**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ДВНЗ «УНІВЕРСИТЕТ БАНКІВСЬКОЇ СПРАВИ»**

**ІНСТИТУТ БАНКІВСЬКИХ ТЕХНОЛОГІЙ ТА БІЗНЕСУ**

**КАФЕДРА КІБЕРБЕЗПЕКИ ТА СОЦІАЛЬНИХ НАУК**

**КУРСОВА РОБОТА**

з дисципліни

**«ОБ’ЄКТНО ОРІЄНТОВНЕ ПРОГРАМУВАННЯ»**

(назва дисципліни)

на тему:  **Створення гри «тетріс»**

Студента 2 курсу 203 групи

спеціальності «Кібербезпека»

Дегтерьов Олександр Олександрович

(прізвище та ініціали)

Керівник

Доцент кафедри

(посада, вчене звання, науковий ступінь,

Гордєєв О.О.

прізвище та ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_

Оцінка: ECTS \_\_\_\_

Члени комісії

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

Київ - 2019

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ДВНЗ «УНІВЕРСИТЕТ БАНКІВСЬКОЇ СПРАВИ»**

**ІНСТИТУТ БАНКІВСЬКИХ ТЕХНОЛОГІЙ ТА БІЗНЕСУ**

**КАФЕДРА КІБЕРБЕЗПЕКИ ТА СОЦІАЛЬНИХ НАУК**

Спеціальність: «Кібербезпека»

Курс 2 Група 203-Кб Семестр 1

**ЗАВДАННЯ**

**На курсову роботу студента/студентки**

Дегтерьова Олександра Олександровича

(прізвище, ім’я, по батькові)

1. Тема курсової роботи:

Розробка гри «Тетріс»

2. Термін здачі студентом закінченої роботи 19.12.2019

3. Вхідні дані до (роботи) інформаційно-аналітичні інтернет джерела, зразки рішень практичних завдань та задач, нормативно-правова база, щодо оформлення технічної документації програмних продуктів, початкові значення для обробки.

4. Зміст пояснювальної записки (перелік питань, які належить розробити). Вступ. Формулювання вимог до програми. Проектування та практична реалізація розв’язку поставленої задачі. Тестування програми і результати її виконання. Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов`язкових креслень) Таблиці та схеми представлено в роботі. Презентація додається.

6. Дата видачі завдання «10» жовтня 2019 р.

**Зміст**

[Вступ 1](#_Toc28023222)

[Розділ 1. Формулювання вимог до програми 2](#_Toc28023223)

[1.1. Аналіз технічного завдання 2](#_Toc28023224)

[1.2. Обґрунтування алгоритму і структури програми 2](#_Toc28023225)

[Розділ 2. Проектування та практична реалізація розв’язку поставленої задачі 4](#_Toc28023226)

[2.1. Оцінка коректності алгоритму та аналіз алгоритму на часову та просторову ефективність 4](#_Toc28023227)

[2.2. Розробка специфікації 4](#_Toc28023228)

[2.3. Опис програми (обґрунтування вибору мови програмування, опис основних процедур та функцій). 4](#_Toc28023229)

[2.4. Опис змінних, та їх ідентифікаторів, що використовувались у програмі 6](#_Toc28023230)

[2.5. Створення об'єктів і розробка головної програми 7](#_Toc28023231)

[2.6. Опис файлів даних та інтерфейсу програми 7](#_Toc28023232)

[Розділ 3. Тестування програми і результати її виконання 8](#_Toc28023233)

[3.1. Інструкція користувача 8](#_Toc28023234)

[Висновки 10](#_Toc28023235)

[Список використаних джерел 11](#_Toc28023236)

[Додатки 12](#_Toc28023237)

[Додаток А. Лістинг програми 12](#_Toc28023238)

Вступ

Відеоігри на сьогоднішній день є повноцінним пластом культури. Уже давно вони стали окремим видом мистецтва, оскільки на розробку однієї гри потрібна ціла команда талановитих розробників: кодерів, дизайнерів, а також людей, які будуть займатись маркетингом фінального продукту.

Найперша відеогра була розроблена 1947 року. Її суть полягала в імітації управління ракетою, інтерфейс був у вигляді радар часів Другої світової війни. Вслід за цією грою почали з’являтись і інші: «Хрестики-нулики», «Віртуальний теніс» та багато других. Серед ігор того часу є і багато тих, котрі в наш час вважаються уже класичними та культовими. Такою грою став і тетріс.

Тетріс – відеогра, що була розроблена радянським програмістом Олексієм Пажитновим 6 червня 1984 року. Сама гра є головоломкою, що базується на використанні геометричних фігур, що складаються з чотирьох квадратів. Найперший «тетріс» був написаний мовою програмування «Паскаль» для комп’ютера «Электроника-60».

Метою цієї курсової роботи є здобути навички розробки ігор мовою програмування «С++». Основою буде алгоритмізація гри та побудова примітивного дизайну в консолі.

Розділ 1. Формулювання вимог до програми

**1.1. Аналіз технічного завдання**

В основі завдання лежить розробка повноцінної гри «Тетріс» мовою програмування «С++». Для цього потрібно побудувати алгоритм роботи гри та розробити дизайн поля гри та блоків, якими буде керувати користувач.

**1.2. Обґрунтування алгоритму і структури програми**

Алгоритм роботи розробленої програми полягає в роботі багатьох функцій, кожна з яких відповідає за певну дію. В першу чергу був розроблений прототип функції руху точки по екрану для більш зручного виводу будь-яких повідомлень, а також самого інтерфейсу гри. За допомогою цього прототипу та функцій, що фіксують натискання клавіш на клавіатурі і буде працювати сама гра. Так само були розроблені функції генерації випадкових фігур, що були задані у вигляді масиву матриць, та видалення уже зібраного ряду на ігровому полі. Майже всі ці функції працюють за допомогою подвійних циклів, які роблять пробіги по ігровому полю, яке, як було вище зазначено, також є двомірним масивом елементів типу «char». Окремо було розроблене стартове меню, яке є змінною типу «const string». Таким ж є головне меню гри, яке дозволяє нам почати гру, переглянути керування або вийти з гри, та саме меню, де детально описано керування в самій грі. Також алгоритм роботи коду програми можна навести за допомогою блок-схеми «рис. 1.1.»:

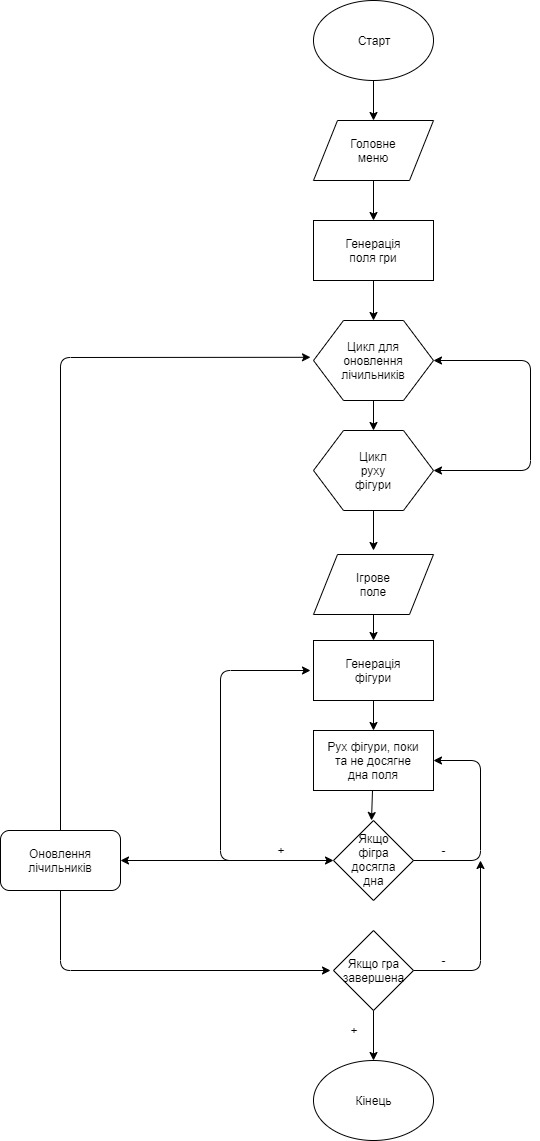


Рисунок 1.1. Блок-схема роботи коду програми

Розділ 2. Проектування та практична реалізація розв’язку поставленої задачі

**2.1. Оцінка коректності алгоритму та аналіз алгоритму на часову та просторову ефективність**

Алгоритм цілком і повністю задовольняє всі вимоги програми «2.2.». В ньому передбачені всі можливості, які є в повноцінній грі «Тетріс», наприклад: можливості повороту фігур, виведення інформації про наступну фігуру та навіть функція паузи в самій грі.

Щодо просторової та часової ефективності, то алгоритм не є ресурсозатратним, оскільки це більш-менш примітивна консольна програма, отже побудова алгоритму компілятором не займає багато часу.

**2.2. Розробка специфікації**

Фігура повинна автоматично рухатись вниз, а користувачу повинна бути надана можливість керувати нею по полю. Фігура повинна залишитись на тій позиції в самому низу поля, куди її помістив користувач; коли з фігур буде складено ряд, цей ряд повинен автоматично знищитись, а на його місце повинні стати фігури, що були над ним. Гра повинна завершитись, коли фігури з дна поля досягнуть його стелі, або за власним бажанням користувача.

**2.3. Опис програми (обґрунтування вибору мови програмування, опис основних процедур та функцій).**

Програма була написана мовою програмування «С++» згідно до поставлених вимог.

Для коректної роботи програми були використані функції, що містились у використаних бібліотеках, а також були розроблені власні функції.

У основі програми лежить процедура роботи подвійного циклу, у якому уже використовуються вищевказані функції, лічильники та робота з ігровим полем.

Використані стандартні функції**:**

* kbhit() – функція з бібліотеки «conio.h». Фіксує сам факт того, що було здійснене натискання клавіші на клавіатурі;
* getch() – функція з бібліотеки «conio.h». Передає код натиснутої клавіші у змінну. У даній програмі спрацьовує при умові, що спрацювала попередня функція;
* rand() – функція отримання випадкового значення вказаного типу;
* srand(time(NULL)) – функція, яка використана для того, щоб при кожному запуску програми була різна випадкова послідовність фігур;
* GetStdHandle() – функція, що є дескриптором пристрою стандартного виводу (простір імен «std»);

Використані створені функції:

* gotoxy() – прототип функції переміщення будь-якого об’єкту виводу в будь-яку точку консолі. Містить в собі вищезазначену стандартну функцію «GetStdHandle»;
* print() – функція заповнення та виводу самого ігрового поля та поля «рахунок» поруч з ним. Поле виводиться з допоміжної змінної, яка є буфером обміну даної функції. Це робиться для того, щоб не сталось ніяких помилок при заповненні. Також цю функція використовує попередню розроблену функцію для коректного виводу поля «рахунок» збоку від самого ігрового поля;
* printnextmap() – функція, що виводить прямо під полем «рахунок» як фігура буде наступною (не без допомоги функції «gotoxy»);
* getkey() – функція, що в подальшому дозволить використовувати клавішу «р» (англійська розкладка) як кнопку паузи в самій грі;
* valnewpos() – булева функція, що перевіряє чи потрібно фігури рухатись далі вниз, чи вона вже досягла дна на даному етапі гри;
* rotatemap() – функція повороту фігури;
* rnd() – функція рандомізації (щоб кожного разу була нова випадкова фігура);
* sleep() – функція, що відповідає за швидкість гри та затримку консолі;
* deleteline() – функція видалення уже зібраного рядка (з додатково розробленою анімацією);
* createmap() – функція створення фігури на полі, відповідно до значень матриць, заданих окремим глобальним масивом;
* clearscreen() – функція очищення екрану;
* startgame() – функція старту самої гри. В цій функції налаштоване все керування грою за допомогою конструкцій «switch-case». Дана функція зібрала в собі майже всі попередньо вказані функції та майже всі глобальні змінні;
* gamemenu() – функція виведення головного меню у вигляді таблиці-інструкції.

**2.4. Опис змінних, та їх ідентифікаторів, що використовувались у програмі**

Однією зі складових гарного коду програми є те, що код описує сам себе. Для цього і підбираються імена для кожної змінної так, щоб користувачу чи іншому програмісту було зрозуміло що за який процес відповідає.

Для розв’язку поставленої задачі в основному використовувались глобальні змінні константного виду (для запобігання їх зміни по ходу кодування програми).

Використані змінні:

* SIZEX, SIZEY – глобальні змінні типу «const int», які відповідають за, відповідно, висоту та ширину ігрового поля;
* screen[SIZEX][SIZEY] – глобальна змінна типу «char», що, по суті, і є ігровим полем;
* FMAPS\_CONT – глобальна змінна типу «const int», що є кількісною характеристикою кількості розроблених фігур;
* KEY\_UP, KEY\_DOWN, KEY\_LEFT, KEY\_RIGHT, KEY\_SPACE, KEY\_ESC, KEY\_ENTER – глобальні змінні типу «const int», в які записані коди зазначених в ідентифікаторах клавіш клавіатури;
* SCR\_SP – глобальна змінна типу «const int», в яку записаний код символу, яким буде заповнений фон ігрового поля (в даному випадку 183 – крапка);
* SCR\_OB – глобальна змінна типу «const int», в яку записаний код символу, з якого будуть складатись ігрові фігури (в даному випадку 14 – нота);
* map[4][4] – глобальна змінна типу «const int», в яку у вигляді нулів та одиниць буде записаний вигляд ігрової фігури;
* GAME\_TITLE – глобальна змінна типу «const string», що є простим титульним рядком, в якому вказана назва гри та ім’я розробника.

**2.5. Створення об'єктів і розробка головної програми**

Серед об’єктів програми можна виділити ігрове поле та блок. Для коректної роботи кожного з цих об’єктів було написано по декілька функцій, які детально описані в «2.3.». Не зважаючи на велику кількість зроблених дій, головна програма є досить простою. Вона включає в себе виведення титульного рядка, фіксування натискань клавіш клавіатури та використання функції «gamemenu», яка включила в собі абсолютно всі глобальні змінні та розроблені функції.

**2.6. Опис файлів даних та інтерфейсу програми**

Весь код до даної програми був написаний на одному файлі формату «cpp», що має назву «Source».

Інтерфейс програми складається зі стартового меню та самої гри.

Стартове меню включає в себе титульний рядок, таблицю з командами для вибору подальшої дії. Коли вибрана дія «Почати гру» програма переносить користувача до ігрового поля, де представлений звичайний «Тетріс».

Розділ 3. Тестування програми і результати її виконання

**3.1. Інструкція користувача**

Оскільки гра «тетріс» функціонує за допомогою натискань клавіш клавіатури, то в інструкцію користувача включена таблиця користування «табл. 3.1.», в якій описані клавіші, що виконують відповідні функції.

|  |  |
| --- | --- |
| Клавіша на клавіатурі | Виконувана функція |
| пробіл | поворот фігури |
| стрілка вгору | поворот фігури |
| стрілка вниз | спуск фігури на дно поля |
| стрілка вліво | зміщення рухомої фігури вліво |
| стрілка вправо | зміщення рухомої фігури вправо |
| Esc | завершення роботи програми |
| Enter | введення |

Таблиця 3.1. Керування

Свою роботу програма починає зі стартового меню «рис. 3.1.», в якому користувачу пропонується розпочати роботу з програмою за допомогою натискання будь-якої клавіші.

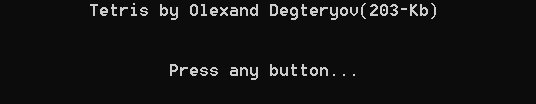


Рисунок 3.1. Стартове меню

Після того як користувач розпочав роботу з програмою, він потрапляє на головне меню «рис. 3.2.», де він може розпочати гру, переглянути керування в самій програмі, або завершити роботу з нею.



Рисунок 3.2. Головне меню

Для зручності було розроблене меню керування «рис. 3.3.», на якому також розписане керування в самій грі. Так само, як і в таблиці 3.1. розписано клавіші клавіатури та відповідні функції, що вони виконують.



Рисунок 3.3. Меню керування

Висновки

В даній курсовій роботі було розроблена комп’ютерна відеогра «Тетріс». Для виконання поставленої задачі були розкриті такі етапи:

* аналіз та опис предметної області;
* блок-схема програми, яка детально розкриває роботу алгоритму;
* кодування гри;
* аналіз результатів;

Дані недоліки не є принциповими до дисципліни «Об’єктно орієнтовне програмування», тому можна вважати, що завдання виконано.

Програму можна використовувати як повноцінну відеогру і приємно проводити час. Програма не займає багато місця на жорсткому диску і не є ресурсозатратною, отже не є вибагливою до потужного комп’ютера.

При оформленні курсової роботи були одержані навички написання програмного забезпечення, а також додатковий досвід роботи з мовою програмування «С++» у середовищі Microsoft Visual Studio.

Теоретичні відомості були закріплені практичною роботою.

Список використаних джерел

1. Уоррен Г. Алгоритмічні трюки для програмістів / Генрі Уоррен. Бостон: Addison-Wesley, 2014. – 509 с. (2).
2. Вступ до алгоритмів / Т. Кормен, Ч. Лейзерсон, Р. Рівест, К. Стайн., 2019. – 1216 с. (3).
3. Бентлі Д. Перли програмування / Джон Бентлі. Бостон: Addison-Wesley, 2000. (2).
4. Поляков К. Програмування мовою "С" / К. Поляков. Київ, 2009. – 45 с.
5. Йен Д. Алгоритм знаходження найкоротшого шляху в загальній мережі / Джин Йен. Маямі: Mathematical Reviews, 1971. – 530 с.
6. Ковалюк Т. В. Основи програмування / Т. В. Ковалюк. Київ: BHV, 2005. – 385 с.
7. Львов М. С. Основи алгоритмізації та програмування / М. С. Львов, О. В. Співаковський. Херсон, 1997. – 256 с.

Додатки

**Додаток А. Лістинг програми**

#include <iostream>

#include <conio.h>

#include <Windows.h>

#include <time.h>

#include <random>

using namespace std;

//====================================Глобальні змінні==============================================

const int KEY\_UP = 72; // код клавіші "вгору"

const int KEY\_DOWN = 80; // код клавіші "вниз"

const int KEY\_LEFT = 75; // код клавіші "вліво"

const int KEY\_RIGHT = 77; // код клавіші "вправо"

const int KEY\_SPACE = 32; // код пробілу

const int KEY\_ESC = 27; // код клавіші "ESC"

const int KEY\_ENTER = 13; // код клавіші "Enter"

const int SIZEX = 16; // Ширина поля (по х)

const int SIZEY = 20; // Довжина поля (по у)

const int FMAP\_COUNTS = 7; // Кількість визначених фігур

const char SCR\_SP = (char)46; // Заповнення поля символами з кодом 183 (крапка)

const char SCR\_OB = (char)14; // Надання фігурі форми за допомогою коду 14 (нота)

int screen[SIZEX][SIZEY] = { 0 }; // Розмірність екрану

int map[4][4]; // Фігура

int px(0), py(0), score(0), nextmap(0); // Змінні для переміщення по осям та рахунок

const string GAME\_TITLE = "\t\t\tTetris by Olexand Degteryov(203-Kb)\n\n\n"; // Титулка програми

const int fmap[FMAP\_COUNTS][4][4] = // визначення фігур за допомогою масиву матриць

{

{

{1, 1, 0, 0},

{1, 1, 0, 0},

{0, 0, 0, 0},

{0, 0, 0, 0}

},

{

{1, 0, 0, 0},

{1, 0, 0, 0},

{1, 0, 0, 0},

{1, 0, 0, 0}

},

{

{0, 0, 1, 0},

{1, 1, 1, 0},

{0, 0, 0, 0},

{0, 0, 0, 0}

},

{

{1, 1, 1, 0},

{0, 0, 1, 0},

{0, 0, 0, 0},

{0, 0, 0, 0}

},

{

{0, 1, 1, 0},

{1, 1, 0, 0},

{0, 0, 0, 0},

{0, 0, 0, 0}

},

{

{1, 1, 0, 0},

{0, 1, 1, 0},

{0, 0, 0, 0},

{0, 0, 0, 0}

},

{

{1, 1, 1, 0},

{0, 1, 0, 0},

{0, 0, 0, 0},

{0, 0, 0, 0}

}

};

//================================================================================================

// ==============================ФУНКЦІЇ===============================

void gotoxy(int xpos, int ypos) // Прототип функції переміщення фігури в конкретну точку

{

COORD scrn; // Координата екрану

HANDLE hOuput = GetStdHandle(STD\_OUTPUT\_HANDLE);// Створення допоміжного засобу для виводу будь-якого з меню

scrn.X = xpos;

scrn.Y = ypos;// Присвоєння значень координат

SetConsoleCursorPosition(hOuput, scrn);// Переміщення

}

void print() // Виведення

{

int i, j; // Лічильники для циклів

int buff[SIZEX][SIZEY]; // Копіювання поля в буфер обміну (допоміжну змінну)

for (i = 0; i < SIZEY; i++)

for (j = 0; j < SIZEX; j++)

buff[j][i] = screen[j][i]; // Виведення поля з буфера

for (i = 0; i < 4; i++)

for (j = 0; j < 4; j++)

if (map[j][i]) buff[j + px][i + py] = 1; // Якщо доступна тільки частина фігури, то позначити її як 1

gotoxy(0, 0); // Перейти в точку з координатами 0, 0

for (i = 0; i < SIZEY; i++)

{

for (j = 0; j < SIZEX; j++)

putchar(buff[j][i] == 0 ? SCR\_SP : SCR\_OB); // Якщо елемент, що перевіряється не є частиною буфера, то він заповнюється фоном поля

putchar('\n');

}

gotoxy(SIZEX + 1, 0); // Відійти вбік

cout << "Score: " << score; // Кількість набраних очок

}

void printnextmap() // Наступна фігура

{

gotoxy(SIZEX + 1, 2); // Відведення даного поля вбік за допомогою функціїї переміщення

cout << "Next: ";

for (int i = 0; i < 4; i++)

{

gotoxy(SIZEX + 2, i + 3); // Відведення далі вбік

for (int j = 0; j < 4; j++)

putchar(fmap[nextmap][j][i] == 0 ? ' ' : SCR\_OB); // Сама фігура (на місці нулів у визначникові ставляться пробіли для коректного виведення)

}

}

int getkey()// Можливість поставити паузу

{

if (\_kbhit())

{

int c;

if ((c = \_getch()) == 224) // Код клавіші "р"(англ.), яка і відповідає за паузу

c = \_getch();

return c;

}

return false;

}

bool valnewpos(int x, int y) // Функція, чи необхідно фігурі рухатись далі вних

{

if (x < 0)

return false;

for (int i = 0; i < 4; i++)

for (int j = 0; j < 4; j++)

if (map[j][i])

{

if ((j + x >= SIZEX) || (i + y >= SIZEY))

return false;// Якщо фігура зайла за межі екрану

if (screen[j + x][i + y])

return false;

}

return true;// Якщо ні, то фігура може рухатись далі вниз

}

int inv(int x) // Допоміжна функція для вісі х

{

return 3 - x;

}

void rotatemap() // Функція повороту фігури

{

int \_map[4][4];

int sx = 4, sy = 4;

for (int i = 0; i < 4; i++)

for (int j = 0; j < 4; j++)

{

\_map[j][i] = map[j][i];// Створення копії фігури для того, щоб повернути її початковий вигляд

if (map[j][i])// Цикл запам'ятовування координат фігури, який не дає їй при повороті вийти за межі

{

if (i < sx) sx = i;

if (inv(j) < sy) sy = inv(j);

}

map[j][i] = 0;

}

for (int i = 0; i < 4; i++)

for (int j = 0; j < 4; j++)

if (\_map[inv(i)][j]) map[j - sx][i - sy] = 1;// Виведення уже перевернутої фігрури

if (!valnewpos(px, py))

for (int i = 0; i < 4; i++)

for (int j = 0; j < 4; j++)

map[j][i] = \_map[j][i];// Повернення фігури у початковий вигляд

}

int rnd(int max) // Функція, що відповідає за випадковість випадання фігур

{

max++;

srand(time(NULL));

return rand() % max;

}

void sleep(int milsec) // Затримка консолі для того, щоб кожен крок фігури був зафіксований

{

clock\_t t = clock();

while (clock() - t < milsec);

}

void deleteline() // Функція видалення уже зібраної лінії

{

int i, j, k, cl;

for (i = SIZEY - 1; i >= 0; i--)

{

cl = 1;

for (j = 0, cl = 1; j < SIZEX; j++)

{

if (!screen[j][i])

cl = 0;

}

if (cl)

{

// Анімація

gotoxy(0, i);

for (k = 0; k < SIZEX; k++)

putchar('\_'), sleep(20);

score += (((i \* (-1)) + SIZEY) \* 10); // Коли лінія зібрана, збільшити кількість набраних очок

for (k = i; k > 0; k--)

for (j = 0; j < SIZEX; j++)

screen[j][k] = screen[j][k - 1]; // Сам процес видалення

i++;

print();

}

}

}

void createmap() // Створення фігури на полі

{

for (int i = 0; i < 4; i++)

for (int j = 0; j < 4; j++)

map[j][i] = fmap[nextmap][j][i];

py = 0;

px = SIZEX / 2;

nextmap = rnd(FMAP\_COUNTS - 1);

printnextmap();

}

void clearscreen() // Очистка екрану

{

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

for (int j = 0; j < SIZEX; j++)

screen[j][i] = 0;

}

void createrndscreen() // Створення випадкового екрану

{

clearscreen();

int rn = rnd(10);

for (int i = SIZEY - 1; i >= (SIZEY - 1) - rn; i--)

for (int j = 0; j < SIZEX; j++)

screen[j][i] = rnd(1);

}

void startgame() // Сама гра

{

time\_t tm;

system("cls");

px = SIZEX / 2;// Значення для падіння фігури по координаті х, виходячи з її центру

py = 0;

score = 0;

tm = clock();

nextmap = rnd(FMAP\_COUNTS - 1);// Наступна випадкова фігура

createmap();// Створення фігури

while (true)

{

int c = getkey();

switch (c)

{

case KEY\_UP: // Коли нажата клавіша "вгору"

rotatemap(); //поворот

break;

case KEY\_SPACE: // Коли нажата клавіша "пробіл"

rotatemap(); //поворот

break;

case KEY\_DOWN: // Коли нажата клавіша "вниз"

for (; valnewpos(px, py + 1); py++);

for (int i = 0; i < 4; i++)

for (int j = 0; j < 4; j++)

if (map[j][i]) screen[px + j][py + i] = 1; // Моментально опустити фігуру донизу

print();

deleteline();// Перевірка лінії на заповненість

createmap();// Нова фігура

break;

case KEY\_LEFT: // Коли нажата клавіша "вліво"

if (valnewpos(px - 1, py))

px--;// Зменшуєм значення координати Х

break;

case KEY\_RIGHT: // Коли нажата клавіша "вправо"

if (valnewpos(px + 1, py)) px++;// Збільшуєм значення координати У

break;

case 'p':// Пауза

\_getch();

break;

case KEY\_ESC:// Вихід

return;

}

if ((clock() - tm) > 720)// Час, за який падає фігура

{

tm = clock();

if (!(valnewpos(px, py + 1)))

{

for (int i = 0; i < 4; i++)

for (int j = 0; j < 4; j++)

if (map[j][i]) screen[px + j][py + i] = 1;

createmap();

deleteline();

}

else

py++;// Поступове падіння фігури вниз

}

print();

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

if (screen[i][0])// Коли координата У фігури і поля співпали, то поле заповнене і гра завершена

{

system("cls");

gotoxy(2, 8);

cout << "GAME OVER";

return;

}

}

}

void gamemenu() // Ігрове меню

{

int p = 1, c = 0;

const char\* GAME\_MENU = " +===============================+\n"

" | 1. START |\n"

" +===============================+\n"

" | 2. CONTROL MENU |\n"

" +===============================+\n"

" | 3. EXIT |\n "

" +===============================+";

system("cls");

cout << GAME\_TITLE << GAME\_MENU;

while (true) // Навігація по меню

{

c = \_getch();

cout << "\a";

switch (c)

{

case '1':

case '2':

case '3':

case '4':

p = c - '0';

case KEY\_ENTER:

switch (p)

{

case 1:

clearscreen();

startgame();

gotoxy(0, SIZEY);

cout << "Press ESC to see main menu...\n";

cout << "\a";

while (\_getch() != KEY\_ESC);

break;

case 2:

system("cls");

cout << GAME\_TITLE <<

"+=======+=======+\n"

"|\x1B |LEFT |\n"

"|\x1A |RIGHT |\n"

"|\x19 |DOWN |\n"

"|\x18/Space|ROTATE |\n"

"|P |PAUSE |\n"

"|ESC |EXIT |\n"

"+=======+=======+\n\n"

"Press...\n";

\_getch();

cout << "\a";

if (\_kbhit())

\_getch();

break;

case 3:

return;

}

system("cls");

cout << GAME\_TITLE << GAME\_MENU;

p = 1;

break;

case KEY\_UP:

if (p > 1) p--;

break;

case KEY\_DOWN:

if (p < 4) p++;

break;

case KEY\_ESC:

return;

}

}

}

//==================================================================

int main()

{

cout << GAME\_TITLE;

cout << "\t\t\t\tPress any button...";

\_getch();

cout << "\a";

if (\_kbhit())

\_getch();

gamemenu();

return 0;

}