицу

spisok->Rows[num]->Cells[0]->Value = Convert::ToString(num+1); //записать порядковый номер крестьянина

string str\_name= derevn.krests[num].get\_name() + " " + derevn.krests[num].get\_surname();

spisok->Rows[num]->Cells[1]->Value = gcnew System::String(str\_name.c\_str()); //внести в ячейку таблицы

spisok->Rows[num]->Cells[2]->Value = Convert::ToString(derevn.krests[num].get\_proizv\_hleb()) + L"/" + Convert::ToString(derevn.krests[num].get\_proizv\_skot());

spisok->Rows[num]->Cells[3]->Value = Convert::ToString(derevn.krests[num].get\_eda\_hleb()\*derevn.get\_speed\_life()) + L"/" + Convert::ToString(derevn.krests[num].get\_eda\_skot()\*derevn.get\_speed\_life());

spisok->Rows[num]->Cells[4]->Value = Convert::ToString(derevn.krests[num].get\_money\_trat()\*derevn.get\_speed\_life());

}

}

}

private: System::Void num\_krest\_pole1\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

int kolvo\_strr=derevn.get\_kolvo\_krest(); //получить кол-во крестьян в списке

if (this->num\_krest\_pole->Text!=L"") //если поле с номером строки не пустое

{

if((Convert::ToDouble(this->num\_krest\_pole->Text) > kolvo\_strr) || (Convert::ToDouble(this->num\_krest\_pole->Text)<1)) //если выбр. строка больше общего кол-ва строк, то отключить кнопку

{

this->izgnat\_butt->Enabled = false;

}

else

this->izgnat\_butt->Enabled = true;

}

else

this->izgnat\_butt->Enabled = false;

if(derevn.get\_flag\_season()==1) //если зима, кнопка Изгнать недоступна в любом случае

{

this->label1->Enabled = false;

this->num\_krest\_pole->Enabled = false;

this->izgnat\_butt->Enabled = false;

this->trud\_butt->Enabled = false;

}

if(derevn.get\_flag\_season()==0) //если лето, включить кнопку открытия Биржи труда

{

this->label1->Enabled = true;

this->num\_krest\_pole->Enabled = true;

this->trud\_butt->Enabled = true;

}

}

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

derevn.Delete\_krest(Convert::ToDouble(this->num\_krest\_pole->Text)); //удалить крестьянина

num\_krest\_pole1\_TextChanged(sender,e); //повторная проверка

tabl\_krest\_Activated(sender,e); //обновить таблицу

}

private: System::Void tabl\_krest\_Load(System::Object^ sender, System::EventArgs^ e) {

num\_krest\_pole1\_TextChanged(sender,e); //проверка

}

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

tabl\_naim^ tabl\_naim\_p = gcnew tabl\_naim(); //указатель на форму

tabl\_naim\_p -> ShowDialog(); //открыть форму

delete(tabl\_naim\_p);

}

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

if(f\_endgame==1) //игра закончилась, закрыть все формы

Close();

else

{

num\_krest\_pole1\_TextChanged(sender,e); //повторная проверка

tabl\_krest\_Activated(sender,e); //обновить таблицу

}

}

};

}

tabl\_naim.h

#pragma once

#include "krest\_and\_village.h"

#include <string>

extern int f\_endgame; //флаг о том, что игра окончена (игрок проиграл) (глобальн.)

namespace kurs2020 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для tabl\_naim

/// </summary>

public ref class tabl\_naim : public System::Windows::Forms::Form

{

public:

tabl\_naim(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~tabl\_naim()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Button^ naim\_butt;

protected:

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

private: System::Windows::Forms::GroupBox^ naim\_box;

private: System::Windows::Forms::DataGridView^ spisok;

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

private: System::Windows::Forms::MaskedTextBox^ num\_krest\_pole;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ number\_tabl;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ name\_tabl;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ proizv\_tabl;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ prozhorl\_tabl;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ rashod\_tabl;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ naim\_money\_tabl;

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

private: System::Windows::Forms::Timer^ timer\_proverk\_krest;

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

protected:

private:

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

#pragma region Windows Form Designer generated code

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

void InitializeComponent(void)

{

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

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(tabl\_naim::typeid));

this->naim\_butt = (gcnew System::Windows::Forms::Button());

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

this->naim\_box = (gcnew System::Windows::Forms::GroupBox());

this->num\_krest\_pole = (gcnew System::Windows::Forms::MaskedTextBox());

this->spisok = (gcnew System::Windows::Forms::DataGridView());

this->number\_tabl = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->name\_tabl = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->proizv\_tabl = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->prozhorl\_tabl = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->rashod\_tabl = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->naim\_money\_tabl = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

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

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

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

this->naim\_box->SuspendLayout();

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

this->SuspendLayout();

//

// naim\_butt

//

this->naim\_butt->Anchor = System::Windows::Forms::AnchorStyles::Top;

this->naim\_butt->Enabled = false;

this->naim\_butt->Font = (gcnew System::Drawing::Font(L"Verdana", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->naim\_butt->Location = System::Drawing::Point(311, 34);

this->naim\_butt->Name = L"naim\_butt";

this->naim\_butt->Size = System::Drawing::Size(103, 28);

this->naim\_butt->TabIndex = 10;

this->naim\_butt->TabStop = false;

this->naim\_butt->Text = L"Нанять";

this->naim\_butt->UseVisualStyleBackColor = true;

this->naim\_butt->Click += gcnew System::EventHandler(this, &tabl\_naim::naim\_butt\_Click);

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(24, 23);

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

this->label1->Size = System::Drawing::Size(281, 13);

this->label1->TabIndex = 12;

this->label1->Text = L"Введите номер крестьянина, которого хотите нанять:";

//

// naim\_box

//

this->naim\_box->Anchor = System::Windows::Forms::AnchorStyles::Top;

this->naim\_box->Controls->Add(this->num\_krest\_pole);

this->naim\_box->Controls->Add(this->label1);

this->naim\_box->Controls->Add(this->naim\_butt);

this->naim\_box->Location = System::Drawing::Point(423, 12);

this->naim\_box->Name = L"naim\_box";

this->naim\_box->Size = System::Drawing::Size(428, 87);

this->naim\_box->TabIndex = 13;

this->naim\_box->TabStop = false;

//

// num\_krest\_pole

//

this->num\_krest\_pole->Location = System::Drawing::Point(22, 39);

this->num\_krest\_pole->Mask = L"000000000000000000000000000000000000000000000";

this->num\_krest\_pole->Name = L"num\_krest\_pole";

this->num\_krest\_pole->PromptChar = ' ';

this->num\_krest\_pole->Size = System::Drawing::Size(283, 20);

this->num\_krest\_pole->TabIndex = 16;

this->num\_krest\_pole->TabStop = false;

this->num\_krest\_pole->TextAlign = System::Windows::Forms::HorizontalAlignment::Right;

this->num\_krest\_pole->TextChanged += gcnew System::EventHandler(this, &tabl\_naim::num\_krest\_pole1\_TextChanged);

//

// spisok

//

this->spisok->AllowUserToAddRows = false;

this->spisok->AllowUserToDeleteRows = false;

this->spisok->AllowUserToResizeColumns = false;

this->spisok->AllowUserToResizeRows = false;

this->spisok->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::Top | System::Windows::Forms::AnchorStyles::Bottom));

this->spisok->ClipboardCopyMode = System::Windows::Forms::DataGridViewClipboardCopyMode::Disable;

this->spisok->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->spisok->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(6) {this->number\_tabl, this->name\_tabl,

this->proizv\_tabl, this->prozhorl\_tabl, this->rashod\_tabl, this->naim\_money\_tabl});

this->spisok->EditMode = System::Windows::Forms::DataGridViewEditMode::EditProgrammatically;

this->spisok->Location = System::Drawing::Point(12, 117);

this->spisok->MultiSelect = false;

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

this->spisok->ReadOnly = true;

this->spisok->RowHeadersVisible = false;

this->spisok->RowHeadersWidthSizeMode = System::Windows::Forms::DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders;

this->spisok->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->spisok->SelectionMode = System::Windows::Forms::DataGridViewSelectionMode::CellSelect;

this->spisok->ShowEditingIcon = false;

this->spisok->Size = System::Drawing::Size(850, 242);

this->spisok->TabIndex = 14;

this->spisok->TabStop = false;

//

// number\_tabl

//

this->number\_tabl->AutoSizeMode = System::Windows::Forms::DataGridViewAutoSizeColumnMode::None;

this->number\_tabl->Frozen = true;

this->number\_tabl->HeaderText = L"№";

this->number\_tabl->Name = L"number\_tabl";

this->number\_tabl->ReadOnly = true;

this->number\_tabl->SortMode = System::Windows::Forms::DataGridViewColumnSortMode::Programmatic;

this->number\_tabl->Width = 40;

//

// name\_tabl

//

this->name\_tabl->AutoSizeMode = System::Windows::Forms::DataGridViewAutoSizeColumnMode::None;

this->name\_tabl->Frozen = true;

this->name\_tabl->HeaderText = L"Имя и фамилия";

this->name\_tabl->Name = L"name\_tabl";

this->name\_tabl->ReadOnly = true;

this->name\_tabl->SortMode = System::Windows::Forms::DataGridViewColumnSortMode::Programmatic;

this->name\_tabl->Width = 180;

//

// proizv\_tabl

//

this->proizv\_tabl->AutoSizeMode = System::Windows::Forms::DataGridViewAutoSizeColumnMode::None;

this->proizv\_tabl->Frozen = true;

this->proizv\_tabl->HeaderText = L"Производительность за один Час Процветания (хлеб/скот)";

this->proizv\_tabl->Name = L"proizv\_tabl";

this->proizv\_tabl->ReadOnly = true;

this->proizv\_tabl->SortMode = System::Windows::Forms::DataGridViewColumnSortMode::Programmatic;

this->proizv\_tabl->Width = 180;

//

// prozhorl\_tabl

//

this->prozhorl\_tabl->AutoSizeMode = System::Windows::Forms::DataGridViewAutoSizeColumnMode::None;

this->prozhorl\_tabl->Frozen = true;

this->prozhorl\_tabl->HeaderText = L"Прожорливость за один Час Процветания (хлеб/скот)";

this->prozhorl\_tabl->Name = L"prozhorl\_tabl";

this->prozhorl\_tabl->ReadOnly = true;

this->prozhorl\_tabl->SortMode = System::Windows::Forms::DataGridViewColumnSortMode::Programmatic;

this->prozhorl\_tabl->Width = 177;

//

// rashod\_tabl

//

this->rashod\_tabl->AutoSizeMode = System::Windows::Forms::DataGridViewAutoSizeColumnMode::None;

this->rashod\_tabl->Frozen = true;

this->rashod\_tabl->HeaderText = L"Денежные расходы за один Час Процветания (руб.)";

this->rashod\_tabl->Name = L"rashod\_tabl";

this->rashod\_tabl->ReadOnly = true;

this->rashod\_tabl->SortMode = System::Windows::Forms::DataGridViewColumnSortMode::Programmatic;

this->rashod\_tabl->Width = 177;

//

// naim\_money\_tabl

//

this->naim\_money\_tabl->AutoSizeMode = System::Windows::Forms::DataGridViewAutoSizeColumnMode::None;

this->naim\_money\_tabl->Frozen = true;

this->naim\_money\_tabl->HeaderText = L"Стоимость найма (руб.)";

this->naim\_money\_tabl->Name = L"naim\_money\_tabl";

this->naim\_money\_tabl->ReadOnly = true;

this->naim\_money\_tabl->SortMode = System::Windows::Forms::DataGridViewColumnSortMode::Programmatic;

this->naim\_money\_tabl->Width = 91;

//

// label2

//

this->label2->Anchor = System::Windows::Forms::AnchorStyles::Top;

this->label2->AutoSize = true;

this->label2->BackColor = System::Drawing::Color::Transparent;

this->label2->Font = (gcnew System::Drawing::Font(L"Lucida Console", 15.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label2->Location = System::Drawing::Point(12, 93);

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

this->label2->Size = System::Drawing::Size(400, 21);

this->label2->TabIndex = 15;

this->label2->Text = L"Доступные для найма крестьяне:";

//

// label3

//

this->label3->AutoSize = true;

this->label3->Location = System::Drawing::Point(2, 2);

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

this->label3->Size = System::Drawing::Size(377, 13);

this->label3->TabIndex = 18;

this->label3->Text = L"Примечание: информация выводится с учётом текущей Скорости жизни";

//

// timer\_proverk\_krest

//

this->timer\_proverk\_krest->Enabled = true;

this->timer\_proverk\_krest->Interval = 1003;

this->timer\_proverk\_krest->Tick += gcnew System::EventHandler(this, &tabl\_naim::timer\_proverk\_krest\_Tick);

//

// tabl\_naim

//

this->AcceptButton = this->naim\_butt;

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

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

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

this->ClientSize = System::Drawing::Size(874, 371);

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

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

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

this->Controls->Add(this->naim\_box);

this->DoubleBuffered = true;

this->MaximizeBox = false;

this->MaximumSize = System::Drawing::Size(890, 410);

this->MinimizeBox = false;

this->MinimumSize = System::Drawing::Size(890, 410);

this->Name = L"tabl\_naim";

this->ShowIcon = false;

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

this->Text = L"Биржа труда";

this->Activated += gcnew System::EventHandler(this, &tabl\_naim::tabl\_naim\_Activated);

this->Load += gcnew System::EventHandler(this, &tabl\_naim::tabl\_naim\_Load);

this->naim\_box->ResumeLayout(false);

this->naim\_box->PerformLayout();

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

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void tabl\_naim\_Activated(System::Object^ sender, System::EventArgs^ e) {

if(f\_endgame==1) //игра закончилась, закрыть все формы

Close();

else

{

//очистить все строки перед выводом

while (0 != spisok->RowCount)

spisok->Rows->RemoveAt(0);

int kolvo\_str=birzh\_trud.get\_kolvo\_krest(); //получить кол-во крестьян в списке

for(int num=0; num<kolvo\_str; num++)

{

spisok->Rows->Add(); //добавить новую строку в таблицу

spisok->Rows[num]->Cells[0]->Value = Convert::ToString(num+1); //записать порядковый номер крестьянина

string str\_name= birzh\_trud.krests[num].get\_name() + " " + birzh\_trud.krests[num].get\_surname();

spisok->Rows[num]->Cells[1]->Value = gcnew System::String(str\_name.c\_str()); //внести в ячейку таблицы

spisok->Rows[num]->Cells[2]->Value = Convert::ToString(birzh\_trud.krests[num].get\_proizv\_hleb()) + L"/" + Convert::ToString(birzh\_trud.krests[num].get\_proizv\_skot());

spisok->Rows[num]->Cells[3]->Value = Convert::ToString(birzh\_trud.krests[num].get\_eda\_hleb()\*derevn.get\_speed\_life()) + L"/" + Convert::ToString(birzh\_trud.krests[num].get\_eda\_skot()\*derevn.get\_speed\_life());

spisok->Rows[num]->Cells[4]->Value = Convert::ToString(birzh\_trud.krests[num].get\_money\_trat()\*derevn.get\_speed\_life());

spisok->Rows[num]->Cells[5]->Value = Convert::ToString(birzh\_trud.krests[num].get\_naim());

}

}

}

private: System::Void num\_krest\_pole1\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

int kolvo\_strr=birzh\_trud.get\_kolvo\_krest(); //получить кол-во крестьян в списке

if (this->num\_krest\_pole->Text!=L"") //если поле с номером строки не пустое

{

int nummk = Convert::ToDouble(this->num\_krest\_pole->Text); //номер выбранного крестьянина

if((nummk > kolvo\_strr) || (nummk<1)) //если выбр. строка больше общего кол-ва строк, то отключить кнопку

{

this->naim\_butt->Enabled = false;

}

else

{

this->naim\_butt->Enabled = true;

if((derevn.get\_budget\_village()-birzh\_trud.krests[nummk-1].get\_naim())<0)

this->naim\_butt->Enabled = false;

}

}

else

this->naim\_butt->Enabled = false;

if(derevn.get\_flag\_season()==1) //если зима, закрыть форму

{

Close();

}

}

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

int nummk = Convert::ToDouble(this->num\_krest\_pole->Text); //номер выбранного крестьянина

derevn.set\_budget\_village(derevn.get\_budget\_village()-birzh\_trud.krests[nummk-1].get\_naim());

derevn.Naim\_krest(birzh\_trud, nummk);

birzh\_trud.Delete\_krest(nummk); //крестьянин перешёл из одного массива в другой. Из начального массива он удаляется

num\_krest\_pole1\_TextChanged(sender,e); //повторная проверка

tabl\_naim\_Activated(sender,e); //обновить таблицу

}

private: System::Void tabl\_naim\_Load(System::Object^ sender, System::EventArgs^ e) {

if(derevn.get\_flag\_season()==1) //если зима, закрыть форму

{

Close();

}

if(birzh\_trud.krests!=0)

{

delete [] birzh\_trud.krests;

}

birzh\_trud.krests = new Krest[5]; //массив крестьян для найма

birzh\_trud.set\_kolvo\_krest(5);

for(int v=0; v<5; v++)

{

birzh\_trud.krests[v].Init\_rand();

}

}

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

if(f\_endgame==1) //игра закончилась, закрыть все формы

Close();

else

{

num\_krest\_pole1\_TextChanged(sender,e); //повторная проверка

if(derevn.get\_flag\_season()==0) //таблицу обновлять только летом

{

tabl\_naim\_Activated(sender,e); //обновить таблицу

}

}

}

};

}

sell\_tovar.h

#pragma once

#include "krest\_and\_village.h"

extern int f\_endgame; //флаг о том, что игра окончена (игрок проиграл) (глобальн.)

namespace kurs2020 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для sell\_tovar

/// </summary>

public ref class sell\_tovar : public System::Windows::Forms::Form

{

public:

sell\_tovar(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~sell\_tovar()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Label^ price\_hleb\_text;

private: System::Windows::Forms::Label^ price\_skot\_text;

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

private: System::Windows::Forms::MaskedTextBox^ num\_sell;

private: System::Windows::Forms::Button^ sell\_butt;

private: System::Windows::Forms::GroupBox^ hleb\_or\_skot;

private: System::Windows::Forms::RadioButton^ skot\_radio;

private: System::Windows::Forms::RadioButton^ hleb\_radio;

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

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

protected:

private:

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

#pragma region Windows Form Designer generated code

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

void InitializeComponent(void)

{

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

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(sell\_tovar::typeid));

this->price\_hleb\_text = (gcnew System::Windows::Forms::Label());

this->price\_skot\_text = (gcnew System::Windows::Forms::Label());

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

this->num\_sell = (gcnew System::Windows::Forms::MaskedTextBox());

this->sell\_butt = (gcnew System::Windows::Forms::Button());

this->hleb\_or\_skot = (gcnew System::Windows::Forms::GroupBox());

this->skot\_radio = (gcnew System::Windows::Forms::RadioButton());

this->hleb\_radio = (gcnew System::Windows::Forms::RadioButton());

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

this->hleb\_or\_skot->SuspendLayout();

this->SuspendLayout();

//

// price\_hleb\_text

//

this->price\_hleb\_text->AutoSize = true;

this->price\_hleb\_text->BackColor = System::Drawing::SystemColors::Control;

this->price\_hleb\_text->BorderStyle = System::Windows::Forms::BorderStyle::Fixed3D;

this->price\_hleb\_text->Font = (gcnew System::Drawing::Font(L"Georgia", 20.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->price\_hleb\_text->Location = System::Drawing::Point(23, 55);

this->price\_hleb\_text->Name = L"price\_hleb\_text";

this->price\_hleb\_text->Size = System::Drawing::Size(355, 33);

this->price\_hleb\_text->TabIndex = 2;

this->price\_hleb\_text->Text = L"Текущая цена за ед. хлеба:";

//

// price\_skot\_text

//

this->price\_skot\_text->AutoSize = true;

this->price\_skot\_text->BackColor = System::Drawing::SystemColors::Control;

this->price\_skot\_text->BorderStyle = System::Windows::Forms::BorderStyle::Fixed3D;

this->price\_skot\_text->Font = (gcnew System::Drawing::Font(L"Georgia", 20.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->price\_skot\_text->Location = System::Drawing::Point(23, 115);

this->price\_skot\_text->Name = L"price\_skot\_text";

this->price\_skot\_text->Size = System::Drawing::Size(402, 33);

this->price\_skot\_text->TabIndex = 3;

this->price\_skot\_text->Text = L"Текущая цена за голову скота:";

//

// label2

//

this->label2->BackColor = System::Drawing::SystemColors::Control;

this->label2->BorderStyle = System::Windows::Forms::BorderStyle::Fixed3D;

this->label2->Font = (gcnew System::Drawing::Font(L"Century", 15.75F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

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

this->label2->ForeColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(0)), static\_cast<System::Int32>(static\_cast<System::Byte>(0)),

static\_cast<System::Int32>(static\_cast<System::Byte>(64)));

this->label2->Location = System::Drawing::Point(41, 195);

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

this->label2->Size = System::Drawing::Size(283, 34);

this->label2->TabIndex = 16;

this->label2->Text = L"Продажа ресурсов, шт.";

this->label2->TextAlign = System::Drawing::ContentAlignment::MiddleCenter;

//

// num\_sell

//

this->num\_sell->Location = System::Drawing::Point(41, 242);

this->num\_sell->Mask = L"000000000000000000000000000000000000000000000";

this->num\_sell->Name = L"num\_sell";

this->num\_sell->PromptChar = ' ';

this->num\_sell->Size = System::Drawing::Size(283, 20);

this->num\_sell->TabIndex = 18;

this->num\_sell->TabStop = false;

this->num\_sell->TextAlign = System::Windows::Forms::HorizontalAlignment::Right;

this->num\_sell->TextChanged += gcnew System::EventHandler(this, &sell\_tovar::num\_sell\_TextChanged);

//

// sell\_butt

//

this->sell\_butt->Enabled = false;

this->sell\_butt->Font = (gcnew System::Drawing::Font(L"Verdana", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->sell\_butt->Location = System::Drawing::Point(124, 280);

this->sell\_butt->Name = L"sell\_butt";

this->sell\_butt->Size = System::Drawing::Size(119, 28);

this->sell\_butt->TabIndex = 20;

this->sell\_butt->TabStop = false;

this->sell\_butt->Text = L"Продать";

this->sell\_butt->UseVisualStyleBackColor = true;

this->sell\_butt->Click += gcnew System::EventHandler(this, &sell\_tovar::sell\_butt\_Click);

//

// hleb\_or\_skot

//

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

this->hleb\_or\_skot->Controls->Add(this->skot\_radio);

this->hleb\_or\_skot->Controls->Add(this->hleb\_radio);

this->hleb\_or\_skot->Location = System::Drawing::Point(366, 195);

this->hleb\_or\_skot->Name = L"hleb\_or\_skot";

this->hleb\_or\_skot->Size = System::Drawing::Size(165, 113);

this->hleb\_or\_skot->TabIndex = 21;

this->hleb\_or\_skot->TabStop = false;

//

// skot\_radio

//

this->skot\_radio->AutoSize = true;

this->skot\_radio->BackColor = System::Drawing::Color::Transparent;

this->skot\_radio->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->skot\_radio->Location = System::Drawing::Point(16, 66);

this->skot\_radio->Name = L"skot\_radio";

this->skot\_radio->Size = System::Drawing::Size(137, 22);

this->skot\_radio->TabIndex = 1;

this->skot\_radio->Text = L"Домашний скот";

this->skot\_radio->UseVisualStyleBackColor = false;

this->skot\_radio->CheckedChanged += gcnew System::EventHandler(this, &sell\_tovar::skot\_radio\_CheckedChanged);

//

// hleb\_radio

//

this->hleb\_radio->AutoSize = true;

this->hleb\_radio->BackColor = System::Drawing::Color::Transparent;

this->hleb\_radio->Checked = true;

this->hleb\_radio->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->hleb\_radio->Location = System::Drawing::Point(16, 29);

this->hleb\_radio->Name = L"hleb\_radio";

this->hleb\_radio->Size = System::Drawing::Size(62, 22);

this->hleb\_radio->TabIndex = 0;

this->hleb\_radio->TabStop = true;

this->hleb\_radio->Text = L"Хлеб";

this->hleb\_radio->UseVisualStyleBackColor = false;

//

// timer\_proverka

//

this->timer\_proverka->Enabled = true;

this->timer\_proverka->Interval = 1003;

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

//

// sell\_tovar

//

this->AcceptButton = this->sell\_butt;

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

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

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

this->ClientSize = System::Drawing::Size(571, 371);

this->Controls->Add(this->hleb\_or\_skot);

this->Controls->Add(this->sell\_butt);

this->Controls->Add(this->num\_sell);

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

this->Controls->Add(this->price\_skot\_text);

this->Controls->Add(this->price\_hleb\_text);

this->MaximizeBox = false;

this->MaximumSize = System::Drawing::Size(587, 410);

this->MinimizeBox = false;

this->MinimumSize = System::Drawing::Size(587, 410);

this->Name = L"sell\_tovar";

this->ShowIcon = false;

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

this->Text = L"Товарная биржа";

this->Activated += gcnew System::EventHandler(this, &sell\_tovar::sell\_tovar\_Activated);

this->hleb\_or\_skot->ResumeLayout(false);

this->hleb\_or\_skot->PerformLayout();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void num\_sell\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

int zapas\_hleb=derevn.get\_kolvo\_hleb(); //получить кол-во хлеба в деревне

int zapas\_skot=derevn.get\_kolvo\_skot(); //получить кол-во скота в деревне

if (this->num\_sell->Text!=L"") //если поле с номером строки не пустое

{

if(this->hleb\_radio->Checked == true) //если выбран хлеб для продажи

{

if((Convert::ToDouble(this->num\_sell->Text) > derevn.get\_kolvo\_hleb()) || (Convert::ToDouble(this->num\_sell->Text) <= 0))

{

this->sell\_butt->Enabled = false;

}

else

{

this->sell\_butt->Enabled = true;

}

}

if(this->skot\_radio->Checked == true) //если выбран скот для продажи

{

if((Convert::ToDouble(this->num\_sell->Text) > derevn.get\_kolvo\_skot()) || (Convert::ToDouble(this->num\_sell->Text) <= 0))

{

this->sell\_butt->Enabled = false;

}

else

{

this->sell\_butt->Enabled = true;

}

}

}

else

this->sell\_butt->Enabled = false;

}

private: System::Void sell\_tovar\_Activated(System::Object^ sender, System::EventArgs^ e) {

this->price\_hleb\_text->Text = L"Текущая цена за ед. хлеба: " + Convert::ToString(derevn.get\_price\_hleb()) + L" руб.";

this->price\_skot\_text->Text = L"Текущая цена за голову скота: " + Convert::ToString(derevn.get\_price\_skot()) + L" руб.";

}

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

num\_sell\_TextChanged(sender, e);

}

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

if(this->hleb\_radio->Checked == true) //если выбран хлеб для продажи

{

derevn.set\_kolvo\_hleb(derevn.get\_kolvo\_hleb()-Convert::ToDouble(this->num\_sell->Text)); //вычесть выбранное кол-ва хлеба из ресурсов деревни

derevn.set\_budget\_village(derevn.get\_budget\_village()+Convert::ToDouble(this->num\_sell->Text)\*derevn.get\_price\_hleb()); //начислить деньги за продажу

}

if(this->skot\_radio->Checked == true) //если выбран скот для продажи

{

derevn.set\_kolvo\_skot(derevn.get\_kolvo\_skot()-Convert::ToDouble(this->num\_sell->Text)); //вычесть выбранное кол-ва скота из ресурсов деревни

derevn.set\_budget\_village(derevn.get\_budget\_village()+Convert::ToDouble(this->num\_sell->Text)\*derevn.get\_price\_skot()); //начислить деньги за продажу

}

num\_sell\_TextChanged(sender, e);

}

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

if(f\_endgame==1) //игра закончилась, закрыть все формы

Close();

else

{

num\_sell\_TextChanged(sender, e); //повторная проверка

sell\_tovar\_Activated(sender,e); //обновить таблицу

}

}

};

}

record\_tabl.h

#pragma once

#include "krest\_and\_village.h"

#include "tabl\_naim.h"

#include <string>

#include <fstream>

extern int f\_endgame; //флаг о том, что игра окончена (игрок проиграл) (глобальн.)

namespace kurs2020 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для record\_tabl

/// </summary>

public ref class record\_tabl : public System::Windows::Forms::Form

{

public:

record\_tabl(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~record\_tabl()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::DataGridView^ spisok;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ name\_tabl;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ rec\_tabl;

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

protected:

private:

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

#pragma region Windows Form Designer generated code

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

void InitializeComponent(void)

{

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(record\_tabl::typeid));

this->spisok = (gcnew System::Windows::Forms::DataGridView());

this->name\_tabl = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->rec\_tabl = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

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

this->SuspendLayout();

//

// spisok

//

this->spisok->AllowUserToAddRows = false;

this->spisok->AllowUserToDeleteRows = false;

this->spisok->AllowUserToResizeColumns = false;

this->spisok->AllowUserToResizeRows = false;

this->spisok->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::Top | System::Windows::Forms::AnchorStyles::Bottom));

this->spisok->BackgroundColor = System::Drawing::Color::Maroon;

this->spisok->ClipboardCopyMode = System::Windows::Forms::DataGridViewClipboardCopyMode::Disable;

this->spisok->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->spisok->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(2) {this->name\_tabl, this->rec\_tabl});

this->spisok->EditMode = System::Windows::Forms::DataGridViewEditMode::EditProgrammatically;

this->spisok->Location = System::Drawing::Point(12, 12);

this->spisok->MultiSelect = false;

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

this->spisok->ReadOnly = true;

this->spisok->RowHeadersVisible = false;

this->spisok->RowHeadersWidthSizeMode = System::Windows::Forms::DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders;

this->spisok->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->spisok->SelectionMode = System::Windows::Forms::DataGridViewSelectionMode::FullRowSelect;

this->spisok->ShowEditingIcon = false;

this->spisok->Size = System::Drawing::Size(503, 467);

this->spisok->TabIndex = 14;

this->spisok->TabStop = false;

//

// name\_tabl

//

this->name\_tabl->AutoSizeMode = System::Windows::Forms::DataGridViewAutoSizeColumnMode::None;

this->name\_tabl->Frozen = true;

this->name\_tabl->HeaderText = L"Имя игрока";

this->name\_tabl->MaxInputLength = 200;

this->name\_tabl->Name = L"name\_tabl";

this->name\_tabl->ReadOnly = true;

this->name\_tabl->SortMode = System::Windows::Forms::DataGridViewColumnSortMode::Programmatic;

this->name\_tabl->Width = 170;

//

// rec\_tabl

//

this->rec\_tabl->AutoSizeMode = System::Windows::Forms::DataGridViewAutoSizeColumnMode::None;

this->rec\_tabl->Frozen = true;

this->rec\_tabl->HeaderText = L"Значение рекорда (кол-во набранных Дней Процветания)";

this->rec\_tabl->MaxInputLength = 200;

this->rec\_tabl->Name = L"rec\_tabl";

this->rec\_tabl->ReadOnly = true;

this->rec\_tabl->SortMode = System::Windows::Forms::DataGridViewColumnSortMode::Programmatic;

this->rec\_tabl->Width = 330;

//

// record\_tabl

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

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

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

this->ClientSize = System::Drawing::Size(527, 491);

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

this->DoubleBuffered = true;

this->MinimizeBox = false;

this->MinimumSize = System::Drawing::Size(543, 530);

this->Name = L"record\_tabl";

this->ShowIcon = false;

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

this->Text = L"Список рекордов";

this->Load += gcnew System::EventHandler(this, &record\_tabl::record\_tabl\_Load);

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

this->ResumeLayout(false);

}

#pragma endregion

private: System::Void record\_tabl\_Activated(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void record\_tabl\_Load(System::Object^ sender, System::EventArgs^ e) {

ifstream fin; //создать объект класса ifstream

fin.open("records.txt", ios::in); //открыть файл

if(fin.is\_open()) //если файл возможно открыть

{

int kolvo\_rec=0; //начальное значение кол-ва рекордов в файле

string strread; //строка для чтения из файла

while(!fin.eof()) //пока не прочитает весь файл

{

getline(fin, strread, '\n'); //получить рекорд из файла

kolvo\_rec+=1; //увеличить кол-во рекордов на 1

}

kolvo\_rec=kolvo\_rec-1; //перенос строки после последнего рекорда засчитывается, поэтому нужно вычесть

fin.close(); //закрыть файл

fin.open("records.txt", ios::in); //открыть файл

for(int g=0; g<kolvo\_rec; g++)

{

spisok->Rows->Add(); //добавить новую строку в таблицу

getline(fin, strread, ';'); //получить имя игрока из файла

spisok->Rows[g]->Cells[0]->Value = gcnew System::String(strread.c\_str()); //внести в ячейку таблицы

getline(fin, strread, '\n'); //получить значение рекорда игрока из файла

spisok->Rows[g]->Cells[1]->Value = gcnew System::String(strread.c\_str()); //внести в ячейку таблицы

}

fin.close(); //закрыть файл

//сортировка игроков по убыванию рекордов (метод пузырька)

for(int v=1; v<kolvo\_rec; v++)

{

for(int b=0; b<kolvo\_rec-v; b++)

if(Convert::ToDouble(spisok->Rows[b]->Cells[1]->Value) < Convert::ToDouble(spisok->Rows[b+1]->Cells[1]->Value))

{

Object^ vsp0 = spisok->Rows[b+1]->Cells[0]->Value;

Object^ vsp1 = spisok->Rows[b+1]->Cells[1]->Value;

spisok->Rows[b+1]->Cells[0]->Value = spisok->Rows[b]->Cells[0]->Value;

spisok->Rows[b+1]->Cells[1]->Value = spisok->Rows[b]->Cells[1]->Value;

spisok->Rows[b]->Cells[0]->Value = vsp0;

spisok->Rows[b]->Cells[1]->Value = vsp1;

}

}

}

else

fin.close(); //закрыть файл

}

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

Close();

}

};

}

spravk.h

#pragma once

extern int f\_endgame; //флаг о том, что игра окончена (игрок проиграл) (глобальн.)

namespace kurs2020 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для spravk

/// </summary>

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

{

public:

spravk(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~spravk()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::TextBox^ spravka\_info;

protected:

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

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

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

private:

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

#pragma region Windows Form Designer generated code

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

void InitializeComponent(void)

{

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

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

this->spravka\_info = (gcnew System::Windows::Forms::TextBox());

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

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

this->SuspendLayout();

//

// spravka\_info

//

this->spravka\_info->Location = System::Drawing::Point(12, 12);

this->spravka\_info->Multiline = true;

this->spravka\_info->Name = L"spravka\_info";

this->spravka\_info->ReadOnly = true;

this->spravka\_info->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->spravka\_info->Size = System::Drawing::Size(311, 326);

this->spravka\_info->TabIndex = 0;

this->spravka\_info->TabStop = false;

this->spravka\_info->Text = resources->GetString(L"spravka\_info.Text");

this->spravka\_info->TextChanged += gcnew System::EventHandler(this, &spravk::spravka\_info\_TextChanged);

//

// cloz

//

this->cloz->Cursor = System::Windows::Forms::Cursors::Hand;

this->cloz->DialogResult = System::Windows::Forms::DialogResult::Cancel;

this->cloz->Location = System::Drawing::Point(12, 344);

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

this->cloz->Size = System::Drawing::Size(311, 23);

this->cloz->TabIndex = 1;

this->cloz->Text = L"Закрыть";

this->cloz->UseVisualStyleBackColor = true;

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

//

// timer\_endgame

//

this->timer\_endgame->Enabled = true;

this->timer\_endgame->Interval = 1003;

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

//

// spravk

//

this->AcceptButton = this->cloz;

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

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

this->CancelButton = this->cloz;

this->ClientSize = System::Drawing::Size(335, 373);

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

this->Controls->Add(this->spravka\_info);

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

this->MaximizeBox = false;

this->MaximumSize = System::Drawing::Size(351, 412);

this->MinimizeBox = false;

this->MinimumSize = System::Drawing::Size(351, 412);

this->Name = L"spravk";

this->ShowInTaskbar = false;

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

this->Text = L"Справка";

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

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

Close();

}

private: System::Void spravka\_info\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

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

if(f\_endgame==1) //игра закончилась, закрыть все формы

Close();

}

};

}

## Б.2 Исходный код реализации на C#

Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

namespace kurs2020\_sharp

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void quit\_butt\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void spravk\_butt\_Click(object sender, EventArgs e)

{

using (Form spr = new spravk())

{

spr.ShowDialog();

}

}

private void new\_game\_butt\_Click(object sender, EventArgs e)

{

Program.f\_endgame = 0; //флаг конца игры (будет равен 1, когда игрок проиграет)

Program.derevn.Init\_new\_game(); //инициализация деревни перед началом новой игры

using (Form mg = new main\_game())

{

mg.ShowDialog();

}

}

private void records\_butt\_Click(object sender, EventArgs e)

{

using (Form rectab = new record\_tabl())

{

rectab.ShowDialog();

}

}

private void load\_game\_butt\_Click(object sender, EventArgs e)

{

Program.f\_endgame = 0; //флаг конца игры (будет равен 1, когда игрок проиграет)

Program.derevn.Init\_load\_game(); //инициализация деревни из файла перед продолжением сохранённой игры

using (Form mg = new main\_game())

{

mg.ShowDialog();

}

}

private void Form1\_Activated(object sender, EventArgs e)

{

if(Program.f\_endgame==1)

{

Program.f\_endgame=0;

Form ig = new itog\_game();

ig.Show();

}

if(!(System.IO.File.Exists("village.txt")) || !(System.IO.File.Exists("krests.txt"))) //проверка, есть ли файлы сохранения. Если хотя бы одного нет, то кнопка загрузки игры отключается

{

this.load\_game\_butt.Enabled=false;

}

else

{

this.load\_game\_butt.Enabled=true;

}

}

}

}

main\_game.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace kurs2020\_sharp

{

public partial class main\_game : Form

{

public main\_game()

{

InitializeComponent();

}

private void go\_to\_main\_menu\_butt\_Click(object sender, EventArgs e)

{

this.Close();

}

private void spravk\_lent\_butt\_Click(object sender, EventArgs e)

{

using (Form sprv = new spravk())

{

sprv.ShowDialog();

}

}

private void spisok\_krest\_butt\_Click(object sender, EventArgs e)

{

using (Form tablk = new tabl\_krest())

{

tablk.ShowDialog();

}

}

private void birzha\_butt\_Click(object sender, EventArgs e)

{

using (Form sellt = new sell\_tovar())

{

sellt.ShowDialog();

}

}

private void main\_game\_Load(object sender, EventArgs e)

{

this.kolvo\_krest\_text.Text = "Крестьяне: " + (Program.derevn.get\_kolvo\_krest())+ " чел.";;

this.skorost\_zhiz\_text.Text = "Скорость жизни: " + (Program.derevn.get\_speed\_life());

this.den\_procv\_text.Text = "Дни Процветания: " + (Program.derevn.get\_days\_procv());

this.budget\_text.Text = "Бюджет: " + (Program.derevn.get\_budget\_village()) + " руб.";

this.hleb\_text.Text = "Хлеб: " + (Program.derevn.get\_kolvo\_hleb()) + " ед.";

this.skot\_text.Text = "Домашний скот: " + (Program.derevn.get\_kolvo\_skot()) + " ед.";

if(Program.derevn.get\_flag\_season()==0)

this.time\_helper.Text = "До конца Лета осталось " + (40 -(Program.derevn.get\_hours\_procv() - Program.derevn.get\_hours\_from\_begin\_of\_season())) + " Часов Процветания.";

if(Program.derevn.get\_flag\_season()==1)

{

this.time\_helper.Text = "До конца Зимы осталось " + (20 -(Program.derevn.get\_hours\_procv() - Program.derevn.get\_hours\_from\_begin\_of\_season())) + " Часов Процветания.";

this.BackgroundImage = Image.FromFile("zima.jpg"); //изменить фон на зимний

}

}

private void main\_timer\_Tick(object sender, EventArgs e)

{

this.save\_complete.Visible = false;

if(Program.f\_endgame==1) //игра закончилась, закрыть все формы

this.Close();

Program.derevn.set\_hours\_procv(Program.derevn.get\_hours\_procv()+1); //кол-во часов процветания увеличилось на 1

if((Program.derevn.get\_hours\_procv()%10)==0) //если прошло 10 часов процветания

{

Program.derevn.set\_days\_procv(Program.derevn.get\_days\_procv()+1);

}

if(Program.derevn.get\_kolvo\_krest()==0) //если нет крестьян

{

Program.derevn.set\_otschet\_hours\_net\_krest(Program.derevn.get\_otschet\_hours\_net\_krest()+1);

}

if(Program.derevn.get\_otschet\_hours\_net\_krest()==2) //если закончилось время отсчёта без крестьян

{

Program.f\_endgame=1;

this.Close();

}

if(Program.derevn.get\_kolvo\_krest()!=0) //если хотя бы один крестьянин нашёлся

{

Program.derevn.set\_otschet\_hours\_net\_krest(0);

}

if(Program.derevn.get\_flag\_season()==0 && Program.derevn.get\_budget\_village()!=0) //если лето и бюджет не нулевой

{

int kolv\_k=Program.derevn.get\_kolvo\_krest();

for(int ko=0; ko<kolv\_k; ko++)

{

if((Program.derevn.get\_budget\_village()-(Program.derevn.krests[ko].get\_money\_trat()\*Program.derevn.get\_speed\_life()))>=0)

{

Program.derevn.set\_kolvo\_hleb(Program.derevn.get\_kolvo\_hleb()+Program.derevn.krests[ko].get\_proizv\_hleb());

Program.derevn.set\_kolvo\_skot(Program.derevn.get\_kolvo\_skot()+Program.derevn.krests[ko].get\_proizv\_skot());

Program.derevn.set\_budget\_village(Program.derevn.get\_budget\_village()-(Program.derevn.krests[ko].get\_money\_trat()\*Program.derevn.get\_speed\_life()));

}

else

break;

}

}

if(Program.derevn.get\_flag\_season()==1) //если зима

{

int kolv\_k=Program.derevn.get\_kolvo\_krest();

for(int ko=0; ko<kolv\_k; ko++)

{

Program.derevn.set\_kolvo\_hleb(Program.derevn.get\_kolvo\_hleb()-(Program.derevn.krests[ko].get\_eda\_hleb()\*Program.derevn.get\_speed\_life()));

Program.derevn.set\_kolvo\_skot(Program.derevn.get\_kolvo\_skot()-(Program.derevn.krests[ko].get\_eda\_skot()\*Program.derevn.get\_speed\_life()));

if(Program.derevn.get\_kolvo\_hleb()>=0 && Program.derevn.get\_kolvo\_skot()>=0)

Program.derevn.krests[ko].set\_otschet\_hours\_net\_edi(0);

if(Program.derevn.get\_kolvo\_hleb()<0 || Program.derevn.get\_kolvo\_skot()<0)

{

if(Program.derevn.get\_kolvo\_hleb()<0)

{

Program.derevn.set\_kolvo\_hleb(0);

}

if(Program.derevn.get\_kolvo\_skot()<0)

{

Program.derevn.set\_kolvo\_skot(0);

}

Program.derevn.krests[ko].set\_otschet\_hours\_net\_edi(Program.derevn.krests[ko].get\_otschet\_hours\_net\_edi()+1);

if(Program.derevn.krests[ko].get\_otschet\_hours\_net\_edi()>=5)

{

string str\_name= Program.derevn.krests[ko].get\_name() + " " + Program.derevn.krests[ko].get\_surname(); //для вывода имени и фамилии

this.event\_helper.Text = "Последнее событие: " + (DateTime.Now.Hour)+ ":" + (DateTime.Now.Minute)+ ":" + (DateTime.Now.Second) + " " + "Крестьянин " + str\_name + " ушёл из деревни из-за недостатка пищи.";

Program.derevn.Delete\_krest(ko+1); //удалить крестьянина (методу передаётся нумерация от единицы, а цикл нумеруется от нуля)

kolv\_k=kolv\_k-1; //крестьян стало на 1 меньше

if(kolv\_k==ko)

break;

}

}

}

}

if(Program.derevn.get\_flag\_season()==0 && (Program.derevn.get\_hours\_procv() - Program.derevn.get\_hours\_from\_begin\_of\_season())==40) //если сейчас лето и прошло 40 часов П. от начала лета (4 дня П. или 2 минуты)

{

Program.derevn.set\_flag\_season(1); //теперь флаг соответствует зиме

this.BackgroundImage = Image.FromFile("zima.jpg"); //изменить фон на зимний

Program.derevn.set\_hours\_from\_begin\_of\_season(Program.derevn.get\_hours\_procv());

}

if(Program.derevn.get\_flag\_season()==1 && (Program.derevn.get\_hours\_procv() - Program.derevn.get\_hours\_from\_begin\_of\_season())==20) //если сейчас зима и прошло 20 часов П. от начала зимы (2 дня П. или 1 минута)

{

Program.derevn.set\_flag\_season(0); //теперь флаг соответствует лету

this.BackgroundImage = Image.FromFile("leto.jpg"); //изменить фон на летний

Program.derevn.set\_hours\_from\_begin\_of\_season(Program.derevn.get\_hours\_procv());

Program.derevn.set\_speed\_life( Program.derevn.get\_speed\_life()+1); //множитель Скорость жизни увеличился на 1

int f\_izmen\_price; //флаг изменения цены на товар (0, если уменьшить; 1, если увеличить)

double proc\_izm; //процент от начального значения цены, на которое она изменится

//ДЛЯ ЦЕНЫ НА ХЛЕБ

f\_izmen\_price = Program.randk.Next(0, 2); //случайное число в интервале от 0 до 1 включительно

proc\_izm = Program.randk.Next(5, 31); //случайное число в интервале от 5 до 30 включительно

proc\_izm = proc\_izm / 100; //перевод процентного значения в десятичное

if(f\_izmen\_price==0)

{

Program.derevn.set\_price\_hleb(Convert.ToInt32(Program.derevn.get\_price\_hleb() - Program.derevn.get\_price\_hleb() \* proc\_izm));

}

if(f\_izmen\_price==1)

{

Program.derevn.set\_price\_hleb(Convert.ToInt32(Program.derevn.get\_price\_hleb() + Program.derevn.get\_price\_hleb() \* proc\_izm));

}

//ДЛЯ ЦЕНЫ НА СКОТ

f\_izmen\_price = Program.randk.Next(0, 2); //случайное число в интервале от 0 до 1 включительно

proc\_izm = Program.randk.Next(5, 31); //случайное число в интервале от 5 до 30 включительно

proc\_izm = proc\_izm / 100; //перевод процентного значения в десятичное

if(f\_izmen\_price==0)

{

Program.derevn.set\_price\_skot(Convert.ToInt32(Program.derevn.get\_price\_skot() - Program.derevn.get\_price\_skot() \* proc\_izm));

}

if(f\_izmen\_price==1)

{

Program.derevn.set\_price\_skot(Convert.ToInt32(Program.derevn.get\_price\_skot() + Program.derevn.get\_price\_skot() \* proc\_izm));

}

}

if(Program.derevn.get\_flag\_season()==1 && (Program.derevn.get\_hours\_procv() - Program.derevn.get\_hours\_from\_begin\_of\_season())==10) //если сейчас зима и прошло 10 часов П. от начала зимы (середина зимы) (1 день П. или 0,5 минуты)

{

int f\_event; //флаг случайного события (0 - приход нового крестьянина, 1 - нападение волков на скот)

f\_event =Program.randk.Next(0, 2); //случайное число в интервале от 0 до 1 включительно

double proc\_izm; //процент от начального значения кол-ва скота, на которое оно изменится

proc\_izm = Program.randk.Next(5, 31); //случайное число в интервале от 5 до 30 включительно

proc\_izm = proc\_izm / 100; //перевод процентного значения в десятичное

int poteri = Convert.ToInt32(Program.derevn.get\_kolvo\_skot() \* proc\_izm); //отдельная переменная для кол-ва потерянного скота

if(f\_event==1 && poteri>0) //нападение волков

{

Program.derevn.set\_kolvo\_skot(Program.derevn.get\_kolvo\_skot() - poteri);

this.event\_helper.Text = "Последнее событие: " + (DateTime.Now.Hour) + ":" + (DateTime.Now.Minute) + ":" + (DateTime.Now.Second) + " " + "Стая волков из леса напала на ваш скот. Потеряно " + (poteri) + " ед. скота.";

}

else //приход крестьянина

{

Program.derevn.Add\_rand\_krest(); //добавление нового случайного крестьянина

int koo = Program.derevn.get\_kolvo\_krest() - 1; //получить номер последнего крестьянина

string str\_name= Program.derevn.krests[koo].get\_name() + " " + Program.derevn.krests[koo].get\_surname(); //для вывода имени и фамилии

this.event\_helper.Text = "Последнее событие: " + (DateTime.Now.Hour) + ":" + (DateTime.Now.Minute) + ":" + (DateTime.Now.Second) + " " + "Крестьянин " + str\_name + " вышел из лесу и присоединился к вам.";

}

}

//ВЫВОД ИНФОРМАЦИИ О ДЕРЕВНЕ

this.kolvo\_krest\_text.Text = "Крестьяне: " + (Program.derevn.get\_kolvo\_krest())+ " чел.";;

this.skorost\_zhiz\_text.Text = "Скорость жизни: " + (Program.derevn.get\_speed\_life());

this.den\_procv\_text.Text = "Дни Процветания: " + (Program.derevn.get\_days\_procv());

this.budget\_text.Text = "Бюджет: " + (Program.derevn.get\_budget\_village()) + " руб.";

this.hleb\_text.Text = "Хлеб: " + (Program.derevn.get\_kolvo\_hleb()) + " ед.";

this.skot\_text.Text = "Домашний скот: " + (Program.derevn.get\_kolvo\_skot()) + " ед.";

if(Program.derevn.get\_flag\_season()==0)

this.time\_helper.Text = "До конца Лета осталось " + (40 -(Program.derevn.get\_hours\_procv() - Program.derevn.get\_hours\_from\_begin\_of\_season())) + " Часов Процветания.";

if(Program.derevn.get\_flag\_season()==1)

this.time\_helper.Text = "До конца Зимы осталось " + (20 -(Program.derevn.get\_hours\_procv() - Program.derevn.get\_hours\_from\_begin\_of\_season())) + " Часов Процветания.";

}

private void timer\_refresh\_Tick(object sender, EventArgs e)

{

//ВЫВОД ИНФОРМАЦИИ О ДЕРЕВНЕ

this.kolvo\_krest\_text.Text = "Крестьяне: " + (Program.derevn.get\_kolvo\_krest()) + " чел."; ;

this.skorost\_zhiz\_text.Text = "Скорость жизни: " + (Program.derevn.get\_speed\_life());

this.den\_procv\_text.Text = "Дни Процветания: " + (Program.derevn.get\_days\_procv());

this.budget\_text.Text = "Бюджет: " + (Program.derevn.get\_budget\_village()) + " руб.";

this.hleb\_text.Text = "Хлеб: " + (Program.derevn.get\_kolvo\_hleb()) + " ед.";

this.skot\_text.Text = "Домашний скот: " + (Program.derevn.get\_kolvo\_skot()) + " ед.";

if (Program.derevn.get\_flag\_season() == 0)

this.time\_helper.Text = "До конца Лета осталось " + (40 - (Program.derevn.get\_hours\_procv() - Program.derevn.get\_hours\_from\_begin\_of\_season())) + " Часов Процветания.";

if (Program.derevn.get\_flag\_season() == 1)

this.time\_helper.Text = "До конца Зимы осталось " + (20 - (Program.derevn.get\_hours\_procv() - Program.derevn.get\_hours\_from\_begin\_of\_season())) + " Часов Процветания.";

}

private void save\_game\_butt\_Click(object sender, EventArgs e)

{

Program.derevn.Save\_game(); //сохранить игру в файл

this.save\_complete.Visible = true; //показать надпись об успешном сохранении

}

}

}

krest\_and\_village.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace kurs2020\_sharp

{

class Krest

{

private string name; //Имя

private string surname; //Фамилия

private int proizv\_hleb; //Кол-во производимого хлеба за Час Процветания

private int proizv\_skot; //Кол-во производимого домашнего скота за Час Процветания

private int eda\_hleb; //Прожорливость: хлеб

private int eda\_skot; //Прожорливость: домашний скот

private int money\_trat; //Денежные расходы за один Час Процветания

private int naim; //Стоимость найма в рублях

private int otschet\_hours\_net\_edi; //Отсчёт времени в Часах Процветания с момента, когда закончилась еда для этого крестьянина

public Krest() //конструктор без параметров

{

name="name";

surname="surname";

proizv\_hleb=0;

proizv\_skot=0;

eda\_hleb=0;

eda\_skot=0;

money\_trat=0;

naim=0;

otschet\_hours\_net\_edi=0;

}

public void Init\_rand() //инициализация полей случайно (в фиксированном диапазоне) для новых крестьян

{

string path= "names.txt";

int kolvo\_name=0; //начальное значение кол-ва слов в файле

//ПОЛУЧИТЬ СЛУЧАЙНОЕ ИМЯ ИЗ ФАЙЛА

using (StreamReader reader = new StreamReader(path)) //открытие файла для чтения

{

name="name";

while(name!=null) //пока не прочитает весь файл

{

name=reader.ReadLine(); //получить строку из файла

kolvo\_name+=1; //увеличить кол-во слов на 1

}

reader.Close(); //закрыть файл

}

int num\_name = Program.randk.Next(0, kolvo\_name-1); //получить случайный номер слова из общего кол-ва

using (StreamReader reader = new StreamReader(path)) //открытие файла для чтения

{

for (int i\_n=0; ; i\_n++)

{

name=reader.ReadLine(); //получить строку из файла

if (i\_n==num\_name) //если это искомое слово, остановить цикл

break;

}

reader.Close(); //закрыть файл

}

//ПОЛУЧИТЬ СЛУЧАЙНУЮ ФАМИЛИЮ ИЗ ФАЙЛА

path= "surnames.txt";

int kolvo\_sur=0; //начальное значение кол-ва слов в файле

using (StreamReader reader = new StreamReader(path)) //открытие файла для чтения

{

surname="surname";

while(surname!=null) //пока не прочитает весь файл

{

surname=reader.ReadLine(); //получить строку из файла

kolvo\_sur+=1; //увеличить кол-во слов на 1

}

reader.Close(); //закрыть файл

}

int num\_sur = Program.randk.Next(0, kolvo\_name-1); //получить случайный номер слова из общего кол-ва

using (StreamReader reader = new StreamReader(path)) //открытие файла для чтения

{

for (int i\_nn=0; ; i\_nn++)

{

surname=reader.ReadLine(); //получить строку из файла

if (i\_nn==num\_sur) //если это искомое слово, остановить цикл

break;

}

reader.Close(); //закрыть файл

}

proizv\_hleb = Program.randk.Next(1, 11); //случайное число в интервале от 1 до 10 включительно

proizv\_skot = Program.randk.Next(0, 3); //случайное число в интервале от 0 до 2 включительно

eda\_hleb = Program.randk.Next(1, 4); //случайное число в интервале от 1 до 3 включительно

eda\_skot = Program.randk.Next(0, 2); //случайное число в интервале от 0 до 1 включительно

money\_trat = Program.randk.Next(10, 101); //случайное число в интервале от 10 до 100 включительно

naim = Program.randk.Next(5, 51); //случайное число в интервале от 5 до 50 включительно

otschet\_hours\_net\_edi=0;

}

//Получение и установление соответствующих полей для крестьянина

public void set\_name(string inn)

{

name=inn;

}

public string get\_name()

{

return name;

}

/////

public void set\_surname(string inn)

{

surname=inn;

}

public string get\_surname()

{

return surname;

}

/////

public void set\_proizv\_hleb(int inn)

{

proizv\_hleb=inn;

}

public int get\_proizv\_hleb()

{

return proizv\_hleb;

}

/////

public void set\_proizv\_skot(int inn)

{

proizv\_skot=inn;

}

public int get\_proizv\_skot()

{

return proizv\_skot;

}

/////

public void set\_eda\_hleb(int inn)

{

eda\_hleb=inn;

}

public int get\_eda\_hleb()

{

return eda\_hleb;

}

/////

public void set\_eda\_skot(int inn)

{

eda\_skot=inn;

}

public int get\_eda\_skot()

{

return eda\_skot;

}

/////

public void set\_money\_trat(int inn)

{

money\_trat=inn;

}

public int get\_money\_trat()

{

return money\_trat;

}

/////

public void set\_naim(int inn)

{

naim=inn;

}

public int get\_naim()

{

return naim;

}

/////

public void set\_otschet\_hours\_net\_edi(int inn)

{

otschet\_hours\_net\_edi=inn;

}

public int get\_otschet\_hours\_net\_edi()

{

return otschet\_hours\_net\_edi;

}

}

class Village

{

private int kolvo\_krest; //Кол-во крестьян в деревне

private int hours\_procv; //Кол-во Часов Процветания с момента начала игры

private int days\_procv; //Кол-во Дней Процветания с момента начала игры (и финальный счёт)

private int speed\_life; //Текущий множитель «Скорость Жизни»

private int flag\_season; //Флаг текущего сезона (равен нулю Летом, равен единице Зимой)

private int kolvo\_hleb; //Общее кол-во хлеба в деревне

private int kolvo\_skot; //Общее кол-во голов домашнего скота в деревне

private int budget\_village; //Бюджет деревни в рублях

private int hours\_from\_begin\_of\_season; //Кол-во Часов Процветания в момент, когда текущее время года только началось

private int otschet\_hours\_net\_krest; //Отсчёт времени в Часах Процветания с момента, когда ушли все крестьяне

private int price\_hleb; //Текущая цена за ед. хлеба на Торговой бирже в рублях

private int price\_skot; //Текущая цена за одну голову домашнего скота на Торговой бирже в рублях

public Krest[] krests = null; //крестьяне

public Village() //конструктор без параметров

{

kolvo\_krest=0;

hours\_procv=0;

days\_procv=0;

speed\_life=0;

flag\_season=0;

kolvo\_hleb=0;

kolvo\_skot=0;

budget\_village=0;

hours\_from\_begin\_of\_season=0;

otschet\_hours\_net\_krest=0;

price\_hleb=0;

price\_skot=0;

krests=null;

}

public void Init\_new\_game() //инициализация при начале новой игры

{

if(krests!=null)

{

krests = null;

}

kolvo\_krest=3;

hours\_procv=1;

days\_procv=0;

speed\_life=1;

flag\_season=0;

kolvo\_hleb=200;

kolvo\_skot=20;

budget\_village=1000;

hours\_from\_begin\_of\_season=0;

otschet\_hours\_net\_krest=0;

price\_hleb=30;

price\_skot=200;

krests = new Krest [3];

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

{

krests[kk] = new Krest();

}

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

{

krests[i].Init\_rand();

}

}

public void Save\_game() //сохранение игры (запись информации о деревне в файл)

{

string path= "village.txt"; //путь для записи информации о деревне (крестьяне отдельно)

using (StreamWriter writer = new StreamWriter(path)) //открытие файла для записи

{

writer.WriteLine(kolvo\_krest); //записать в файл

writer.WriteLine(hours\_procv); //записать в файл

writer.WriteLine(days\_procv); //записать в файл

writer.WriteLine(speed\_life); //записать в файл

writer.WriteLine(flag\_season); //записать в файл

writer.WriteLine(kolvo\_hleb); //записать в файл

writer.WriteLine(kolvo\_skot); //записать в файл

writer.WriteLine(budget\_village); //записать в файл

writer.WriteLine(hours\_from\_begin\_of\_season); //записать в файл

writer.WriteLine(otschet\_hours\_net\_krest); //записать в файл

writer.WriteLine(price\_hleb); //записать в файл

writer.WriteLine(price\_skot); //записать в файл

writer.Close(); //закрыть файл

}

path= "krests.txt"; //путь для записи информации о крестьянах

using (StreamWriter writer = new StreamWriter(path)) //открытие файла для чтения

{

for (int m = 0; m < kolvo\_krest; m++)

{

writer.WriteLine(krests[m].get\_name()); //записать в файл

writer.WriteLine(krests[m].get\_surname()); //записать в файл

writer.WriteLine(krests[m].get\_proizv\_hleb()); //записать в файл

writer.WriteLine(krests[m].get\_proizv\_skot()); //записать в файл

writer.WriteLine(krests[m].get\_eda\_hleb()); //записать в файл

writer.WriteLine(krests[m].get\_eda\_skot()); //записать в файл

writer.WriteLine(krests[m].get\_money\_trat()); //записать в файл

writer.WriteLine(krests[m].get\_naim()); //записать в файл

writer.WriteLine(krests[m].get\_otschet\_hours\_net\_edi()); //записать в файл

writer.WriteLine("///////"); //записать в файл

}

writer.Close(); //закрыть файл

}

}

public void Init\_load\_game() //инициализация при загрузке сохранённой игры

{

if(krests!=null)

{

krests = null;

}

string path= "village.txt"; //путь для чтения информации о деревне (крестьяне отдельно)

string strread; //строка для чтения из файла

using (StreamReader reader = new StreamReader(path)) //открытие файла для чтения

{

kolvo\_krest=Convert.ToInt32(reader.ReadLine()); //получить строку из файла, записать в нужное поле

hours\_procv=Convert.ToInt32(reader.ReadLine()); //получить строку из файла, записать в нужное поле

days\_procv=Convert.ToInt32(reader.ReadLine()); //получить строку из файла, записать в нужное поле

speed\_life=Convert.ToInt32(reader.ReadLine()); //получить строку из файла, записать в нужное поле

flag\_season=Convert.ToInt32(reader.ReadLine()); //получить строку из файла, записать в нужное поле

kolvo\_hleb=Convert.ToInt32(reader.ReadLine()); //получить строку из файла, записать в нужное поле

kolvo\_skot=Convert.ToInt32(reader.ReadLine()); //получить строку из файла, записать в нужное поле

budget\_village=Convert.ToInt32(reader.ReadLine()); //получить строку из файла, записать в нужное поле

hours\_from\_begin\_of\_season=Convert.ToInt32(reader.ReadLine()); //получить строку из файла, записать в нужное поле

otschet\_hours\_net\_krest=Convert.ToInt32(reader.ReadLine()); //получить строку из файла, записать в нужное поле

price\_hleb=Convert.ToInt32(reader.ReadLine()); //получить строку из файла, записать в нужное поле

price\_skot=Convert.ToInt32(reader.ReadLine()); //получить строку из файла, записать в нужное поле

reader.Close(); //закрыть файл

}

krests = new Krest [kolvo\_krest];

for (int kk = 0; kk < kolvo\_krest; kk++)

{

krests[kk] = new Krest();

}

path = "krests.txt"; //путь для чтения информации о крестьянах

using (StreamReader reader = new StreamReader(path)) //открытие файла для чтения

{

for (int m = 0; m < kolvo\_krest; m++)

{

krests[m].set\_name(reader.ReadLine()); //получить строку из файла, записать в нужное поле

krests[m].set\_surname(reader.ReadLine()); //получить строку из файла, записать в нужное поле

krests[m].set\_proizv\_hleb(Convert.ToInt32(reader.ReadLine())); //получить строку из файла, записать в нужное поле

krests[m].set\_proizv\_skot(Convert.ToInt32(reader.ReadLine())); //получить строку из файла, записать в нужное поле

krests[m].set\_eda\_hleb(Convert.ToInt32(reader.ReadLine())); //получить строку из файла, записать в нужное поле

krests[m].set\_eda\_skot(Convert.ToInt32(reader.ReadLine())); //получить строку из файла, записать в нужное поле

krests[m].set\_money\_trat(Convert.ToInt32(reader.ReadLine())); //получить строку из файла, записать в нужное поле

krests[m].set\_naim(Convert.ToInt32(reader.ReadLine())); //получить строку из файла, записать в нужное поле

krests[m].set\_otschet\_hours\_net\_edi(Convert.ToInt32(reader.ReadLine())); //получить строку из файла, записать в нужное поле

strread = reader.ReadLine(); //прочитать строку "///////", отделяющую данные разных крестьян

}

reader.Close(); //закрыть файл

}

}

public void Delete\_krest(int number) //удаление крестьянина с выбранным номером (нумерация от единицы!)

{

Krest[] kre\_change = new Krest[kolvo\_krest-1]; //временный массив крестьян

for (int kk = 0; kk < (kolvo\_krest - 1); kk++)

{

kre\_change[kk] = new Krest();

}

for(int j=0, k1=0; j<kolvo\_krest; j++) //перебрать весь массив крестьян в деревне

{

if(j!=(number-1)) //если номер крестьянина не равен искомому, то скопировать его в новый массив

{

kre\_change[k1]=this.krests[j];

k1++;

}

}

krests = null; //удалить старый массив

krests = kre\_change; //теперь указатель указывает на новый массив

kolvo\_krest=kolvo\_krest-1; //теперь на 1 крестьянина меньше

}

public void Add\_rand\_krest() //добавление случайного крестьянина к имеющимся (особое событие, возможное зимой)

{

Krest[] kre\_change = new Krest[kolvo\_krest+1]; //временный массив крестьян

for (int kk = 0; kk < (kolvo\_krest + 1); kk++)

{

kre\_change[kk] = new Krest();

}

int j;

for(j=0; j<kolvo\_krest; j++) //перебрать весь массив крестьян в деревне

{

kre\_change[j]=this.krests[j];

}

kre\_change[j].Init\_rand(); //новый крестьянин генерируется случайно

krests = null; //удалить старый массив

krests = kre\_change; //теперь указатель указывает на новый массив

kolvo\_krest=kolvo\_krest+1; //теперь на 1 крестьянина больше

}

public void Naim\_krest(Village vil\_naim, int numk) //найм крестьянина с выбранным номером из биржи труда (нумерация от единицы!)

{

Krest[] kre\_change = new Krest[kolvo\_krest+1]; //временный массив крестьян

for (int kk = 0; kk < (kolvo\_krest + 1); kk++)

{

kre\_change[kk] = new Krest();

}

int j;

for(j=0; j<kolvo\_krest; j++) //перебрать весь массив крестьян в деревне

{

kre\_change[j]=this.krests[j];

}

kre\_change[j]=vil\_naim.krests[numk-1]; //новый крестьянин берётся из массива найма под нужным номером (нумерация от единицы, а не от нуля!)

krests = null; //удалить старый массив

krests = kre\_change; //теперь указатель указывает на новый массив

kolvo\_krest=kolvo\_krest+1; //теперь на 1 крестьянина больше

}

//Получение и установление соответствующих полей для деревни

public void set\_kolvo\_krest(int inn)

{

kolvo\_krest=inn;

}

public int get\_kolvo\_krest()

{

return kolvo\_krest;

}

/////

public void set\_hours\_procv(int inn)

{

hours\_procv=inn;

}

public int get\_hours\_procv()

{

return hours\_procv;

}

/////

public void set\_days\_procv(int inn)

{

days\_procv=inn;

}

public int get\_days\_procv()

{

return days\_procv;

}

/////

public void set\_speed\_life(int inn)

{

speed\_life=inn;

}

public int get\_speed\_life()

{

return speed\_life;

}

/////

public void set\_flag\_season(int inn)

{

flag\_season=inn;

}

public int get\_flag\_season()

{

return flag\_season;

}

/////

public void set\_kolvo\_hleb(int inn)

{

kolvo\_hleb=inn;

}

public int get\_kolvo\_hleb()

{

return kolvo\_hleb;

}

/////

public void set\_kolvo\_skot(int inn)

{

kolvo\_skot=inn;

}

public int get\_kolvo\_skot()

{

return kolvo\_skot;

}

/////

public void set\_budget\_village(int inn)

{

budget\_village=inn;

}

public int get\_budget\_village()

{

return budget\_village;

}

/////

public void set\_hours\_from\_begin\_of\_season(int inn)

{

hours\_from\_begin\_of\_season=inn;

}

public int get\_hours\_from\_begin\_of\_season()

{

return hours\_from\_begin\_of\_season;

}

/////

public void set\_otschet\_hours\_net\_krest(int inn)

{

otschet\_hours\_net\_krest=inn;

}

public int get\_otschet\_hours\_net\_krest()

{

return otschet\_hours\_net\_krest;

}

/////

public void set\_price\_hleb(int inn)

{

price\_hleb=inn;

}

public int get\_price\_hleb()

{

return price\_hleb;

}

/////

public void set\_price\_skot(int inn)

{

price\_skot=inn;

}

public int get\_price\_skot()

{

return price\_skot;

}

}

}

tabl\_krest.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace kurs2020\_sharp

{

public partial class tabl\_krest : Form

{

public tabl\_krest()

{

InitializeComponent();

}

private void trud\_butt\_Click(object sender, EventArgs e)

{

using (Form tabnaim = new tabl\_naim())

{

tabnaim.ShowDialog();

}

}

private void tabl\_krest\_Activated(object sender, EventArgs e)

{

if(Program.f\_endgame==1) //игра закончилась, закрыть все формы

this.Close();

else

{

//очистить все строки перед выводом

while (0 != spisok.RowCount)

spisok.Rows.RemoveAt(0);

int kolvo\_str=Program.derevn.get\_kolvo\_krest(); //получить кол-во крестьян в списке

for(int num=0; num<kolvo\_str; num++)

{

spisok.Rows.Add(); //добавить новую строку в таблицу

spisok.Rows[num].Cells[0].Value = (num+1); //записать порядковый номер крестьянина

spisok.Rows[num].Cells[1].Value = (Program.derevn.krests[num].get\_name() + " " + Program.derevn.krests[num].get\_surname()); //внести в ячейку таблицы

spisok.Rows[num].Cells[2].Value = (Program.derevn.krests[num].get\_proizv\_hleb()) + "/" + (Program.derevn.krests[num].get\_proizv\_skot());

spisok.Rows[num].Cells[3].Value = (Program.derevn.krests[num].get\_eda\_hleb()\*Program.derevn.get\_speed\_life()) + "/" + (Program.derevn.krests[num].get\_eda\_skot()\*Program.derevn.get\_speed\_life());

spisok.Rows[num].Cells[4].Value = (Program.derevn.krests[num].get\_money\_trat()\*Program.derevn.get\_speed\_life());

}

}

}

private void num\_krest\_pole\_TextChanged(object sender, EventArgs e)

{

int kolvo\_strr=Program.derevn.get\_kolvo\_krest(); //получить кол-во крестьян в списке

if (this.num\_krest\_pole.Text!="") //если поле с номером строки не пустое

{

if((Convert.ToDouble(this.num\_krest\_pole.Text) > kolvo\_strr) || (Convert.ToDouble(this.num\_krest\_pole.Text)<1)) //если выбр. строка больше общего кол-ва строк, то отключить кнопку

{

this.izgnat\_butt.Enabled = false;

}

else

this.izgnat\_butt.Enabled = true;

}

else

this.izgnat\_butt.Enabled = false;

if(Program.derevn.get\_flag\_season()==1) //если зима, кнопка Изгнать недоступна в любом случае

{

this.label1.Enabled = false;

this.num\_krest\_pole.Enabled = false;

this.izgnat\_butt.Enabled = false;

this.trud\_butt.Enabled = false;

}

if(Program.derevn.get\_flag\_season()==0) //если лето, включить кнопку открытия Биржи труда

{

this.label1.Enabled = true;

this.num\_krest\_pole.Enabled = true;

this.trud\_butt.Enabled = true;

}

}

private void izgnat\_butt\_Click(object sender, EventArgs e)

{

Program.derevn.Delete\_krest(Convert.ToInt32(this.num\_krest\_pole.Text)); //удалить крестьянина

num\_krest\_pole\_TextChanged(sender, e); //повторная проверка

tabl\_krest\_Activated(sender,e); //обновить таблицу

}

private void tabl\_krest\_Load(object sender, EventArgs e)

{

num\_krest\_pole\_TextChanged(sender, e); //проверка

}

private void timer\_proverk\_krest\_Tick(object sender, EventArgs e)

{

if (Program.f\_endgame == 1) //игра закончилась, закрыть все формы

this.Close();

else

{

num\_krest\_pole\_TextChanged(sender, e); //повторная проверка

tabl\_krest\_Activated(sender, e); //обновить таблицу

}

}

}

}

tabl\_naim.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace kurs2020\_sharp

{

public partial class tabl\_naim : Form

{

public tabl\_naim()

{

InitializeComponent();

}

private void tabl\_naim\_Activated(object sender, EventArgs e)

{

if(Program.f\_endgame==1) //игра закончилась, закрыть все формы

Close();

else

{

//очистить все строки перед выводом

while (0 != spisok.RowCount)

spisok.Rows.RemoveAt(0);

int kolvo\_str=Program.birzh\_trud.get\_kolvo\_krest(); //получить кол-во крестьян в списке

for(int num=0; num<kolvo\_str; num++)

{

spisok.Rows.Add(); //добавить новую строку в таблицу

spisok.Rows[num].Cells[0].Value = (num+1); //записать порядковый номер крестьянина

spisok.Rows[num].Cells[1].Value = (Program.birzh\_trud.krests[num].get\_name() + " " + Program.birzh\_trud.krests[num].get\_surname()); //внести в ячейку таблицы

spisok.Rows[num].Cells[2].Value = (Program.birzh\_trud.krests[num].get\_proizv\_hleb()) + "/" + (Program.birzh\_trud.krests[num].get\_proizv\_skot());

spisok.Rows[num].Cells[3].Value = (Program.birzh\_trud.krests[num].get\_eda\_hleb() \* Program.derevn.get\_speed\_life()) + "/" + (Program.birzh\_trud.krests[num].get\_eda\_skot() \* Program.derevn.get\_speed\_life());

spisok.Rows[num].Cells[4].Value = (Program.birzh\_trud.krests[num].get\_money\_trat() \* Program.derevn.get\_speed\_life());

spisok.Rows[num].Cells[5].Value = (Program.birzh\_trud.krests[num].get\_naim());

}

}

}

private void tabl\_naim\_Load(object sender, EventArgs e)

{

if(Program.derevn.get\_flag\_season()==1) //если зима, закрыть форму

{

this.Close();

}

if(Program.birzh\_trud.krests!=null)

{

Program.birzh\_trud.krests=null;

}

Program.birzh\_trud.krests = new Krest[5]; //массив крестьян для найма

Program.birzh\_trud.set\_kolvo\_krest(5);

for (int kk = 0; kk < 5; kk++)

{

Program.birzh\_trud.krests[kk] = new Krest();

}

for(int v=0; v<5; v++)

{

Program.birzh\_trud.krests[v].Init\_rand();

}

}

private void num\_krest\_pole\_TextChanged(object sender, EventArgs e)

{

int kolvo\_strr = Program.birzh\_trud.get\_kolvo\_krest(); //получить кол-во крестьян в списке

if (this.num\_krest\_pole.Text !="") //если поле с номером строки не пустое

{

double nummk = Convert.ToDouble(this.num\_krest\_pole.Text); //номер выбранного крестьянина

if ((nummk > kolvo\_strr) || (nummk < 1)) //если выбр. строка больше общего кол-ва строк, то отключить кнопку

{

this.naim\_butt.Enabled = false;

}

else

{

this.naim\_butt.Enabled = true;

if ((Program.derevn.get\_budget\_village() - Program.birzh\_trud.krests[Convert.ToInt32(nummk) - 1].get\_naim()) < 0)

this.naim\_butt.Enabled = false;

}

}

else

this.naim\_butt.Enabled = false;

if (Program.derevn.get\_flag\_season() == 1) //если зима, закрыть форму

{

Close();

}

}

private void naim\_butt\_Click(object sender, EventArgs e)

{

int nummk = Convert.ToInt32(this.num\_krest\_pole.Text); //номер выбранного крестьянина

Program.derevn.set\_budget\_village(Program.derevn.get\_budget\_village() - Program.birzh\_trud.krests[nummk - 1].get\_naim());

Program.derevn.Naim\_krest(Program.birzh\_trud, nummk);

Program.birzh\_trud.Delete\_krest(nummk); //крестьянин перешёл из одного массива в другой. Из начального массива он удаляется

num\_krest\_pole\_TextChanged(sender,e); //повторная проверка

tabl\_naim\_Activated(sender,e); //обновить таблицу

}

private void timer\_proverk\_krest\_Tick(object sender, EventArgs e)

{

if (Program.f\_endgame == 1) //игра закончилась, закрыть все формы

Close();

else

{

num\_krest\_pole\_TextChanged(sender, e); //повторная проверка

if (Program.derevn.get\_flag\_season() == 0) //таблицу обновлять только летом

{

tabl\_naim\_Activated(sender, e); //обновить таблицу

}

}

}

}

}

sell\_tovar.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace kurs2020\_sharp

{

public partial class sell\_tovar : Form

{

public sell\_tovar()

{

InitializeComponent();

}

private void sell\_tovar\_Activated(object sender, EventArgs e)

{

this.price\_hleb\_text.Text = "Текущая цена за ед. хлеба: " + (Program.derevn.get\_price\_hleb()) + " руб.";

this.price\_skot\_text.Text = "Текущая цена за голову скота: " + (Program.derevn.get\_price\_skot()) + " руб.";

}

private void num\_sell\_TextChanged(object sender, EventArgs e)

{

int zapas\_hleb=Program.derevn.get\_kolvo\_hleb(); //получить кол-во хлеба в деревне

int zapas\_skot=Program.derevn.get\_kolvo\_skot(); //получить кол-во скота в деревне

if (this.num\_sell.Text!="") //если поле с номером строки не пустое

{

if(this.hleb\_radio.Checked == true) //если выбран хлеб для продажи

{

if ((Convert.ToDouble(this.num\_sell.Text) > Program.derevn.get\_kolvo\_hleb()) || (Convert.ToDouble(this.num\_sell.Text) <= 0))

{

this.sell\_butt.Enabled = false;

}

else

{

this.sell\_butt.Enabled = true;

}

}

if(this.skot\_radio.Checked == true) //если выбран скот для продажи

{

if ((Convert.ToDouble(this.num\_sell.Text) > Program.derevn.get\_kolvo\_skot()) || (Convert.ToDouble(this.num\_sell.Text) <= 0))

{

this.sell\_butt.Enabled = false;

}

else

{

this.sell\_butt.Enabled = true;

}

}

}

else

this.sell\_butt.Enabled = false;

}

private void skot\_radio\_CheckedChanged(object sender, EventArgs e)

{

num\_sell\_TextChanged(sender, e);

}

private void sell\_butt\_Click(object sender, EventArgs e)

{

if (this.hleb\_radio.Checked == true) //если выбран хлеб для продажи

{

Program.derevn.set\_kolvo\_hleb(Program.derevn.get\_kolvo\_hleb() - Convert.ToInt32(this.num\_sell.Text)); //вычесть выбранное кол-ва хлеба из ресурсов деревни

Program.derevn.set\_budget\_village(Program.derevn.get\_budget\_village() + Convert.ToInt32(this.num\_sell.Text) \* Program.derevn.get\_price\_hleb()); //начислить деньги за продажу

}

if (this.skot\_radio.Checked == true) //если выбран скот для продажи

{

Program.derevn.set\_kolvo\_skot(Program.derevn.get\_kolvo\_skot() - Convert.ToInt32(this.num\_sell.Text)); //вычесть выбранное кол-ва скота из ресурсов деревни

Program.derevn.set\_budget\_village(Program.derevn.get\_budget\_village() + Convert.ToInt32(this.num\_sell.Text) \* Program.derevn.get\_price\_skot()); //начислить деньги за продажу

}

num\_sell\_TextChanged(sender, e);

}

private void timer\_proverka\_Tick(object sender, EventArgs e)

{

if (Program.f\_endgame == 1) //игра закончилась, закрыть все формы

this.Close();

else

{

num\_sell\_TextChanged(sender, e); //повторная проверка

sell\_tovar\_Activated(sender, e); //обновить таблицу

}

}

}

}

itog\_game.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

namespace kurs2020\_sharp

{

public partial class itog\_game : Form

{

public itog\_game()

{

InitializeComponent();

}

private void close\_Click(object sender, EventArgs e)

{

this.Close();

}

private void itog\_game\_Load(object sender, EventArgs e)

{

this.your\_scores.Text = Convert.ToString(Program.derevn.get\_days\_procv());

}

private void name\_player\_TextChanged(object sender, EventArgs e)

{

if(this.name\_player.Text=="") //если поле с именем игрока пустое

{

this.ok\_record.Enabled=false;

}

else

{

this.ok\_record.Enabled=true;

}

}

private void ok\_record\_Click(object sender, EventArgs e)

{

string path= "records.txt"; //путь для записи рекордов

using (StreamWriter stream = new StreamWriter(path, true)) //добавление записи в файл

{

stream.WriteLine(this.name\_player.Text); //записать имя игрока в файл

stream.WriteLine(this.your\_scores.Text); //записать рекорд игрока в файл (реализовано так, чтобы рекорд нельзя было изменить, начав новую игру, не закрыв окно с финальным счётом)

stream.WriteLine("///////"); //записать разделитель в файл (отделяет рекорды друг от друга)

stream.Close();

}

this.Close();

}

}

}

record\_tabl.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

namespace kurs2020\_sharp

{

public partial class record\_tabl : Form

{

public record\_tabl()

{

InitializeComponent();

}

private void record\_tabl\_Load(object sender, EventArgs e)

{

string path= "records.txt"; //путь для чтения рекордов

if(System.IO.File.Exists("records.txt")) //если файл существует

{

int kolvo\_rec=0; //начальное значение кол-ва рекордов в файле

string strread="strread"; //строка для чтения из файла

using (StreamReader reader = new StreamReader(path)) //открытие файла для чтения

{

while(strread!=null) //пока не прочитает весь файл

{

strread=reader.ReadLine(); //получить строку из файла

strread=reader.ReadLine(); //получить строку из файла

strread=reader.ReadLine(); //получить строку из файла

kolvo\_rec+=1; //увеличить кол-во рекордов на 1

}

kolvo\_rec = kolvo\_rec -1 ; //уменьшить кол-во рекордов на 1 (в конце цикла while добавляется лишее)

reader.Close(); //закрыть файл

}

using (StreamReader reader = new StreamReader(path)) //открытие файла для чтения

{

for(int g=0; g<kolvo\_rec; g++)

{

spisok.Rows.Add(); //добавить новую строку в таблицу

spisok.Rows[g].Cells[0].Value = reader.ReadLine(); //внести в ячейку таблицы имя игрока

spisok.Rows[g].Cells[1].Value = reader.ReadLine(); //внести в ячейку таблицы рекорд игрока

strread=reader.ReadLine(); //получить строку из файла (разделитель рекордов "///////")

}

reader.Close(); //закрыть файл

}

//сортировка игроков по убыванию рекордов (метод пузырька)

for(int v=1; v<kolvo\_rec; v++)

{

for(int b=0; b<kolvo\_rec-v; b++)

if(Convert.ToDouble(spisok.Rows[b].Cells[1].Value) < Convert.ToDouble(spisok.Rows[b+1].Cells[1].Value))

{

Object vsp0 = spisok.Rows[b+1].Cells[0].Value;

Object vsp1 = spisok.Rows[b+1].Cells[1].Value;

spisok.Rows[b+1].Cells[0].Value = spisok.Rows[b].Cells[0].Value;

spisok.Rows[b+1].Cells[1].Value = spisok.Rows[b].Cells[1].Value;

spisok.Rows[b].Cells[0].Value = vsp0;

spisok.Rows[b].Cells[1].Value = vsp1;

}

}

}

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace kurs2020\_sharp

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

//Обявления переменных

public static Village derevn = new Village(); //объект, в котором хранится вся информация о деревне (глобальн.)

public static Village birzh\_trud = new Village(); //объект, в котором хранятся крестьяне для найма (глобальн.)

public static int f\_endgame = 0; //флаг о том, что игра окончена (игрок проиграл) (глобальн.)

public static int r1 = (int)(DateTime.Now.Subtract(new DateTime(1970, 1, 1)).TotalSeconds); //ключ рандома, полученный по текущему времени

public static Random randk = new Random(r1); //рандом

}

}

# Приложение В Руководство системного администратора

На переносном диске содержится архивированная папка с игрой. Для работы с данной программой необходимо разархивировать данную папку и запустить “kurs2020.exe”   
в реализации на C++ или “kurs2020\_sharp.exe” в реализации на C#. Программа готова к использованию.

Сообщите всем пользователям, что вмешиваться во внутренние файлы игры и изменять либо удалять их крайне не рекомендуется, так как легко может быть нарушена работоспособность игры.

Для функционирования программы требуется:

* Операционная система Windows XP SP3 или выше;
* Оперативная память 50МБ или выше;
* Жесткий диск: 12МБ для хранения программы и вспомогательных файлов, дополнительное место для хранения сохранения игры;
* .NET Framework версии 2.0 или выше;
* Наличие мыши, клавиатуры, монитора;
* Разрешение экрана 1024 x 768 или выше.

# Приложение Г Руководство пользователя

В игре «Управление деревней» вы берёте на себя роль управляющего деревни, и вам следует помочь крестьянам продержаться как можно дольше.

Для запуска игры необходимо дважды нажать на файл “kurs2020.exe” в реализации на C++ или “kurs2020\_sharp.exe” в реализации на C#. Откроется главное меню. В нём содержится несколько кнопок, позволяющий начать новую игру, загрузить сохранённую игру, открыть список с рекордами, открыть справку или же выйти из игры.

Условные обозначения:

• 1 Час Процветания = 3 секунды реального времени.

• 1 День Процветания = 30 секунд реального времени.

• Скорость жизни - это множитель, увеличивающий прожорливость крестьян и их денежные расходы.

• Прожорливость - это кол-во расходуемой пищи крестьянином Зимой.

Начиная новую игру, вы получаете в своё распоряжение деревню с небольшим количеством начальных ресурсов и тремя крестьянами.

Типы ресурсов в игре:

• Хлеб

• Домашний скот

• Деньги

Деньги хранятся в вашем бюджете. Они нужны для того, чтобы нанимать крестьян и оплачивать их труд. Хлеб и скот производятся крестьянами, а затем их можно продать, или же оставить на Зиму.

В игре существует два времени года: Лето и Зима.

Летом крестьяне каждый Час Процветания производят хлеб и скот в обмен на деньги. Производительность каждого крестьянина индивидуальна, как и его денежные расходы. Если денег нет, то крестьянин просто ничего не производит (уходить он не будет). Зимой крестьяне ничего не производят, но начинают съедать пищевые запасы. Их прожорливость также индивидуальна. Но Зимой они не тратят деньги.

Крестьян можно нанимать на бирже труда, а также изгонять из деревни. Изгонять и нанимать можно только Летом. Продавать хлеб и скот можно на товарной бирже в любое время года.

Лето длится 4 Дня Процветания, или же 40 Часов Процветания, или же 2 реальные минуты.

Зима длится 2 Дня Процветания, или же 20 Часов Процветания, или же 1 реальную минуту.

В конце каждой Зимы Скорость Жизни увеличивается на 1. Следовательно, играть становится сложнее.

Игра заканчивается, когда из деревни уходят все крестьяне и проходит 2 Часа Процветания (6 реальных секунд). Финальным счётом являются набранные за игру Дни Процветания, которые вносятся в список рекордов вместе с именем игрока.

Зимой, если крестьянину не хватило пищи, он начинает отсчёт. Крестьянин уходит из деревни через 5 Часов Процветания (15 реальных секунд), если поесть ему так и не удалось.

В основном окне игры отображается текущее значение показателей деревни: значение Скорости Жизни, кол-ва Дней Процветания, кол-ва крестьян, а также кол-во ресурсов (деньги, хлеб и скот). Внизу отображается информация от Помощника: время до конца текущего сезона, а также последнее произошедшее событие. Фон главного окна игры показывает текущее время года: летнее или зимнее.

Справа есть кнопки для открытия списка крестьян и товарной биржи. Из списка крестьян можно попасть на биржу труда.

В верхнем меню есть кнопки сохранения игры, справки, а также выхода в главное меню. Игра не сохраняется автоматически, игрок сохраняет игру самостоятельно, когда ему нужно.

Список крестьян позволяет узнать характеристики ваших крестьян. Поле для ввода номера крестьянина позволяет его изгнать. Не рекомендуется изгонять всех сразу, иначе наступит конец игры. Из этого же окна можно открыть биржу труда с помощью соответствующей кнопки.

Биржа труда содержит список из пяти случайных крестьян, которых можно нанять. При каждом заходе на биржу труда таблица обновляется пятью новыми случайными крестьянами. Есть поле для ввода номера крестьянина, которого вы желаете нанять. Стоимость найма крестьянина указана в последнем столбике. Если у вас недостаточно денег для найма, кнопка подтверждения будет неактивной.

Товарная биржа, в которую можно попасть из главного окна игры, позволяет продать ваши ресурсы. Покупать ресурсы нельзя. В окне биржи отображаются текущие цены на хлеб и скот, а также есть поле для ввода желаемого кол-ва на продажу. Вид ресурсов (хлеб или скот) выбирается с помощью радиокнопок правее поля ввода. Кнопка продажи неактивна, если введённое число больше, чем у вас есть в наличии.

В середине Зимы происходит случайное событие: к вам в деревню бесплатно приходит новый крестьянин, или же волки нападают на домашний скот.

Цены на товарной бирже случайно немного изменяются в конце каждой Зимы.