Міністерство освіти і науки України

Національний технічний університет

«Харківський політехнічний інститут»

кафедра “Інформатика та інтелектуальна власність”

**КУРСОВА РОБОТА**

з дисципліни “Об’єктно – орієнтоване програмування”

**Тема**

“ Гра в Шибинецю”

Виконавець

студент групи КН-321а Денис БОРОДАЙ

Керівник курсової роботи

ст. викладач кафедри ІІВ Андрій ІВАШКО

Харків 2022

# ЗАВДАННЯ

Виконати завдання згідно двадцять шостого варіанту на тему «гра в Шибенецю». Програма загадує слово, а користувач має його відгадати. Гравцеві надається до семи спроб.

Використати засвоєнні навички та знання з курсу. Модифікувати код своєї роботи за допомогою об’єктно – орієнтованго програмування.

Інтерфейс програми має містити в собі такі елементи:

1. меню гри;
2. вибір режиму для одного гравця;
3. вибір режиму для двох гравців;
4. розділ ознайомлення з правилами гри;
5. розділ ознайомлення з застосунком;
6. закінчення роботи застосунку.
7. поле для малюнку персонажа;
8. поле виводу;
9. поле вводу;
10. поле використаних літер.

# РЕФЕРАТ

Пояснювальна записка містить: 73 стор., 22 рис., 6 додатків, 4 джерела.

Предмет дослідження – сучасні методи розробки програм таких, як структурне, прикладне та об’ктно – орієнтоване програмування та візуальне проектування на мові С++ використовуючи CLR компоненти для створення Windows Forms.

Мета курсової роботи – систематизація, поглиблення і активне застосування знань з об’єктно – орієнтованого програмування, закріплення знань, отриманих в лекційному курсі, а також на практичних і лабораторних заняттях.

Метод дослідження – вивчення літератури, складання і налагодження програм на комп'ютері.

Програма “Гра в Шибинецю”, що реалізована в цій роботі, призначена для покращення логіки та пам’яті гравця. Розроблено проект " Гра в Шибинецю " повністю відповідний умові завдання.

Ключові слова: ШИБЕНЕЦЯ, ПЕРСОНАЖ, C++, СLR, ФУНКЦІЯ, ФАЙЛ, ГРА, ЛОГІКА, ГРАВЕЦЬ, КОРИСТУВАЧ, СЛОВО, ЗАГАДКОВЕ СЛОВО, ПРОГРАМА, ПОЛЕ, ПАНЕЛЬ, НОВА ВЕРСІЯ.

ЗМІСТ

[ЗАВДАННЯ 2](#_Toc104830487)

[РЕФЕРАТ 3](#_Toc104830488)

[ВСТУП 5](#_Toc104830489)

[1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 6](#_Toc104830490)

[1.1 Основні особливості середовища програмування VS 6](#_Toc104830491)

[1.2 Стандартні компоненти 7](#_Toc104830492)

[2 РОЗРОБКА ПРОГРАМИ 8](#_Toc104830493)

[2.1 Опис алгоритму гри в “Шибинецю” 8](#_Toc104830494)

[2.2 Опис алгоритмів створення елементів форми 8](#_Toc104830495)

[2.2 Опис використовуваних методів і подій 11](#_Toc104830496)

[2.2.1 Алгоритм режиму гри для одного гравця 11](#_Toc104830497)

[2.2.2 Алгоритм режиму гри для двох гравців 11](#_Toc104830498)

[2.2.3 Алгоритм перевірки слова 12](#_Toc104830499)

[2.2.3 Алгоритм вимальовування персонажу 13](#_Toc104830500)

[3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ 15](#_Toc104830501)

[3.1 Інтерфейс користувача 15](#_Toc104830502)

[3.2 Інструкція користувача 18](#_Toc104830503)

[ВИСНОВКИ 19](#_Toc104830504)

[СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ 20](#_Toc104830505)

[ДОДАТОК А 21](#_Toc104830506)

[ДОДАТОК Б 37](#_Toc104830507)

[ДОДАТОК В 44](#_Toc104830508)

[ДОДАТОК Г 48](#_Toc104830509)

[ДОДАТОК Д 52](#_Toc104830510)

# ВСТУП

Комп’ютерні технології – це невід’ємна частина повсякденного життя. Вони спрощують процеси, а не можливе перетворюють в реальність. Та саме сьогоденні системи зробили найбільший прорив за всю історію. За допомогою таких технології людина може знати майбутнє та минуле, дати точне пояснення невідомому і відкрити вже давно забуте. Тому вони так швидко розвиваються та мають велику значимість для людства.

Важко уявити медицину без сучасних технологій, подорожі без транспорту та GPS, просте спілкування на відстані або навчання. Тому не дивно що звичайні ігри також стали залежними від комп’ютерних технології.

Ігри є соціальною частиною суспільства. Вони відображають різні сфери діяльності. Сфера освіти не стала виключенням, тому у сьогоденних закладах освіти навчальний процес набагато привабливіший, ніж був раніше. Адже граючись, людина більше запам’ятовує, бо для досягнення деякої цілі потрібно дечому навчитися та конкуренція між іншими учнями цьому сприяє. А отже, у багатьох школах нерідко зустрічаються інтерактивні дошки, за допомогою яких і відбувається сучасний процес навчання.

Гра в «Шибеницю» або її варіації є одною із популярних навчально-розвивальних ігор. Тому її часто використовують в закладах освіти на конкретних заняттях. Вона допомагає учням вивчити нові слова та вдосконалювати мовні навички.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Основні особливості середовища програмування VS

Visual Studio[1] – інтегроване середовище розробки програмного забезпечення від фірми Microsoft. Дане середовище дозволяє створювати різноманітні програмні продукти: консольні програми, програми з графічним інтерфейсом, наприклад віконні додатки Windows Forms, а також Web-додатки тощо. Середовище Visual Studio дозволяє розробляти додатки, використовуючи різні мови програмування: Visual C#, Visual Basic, Visual F#, Visual C++, Python і т.д. Приклад мов підтримуваних Visual Studio продемонстровано на рисунку 1.1. Також існує можливість розробляти додатки не тільки під Windows, а і під інші популярні платформи: Android, iOS.

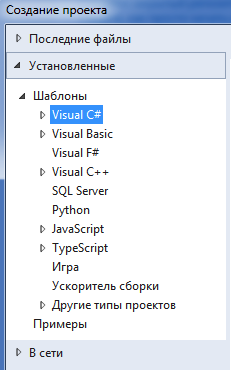


Рисунок 1.1 – Мови підтримувані Visual Studio

## 1.2 Стандартні компоненти

Основними компонентами для створення додатку є компоненти CLR пакету Microsoft .Net Framework [2].

Особливостями проекту такого типу є те, що CLR транслює початковий код [3], підтримуваний .Net Framework, у байт-код мовою IL, а вже потім у нативний код (NC). Це дозволяє створити проект за допомогою різних мов програмувань, використовуючи їхні особливості. Схема виконання коду за допомогою CLR на мовах .NET продемонстровано на рисунку 1.2.

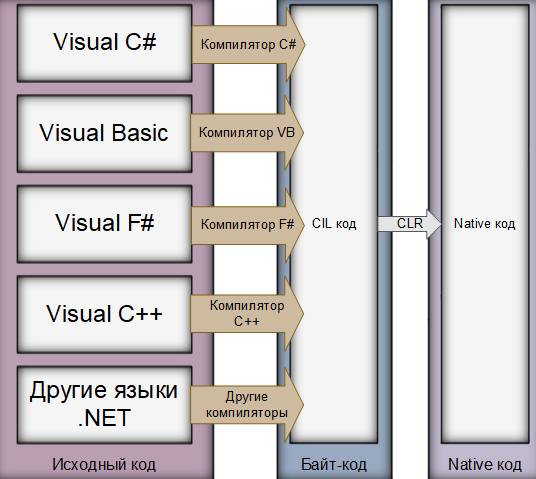


Рисунок 1.2 – Приклад роботи компонента CLR

# 2 РОЗРОБКА ПРОГРАМИ

## 2.1 Опис алгоритму гри в “Шибинецю”

Один з гравців загадує слово — пише на папері першу та останню літери слова й позначає місця для інших букв, наприклад, рисками (існує також варіант, коли спочатку невідомо жодної літери). Також малює шибеницю з петлею. Слово має бути іменником, загальною назвою, у називному відмінку однини, або множини (за відсутності у слова форми однини). Другий гравець пропонує літеру, яка може входити в це слово. Якщо така літера є в слові, то перший гравець пише її над відповідними рисками — стільки разів, скільки вона зустрічається в слові. Якщо такої літери немає, то до шибениці домальовують коло в петлі, що зображає голову. Другий гравець продовжує відгадувати літери доти, доки не відгадає все слово. За кожну неправильну відповідь перший гравець додає одну частину тулуба до шибениці (зазвичай їх сім :голова, шия, тулуб, дві руки та дві ноги). Якщо тулуб у шибениці намальований повністю, то гравець, що відгадував, вважається повішеним. Якщо гравцеві вдається вгадати слово, він виграє і може загадувати слово.

Тому, форма має нести у собі такі елементи, які мають бути використаними як на аркуші паперу під час гри, а саме: малюнок шибениці, поле для закодованого слова, поле використаних літер.

## 2.2 Опис алгоритмів створення елементів форми

Знаючи правила гри та основні її елементи можна створити досить просту за інтерфейсом форму.

Кожна програма містить у собі головні елементи управління. У цьому випадку головними елементами управління є стрічка меню (MenuStrip) під назвою MenuStrip1. Вона у собі містить елементи управління такі як: вихід з програми, правила гри, докладніше про програму, а також режими гри, які поділяються на одиночний та гра двох гравців. Стрічка меню та її елементи продемонстровано на рисунку 2.1.

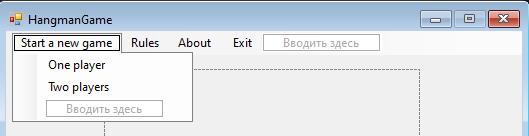


Рисунок 2.1 - Стрічка меню

Для гри Шибениця місце має буди для поля малюнку (PictureBox), у якому буде зображено персонаж гри шибеник. Тому створено поле малюнку під назвою picture\_of\_hangman, яку продемонстровано на рисунку 2.2.

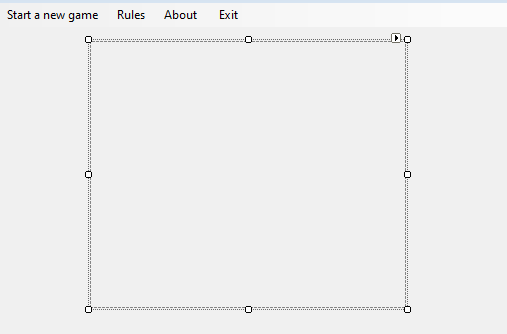


Рисунок 2.2 - Поле малюнку

Оскільки це поле не передбачає взаємодії на пряму з користувачем, то воно має статус заблоковане (Locked). Отже, користувач не зможе додати свою картинку або переміщати поле малюнку по формі під час користування програмою.

Для відображення закодованого і розкодованого слова використовується елемент лейбл (Lable). Тому під полем малюнку було розміщено лайбл під назвою OutPutGuees, який продемонстровано на рисунку 2.3

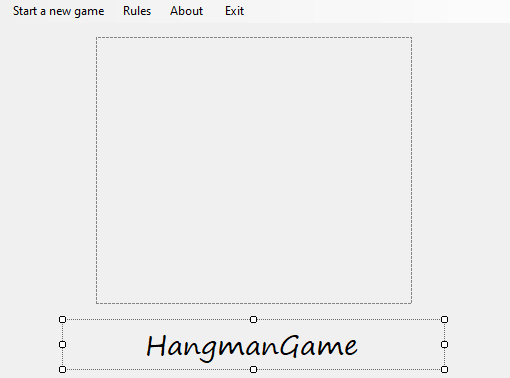


Рисунок 2.3 - Лейбл виводу слова

Щоб відгадати слово потрібно кожного разу вибирати літеру та передавати її до програми, а також запам’ятовувати вже використані. Для зручності було створено панель, яка у собі містить кнопки літер. Так у keyboardPanel розміщено двадцять шість таких, які відповідають англійському алфавіту від A до Z, що продемонстровано на рисунку 2.4.

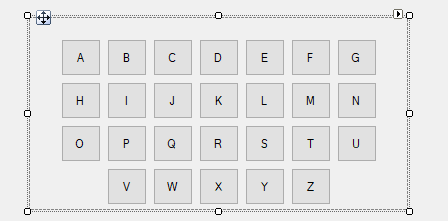


Рисунок 2.4 - Панель літер

Панель за замовчуванням не активна, тому користувач не зможе скористатися її функціями до початку гри.

Для того щоб не створювати іншу форму для режиму двох гравців, за полем малюнку було додано лейбл, який вказує що потрібно ввести загадкове слово для гри, поле вводу (TextBox) слова та кнопку підтвердження, рис.2.5. При вибору режиму двох гравців, приховується панель keyboardPanel та поле малюнку picture\_of\_hangman, щоб взаємодіяти з цими елементами. Початковий лейбел не приховується, лише для того щоб було зрозуміло яке кодове слово було до цього.

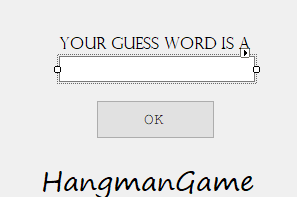


Рисунок 2.5 - Полле вводу загадкового слова

Повний код вище описаних елементів форми наведено у додатку A.

## 2.2 Опис використовуваних методів і подій

### 2.2.1 Алгоритм режиму гри для одного гравця

При вибору режиму в гри для одного гравця, визивається функція включення панелі keyboardPanel, навмання обирання слова програмою з текстового файлу слів, що наведений у додатку Д та переведення цього слова в верхні регістри. Далі слово маскується решітками на виводиться у лейбл OutPutGuees. Приклад алгоритму:

System::Void Hangman::HangmanGame::OnePlayerMod\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.startNewGame();

OnButtonsKeyboard();

std::string CodeWordSet = LoadRandomWord();

std::string DisplayGuessSet;

transform(CodeWordSet.begin(), CodeWordSet.end(), CodeWordSet.begin(), ::toupper);

DisplayGuessSet = CodeWordSet;

for (int i = 0; i < CodeWordSet.length(); i++)

DisplayGuessSet[i] = '#';

Game.setCodeWord(CodeWordSet);

Game.setDisplayGuess(DisplayGuessSet);

OutPutGuees -> Text = Convert\_string\_to\_String(Game.getDisplayGuess());

picture\_of\_hangman->Show();

keyboardPanel->Show();

InputButton->Enabled = false;

}

Повний код функцій LoadRandomWord та toupperHangman наведено в додатку В.

### 2.2.2 Алгоритм режиму гри для двох гравців

При вибору режиму для двох гравців приховується поле малюнку та панель літер і надається можливість користувачеві взаємодіяти з текстовим полем вводу. Приклад такого методу форми наведено нижче.

System::Void Hangman::HangmanGame::OnePlayerMod\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.startNewGame();

OnButtonsKeyboard();

std::string CodeWordSet = LoadRandomWord();

std::string DisplayGuessSet;

transform(CodeWordSet.begin(), CodeWordSet.end(), CodeWordSet.begin(), ::toupper);

DisplayGuessSet = CodeWordSet;

for (int i = 0; i < CodeWordSet.length(); i++)

DisplayGuessSet[i] = '#';

Game.setCodeWord(CodeWordSet);

Game.setDisplayGuess(DisplayGuessSet);

OutPutGuees -> Text = Convert\_string\_to\_String(Game.getDisplayGuess());

picture\_of\_hangman->Show();

keyboardPanel->Show();

InputButton->Enabled = false;

}

Після отримання доступу до текстового поля, в це поле є можливість записати слово. Після вводу загадкового слова, викликається біліотечний конструктор, що переводить його літери у верхній регістр, закодоване слово відображається в лейблі, а поле малюнку та панель літер відображаються. Приклад такого методу наведено нижче.

System::Void Hangman::HangmanGame::InputButton\_Click(System::Object^ sender, System::EventArgs^ e)

{

std::string CodeWordSet;

CodeWordSet = msclr::interop::marshal\_as<std::string>(InputTextBox->Text);

transform(CodeWordSet.begin(), CodeWordSet.end(), CodeWordSet.begin(), ::toupper);

bool input = false;

for (size\_t i = 0; i < CodeWordSet.length(); i++) {

if (CodeWordSet[i] >= 'A' && CodeWordSet[i] <= 'Z')

input = true;

else { input = false; MessageBox::Show("False input", "Error");break;}

}

if(input == true) {

std::string DisplayGuessSet = CodeWordSet;

for (int i = 0; i < CodeWordSet.length(); i++)

DisplayGuessSet[i] = '#';

Game.setCodeWord(CodeWordSet);

Game.setDisplayGuess(DisplayGuessSet);

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

picture\_of\_hangman->Show();

keyboardPanel->Show(); }

}

Повний текст режиму для двох гравців наведено у додатку Б.

### 2.2.3 Алгоритм перевірки слова

Метод, що представлений нижче, слугує для перевірки наявності літери у слові, підрахунку помилок з виведенням повідомлення про залишок спроб.

void HangmanStartTheGame::check(char response)

{

bool goodGuess = false;

for (int i = 0;i < code\_word.length();i++) {

if (response == code\_word[i]) {

display\_guess[i] = code\_word[i];

opened++;

goodGuess = true;

}

}

if (!goodGuess) {

mistakes++;

DrawingDeadMan(mistakes);

}

if (opened == code\_word.length()) {

Won = true;

MessageBox::Show("You`ve Won ;)", "Rules");

}

else if (mistakes == 7) {

Lose = true;

MessageBox::Show("You`ve Lose :(", "Rules");

display\_guess = code\_word;

}

}

Приклад використання методу перевірки вже в мутоді однієї з кнопок панелі keyboardPanel:

System::Void Hangman::HangmanGame::A\_button\_Click(System::Object^ sender, System::EventArgs^ e )

{

Game.check('A');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

A\_button->Enabled = false;

}

Таким чином, за дією з кнопкою, у метод перевірки передається літера тотожна назві цієї кнопки, виведення правильної відповіді та її виключення, для уникнення повторного використання.

Повний текст методу та методів інших кнопок наведено у додатку Б

### 2.2.3 Алгоритм вимальовування персонажу

Існують три мітки, що сигналізують для поля малюнку яку картинку треба вимальовувати, а саме Won, Lose, mistakes. Ці картинки називаються спрайтами, їх вісім. Всі вони відповідають фазі помилкової гри. Тобто за одну помилку відповідає відповідний спрайт. Якщо загадкове слово було розгадане або зроблено сім помилок при грі, то доступ до панелі літер буде заблокований. Приклад такого методу наведено нижче.

System::Void Hangman::HangmanGame::keyboardPanel\_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e)

{

Bitmap^ zero\_mistake = gcnew Bitmap("sprites/0.bmp");

Bitmap^ one\_mistake = gcnew Bitmap("sprites/1.bmp");

Bitmap^ two\_mistakes = gcnew Bitmap("sprites/2.bmp");

Bitmap^ three\_mistakes = gcnew Bitmap("sprites/3.bmp");

Bitmap^ four\_mistakes = gcnew Bitmap("sprites/4.bmp");

Bitmap^ five\_mistakes = gcnew Bitmap("sprites/5.bmp");

Bitmap^ six\_mistakes = gcnew Bitmap("sprites/6.bmp");

Bitmap^ seven\_mistakes = gcnew Bitmap("sprites/7.bmp");

if (Game.getMistakes() == 0) picture\_of\_hangman->Image = zero\_mistake;

else if (Game.getMistakes() == 1) picture\_of\_hangman->Image = one\_mistake;

else if (Game.getMistakes() == 2) picture\_of\_hangman->Image = two\_mistakes;

else if (Game.getMistakes()==3) picture\_of\_hangman->Image = three\_mistakes;

else if (Game.getMistakes() == 4) picture\_of\_hangman->Image = four\_mistakes;

else if (Game.getMistakes() == 5) picture\_of\_hangman->Image = five\_mistakes;

else if (Game.getMistakes() == 6) picture\_of\_hangman->Image = six\_mistakes;

else if (Game.getMistakes()==7) picture\_of\_hangman->Image = seven\_mistakes;

// shut down the keyboard

if (Game.getWon() == true || Game.getLose() == true)

keyboardPanel->Enabled = false;

}

Повний текст програми наведено у додатку Б. Спрайти та опис до них наведено у додатку Е.

# 3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

## 3.1 Інтерфейс користувача

Вікно програми середнього розміру (532;626) без можливості збільшити, зменшити або розширити. Після запуску програми користувач може побачити меню гри з головними елементами управління у верхньому кутку вікна, поле малюнку по середині ігрового поля та не активну панель літер, що продемонстровано на рисунку 3.1.

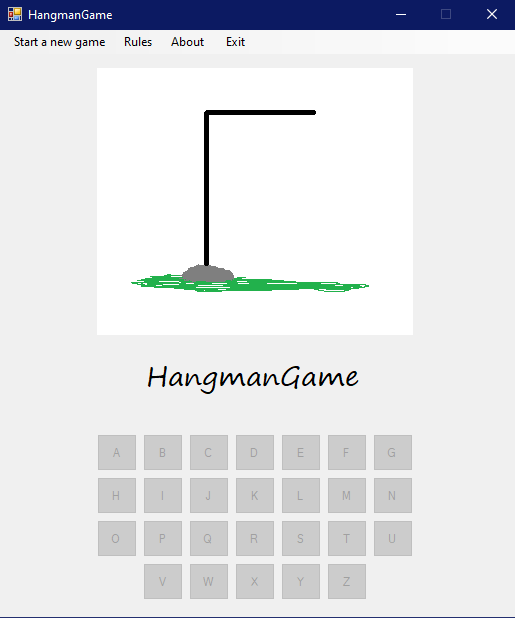


Рисунок 3.1 – Головне меню

У меню користувач може обрати два режими гри, а саме: гра для одного гравця, гра для двох гравців. Меню вибору режимів гри продемонстровано на рисунку 3.2.

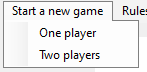


Рисунок 3.2 - Режими гри

Після того як користувач обрав гру для одного гравця, йому надається доступ до керування панеллю літер та закодоване слово, обране системою, що продемонстровано рисунку 3.3.

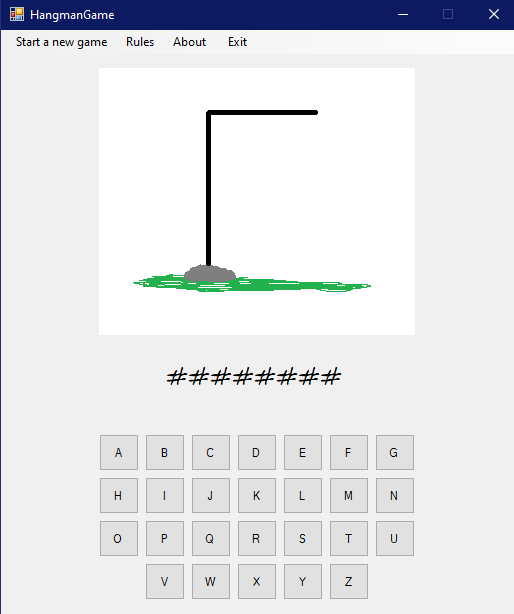


Рисунок 3.3 – Початок гри

Якщо було обрано режим для двох гравців, гравець, що загадує слово іншому, зустрічає меню загаданого слова, у якому йому потрібно ввести його у потрібне поле та підтвердити, натиснувши кнопку ОК, що продемонстровано на рисунку 3.4.

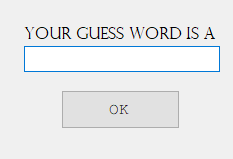


Рисунок 3.4 – Поле вводу загадкового слова

Підтвердивши слово гравцем, що загадує слово, гравець, що відгадує слово, попаде до ігрового поля, на таке саме поле як у режими для одного гравця, див. рис. 3.3.

Під час гри користувач може ознайомитися з правилами гри в шибеницю, для цього потрібно у меню обрати розділ Rules. Правила гри відображаються, згідно рисунку 3.5, у вигляді програмного повідомлення.

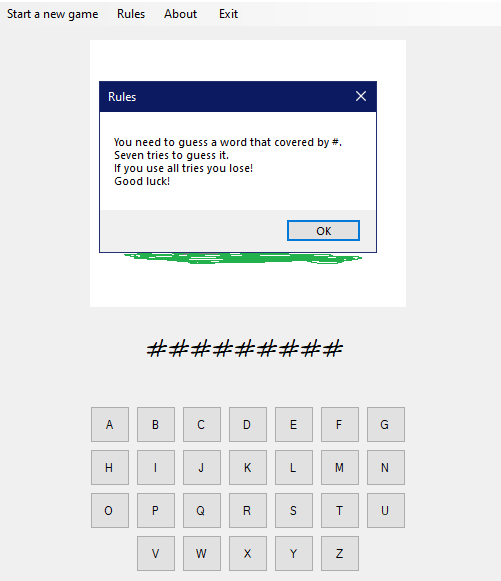


Рисунок 3.5 – Правила гри

У розділі About, користувач може ознайомитися з творцем гри, а кнопка Exit відповідає про припинення роботи користувачем програми та вихід з неї.

## 3.3 Порівняня попердньої версії з новою

У новій версії було перебудовано код за допомогою ООП, тому було рішення відійти від оголошення глобальних зміних. Замість них було створено відповідний клас HangmanStartTheGame, який у полі данних має шість керуючих елементів гри такі як: кодове слово, закодоване слово, помилки, правильні відпводі, флаг виграшу та програшу. Щоб оперувати цими елементами, було створено відповідни сетери та гетери, а також функія перевірки є методом класу. Приклад цього класу наведено нижче.

class HangmanStartTheGame {

std::string code\_word;

std::string display\_guess;

int opened = 0;

int mistakes = 0;

bool Won = false;

bool Lose = false;

public:

//reset the Game

void startNewGame() {

this->opened = 0;

this->mistakes = 0;

this->Won = false;

this->Lose = false;

}

HangmanStartTheGame() {

opened = 0;

mistakes = 0;

Won = false;

Lose = false;

}

// setters

void setCodeWord(const string CodeWord) { this->code\_word = CodeWord; };

void setDisplayGuess(const string DisplayGuess) { this->display\_guess = DisplayGuess; };

//getter

std::string getCodeWord() const { return this->code\_word; } ;

std::string getDisplayGuess() const { return this->display\_guess; };

int getOpened() const { return this->opened; };

int getMistakes() const { return this->mistakes; };

bool getWon() const { return this->Won; };

bool getLose() const { return this->Lose; };

// main functions

void check(char response);

};

Повний код класу програми наведено у додатку Г.

За допомогою бібліотеки STL string було замінено динамічні масиви рядків на рядки типу string, що допомогло не звертати увагу на очищення пам’яті після роботи з такими рядками. А за допомогою бібліотек fstream та vector було перебудовано функцію з пошуком випашкового слова для режиму гри наодинці, тому код нової функції виглядає саме так:

string LoadRandomWord()

{

int lineCount = 0;

std::string word;

std::vector<string> v;

std::ifstream reader("words.txt");

if (reader.is\_open()) {

while (getline(reader, word))

v.push\_back(word);

int randomLine = rand() % v.size();

word = v.at(randomLine);

reader.close();

}

return word;

}

Повний текст програми наведено у додатку Б.

У методах форми було використано методи створеного класу гри, описаного вище. Код методів форми та використання таких методів було вже описано вище. Повний код програми наведено у додатку Б.

Також додано перевірку на коректних ввід данних, що продемонстровано на рисунку 3.6.

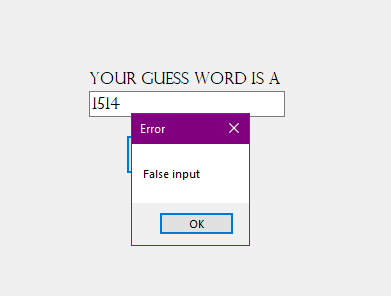


Рисунок 3.6 – Помилка на коректність вводу

Таким чином, гравець, що задає слово для опенента, не може увести числові данні та літерали не з англійського алфавіта, оскільки введення таких данних може нашкодити програмі та отримати неправомірну вигоду над опонентом, яка порушує правила гри.

У новій версії додатку було прибрано підказку попереднього слова, яке було у грі, тому що це не надає додаткової інформації користувачеві. Тому вигляд при введені кодового слова буде виглядати так, як це продемонстровано на рисунку 3.7.

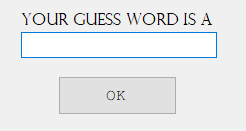


Рисунок 3.7 – Новий вигляд задання кодового слова

## 3.2 Інструкція користувача

При відкритті застосунку, користувач опиняється в головному меню гри. У верхньому кутку користувач може звернутися до розділів кeрівних елементів програми. В розділі «Початок нової гри» користувач може вибрати один з режимів гри. У режимі «Гра для одного гравця» користувач буде грати проти комп’ютера, який вибере слово із списку слів. Заздалегідь список слів у програмі дізнатися не можна, а також під час гри. Користувачеві дається необмежений час для того щоб розгадати загадкове слово. Також, користувач може помилитися не більше шести разів. На сьому помилку гравцеві зараховується поразка. З кожною помилкою на шибениці поступово вимальовується персонаж. Режим «Гра для двох гравців» аналогічний описаному раніше, але вже у якості противника виступає інший користувач. Один гравець, що загадує слово має його ввести у відповідному полі, а вже потім підтвердити свій вибір. Його опонент має відгадати слово, але якщо не впорається з завданням, то гравець, що загадав, отримає перемогу. Ознайомитися у будь який час з правилами гри можна у розділі «Правила».

# ВИСНОВКИ

Під час роботи було використано наступні основні теми з курсу:

* класи;
* конструктори класів;
* методи класів;
* робота з файлами та потоками бібліотеки fstream;
* робота з STL бібліотеками.

Закріплено старі знання з компонентом CLR за допомогою додаткових джерел [4].

Було детально проаналізовано побудовано логіку гри, щоб модифікувати код за допомогою об’єктно – орієнтованого програмування. Труднощів під час роботи не виникало. Такий підхід вирішив низьку проблем, а найголовніше допоміг оптимізувати програму та пришвидчити її роботу. Також, якщо виникне потреба у доопрацюванні або для додавання нового функціанолу гри, такий код дає більше можливостей, ніж постійно оголошувати низку повторно оперуємих змінних, що зазвичай приводить до збоїв у програмному продукті.

У цій роботі було закріплено низку значущих тем та методів, які багаторазово використовуються у різного рівня програмування. Ці знання є фундаментальними й без них переходити на інший, складніший, рівень неможливо, оскільки він потребує базових навичок та вмінь.

Проаналізувавши проблематику створення застосунків у освітній сфері можна дійти висновку, що за підтримкою сучасних технології та їх подальшим розвитком розвиваються й сфери, які використовують новітні комп'ютерні технології. Для того, щоб покращити рівень освітнього процесу, потрібно долучити фахівців з комп'ютерних технологій у процес вдосконалення навчального процесу на різних освітніх рівнях. Це надасть високий рівень підготовки для майбутніх спеціалістів. Тому не варто недооцінювати можливості сучасного світу.

# СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Microsoft Visual Studio (IDE) overview // <https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022>, 15.05.2022.
2. Microsoft Common Language Runtime (CLR) overview // <https://docs.microsoft.com/en-us/dotnet/standard/clr>, 15.05.2022.
3. Wikipedia Common Language Runtime // <https://en.wikipedia.org/wiki/Common_Language_Runtime>, 15.05.2022.
4. SamoYchiteli.ru Иллюстрированный самоучитель по Architecture .NET // <http://samoychiteli.ru/document26393.html>, 15.05.2022.

# ДОДАТОК А

Текст програми файлу HangmanGame.h

#pragma once

namespace Hangman {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// </summary>

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

{

public:

HangmanGame(void)

{

InitializeComponent();

//

//

}

protected:

/// <summary>

/// </summary>

~HangmanGame()

{

if (components)

{

delete components;

}

}

public: System::Windows::Forms::PictureBox^ picture\_of\_hangman;

public: System::Windows::Forms::Button^ A\_button;

public: System::Windows::Forms::Button^ B\_button;

public: System::Windows::Forms::Button^ C\_button;

public: System::Windows::Forms::Button^ D\_button;

public: System::Windows::Forms::Button^ E\_button;

public: System::Windows::Forms::Button^ F\_button;

public: System::Windows::Forms::Button^ G\_button;

public: System::Windows::Forms::Button^ H\_button;

public: System::Windows::Forms::Button^ I\_button;

public: System::Windows::Forms::Button^ J\_button;

public: System::Windows::Forms::Button^ K\_button;

public: System::Windows::Forms::Button^ L\_button;

public: System::Windows::Forms::Button^ M\_button;

public: System::Windows::Forms::Button^ N\_button;

public: System::Windows::Forms::Button^ O\_button;

public: System::Windows::Forms::Button^ P\_button;

public: System::Windows::Forms::Button^ Q\_button;

public: System::Windows::Forms::Button^ R\_button;

public: System::Windows::Forms::Button^ S\_button;

public: System::Windows::Forms::Button^ T\_button;

public: System::Windows::Forms::Button^ U\_button;

public: System::Windows::Forms::Button^ V\_button;

public: System::Windows::Forms::Button^ W\_button;

public: System::Windows::Forms::Button^ X\_button;

public: System::Windows::Forms::Button^ Y\_button;

public: System::Windows::Forms::Button^ Z\_button;

protected:

protected:

protected:

private:

public: System::Windows::Forms::Panel^ keyboardPanel;

public:

public: System::Windows::Forms::Label^ OutPutGuees;

public: System::Windows::Forms::ToolStripMenuItem^ startANewGameToolStripMenuItem;

public: System::Windows::Forms::ToolStripMenuItem^ OnePlayerMod;

public: System::Windows::Forms::ToolStripMenuItem^ TwoPlayersMod;

public: System::Windows::Forms::ToolStripMenuItem^ Rules;

public: System::Windows::Forms::ToolStripMenuItem^ aboutToolStripMenuItem;

public: System::Windows::Forms::ToolStripMenuItem^ exitToolStripMenuItem;

public: System::Windows::Forms::MenuStrip^ menuStrip1;

public: System::Windows::Forms::Label^ InputLabel;

public:

public: System::Windows::Forms::TextBox^ InputTextBox;

public: System::Windows::Forms::Button^ InputButton;

public:

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

private:

/// <summary>

/// </summary>

#pragma region Windows Form Designer generated code

/// <summary>

/// </summary>

void InitializeComponent(void)

{

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

this->picture\_of\_hangman = (gcnew System::Windows::Forms::PictureBox());

this->A\_button = (gcnew System::Windows::Forms::Button());

this->B\_button = (gcnew System::Windows::Forms::Button());

this->C\_button = (gcnew System::Windows::Forms::Button());

this->D\_button = (gcnew System::Windows::Forms::Button());

this->E\_button = (gcnew System::Windows::Forms::Button());

this->F\_button = (gcnew System::Windows::Forms::Button());

this->G\_button = (gcnew System::Windows::Forms::Button());

this->H\_button = (gcnew System::Windows::Forms::Button());

this->I\_button = (gcnew System::Windows::Forms::Button());

this->J\_button = (gcnew System::Windows::Forms::Button());

this->K\_button = (gcnew System::Windows::Forms::Button());

this->L\_button = (gcnew System::Windows::Forms::Button());

this->M\_button = (gcnew System::Windows::Forms::Button());

this->N\_button = (gcnew System::Windows::Forms::Button());

this->O\_button = (gcnew System::Windows::Forms::Button());

this->P\_button = (gcnew System::Windows::Forms::Button());

this->Q\_button = (gcnew System::Windows::Forms::Button());

this->R\_button = (gcnew System::Windows::Forms::Button());

this->S\_button = (gcnew System::Windows::Forms::Button());

this->T\_button = (gcnew System::Windows::Forms::Button());

this->U\_button = (gcnew System::Windows::Forms::Button());

this->V\_button = (gcnew System::Windows::Forms::Button());

this->W\_button = (gcnew System::Windows::Forms::Button());

this->X\_button = (gcnew System::Windows::Forms::Button());

this->Y\_button = (gcnew System::Windows::Forms::Button());

this->Z\_button = (gcnew System::Windows::Forms::Button());

this->keyboardPanel = (gcnew System::Windows::Forms::Panel());

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

this->startANewGameToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->OnePlayerMod = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->TwoPlayersMod = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->Rules = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->aboutToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->exitToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());

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

this->InputTextBox = (gcnew System::Windows::Forms::TextBox());

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

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

this->keyboardPanel->SuspendLayout();

this->menuStrip1->SuspendLayout();

this->SuspendLayout();

//

// picture\_of\_hangman

//

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

this->picture\_of\_hangman->Location = System::Drawing::Point(98, 38);

this->picture\_of\_hangman->Name = L"picture\_of\_hangman";

this->picture\_of\_hangman->Size = System::Drawing::Size(316, 267);

this->picture\_of\_hangman->TabIndex = 2;

this->picture\_of\_hangman->TabStop = false;

this->picture\_of\_hangman->Click += gcnew System::EventHandler(this, &HangmanGame::picture\_of\_hangman\_Click);

//

// A\_button

//

this->A\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->A\_button->Location = System::Drawing::Point(32, 22);

this->A\_button->Name = L"A\_button";

this->A\_button->Size = System::Drawing::Size(40, 37);

this->A\_button->TabIndex = 6;

this->A\_button->Text = L"A";

this->A\_button->UseVisualStyleBackColor = true;

this->A\_button->Click += gcnew System::EventHandler(this, &HangmanGame::A\_button\_Click);

//

// B\_button

//

this->B\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->B\_button->Location = System::Drawing::Point(78, 22);

this->B\_button->Name = L"B\_button";

this->B\_button->Size = System::Drawing::Size(40, 37);

this->B\_button->TabIndex = 7;

this->B\_button->Text = L"B";

this->B\_button->UseVisualStyleBackColor = true;

this->B\_button->Click += gcnew System::EventHandler(this, &HangmanGame::B\_button\_Click);

//

// C\_button

//

this->C\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->C\_button->Location = System::Drawing::Point(124, 22);

this->C\_button->Name = L"C\_button";

this->C\_button->Size = System::Drawing::Size(40, 37);

this->C\_button->TabIndex = 8;

this->C\_button->Text = L"C";

this->C\_button->UseVisualStyleBackColor = true;

this->C\_button->Click += gcnew System::EventHandler(this, &HangmanGame::C\_button\_Click);

//

// D\_button

//

this->D\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->D\_button->Location = System::Drawing::Point(170, 22);

this->D\_button->Name = L"D\_button";

this->D\_button->Size = System::Drawing::Size(40, 37);

this->D\_button->TabIndex = 9;

this->D\_button->Text = L"D";

this->D\_button->UseVisualStyleBackColor = true;

this->D\_button->Click += gcnew System::EventHandler(this, &HangmanGame::D\_button\_Click);

//

// E\_button

//

this->E\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->E\_button->Location = System::Drawing::Point(216, 22);

this->E\_button->Name = L"E\_button";

this->E\_button->Size = System::Drawing::Size(40, 37);

this->E\_button->TabIndex = 10;

this->E\_button->Text = L"E";

this->E\_button->UseVisualStyleBackColor = true;

this->E\_button->Click += gcnew System::EventHandler(this, &HangmanGame::E\_button\_Click);

//

// F\_button

//

this->F\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->F\_button->Location = System::Drawing::Point(262, 22);

this->F\_button->Name = L"F\_button";

this->F\_button->Size = System::Drawing::Size(40, 37);

this->F\_button->TabIndex = 11;

this->F\_button->Text = L"F";

this->F\_button->UseVisualStyleBackColor = true;

this->F\_button->Click += gcnew System::EventHandler(this, &HangmanGame::F\_button\_Click);

//

// G\_button

//

this->G\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->G\_button->Location = System::Drawing::Point(308, 22);

this->G\_button->Name = L"G\_button";

this->G\_button->Size = System::Drawing::Size(40, 37);

this->G\_button->TabIndex = 12;

this->G\_button->Text = L"G";

this->G\_button->UseVisualStyleBackColor = true;

this->G\_button->Click += gcnew System::EventHandler(this, &HangmanGame::G\_button\_Click);

//

// H\_button

//

this->H\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->H\_button->Location = System::Drawing::Point(32, 65);

this->H\_button->Name = L"H\_button";

this->H\_button->Size = System::Drawing::Size(40, 37);

this->H\_button->TabIndex = 13;

this->H\_button->Text = L"H";

this->H\_button->UseVisualStyleBackColor = true;

this->H\_button->Click += gcnew System::EventHandler(this, &HangmanGame::H\_button\_Click);

//

// I\_button

//

this->I\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->I\_button->Location = System::Drawing::Point(78, 65);

this->I\_button->Name = L"I\_button";

this->I\_button->Size = System::Drawing::Size(40, 37);

this->I\_button->TabIndex = 14;

this->I\_button->Text = L"I";

this->I\_button->UseVisualStyleBackColor = true;

this->I\_button->Click += gcnew System::EventHandler(this, &HangmanGame::I\_button\_Click);

//

// J\_button

//

this->J\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->J\_button->Location = System::Drawing::Point(124, 65);

this->J\_button->Name = L"J\_button";

this->J\_button->Size = System::Drawing::Size(40, 37);

this->J\_button->TabIndex = 15;

this->J\_button->Text = L"J";

this->J\_button->UseVisualStyleBackColor = true;

this->J\_button->Click += gcnew System::EventHandler(this, &HangmanGame::J\_button\_Click);

//

// K\_button

//

this->K\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->K\_button->Location = System::Drawing::Point(170, 65);

this->K\_button->Name = L"K\_button";

this->K\_button->Size = System::Drawing::Size(40, 37);

this->K\_button->TabIndex = 16;

this->K\_button->Text = L"K";

this->K\_button->UseVisualStyleBackColor = true;

this->K\_button->Click += gcnew System::EventHandler(this, &HangmanGame::K\_button\_Click);

//

// L\_button

//

this->L\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->L\_button->Location = System::Drawing::Point(216, 65);

this->L\_button->Name = L"L\_button";

this->L\_button->Size = System::Drawing::Size(40, 37);

this->L\_button->TabIndex = 17;

this->L\_button->Text = L"L";

this->L\_button->UseVisualStyleBackColor = true;

this->L\_button->Click += gcnew System::EventHandler(this, &HangmanGame::L\_button\_Click);

//

// M\_button

//

this->M\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->M\_button->Location = System::Drawing::Point(262, 65);

this->M\_button->Name = L"M\_button";

this->M\_button->Size = System::Drawing::Size(40, 37);

this->M\_button->TabIndex = 18;

this->M\_button->Text = L"M";

this->M\_button->UseVisualStyleBackColor = true;

this->M\_button->Click += gcnew System::EventHandler(this, &HangmanGame::M\_button\_Click);

//

// N\_button

//

this->N\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->N\_button->Location = System::Drawing::Point(308, 65);

this->N\_button->Name = L"N\_button";

this->N\_button->Size = System::Drawing::Size(40, 37);

this->N\_button->TabIndex = 19;

this->N\_button->Text = L"N";

this->N\_button->UseVisualStyleBackColor = true;

this->N\_button->Click += gcnew System::EventHandler(this, &HangmanGame::N\_button\_Click);

//

// O\_button

//

this->O\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->O\_button->Location = System::Drawing::Point(32, 108);

this->O\_button->Name = L"O\_button";

this->O\_button->Size = System::Drawing::Size(40, 37);

this->O\_button->TabIndex = 20;

this->O\_button->Text = L"O";

this->O\_button->UseVisualStyleBackColor = true;

this->O\_button->Click += gcnew System::EventHandler(this, &HangmanGame::O\_button\_Click);

//

// P\_button

//

this->P\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->P\_button->Location = System::Drawing::Point(78, 108);

this->P\_button->Name = L"P\_button";

this->P\_button->Size = System::Drawing::Size(40, 37);

this->P\_button->TabIndex = 21;

this->P\_button->Text = L"P";

this->P\_button->UseVisualStyleBackColor = true;

this->P\_button->Click += gcnew System::EventHandler(this, &HangmanGame::P\_button\_Click);

//

// Q\_button

//

this->Q\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->Q\_button->Location = System::Drawing::Point(124, 108);

this->Q\_button->Name = L"Q\_button";

this->Q\_button->Size = System::Drawing::Size(40, 37);

this->Q\_button->TabIndex = 22;

this->Q\_button->Text = L"Q";

this->Q\_button->UseVisualStyleBackColor = true;

this->Q\_button->Click += gcnew System::EventHandler(this, &HangmanGame::Q\_button\_Click);

//

// R\_button

//

this->R\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->R\_button->Location = System::Drawing::Point(170, 108);

this->R\_button->Name = L"R\_button";

this->R\_button->Size = System::Drawing::Size(40, 37);

this->R\_button->TabIndex = 23;

this->R\_button->Text = L"R";

this->R\_button->UseVisualStyleBackColor = true;

this->R\_button->Click += gcnew System::EventHandler(this, &HangmanGame::R\_button\_Click);

//

// S\_button

//

this->S\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->S\_button->Location = System::Drawing::Point(216, 108);

this->S\_button->Name = L"S\_button";

this->S\_button->Size = System::Drawing::Size(40, 37);

this->S\_button->TabIndex = 24;

this->S\_button->Text = L"S";

this->S\_button->UseVisualStyleBackColor = true;

this->S\_button->Click += gcnew System::EventHandler(this, &HangmanGame::S\_button\_Click);

//

// T\_button

//

this->T\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->T\_button->Location = System::Drawing::Point(262, 108);

this->T\_button->Name = L"T\_button";

this->T\_button->Size = System::Drawing::Size(40, 37);

this->T\_button->TabIndex = 25;

this->T\_button->Text = L"T";

this->T\_button->UseVisualStyleBackColor = true;

this->T\_button->Click += gcnew System::EventHandler(this, &HangmanGame::T\_button\_Click);

//

// U\_button

//

this->U\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->U\_button->Location = System::Drawing::Point(308, 108);

this->U\_button->Name = L"U\_button";

this->U\_button->Size = System::Drawing::Size(40, 37);

this->U\_button->TabIndex = 26;

this->U\_button->Text = L"U";

this->U\_button->UseVisualStyleBackColor = true;

this->U\_button->Click += gcnew System::EventHandler(this, &HangmanGame::U\_button\_Click);

//

// V\_button

//

this->V\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->V\_button->Location = System::Drawing::Point(78, 151);

this->V\_button->Name = L"V\_button";

this->V\_button->Size = System::Drawing::Size(40, 37);

this->V\_button->TabIndex = 27;

this->V\_button->Text = L"V";

this->V\_button->UseVisualStyleBackColor = true;

this->V\_button->Click += gcnew System::EventHandler(this, &HangmanGame::V\_button\_Click);

//

// W\_button

//

this->W\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->W\_button->Location = System::Drawing::Point(124, 151);

this->W\_button->Name = L"W\_button";

this->W\_button->Size = System::Drawing::Size(40, 37);

this->W\_button->TabIndex = 28;

this->W\_button->Text = L"W";

this->W\_button->UseVisualStyleBackColor = true;

this->W\_button->Click += gcnew System::EventHandler(this, &HangmanGame::W\_button\_Click);

//

// X\_button

//

this->X\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->X\_button->Location = System::Drawing::Point(170, 151);

this->X\_button->Name = L"X\_button";

this->X\_button->Size = System::Drawing::Size(40, 37);

this->X\_button->TabIndex = 29;

this->X\_button->Text = L"X";

this->X\_button->UseVisualStyleBackColor = true;

this->X\_button->Click += gcnew System::EventHandler(this, &HangmanGame::X\_button\_Click);

//

// Y\_button

//

this->Y\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->Y\_button->Location = System::Drawing::Point(216, 151);

this->Y\_button->Name = L"Y\_button";

this->Y\_button->Size = System::Drawing::Size(40, 37);

this->Y\_button->TabIndex = 30;

this->Y\_button->Text = L"Y";

this->Y\_button->UseVisualStyleBackColor = true;

this->Y\_button->Click += gcnew System::EventHandler(this, &HangmanGame::Y\_button\_Click);

//

// Z\_button

//

this->Z\_button->Cursor = System::Windows::Forms::Cursors::Hand;

this->Z\_button->Location = System::Drawing::Point(262, 151);

this->Z\_button->Name = L"Z\_button";

this->Z\_button->Size = System::Drawing::Size(40, 37);

this->Z\_button->TabIndex = 31;

this->Z\_button->Text = L"Z";

this->Z\_button->UseVisualStyleBackColor = true;

this->Z\_button->Click += gcnew System::EventHandler(this, &HangmanGame::Z\_button\_Click);

//

// keyboardPanel

//

this->keyboardPanel->AutoSize = true;

this->keyboardPanel->Controls->Add(this->Z\_button);

this->keyboardPanel->Controls->Add(this->Y\_button);

this->keyboardPanel->Controls->Add(this->X\_button);

this->keyboardPanel->Controls->Add(this->W\_button);

this->keyboardPanel->Controls->Add(this->V\_button);

this->keyboardPanel->Controls->Add(this->U\_button);

this->keyboardPanel->Controls->Add(this->T\_button);

this->keyboardPanel->Controls->Add(this->S\_button);

this->keyboardPanel->Controls->Add(this->R\_button);

this->keyboardPanel->Controls->Add(this->Q\_button);

this->keyboardPanel->Controls->Add(this->P\_button);

this->keyboardPanel->Controls->Add(this->O\_button);

this->keyboardPanel->Controls->Add(this->N\_button);

this->keyboardPanel->Controls->Add(this->M\_button);

this->keyboardPanel->Controls->Add(this->L\_button);

this->keyboardPanel->Controls->Add(this->K\_button);

this->keyboardPanel->Controls->Add(this->J\_button);

this->keyboardPanel->Controls->Add(this->I\_button);

this->keyboardPanel->Controls->Add(this->H\_button);

this->keyboardPanel->Controls->Add(this->G\_button);

this->keyboardPanel->Controls->Add(this->F\_button);

this->keyboardPanel->Controls->Add(this->E\_button);

this->keyboardPanel->Controls->Add(this->D\_button);

this->keyboardPanel->Controls->Add(this->C\_button);

this->keyboardPanel->Controls->Add(this->B\_button);

this->keyboardPanel->Controls->Add(this->A\_button);

this->keyboardPanel->Enabled = false;

this->keyboardPanel->Location = System::Drawing::Point(66, 382);

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

this->keyboardPanel->Size = System::Drawing::Size(379, 193);

this->keyboardPanel->TabIndex = 32;

this->keyboardPanel->Paint += gcnew System::Windows::Forms::PaintEventHandler(this, &HangmanGame::keyboardPanel\_Paint);

//

// OutPutGuees

//

this->OutPutGuees->Anchor = static\_cast<System::Windows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::Left | System::Windows::Forms::AnchorStyles::Right));

this->OutPutGuees->AutoEllipsis = true;

this->OutPutGuees->Font = (gcnew System::Drawing::Font(L"Segoe Print", 20.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->OutPutGuees->ForeColor = System::Drawing::SystemColors::InactiveCaptionText;

this->OutPutGuees->Location = System::Drawing::Point(66, 322);

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

this->OutPutGuees->Size = System::Drawing::Size(379, 47);

this->OutPutGuees->TabIndex = 33;

this->OutPutGuees->Text = L"HangmanGame";

this->OutPutGuees->TextAlign = System::Drawing::ContentAlignment::TopCenter;

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

//

// startANewGameToolStripMenuItem

//

this->startANewGameToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(2) {

this->OnePlayerMod,

this->TwoPlayersMod

});

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

this->startANewGameToolStripMenuItem->Size = System::Drawing::Size(110, 20);

this->startANewGameToolStripMenuItem->Text = L"Start a new game";

//

// OnePlayerMod

//

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

this->OnePlayerMod->Size = System::Drawing::Size(180, 22);

this->OnePlayerMod->Text = L"One player";

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

//

// TwoPlayersMod

//

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

this->TwoPlayersMod->Size = System::Drawing::Size(180, 22);

this->TwoPlayersMod->Text = L"Two players";

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

//

// Rules

//

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

this->Rules->Size = System::Drawing::Size(47, 20);

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

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

//

// aboutToolStripMenuItem

//

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

this->aboutToolStripMenuItem->Size = System::Drawing::Size(55, 20);

this->aboutToolStripMenuItem->Text = L"About ";

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

//

// exitToolStripMenuItem

//

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

this->exitToolStripMenuItem->Size = System::Drawing::Size(38, 20);

this->exitToolStripMenuItem->Text = L"Exit";

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

//

// menuStrip1

//

this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(4) {

this->startANewGameToolStripMenuItem,

this->Rules, this->aboutToolStripMenuItem, this->exitToolStripMenuItem

});

this->menuStrip1->Location = System::Drawing::Point(0, 0);

this->menuStrip1->Name = L"menuStrip1";

this->menuStrip1->Size = System::Drawing::Size(516, 24);

this->menuStrip1->TabIndex = 0;

this->menuStrip1->Text = L"menuStrip1";

//

// InputLabel

//

this->InputLabel->AutoSize = true;

this->InputLabel->Font = (gcnew System::Drawing::Font(L"Perpetua Titling MT", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->InputLabel->Location = System::Drawing::Point(161, 199);

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

this->InputLabel->Size = System::Drawing::Size(200, 19);

this->InputLabel->TabIndex = 34;

this->InputLabel->Text = L"Your guess word is a";

this->InputLabel->TextAlign = System::Drawing::ContentAlignment::MiddleCenter;

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

//

// InputTextBox

//

this->InputTextBox->Font = (gcnew System::Drawing::Font(L"Perpetua Titling MT", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->InputTextBox->Location = System::Drawing::Point(165, 221);

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

this->InputTextBox->Size = System::Drawing::Size(196, 26);

this->InputTextBox->TabIndex = 35;

this->InputTextBox->TextChanged += gcnew System::EventHandler(this, &HangmanGame::InputTextBox\_TextChanged);

//

// InputButton

//

this->InputButton->Cursor = System::Windows::Forms::Cursors::Hand;

this->InputButton->Font = (gcnew System::Drawing::Font(L"Miriam Mono CLM", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(177)));

this->InputButton->Location = System::Drawing::Point(202, 265);

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

this->InputButton->Size = System::Drawing::Size(119, 39);

this->InputButton->TabIndex = 36;

this->InputButton->Text = L"OK";

this->InputButton->UseVisualStyleBackColor = true;

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

//

// HangmanGame

//

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

this->AutoSizeMode = System::Windows::Forms::AutoSizeMode::GrowAndShrink;

this->ClientSize = System::Drawing::Size(516, 587);

this->Controls->Add(this->picture\_of\_hangman);

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

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

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

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

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

this->Controls->Add(this->menuStrip1);

this->MainMenuStrip = this->menuStrip1;

this->MaximizeBox = false;

this->Name = L"HangmanGame";

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

this->Text = L"HangmanGame";

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

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

this->keyboardPanel->ResumeLayout(false);

this->menuStrip1->ResumeLayout(false);

this->menuStrip1->PerformLayout();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

public: System::Void HangmanGame\_Load(System::Object^ sender, System::EventArgs^ e) {

}

public: System::Void groupBox1\_Enter(System::Object^ sender, System::EventArgs^ e) {

}

public: System::Void exitToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void aboutToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void Rules\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void OnePlayerMod\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void TwoPlayersMod\_Click(System::Object^ sender, System::EventArgs^ e);

//keyboard

public: System::Void keyboardPanel\_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e);

public: System::Void A\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void B\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void C\_button\_Click(System::Object ^ sender, System::EventArgs ^ e);

public: System::Void D\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void E\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void F\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void G\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void H\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void I\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void J\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void K\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void L\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void M\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void N\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void O\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void P\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void Q\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void R\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void S\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void T\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void U\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void V\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void W\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void X\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void Y\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void Z\_button\_Click(System::Object^ sender, System::EventArgs^ e);

public: System::Void InputButton\_Click(System::Object^ sender, System::EventArgs^ e);

// turn on the buttons to play

public: System::Void OnButtonsKeyboard() {

keyboardPanel->Enabled = true;

A\_button->Enabled = true;

B\_button->Enabled = true;

C\_button->Enabled = true;

D\_button->Enabled = true;

E\_button->Enabled = true;

F\_button->Enabled = true;

G\_button->Enabled = true;

H\_button->Enabled = true;

I\_button->Enabled = true;

J\_button->Enabled = true;

K\_button->Enabled = true;

L\_button->Enabled = true;

M\_button->Enabled = true;

N\_button->Enabled = true;

O\_button->Enabled = true;

P\_button->Enabled = true;

Q\_button->Enabled = true;

R\_button->Enabled = true;

S\_button->Enabled = true;

T\_button->Enabled = true;

U\_button->Enabled = true;

V\_button->Enabled = true;

W\_button->Enabled = true;

X\_button->Enabled = true;

Y\_button->Enabled = true;

Z\_button->Enabled = true;

};

public: System::Void InputTextBox\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

public: System::Void InputLabel\_Click(System::Object^ sender, System::EventArgs^ e) {

}

public: System::Void OutPutGuees\_Click(System::Object^ sender, System::EventArgs^ e) {

}

public: System::Void picture\_of\_hangman\_Click(System::Object^ sender, System::EventArgs^ e) {

}

};

}

# ДОДАТОК Б

**Текст програми файлу HangmanGame.cpp**

#include "HangmanGame.h"

#include "Hangman Funcions.h"

#include <msclr\marshal\_cppstd.h>

HangmanStartTheGame Game;

// main function for form

void main() {

srand(time(NULL));

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Hangman::HangmanGame form;

Application::Run(% form);

}

// ONE PLAYER MOD

System::Void Hangman::HangmanGame::OnePlayerMod\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.startNewGame();

OnButtonsKeyboard();

std::string CodeWordSet = LoadRandomWord();

std::string DisplayGuessSet;

transform(CodeWordSet.begin(), CodeWordSet.end(), CodeWordSet.begin(), ::toupper);

DisplayGuessSet = CodeWordSet;

for (int i = 0; i < CodeWordSet.length(); i++)

DisplayGuessSet[i] = '#';

Game.setCodeWord(CodeWordSet);

Game.setDisplayGuess(DisplayGuessSet);

OutPutGuees -> Text = Convert\_string\_to\_String(Game.getDisplayGuess());

picture\_of\_hangman->Show();

keyboardPanel->Show();

InputButton->Enabled = false;

}

// TWO PLAYERs MOD

System::Void Hangman::HangmanGame::InputButton\_Click(System::Object^ sender, System::EventArgs^ e)

{

std::string CodeWordSet;

CodeWordSet = msclr::interop::marshal\_as<std::string>(InputTextBox->Text);

transform(CodeWordSet.begin(), CodeWordSet.end(), CodeWordSet.begin(), ::toupper);

bool input = false;

for (size\_t i = 0; i < CodeWordSet.length(); i++) {

if (CodeWordSet[i] >= 'A' && CodeWordSet[i] <= 'Z')

input = true;

else {

input = false;

MessageBox::Show("False input", "Error");

break;

}

}

if(input == true) {

std::string DisplayGuessSet = CodeWordSet;

for (int i = 0; i < CodeWordSet.length(); i++)

DisplayGuessSet[i] = '#';

Game.setCodeWord(CodeWordSet);

Game.setDisplayGuess(DisplayGuessSet);

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

picture\_of\_hangman->Show();

keyboardPanel->Show();

}

}

System::Void Hangman::HangmanGame::TwoPlayersMod\_Click(System::Object^ sender, System::EventArgs^ e)

{

picture\_of\_hangman->Hide();

keyboardPanel->Hide();

Game.startNewGame();

InputButton->Enabled = true;

OutPutGuees->Text = " ";

OnButtonsKeyboard();

}

// KEYBORAD functions

System::Void Hangman::HangmanGame::keyboardPanel\_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e)

{

Bitmap^ zero\_mistake = gcnew Bitmap("sprites/0.bmp");

Bitmap^ one\_mistake = gcnew Bitmap("sprites/1.bmp");

Bitmap^ two\_mistakes = gcnew Bitmap("sprites/2.bmp");

Bitmap^ three\_mistakes = gcnew Bitmap("sprites/3.bmp");

Bitmap^ four\_mistakes = gcnew Bitmap("sprites/4.bmp");

Bitmap^ five\_mistakes = gcnew Bitmap("sprites/5.bmp");

Bitmap^ six\_mistakes = gcnew Bitmap("sprites/6.bmp");

Bitmap^ seven\_mistakes = gcnew Bitmap("sprites/7.bmp");

if (Game.getMistakes() == 0) {

picture\_of\_hangman->Image = zero\_mistake;

}

else if (Game.getMistakes() == 1) {

picture\_of\_hangman->Image = one\_mistake;

}

else if (Game.getMistakes() == 2) {

picture\_of\_hangman->Image = two\_mistakes;

}

else if (Game.getMistakes() == 3) {

picture\_of\_hangman->Image = three\_mistakes;

}

else if (Game.getMistakes() == 4) {

picture\_of\_hangman->Image = four\_mistakes;

}

else if (Game.getMistakes() == 5) {

picture\_of\_hangman->Image = five\_mistakes;

}

else if (Game.getMistakes() == 6) {

picture\_of\_hangman->Image = six\_mistakes;

}

else if (Game.getMistakes() == 7) {

picture\_of\_hangman->Image = seven\_mistakes;

}

// shut down the keyboard

if (Game.getWon() == true || Game.getLose() == true) {

keyboardPanel->Enabled = false;

}

}

System::Void Hangman::HangmanGame::A\_button\_Click(System::Object^ sender, System::EventArgs^ e )

{

Game.check('A');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

A\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::B\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('B');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

B\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::C\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('C');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

C\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::D\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('D');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

D\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::E\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('E');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

E\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::F\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('F');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

F\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::G\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('G');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

G\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::H\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('H');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

H\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::I\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('I');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

I\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::J\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('J');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

J\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::K\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('K');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

K\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::L\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('L');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

L\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::M\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('M');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

M\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::N\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('N');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

N\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::O\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('O');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

O\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::P\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('P');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

P\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::Q\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('Q');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

Q\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::R\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('R');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

R\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::S\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('S');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

S\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::T\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('T');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

T\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::U\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('U');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

U\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::V\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('V');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

V\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::W\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('W');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

W\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::X\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('X');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

X\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::Y\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('Y');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

Y\_button->Enabled = false;

}

System::Void Hangman::HangmanGame::Z\_button\_Click(System::Object^ sender, System::EventArgs^ e)

{

Game.check('Z');

OutPutGuees->Text = Convert\_string\_to\_String(Game.getDisplayGuess());

Z\_button->Enabled = false;

}

// about game

System::Void Hangman::HangmanGame::aboutToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

author();

}

// game rules

System::Void Hangman::HangmanGame::Rules\_Click(System::Object^ sender, System::EventArgs^ e)

{

rules();

}

// exit bar

System::Void Hangman::HangmanGame::exitToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

Application::Exit();

}

# ДОДАТОК В

**Текст програми файлу Hangman Functions.cpp**

#include "Hangman Funcions.h"

void HangmanStartTheGame::check(char response)

{

bool goodGuess = false;

for (int i = 0;i < code\_word.length();i++) {

if (response == code\_word[i]) {

display\_guess[i] = code\_word[i];

opened++;

goodGuess = true;

}

}

if (!goodGuess) {

mistakes++;

DrawingDeadMan(mistakes);

}

if (opened == code\_word.length()) {

Won = true;

MessageBox::Show(L"You`ve Won 😊", "HangmanGame");

}

else if (mistakes == 7) {

Lose = true;

MessageBox::Show(L"You`ve Lose 😥", "HangmanGame");

display\_guess = code\_word;

}

}

// Draw function

void DrawingDeadMan(int incorrect\_answer) {

if (incorrect\_answer == 1) {

MessageBox::Show("Mistakes 1/7", "Opps, you wrong");

}

else if (incorrect\_answer == 2) {

MessageBox::Show("Mistakes 2/7", "Opps, you wrong");

}

else if (incorrect\_answer == 3) {

MessageBox::Show("Mistakes 3/7", "Opps, you wrong");

}

else if (incorrect\_answer == 4) {

MessageBox::Show("Mistakes 4/7", "Opps, you wrong");

}

else if (incorrect\_answer == 5) {

MessageBox::Show("Mistakes 5/7", "Opps, you wrong");

}

else if (incorrect\_answer == 6) {

MessageBox::Show("Mistakes 6/7", "Opps, you wrong");

}

else if (incorrect\_answer == 7) {

MessageBox::Show("Mistakes 7/7", "Opps, you wrong");

}

}

// Get lib word function

string LoadRandomWord()

{

int lineCount = 0;

std::string word;

std::vector<string> v;

std::ifstream reader("words.txt");

if (reader.is\_open()) {

while (getline(reader, word))

v.push\_back(word);

int randomLine = rand() % v.size();

word = v.at(randomLine);

reader.close();

}

return word;

}

// rules

void rules() {

MessageBox::Show(" You need to guess a word that covered by #. \n Seven tries to guess it. \n If you use all tries you lose! \n Good luck! ", "Rules");

}

// rights

void author() {

MessageBox::Show(" created by Plotva.Inc ", "Author");

}

// Convert CLR funtions

//convert System::string^ TO std::string

std::string& Convert\_String\_to\_string(String^ s, std::string& os) {

using namespace Runtime::InteropServices;

const char\* chars =

(const char\*)(Marshal::StringToHGlobalAnsi(s)).ToPointer();

os = chars;

Marshal::FreeHGlobal(IntPtr((void\*)chars));

return os;

}

//convert std::string TO System::string^

String^ Convert\_string\_to\_String(std::string& os, String^ s) {

s = gcnew System::String(os.c\_str());

return s;

}

//convert System::string^ TO std::string

std::string& Convert\_String\_to\_string(String^ s) {

std::string os;

using namespace Runtime::InteropServices;

const char\* chars =

(const char\*)(Marshal::StringToHGlobalAnsi(s)).ToPointer();

os = chars;

Marshal::FreeHGlobal(IntPtr((void\*)chars));

return os;

}

//convert std::string TO System::string^

String^ Convert\_string\_to\_String(std::string& os) {

System::String^ s = gcnew System::String(os.c\_str());

return s;

}

//convert char TO System::string^

System::String^ Convert\_char\_to\_String(char ch) {

char\* chr = new char();

chr[0] = ch;

System::String^ str;

for (int i = 0; chr[i] != '\0'; i++)

str += wchar\_t(ch);

return str;

}

//convert System::string^ TO char\*

char\* Convert\_String\_to\_char(System::String^ string) {

using namespace System::Runtime::InteropServices;

return (char\*)(void\*)Marshal::StringToHGlobalAnsi(string);

}

//convert char\* TO System::string^

System::String^ Convert\_char\_to\_String(char\* ch) {

char\* chr = new char();

chr = ch;

System::String^ str;

for (int i = 0; chr[i] != '\0'; i++)

str += wchar\_t(ch[i]);

return str;

}

# ДОДАТОК Г

**Текст програми Hangman Functions.h**

#pragma once

#include <iostream>

#include <algorithm>

#include <string>

#include <fstream>

#include <vector>

#include "windows.h";

using namespace System::Windows::Forms;

using namespace System;

using namespace std;

class HangmanStartTheGame {

std::string code\_word;

std::string display\_guess;

int opened = 0;

int mistakes = 0;

bool Won = false;

bool Lose = false;

public:

//reset the Game

void startNewGame() {

this->opened = 0;

this->mistakes = 0;

this->Won = false;

this->Lose = false;

}

HangmanStartTheGame() {

opened = 0;

mistakes = 0;

Won = false;

Lose = false;

}

// setters

void setCodeWord(const string CodeWord) { this->code\_word = CodeWord; };

void setDisplayGuess(const string DisplayGuess) { this->display\_guess = DisplayGuess; };

//getter

std::string getCodeWord() const { return this->code\_word; } ;

std::string getDisplayGuess() const { return this->display\_guess; };

int getOpened() const { return this->opened; };

int getMistakes() const { return this->mistakes; };

bool getWon() const { return this->Won; };

bool getLose() const { return this->Lose; };

// main functions

void check(char response);

};

void DrawingDeadMan(int incorrect\_answer); // Draw function

string LoadRandomWord(); // Get lib word function

void rules(); // rules menu

void author(); // rights

// Convert functions

std::string& Convert\_String\_to\_string(String^ s, std::string& os); //convert System::string^ TO std::string

String^ Convert\_string\_to\_String(std::string& os, String^ s); //convert std::string TO System::string^

std::string& Convert\_String\_to\_string(String^ s); //convert System::string^ TO std::string

String^ Convert\_string\_to\_String(std::string& os); //convert std::string TO System::string^

System::String^ Convert\_char\_to\_String(char\* ch); //convert char\* TO System::string^

char\* Convert\_String\_to\_char(System::String^ string); //convert System::string^ TO char\*

# ДОДАТОК Д

**Текст програми words.txt**

Plate

Hangman

Car

Wash

Computer

abruptly

absurd

abyss

affix

askew

avenue

awkward

axiom

azure

bagpipes

bandwagon

banjo

bayou

beekeeper

bikini

blitz

blizzard

boggle

bookworm

boxcar

buckaroo

buffalo

buffoon

buxom

buzzard

buzzing

buzzwords

caliph

cobweb

cockiness

croquet

crypt

curacao

cycle

daiquiri

dirndl

disavow

dizzying

duplex

dwarves

embezzle

equip

espionage

euouae

exodus

faking

fishhook

fixable

fjord

flapjack

flopping

fluffiness

flyby

foxglove

frazzled

frizzled

fuchsia

funny

gabby

galaxy

galvanize

gazebo

giaour

gizmo

glowworm

glyph

gnarly

gnostic

gossip

grogginess

haiku

haphazard

hyphen

iatrogenic

icebox

injury

ivory

ivy

jackpot

jaundice

jawbreaker

jaywalk

jazziest

jazzy

jelly

jigsaw

jinx

jiujitsu

jockey

jogging

joking

jovial

joyful

juicy

jukebox

jumbo

kayak

kazoo

keyhole

khaki

kilobyte

kiosk

kitsch

kiwifruit

klutz

knapsack

larynx

lengths

lucky

luxury

lymph

marquis

matrix

megahertz

microwave

mnemonic

mystify

naphtha

nightclub

nowadays

numbskull

nymph

onyx

ovary

oxidize

oxygen

pajama

peekaboo

phlegm

pixel

pizazz

pneumonia

polka

pshaw

psyche

puppy

puzzling

quartz

queue

quips

quixotic

quiz

quizzes

quorum

razzmatazz

rhubarb

rhythm

rickshaw

schnapps

scratch

shiv

snazzy

sphinx

spritz

squawk

staff

strength

strengths

stretch

stronghold

stymied

subway

swivel

syndrome

thriftless

thumbscrew

topaz

transcript

transgress

transplant

triphthong

twelfth

twelfths

unknown

unworthy

unzip

uptown

vaporize

vixen

vodka

voodoo

vortex

voyeurism

walkway

waltz

wave

wavy

waxy

wellspring

wheezy

whiskey

whizzing

whomever

wimpy

witchcraft

wizard

woozy

wristwatch

wyvern

xylophone

yachtsman

yippee

yoked

youthful

yummy

zephyr

zigzag

zigzagging

zilch

zipper

zodiac

zombie

# ДОДАТОК Е

**Візуальна части з гри**

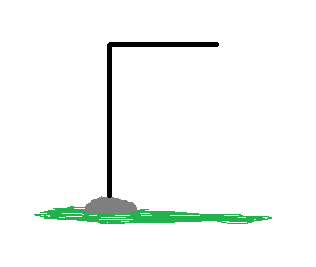


Рисунок А.1 – Шибенеця на початку гри

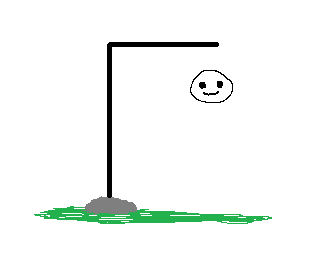


Рисунок А.2 – Шибенеця при першій помилці

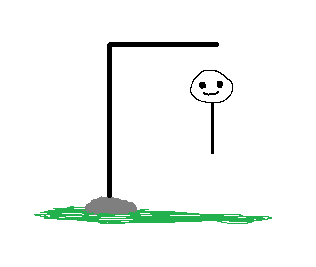


Рисунок А.3 – Шибенеця при другій помилці

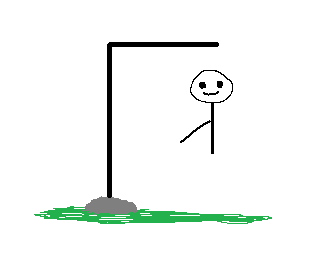


Рисунок А.4 – Шибенеця при четвертій помилці

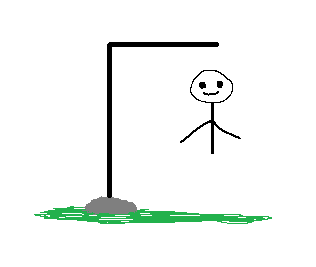


Рисунок А.5 – Шибенеця при п’ятій помилці

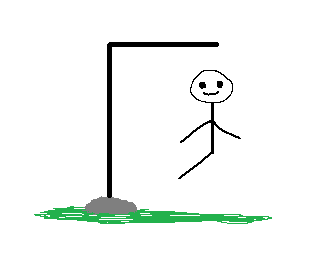


Рисунок А.6 – Шибенеця при шостій помилці

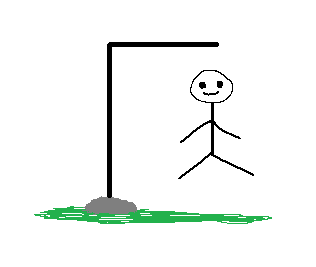


Рисунок А.7 – Шибенеця при сьомій помилці

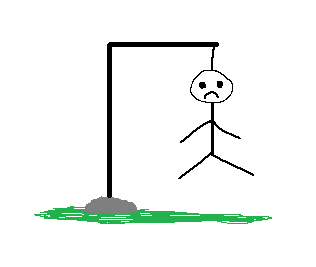


Рисунок А.8 – Шибенеця при сьомій помилці, останній