Лабораторная работа 12

По дисциплине «Основы программной инженерии»

На тему «Технологии разработки ПО. Командная работа над проектом»

Выполнили:

Студенты 1 курса 9 группы

Хуторцов Кирилл Владимирович

Милочкин Илья Игоревич

Яговдик Владислав Владимирович

Семуков Евгений Дмитриевич

Филипчик Иван Владимирович

Прокопович Александр Дмитриевич

Преподаватель: Архипенко О.А.

2023, Минск

* **Введение**
* **Цель**

Создание трёх консольных игр на языке C++. Цель – объединить три консольные игры в одном навигационном меню. При создании данного проекта мы использовали навыки, полученные во время обучения.

**Задачи**

-Разработать три игры в консоли.

-Создать игровой алгоритм.

- При разработке использовать известные для нас команды ЯП С/С++;

- При необходимости, дополнительно изучить команды, которые помогут реализовать приложение.

* **План разработки приложения**

-Создать примитивный консольный интерфейс к игре и оформление.

-Разработать:

* Правила игры
* Графическое представление в консоли
* Логику приложения
* Выход из приложения, в зависимости от результата.
* **Меню**

Меню представляет собой стандартную консоль windows с изменённой цветовой палитрой.

**Игра 1**

**2048**

**Игровой процесс**

При запуске игры перед нами появляется консоль. Для начала игры у нас просят ввести любой символ, или предоставляется возможность еще до начала игры завершить работу приложения.   
 После нажатия любой кнопки для начала игры, перед игроком появляются короткий свод правил и игровое поле 4х4 клеточки.

Правила игры:

* В каждом раунде появляется плитка номинала «2» (с вероятностью 90%) или «4» (с вероятностью 10%).
* Нажатием кнопок управления игрок может скинуть все плитки игрового поля в одну из 4 сторон. Если при сбрасывании две плитки одинакового номинала «налетают» одна на другую, то они превращаются в одну, номинал которой равен сумме соединившихся плиток. После каждого хода на свободной секции поля появляется новая плитка номиналом «2» или «4». Если при нажатии кнопки местоположение плиток или их номинал не изменится, то ход не совершается.
* За каждое соединение игровые очки увеличиваются на номинал получившейся плитки.
* Игра заканчивается победой, если игрок получает плитку номиналом «2048» или, если после очередного хода невозможно совершить действие.

**Техническая документация к игре**

Пользовательская история для проекта:

Как **пользователь** я хочу **использовать «2048\_game»**, чтобы развивать логическое мышление.

Диаграмма вариантов использования для проекта:



**Модуль main\_2048**

Пользовательская история для main\_2048:

Как **пользователь** я хочу, чтобы при входе в приложение **было меню** входа/выхода из игры, чтобы можно было управлять приложением независимо от чего-либо.

Диаграмма вариантов использования для main\_2048:



**Модуль display**

Пользовательская история для display:

Как **пользователь** я хочу, чтобы **было графическое представление игрового поля**, для того чтобы визуально приятно использовать приложение.

Диаграмма вариантов использования для display:



**Модуль game\_process**

Пользовательские истории для game\_process:

Как **пользователь** я хочу, чтобы **было легко осуществляемое управление** игровым процессом, чтобы легко управлять игровым процессом.

Как **пользователь** я хочу, чтобы **при случайном нажатии неправильной кнопки результат игрового процесса не менялся**, чтобы можно было беспроблемно продолжить игру.

Как **пользователь** я хочу, чтобы **была возможность завершения игры**, чтобы при необходимости досрочно завершить игру.

Диаграмма вариантов использования для game\_process:



**Игра 2**

**Змейка**

**Игровой процесс**

При запуске игры перед нами появляется консоль и начинается игра.

Правила игры:

* Появляется змейка и фрукт.
* Нажатием кнопок управления игрок может управлять двидение змейки.
* После того, как змейка съест фрукт, она увеличится в размерах, а новый фрукт появится в новом месте. Результат пишется выше.
* Игра заканчивается тогда, когда змейка столкнётся с собой или с границей поля.

**Техническая документация к игре**

Пользовательская история для проекта:

Как **пользователь** я хочу **использовать «snake\_game»**, чтобы развивать логическое мышление.



А также , как **пользователь** я хочу, чтобы **было графическое представление игрового поля**, для того чтобы визуально приятно использовать приложение.



Как **пользователь** я хочу, чтобы **было легко осуществляемое управление** игровым процессом, чтобы легко управлять игровым процессом.

Как **пользователь** я хочу, чтобы **при случайном нажатии неправильной кнопки результат игрового процесса не менялся**, чтобы можно было беспроблемно продолжить игру.



**Игра 3**

**Tetris**

**Структура программы**

**Основные переменные**

- bool gameOver: Переменная, отвечающая за состояние игры (true - завершена, false - продолжается).

- const int width: Ширина игрового поля.

- const int height: Высота игрового поля.

- int score: Очки игрока.

- int field[18][16]: Двумерный массив, представляющий игровое поле.

- int shapes[7][4][2]: Массив фигур Тетриса.

**Структура Point**

- struct Point: Структура, представляющая точку с координатами x и y.

**Основные функции**

- 1.bool CheckCollision(int x, int y, int shape)

- Описание:

- Это функция CheckCollisionдля проверки возможности текущего перемещения фигуры по заданным координатам (x, y)на игровом поле. Она проверила на коллизии с границами полей и уже установленными блоками.

-

- Параметры:

- int x: Координата x куда, удалось переместить текущую фигуру.

- int y: Координата и куда мы хотим переместить текущую фигуру.

- int shape: Номер текущей фигуры.

- Возвращаемое значение:

- true, если обнаружена коллизия (нельзя переместить фигуру в заданные координаты).

- false, если коллизии не обнаружены (возможно переместить фигуру).

- 2.void ClearLines()

- Описание:

- Функция ClearLinesотвечает за проверку и удаление всех заполненных горизонтальных линий на игровом поле. После удаления линий происходит сдвиг верхних линий вниз, и игрок получает очки за каждую удаленную линию.

-

- Параметры:

- Отсутствуют.

- Возвращаемое значение:

- Отсутствует.

-

- 3.void Draw()

- Описание:

- Функция Drawотвечает за обеспечение текущего состояния игрового поля в консоли. Она выводит на экран текущую фигуру, уже установленные блоки и границы игрового поля.

-

- Параметры:

- Отсутствуют.

- Возвращаемое значение:

- Отсутствует.

-

- 4.void AddToField(int x, int y, int shape)

- Описание:

- AddToFieldпредоставление текущей фигуры к уже установленным блокам на игровом поле после того, как она коснулась дна поля и не смогла двигаться дальше .

-

- Параметры:

- int x: Координата x верхний левый угол текущей фигуры.

- int y: Координата и верхний левый угол текущей фигуры.

- int shape: Номер текущей фигуры.

- Возвращаемое значение:

- Отсутствует.

-

- 5.void RotateShape()

- Описание:

- Функция RotateShapeотвечает за изменение текущей ситуации вокруг своего центра. Перед поворотом происходит проверка возможности поворота, чтобы избежать столкновений с границами полей и уже установленными блоками.

-

- Параметры:

- Отсутствуют.

- Возвращаемое значение:

- Отсутствует.

-

- 6.void PlaceNewShape()

- Описание:

- Назначение PlaceNewShapeразмещает новую случайную фигуру в верхней части игрового поля. Перед размещением происходит проверка международных коллизий на наличие новых фигур.

-

- Параметры:

- Отсутствуют.

- Возвращаемое значение:

- Отсутствует.

-

- 7.void Update()

- Описание:

- Функция Updateобновляет состояние игры. Она отвечает за движение текущей фигуры вниз, проверяя при этом наличие коллизий. В случае достижения дна фигуры добавляются к установленным блокам, происходит удаление заполненных линий и размещение новых фигур.

-

- Параметры:

- Отсутствуют.

- Возвращаемое значение:

- Отсутствует.

-

- 8.void Input()

- Описание:

- Функция Inputобрабатывает вводы с помощью кнопок для управления текущей фигурой. Она реагирует на клавиши A (влево), D (вправо падение), S (ускоренное падение), W (поворот) и X (завершение игры).

-

- Параметры:

- Отсутствуют.

- Возвращаемое значение:

- Отсутствует.

-

- 9.int main()

- Описание:

- Функция mainявляется основным режимом программы. Она придумывает игру, запускает цикл обновления и отображения, обрабатывает ввод пользователя и завершает игру по условию завершения.

- Параметры:

- Отсутствуют.

- Возвращаемое значение:

- Целочисленное значение, обозначающее успешное завершение программы.

**Игровой процесс**

1. Игрок начинает игру с пустого поля.

2. Случайная фигура появляется в верхней части поля.

3. Игрок управляет фигурой, перемещая, вращая и ускоряя ее падение.

4. Когда фигура достигает нижней границы или касается другой фигуры, она остается на месте.

5. Если заполняется горизонтальная линия, она удаляется, и игрок получает очки.

6. Игра завершается, когда новая фигура не может быть размещена в верхней части поля.

**Управление**

- A: Движение фигуры влево.

- D: Движение фигуры вправо.

- S: Ускоренное падение фигуры.

- W: Поворот фигуры.

- X: Завершение игры.

**Настройки игры**

- frameDuration: Длительность одного кадра в миллисекундах (регулируется для изменения скорости игры).

**Завершение игры**

Игра завершается, когда новая фигура не может быть размещена в верхней части поля. После завершения игры выводится итоговый счет.