РЕФЕРАТ

Пояснювальна записка: ___ с., ___ рис., ___ табл., __ дод., ___ джерел.

Об'єкт розробки: навчальна комп’ютерна гра.

Мета кваліфікаційної роботи: розробка навчальної комп’ютерної гри «Український тетрис» для полегшення в ігровій та цікавій формі вивчення сучасного українського правопису.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування програми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження програми, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення роботи полягає в створенні розвивально – ігрової програми для учнів школи з метою закріплення вивченого сучасного правопису української мови та яка носить характер цікавих тренувальних вправ.

Актуальність теми кваліфікаційної роботи визначається великим попитом на подібні розробки, через те, що сьогодні у навчальних закладах з метою активізації пізнавальної діяльності учня потрібно використовувати усі можливі мультимедійні технології для проведення уроків, практичних і самостійних занять.

Список ключових слів: НАВЧАЛЬНА ГРА, ТЕТРІС, УКРАЇНСЬКА МОВА, ПРОГРАМА, АЛГОРИТМ, НАЛАГОДЖЕННЯ, ТЕСТУВАННЯ, ПРОЄКТУВАННЯ, ОРФОГРАФІЯ.

ABSTRACT

Explanatory note: ___ pp., ___ fig., ___ table, __ appendix, ___ sources.

Object of development: educational computer game.

The purpose of the qualification work: development of an educational computer game "Ukrainian Tetris" to facilitate the study of modern Ukrainian spelling in a playful and interesting way.

The introduction considers the analysis and current state of the problem, specifies the purpose of the qualification work and its scope, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development are defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes the existing solutions, selects the platform for development, performs design and development of the program, describes the algorithm and structure of the program, determines the input and output data, provides characteristics of the parameters of hardware, describes the call and download program, describes the program .

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical significance of the work is to create a developmental - game program for school students in order to consolidate the studied modern spelling of the Ukrainian language and which has the character of interesting training exercises.

The relevance of the topic of qualification work is determined by the great demand for such developments, due to the fact that today in educational institutions in order to enhance the cognitive activity of the student must use all possible multimedia technologies for lessons, practical and independent classes.

List of keywords: EDUCATIONAL GAME, TETRIS, UKRAINIAN LANGUAGE, PROGRAM, ALGORITHM, ADJUSTMENT, TESTING, DESIGN, SPELLING.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ІС – інформаційна система;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер.

ЗМІСТ

ВСТУП

Новітні інформаційні технології стрімко ввійшли в усі сфери нашого життя, не винятком стала й освіта. Ведуться дискусії з приводу ефективності та доцільності використання інформаційних технологій на уроках чи заняттях у початковій та середній школі, але не використовувати їх було б безглуздям. Можливості сучасного уроку й системи освіти значно розширюються завдяки використанню мультимедійних, інтерактивних технологій, Інтернету. Сьогодні перед педагогами стоїть важливе завдання – виховати та підготувати молодь, спроможну активно включатися в якісно новий етап розвитку сучасного суспільства, пов'язану з інформацією. Ні для кого не є новиною, що дитина опановує комп'ютер раніше, ніж навчається грамотно писати та читати. Згідно з сучасною концепцією навчання дедалі більше уваги приділяється оптимізації та індивідуалізації шкільної освіти.

В Україні використання інформаційних технологій у навчальному процесі, окрім уроків інформатики, почалося декілька років тому, особливий розвиток даний напрямок отримав в умовах введення карантинних заходів на території нашої країни та перехід до дистанційної форми навчання. До цього часу мало хто використовував інформацію з Інтернету з дидактичною метою. Але за останні роки наше суспільство було вимушено зробити крок уперед.

Комп'ютери стрімко увійшли в різноманітні сфери нашої повсякденної діяльності, тому широке запровадження комп'ютерної техніки в процесі навчання є важливим завданням.

Необхідним є осмислення критеріїв доцільності використання мультимедійних засобів в процесі навчання:

– підвищується ефективність навчання більше, ніж при використанні традиційних засобів навчання;

– неможливість реалізації певних засобів навчання у вигляді матеріальних об'єктів (оригінали у природних або штучних умовах);

– недостатня наочність та зрозумілість або надлишкова складність відповідних вербально – знакових, графічних (статичних або динамічних), знакових, логічно математичних моделей.

Сьогодні у навчальних закладах з метою активізації пізнавальної діяльності учня потрібно використовувати усі можливі мультимедійні технології для проведення уроків, практичних і самостійних занять. Найбільш ефективними є такі мультимедійні продукти, як: освітні програми разом з іграми або освітні програми (інтерактивні, ігрові, розважальні), мета яких – викликати інтерес і бажання пізнавати більше, підвищити якість засвоєння нового матеріалу.

Актуальність даної роботи полягає у необхідності створення розвивально – ігрової програми для учнів школи з метою закріплення вивченого сучасного правопису української мови та носить характер тренувальних вправ. Дана робота направлена на розробку навчальної гри для легкого засвоєння правил українського правопису.

Метою кваліфікаційної роботи є облегшити в ігровій та цікавій формі вивчення українського правопису для школярів. Для досягнення мети під час виконання роботи вирішені наступні завдання:

– ознайомлення з теоретичними даними правил українського правопису та про сучасні зміни в ньому;

– розробка правил та концепції гри на основі відомої гри «Тетрис»;

– розробка алгоритму гри з реалізацією перевірки відповідей гравця;

– програмна реалізація візуальної частини гри;

В програмі реалізована гра «Тетрис», елементами якої є слова української мови з пропущеними літерами та випадаючі кубики з літерами, які потрібно вставити у відповідні пропущені для них місця. Гра має зручний привабливий інтерфейс. Є можливість скористатися довідковим матеріалом, тимчасово припинити гру і повернутися до збереженої гри, переглянути рейтинг гравців.

Робота носить прикладний характер. Програма може бути використана на уроках української мови та в позаурочних заняттях.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

1.1.1. Передумови створення нового українського правопису

Мова - запорука тривкої ідентичності нації, основа її етнокультурної цілісності. Правопис - це еталон писемної літературної мови, яка, за слушним спостереженням Юрія Шевельова, є штучним витвором високорозвиненого суспільства, а не відтворенням почутого «з уст народу» [19]. Правопис складається з трьох підсистем: графіки (букв, якими позначають найтиповіші звуки), орфографії (закономірностей поєднання букв для передавання на письмі звукового образу української мови) і пунктуації (розділових знаків, за допомогою яких позначають змістове й інтонаційне членування висловленої думки). Кожна з цих підсистем, як і кожний з розділів правопису, мають свою історію.

У 1918 році опубліковано проєкт офіційного українського правописного кодексу, запропонований професором Іваном Огієнком, у доопрацюванні якого згодом узяли участь академік Агатангел Кримський і професор Євген Тимченко. У 1919 році цей проєкт був виданий під назвою «Головніші правила українського правопису». Цього ж року спільне зібрання Української академії наук схвалило «Найголовніші правила українського правопису» - перший в історії України офіційний загальнодержавний правописний кодекс. В умовах складних геополітичних змін, війн і соціальних катаклізмів він справляв визначальний вплив на формування української мови як національного, а не етнологічного феномену. Його творці орієнтувалися, по-перше, на специфіку історичної й діалектної основи української мови, на її характерні ознаки порівняно з іншими слов’янськими мовами, і, по-друге, на мовну практику визначних українських письменників і перекладачів. У 1925 році уряд УСРР створив при Народному комісаріаті освіти Державну комісію для

впорядкування українського правопису, а в 1927 році за наслідками роботи Всеукраїнської правописної конференції, що відбулася в Харкові, комісія підготувала проєкт всеукраїнського правопису, в опрацюванні якого взяли участь мовознавці з Наддніпрянщини й західноукраїнських земель. Від імені держави 6 вересня 1928 року цей Український правопис, який отримав неофіційну назву «харківський», затвердив нарком освіти УСРР Микола Скрипник.

У 1989 році затверджено і в 1990 році опубліковано нову редакцію Українського правопису, у якій поновлено букву ґ, уточнено й доповнено окремі правописні норми.

У 2015–2018 роках Українська національна комісія з питань правопису, до якої увійшли фахівці мовознавчих установ Національної академії наук України та представники закладів вищої освіти з різних регіонів України, розробила проєкт нової редакції Українського правопису. Після громадського обговорення його схвалено на спільному засіданні Президії Національної академії наук України та Колегії Міністерства освіти і науки України 24 жовтня 2018 року.

Сучасна редакція Українського правопису повертає до життя деякі особливості правопису 1928 року, які є частиною української орфографічної традиції і поновлення яких має сучасне наукове підґрунтя.

Нинішня українська мова - це багатофункціональна мова з розвиненою різноплановою стилістикою, сучасною науковою термінологією, це мова, яка взаємодіє з багатьма світовими мовами. У сучасній редакції Українського правопису збережено підхід до мови як до знакової системи й суспільного явища. Правописна норма, з одного боку, ґрунтується на тісних зв’язках елементів сучасної мовної системи, що відображається на письмі, а з другого боку, - на дотриманні мовної традиції і на залежності від неї. Здебільшого традиція відображає вчорашні або позавчорашні параметри системи. Так само, як і система, традиція є закономірним регулятором мовної норми.

Сучасна українська мова - відкрита й динамічна. Змінюється словник, з'являються нові терміни в різних сферах суспільної комунікації, виникає потреба адаптації загальних і власних назв до раніше сформульованих орфографічних правил. Для чого переглядати ці правила? Для того щоб відреагувати на зміни в сучасній мовно-писемній практиці, визначити правила написання нових запозичених слів, нових власних назв, усунути застарілі формулювання та спростити й, де це можливо, уніфікувати орфографічні норми.

Хоч уточнення й коригування орфографічних правил - це завдання насамперед професійних мовознавців, співтворцями Українського правопису є також широке коло освічених українців, чия писемна практика узвичаює те чи те написання. Кожна мова засвоює іншомовні елементи, і вони здебільшого відрізняються від питомих елементів за своїми системними ознаками. В історії правопису написання слів іншомовного походження було й залишається предметом гострих дискусій. Ці слова значною мірою зберігають свій іншомовний колорит.

Намагання передати звукову форму запозичуваних слів або особливості їхнього оригінального написання веде до проникнення в українську мову нових звуків і їхніх поєднань, екзотичних граматичних явищ. Водночас, стаючи частиною української мовної системи, ці слова поступово адаптуються до української фонетики морфології, орфографії. При цьому вони не завжди вкладаються в традиційні українські парадигми й нерідко потрапляють до категорії невідмінюваних: ательє, кафе, буржуа, какао, мадам, радіо, таксі, ханум і т. ін. Одні запозичення приходять в українську мову писемним, інші - усним шляхом. У мові-джерелі звучання слова часто не збігається з прочитанням сукупності букв, з яких воно складається (Carlisle - Карлайл, Rambouillet - Рамбує, Worcester - Вустер, billet - білет, pioneer - піонер і т. ін.). В українській, так само, як і в багатьох інших мовах, і транскрибування, яке передає звукову форму іншомовного слова, і транслітерування, яке відтворює його графічну форму, і комбіноване застосування цих підходів, і орфографічна

адаптація слова були й залишаються природними й легітимними способами поповнення національного лексикону.

Реагуючи на виклики мовної практики, сучасна редакція правопису розширила межі використання орфографічних варіантів. Кожний історичний період розвитку мови має свою варіантну динаміку. Пропонуючи в новій редакції правопису низку орфографічних варіантів, кодифікатори виходили з того, що варіативність - це органічна частина правописного кодексу і тією чи іншою мірою вона притаманна кожній мові на різних етапах її історичного розвитку. Відповідь на те, який з варіантів залишиться в минулому, зможе дати лише майбутнє. Спадкоємність у мові - це зв'язок між поколіннями, які жили, живуть і житимуть в Україні. Пошук балансу між системними параметрами сучасної мови, з одного боку, й різночасовими прикметами української мовної традиції, з другого боку, - найскладніше із завдань, що поставали перед творцями національного правопису на кожному з етапів його розвитку. Нова редакція правопису є кроком до розв'язання цього завдання з позицій історичної й етнографічної соборності української мови й української нації .

1.1.2. Найбільші новації Українського правопису 2019 року

22 травня 2019 Кабінетом Міністрів України було схвалено постанову № 437 «Питання українського правопису», якою вводиться в дію нова редакція «Українського правопису» [17]. Багато нововведень допускають варіантне написання, що потенційно відкриває широкі можливості для створення в умовах набуття чинності нового правопису стеганографічних алгоритмів.

Наведемо кілька основних найбільших новацій Українського правопису 2019 року:

1. Пишемо архимандрит, архиєрей, архиєпископ, архиєрей тощо через И – поряд із попередніми архімандрит, архієрей, архієпископ, архієрей.

2. Пишемо також гідности, незалежности, радости, смерти, чести, хоробрости; крови, любови, осени, соли, Руси, Білоруси – як варіант, із -И в кінці в родовому відмінку однини; також лишається дотеперішня норма на -І.

3. «Топ-100», «топ-десять» – «поза законом».

4. Назви сайтів та інших інтернет-сервісів пишемо тільки українською, з обов'язковим відмінюванням: твітер, ґуґл; мережа «Фейсбук», енциклопедія «Вікіпедія»; фейсбука, ютуба, імейла (із закінченням -А, -Я в родовому відмінку однини).

5. Слова «ви», «ваш» тощо завжди пишемо з малої літери, якщо тільки це не лист із персональним зверненням до однієї особи з виявом особливої ввічливості.

6. Пишемо тільки разом: віцепрем'єр, віцеконсул, ексчемпіонка, ексміністр, екпрезидент, контрадмірал, вебсайт, вебсторінка, преміумклас, максісукня, мідімода, мініспідниця, топменеджер, топмодель, лейбгвардієць, лейбмедик, оберофіцер, оберлейтенант, оберпрокурор, штабскапітан, унтерофіцер тощо.

7. Пишемо тільки окремо пів у значенні половина: пів аркуша, пів відра, пів години, пів літра, пів міста, пів огірка, пів острова, пів яблука, пів ящика, пів ями; пів Європи, пів Києва, пів України.

8. Пишемо тільки двоактний, двоокис, двооксид, двоопуклий, двоосьовий, триатомний, триокисень, чотириосьовий – а не «двох-», «трьох-», «чотирьох-», як раніше, перед наступним А чи О).

9. Пишемо інакше слова: священник (через два Н); проєкт, проєкція (через Є); госпіс (через Г).

10. Пишемо також через Ґ: Верґілій, Ґарсія, Геґель, Ґеорґ, Ґете, Ґреґуар, Ґуллівер тощо, поряд із попередніми формами через Г.

11. Пишемо також через Т: анатема, дитирамб, етер, катедра, міт, мітологія, Атени, Демостен та ін., поряд із формами через Ф (дифірамб, ефір, міф, Афіни, Демосфен).

12. Пишемо також через АВ: авдієнція, авдиторія, лавреат, павза, фавна тощо, поряд із формами через АУ (пауза, фауна).

1.1.3. Створення гри

В інтернеті можна знайти багато різних програм, за допомогою яких можна створити комп’ютерну гру, як для новачка так і для професіонала.

Ці програми дають змогу створити різноманітні ігрові середовища як з використанням програмування так і без нього.

При створенні навіть простої гри, необхіно використовувати не одну, а декілька програм. Це графічні редактори, програми комп’ютерного моделювання ігрового середовища, а також програмні середовища для створення програм керування ігровими героями.

Крім того, кожен етап процесу створення комп'ютерної гри – надзвичайно ресурсоємний процес, тому створенням ігор зазвичай займаються не індивідуальні розробники, а команди, починаючи з невеликих компаній і закінчуючи великими студіями (наприклад, Blizzard, Electronic Arts, Bethesda, Ubisoft).

Комп'ютерні ігри за технологією створення можна поділити на двовимірні та тривимірні. Двовимірні комп'ютерні ігри базуються на використанні двовимірних спрайтів (графічних файлів невеликого розміру), тому вони є більш простими у розробці й менш вимогливими до апаратного забезпечення, у той час як тривимірні ігри, через великий обсяг інформації для зберігання (текстури, багатополігональні сцени) та обчислення (прорахунок фізики освітлення та спецефектів, емуляції фізичних процесів тощо) навантажують всі вузли комп'ютера: центральний процесор, відеоакселератор, оперативну пам'ять, накопичувач. Крім того, створення елементів (ассетів) тривимірної гри вимагає від розробників набагато більше зусиль та витраченого часу.

Раніше професійні комп'ютерні ігри могли створювати лише великі та середні ігрові студії, через те, що «серце» організму комп'ютерної гри, ігрові движки, були власністю компаній, які ретельно, ніби військову таємницю, оберігали їх від будь-кого, хто не належав до кола ведучих розробників компаній.

Близько десяти років тому ситуація змінилася. Компанії CryTek та Epic Games розпочали надавати свої ігрові движки (CryEngine та Unreal Engine) у якості комерційного продукту в оренду спочатку іншим компаніям, а потім й приватним особам. Згодом на ринку з'явилася компанія Unity, яка розробила власний ігровий движок спеціально для комерційного розповсюдження. Відтоді, навіть невеликі розробники отримали нагоду створювати власні комп'ютерні ігри професійного рівня.

На цьому корпорації – розробники ігрових движків не зупинилися, і близько 5 років тому компанія Epic Games вирішила надати безкоштовну версію свого програмного продукту будь-кому, навіть дозволила використовувати його у комерційних цілях за умови, що сумарна вигода того, хто користується Unreal Engine для створення платних комп'ютерних ігор, не перевищує 3000 американських доларів на рік. Отже, тепер навіть аматори та ентузіасти можуть створювати власні комп'ютерні ігри високого ґатунку. Незабаром до Epic Games приєдналися Unity та CryTek.

Як було зазначено раніше, окрім ігрового движка для створення комп'ютерної ігри використовують багато різних програмних продуктів для створення елементів гри (ігрових ассетів), серед яких є як комерційні, так і повністю безкоштовні програми.

Тривимірна комп'ютерна гра – це поєднання тривимірних комп'ютерних моделей, графічних текстур, прорахунку світла (джерел освітлення) та різноманітних фізичних процесів (наприклад, руйнація будівлі) та ігрової логіки (скриптів, реалізованих однією з мов програмування).

Проблема комплексного створення повноцінної комп’ютерної гри однією людиною полягає у необхідності опанування великою кількістю програмних продуктів, які у свою чергу складаються із кількох підрозділів, кожен з яких потребує великої кількості часу для запам’ятовування функціональних особливостей та інструментів.

Розробка концепту гри – це період обмірковування, створення чернеток, планів та алгоритмів. Під час цього етапу доцільно використовувати програмне забезпечення для схематичних зображень та скетчів (FreePlane, Krita). Цей етап ставить метою створення «скелету», загального плану подальшої розробки, якого доцільно дотримуватись.

Після того як визначені вимоги до програми та складено алгоритм рішення, алгоритм записується на обраною мовою програмування. В результаті виходить вихідна програма. Складання тесту програми, напевно, найскладніший з етапів, що вимагає найбільшої уваги. Щоб цей текст був зрозумілий користувачеві й упорядника, використовуються коментарі.

Моделювання – створення об'єктів у просторі із графічних примітивів. Поєднуючи такі примітиви один з одним, розробник утворює поверхні, які виглядають як реальні або уявні об'єкти.

Наступним етапом є текстурування об’єктів. Розробник ігор створює графічні зображення, які накладаються на об'єкти та додають їх поверхні реалістичного вигляду. За необхідності об'єкти анімуються.

Останнім етапом є перевірка гри на логічні, графічні та інші помилки, по завершені цього етапу гру можна вважати готовою до випуску.

Налагодження - це процес виявлення і усунення несправностей. Помилки в програмі поділяють на дві групи: синтаксичні (помилки в тексті) і алгоритмічні. Синтаксичні помилки - найбільш легко усуваються. Алгоритмічні помилки виявити важче. Етап налагодження можна вважати закінченим, якщо програма правильно працює на одному-двох наборах вхідних даних.

1.1.4. Класична гра «Тетрис»

Тетрис - відеогра-головоломка, розроблена Олексієм Пажитновим та його колегами. Перша версія гри була представлена 6 червня 1984 року. Назву гри автор створив поєднавши грецький префікс «тетра-» зі словом «теніс».

Ця гра в тому чи іншому вигляді існує майже для кожної ігрової консолі та операційної системи, а також для інших електронних пристроїв: мобільних телефонів, портативних медіаплеєрів, кишенькових комп'ютерів тощо.

Правила гри: випадкові фігурки тетраміно падають зверху в прямокутний стакан шириною 10 і висотою 20 клітин. У польоті гравець може повертати фігурку та рухати її по горизонталі. Також можна «скидати» фігурку, тобто прискорювати її падіння, коли вже вирішено, куди фігурка повинна впасти. Фігурка летить, поки не наткнеться на іншу фігурку або на дно склянки. Якщо при цьому заповнився горизонтальний ряд з 10 кліток, він пропадає і все, що вище нього, опускається на одну клітку. У спеціальному полі гравець бачить фігурку, яка буде слідувати після поточної - ця підказка дозволяє планувати свої дії. Темп гри поступово збільшується.

Назва гри походить від кількості клітин, з яких складається кожна фігура. Гра закінчується, коли нова фігурка не може поміститися в стакан. Гравець отримує бали за кожну фігурку, тому його задача - заповнювати ряди, не заповнюючи саму склянку якомога довше, щоб таким чином отримати якомога більше балів.

Гра реалізована практично на всіх сучасних комп'ютерах, включаючи ПК, мобільні телефони, ігрові відеоприставки, телевізори (як додаткова функція), безліч кишенькових ігрових пристроїв. Є варіанти гри для всіх поширених ОС, а також для Java. Мабуть, найбільшої популярності набула реалізація тетрісу для ігрової консолі Game Boy і відеоприставки NES (і її численних клонів).

У багатьох реалізаціях стакан спочатку не порожній. Є реалізації (наприклад, безкоштовна Gravytris для Microsoft Windows) з більш реалістичними правилами гравітації: наприклад, при пропажі горизонтального

ряду блоки, які вище його, з'єднуються в зв'язкові області та кожна область падає, поки не наткнеться на блок, це може привести до заповнення нових лав і новим падінь, і так далі.

Були написані трьох- (Blockout фірми California Games, 1989), чотирьох- (HyperTetris, 1996) і навіть n-мірні (Polytope Tetris, 2003) варіанти тетрісу, а також модифікації для двох і більше гравців.

Існують також версії гри, в яких гра ведеться не так на бали, а на відкриття захованого зображення. Для полегшення гри є варіанти без складних S-і Z-образних фігур і без збільшення швидкості гри.

1.2. Призначення розробки та галузь застосування

У наш час різноманітність комп’ютерних ігор на світовому ринку просто вражає. Навіть з’явилась нова хвороба – ігроманія, що поглинає тисячі підлітків. Та комп’ютерна гра може стати справжнім помічником у вивченні нового матеріалу, проведенні лабораторних робіт, або ж в закріпленні вже набутих знань. Таких посібників багато з математики, хімії, фізики, літератури, історії та інших предметів. Вчені давно довели, що діти найкраще сприймає інформацію у вигляді гри, або з ігровими елементами.

На сучасному ринку комп’ютерних ігор пізнавальних ігор з української мови майже немає, тому і було вирішено створити програму, що допоможе учням покращити свої знання української орфографії.

Ця проблема стала дуже актуальною в останній час. Виникає одразу питання – чому подібних програм з англійської мови та інших предметів досить багато, а саме з української – одиниці, тому і виникла ідея створити комп’ютерну гру, яка б могла зацікавити молоде покоління і, можливо, викликати бажання відкрити словник і подивитись правильне написання того, чи іншого слова.

В роботі реалізована гра «Тетрис», елементами якої є слова української мови з пропущеними літерами та випадаючі кубики з літерами, які потрібно вставити у відповідні пропущені для них місця.

Ше більшої актуальності ця гра набула у зв'язку з публікацією нового українського правопису. Використання даної гри для як під час навчання, так і під час підготовки до ЗНО, допомагає легко перевіряти знання, засвоювати нові правила у приємній для молоді ігровій формі без папірців та книжок.

Програма має розважальний і повчальний характер. Вона створена для того, щоб допомогти учням покращити знання української орфографії.

1.3. Підстава для розробки

Підставою для розробки кваліфікаційної роботи на тему «Розробка навчальної комп'ютерної гри «Український тетрис» засобами мови C# » є наказ по Національному технічному університету «Дніпровська політехніка» від __.__. 2021р. № ____- .

1.4. Постановка завдання

Завданням проекту є: розробити навчальну комп'ютерну гру «Український тетрис» засобами мови C#.

Метою кваліфікаційної роботи є облегшити в ігровій та цікавій формі вивчення українського правопису для школярів.

Для досягнення мети під час виконання роботи потрібно вирішити наступні завдання:

– ознайомитись з теоретичними відомостями правил українського правопису та про сучасні зміни в ньому;

– розробити правила та концепцію гри на основі відомої гри «Тетрис»;

– розробити алгоритм гри з реалізацією перевірки відповідей гравця;

– програмно реалізувати візуальну частину гри;

– протестувати розроблену програму та виправити можливі недоліки.

Робота присвячена аналізу алгоритму популярної гри «Тетрис» та створенню власного варіанту гри «Тетрис», призначеного для допомоги у засвоєнні українського правопису.

Було поставлено задачу створити комп’ютерну гру з наступними характеристиками:

– гра повинна бути цікавою та пізнавальною для користувачів;

– гра повинна мати привабливий інтерфейс;

– гра повинна розвивати розумові здібності і тренувати швидкість розумових процесів та застосовувати набуті знання.

Правила гри: на основній формі повинно знаходитися ігрове поле (стакан 13х15), на якому з’являються нові слова. Одна буква прихована від гравця. Зверху починають рухатись одна за одною букви. Якщо гравець доповнить слово потрібною буквою, то строчка зникне. Мета гри - зібрати якомога більше слів та балів. Символ бомби означає, що потрапивши на букву він знищить її. Символ « _» означає, що слово потрібно з ’єднати.

Програма повинна забезпечувати можливість додавати до гри нові слова, змінювати швидкість падіння букв та зберігати результат .

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

«Український тетрис» - це перш за все гра для учнів. Тому, при розробці інтерфейсу користувача необхідно врахувати ряд особливостей:

1. Інтерфейс повинен бути інтуїтивно зрозумілий простому користувачеві комп'ютера.
2. Колірна схема не повинна дратувати людини.

Програма призначена для корисного та приємного проведення часу. Вона не повинна мати сильно складну систему управління. Кожному пункту меню та

команді має бути присвоєно назву, яка відповідає функціональному призначенню. Це полегшить роботу користувача. До того ж користувачу потрібно надати можливість ознайомитися з правилами гри для розуміння її принципів та умовами керування грою.

Кольори інтерфейсу не повинні нести жодної психічної навантаження, щоб проведення часу за грою для людини було приємним.

1.5.2. Вимоги до інформаційної безпеки

Основні вимоги до інформаційної безпеки:

- цілісність даних;
- захист від неавторизованого редагування;
- доступність інформації для всіх користувачів.

1.5.3. Вимоги до складу та параметрів технічних засобів

Вимоги до параметрів комп’ютера:

- підтримка 64-розрядного процесору та операційної системи;
- процесор Intel Core i5 3470 3.2 GHz/AMD X8 FX-8350 4 GHz;
- 5 мегабайт місця на вінчестері;
- 64 мегабайти оперативної пам’яті;
- розподільна спроможність монітора – 1024*768 пікселів та більше;

1.5.4. Вимоги до інформаційної та програмної сумісності

Дана гра повинна бути реалізована засобами мови C# та працювати як під Windows.9X, так і більш пізніші версії.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Метою кваліфікаційної роботи є облегшити в ігровій та цікавій формі вивчення українського правопису для школярів.

В роботі реалізована гра «Тетрис», елементами якої є слова української мови з пропущеними літерами та випадаючі кубики з літерами, які потрібно вставити у відповідні пропущені для них місця.

Програма забезпечує можливість додавати до гри нові слова, змінювати швидкість падіння букв та зберігати результат .

2.2. Опис застосованих математичних методів

У створенні алгоритму використовувався масив як математичний аналог поля "Тетрис".

Кожний осередок масиву відповідає певній області поля гри. Кожна область поля гри може бути заповнена фігурою або бути порожньою. Відповідно, кожна область поля може приймати два значення. Для цих цілей можна використовувати логічні змінні.

Кожна фігура (слово) має певну кількість букв (осередків) і займає кілька областей поля гри. Отже в масиві, осередки, відповідні заповненим областям поля, матимуть логічне значення true. Для осередків відповідних порожнім областям буде присвоєно логічне значення false.

Кожна горизонталь поля тетрису відповідає рядку двовимірного масиву, а вертикаль - стовбцю.

Рух фігур проводиться через рівні проміжки часу, тобто відбувається повторення алгоритму через рівні проміжки часу. Рівні проміжки часу можна забезпечити за допомогою елементу мови C # таймеру. Кожен тик таймера

буде відбуватися повторення алгоритму руху фігури. Це представлено як послідовна зміна значень в осередках масиву.

Щоб гра була працездатною, необхідно кожному осередку масиву кожен тик таймера присвоювати значення, які відповідають областям поля. Оскільки кількість осередків велике (їх 13x15=195), зручно було використовувати циклічні конструкції. Це дозволило скоротити програмний код і кількість помилок в ньому.

Короткостроковою метою гри «Український тетрис» є повне заповнення буквами слова для подальшого його знищення та очищення на ігровому полі та отриманні очок. Для цього необхідно було внести в програму функцію перевірки заповнення слова. Для масиву це представляється як перевірка осередків кожного рядка на однаковість значень.

Математична частина алгоритму будується на мінімальному числі елементів: двовимірний масив, таймер, логічна змінна, циклічна конструкція і умовна конструкція.

2.3. Опис використаних технологій та мов програмування

Запропонована програма написана на об'єктно-орієнтованій мові програмування C# в середовищі програмування розробки Microsoft Visual Studio 2017 Community.

Мова програмування C# широко використовується для розробки додатків під Windows. Мова має низку переваг, серед яких є:

- можливість розширення системи (в C# можна спокійно довантажувати будь які програми розширення .exe, імпортувати класи і об’єкти з інших програм;
- доступність мови і супроводу (читаність коду, документованість мови);
- ступінь відкритості вихідних текстів бібліотек, виконуваних програм;
- швидкість роботи, зручність розробки, трудомісткість написання.

C# – проста, сучасна об'єктно-орієнтована і типобезпечна мова програмування. C# відноситься до широко відомого сімейства мов C.

C# є об'єктно-орієнтованою мовою, але підтримує також і компонентно-орієнтоване програмування. Розробка сучасних додатків все більше тяжіє до створення програмних компонентів у формі автономних і самописних пакетів, що реалізують окремі функціональні можливості. Важлива особливість таких компонентів - це модель програмування на основі властивостей, методів і подій. Кожен компонент має атрибути, які надають декларативні відомості про компоненті, а також вбудовані елементи документації. Таким чином C# відмінно підходить для створення і застосування програмних компонентів.

Розглянемо далі функції мови C#, що забезпечують надійність і стійкість додатків: прибирання сміття автоматично звільняє пам'ять, зайняту знищеними і невикористовуваними об'єктами; обробка виключень дає структурований і розширюваний спосіб виявляти і обробляти помилки; сувора типізація мови не дозволяє звертатися до неініціалізованих змінних, виходити за межі масиву або виконувати неконтрольоване приведення типів.

У C# існує єдина система типів. Всі типи C#, включаючи типи-примітиви, такі як int і double, успадковують від одного кореневого типу object. Таким чином, всі типи використовують загальний набір операцій, і значення будь-якого типу можна зберігати, передавати і обробляти схожим чином. Крім того, C# підтримує призначені для користувача посилальні типи і типи значень, дозволяючи як динамічно виділяти пам'ять для об'єктів, так і зберігати спрощені структури в стеці.

Щоб забезпечити сумісність програм і бібліотек C# при подальшому розвитку, при розробці C# багато уваги було приділено управлінню версіями. Багато мов програмування обходять увагою це питання, і в результаті програми на цих мовах виходять з ладу частіше, ніж хотілося б, при виході нових версій залежних бібліотек. Питання управління версіями істотно вплинули на такі аспекти розробки C#, як роздільні модифікатори virtual і override, правила

вирішення перевантаження методів і підтримка явного оголошення членів інтерфейсу[16].

Microsoft Visual Studio – лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів. Дані продукти дозволяють розробляти, як консольні додатки, так і додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби, як в рідному, так і в керованому кодах для всіх платформ, підтримуваних Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework і Silverlight.

Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторінга коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і відладчик машинного рівня. Решта інструментів включають в себе редактор форм для спрощення створення графічного інтерфейсу додатку, веб-редактор, дизайнер класів і дизайнер схеми бази даних.

Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (як, наприклад, Subversion і Visual SourceSafe), додавання нових наборів інструментів (наприклад, для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування) або інструментів для інших аспектів процесу розробки програмного забезпечення (наприклад, клієнт Team Explorer для роботи з Team Foundation Server).

У Visual Studio застосовуються різні технології Для наочності вони представлені нижче :

– .NET Framework;
– .NET Framework 3.5;
– .NET Framework 3.0;
– .NET Compact Framework.

.NETFramework – невід'ємне компоненти Windows, які підтримують побудову і розробку додатків нового покоління.

– Windows Presentation Foundation (WPF);
– Windows Communication Foundation (WCF);
– Windows Workflow Foundation;
– Silverlight;
– Windows Forms;
– ASP.NET (AJAX);
– LINQ.

Також у Visual Studio існує можливість програмування наступними мовами програмування:

– Visual Basic;
– Visual C #;
– Visual C ++;
– JScript.

Розробка велася в інтегрованому середовищі розробки Microsoft Visual Studio 2017 Community. Переваги даної IDE:

— безкоштовне середовище для розробки як ПЗ, так і веб-додатків на майже будь-якій мові;
— повна підтримка мови C#;
— візуальний конструктор форм та візуальне представлення властивостей елементів форми;
— гнучкі інструменти управління проектом;
— зручний редактор файлу ресурсів;
— підтримка сторонніх модулів;
— велика кількість зручних комбінацій клавіш для збільшення швидкості розробки;
— вбудований відладчик коду;
— підсвічування синтаксису та інструменти для рефакторингу коду;
— наявність IntelliSence.

2.4. Опис структури системи та алгоритмів її функціонування

2.4.1. Розробка правил та концепції гри

Гра «Український тетрис» реалізована на основі відомої гри «Тетрис».

Правила гри: на основній формі знаходиться ігрове поле (прямокутний стакан 13x15), на якому з’являються нові слова українською мовою. Одна буква прихована від гравця. Зверху починають рухатись одна за одною випадкові символи: букви, символ бомби (означає, що потрапивши на букву він знищить її) та символ « _» (означає, що слово потрібно з ’єднати). У польоті гравець може рухати символ по горизонталі. Також можна «скидати» символ, тобто прискорювати його падіння, коли вже вирішено, куди він повинен впасти. Символ летить, поки не наткнеться на слово або на дно склянки. Якщо гравець доповнить слово потрібною буквою, то слово зникне і все, що знаходилося вище нього, опускається на одну клітку вниз. Гравець отримує бали за кожне вірно складене слово. Гра закінчується, коли новий символ не може поміститися в стакан, тому його задача - заповнювати слова буквами, не заповнюючи саму склянку якомога довше, щоб таким чином отримати якомога більше балів.

Програма забезпечує можливість додавати до гри нові слова, змінювати швидкість падіння символів та зберігати результат .

2.4.2. Розробка алгоритму гри

На етапі розробки алгоритму необхідно визначити послідовність дій, які треба виконати для отримання результату. Якщо завдання може бути вирішена декількома способами і, отже, можливі різні варіанти алгоритму рішення, то програміст, використовуючи деякий критерій, наприклад, швидкість рішення алгоритму, вибирає найбільш відповідне рішення. Результатом етапу розробки алгоритму є докладний словесний опис алгоритму або його блок-схема.

Алгоритм, що реалізує логіку гри, виглядає наступним чином:

1. Запуск програми.

2. Після запуску програми відбувається оголошення двовимірного масиву, з подальшим його заповненням порожніми значеннями.

3. Гра починається при запуску таймера.

4. Далі відбувається вибір букви за допомогою генератора випадкових чисел. Відповідно до обраної букви відбувається заповнення перших двох рядків масиву.

5. Потім відбувається вхід в цикл падіння букви. Тіло циклу являє собою зсув слова на один рядок нижче і перевірка на натискання кнопок "Вправо", "Ліворуч (при натисканні цих кнопок відбувається зсув фігури вправо, вліво). Умова закінчення циклу - відсутність вільного місця під буквою.

6. Після виходу з циклу починається перевірка правильно заповнених слів:

– якщо такі є, то відбувається їх очищення і зрушення всіх рядків, що знаходяться вище. Далі відбувається новий вибір букви і повторення алгоритму;

– якщо ж заповнених рядків немає, то відбувається перевірка на вільне місце в третьому рядку. Якщо третій рядок вільний, то відбувається вибір букви і виконання алгоритму циклу. При заповнювання третього рядка відбувається зупинка таймера і виведення результатів гри.

7. Відбувається вихід з гри.

Блок-схема даного алгоритму наведена на рис. 2.1:

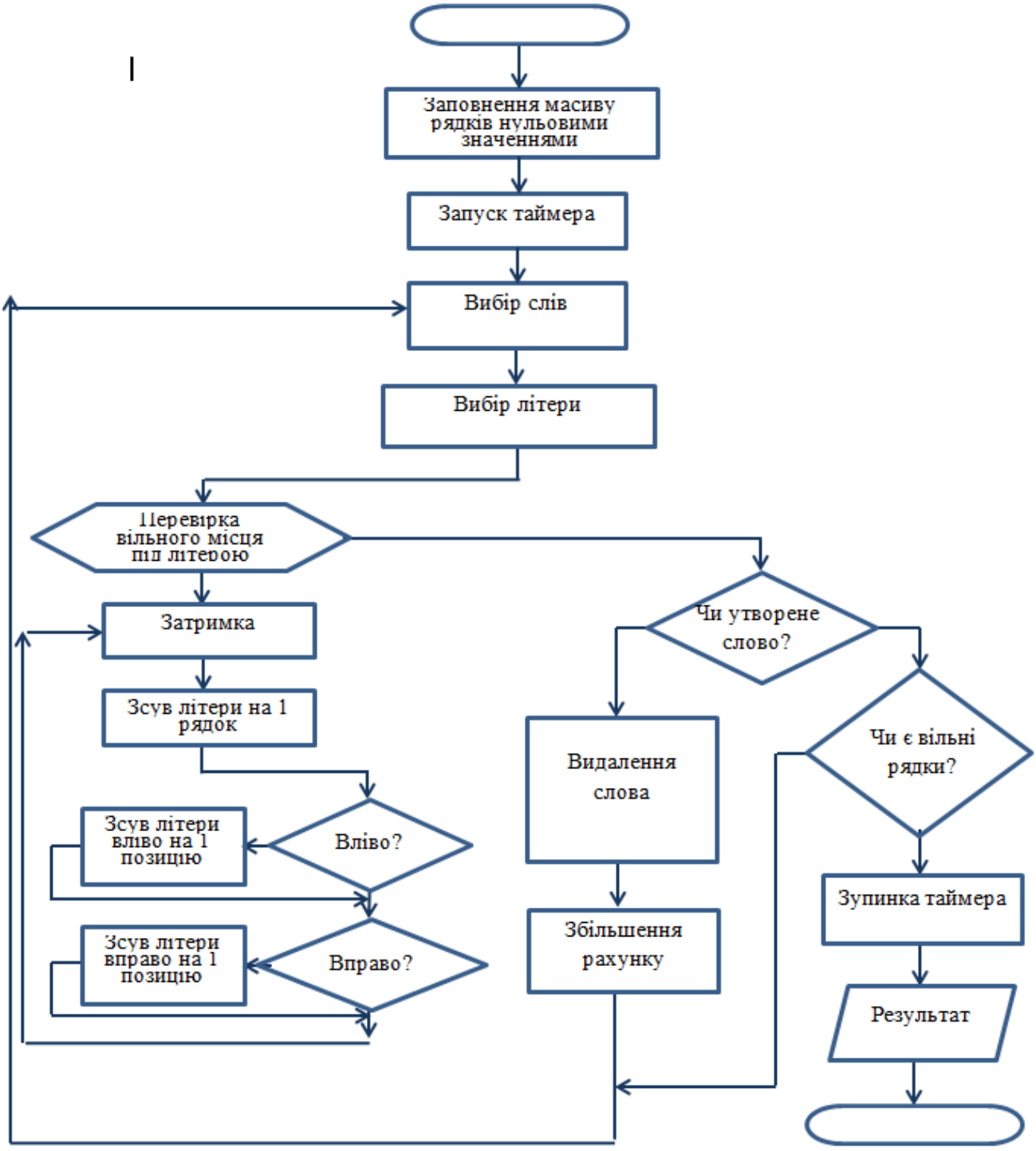

Рис.2.1. Блок-схема алгоритму гри

2.4.3. Опис структури та роботи програми

Головна форма програми містить такі підменю:

– «Нова гра»: гравець може розпочати нову гру;

– «Вихід»: вихід з гри;
– «Додати слово»: допомагає доповнювати масив слів;
– «Чемпіони»: перегляд списку досягнень;
– «Продовжити гру»: повернення до режиму гри;
– «Інформація»: правила гри та значення деяких символів.

Якщо гравець повинен відлучитись чи просто припинити гру на деякий час, він може закрити ігрове поле, а після натиснути на кнопку «повернутись у гру» і продовжити далі.

Структура роботи програми схематично зображена на рис. 2.2:

Рис. 2.2. Структурна схема програми

2.4.4. Програмна реалізація компонентів програми

2.4.4.1. Реалізація графічної частини алгоритму

GDH+ інтерфейс графічних пристроїв. Додатки з графікою, ігри, Computer Aided Design/Computer Aided Manufacture (CAD/CAM - проектування/виробництво за допомогою комп'ютера), програми для малювання, для створення графіків і багато інших типів додатків вимагають від розробників написання коду для роботи з графікою. Використання створюваних користувачем керуючих елементів також передбачає роботу з

графікою. За допомогою своєї бібліотеки класів компанія Microsoft зробила написання коду для роботи з графікою як ніколи простим.

Можливості GDI+:

– робота з окремими частинами малюнків;
– малювання зображення;
– вивід на друк;
– попередній перегляд;
– простір імен Drawing2D;
– простір імен Imaging.

У класі Graphics вміщені поверхні малювання GDI+. Є три основних типи поверхонь малювання:

– вікна і керуючі елементи на екрані;
– сторінки, що посилаються на принтер;
– растрові зображення в пам'яті.

Гра «Український тетрис» вимагає зображення областей поля у вигляді в квадратів. Для цього була використана структура мови C # Rectangle (прямокутник). Для її опису необхідні координати верхнього лівого кута прямокутника (що відповідає значенням вертикалі і горизонталі поля гри), а також його розміри. Структура Region дозволяє об'єднувати різні зображення в одне.

Для заповнення певної області малюнка кольором можливе використання класу Brush. Клас Brush - це абстрактний клас. Для створення екземпляра класу Brush використовуються класи, похідні від класу Brush, такі як SolidBrush, TextureBrush і LinearGradientBrush. Клас Brush знаходиться в просторі імен System. Drawing. Класи TextureBrush і LinearGradientBrush знаходяться в просторі імен System. Drawing. Drawing2D. Ось що дозволяє робити кожен з цих класів:

– SolidBrush заповнює фігуру суцільним кольором;
– TextureBrush дозволяє заповнювати фігуру малюнком, що зберігається в двійковому поданні. При створенні такої кисті потрібно також задавати

обрамляє прямокутник і режим обрамлення. Обрамлений прямокутник визначає, яку порцію малюнка ми повинні використовувати при малюванні, - використовувати весь малюнок цілком абсолютно необов'язково. Для режиму обрамлення існує кілька можливостей, включаючи Tile (черепиця) - TileFiipx, TileFiipY і TileFiipXY, що дозволяють послідовно розбивати зображення на окремі квадрати. За допомогою TextureBrush можна створювати дуже цікаві і дуже вражаючі ефекти;

– LinearGradientBrush містить кисть, яка дозволяє малювати плавний перехід від одного кольору до іншого, причому перший колір переходить в другій під певним кутом. Кути при цьому задаються в градусах. Кут, що дорівнює 0, означає, що перехід від одного кольору до іншого здійснюється зліва направо. Кут, що дорівнює 90 °, означає, що перехід від одного кольору до іншого здійснюється зверху вниз;

Принцип побудови зображення поля гри простий. За допомогою циклічної і умовної конструкцій перевіряються осередки масиву на однакові логічні значення. Оскільки кожному осередку відповідає певна мінімальна область поля гри, то знаючи значення рядка і стовпця масиву можлива побудова структури Rectangle. Об'єднуючи структури Rectangle одного типу, можна отримати два типи Region. Зафарбовуючи ці два Region різними видами "кистей" отримуємо зображення поля.

Так як гра «Український тетрис» побудована на використанні фактора часу, необхідно постійно графічно оновлювати поле цієї гри. Для цього в кожен тик таймера буде відбуватися наведений вище принцип побудови зображення поля гри.

При розробці програми передбачалося використання середовища C # Microsoft Visual Studio 2017 Community. Це середовище розробки програмного забезпечення містить набір шаблонів, які часто використовують при розробки програм. У програмі я використовувався ряд шаблонів:

– Button;

– Label;

– PictureBox;
– TextBox;
– ContextMenuStrip;
– Timer.

Це полегшило завдання в написанні програми, так як частина коду була згенерована автоматично.

2.4.4.2. Реалізація подій гри

Однією з найважливіших функцій будь-якої мови програмування є надання можливостей для управління програмою в ручну. Це має на увазі створення приємного для користувача інтерфейсу.

Ігровим полем є таблиця DataGridView, після її змінення та адаптації. Код програми розбитий на події: «Натиснення на кнопки 1, 2, 3, 4», «Натиснення на клітинку ігрового поля»:

1. Подія «Натиснення на кнопку Нова гра»: Змінна cl відповідає за номер кліку. Він може бути першим чи другим. Якщо гравець почав нову гру, то нам перш за все потрібно помітити cl = 1 та очистити старе поле.

```
for (int i = 0; i < 15; i++)
   for (int j = 0; j < 8; j++)
   {
      dataGridView1.Rows[i].Cells[j].Style.BackColor = Color.White;
      dataGridView1.Rows[i].Cells[j].Style.SelectionBackColor = Color.White;
      dataGridView1.Rows[i].Cells[j].Style.SelectionForeColor = Color.Black;
      dataGridView1.Rows[i].Cells[j].Value = null;
   }
```

Потім потрібно заповнити випадковими словами перші три рядки. Змінна t – випадкова цифра, яка буде першою для пари. Друга буде 10 – t, щоб ми могли їх зафарбувати. D1 та d2 – випадкові координати для пари.

```
int t = rnd.Next(1, 10);
```

```
int d1, d2;
d1 = rnd.Next(0, 3);
d2 = rnd.Next(0, 8);
while (dataGridView1.Rows[d1].Cells[d2].Value != null)
{
    d1 = rnd.Next(0, 3);
    d2 = rnd.Next(0, 8);
}
```

Будемо генерувати нові координати, поки не знайдемо вільну комірку.

```
dataGridView1.Rows[d1].Cells[d2].Value = t;
d1 = rnd.Next(0, 3);
d2 = rnd.Next(0, 8);
while (dataGridView1.Rows[d1].Cells[d2].Value != null)
{
    d1 = rnd.Next(0, 3);
    d2 = rnd.Next(0, 8);
}
```

І теж саме для другої.

```
dataGridView1.Rows[d1].Cells[d2].Value = 10 - t;
```

2. Подія «Зустріч символа зі словом»:

Пройдемося по полю і всі білі непусті комірки будемо заносити у масив val. Коли знайдемо першу пусту - вийдемо з циклів і підтримаємо змінні ksto та kols, які будуть містити координати останньої незаповненої. Якщо розмір масиву не буде перевищувати вільне місце, то ми можемо перенести його в кінець поля.

```
private void button2_Click(object sender, EventArgs e)
{
   int[] val = new int[1000];
   bool n = false;
   int k = 0;
```

```
            for (int i = 0; i < 15; i++)
            {
               for (int j = 0; j < 8; j++)
                  if (dataGridView1.Rows[i].Cells[j].Style.BackColor == Color.White
&& dataGridView1.Rows[i].Cells[j].Value != null)
                  {
                     val[k] = (int)dataGridView1.Rows[i].Cells[j].Value;
                     k++;
                  } else if (dataGridView1.Rows[i].Cells[j].Value == null)
                  {
                     n = true;
                     ksto = j;
                     kols = i;
                     break;
                  }
                  if (n)               break;
                  }
                   if (14 - kols > k / 7)
                   {
                     int kt = 0, ts = kols;
                     while (kt != k)
                     {
                        dataGridView1.Rows[ts].Cells[ksto].Value = val[kt];
                        if (ksto == 7)
                        {
                           ksto = 0;
                           ts++;
                        }
                        else
                           ksto++;
```

```
                kt++;
              }
              kols = ts;
            }
        }
```

3. Подія «очищення поля»:

```
        if (cl == 1)
        {
        if (dataGridView1.Rows[x].Cells[y].Style.BackColor == Color.White)
             {
                dataGridView1.Rows[x].Cells[y].Style.BackColor = Color.DarkGray;
       dataGridView1.Rows[x].Cells[y].Style.SelectionBackColor = Color.DarkGray;
                cl = 2;
                it = y;
                jt = x;
                if (dataGridView1.Rows[x].Cells[y].Value != null)
              valt = (int)dataGridView1.Rows[x].Cells[y].Value;
```

Запам’ятаємо значення клітинки, її x та y – вони знадобляться для перевірки зафарбування. Зробимо cl = 2;

```
        }
        else {
                 cl = 1;
                 dataGridView1.Rows[x].Cells[y].Style.BackColor = Color.White;
           dataGridView1.Rows[x].Cells[y].Style.SelectionBackColor = Color.White;
             }
         }
        else
        {
            if (dataGridView1.Rows[x].Cells[y].Style.BackColor == Color.DarkGray)
            {
```

```
            dataGridView1.Rows[x].Cells[y].Style.BackColor = Color.White;
        dataGridView1.Rows[x].Cells[y].Style.SelectionBackColor = Color.White;
            cl = 1;
        }
        else if (dataGridView1.Rows[x].Cells[y].Style.BackColor == Color.White)
        {
            int it1 = it;
            int jt1 = jt;
```

Коли ми впевнились що другою клітинкою буде біла, будемо перевіряти всі чотири напрямки від другої, якщо в процесі обходу ми попадемо на it jt, то повинні зафарбувати обидві.

```
        void zak(int a, int b, int c, int d)
        {
            dataGridView1.Rows[a].Cells[b].Style.BackColor = Color.Black;
            dataGridView1.Rows[c].Cells[d].Style.BackColor = Color.Black;
            dataGridView1.Rows[a].Cells[b].Style.SelectionBackColor = Color.Black;
            dataGridView1.Rows[c].Cells[d].Style.SelectionBackColor = Color.Black;
        }
```

Обхід доверху.

```
        while (jt1 > 0 && dataGridView1.Rows[jt1 - 1].Cells[y].Style.BackColor ==
Color.Black)
            jt1--;
            if (jt1-1==x&&it1 == y && (valt +
(int)dataGridView1.Rows[x].Cells[y].Value == 10 || valt ==
(int)dataGridView1.Rows[x].Cells[y].Value))
            {
                zak(x, y, jt, it);
                cl = 1;
            }
```

Обхід донизу.

```
if (cl != 1)
{
    it1 = it;
    jt1 = jt;
while (jt1 < 15 && dataGridView1.Rows[jt1 + 1].Cells[y].Style.BackColor ==
Color.Black)
        jt1++;
        if (jt1 + 1 == x && it1 == y && (valt +
(int)dataGridView1.Rows[x].Cells[y].Value == 10 || valt ==
(int)dataGridView1.Rows[x].Cells[y].Value))
        {
            zak(x, y, jt, it);
            cl = 1;
        }
    }
```

Обхід праворуч з переходами рядків.

```
if (cl != 1)
{
    b = false;
    it1 = it;
    jt1 = jt;
    while (true)
    {
        if (it1 == 7)
        {
            jt1++;
            it1 = 0;
    if (dataGridView1.Rows[jt1].Cells[it1].Style.BackColor != Color.Black)
            {
                b = true;
```

```
                    break;
                }
            }
if (jt1 == 15 || dataGridView1.Rows[jt1].Cells[it1 + 1].Style.BackColor !=
Color.Black)
                break;
                it1++;
        }
if (((jt1 == x && it1 + 1 == y) || (b && jt1 == x && it1 == y)) && (valt +
(int)dataGridView1.Rows[x].Cells[y].Value == 10 || valt ==
(int)dataGridView1.Rows[x].Cells[y].Value))
        {
            zak(x, y, jt, it);
            cl = 1;
        }
        Обхід ліворуч з переходами рядків.
        if (cl != 1)
        {
            b = false;
            it1 = it;
            jt1 = jt;
            while (true)
            {
                if (it1 == 0)
                {
                    jt1--;
                    it1 = 7;
                    if (dataGridView1.Rows[jt1].Cells[it1].Style.BackColor !=
Color.Black)
                    {
```

```
                    b = true;
                    break;
                }
            }
if (jt1 == -1 || dataGridView1.Rows[jt1].Cells[it1 - 1].Style.BackColor !=
Color.Black)
                    break;
            it1--;
        }
        if (((jt1 == x && it1 - 1 == y) || (b && jt1 == x && it1 == y))&& (valt +
(int)dataGridView1.Rows[x].Cells[y].Value == 10 || valt ==
(int)dataGridView1.Rows[x].Cells[y].Value))
        {
            zak(x, y, jt, it);
            cl = 1;
        }
    }
```

4. Подія «Натиснення на кнопку виходу з гри»:

Створимо текстовий файл з назвою SaveTetris.txt в папці «Документи» гравця. Будемо записувати в нього цифри від -9 до 9. Якщо цифра від’ємна, то відповідна клітинка зафарбована и має значення –число. Якщо додатна, то незафарбована. Якщо 0 – порожня.

```
String s = System.Environment.GetFolderPath
(System.Environment.SpecialFolder.Personal) + "\\SaveTetris.txt";
File.Delete(pp);
File.AppendAllText(pp, "");
for (int i = 0; i < 15; i++)
   for (int j = 0; j < 8; j++)
       if (dataGridView1.Rows[i].Cells[j].Value == null)
          File.AppendAllText(pp, "0");
```

```
        else if (dataGridView1.Rows[i].Cells[j].Style.BackColor == Color.White)
                File.AppendAllText(pp,
dataGridView1.Rows[i].Cells[j].Value.ToString());
            else
                File.AppendAllText(pp, '-' +
dataGridView1.Rows[i].Cells[j].Value.ToString());
```

5. Подія «Натиснення на кнопку продовження гри»:

Змінимо cl на 1, відкриємо текстовий файл SaveTetris.txt, пройдемося по всім цифрами, які містяться там. Формули для обчислення координат поточної комірки: t/8 і t % 8, де t – порядковий номер в файлі. Оберненим до способу для зберігання гри розкодуємо значення і замінимо властивості поля. Для зчитування з файлу будемо використовувати streamreader, який зчитає вміст файлу і запише його у змінну типу string.

```
cl = 1;
StreamReader streamReader = new StreamReader(pp);
string s = "";
s += streamReader.ReadLine();
int t = 0;
for (int i = 0; i < s.Length; i++)
{
    if (t / 8 > 14)
        break;
    if (s[i] == '-')
    {
        i++;
        dataGridView1.Rows[t / 8].Cells[t % 8].Value = s[i] - '0';
        dataGridView1.Rows[t / 8].Cells[t % 8].Style.BackColor = Color.Black;
dataGridView1.Rows[t / 8].Cells[t % 8].Style.SelectionBackColor = Color.Black;
    } else if (s[i] != '0')
    {
```

```
                dataGridView1.Rows[t / 8].Cells[t % 8].Value = s[i] - '0';
            dataGridView1.Rows[t / 8].Cells[t % 8].Style.BackColor = Color.White;
        dataGridView1.Rows[t / 8].Cells[t % 8].Style.SelectionBackColor = Color.White;
            kols = t / 8;
                }
                else
                {
            dataGridView1.Rows[t / 8].Cells[t % 8].Style.BackColor = Color.White;
        dataGridView1.Rows[t / 8].Cells[t % 8].Style.SelectionBackColor = Color.White;
        dataGridView1.Rows[t / 8].Cells[t % 8].Style.SelectionForeColor = Color.Black;
                dataGridView1.Rows[t / 8].Cells[t % 8].Value = null;
                }
                t++;
            }
            streamReader.Close();
```

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Вхідними дані є ті дані, які користувач повідомляє програмі. Вхідні дані представлені у вигляді програмного коду, який необхідно виконати за певних діях користувача, а саме:

– натискання клавіш Left, Right, Up, Space;

– робота користувача з рядком меню.

Вхідним масивом слів є текстовий файл, що дає можливість доповнювати і вибирати набір слів, що пропонуються у грі.

Вихідні дані в програмі представлені у вигляді графічного відображення вікна гри.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Для розробки даної гри використовувався комп’ютер з наступними параметрами:

- операційна система Windows XP/7/8/10;
- процесор Dual Core Intel чи AMD з частотою 2.8 GHz;
- оперативна пам'ять 4GB ОЗУ;
- відеокарта nVidia GeForce 8600/9600GT, ATI/AMD
- Radeon HD2600/3600;
- DirectX версії 9.0c;
- жорсткий диск мінімум в 15 GB.

2.6.2. Використані програмні засоби

Запропонована реалізація комп’ютерної гри написана на об'єктно-орієнтованій мові програмування C# в середовищі програмування розробки Microsoft Visual Studio 2017 Community.

2.6.3. Виклик та завантаження програми

Для забезпечення доступу до словника гри потрібні файли words.txt (файл словника) та letters.txt (шукані букви), які знаходяться у поточному каталозі. Програма також використовує інші графічні й звукові файли, що містяться в папці директорії.

Програма не потребує попереднього встановлення. Гра має невеликий розмір і для початку достатньо скопіювати гру у будь-який розділ на комп’ютері і запустити її на виконання. Гра запускається виконуваним одним файлом Tetris.exe.

2.6.4. Опис інтерфейсу користувача

Після запуску програми, користувачу стає доступна головна форма (рис. 2.3), яка містить такі підменю:

– «Нова гра»: гравець може розпочати нову гру;
– «Вихід»: вихід з гри;
– «Додати слово»: допомагає доповнювати масив слів;
– «Чемпіони»: перегляд списку досягнень;
– «Продовжити гру»: повернення до режиму гри;
– «Інформація»: правила гри та значення деяких символів.

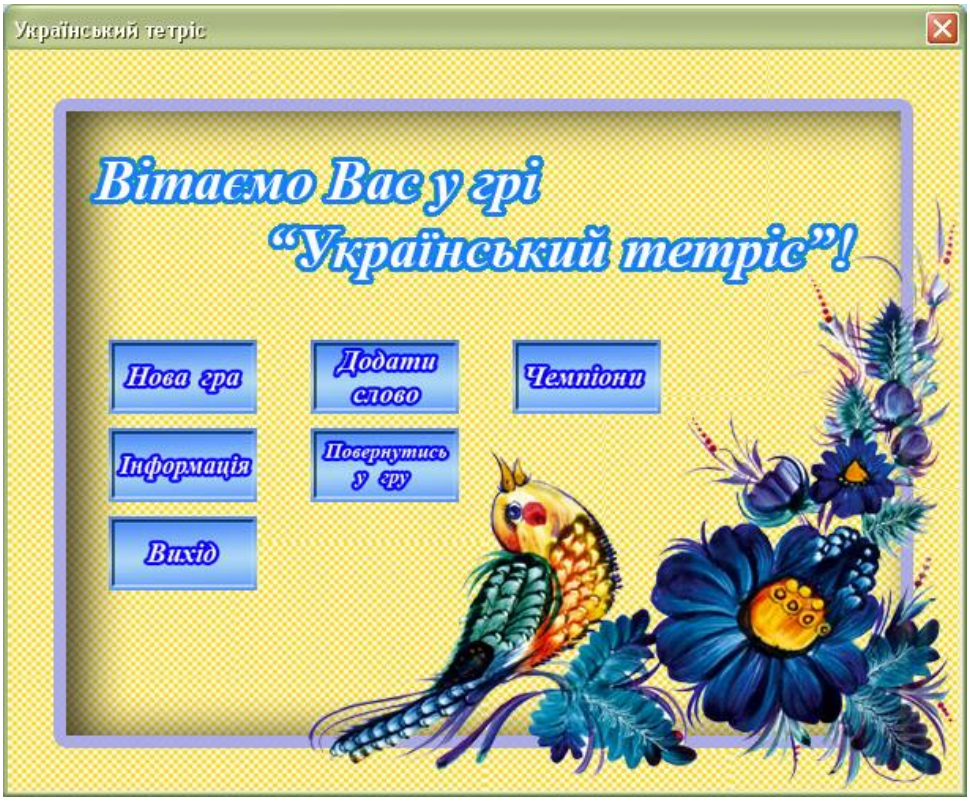

Рис. 2.3. Головна форма гри

Для отримання інформації про правила гри, користувач може скористатися пунктом меню «Інформація» (рис. 2.4)

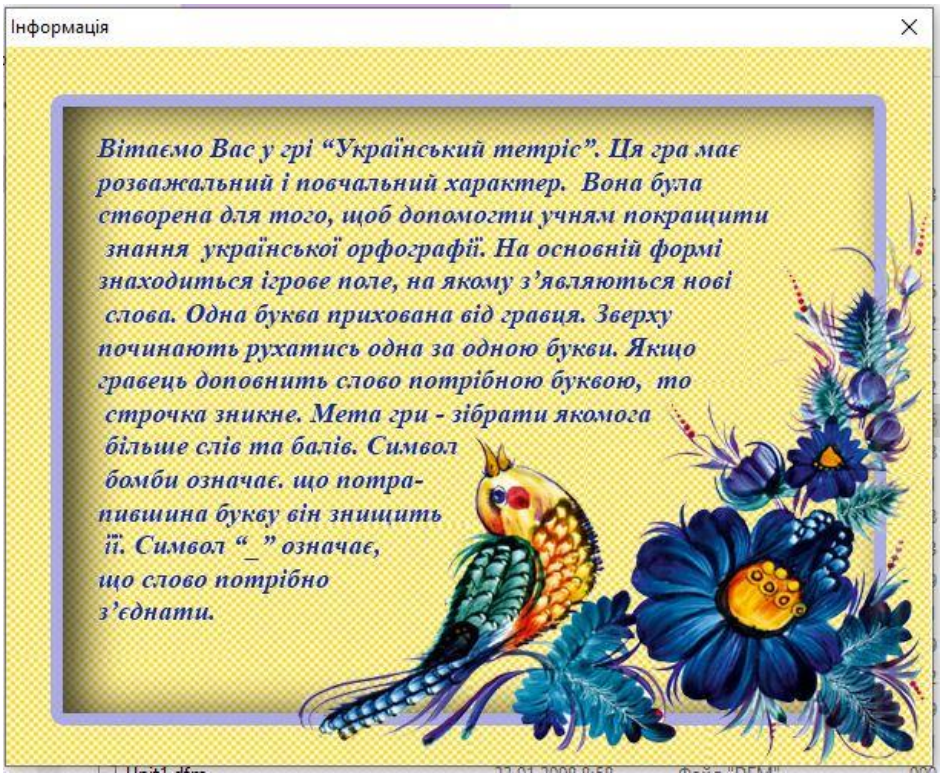

Рис. 2.4. Вкладка «Інформація»

Почати гру користувач може, натиснув на пункт «Нова гра» (рис. 2.5).

Рис. 2.5. Форма для гри

Правила гри: на основній формі знаходиться ігрове поле (прямокутний стакан 13x15), на якому з інтервалом в певний час знизу поля з'являється нове

слово українською мовою. Одна буква в ньому прихована від гравця запитальним знаком. Зверху починають рухатись одна за одною випадкові символи: букви, символ бомби (означає, що потрапивши на букву він знищить її) та символ « _» (означає, що слово потрібно з ’єднати). У польоті гравець може рухати символ по горизонталі. Також можна «скидати» символ, тобто прискорювати його падіння, коли вже вирішено, куди він повинен впасти. Символ летить, поки не наткнеться на слово або на дно склянки. Якщо гравець доповнить слово потрібною буквою, то слово зникне і все, що знаходилося вище нього, опускається на одну клітку вниз. Гравець отримує бали за кожне вірно складене слово. Гра закінчується, коли хоча б один символ досягає першої ступені або новий символ не може поміститися в стакан, тому задача гравця - заповнювати слова буквами, не заповнюючи саму склянку якомога довше, щоб таким чином отримати якомога більше балів.

Кожна буква має своє забарвлення.  - буква, що рухається,  - буква, що закінчила свій хід,  - складова слова.  - особливий символ. Якщо він потрапляє на «платформу» , то прилеглі до нього букви зникають (рис. 2.6 – 2.8).

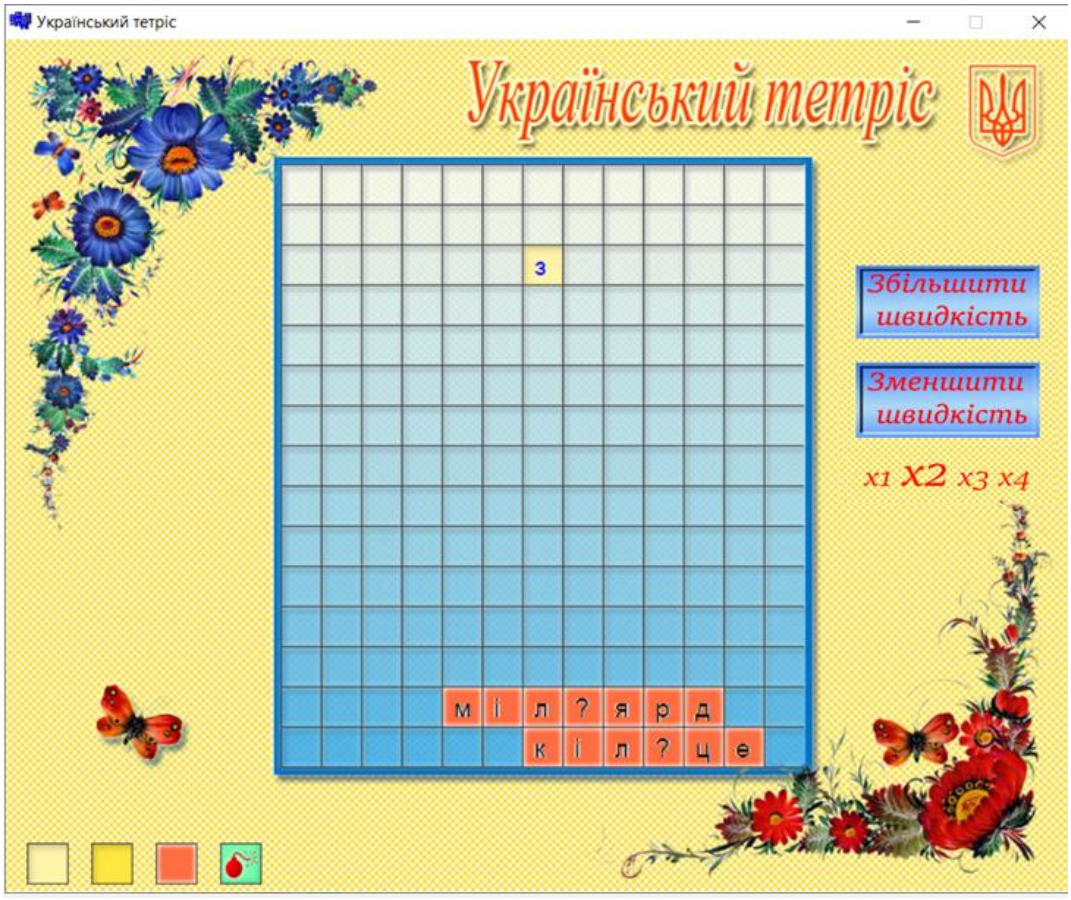

Рис. 2.6. Процес гри

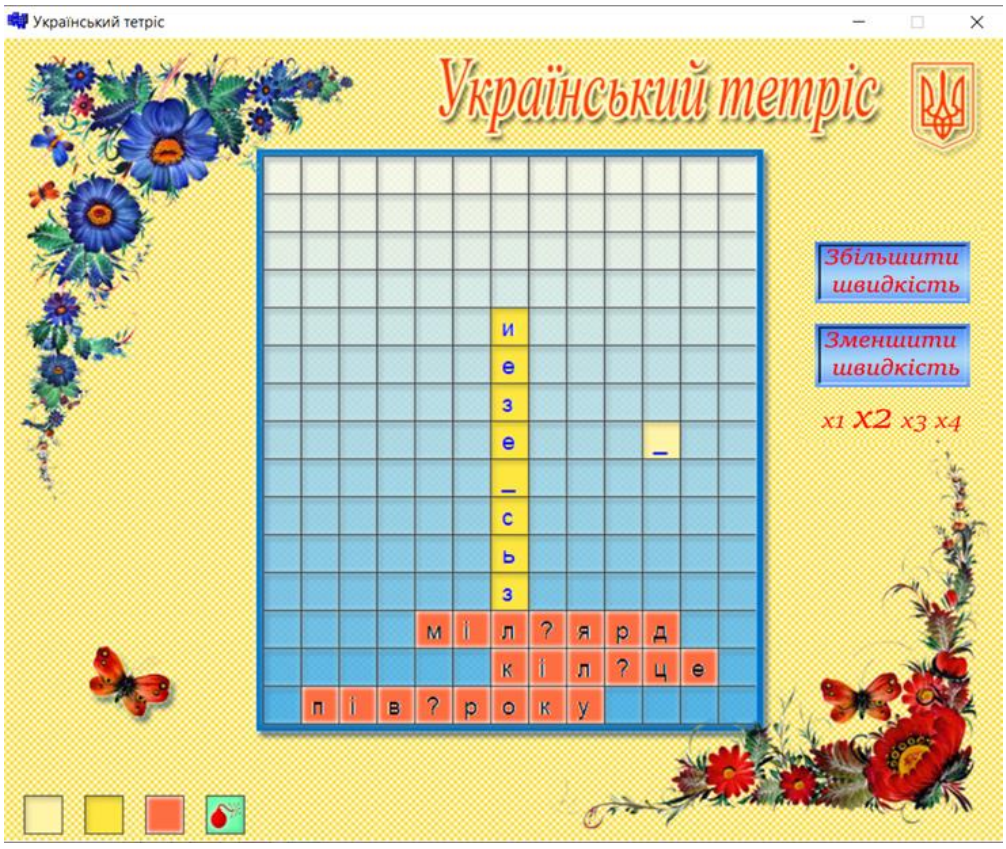

Рис. 2.7. Процес гри

Рис. 2.8. Процес гри

Програма надає можливість змінювати швидкість падіння символів. Праворуч є кнопки, за допомогою яких можна змінювати швидкість падіння букв (рис. 2.9), таким чином кількість очок збільшується.

Рис. 2.9. Налаштування швидкості

Програма надає можливість додавати до гри нові слова. Для цього у головного меню програми необхідно обрати пункт «Додати слово» (рис. 2.10) та завантажити нові слова до файлу-словнику words.txt (2.11).

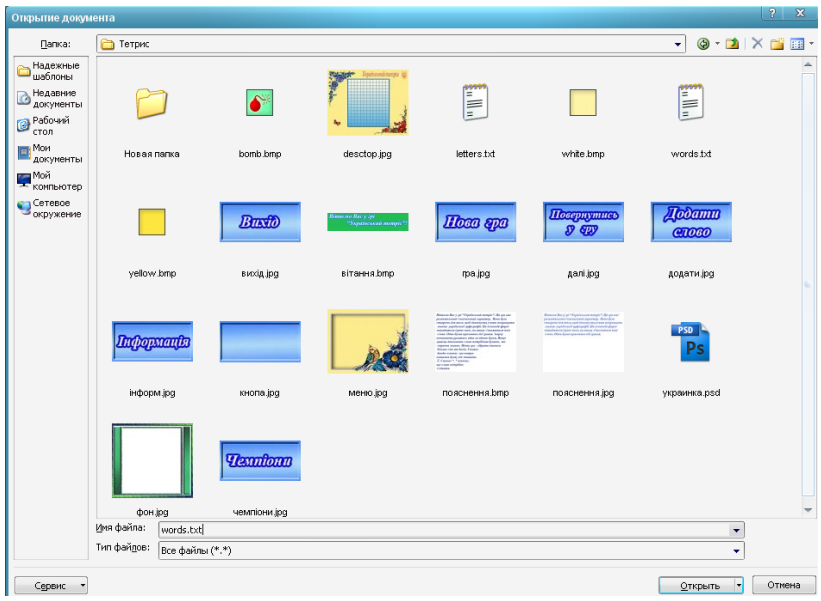

Рис. 2.10. Відкриття файлу для запису нових слів

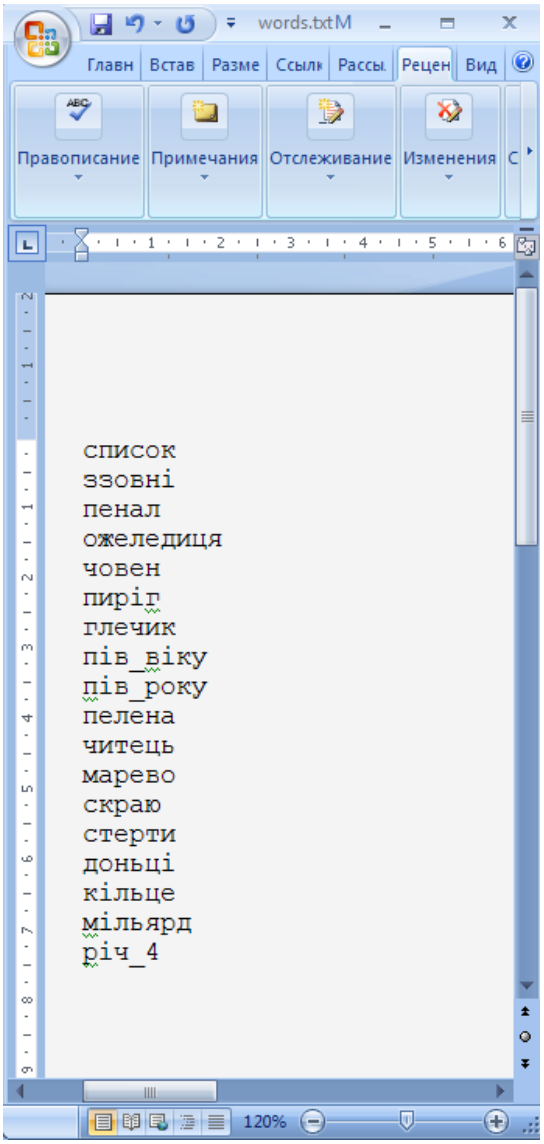

Рис. 2.11. Зміст файлу словника

Додавши нове слово до словнику програми, програма випадковим чином приховає одну з літер з кожного слова та видасть його зображення на екран під час гри. Буква ж, яка була схована в даному слові буде зберігатися у відповідному файлі letters.txt (рис. 2.12).

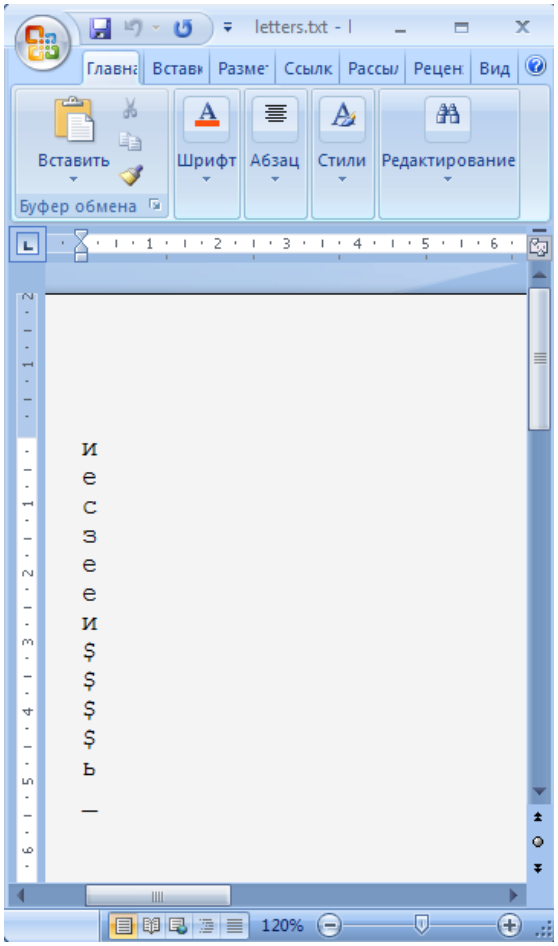

Рис. 2.12. Файл шуканих букв

Таким чином, співставляючи вставлену користувачем букву під час гри з відповідною буквою з файлу, програма визначає правильність орфографій.

Якщо гравець повинен відлучитись чи просто припинити гру на деякий час, він може закрити ігрове поле, а після натиснути на кнопку «повернутись у гру» і продовжити далі. Це можливо завдяки функції збереження гри та пункту «Повернутися в гру».

Вихід з програми виконується натисканням пункту «Вихід».

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1.Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

- передбачуване число операторів - 800;
- коефіцієнт складності програми - 1,5;
- коефіцієнт корекції програми в ході її розробки - 0,05;
- годинна заробітна плата програміста, грн / год - 80.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де:

to - витрати праці на підготовку й опис поставленої задачі (приймається 50);

ти - витрати праці на дослідження алгоритму рішення задачі;

та - витрати праці на розробку блок-схеми алгоритму;

тп - витрати праці на програмування по готовій блок-схемі;

тотл - витрати праці на налагодження програми на ЕОМ;

тд - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \times C \times (1+ p) \text{ , людино-годин,} \quad (3.2)$$

де q - передбачуване число операторів;

С - коефіцієнт складності програми;

р - коефіцієнт кореляції програми в ході її розробки.

$$Q = 800 \cdot 1{,}5 \cdot (1 + 0{,}05) = 1260 \text{ людино-годин.} \quad (3.3)$$

Витрати праці на вивчення опису задачі ti визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{QB}{(75...85)K} \text{ , людино-годин,} \quad (3.3)$$

де В - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі; В=1.2 ... 1.5;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Витрати праці на розробку алгоритму рішення задачі:

$$t_u = \frac{1260 \cdot 1{,}2}{80 \cdot 0{,}8} = 23 \text{ , людино-годин.} \quad (3.4)$$

Витрати на складання програми по готовій блок-схемі:

$$t\alpha = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \quad (3.5)$$

$$t_a = \frac{1260}{20 \cdot 0{,}8} = 78 \text{ людино-годин.} \quad (3.6)$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \tag{3.7}$$

$$t_n = \frac{1260}{25 \cdot 0{,}8} = 63 \text{ людино-годин.} \tag{3.8}$$

Витрати праці на налагодження програми на ЕОМ:

$$t_{отл} = \frac{Q}{(4...5)K} \quad \text{людино-годин.} \tag{3.9}$$

$$t_{отл} = \frac{1260}{4 \cdot 0{,}8} = 393 \text{ людино-годин.} \tag{3.10}$$

за умови комплексного налагодження завдання:

$$t^{к}_{отл} = 1{,}2 \cdot t_{отл}; \tag{3.11}$$

$$t_{отл} = 393 * 1{,}2 = 472{,}5 \tag{3.12}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, \text{ людино-годин,} \tag{3.13}$$

де tдр - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{(15...20)K}, \text{ людино-годин.} \tag{3.14}$$

$$t_{др} = \frac{1260}{15 \cdot 0{,}8} = 105 \text{ людино-годин,} \tag{3.15}$$

tдо - трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др}, \text{ людино-годин.} \tag{3.16}$$

$$tdo = 0{,}75 \cdot 105 = 78 \quad (3.17)$$

$$t_{\partial} = 105 + 78 = 183 \text{ людино-годин.} \quad (3.18)$$

Отримуємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 23 + 78 + 63 + 472 + 183 = 871 \text{ людино-годин.} \quad (3.19)$$

3.2. Витрати на створення програмного забезпечення

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.20)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн,} \quad (3.20)$$

де:

t - загальна трудомісткість, людино-годин;

Спр - середня годинна заробітна плата програміста, грн/година

$$Ç_{ÇÏ} = 871 \cdot 80 = 69680 \text{ ãðí.} \quad (3.21)$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \times C_{М}, \text{ грн,} \quad (3.22)$$

де:

тотл - трудомісткість налагодження програми на ЕОМ, год.

Смч - вартість машино-години ЕОМ, 7 грн/год.

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

$$З_{МВ} = 472 \times 7 = 3307 \text{ грн.} \tag{3.23}$$

$$Êïï = 69680 + 3307 = 72987 \text{ ãðí.} \tag{3.24}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ міс.} \tag{3.25}$$

де:

Bk - число виконавців;

Fp - місячний фонд робочого часу (при 40 годинному робочому тижні Fp=176 годин).

$$T = \frac{871}{1 \cdot 176} = 4{,}9 \text{ міс.} \tag{3.26}$$

Визначено трудомісткість розробленої інформаційної системи (871 люд-год), проведений підрахунок вартості роботи по створенню програми (72987 грн.) та розраховано час на його створення (4,9 міс).

ВИСНОВКИ

На сучасному ринку комп’ютерних пізнавальних ігор з української мови майже немає, тому і було вирішено створити програму, що допоможе учням покращити свої знання української орфографії.

Метою кваліфікаційної роботи є облегшити в ігровій та цікавій формі вивчення українського правопису для школярів. Для досягнення мети під час виконання роботи вирішені наступні завдання:

– ознайомлення з теоретичними даними правил українського правопису та про сучасні зміни в ньому;

– розробка правил та концепції гри на основі відомої гри «Тетрис»;

– розробка алгоритму гри з реалізацією перевірки відповідей гравця;

– програмна реалізація візуальної частини гри;

В програмі реалізована гра «Тетрис», елементами якої є слова української мови з пропущеними літерами та випадаючі кубики з літерами, які потрібно вставити у відповідні пропущені для них місця. Гра має зручний привабливий інтерфейс. Є можливість скористатися довідковим матеріалом, тимчасово припинити гру і повернутися до збереженої гри, переглянути рейтинг гравців.

Запропонована програма написана на об'єктно-орієнтованій мові програмування C# в середовищі програмування розробки Microsoft Visual Studio 2017 Community.

Робота носить прикладний характер. Програма має розважальний і повчальний характер. Вона створена для того, щоб допомогти учням покращити знання української орфографії. Програма може бути використана на уроках української мови та в позаурочних заняттях.

Ше більшої актуальності ця гра набула у зв’язку з публікацією нового українського правопису. Використання даної гри для як під час навчання, так і під час підготовки до ЗНО, допомагає легко перевіряти знання, засвоювати нові правила у приємній для молоді ігровій формі без папірців та книжок.

В економічному розділі визначено трудомісткість розробленої інформаційної системи (871 люд-год), проведений підрахунок вартості роботи по створенню програми (72987 грн.) та розраховано час на його створення (4,9 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання : (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1:2006. – Чинний з 2007–07–01. – К. : Держспоживстандарт України, 2007. – 47 с. – (Система стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України).

2. Бусигін Б.С., Коротенко Г.М., Коротенко Л.М. Прикладна інформатика. Підручник для студентів комп'ютерних спеціальностей. – Дніпропетровськ: Видавництво НГУ, 2004. – 559 с. URL: http://www.programmer.dp.ua/book-ua-k01.php. дата звернення: 15.03.2019.

3. Бусыгин Б.С., Дивизинюк М.М., Коротенко Г.М., Коротенко Л.М. Введение в современную информатику. Учебник. – Севастополь: Издательство СНУЯЭиП, 2005. – 644 с. / URL: http://www.programmer.dp.ua/book-ru-k02.php. дата звернення: 15.01.2018.

4. Ватсон К. Си Шарп/К. Ватсон. - М.: Лори, 2005. - 862 с.

5. Глобальная сеть рефератов – URL: https://otherreferats.allbest.ru/radio/00312640_0.html/ . Дата звернення 26.02.2020

6. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.

7. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. - Київ : Держстандарт України, 1994. – 88 с.

8. Либерти, Д. Программирование на Си Шарп. - М.: Символ-плюс, 2005. - 684 с.

9. Жарков В.А. Компьютерная графика, мультимедиа и игры на VisualC# 2005. - М.: Жарков пресс, 2005. - 812с.

10. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи ” / Укладачі О.Г.

Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 23с.

11. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 122 «Комп'ютерні науки» галузі знань 12 Інформаційні технології/, Л.М. Коротенко , О.С. Шевцова; Нац. гірн. ун-т. – Д : ДВНЗ НГУ, 2019. – 65 с.

12. Новий правопис: повний текст та основні зміни URL: https://osvitoria.media/experience/novyj-pravopys-povnyj-tekst-ta-osnovni-zminy/ Дата звернення 26.02.2021

13. Сайт Теургия.org – URL: https://teurgia.org/tseremonialnaya-magiya/grimuary/959-steganografiya-iogann-tritemij/ Дата звернення 26.02.2021

14. Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги : ДСТУ 3582-97. – Чинний від 1998–07–01. – К. : Держстандарт України, 1998. – 24 с. – (Державний стандарт України).

15. Офіційний сайт середовища розробки Visual Studio Code URL: https://code.visualstudio.com/docs Дата звернення 26.02.2021

16. Питання українського правопису: Постанова Кабінету Міністрів України від 22.05.2019 № 437 / Кабінет Міністрів України – URL: https://zakon.rada.gov.ua/laws/show/437-2019-%D0%BF. Дата звернення 26.02.2021

17. Розробка програми для гри Тетрис URL: https://smekni.com/a/3105802/razrabotka-programmy-dlya-igry-tetris-2/ Дата звернення 26.02.2021

18. Український правопис URL: https://pravopys.net Дата звернення 26.02.2021

19. Файловий архів студентів – URL: https://studfiles.net/preview/4452619// Дата звернення 26.02.2021

20. Шилдт Г. C#: Учебный курс. - Спб.: Питер, 2003. - 512с

ДОДАТОК А

КОД ПРОГРАМИ

MainWindow.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using WpfApp1.Sections;
namespace WpfApp1
{
   public partial class MainWindow : Window
   {
      public MainWindow()
      {
         InitializeComponent();
         Home_page home_Page = new Home_page();
         Frame_main.Navigate(home_Page);
      }
      private void Exit_Click(object sender, RoutedEventArgs e)
      {
         this.Close();
      }
      private void WindowMinimize_Click(object sender, RoutedEventArgs e)
```

```
{
    this.WindowState = WindowState.Minimized;
}
private void Grid_MouseDown(object sender, MouseButtonEventArgs e)
{
    try
    {
        this.DragMove();
    }
    catch
    {
    }
}
private void bt_section_1_Click(object sender, RoutedEventArgs e)
{
    Section_1 section_1 = new Section_1();
    Frame_main.Navigate(section_1);
}
private void bt_section_2_Click(object sender, RoutedEventArgs e)
{
    Section_2 section_2 = new Section_2();
    Frame_main.Navigate(section_2);
}
private void bt_section_3_Click(object sender, RoutedEventArgs e)
{
    Section_3_theory section_3 = new Section_3_theory();
    Frame_main.Navigate(section_3);
}
private void Menu_Click(object sender, RoutedEventArgs e)
{
    if (Section_1.mark >= 5)
    {
        bt_section_2.IsEnabled = true;
        if (Section_2.mark >= 5)
        {
```

```
                bt_section_3.IsEnabled = true;
            }
            else
            {
                bt_section_3.IsEnabled = false;
            }
        }
        else
        {
            bt_section_2.IsEnabled = false;
        }
    }
    private void bt_introduction_Click(object sender, RoutedEventArgs e)
    {
        Introduction introduction = new Introduction();
        Frame_main.Navigate(introduction);
    }
    private void bt_cval_Click(object sender, RoutedEventArgs e)
    {
        Pracktice pracktice = new Pracktice();
        Frame_main.Navigate(pracktice);
    }
  }
}
        for (int i = 0; i < 15; i++)
            for (int j = 0; j < 8; j++)
            {
                dataGridView1.Rows[i].Cells[j].Style.BackColor = Color.White;
                dataGridView1.Rows[i].Cells[j].Style.SelectionBackColor = Color.White;
                dataGridView1.Rows[i].Cells[j].Style.SelectionForeColor = Color.Black;
                dataGridView1.Rows[i].Cells[j].Value = null;
            }
        int t = rnd.Next(1, 10);
        int d1, d2;
        d1 = rnd.Next(0, 3);
```

```
        d2 = rnd.Next(0, 8);
        while (dataGridView1.Rows[d1].Cells[d2].Value != null)
        {
            d1 = rnd.Next(0, 3);
            d2 = rnd.Next(0, 8);
        }
        dataGridView1.Rows[d1].Cells[d2].Value = t;
        d1 = rnd.Next(0, 3);
        d2 = rnd.Next(0, 8);
        while (dataGridView1.Rows[d1].Cells[d2].Value != null)
        {
            d1 = rnd.Next(0, 3);
            d2 = rnd.Next(0, 8);
        }
        dataGridView1.Rows[d1].Cells[d2].Value = 10 - t;
        private void button2_Click(object sender, EventArgs e)
        {
           int[] val = new int[1000];
           bool n = false;
           int k = 0;
           for (int i = 0; i < 15; i++)
           {
              for (int j = 0; j < 8; j++)
                 if (dataGridView1.Rows[i].Cells[j].Style.BackColor == Color.White &&
dataGridView1.Rows[i].Cells[j].Value != null)
                 {
                    val[k] = (int)dataGridView1.Rows[i].Cells[j].Value;
                    k++;
                 } else if (dataGridView1.Rows[i].Cells[j].Value == null)
                 {
                    n = true;
                    ksto = j;
                    kols = i;
                    break;
                 }
```

```
            if (n)              break;
            }
             if (14 - kols > k / 7)
             {
                int kt = 0, ts = kols;
                while (kt != k)
                {
                   dataGridView1.Rows[ts].Cells[ksto].Value = val[kt];
                   if (ksto == 7)
                   {
                      ksto = 0;
                      ts++;
                   }
                   else
                      ksto++;
                   kt++;
                }
                kols = ts;
             }
          }
          if (cl == 1)
          {
          if (dataGridView1.Rows[x].Cells[y].Style.BackColor == Color.White)
               {
                  dataGridView1.Rows[x].Cells[y].Style.BackColor = Color.DarkGray;
      dataGridView1.Rows[x].Cells[y].Style.SelectionBackColor = Color.DarkGray;
                  cl = 2;
                  it = y;
                  jt = x;
                  if (dataGridView1.Rows[x].Cells[y].Value != null)
                  valt = (int)dataGridView1.Rows[x].Cells[y].Value;
```

Запам’ятаємо значення клітинки, її x та y – вони знадобляться для перевірки зафарбування. Зробимо cl = 2;

```
          }
          else {
```

```
                cl = 1;
                dataGridView1.Rows[x].Cells[y].Style.BackColor = Color.White;
            dataGridView1.Rows[x].Cells[y].Style.SelectionBackColor = Color.White;
              }
          }
          else
          {
              if (dataGridView1.Rows[x].Cells[y].Style.BackColor == Color.DarkGray)
              {
                  dataGridView1.Rows[x].Cells[y].Style.BackColor = Color.White;
        dataGridView1.Rows[x].Cells[y].Style.SelectionBackColor = Color.White;
                  cl = 1;
              }
              else if (dataGridView1.Rows[x].Cells[y].Style.BackColor == Color.White)
              {
                  int it1 = it;
                  int jt1 = jt;
          void zak(int a, int b, int c, int d)
          {
              dataGridView1.Rows[a].Cells[b].Style.BackColor = Color.Black;
              dataGridView1.Rows[c].Cells[d].Style.BackColor = Color.Black;
              dataGridView1.Rows[a].Cells[b].Style.SelectionBackColor = Color.Black;
              dataGridView1.Rows[c].Cells[d].Style.SelectionBackColor = Color.Black;
          }
          while (jt1 > 0 && dataGridView1.Rows[jt1 - 1].Cells[y].Style.BackColor == Color.Black)
              jt1--;
              if (jt1-1==x&&it1 == y && (valt + (int)dataGridView1.Rows[x].Cells[y].Value == 10 ||
valt == (int)dataGridView1.Rows[x].Cells[y].Value))
              {
                  zak(x, y, jt, it);
                  cl = 1;
              }
          if (cl != 1)
          {
              it1 = it;
```

```
                jt1 = jt;
 while (jt1 < 15 && dataGridView1.Rows[jt1 + 1].Cells[y].Style.BackColor == Color.Black)
                    jt1++;
                    if (jt1 + 1 == x && it1 == y && (valt + (int)dataGridView1.Rows[x].Cells[y].Value
== 10 || valt == (int)dataGridView1.Rows[x].Cells[y].Value))
                    {
                        zak(x, y, jt, it);
                        cl = 1;
                    }
                }
            if (cl != 1)
            {
                b = false;
                it1 = it;
                jt1 = jt;
                while (true)
                {
                    if (it1 == 7)
                    {
                        jt1++;
                        it1 = 0;
                if (dataGridView1.Rows[jt1].Cells[it1].Style.BackColor != Color.Black)
                        {
                            b = true;
                            break;
                        }
                    }
if (jt1 == 15 || dataGridView1.Rows[jt1].Cells[it1 + 1].Style.BackColor != Color.Black)
                        break;
                        it1++;
            }
if (((jt1 == x && it1 + 1 == y) || (b && jt1 == x && it1 == y)) && (valt +
(int)dataGridView1.Rows[x].Cells[y].Value == 10 || valt ==
(int)dataGridView1.Rows[x].Cells[y].Value))
            {
```

```
            zak(x, y, jt, it);
            cl = 1;
        }
        if (cl != 1)
        {
            b = false;
            it1 = it;
            jt1 = jt;
            while (true)
            {
                if (it1 == 0)
                {
                    jt1--;
                    it1 = 7;
                if (dataGridView1.Rows[jt1].Cells[it1].Style.BackColor != Color.Black)
                    {
                        b = true;
                        break;
                    }
                }
if (jt1 == -1 || dataGridView1.Rows[jt1].Cells[it1 - 1].Style.BackColor != Color.Black)
                    break;
                it1--;
            }
            if (((jt1 == x && it1 - 1 == y) || (b && jt1 == x && it1 == y))&& (valt +
(int)dataGridView1.Rows[x].Cells[y].Value == 10 || valt ==
(int)dataGridView1.Rows[x].Cells[y].Value))
            {
                zak(x, y, jt, it);
                cl = 1;
            }
        }
        String s = System.Environment.GetFolderPath
(System.Environment.SpecialFolder.Personal) + "\\SaveTetris.txt";
        File.Delete(pp);
```

```
        File.AppendAllText(pp, "");
        for (int i = 0; i < 15; i++)
            for (int j = 0; j < 8; j++)
                if (dataGridView1.Rows[i].Cells[j].Value == null)
                    File.AppendAllText(pp, "0");
       else if (dataGridView1.Rows[i].Cells[j].Style.BackColor == Color.White)
                    File.AppendAllText(pp, dataGridView1.Rows[i].Cells[j].Value.ToString());
                else
                    File.AppendAllText(pp, '-' + dataGridView1.Rows[i].Cells[j].Value.ToString());
        cl = 1;
        StreamReader streamReader = new StreamReader(pp);
        string s = "";
        s += streamReader.ReadLine();
        int t = 0;
        for (int i = 0; i < s.Length; i++)
        {
            if (t / 8 > 14)
                break;
            if (s[i] == '-')
            {
                i++;
                dataGridView1.Rows[t / 8].Cells[t % 8].Value = s[i] - '0';
                dataGridView1.Rows[t / 8].Cells[t % 8].Style.BackColor = Color.Black;
  dataGridView1.Rows[t / 8].Cells[t % 8].Style.SelectionBackColor = Color.Black;
             } else if (s[i] != '0')
             {
                dataGridView1.Rows[t / 8].Cells[t % 8].Value = s[i] - '0';
         dataGridView1.Rows[t / 8].Cells[t % 8].Style.BackColor = Color.White;
  dataGridView1.Rows[t / 8].Cells[t % 8].Style.SelectionBackColor = Color.White;
            kols = t / 8;
                }
                else
                {
           dataGridView1.Rows[t / 8].Cells[t % 8].Style.BackColor = Color.White;
dataGridView1.Rows[t / 8].Cells[t % 8].Style.SelectionBackColor = Color.White;
```

```
 dataGridView1.Rows[t / 8].Cells[t % 8].Style.SelectionForeColor = Color.Black;
                dataGridView1.Rows[t / 8].Cells[t % 8].Value = null;
              }
                t++;
          }
          streamReader.Close();
```

App..cs

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Threading.Tasks;
using System.Windows;
namespace WpfApp1
{
   public partial class App : Application
   {
   }
}
          private void Form1_Load (object sender, EventArgs e)
          {
          for (i = 0; i < 13; i++) // заполнение массива пустыми значениями
          {
          for (j = 0; j < 15; j++)
          {
          Tet [i, j] = false;
          Res = true;
          }
          }
          SF = false; // фигура не существует
          textBox2. Text = "0";
          }
          private void timer_Tick (object sender, EventArgs e)
```

```
{
if (SF == false)
{
Random a = new Random (); //
NumbeF = a. Next (0,6);
SF = true;
P = true; //
}
if (SF == true)
{
if (NumbeF == 0) //
{
if (P == true) //
{
Tet [0, 7] = true;
Tet [0, 8] = true;
Tet [1, 7] = true;
Tet [1, 8] = true;
P = false; //
i = 1;
j = 7;
}
if (i < 23 && (Tet [i + 1, j] == false && Tet [i + 1, j + 1] == false))
{
Tet [i - 1, j] = false;
Tet [i - 1, j + 1] = false;
Tet [i + 1, j] = true;
Tet [i + 1, j + 1] = true;
i = i + 1;
}
else
{
SF = false; //
}
}
```

```
if (NumbeF == 1) //
{
if (P == true) //
{
Tet [0, 7] = true;
Tet [1, 7] = true;
Tet [1, 8] = true;
Tet [1, 9] = true;
P = false;
i = 1;
j = 8;
Razp = 0;
}
if (Razp == 0) //
{
if (i < 23 && Tet [i + 1, j - 1] == false && Tet [i + 1, j] == false && Tet [i + 1, j + 1] ==
false) //
{
Tet [i, j] = false;
Tet [i, j + 1] = false;
Tet [i - 1, j - 1] = false;
Tet [i + 1, j - 1] = true;
Tet [i + 1, j] = true;
Tet [i + 1, j + 1] = true;
i = i + 1;
}
else
{
SF = false;
}
}
if (Razp == 1) //
{
if (i < 22 && Tet [i + 2, j - 1] == false && Tet [i + 2, j] == false) //
{
```

```
Tet [i - 1, j] = false;
Tet [i + 1, j - 1] = false;
Tet [i + 2, j - 1] = true;
Tet [i + 2, j] = true;
i = i + 1;
}
else
{
SF = false;
}
}
if (Razp == 2) //
{
if (i < 22 && Tet [i + 1, j - 1] == false && Tet [i + 1, j] == false && Tet [i + 2, j + 1] == false) //
{
Tet [i, j] = false;
Tet [i, j + 1] = false;
Tet [i, j - 1] = false;
Tet [i + 1, j - 1] = true;
Tet [i + 1, j] = true;
Tet [i + 2, j + 1] = true;
i = i + 1;
}
else
{
SF = false;
}
}
if (Razp == 3) //
{
if (i < 22 && Tet [i, j + 1] == false && Tet [i + 2, j] == false)
{
Tet [i - 1, j] = false;
Tet [i - 1, j + 1] = false;
```

```
            Tet [i, j + 1] = true;
            Tet [i + 2, j] = true;
            i = i + 1;
            }
            else
            {
            SF = false;
            }
            }
            }
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace WpfApp1.Sections
{
   /// <summary>
   /// </summary>
   public partial class Answer : Page
   {
      public Answer()
      {
         InitializeComponent();
      }
   }
}
```

```
using System;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
namespace WpfApp1.Sections
{
    /// <summary>
    /// </summary>
    public partial class Home_page : Page
    {
        public Home_page()
        {
            InitializeComponent();
        }
    }
}

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
```

```
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Windows.Xps.Packaging;
{
   public partial class Introduction : Page
   {
      public Introduction()
      {
         InitializeComponent();
         doc.Close();
      }
   }
}

using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.IO;
using System.Globalization;
namespace WpfApp1.Sections
{
   /// <summary>
   /// Interaction logic for Pracktice.xaml
```

```
/// </summary>
public partial class Pracktice : Page
{
    public Pracktice()
    {
        InitializeComponent();
    }
    class Atom
    {
        private int index;
        private string name;
        private float mass;
        private float charge;
        private float sigma;
        private float epsilon;
        private Array coord_n_speed;
        public (int index, string name, float mass, float charge, float sigma, float epsilon, Array coord_n_speed)
        {
            this.index = index;
            this.name = name;
            this.mass = mass;
            this.charge = charge;
            this.sigma = sigma;
            this.epsilon = epsilon;
            this.coord_n_speed = coord_n_speed;
        }
        public string Describe()
        {
            return "Atom name is " + this.name + ", mass is " + this.mass.ToString() + ", charge is " + this.charge.ToString();
        }
        public string Name()
        {
            return this.name;
```

```
    }
    public int Index()
    {
        return this.index;
    }
    public float Charge()
    {
        return charge;
    }
    public float Mass()
    {
        return mass;
    }
    public float Epsilon()
    {
        return epsilon;
    }
    public float Sigma()
    {
        return sigma;
    }
    public Array Coord_n_speed()
    {
        return coord_n_speed;
    }
}
    {
        this.index = index;
        this.name = name;
        this.atom = atom;
        this.bonds = bonds;
        this.angels = angels;
        this.dihedrals = dihedrals;
    }
    public string Describe()
```

```
        {
        }
    }
    public int[,] TransformXY(double[,] xyz)
    {
        int[,] xy = new int[11, 2];
        for (int i = 0; i < 11; i++)
        {
            xy[i, 0] = (int)(xyz[i, 0] * 50 - 300);
            xy[i, 1] = (int)(xyz[i, 2] * 50);
        }
        return xy;
    }

    void ()
    {
            pt = new Point[11] { new Point(TransformXY(xyz)[0, 0], TransformXY(xyz)[0, 1]), new
Point(TransformXY(xyz)[1, 0], TransformXY(xyz)[1, 1]),
            new Point(TransformXY(xyz)[2, 0], TransformXY(xyz)[2, 1]), new
Point(TransformXY(xyz)[3, 0], TransformXY(xyz)[3, 1]), new Point(TransformXY(xyz)[4, 0],
TransformXY(xyz)[4, 1]),
            new Point(TransformXY(xyz)[5, 0], TransformXY(xyz)[5, 1]), new
Point(TransformXY(xyz)[6, 0], TransformXY(xyz)[6, 1]), new Point(TransformXY(xyz)[7, 0],
TransformXY(xyz)[7, 1]),
            new Point(TransformXY(xyz)[8, 0], TransformXY(xyz)[8, 1]), new
Point(TransformXY(xyz)[9, 0], TransformXY(xyz)[9, 1]), new Point(TransformXY(xyz)[10, 0],
TransformXY(xyz)[10, 1]) };
        Pen pen = new Pen(Brushes.Black, 3);
        OpenFileDialog openFile1 = new OpenFileDialog();
        if (openFile1.ShowDialog() == true)
        {
            coord = File.ReadAllLines(openFile1.FileName);
        }
        char tab = '     ';
        OpenFileDialog openFile2 = new OpenFileDialog();
```

```
 if (openFile2.ShowDialog() == true)
 {
    param = File.ReadAllLines(openFile2.FileName);
 }
 OpenFileDialog openFile3 = new OpenFileDialog();
 if (openFile3.ShowDialog() == true)
 {
    connections = File.ReadAllLines(openFile3.FileName);
 }
 par.Clear();
 listb_analizet.Items.Clear();
int i_bonded = 0, i_notrealistic = 0, i_angles = 0, i_dihedralas = 0;
 for (int i = 0; i < param.Length; i++)
 {
    if (param[i].StartsWith("#bonding parameters") == true)
    {
       i_bonded = i;
    }
    else if (param[i].StartsWith("#not realistic bonds") == true)
    {
       i_notrealistic = i;
    }
    else if (param[i].StartsWith("#angles") == true)
    {
       i_angles = i;
    }
    else if (param[i].StartsWith("#dihedrals") == true)
    {
       i_dihedralas = i;
    }
 }
 for (int i = i_dihedralas + 1; i < param.Length; i++)
 {
    for (int j = 0; j < 10; j++)
    {
```

```
        dihedrals_param[i - i_dihedralas - 1, j] = param[i].Split(tab)[j];
    }
}
for (int i = i_angles + 1; i < i_dihedralas; i++)
{
    for (int j = 0; j < 5; j++)
    {
        angles_param[i - i_angles - 1, j] = param[i].Split(tab)[j];
    }
}
for (int i = i_notrealistic + 1; i < i_angles; i++)
{
    for (int j = 0; j < 4; j++)
    {
        not_realistic[i - i_notrealistic - 1, j] = param[i].Split(tab)[j];
    }
}
for (int i = i_bonded + 2; i < i_notrealistic; i++)
{
    for (int j = 0; j < 4; j++)
    {
        bonds_param[i - i_bonded - 2, j] = param[i].Split(tab)[j];
    }
}
for (int i = 1; i < i_bonded; i++)
{
    param_sab = param[i].Split(tab);
    for (int j = 0; j < 4; j++)
    {
        param_sub[j] = param_sab[j + 1];
    }
    string[] clone = new string[param_sub.Length];
    for (int k = 0; k < clone.Length; k++)
    {
        clone[k] = param_sub[k];
```

```
            }
            par.Add(param_sab[0], clone);
        }
    private void listb_analizet_Selected(object sender, RoutedEventArgs e)
    {
        Ellipse border = new Ellipse();
        if(listb_analizet.SelectedIndex != 11)
        {
            int number = listb_analizet.SelectedIndex;
            border.Width = 44;
            border.Height = 44;
  border.Margin = new Thickness(2 * pt[number].X - 44 - 855, 2 * pt[number].Y - 44 - 708, 0, 0);
            border.Fill = radialGradientBrush;
        }
        DrawMolecula2d();
        Graf.Children.Add(border);
    }
  }
}

using System.Windows.Controls;
using System.IO;
using System.IO.Packaging;
using System.Windows.Xps;
using System.Windows.Xps.Packaging;
using MaterialDesignThemes.Wpf;
using WpfApp1.Sections;
using System.Windows;
namespace WpfApp1.Sections
{
    public partial class Section : Page
  {
    public static int mark = 0;
    public Section_1()
    {
```

```
                TextBlock Congratulation1 = new TextBlock();
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
namespace WpfApp1.Sections
{
   public partial class Section_1_test : Page
   {
      public static int a1, a2, a3;
      public static string a4, a5;
      public Section_1_test()
      {
         InitializeComponent();
      }
}

using System.Windows.Controls;
using System.IO;
using System.IO.Packaging;
using System.Windows.Xps;
using System.Windows.Xps.Packaging;
namespace WpfApp1.Sections
{
   public partial class Section_1_: Page
   {
      public Section_1()
      {
```

```
            InitializeComponent();
            dc_viewer.Document = doc.GetFixedDocumentSequence();
            dc_viewer.Zoom = 80;
            doc.Close();
        }
    }
}

using System;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Navigation;
using System.Windows.Shapes;
namespace WpfApp1.Sections
{
    public partial class Sec_2 : Page
    {
        public static int mark = 0;
        public Section_2()
        {
            InitializeComponent();
        }
            String s = System.Environment.GetFolderPath
(System.Environment.SpecialFolder.Personal) + "\\SaveTetris.txt";
            File.Delete(pp);
            File.AppendAllText(pp, "");
            for (int i = 0; i < 15; i++)
                for (int j = 0; j < 8; j++)
                    if (dataGridView1.Rows[i].Cells[j].Value == null)
                        File.AppendAllText(pp, "0");
           else if (dataGridView1.Rows[i].Cells[j].Style.BackColor == Color.White)
                        File.AppendAllText(pp, dataGridView1.Rows[i].Cells[j].Value.ToString());
                    else
```

```
                File.AppendAllText(pp, '-' + dataGridView1.Rows[i].Cells[j].Value.ToString());
        cl = 1;
        StreamReader streamReader = new StreamReader(pp);
        string s = "";
        s += streamReader.ReadLine();
        int t = 0;
        for (int i = 0; i < s.Length; i++)
        {
            if (t / 8 > 14)
               break;
            if (s[i] == '-')
            {
               i++;
               dataGridView1.Rows[t / 8].Cells[t % 8].Value = s[i] - '0';
               dataGridView1.Rows[t / 8].Cells[t % 8].Style.BackColor = Color.Black;
  dataGridView1.Rows[t / 8].Cells[t % 8].Style.SelectionBackColor = Color.Black;
             } else if (s[i] != '0')
             {
               dataGridView1.Rows[t / 8].Cells[t % 8].Value = s[i] - '0';
         dataGridView1.Rows[t / 8].Cells[t % 8].Style.BackColor = Color.White;
  dataGridView1.Rows[t / 8].Cells[t % 8].Style.SelectionBackColor = Color.White;
           kols = t / 8;
               }
               else
               {
           dataGridView1.Rows[t / 8].Cells[t % 8].Style.BackColor = Color.White;
dataGridView1.Rows[t / 8].Cells[t % 8].Style.SelectionBackColor = Color.White;
dataGridView1.Rows[t / 8].Cells[t % 8].Style.SelectionForeColor = Color.Black;
               dataGridView1.Rows[t / 8].Cells[t % 8].Value = null;
               }
               t++;
        }
        streamReader.Close();
  }
                                                                                    }
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_ doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_.ppt	Презентація кваліфікаційної роботи