**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО»**

**Інститут прикладного системного аналізу**

**Кафедра математичних методів системного аналізу**

**Курсова робота**

**з дисципліни «Алгоритмізація та програмування»**

**на тему: «Гра Пятнашки»**

Виконав:

студент 1 курсу, групи КА-07

Висоцький Дмітрій \_\_\_\_\_\_\_\_\_\_

Київ – 2021 рік

**ЗМІСТ**

[ВСТУП 3](#_Toc73711240)

[РОЗДІЛ 1 6](#_Toc73711241)

[**ОБГРУТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ, ПОСТАНОВКА ЗАДАЧІ ПРОЕКТУВАННЯ 6**](#_Toc73711242)

[**1.1 Проблеми та існуючі методи розробки 6**](#_Toc73711243)

[**1.2 Обґрунтування вибору середовища та інструментів розробки 9**](#_Toc73711244)

[**1.3 Уточнена постановка задачі 10**](#_Toc73711245)

[РОЗДІЛ 2 12](#_Toc73711246)

[**РОЗРОБКА АЛГОРИТМУ РОЗВ’ЯЗКУ ЗАДАЧІ 12**](#_Toc73711247)

[**2.1 Алгоритм реалізації двовимірного масиву в одновимірному 12**](#_Toc73711248)

[**2.2 Алгоритм переміщення частин головоломки 12**](#_Toc73711249)

[**2.3 Алгоритм перемішування головоломки 15**](#_Toc73711250)

[**2.4 Алгоритм перевірки на зібраність головоломки 16**](#_Toc73711251)

[РОЗДІЛ 3 17](#_Toc73711252)

[**ОПИС РОЗРОБЛЕНОГО ПРОГРАМНОГО ПРОДУКТУ 17**](#_Toc73711253)

[**3.1 Опис структури програми 17**](#_Toc73711254)

[**3.2 Опис основних функцій 19**](#_Toc73711255)

[**3.3 Опис інтерфейсу 21**](#_Toc73711256)

[**3.4 Перевірка роботи програми 22**](#_Toc73711257)

[Висновки 25](#_Toc73711258)

[Список використаних джерел 26](#_Toc73711259)

[ДОДАТКИ 27](#_Toc73711260)

ВСТУП

**П'ятна́шки** (п'ятнадцять) — популярна [головоломка](https://uk.wikipedia.org/wiki/%D0%93%D0%BE%D0%BB%D0%BE%D0%B2%D0%BE%D0%BB%D0%BE%D0%BC%D0%BA%D0%B0), придумана у [1878 році](https://uk.wikipedia.org/wiki/1878) Ноєм Чепменом. Складається з 15 однакових [квадратних](https://uk.wikipedia.org/wiki/%D0%9A%D0%B2%D0%B0%D0%B4%D1%80%D0%B0%D1%82) пластинок з нанесеними числами від 1 до 15. Пластинки поміщаються в квадратну коробку, довжина сторони якої в чотири рази більша довжини сторони пластинок, відповідно в коробці залишається незаповненим одне квадратне поле. Мета гри — переміщаючи пластинки по коробці добитися впорядковування їх по номерах (як зображено на рисунку), бажано зробивши якомога менше переміщень.



«Рисунок 1 - головоломка»

Винахідником головоломки був Ной Палмер Чепмен, поштмейстер з [Канастоти](https://uk.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BD%D0%B0%D1%81%D1%82%D0%BE%D1%82%D0%B0_(%D0%9D%D1%8C%D1%8E-%D0%99%D0%BE%D1%80%D0%BA)), який ще в [1874 році](https://uk.wikipedia.org/wiki/1874) показував друзям головоломку, що складалася з шістнадцяти пронумерованих квадратиків, які треба було скласти в ряди по чотири штуки так, щоб сума чисел в кожному ряду була рівна 34. Потім син Ноя Чепмена, Френк Чепмен привіз допрацьовані головоломки в [Сиракузи (штат Нью-Йорк)](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%80%D0%B0%D0%BA%D1%83%D0%B7%D0%B8_(%D0%9D%D1%8C%D1%8E-%D0%99%D0%BE%D1%80%D0%BA)), а потім в [Гартфорд](https://uk.wikipedia.org/wiki/%D0%93%D0%B0%D1%80%D1%82%D1%84%D0%BE%D1%80%D0%B4_(%D0%9A%D0%BE%D0%BD%D0%BD%D0%B5%D0%BA%D1%82%D0%B8%D0%BA%D1%83%D1%82)) (Коннектикут), де слухачі [Американської школи для глухих](https://uk.wikipedia.org/w/index.php?title=%D0%90%D0%BC%D0%B5%D1%80%D0%B8%D0%BA%D0%B0%D0%BD%D1%81%D1%8C%D0%BA%D0%B0_%D1%88%D0%BA%D0%BE%D0%BB%D0%B0_%D0%B4%D0%BB%D1%8F_%D0%B3%D0%BB%D1%83%D1%85%D0%B8%D1%85&action=edit&redlink=1) почали виробництво головоломки. До [1879 року](https://uk.wikipedia.org/wiki/1879) вона вже продавалася не тільки в Хартфорді, але і в [Бостоні](https://uk.wikipedia.org/wiki/%D0%91%D0%BE%D1%81%D1%82%D0%BE%D0%BD). Тоді про гру дізнався художник по дереву Маттіас Райс. У грудні 1879 року він почав бізнес по виробництву нової головоломки під назвою «Дорогоцінна головоломка» [англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *Gem Puzzle*. На початку 1880 року Чарльз Певі, [дантист](https://uk.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D1%82%D0%B8%D1%81%D1%82) з [Вустера](https://uk.wikipedia.org/wiki/%D0%92%D1%83%D1%81%D1%82%D0%B5%D1%80_(%D0%9C%D0%B0%D1%81%D1%81%D0%B0%D1%87%D1%83%D1%81%D0%B5%D1%82%D1%81)), привернув увагу громадськості, запропонував грошову винагороду за рішення задачі збирання головоломки, що додало популярності новій забаві. Весною того ж року гра досягла [Європи](https://uk.wikipedia.org/wiki/%D0%84%D0%B2%D1%80%D0%BE%D0%BF%D0%B0). [21 лютого](https://uk.wikipedia.org/wiki/21_%D0%BB%D1%8E%D1%82%D0%BE%D0%B3%D0%BE) [1880 року](https://uk.wikipedia.org/wiki/1880) Ной Чепмен спробував оформити патент на свій винахід, проте заявка на патент була відхилена, оскільки мало відрізнялася від вже оформленого трьома роками раніше патенту «Хитрі блоки».

Сьогодні усім улюблені п'ятнашки стали не тільки грою, що розвиває інтелект та математичні здібності, а й являється елементом змагань у швидкісній збірці головоломок. WCA – World cube assosiation, це всесвітньо-відомі змагання швидкісної збірці головоломок, основними дисциплінами якої є збірка кубиків 3 на 3, 4 на 4, 5 на 5 і тд. Розглянута головоломка поки що входить у неофіційну дисципліну, але уже існує чимало охочих ставити світові рекорди на цьому снаряді.

Гра п'ятнашки є фізичним втіленням мат логіки на справді. Перестановки чисел від одного до шістнадцяти та зміна їх положень яскраво демострою елементи комбінаторики. Позначивши пусту клітину числом 16, властивості головоломки можна одержувати із властивостей [перестановок](https://uk.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B0) чисел 1..16. Так кожне переміщення у головоломці відповідає деякій транспозиції числа 16 і деякого іншого числа. Кожна така транспозиція змінює парність перестановки чисел. Якщо в початковому розміщенні пуста клітина розташована у нижньому правому куті, то на своє місце вона може повернутися за парну кількість кроків, тобто парність у цьому випадку не зміниться. Тому якщо початкова перестановка є непарною (як наприклад на другому малюнку) то головоломка розв'язку не має. У загальному випадку розв'язку не існує, якщо непарним є число n+i+j,де *n* — рівно нулю для парних початкових перестановок чисел 1..16 і одиниці для непарних, *i, j* — позиція пустої клітини.

Натомість кожне початкове розміщення, для якого число n+i+j є парним, має розв'язок головоломки. Цей факт достатньо довести для випадку коли у початковому розміщенні пуста клітина розміщується у нижньому правому куті. Потрібно розглянути множину перестановок, що можна одержати з правильного розміщення (що є розвязком головоломки). Треба довести, що ця множина є множиною парних перестановок чисел 1..15.

Спершу можна побачити, що серед перестановок, які можна здійснити є цикл *(11, 12, 15)* (здійснюючи рух південь-схід-північ-захід). Тоді можна здійснити перестановки так щоб числа 11 і 12 були фіксовані, а на місці 15 опиниться довільне інше число. Тобто в групі перестановок також є цикли *(11, 12, x)* для довільного *x* з множини 1-10, 13, 14. Такі цикли породжують групу парних перестановок, що й доводить твердження.

РОЗДІЛ 1

ОБГРУТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ, ПОСТАНОВКА ЗАДАЧІ ПРОЕКТУВАННЯ

* 1. Проблеми та існуючі методи розробки

Перша мова програмування, что доступна для програмування, а саме роботи з масивами на платформі Mac OS є мова С.

C — універсальна, процедурна, імперативна мова програмування загального призначення, розроблена у 1972 році Деннісом Рітчі у Bell Telephone Laboratories з метою написання нею операційної системи UNIX. Хоча С і було розроблено для написання системного програмного забезпечення, наразі вона досить часто використовується для написання прикладного програмного забезпечення. С імовірно, є найпопулярнішою у світі мовою програмування за кількістю вже написаного нею програмного забезпечення, доступного під вільними ліцензіями коду та кількості програмістів, котрі її знають. Версії компіляторів для мови С існують для багатьох операційних систем та апаратних архітектур. C здійснила великий вплив на інші мови програмування, особливо на C++.

Для того, аби створити інтерфейс, нам потрібно використовувати ООП(Об'є́ктно-орієнто́ване програмува́ння), тому використовуємо мову С++.

C++ — мова програмування високого рівня з підтримкою кількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної. Розроблена Б'ярном Страуструпом (англ. Bjarne Stroustrup) в AT&T Bell Laboratories(Мюррей-Хілл, Нью-Джерсі) 1979 року та початково отримала назву «Сі з класами».

Для того, щоб зробити інтерфейс, потрібно використовувати спеціальну оболочку. Найпоширеніші з них: SFML, QT, Cairo, Cocos2D-X, GTK+, Open GL.

SFML (Проста і Швидка Мультимедійна Бібліотека) є портативним і легким у використанні API для мультимедіа-програмування. Написано мовою C++, але доступні прив'язки і для C, D, Python, Ruby, OCaml, .Net. Це об'єктно-орієнотована альтернатива для SDL.

SFML забезпечує використання 2D-графіки з апаратним прискорення OpenGL. SFML також може бути використаний для OpenGL-проектів. SFML також надає різні модулі для полегшення програмування ігор і мультимедійних додатків.



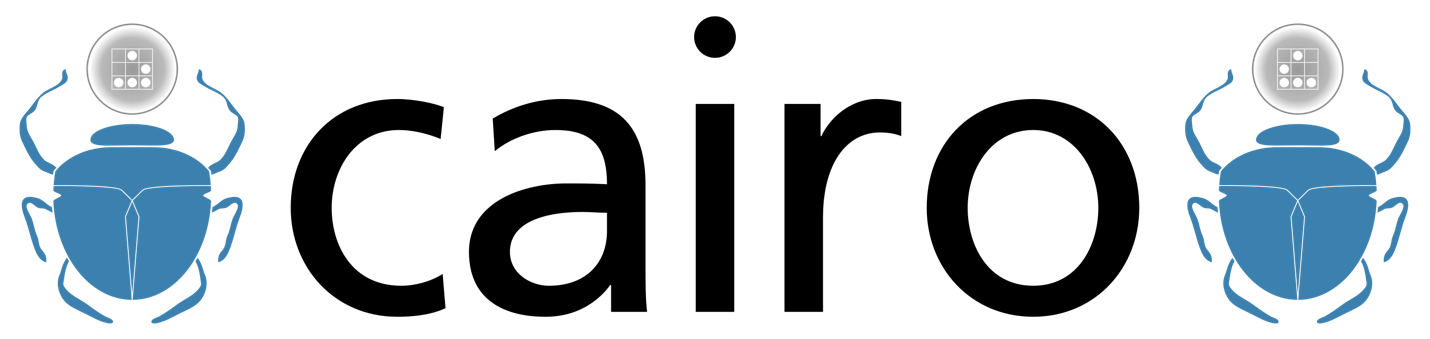
«Рисунок 1.1.1 – логотип sfml»

Qt (варіант вимови від розробників *cute* — *к'ют*) — [крос-платформовий](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B3%D0%B0%D1%82%D0%BE%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%BE%D0%B2%D1%96%D1%81%D1%82%D1%8C) інструментарій розробки [програмного забезпечення](https://uk.wikipedia.org/wiki/%D0%9F%D0%97) (ПЗ) [мовою програмування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) [C++](https://uk.wikipedia.org/wiki/C%2B%2B). Дозволяє запускати написане за його допомогою ПЗ на більшості сучасних [операційних систем](https://uk.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) (ОС), просто [компілюючи](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%96%D0%BB%D1%8F%D1%82%D0%BE%D1%80) текст програми для кожної операційної системи без зміни [сирцевого коду](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%80%D1%86%D0%B5%D0%B2%D0%B8%D0%B9_%D0%BA%D0%BE%D0%B4). Містить всі основні [класи](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), які можуть бути потрібні для розробки [прикладного програмного забезпечення](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F), починаючи з елементів [графічного інтерфейсу](https://uk.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D1%96%D1%87%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) й закінчуючи класами для роботи з мережею, [базами даних](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85), [OpenGL](https://uk.wikipedia.org/wiki/OpenGL), [SVG](https://uk.wikipedia.org/wiki/SVG) і [XML](https://uk.wikipedia.org/wiki/XML). Бібліотека дозволяє керувати [потоками](https://uk.wikipedia.org/wiki/%D0%9D%D0%B8%D1%82%D0%BA%D0%B0_(%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), працювати з [мережею](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0)та забезпечує крос-платформовий доступ до [файлів](https://uk.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB).



«Рисунок 1.1.2 – логоип qt»

Cairo – це програмна бібліотека для відтворення векторної графіки з відкритим вихідним кодом. Включає в себе апаратно-незалежний прикладний програмний інтерфейс для розробників програмного забезпечення. Cairo надає графічні примітиви для відтворення двовимірних зображень за допомогою різноманітних бекендов. Коли є можливість, Cairo використовує апаратне прискорення.



«Рисунок 1.1.3 – логотип cairo»

Cocos2d - крос-платформний фреймворк, який використовується для розробки інтерактивних додатків і ігор (переважно для мобільних пристроїв). Є відкритим програмним забезпеченням. Cocos2d містить безліч відгалужень, таких як Cocos2d-ObjC, Cocos2d-x, Cocos2d-html5 і Cocos2d-XNA. Також в співтоваристві Cocos2d є кілька незалежних редакторів, призначених для редагування спрайтів, частинок, шрифтів і тайлових карт.



«Рисунок 1.1.4 – логотип cocos»

OpenGL (Open Graphics Library) - специфікація, що визначає платформонезавісимость (незалежний від мови програмування) програмний інтерфейс для створення програмного забезпечення, що використовують двовимірну і тривимірну комп'ютерну графіку. Розробляється в США і Європі, має тип ліцензій GNU- / EU /.



«Рисунок 1.1.5 – логотип open gl»

### **1.2 Обґрунтування вибору середовища та інструментів розробки**

Однією з найпопулярніших мов програмування є C++, яка застосовується повсюди, починаючи від веб-браузерів до тривимірних відеоігор, повністю оновлено і опубліковано як стандарт ISO/IEC 14882:2011 «Інформаційні технології. Мови програмування. C++».

У стандарті ISO/IEC 14882:2011 дається визначення мови програмування і встановлюються вимоги до використання. Це перший великий перегляд стандарту, починаючи з 1998 року, відомий як С++11. Нові функції розширюють традиційні характеристики гнучкості та ефективності даної мови програмування, а також доповнюють його спрощеними можливостями використання інших сучасних мов програмування.

«В останні роки в галузі приділялося більше уваги новим середовищам програмування типу Java, які робили акцент на зручності програміста на шкоду потужності і продуктивності», - роз'яснює голова робочої групи ISO розробника-стандарту Херб Саттер.

«Ці мови, як і раніше будуть використовуватися. Проте «традиційне» програмування з допомогою таких мов як С++, що базується на виразності і розумній філософії, що полягає в уникненні зайвої функціональності, не пішло в минуле. Тепер з використанням кращих характеристик керованих мов код С++ чистий і надійний, як і коди інших сучасних мов, а продуктивність висока, а також можливий повний доступ до систем більш низького рівня».

«С++ продовжує використовуватися повсюдно, включаючи компілятори, веб-браузери типу Chrome від Google, платформи соціальних мереж типу Facebook і операційні системи типу Windows».

C++ - мова програмування загального призначення на основі мови C (стандарт ISO/IEC 9899:1999). Додаткво до можливостей C, C++ пропонує додаткові типи даних, класи, шаблони, винятки, простор імен, можливості використання функції одного оператора або одного імені для виконання різних операцій, посилання, оператори управління вільною пам'яттю і додаткові бібліотеки.

«C++ залишається одним з найважливіших інструментів розробників, і вибирається для ресурсоємних додатків практично в будь-якій області», - каже Саттер. «Завдяки новій редакції користувачі отримають переваги від спільної роботи експертів всіх країн світу, які підготували оновлений і покращений міжнародний стандарт, який гармонізує і поширює передовий досвід».

SFML бібліотека використовується здебільшого невеликими стартапами і програмістами, для яких створення ігор - хобі. SFML популярний серед невеликих команд завдяки тому, що розробка графічної частини програми не вимагає написання великих обсягів коду.

Ця програма є достаньо простою, тому в данному проєкті буде використано можливості графічної бібліотеку SFML. Це є найбільш популярною бібліотекою для створення графіки на мові С++. Дана бібліотека найпростіше підключається до програмного середовища Xcode. Паралельним плюсом є те, що створення та робота є достатньо простою, адже усе відбувається в Xcode.

### **1.3 Уточнена постановка задачі**

Дана задача вимагає розбиття на додаткові підзадачі, оскільки так буде більш коректно та більш сприйнятливо. Гра пятнашки – це двовимірний масив, що записаний в одновиміному. Так, саме одновимірній масив є костяком нашої задачі, цей вибір зумовлений тим, що нам потрібні індекси при подальшому порівнянні. Тому потрібно зробити план, що має такий вигляд:

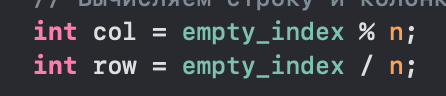
1. Дослідити, що таке пятнашки, як працюють та підготувати предметну базу для опрацювання данної задачі.
2. Проаналізувати підзадачі данної задачі
3. Побудувати модель задачі
4. Розробити примітивну програму для розуміння недоліків постановки
5. Усунення недоліків постановки
6. Перенести данний проєкт в середовище SFML. Зміна коду та підгон до данного середеовища. Створення вікон та розробка коду для самого додатку.
7. Тестування та перевірка коректності розробленої програми. Усунення недоліків та продовження тестування.
8. Розробити інтерфейс користувача

РОЗДІЛ 2

РОЗРОБКА АЛГОРИТМУ РОЗВ’ЯЗКУ ЗАДАЧІ

2.1 Алгоритм реалізації двовимірного масиву в одновимірному

Як вже було визначено у минулому пункті, у данній реалізації буде використовуватись одновимірний масив. Звичайно, обовязково потрібно буде потрібно виділяти позицію, а саме ряд та колонка, для того, щоб коретно визначати знаходження кожної частинки. В алгоритмі не віділено місце для данної функції підзадачі задачі. Коректніше буде назвати данну процедуру підзадача підзадачі. Ця частина допомагає більш точно зрозуміти реалізацію програми.



«Рисунок 2.1.1 – визначення координат»

### **2.2 Алгоритм переміщення частин головоломки**

Переміщення частин головоломки буде реалізовано шляхом натиснення клавіш. Потрібно весь час знати положення пустої плажки. Це потрібно для того, аби при натисканні клавіш ми знали куди плажка має рухатись. Відповідно до клавіші, потрібно визначати яку саме плажку потрібно пересунути на місце пустої. Це реалізовано за таким алгоритмом:

1. Маємо положення пустої плажки
2. Визначаємо яку саме клавішу було натиснуто
3. Якщо натиснуто вниз, то звертаємося на плажку, що знаходиться вище, якщо натиснуто вверх, то звертаємося на плажку,що знаходиться нище, якщо натиснуто вправо, то звертаємося на плажку,що знаходиться зліва, якщо натиснуто вліво, то звертаємося на плажку,що знаходиться зправа. Оскільки ми маємо індекс пустої плажки, то процедура знаходження відповідної плажки, що потрібно пересунути, відповідна знаходження сусідних цілих координат.
4. Присвоюємо відповідної плажки до пустої. Позиція пустої стає старою позицією відповідної.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 13 | 14 | 15 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 |  |
| 13 | 14 | 15 | 12 |

«Рисунок 2.2.1 – переміщення плашок»

### **2.3 Алгоритм перемішування головоломки**

Використовуваємо доволі простий алгоритм перемішування данної головоломки. Потрібно використовувати функцію переміщення для того, щоб можна було потім зібрати головоломку. Оскільки перенесення буде некоректне, тому що при збірці ми не можемо використовувати подібні операції. Тому використовуємо визначену функцію для переміщення елементів головоломки. За одну зміну положення ми вибіраємо сусідню плажку до пустої та міняємо їх місцями. Таких перетворень кожен раз буде 100 разів.

|  |  |  |  |
| --- | --- | --- | --- |
| 15 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 13 | 14 | 1 |  |

«Рисунок 2.3.1 - паритет»

### **2.4 Алгоритм перевірки на зібраність головоломки**

Головоломка рахується зібраною, коли усі плажки стоять по порядку та пуста займає останнє місце у масиві. Алгоритм цього достаньо простий. Ми проходимось циклом з кроком 1, перевіряючи чи конткретний елемент є меншим ніж індекс наступного. Якщо хоч раз така операція видасть негативний результат, то цикл зупинеться та видасть, що головоломка не є зібраною.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 13 | 14 | 15 |  |

«Рисунок 2.4.1 – зібрана головоломка»

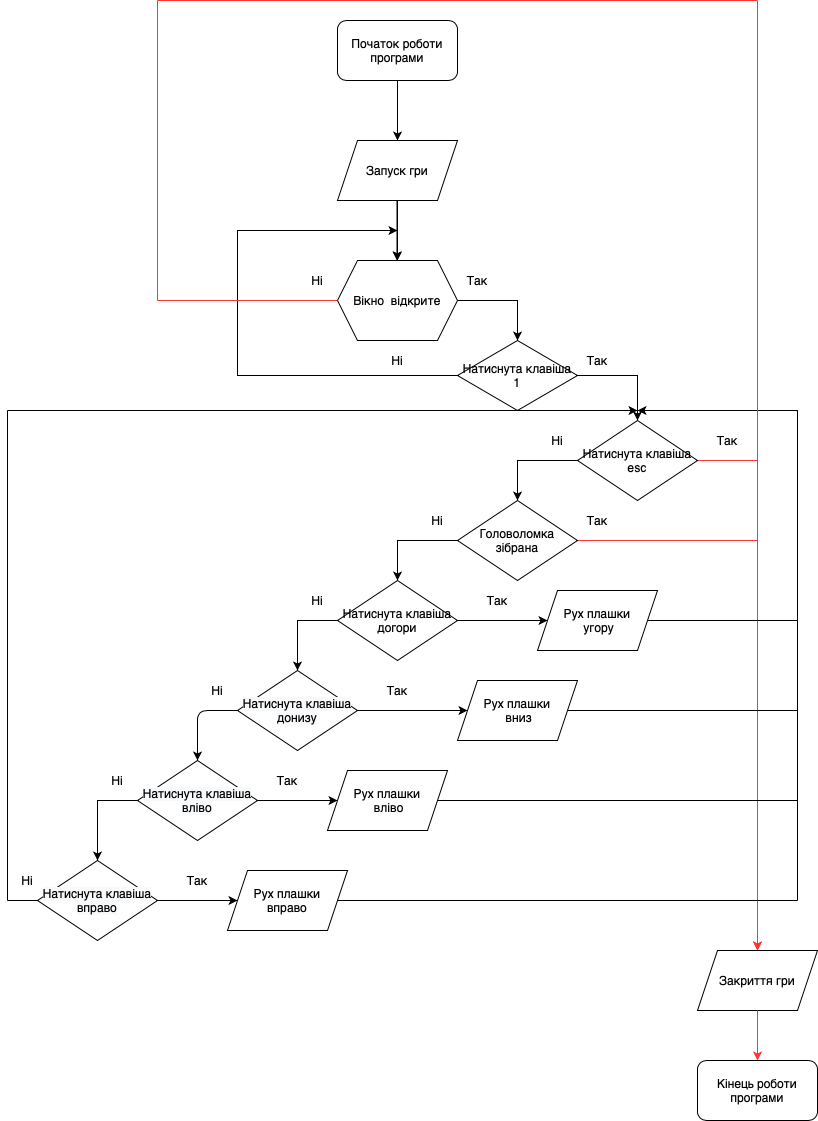
РОЗДІЛ 3

ОПИС РОЗРОБЛЕНОГО ПРОГРАМНОГО ПРОДУКТУ

3.1 Опис структури програми

Розроблена консольна гра включає в себе як і саму гру, так і створене меню, що дозволяє користувачеві вибирати дії. Управління гри складається повністю з 6 клавіш. Клавіші: 1, esc, стрілка догори, стрілка донизу, стрілка вліво, стрілка вправо. Розглянемо варіанти розвитку дій при управлінні в частині гри “Меню”. Якщо натиснути клавішу 1, то гра запуститься, інша операція – вихід з гри при натисненні клавіші esc. Розглянемо варіанти розвитку дій при управлінні в частині гри “Гра”. Управління плашками складається з клавіш: стрілка догори, стрілка донизу, стрілка вліво, стрілка вправо. При натисненні однієї з цих клавіш, програма сама визначає відповідну плашку, що потрібно рухати на місце пустої. Гра триває доти, доки користувач не збере повністю головоломку. При закінченої збірці, користувача виводить на головне меню та далі він вибирає дії, описані вище.

Наведемо графічну схему розвитку подій.



«Рисунок 3.1.1 - блоксхема»

### **3.2 Опис основних функцій**

Дана задача не є лінійною, тому вимагає розбиття на підзадачі. Для вирішення кожної конкретної підзадачі було створено окрему функцію.

Перша функція: Init().

Ця функція викорисовується для заповнення плашок змінними типу int. Вона проходиться ціклом по подновимінорному масиву elements[n\*n] та за допомогою цілого лічильника int і заповнює плажки відповідними значеннями. Останній елемент – пуста плажка. Йому присвоюється значення нуль. У кінці роботи функції, змінні solved присвоюється значення true, оскільки це правильно, та це початкове значення булевої змінної, що буде використовуватись пізніше. Передача параметрів не відбувається, адже ця функія входить до класу.

Друга функція: Check().

Ця функція використовується для перевірки головломки на зібраність. Це булева функція, тому вона повертає значення true або false. Змінна solved це значення данної булевої функції. Данна функція проходиться циклом спочатку та перевіряє чи кожен елемент більше нуля та чи кожен наступний більше ніж попередній. Таким чином, нам не потрібно визначати місцезнаходження пустої, плашки, ажде якщо при проходженні циклу, функція ні разу не видала значення false, тоді пуста плажка займає останню позицію та головоломка рахується зібраною. Передача параметрів не відбувається, адже ця функія входить до класу.

Третя функція: Move(command).

У функції main, ми використовуємо масив, що складається з 4 літер, як алфавіт для кодування натиску клавіш вверх, вниз, вліво, вправо. Тому при викликанні цієї функції ми передаємо туди команду. Спочатку ми визначаємо положення плажки, будь то двовимірний масив. Наступним кроком ми ідентифікуємо команду та в залежності від її значення, ми визначаємо move\_index. Це індекс сусідньої плашки до пустої. Наступним кроком ми міняємо місцями плажки, використовуючи додаткову змінну temp. Останнім кроком, ми перевіряємо, чи є головоломка зібрана, викликаючи функцію check та присвоюємо змінній solved її значення.

Четверта функція: draw(sf::RenderTarget& target, sf::RenderStates states).

Данна функція відповідає за візуальне оформлення програми. Параметри, що ми туди передаєм – це параметри нашого данного поля, а саме класа Game. Спочатку ми ініціалізуємо кольори за допомогою rgb палітри. Наступним кроком ми позицтонуєм наше поле за допомогою вектора. Розставляємо плашки, визначаємо відстань між ними. Товщинк пропусків, стінок і тд. Далі ми зафарбовуємо елементи нашої гри. Це плашки, та задній фон. Наступним кроком є визначення, чи потрібно малювати меню. Тут використовуємо значення змінної solved та відповідно до нього створюємо меню. Підгружаємо шрифт. Визначаємо розмір, колір та позицію тексту. Далі виходимо з умови та працюємо з плашками, що більше нуля. Ту, що нуль не чіпаємо. Визначаємо позицію кожної плашки та малюємо її. Поверх неї малюємо цифру, що відповідає кожній з плашок.

П’ята функція: main().

Спочатку ми створюємо вікно фіксованого розміру, а саме 1000 на 1000. Далі ми встановлюємо ліміт частоти кадрів у секунду – 60. Підгружаємо шрифти та встановлюємо наше ігрове поле у вікно. Для перемішування ми використовуємо цілу змінну move\_times. Далі ми відкриваємо наше вікно. Якщо натиснута клавіша esc, тоді воно закрівається. Якщо ж ми хочемо продовжити роботу, то у цьому циклі while маємо умову, що якщо ще не зібрана, то кодуємо команду для передачі у функцію move. Якщо вона вже зібрана та натиснута кнопка запуску гри, то викликаємо ф. Init() та присвоюємо значенні змінній move\_times 100. Далі потрібно перемішати. Для цього ми створили додатковий масив ar, що складається з символів, які відповідають натискам клавіш. За один прохід ми вибираємо один з символів та, зменшуємо наш лічільник на 1 та міняємо місцями плашки.

### **3.3 Опис інтерфейсу**

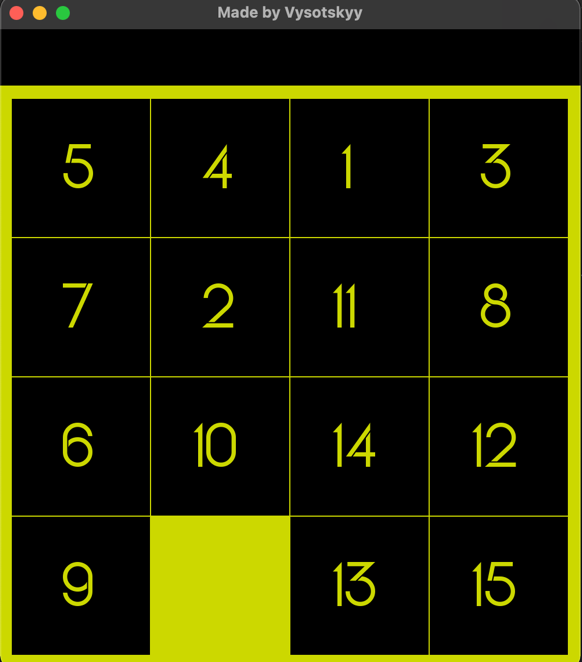
При запуску програми, відкривається меню, що зображено на 2 рисунку. Наступним кроком користувач має натиснути клавішу, відповідно до свого бажання. При натисканні, гра запускається. Скріншот зображено на рис, 12. Користувач баче, як поле перемішується. Через деякий час гра перестає перемішуватись та користувач починає грати, натискаючі клавіши управління.

Одразу як користувач зібрав головломку, висвітлються меню та алгоритм користування повторюється.

Скріншоти обох частин інтерфейсу:



«Рисунок 3.3.1 – головне меню»



«Рисунок 3.3.2 - гра»

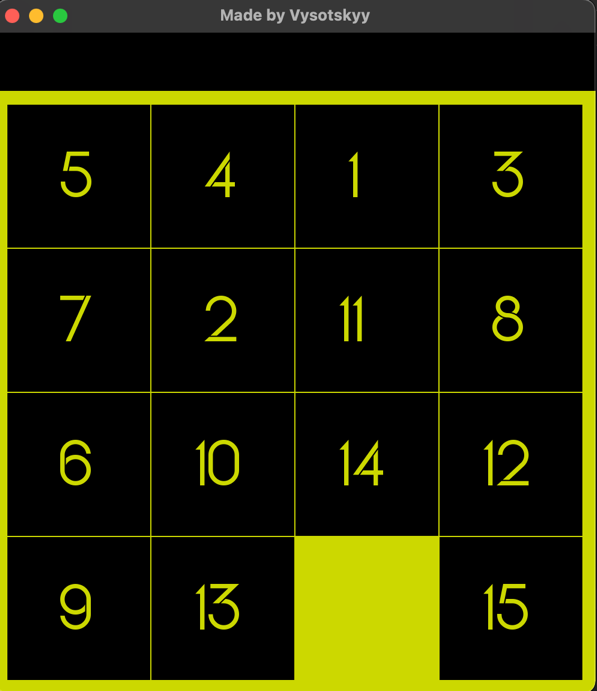
### **3.4 Перевірка роботи програми**

Крок 1, запустимо програму:

****

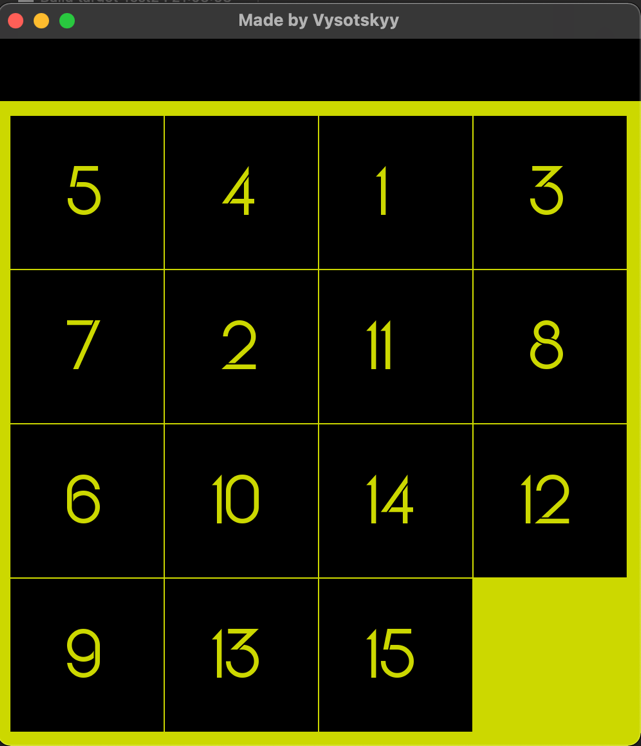
«Рис. 3.3.3 – запуск гри»

Крок 2, почнемо гру, натиснемо 1:



«Рисунок 3.3.4 – початок гри»

Крок 3, зробимо перше переміщення:



«Рисунок 3.3.5 – перше переміщення»

Крок 4, зберемо головоломку та повернемось на головне меню:

****

«Рисунок 3.3.6 – головне меню після збірки»

Крок 5, натиснемо клавішу esc та завершимо роботу.

# Висновки

Під час виконання курсової роботи було розроблено та реалізовано алгоритм гри Пятнашки.

У першому розділі виконано огляд інформаційних джерел з досліджуваної тематики, проблем та існуючих методів розробки. *Об*ґрунтовано вибір мови та середовища програмування*,* а також сформульовано уточнену постановку задачі.

У другому розділі досліджено методи вирішення задачі проектування, розглянуто і побудовано алгоритми для використання одновимірного масиву, хоча задача передбачає двовимірний, пояснено його перевагу, переміщення плашок, визначення чи головоломка зібрана, пояснено їхні особливості.

У третьому розділі представлено опис програми, окремих її функцій та змінних у вигляді таблиць. Також розроблено інструкцію користувача, в якій викладено проілюстровані рисунками основні вказівки, необхідні для роботи з програмою. Коректність роботи програми перевірено на тестуваннях.

У результаті роботи отримано програмний продукт, що відповідає технічному завданню курсової роботи. У подальшому розроблене програмне забезпечення можливе бути удосконалене за рахунок покращення інтерфейсу користувача, додавання різних рівнів складності та можливе збереження прогресу.

Отже, розроблене програмне забезпечення працює коректно, воно є зручним і зрозумілим для користувача. Всі функції, що були передбачені технічним завданням, реалізовані. Всі умови технічного завдання виконані.

# Список використаних джерел

|  |  |
| --- | --- |
| Електронні видання | |
| Програмне забезпечення sfml для створення графіки. Алгоритм підключення бібліотек до програми на операційній системі Mac OS, Xcode  <https://www.sfml-dev.org> | Programne zabezpechennia sfml dlia stvorennia grafiku. Alorythm pidkliuchennia bibliotek do programu na operaciyniy sustemi Mac OS, Xcode  <https://www.sfml-dev.org> |
| Приклад створення пятнашок як програму для персонального компютера. Алгоритм реалізації плашок та анімації переміщення.  [www.tproger.ru](http://www.tproger.ru) | Priklad stvorennia piatnashok yak programu dlia personalnogo kompiutera. Algoritm realizacii plashok ta peremishchennia.  [www.tproger.ru](http://www.tproger.ru) |
| Програмне забезпечення qt для створення графіки. Алгоритм підключення бібліотек до програми на операційній системі Mac OS, Xcode  <https://www.qt.io> | Programne zabezpechennia qt dlia stvorennia grafiku. Alorythm pidkliuchennia bibliotek do programu na operaciyniy sustemi Mac OS, Xcode  <https://www.qt.io> |
| Механізм роботи алгоритму “П'ятна́шки”, історія та математична теорія алгоритму.  <https://uk.wikipedia.org/wiki/П%27ятнашки> | Mekhanizm roboty algorytmu “Piatnashku”, istoria ta matematuchna teoriya algorutmu.  <https://uk.wikipedia.org/wiki/П%27ятнашки> |
| Онлайн модель робочої гри  <https://metaschool.ru/pub/games/puzzle15/puzzle15.php> | Online model robochoi gru  <https://metaschool.ru/pub/games/puzzle15/puzzle15.php> |
| Програмне забезпечення cairo для створення графіки. Алгоритм підключення до програми на операційній системі Mac OS, Xcode <https://www.cairographics.org/cairomm/> | Programne zabezpechennia cairo dlia stvorennia grafiku. Alorythm pidkliuchennia bibliotek do programu na operaciyniy sustemi Mac OS, Xcode  <https://www.cairographics.org/cairomm/> |
| Робота з вікнами, відкриття, закриття і тд.  https://progressor-blog.ru/sfml/otkrytie-i-upravlenie-oknom-sfml/ | Robota z viknamu, vidkruttia, zakruttia I td.  https://progressor-blog.ru/sfml/otkrytie-i-upravlenie-oknom-sfml/ |

# ДОДАТКИ

Код програми:

#include <SFML/Graphics.hpp>

#define n 4

#define size n\*n

#define window\_size 1000

#define square 240

#include "ResourcePath.hpp"

**const** **int** cof = 2;

**class** Game : **public** sf::Drawable, **public** sf::Transformable

{

**public**:

**int** elements[size];

**int** empty\_index;

**bool** solved;

sf::Font font;

Game();

**void** Init();

**bool** Check();

**void** Move(**char** command);

**virtual** **void** draw(sf::RenderTarget& target, sf::RenderStates states) **const**;

};

Game::Game()

{

font.loadFromFile(resourcePath() + "Maxellight.ttf");

Init();

}

**void** Game::Init()

{

**for** (**int** i = 0; i < size - 1; i++)

{

elements[i] = i + 1; // writing numbers in array

}

empty\_index = size - 1;

elements[empty\_index] = 0; // last number is zero

solved = **true**; // fixing that it is solved

}

**bool** Game::Check()

{

**for** ( **int** i = 0; i <size; i++)

{

**if** (elements[i] > 0 && elements[i] != i + 1)

{

**return** **false**;

}

}

**return** **true**;

}

**void** Game::Move(**char** command)

{

// initalizing empty position

**int** col = empty\_index % n;

**int** row = empty\_index / n;

//searching neighbour position due to command

**int** move\_index = -1;

**if** ((command == 'l') && (col < (n - 1)))

{

move\_index = empty\_index + 1;

}

**if** ((command == 'r') && (col > 0))

{

move\_index = empty\_index - 1;

}

**if** ((command == 'u') && (row < (n - 1)))

{

move\_index = empty\_index + n;

}

**if** ((command == 'd') && (row > 0))

{

move\_index = empty\_index - n;

}

// swapping

**if** ((empty\_index >= 0) && (move\_index >= 0))

{

**int** temp = elements[empty\_index];

elements[empty\_index] = elements[move\_index];

elements[move\_index] = temp;

empty\_index = move\_index;

}

solved = Check();

}

**void** Game::draw(sf::RenderTarget& target, sf::RenderStates states) **const**

{

states.transform \*= getTransform();

sf::Color color = sf::Color(200, 100, 200);

sf::Color color\_golf = sf::Color(204, 214, 19); // setting gold color

sf::RectangleShape shape(sf::Vector2f(window\_size, window\_size)); // creating a field

shape.setOutlineThickness(3.f);

shape.setOutlineColor(color\_golf);

shape.setFillColor(color\_golf);

target.draw(shape, states);

shape.setSize(sf::Vector2f(square - 2, square - 2));

shape.setOutlineThickness(2.f);

shape.setOutlineColor(color\_golf);

shape.setFillColor(sf::Color::Black);

sf::Text text\_pl("", font, 52 \* cof);

**for** ( **int** i = 0; i < size; i++)

{

shape.setOutlineColor(color\_golf);

text\_pl.setFillColor(color\_golf);

text\_pl.setString(std::to\_string(elements[i]));

**if** (solved)

{

target.clear();

sf::Text labelMenu("Menu", font, 20 \* cof);

labelMenu.setFillColor(color\_golf);

labelMenu.setPosition(250.f \*cof - 50, 170 \* cof);

target.draw(labelMenu, states);

sf::Text descMenu("Start new game - 1 | Exit - Esc ", font, 20 \* cof);

descMenu.setFillColor(color\_golf);

descMenu.setPosition(250.f, 200.f \* cof);

target.draw(descMenu, states);

shape.setOutlineColor(color\_golf);

shape.setFillColor(color\_golf);

text\_pl.setFillColor(color\_golf);

}

**if** (elements[i] > 0)

{

sf::Vector2f position(i % n \* square + 10.f \* cof, i / n \* square + 10.f \* cof);

shape.setPosition(position);

text\_pl.setPosition(position.x + 35.f \* cof + (elements[i] < 10 ? 15.f : 0.f), position.y + 25.f \* cof);

target.draw(shape, states);

target.draw(text\_pl, states);

}

}

}

**int** main()

{

sf::RenderWindow window(sf::VideoMode(500 \* cof, 550 \* cof), "Made by Vysotskyy");

window.setFramerateLimit(60);

sf::Font font;

**if**(!font.loadFromFile(resourcePath() + "Maxellight.ttf")){ // connecting a font

**return** 0;

}

Game game; // creating a game

game.setPosition(0,100);

sf::Event event;

**int** move\_times = 0; //counter for mixing it up

**while** (window.isOpen())

{

**while** (window.pollEvent(event))

{

**if** (event.type == sf::Event::Closed) window.close();

**if** (event.type == sf::Event::KeyPressed)

{

**if** (event.key.code == sf::Keyboard::Escape)

{

window.close();

}

**if**(!game.solved){ //you can't press one of those keys if the game wasn't started

**if** (event.key.code == sf::Keyboard::Left)

{

game.Move('l');

}

**if** (event.key.code == sf::Keyboard::Right)

{

game.Move('r');

}

**if** (event.key.code == sf::Keyboard::Up)

{

game.Move('u');

}

**if** (event.key.code == sf::Keyboard::Down)

{

game.Move('d');

}

}

// starting game

**if** (event.key.code == sf::Keyboard::Num1)

{

game.Init();

move\_times = 100; // 100 times of swapping

}

}

}

**char** ar[4] = {'l','r','u','d'}; // array of commands

**if** (move\_times-- > 0)

{

(game.Move(ar[rand()%4])); //getting rand of commands

}

window.clear();

window.draw(game);

window.display();

}

**return** 0;

}