Курсова робота

з дисципліни «Алгоритмізація та програмування»

на тему: Ігрові програми на випадковий рух, гра «Іподром»

Виконав: студент групи ІС-91

Коган А.М.

Прийняв: Просянкіна-Жарова Т.І.

Засвідчую, що у цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис)

Зміст

[ВСТУП 3](#_Toc42684811)

[1. РОЗДІЛ ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ ІГРОВОЇ ПРОГРАМИ НА ВИПАДКОВИЙ РУХ ОБЪЕКТА «ІПОДРОМ». 4](#_Toc42684812)

[1.1. Опис гри «Іподром». 4](#_Toc42684813)

[1.2. Вибір засобів розробки програми. 5](#_Toc42684814)

[1.3. Постановка задачі проектування. 6](#_Toc42684815)

[2. РОЗДІЛ ОБГРУНТУВАННЯ МЕТОДІВ РОЗВ’ЯЗКУ ЗАДАЧІ ТА РОЗРОБКА АЛГОРИТМУ. 8](#_Toc42684816)

[2.1. Опис методу розв’язку. 8](#_Toc42684817)

[2.2. Алгоритм розв’язку задачі. 9](#_Toc42684818)

[2.3. Графічне зображення алгоритму. 10](#_Toc42684819)

[3. ОПИС РОЗРОБЛЕНОЇ ПРОГРАМИ «ІПОДРОМ». 11](#_Toc42684820)

[3.1. Опис головних структур та змінних програми. 11](#_Toc42684821)

[3.2.Опис графічного інтерфейсу програми. 15](#_Toc42684822)

[3.3 Результати виконання програми. 17](#_Toc42684823)

[ВИСНОВКИ. 19](#_Toc42684824)

[Перелік використаних джерел. 20](#_Toc42684825)

[Додаток А. Код програми с++. 22](#_Toc42684826)

# ВСТУП

Ще в 1960 х роках, коли поряд з першими комп’ютерами IBM використовували телетайп, широке розповсюдження взяли ігри що базуються на символьному кодуванні (як ASCII)(1). Найбільш популярні стали такі ігри у 1970-1980-х роках, коли потужність персональних комп'ютерів ще не дозволяла обробляти складнішу графіку. Пізніше на початку 1990 текстові ігри перейшли в online.(2) Недавні дослідження показують вони і досі популярні серед користувачів ПК.(3) Тож тему дослідження можна вважати актуальною.

Мета курсової роботи - розробити та реалізувати закінчений програмний продукт для вдосконалення вивченого матеріалу з алгоритмізації та програмування. Тому обраний графічний інтерфейс відповідає завданню курсової роботи.

Завдання курсової роботи - на основі аналізу літературних джерел оцінити існуючі методи та алгоритми реалізації складових програмного продукту, скласти структурний алгоритм програми та реалізувати його з використанням процедурного програмування у вигляді гри “Іподром”.   
Практичним значенням цієї роботи є вдосконалення знання мови с++, яка як засвідчує статистика GitHub стабільно займає місце серед найбільш використовуваних мов на платформі протягом року. (4)

У результаті виконання курсового проекту було розроблено задане програмне забезпечення, у якому у консольному режимі MS Windows реалізовано графічний інтерфейс користувача з можливістю зробити “ставку” на карету переможця, та спостерігати за процесом так званих “гонок” об’єктів.

Пояснювальна записка курсової роботи складається з вступу, висновків та трьох розділів. У першому розділі виконана постановка задачі з розробки гри, у другому обґрунтування застосованих алгоритмів. У третьому - опис завершеної програми.

# РОЗДІЛ 1

# ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ ІГРОВОЇ ПРОГРАМИ НА ВИПАДКОВИЙ РУХ ОБЪЕКТА «ІПОДРОМ»

## 

## 1.1. Опис гри «Іподром»

Програма реалізує одночасний рух декількох об'єктів з випадковою швидкістю. У нашому разі, для спрощення ~~ми~~ визначимо 8 об'єктів  цифрами, які будуть їх ідентифікувати. Цей об'єкт будемо іменувати каретою (Carriage).

У кожній карети є швидкість яка задається при старті програми. Та може змінитися, якщо на шляху цієї карети зустрінеться перешкода. Перешкоду ~~ми~~ визначимо через зірку (\*). У разі зіткнення швидкість карети буде зменшена. Графа "score" набуває значення від 0 до 100 і відображає процент пройденого шляху.

Користувач повинен обрати через меню старт гри, потім зробити ставку на карету. Оскільки гра називається “Іподром” рух карет буде по колу.

Ігрове поле сформоване за допомогою  двомірного масиву символів. Оскільки висота символів не дорівнює їх довженні, формуючи двомірний масив розмірністю (M, N), де M дорівнює N, на екрані отримуємо прямокутник.

Для того, щоб результати гри не повторювались, реалізована генерація випадкових величин (~~дивись п.2.2~~). Швидкість руху об’єктів буде змінена кожну ітерацію циклу у межах від одного до трьох символів за кожен цикл гри.

Правила гри передбачають що кожен об’єкт:

* Буде рухатись по своєму колу.
* Змінює свою швидкість на протязі гри.
* Може втрапити на штрафне поле, у цьому разі швидкість об’єкта буде знижена на одну умовну одиницю.

Користувач перемагає у разі якщо обрана карета першою дійде до фінішу.

Коли переможець прийде до фінішу користувачу буде показано фінальне повідомлення, залежно від результату гри, це буде стилізоване “YOU WIN” або “YOU LOSE”.

Більш ускладнена реалізація винагород може бути розглянута у грі аналозі DOG RACE(5) яка була додана у збірник 101 basic computer games.

«Ми» - застосовувати в тексті не можна. У розділі 1 повинні біти розставлені посилання на використані джерела. Форматування всього тексту записки повинно бути таким, як в цьому підрозділі (див. методичні вказівки)

## 1.2. Обґрунтування вибору засобів розробки програми

Серед розглянутих мов для реалізації програми була BASIC, дуже розповсюджена мова у час становлення ігор з текстовим інтерфейсом. Багато гарних ігор цього жанру вже існують, існує навидь збірник текстових ігор з використанням цієї мови(6). ? Код програм BASIC був використаний для загального розуміння  алгоритмів  у іграх що базуються на текстовому інтерфейсі. Але щоб підвищити практичне значення курсової роботи для реалізації програми була обрана більш розповсюджена мова С++.

Мову С++ використовують для системного програмування, розробки програмного забезпечення, написання драйверів, потужних серверних та клієнтських програм, а також для розробки розважальних програм, наприклад, відеоігор. С++ суттєво вплинула на інші популярні сьогодні мови програмування як C#  та Java.(7)Серед переваг мови С++ требо відокремити те, що мова об’єктно орієнтована. Це дає можливість відокремити змінні, що стосуються об’єкту Carriage. Для спрощення гра реалізована у одному файлі Ipodrom.cpp де знаходяться усі класи та функції.

Мова С має ряд достоїнств, які роблять її універсальним інструментом програміста:

* простота;
* відносно висока швидкодія та відносно малий розмір програм, написаних на С;
* високий рівень мобільності програм, написаних на С
* суміщення можливостей мови низького рівня зі зручністю мови програмування високого рівня. (8)

Все перелічене вище робить мову С зручним засобом для розробки програм різноманітного призначення.

Середовищем розробки обрана Visual Studio Community 2019. яка дозволяє розробляти консольні програми та підтримує мову с++.

~~Для оформлення курсової роботи використано текстовий̆ редактор Microsoft Word 2010~~~~(9), додаток draw.io~~~~(10) для малювання блок-схем алгоритмів. Для пошуку інформації̈ у мережі Інтернет веб-браузер Google Chrome~~~~(11). Також для редагування застосований редактор notepad++~~~~(12). –~~ це пишуть у вступі

## 1.3. Постановка задачі проектування. – зробіть за зразком

Розглянувши опис гри та аналізуючи ігри аналоги (5) ми можемо сформулювати основні етапи розробки:

* Завдання масиву з символів для малювання поля.
* Формування класу Carriage з усіма належними йому змінними.
* Завдання класу об’єкту Carriage з усіма його полями, відповідальними за збереження швидкості, напрямку руху, ідентифікатором, позицію по осям x та y, та прогрес (процент від всього шляху).
* Завдання координат перешкоди.
* Початкові змінні повинні бути ініціалізовані в окремому методі.
* Розробка вступного етапу взаємодії з користувачем реалізована через меню та збір даних у методі input();
* Відокремлення малювання від логіки (функції draw() та logic() у циклі).
* Умовою закінчення гри є зміна прапору gameOver.
* Зміна прапору можлива у методі який відповідає за логіку.

Після закінчення етапу програмування йде етап тестування та випуску готового продукту, до яких належить:

* Перевірка працездатності програми з масштабуванням ігрового поля, завданням тестових позицій та швидкості.
* Компілювання коду у готовий продукт.
* Довід працездатності закінченого програмного продукту.

# РОЗДІЛ 2 ОБГРУНТУВАННЯ МЕТОДІВ РОЗВ’ЯЗКУ ЗАДАЧІ ТА РОЗРОБКА АЛГОРИТМУ

## 2.1. Опис методу розв’язку.

За умовами гри ми повинні мати ігрове поле таке, щоб ми мали можливість показати користувачу перегони кареток кількість яких буде 8, тож  оскільки переміщення буде по колу, поле повинно бути більшим ніж 16 знаків. Ще дві клітини це поля. Також посеред поля розташовані клітини по яким рух здійснюватись не буде.

Треба зазначити, що оскільки висота консолі може відрізнятись залежно від операційний системи користувача,  бажано використати поле мінімальної розмірності. Але таке, щоб окружність описана об'єктом по колу з мінімальним та максимальним радіусом відрізнялись на так звану фору - перевагу, що заздалегідь надається слабшому супернику.  У нашому разі слабкіший елемент гри, який розташований на колі з максимальним радіусом.

На легкоатлетичному стадіоні з різниця між доріжками нівелюється через спеціальні правила розмітки. При тому, що довжина одного кола на стадіоні може бути порядку 300-400 метрів, а ширина доріжки 1.25(13) метрів - різниця не така велика, і зображення фори цілком виправдано.

Для зображення графічних об'єктів ми використовуємо двомірний масив знаків, які будуть розташовані за осями x та y.

У грі Іподром фора не буде зазначена, для спрощення. Максимальний шлях який подолає кожен об'єкт буде дорівнювати довжині найвіддаленіший від центру стадіону доріжці. Я обрав квадратне поле з довжиною сторони 40 символів, це буде достатньо для відображення інтерфейсу та доведення працездатності гри, за необхідністю ігрове поле може бути поширене.

Довжина максимального шляху дорівнює 38\*4 + 4, де 38 знаків у внутрішньому колі, на кожну сторону (4 сторони) та 4 знаків завдяки яким коло розширюється, усього 156 знаків. Відносно цього шляху буде розрахований прогрес для кожного об'єкту Carriage та рівно стільки він повинен “проїхати”.

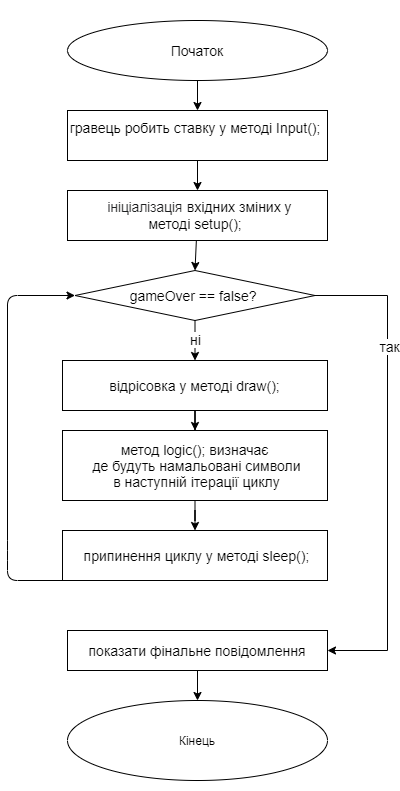
   Також  ми можемо визначити систему винагород за переваги на гонках означеного об’єкту Carriage  - щоб не ускладнювати алгоритм - виведемо текстову картинку "You win" або "You lose".

## 2.2. Алгоритм розв’язку задачі.

Зображення руху об’єктів буде реалізовано наступним чином:

* Позначимо вхідні дані - розмір поля, змінні x, y, для перешкоди, score - для визначення подоланого шляху обраного об’єкту. GameOver
* Визначимо об’єкт Carriage та його поля, зв’язаний список об’єктів Carriage тому що буде зручно оперувати з ним у циклі, у методі draw.
* У методі main можемо розділили нашу програму на головні методи, input, setup, draw, logic.
* У методі input гравець обиратиме який об’єкт, за його думкою, прийде першим. Це реалізовано за допомогою меню.
* У методі setup будуть ініціалізовані змінні з пункту 1 цього алгоритму.
* Методи draw та logic будуть у циклі, умова виходу з нього це коли GameOver == true. draw відповідає за малювання знаків, logic за то як вони будуть розташовані.
* Також для того, щоб уповільнити темп гри, кожен цикл ми визиваємо функцію sleep - яка призупиняє виконання поточного потоку, поки не закінчиться інтервал очікування.(14)
* Залежно від результату гри показати повідомлення користувачу.

## 2.3. Графічне зображення алгоритму.

Узагальнений алгоритм роботи гри зображений у наступній блок-схемі (малюнок 1).

Малюнок 1 Блок схема алгоритму.

# 3. ОПИС РОЗРОБЛЕНОЇ ПРОГРАМИ «ІПОДРОМ».

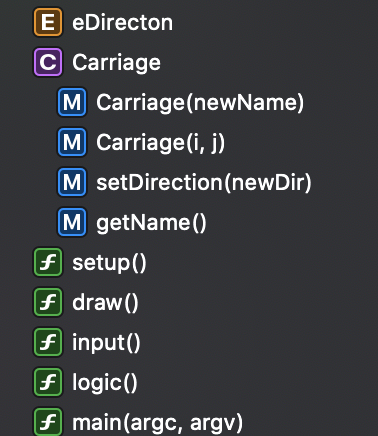
## 3.1. Опис головних структур та змінних програми.

Перш за все треба почати з розуміння реалізації  видимості змінних у с++.

У с++ Кожна змінна має свою зону видимості, тобто таку область, в якій можна працювати зі змінною. За межами цієї області, про даної змінної нічого відомо не з буде, а значить і використовувати її не можна. Отже, змінна знаходиться в області видимості, якщо до неї можна отримати доступ.

Існують локальні і глобальні змінні. Так ось, змінні, оголошені усередині функції, називаються локальними. Локальні змінні мають свої області видимості, цими областями є функції, в яких оголошені змінні. Таким чином, в різних функціях можна використовувати змінні з однаковими іменами, що в свою чергу дуже зручно. Поділ змінних на глобальні та локальні відповідає одному з головних правил програмування, а саме - принципом найменших привілеїв. Тобто, змінні, оголошені всередині однієї функції, повинні бути доступні тільки для цієї функції і нічого іншого, в кінці кінців, вони створювалися саме для цієї функції. Глобальні змінні оголошуються поза тілом якої-небудь функції, і тому область видимості таких змінних поширюється на всю програму. Зазвичай глобальні змінні оголошуються перед головною функцією, можна оголошувати і після функції main(), але тоді дана змінна не буде доступна в функції main().(15)

За допомогою IDE у даному разі XCode(16) зробимо знімок з вікна структури класу main.cpp у якому реалізована програма. (мал. 2).



Малюнок 2. Опис змінних, фрагмент інтерфейсу XCode.

За цими даними складемо таблиці у яких буде описані методи та змінні.

|  |  |  |
| --- | --- | --- |
| Назва змінної | Тип даних | Опис |
| gameOver | bool | Прапор за яким гра буде закінчена. |
| width | int |  |
| height | int | Висота поля у знаках. |
| trapX | int | Координати пастки за віссю X |
| trapY | int | Координати пастки за віссю Y |

Таблиця 1. Опис змінних.

|  |  |  |
| --- | --- | --- |
| Назва змінної | Тип даних | Опис |
| score | int | Скільки пройшла обрана карета. |
| bitCarriage | int | Ідентифікатор обраної карети. |
| List <Carriage> cars | LinkedList | Структура де зберігаються усі карети. |
| eDirection | enum | { STOP = 0, LEFT, RIGHT, UP, DOWN};  Структура enum для більш інтуїтивного позначення напрямку руху. |

Таблиця 2. Опис змінних, продовження.

Також окремо опишемо змінні класу Carriage.

|  |  |  |
| --- | --- | --- |
| Назва змінної | Тип даних | Опис |
| x | int | Координата x цієї карети. |
| y | int | Координата y цієї карети. |
| progress | int | Скільки пройшла ця карета від всього шляху. |
| name | int | Ім’я карети - для спрощення номер. |
| dir | enum eDirecton | { STOP = 0, LEFT, RIGHT, UP, DOWN}; |
| speed | int | Швидкість цієї карети. |

Таблиця 3. Опис змінних класу Carriage.

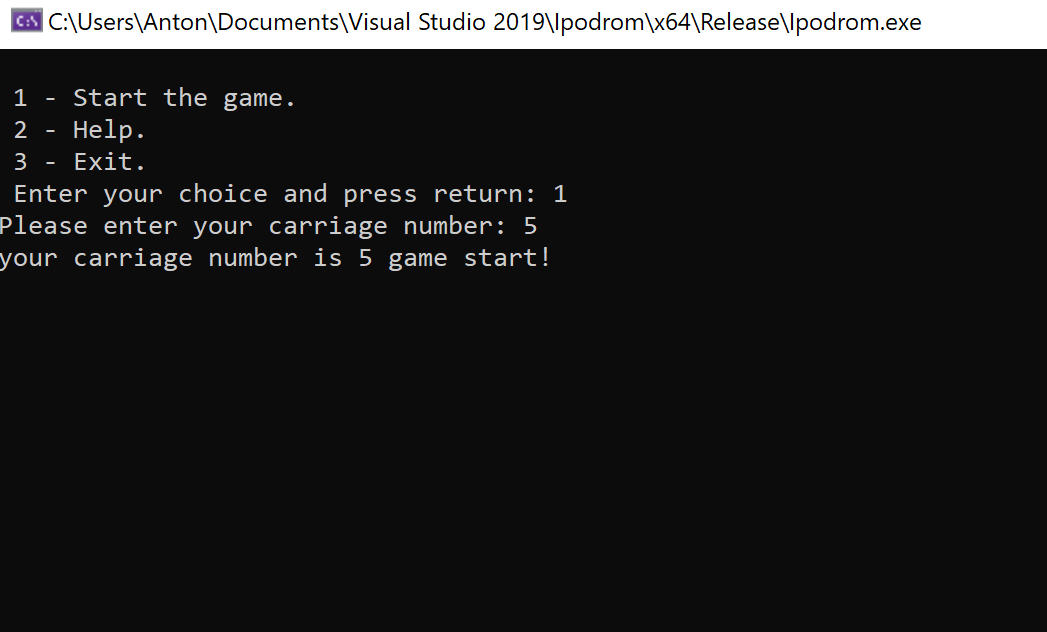
Наступна таблиця пояснює використанні функції.

|  |  |  |
| --- | --- | --- |
| Назва функції / вхідного параметру | Семантика параметрів/тип | Призначення / опис |
| Input | void | Реалізовано меню через яке гравець робить ставку на карету. |
| Setup | void | Ініціалізація початкових змінних. |
| Draw | void | Малювання поля та інших об'єктів у текстовому представленні. |
| Logic | void | Логіка гри, тут позначена майбутня поведінка об’єктів. |
| Carriage   .setDirection |  | Задати напрямок руху |
| параметр   (eDirecton newDir) | enum | Параметр enum STOP, LEFT, RIGHT, UP, DOWN |
| Carriage  .getName | int | повертає значення ідентифікатору об’єкта. |

Таблиця 4. Опис функцій.

## 3.2.Опис графічного інтерфейсу програми.

Взаємодія з користувачем реалізована через меню (мал. 3) далі у консолі він повинен побачити самі перегони (мал. 4)  , де кожен об’єкт буде представлений цифрою від 1 до 8.  Нижче наведені реалізація меню та процесу гри.



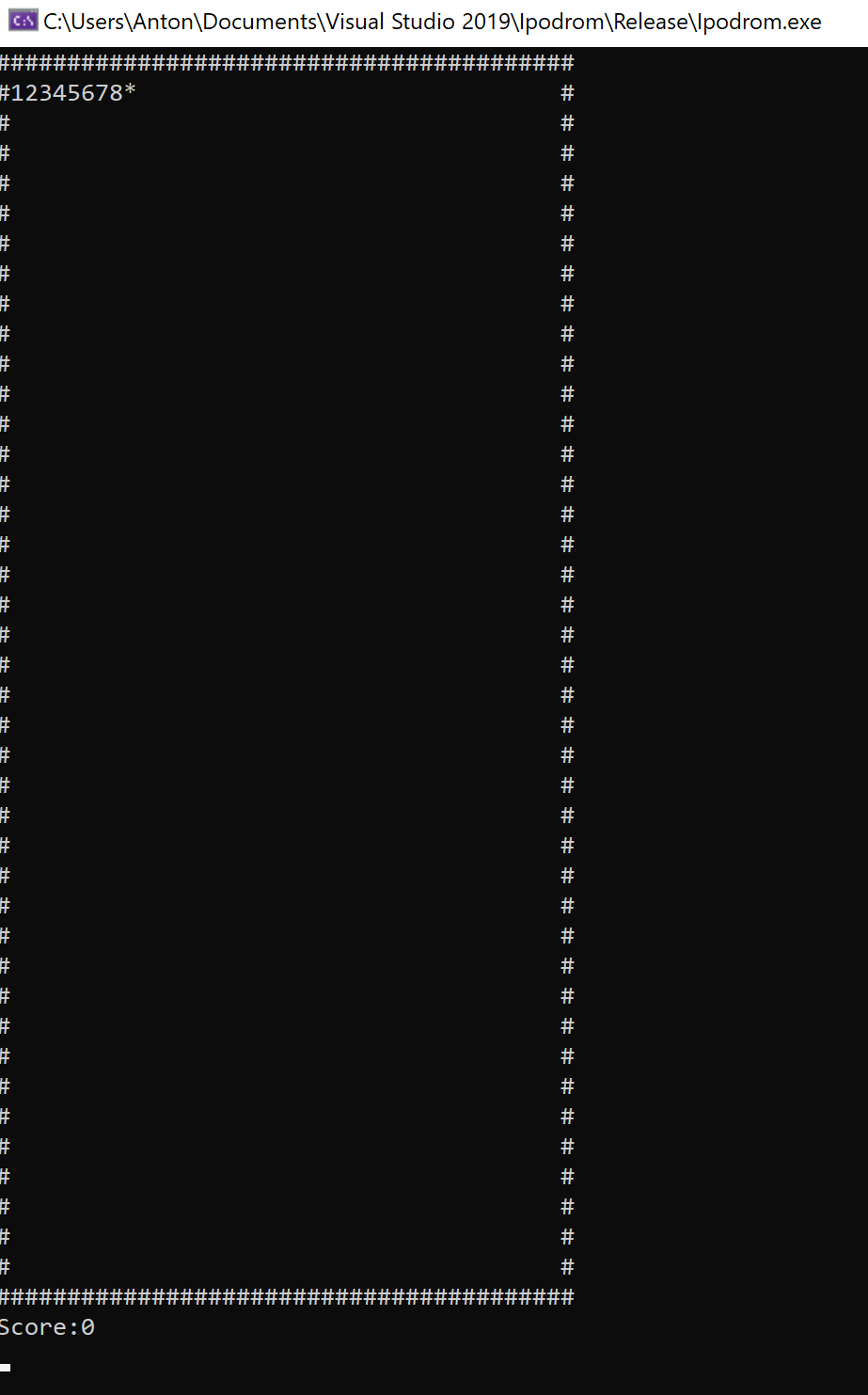
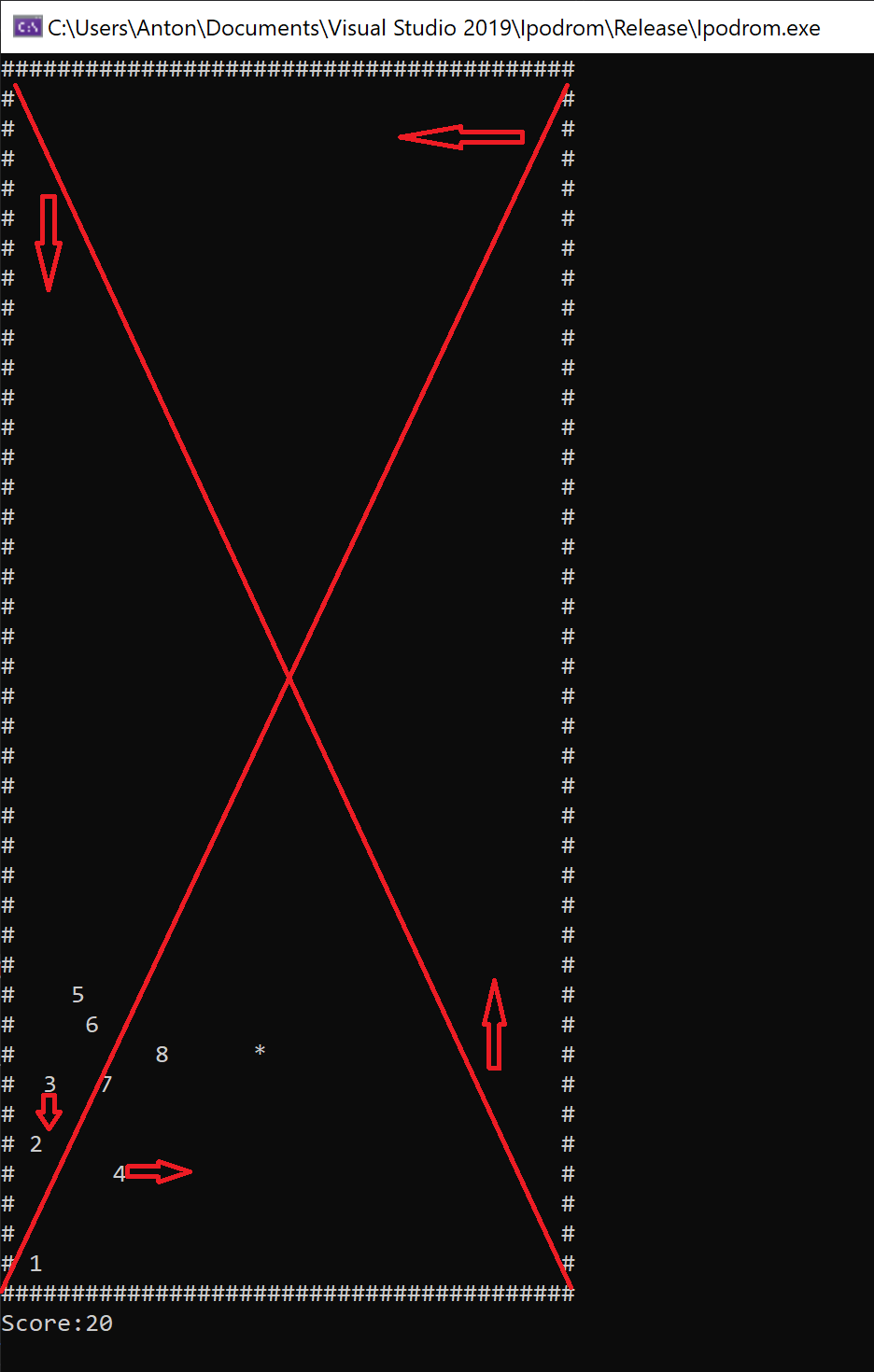
Малюнок 3 Стартове меню.

Гравцю надається вибір з трьох пунктів:

1. Початок гри.
2. Поміч.
3. Вихід.

У другому пункті зазначено що гравець повинен обрати карету та почати гру. Цей пункт може бути поширений, але зараз він відповідає нескладним правилам гри.

Далі, на початку гри, усі об’єкти розташовані в одну стрічку. (мал.4). Напрямок руху змінюється залежно від позиції об’єкту. (мал.5).



Малюнок 4. Стартова позиція перегонів.

Малюнок 5. Зміна напрямку руху.

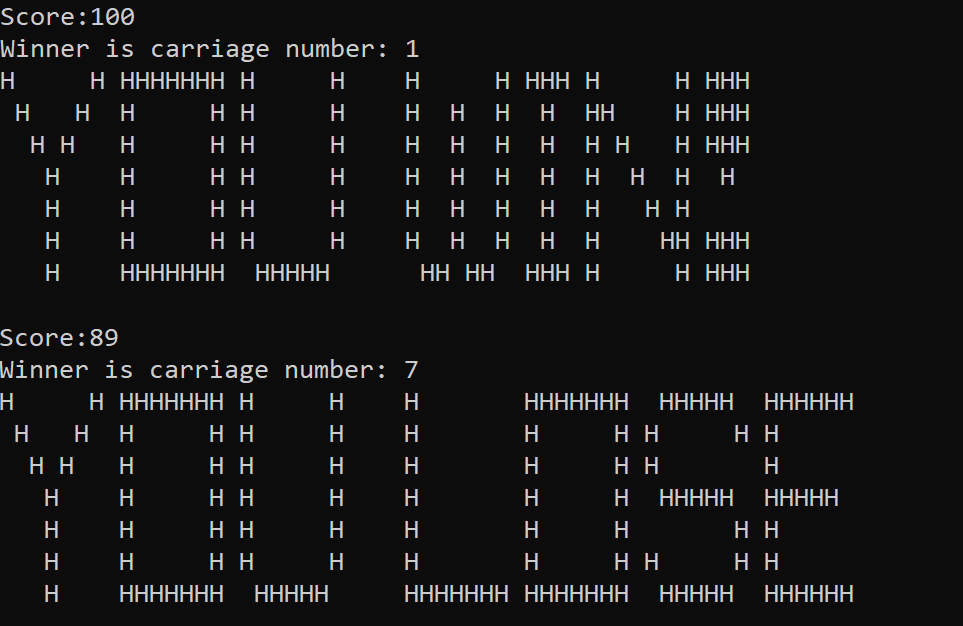
## 3.3 Результати виконання програми.

Після того, як перша карета прийде до фінішу, її номер буде відображено у фінальному повідомленні (мал.6), та гра закінчиться.



Малюнок 6. Перший гравець дійшов до фінішу.

Залежно від результату буде показано фінальне повідомлення. (мал.7).



Малюнок 7. Фінальне повідомлення.

## ВИСНОВКИ.

Основні етапи розробки програмного продукту були виконані та результатом цієї роботи є програмний продукт, а конкретніше гра “ІПОДРОМ”.  Вдалось покращити свої знання у мові c++ та получити досвід програмування у середі Visual Studio також вдалось покращити свої вміння оперувати зі структурами даних.

Спрощення, такі як зображення об’єкта через його ім’я та назва об’єкта цифрою ідентифікатором не сприяли на роботу програми. Концепція ігор на базі тексту була реалізована певним чином. У наступній версії гри кожен об’єкт буде мати  нормальне ім’я. Також буде можливим додатково налаштувати об’єкт вибором відповідного символу.

Після того як я розглянув концепцію текстових ігор та проаналізував їх розвиток, виявились нові шляхи вдосконалення логіки гри. Ставки можливо реалізувати через зберігання стану на протязі серіі ігор. Але було б надлишковим реалізовувати складну логіку для ознайомлення з алгоритмами програмування. Тому вважаю що мета курсової роботи досягнута.

## Перелік використаних джерел.

1. **Tanenbaum, Andrew S.** *Modern Operating Systems.* Pearson : s.n., 2013. p. 29.

2. Wikipedia. *"Text-based Game.".* [Online] [Cited: April 29, 2020.] https://en.wikipedia.org/wiki/Text-based\_game#History.

3. **Pierre Lacombe, Gabriel Feraud, Clement Riviere.** *Text-Based Games and Narrative-Based Video Games.* Pearson : CRC Press; 1 edition (December 23, 2019), 2019. 10.1201/9780367814335-7.

4. The State of the Octoverse. . [З мережі] [Цитовано: 29 April 2020 p.] https://octoverse.github.com/..

5. *101 basic computer games.* Massachusetts 01754 : Digital Equipment Corporation Maynard, 1975.

6. BASIC. *Wikipedia.* [З мережі] [Цитовано: 22 April 2020 p.] https://en.wikipedia.org/wiki/BASIC.

7. C. *Wikipedia.* [З мережі] [Цитовано: 24 April 2020 p.] https://uk.wikipedia.org/wiki/C++.

8. **Д. Д. Татарчук, Ю. В. Діденко.** *ПРОГРАМУВАННЯ МОВАМИ С та С++.* Київ : Д. Д. Татарчук, Ю. В. Діденко, 2012. с. 5.

9. Download Office 2010: Microsoft Office. *Microsoft Office.* [З мережі] [Цитовано: 1 May 2020 p.] https://www.microsoft.com/en-us/microsoft-365/previous-versions/office-2010.

10. Free Flowchart Maker and Diagrams Online. *Diagrams.net .* [З мережі] http://draw.io/.

11. **"Google Chrome - Download the Fast, Secure Browser from Google." Google.x.** Google Chrome - Download the Fast, Secure Browser from Google. *Google.* [З мережі] https://www.google.com/chrome/.

12. Downloads. *notepad++.* [З мережі] [Цитовано: 07 May 2020 p.] https://notepad-plus-plus.org/downloads/v7.8.6/.

13. **ред. Г. И. Кукушкин.** Энциклопедический словарь по физической культуре и спорту. місце видання невідоме : Физкультура и спорт, 1962, том 2, с. 388.

14. Thread.Sleep Method (System.Threading). *Microsoft Docs.* [З мережі] https://docs.microsoft.com/en-us/dotnet/api/system.threading.thread.sleep?view=netcore-3.1.

15. **Cppstudio.com, cppstudio.com/uk/post/415/.** “Локальні і глобальні змінні в С”. *Cppstudio.com.* [З мережі] http://cppstudio.com/uk/post/415/.

16. **Inc, Apple.** Xcode. [З мережі] https://developer.apple.com/xcode/.

17. Visual Studio 2019: Download for Free. [З мережі] [Цитовано: 07 January 2020 p.] https://visualstudio.microsoft.com/vs/.

## Додаток А. Код програми с++.

// Ipodrom.cpp : This file contains the 'main' function. Program execution begins and ends there.

// main.cpp

// Ipodrom

//

// Created by Anton Kogan on 2/17/20.

// Copyright © 2020 Anton Kogan. All rights reserved.

//

#include <iostream>

#include <conio.h>

#include <list>

#include <windows.h>

#include <cstring>

#include <time.h> /\* time for rand \*/

using namespace std;

bool gameOver;

const int width = 40;

const int height = 40;

const int maxProgress = 38 \* 4 + 4; //156 - biggest round

int trapX, trapY, score;

int bitCarriage = -1;

enum eDirecton { STOP = 0, LEFT, RIGHT, UP, DOWN };

class Carriage {

public: int x, y, progress, speed;

int name;

private: eDirecton dir;

public:

Carriage(int newName) {

x = newName; y = 0;

name = newName;

dir = STOP;

progress = 0;

speed = rand() % 2 + 1;

}

Carriage(int i, int j) {

x = i;

y = j;

dir = STOP;

}

void setDirection(eDirecton newDir) {

dir = newDir;

}

eDirecton getDirection() {

return dir;

}

int getName() {

return name;

}

};

list<Carriage> carriages;

void setup()

{

//setting console window size to match game field

HWND console = GetConsoleWindow();

RECT ConsoleRect;

GetWindowRect(console, &ConsoleRect);

MoveWindow(console, ConsoleRect.left, ConsoleRect.top, 800, 800, TRUE);

srand(time(NULL));

for (int i = 1; i < 9; i++) carriages.push\_back(Carriage(i)); // 8 carriages to choose

trapX = 9;

trapY = 0;

score = 0;

}

void draw()

{

system("cls");

for (int w = 0; w < width + 1; w++)

cout << "#";

cout << endl;

for (int h = 0; h < height; h++)

{

for (int w = 0; w < width; w++)

{

bool print = false;

if (w == 0)

{

cout << "#";

print = true;

}

else if (h == trapY && w == trapX) //trap

{

cout << "\*";

print = true;

} else { // or carriage

list<Carriage>::iterator p = carriages.begin();

while (p != carriages.end()) {

if (!print && h == p->y && w == p->x ) {

cout << p->getName();

print = true;

}

p++;

}//or empty space

if(!print) cout << " ";

}

if (w == width - 1)

cout << "#";

}

cout << endl;

}

for (int i = 0; i < width + 1; i++)

cout << "#";

cout << endl;

cout << "Score:" << score << endl;

}

void input()

{

int choice = 0;

do

{

cout << endl

<< " 1 - Start the game.\n"

<< " 2 - Help.\n"

<< " 3 - Exit.\n"

<< " Enter your choice and press return: ";

cin >> choice;

switch (choice)

{

case 1:

cout << "Please enter your carriage number: ";

cin >> bitCarriage;

cout << "your carriage number is " << bitCarriage << " game start!\n";

gameOver = false;

Sleep(1500);

break;

case 2:

cout << "\n first enter 1,\n next step is to choose Carriage number,\n after that game will start\n";

break;

case 3:

gameOver = true;

cout << "End of Program.\n";

return;

default:

cout << "Not a Valid Choice. \n"

<< "Choose again.\n";

break;

}

} while (choice != 1);

}

void logic()

{

bool hasWinner = false;

list<Carriage>::iterator p = carriages.begin();

while (p != carriages.end()) {

if (p->x == trapX && p->y == trapY)//horse trapped

{

cout << "carriage " << p->getName() << " is trapped somewhere; speed -1 \n";

Sleep(5000);

p->speed -= 1;

}

if (p->progress >= maxProgress) {

gameOver = true;

if (hasWinner) cout << "right after winner comes carriage number: " << p->getName() << "\n";

else {

cout << "Winner is carriage number: " << p->getName() << "\n";

hasWinner = true;

if (bitCarriage == p->getName()) {

cout << "H H HHHHHHH H H H H HHH H H HHH \n";

cout << " H H H H H H H H H H HH H HHH \n";

cout << " H H H H H H H H H H H H H HHH \n";

cout << " H H H H H H H H H H H H H \n";

cout << " H H H H H H H H H H H H \n";

cout << " H H H H H H H H H H HH HHH \n";

cout << " H HHHHHHH HHHHH HH HH HHH H H HHH \n";

} else {

cout << "H H HHHHHHH H H H HHHHHHH HHHHH HHHHHH \n";

cout << " H H H H H H H H H H H H \n";

cout << " H H H H H H H H H H H \n";

cout << " H H H H H H H H HHHHH HHHHH \n";

cout << " H H H H H H H H H H \n";

cout << " H H H H H H H H H H H \n";

cout << " H HHHHHHH HHHHH HHHHHHH HHHHHHH HHHHH HHHHHH \n";

}

}

}

if (p->getName() == bitCarriage) {

score = p->progress \* 100 / maxProgress;//percentage from all way

}

p->speed += (rand() % 2 - 1);// speed change randomly every step, that will average speed difference

if (p->speed < 0) p->speed = 1; // to prevent moving backwards

p->progress += p->speed;

if (p->getDirection() == STOP) p->setDirection(DOWN);

//here is logic to put carriages on their tails, based on index that is name here. For simplicity

if (p->y >= height - p->getName()) {p->setDirection(RIGHT);}

if (p->x >= width - p->getName()) {p->setDirection(UP);}

if (p->x > 8 && p->y < p->getName()) p-> setDirection(LEFT);

if (p->y < 8 && p->x < p->getName() + 1) p->setDirection(DOWN);

//new trap is placed only on the way

if(p->name == 8){

if (p->getDirection() == DOWN) {

trapX = (rand() % 8);

trapY = rand() % 40;

}

else if (p->getDirection() == RIGHT) {

trapX = (rand() % 40);

trapY = rand() % 8 + 32;

}

else if (p->getDirection() == UP) {

trapX = (rand() % 8 + 32);

trapY = rand() % 40;

}

else if (p->getDirection() == LEFT) {

trapX = (rand() % 40);

trapY = rand() % 8;

}

}

if (p->getDirection() == DOWN) {

p->y = p->y + p->speed;

}

else if (p->getDirection() == RIGHT) {

p->x = p->x + p->speed;

}

else if (p->getDirection() == UP) {

p->y = p->y - p->speed;

}

else if (p->getDirection() == LEFT) {

p->x = p->x - p->speed;

}

p++;

}

}

int main(int argc, const char\* argv[])

{

input();

setup();

while (!gameOver)

{

draw();

logic();

Sleep(5);

}

return 0;

}