МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования «Балтийский федеральный университет имени
Иммануила Канта»

ОНК «Институт высоких технологий»

ОТЧЁТ О ПРОХОЖДЕНИИ УЧЕБНОЙ ТЕХНОЛОГИЧЕСКОЙ (ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ) ПРАКТИКИ

на базе Высшей школы компьютерных наук и искусственного интеллекта

Выполнил Бычков Михаил Владиславович

_____

студент очной формы обучения 1 курса
направления подготовки 02.03.03 «Математическое обеспечение
и администрирование операционных систем»
профиль обучения «Разработка баз данных
и интернет-приложений»

Руководитель практики
Ассистент _____ Полковский О.А.

г. Калининград
2025 г.

Оглавление

ВВЕДЕНИЕ

Вид практики – Учебная технологическая (проектно-технологическая) практика (далее Учебная практика).

Цель учебной практики: получение первичных профессиональных умений навыков.

Задачи учебной практики:

1. Закрепление и углубление теоретических знаний в области информационных технологий;
2. Приобретение и развитие первичных профессиональных навыков и умений в области прикладной математики и информатики.

ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Условия и Циклы

Условные операторы в C++ используются для выполнения различных действий в зависимости от выполнения определенного условия.Основные конструкции для работы с условиями:

1.**if:** Позволяет выполнить блок кода, если заданное условие истинно.

2.**else:** Используется в паре с оператором if, чтобы определить блок кода, который выполнится,если условие ложное.

3.**else if:** Позволяет проверить несколько условий последовательно.

4.**switch:** Используется для выбора одного из множества вариантов.

Циклы позволяют многократно выполнять один и тот же блок кода,что особенно полезно при работе с массивами или при необходимости повторить действия,пока выполняется определенное условие.Основные конструкции циклов:

1.**for:** Используется для выполнения блока кода фиксированное количество раз.

2.**while:** Выполняет блок кода, пока условие истинно.Условие проверяется перед каждой итерацией.

3.**do while:** Похож на цикл while, но условие проверяется после выполнения блока кода, что гарантирует, что код выполнится хотя бы один раз.

Массивы

Массивы — это один из основных типов данных в языке программирования C++, который позволяет хранить несколько значений одного типа в одной переменной. Они представляют собой последовательность элементов, доступ к которым осуществляется по индексу.

Массив в C++ определяется с указанием типа элементов, количества элементов и инициализации.

Например, для создания массива из 5 целых чисел можно использовать следующий код:

int numbers[5];

Массивы можно инициализировать при их объявлении. Например:

int numbers[5] = {1, 2, 3, 4, 5};

Если размер массива не указан, то он будет автоматически определён по количеству инициализируемых значений:

int numbers[] = {1, 2, 3, 4, 5}; // Компилятор сам определит размер как 5

Элементы массива доступны по их индексу, который начинается с нуля. Для доступа к первому элементу массива используется индекс 0, для второго — индекс 1 и так далее. Например:

int first = numbers[0]; // Получаем первый элемент массива

numbers[2] = 10; // Изменяем третий элемент массива на 10

C++ также поддерживает многомерные массивы, которые позволяют хранить данные в виде таблиц. Наиболее распространённым является двумерный массив.

Например, для создания двумерного массива 3x3:

int matrix[3][3];

Доступ к элементам двумерного массива осуществляется с помощью двух индексов:

matrix[1][2] = 5; // Присваиваем значение 5 элементу во втором ряду и третьем столбце

ГЛАВА 2. ЗАДАНИЕ НА ПРАКТИКУ

Задача №1 “«Создание игрового поля»

Реализовать двумерный массив,который будет представлять игровое поле для игры в “Крестики-нолики”. Поле должно быть размером 3x3, и каждое поле должно хранить символ, представляющий состояние ячейки: пустая ячейка,’X’ или ‘O’.

Задача №2 «Создание объектов игрового поля, курсора, нолика и крестика»

Создать графические объекты,которые будут представлять элементы игры: курсор,нолик и крестик.Для этого необходимо использовать библиотеку SFML, которая позволяет создавать и управлять граффическими формами.Каждый из этих объектов имеет свои параметры и свойства.

Задача №3 «Отрисовка всех элементов игры: крестики, нолики, курсор »

Отрисовать все графические элементы на игровом поле.Необходимо реализовать функции, которые отвечают за отрисовку сетки,крестиков ноликов и курсора.

Задача №4 «Обработка событий»

Реализовать обработку событий. Игра должна корректно реагировать на действия пользователя.При нажатии клавиш WASD курсор перемещается по ячейкам, а при нажатии пробела игрок совершает ход. Поле очищается и игра начинается заново при нажатии на клавишу Escape.

ГЛАВА 3. ВЫПОЛНЕНИЕ ЗАДАНИЙ НА ПРАКТИКУ

Решение задачи №1

Для решения задачи по созданию игрового поля был использован двумерный массив размером 3x3, который хранит состояние каждой ячейки. Каждая ячейка инициализируется пробелом, что означает, что она пуста.

Реализация. В коде была создана переменная state, представляющая игровое поле(Рисунок *1*)

```
char state[3][3] = {
    {' ', ' ', ' '},
    {' ', ' ', ' '},
    {' ', ' ', ' '}
};
```

Рисунок 1.Двумерный массив

Решение задачи №2 «Создание объектов игрового поля, курсора, нолика и крестика»

Для создания графических объектов, представляющих элементы игры, использовалась библиотека SFML. Были созданы три объекта: circle для ноликов ('O'), line для крестиков ('X') и cursor для отображения текущего положения курсора.

Реализация. Объекты были созданы следующим образом(Рисунок *2*):

```
sf::CircleShape    circle;
sf::RectangleShape line;
sf::RectangleShape cursor;
```

Рисунок 2.Создание объектов

Каждый объект был настроен с помощью соответствующих методов, например, для круга было установлено количество точек(Рисунок *3*):

```
circle.setPointCount(60);
```

Рисунок 3.Настройка вершин

Сложности. Основной сложностью было правильно настроить размеры и цвета объектов, чтобы они хорошо вписывались в интерфейс игры и были визуально понятны игроку.

Решение задачи №3 «Отрисовка всех элементов игры: крестики, нолики, курсор, победная линия»

Для отрисовки всех элементов игры были реализованы несколько функций. Эти функции отвечали за отрисовку сетки, крестиков, ноликов и курсора на экране.

Реализация. Например, функция для отрисовки сетки была реализована следующим образом(Рисунок 4):

```
auto DrawGrid = [&] (float space, float width, float position_x, float position_y)
{
    line.setSize({space*3, width});
    line.setOrigin(line.getGeometricCenter());
    line.setFillColor(sf::Color(128, 128, 128));
    // horizontal
    line.setRotation(sf::degrees(0.f));
    line.setPosition({position_x, position_y - space/2});
    window.draw(line);
    line.setPosition({position_x, position_y + space/2});
    window.draw(line);
    // vertical
    line.setRotation(sf::degrees(90.f));
    line.setPosition({position_x - space/2, position_y});
    window.draw(line);
    line.setPosition({position_x + space/2, position_y});
    window.draw(line);
};
```

Рисунок 4.Сетка

А также были созданы функции для отрисовки 'X' и 'O'(Рисунок *5*,Рисунок *6*):

```
auto DrawX = [&] (float size, float width, float position_x, float position_y)
{
    line.setSize({size, width});
    line.setOrigin(line.getGeometricCenter());
    line.setFillColor(sf::Color(170, 255, 0));
    line.setPosition({position_x, position_y});
    line.setRotation(sf::degrees(45.f));
    window.draw(line);
    line.setRotation(sf::degrees(-45.f));
    window.draw(line);
};
```

Рисунок 5.Функция "X"

```
// lambda function - draw O
auto DrawO = [&] (float size, float width, float position_x, float position_y)
{
    circle.setRadius(size/2);
    circle.setOrigin(circle.getGeometricCente
    circle.setFillColor(sf::Color::Transparen (int)255
    circle.setOutlineColor(sf::Color(0, 170, 255));
    circle.setOutlineThickness(-width);
    circle.setPosition({position_x, position_y});
    window.draw(circle);
};
```

Рисунок 6.Функция "O"

Сложности. Основной сложностью было обеспечить правильное позиционирование всех элементов на экране, чтобы они не перекрывались и были видны игроку.

Решение задачи №4 «Обработка событий»

Для обработки событий в игре был реализован цикл, который реагировал на нажатия клавиш и другие действия пользователя. Это позволяло игроку управлять курсором и совершать ходы.

Реализация. В коде был использован следующий подход(Рисунок *7*):

```
while (std::optional event = window.pollEvent())
{
    // when close button is clicked
    if (event->is<sf::Event::Closed>())
    {
        // close window
        window.close();
    }

    // when window is resized
    else if (event->is<sf::Event::Resized>())
    {
        // update view
        sf::View view(sf::FloatRect({0.f, 0.f}, sf::Vector2f(window.getSize())));
        window.setView(view);
    }

    // when keyboard is pressed
    else if (auto* key = event->getIf<sf::Event::KeyPressed>())
    {
        // no winner yet
        if (winner == ' ')
        {
            // move cursor with WASD
            if      ((key->scancode == sf::Keyboard::Scancode::A) and (cursor_x > 0))  cursor_x--;
            else if ((key->scancode == sf::Keyboard::Scancode::D) and (cursor_x < 2))  cursor_x++;
            else if ((key->scancode == sf::Keyboard::Scancode::W) and (cursor_y > 0))  cursor_y--;
            else if ((key->scancode == sf::Keyboard::Scancode::S) and (cursor_y < 2))  cursor_y++;
```

Рисунок 7.Настройка WASD

Сложности. Основной сложностью было правильно реализовать логику обработки событий, чтобы все действия игрока корректно отражались на игровом поле и не вызывали ошибок.

Подробнее рассмотреть код и реализацию очистки поля через Escape можно в разделе “Приложение”.

ЗАКЛЮЧЕНИЕ

В ходе практики по созданию игры «Крестики-Нолики» были изучены основы языка программирования C++ и графической библиотеки SFML. Задачи были направлены на закрепление теоретического материала, связанного с графическим программированием, обработкой событий и управлением состоянием игры. Особое внимание было уделено работе с двумерными массивами для представления игрового поля, а также отрисовке графических объектов, таких как курсор, крестики и нолики.

В процессе работы над проектом я научился создавать и управлять графическими элементами, обрабатывать пользовательские события и реализовывать логику игры. Были изучены функции, позволяющие эффективно организовывать код и улучшать его читаемость. Также особое внимание было уделено отрисовке всех элементов игры и обеспечению корректного взаимодействия между ними.

В результате практики были усовершенствованы мои компетенции в области программирования, закреплены теоретические навыки работы с C++. Я научился выполнять поставленные задачи, изменять готовые решения по мере нахождения ошибок и искать альтернативные пути их решения. Также я освоил новую для меня среду программирования Visual Studio и изучил основы работы с системами контроля версий, такими как GitHub.

По мере прохождения учебной практики я выполнил несколько задач, в которых использовал полученные знания и закрепил навыки работы с C++. В течение практики задачи были успешно выполнены, а цели достигнуты. Это позволило мне значительно улучшить свои навыки программирования и получить практический опыт, который будет полезен в дальнейшей учебе и профессиональной деятельности.

СПИСОК ЛИТЕРАТУРЫ

Перечень учебной литературы ресурсов сети «Интернет», необходимой для проведения практики

1. Столяров, А. В. Программирование: Введение в профессию «Азы программирования» / Издание изд-ва ДМК Пресс, 2021 г. — 655 с. — ISBN 978-5-97060-945-3. – Текст: электронный. – URL: http://www.stolyarov.info/books/pdf/progintro_dmkv1.pdf (дата обращения: 02.07.2025). – Режим доступа: бесплатный
2. Шилдт Герберт, Программирование: C++ для начинающих. Шаг за шагом, пер. с англ. - М.: ЭКОМ Паблишерз, 2013. - 640 с.: ил. - ISBN: 978-5-9790-0127-2. Текст: электронный – URL: https://codelibs.ru/s-dlya-nachinayushih-shag-za-shagom/ (дата обращения: 02.07.2025). – Режим доступа: бесплатный

Перечень ресурсов информационно-телекоммуникационной сети «Интернет»

1. Основы программмирования C++ для начинающих – режим доступа: https://www.youtube.com/watch?v=kRcbYLK3OnQ&list=PLQOaTSbfxUtCrKs0nicOg2npJQYSPGO9r
2. Официальный сайт SFML с документацией и базовыми уроками – режим доступа: https://www.sfml-dev.org/tutorials/3.0/graphics/draw/\
3. Видеокурс: Написание игра на SFML – режим доступа: https://www.youtube.com/watch?v=odTop02dz0o&list=PLaZGZkAuB1Eth1p18HnXramTzkgJlEH_L
4. Видеоурок: Как создать игру Крестики-Нолики – режим доступа: https://youtu.be/ZGXvkHutgAQ?si=9Ie4qc34ivYdetP2

ПРИЛОЖЕНИЕ

```
1  #include <SFML/Graphics.hpp>
2
3
4  // main program
5  int main()
6  {
7      // state of spaces
8      char state[3][3] = {
9          {' ',' ',' '},
10         {' ',' ',' '},
11         {' ',' ',' '}
12     };
13
14     // position of cursor (index)
15     int cursor_x = 1;
16     int cursor_y = 1;
17
18     // turn
19     bool is_X_turn = true;
20
21     // winner
22     char winner = ' ';
23
24
25     // create window
26     sf::RenderWindow window(sf::VideoMode({600, 600}), "Title");
27
28     // create shapes
29     sf::CircleShape    circle;
```

```
30    sf::RectangleShape line;
31    sf::RectangleShape cursor;
32    circle.setPointCount(60);
33
34
35    // lambda function - draw grid
36    auto DrawGrid = [&] (float space, float width, float position_x, float position_y)
37    {
38        line.setSize({space*3, width});
39        line.setOrigin(line.getGeometricCenter());
40        line.setFillColor(sf::Color(128, 128, 128));
41        // horizontal
42        line.setRotation(sf::degrees(0.f));
43        line.setPosition({position_x, position_y - space/2});
44        window.draw(line);
45        line.setPosition({position_x, position_y + space/2});
46        window.draw(line);
47        // vertical
48        line.setRotation(sf::degrees(90.f));
49        line.setPosition({position_x - space/2, position_y});
50        window.draw(line);
51        line.setPosition({position_x + space/2, position_y});
52        window.draw(line);
53    };
54
55    // lambda function - draw X
56    auto DrawX = [&] (float size, float width, float position_x, float position_y)
57    {
58        line.setSize({size, width});
59        line.setOrigin(line.getGeometricCenter());
60        line.setFillColor(sf::Color(170, 255, 0));
```

```
61          line.setPosition({position_x, position_y});
62          line.setRotation(sf::degrees(45.f));
63          window.draw(line);
64          line.setRotation(sf::degrees(-45.f));
65          window.draw(line);
66      };
67
68      // lambda function - draw O
69      auto DrawO = [&] (float size, float width, float position_x, float position_y)
70      {
71          circle.setRadius(size/2);
72          circle.setOrigin(circle.getGeometricCenter());
73          circle.setFillColor(sf::Color::Transparent);
74          circle.setOutlineColor(sf::Color(0, 170, 255));
75          circle.setOutlineThickness(-width);
76          circle.setPosition({position_x, position_y});
77          window.draw(circle);
78      };
79
80      // lambda function - draw winning line
81      auto DrawWinningLine = [&] (float size, float width, float position_x, float position_y, float
angle, float scale = 1.f)
82      {
83          line.setSize({size*scale, width});
84          line.setOrigin(line.getGeometricCenter());
85          line.setFillColor(sf::Color::White);
86          line.setPosition({position_x, position_y});
87          line.setRotation(sf::degrees(angle));
88          window.draw(line);
89      };
90
91      // lambda function - draw cursor
```

```
92    auto DrawCursor = [&] (float size, float width, float position_x, float position_y)
93    {
94        cursor.setSize({size, size});
95        cursor.setOrigin(cursor.getGeometricCenter());
96        cursor.setFillColor(sf::Color::Transparent);
97        cursor.setOutlineColor(sf::Color::White);
98        cursor.setOutlineThickness(-width);
99        cursor.setPosition({position_x, position_y});
100         window.draw(cursor);
101   };
102
103
104   // while window is still open
105   while (window.isOpen())
106   {
107       // handle events
108       while (std::optional event = window.pollEvent())
109       {
110           // when close button is clicked
111           if (event->is<sf::Event::Closed>())
112           {
113               // close window
114               window.close();
115           }
116
117           // when window is resized
118           else if (event->is<sf::Event::Resized>())
119           {
120               // update view
121               sf::View view(sf::FloatRect({0.f, 0.f}, sf::Vector2f(window.getSize())));
122               window.setView(view);
```

```
123         }
124
125         // when keyboard is pressed
126         else if (auto* key = event->getIf<sf::Event::KeyPressed>())
127         {
128             // no winner yet
129             if (winner == ' ')
130             {
131                 // move cursor with WASD
132                     if      ((key->scancode == sf::Keyboard::Scancode::A) and (cursor_x > 0))
cursor_x--;
133                     else if ((key->scancode == sf::Keyboard::Scancode::D) and (cursor_x < 2))
cursor_x++;
134                     else if ((key->scancode == sf::Keyboard::Scancode::W) and (cursor_y > 0))
cursor_y--;
135                     else if ((key->scancode == sf::Keyboard::Scancode::S) and (cursor_y < 2))
cursor_y++;
136
137                 // mark X or O (if empty) with space
138                                     if ((key->scancode == sf::Keyboard::Scancode::Space) and
(state[cursor_y][cursor_x] == ' '))
139                 {
140                     // mark
141                     if (is_X_turn)  state[cursor_y][cursor_x] = 'X';
142                     else            state[cursor_y][cursor_x] = 'O';
143                     // switch turn
144                     is_X_turn = not is_X_turn;
145                 }
146             }
147
148             // restart with Esc
149             if (key->scancode == sf::Keyboard::Scancode::Escape)
150             {
```

```
151                 // reset state
152                 for (int i = 0;  i < 3;  i++)
153                     for (int j = 0;  j < 3;  j++)
154                         state[i][j] = ' ';
155                 // reset position of cursor
156                 cursor_x = 1;
157                 cursor_y = 1;
158                 // reset turn
159                 is_X_turn = true;
160                 // reset winner
161                 winner = ' ';
162             }
163         }
164     }
165
166     // size of window
167     float window_w   = window.getView().getSize().x;
168     float window_h   = window.getView().getSize().y;
169     float window_min = (window_w < window_h) ? window_w : window_h;
170
171     // parameters
172     float space = 0.3f * window_min;  // size of space
173     float size  = 0.8f * space;       // size of mark
174     float width = 0.1f * size;        // line width
175
176     // fill window with color
177     window.clear(sf::Color(64, 64, 64));
178
179     // draw grid
180     DrawGrid(space, width/4, window_w/2, window_h/2);
181
```

```
182         // loop through rows
183         for (int j = 0;  j < 3;  j++)
184         {
185            // loop through columns
186            for (int i = 0;  i < 3;  i++)
187            {
188               // draw O
189               if (state[j][i] == 'O')
190                  DrawO(size, width, window_w/2 + space*(i - 1), window_h/2 + space*(j - 1));
191               // draw X
192               else if (state[j][i] == 'X')
193                  DrawX(size, width, window_w/2 + space*(i - 1), window_h/2 + space*(j - 1));
194            }
195         }
196
197         // draw winning line
198         {
199            for (int i = 0;  i < 3;  i++)
200            {
201               // horizontal
202               if ((state[i][0] != ' ') and (state[i][0] == state[i][1]) and (state[i][1] == state[i][2]))
203               {
204                  DrawWinningLine(space*3, width/2, window_w/2, window_h/2 + space*(i - 1),
0.f);
205                  winner = state[i][0];
206               }
207               // vertical
208               if ((state[0][i] != ' ') and (state[0][i] == state[1][i]) and (state[1][i] == state[2][i]))
209               {
210                  DrawWinningLine(space*3, width/2, window_w/2 + space*(i - 1), window_h/2,
90.f);
211                  winner = state[0][i];
```

```
212                 }
213             }
214             // diagonal 1
215             if ((state[0][0] != ' ') and (state[0][0] == state[1][1]) and (state[1][1] == state[2][2]))
216             {
217                 DrawWinningLine(space*3, width/2, window_w/2, window_h/2,  45.f, 1.4f);
218                 winner = state[0][0];
219             }
220             // diagonal 2
221             if ((state[0][2] != ' ') and (state[0][2] == state[1][1]) and (state[1][1] == state[2][0]))
222             {
223                 DrawWinningLine(space*3, width/2, window_w/2, window_h/2, -45.f, 1.4f);
224                 winner = state[0][2];
225             }
226         }
227
228         // draw cursor
229         if (winner == ' ')
230                 DrawCursor(space, width/2, window_w/2 + space*(cursor_x - 1), window_h/2 +
space*(cursor_y - 1));
231
232         // display
233         window.display();
234     }
235
236
237     // program end successfully
238     return 0;
239 }
```