Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Институт информатики и вычислительной техники

09.03.01 "Информатика и вычислительная техника"

профиль "Программное обеспечение средств вычислительной техники и автоматизированных систем"

**Курсовая работа**

по дисциплине «Программирование» на тему: игра «Тетрис»

Выполнил:

Cтудент гр. ИП-114 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Яворский Д. И./

ФИО студента

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Проверил:

Ассистент кафедры ПМиК \_\_\_\_\_\_\_\_\_\_\_\_\_/Агалаков А. А./

ФИО преподавателя

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Новосибирск 2022

# Оглавление

[Задание](#_aspcv1oo7hvp) 3

[Теория](#_rqviu9off406) 3

[Ход работы](#_28q6mm3n8wvg) 3

[Демонстрация работы](#_z0c1o2bz3xkn) 4

[Вывод](#_dcy441ati36w) 6

[Литература](#_mj8wffr7qayv) 6

[Приложение](#_sp9m96ky3e65) 7

# 

# Задание

Реализовать игру «Тетрис».

# Теория

Тетрисом называется игра-головоломка с падающими блоками. В этой игре, мы имеем 7 разных фигур, называемых так: S-фигура, Z-фигура, T-фигура, L-фигура, фигура-линия, фигура "Г", и квадрат. Каждая из этих фигур формируется с помощью четырёх квадратиков. Фигуры падают вниз на «доску». Цель игры Тетрис – перемещать и вращать фигуры так, чтобы их приземлилось как можно больше. Если мы сумеем сформировать ряд, ряд исчезает, и мы получаем очки. Мы играем в Тетрис до тех пор, пока не заполнится всё поле фигурами.

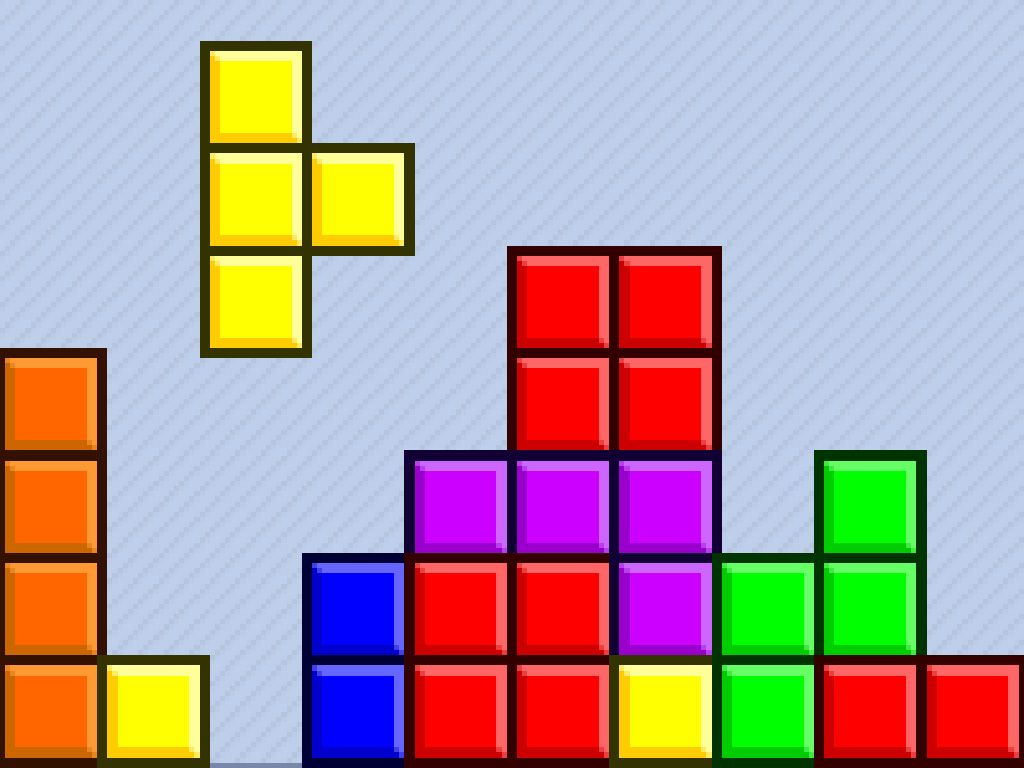


Рисунок 1. – Вид обычного тетриса

# Ход работы

Первым делом было создано поле для игры шириной 15 клеток и высотой 25, прописана проверка на пустое поле в начале, добавлен счёт с начальным значением 0 и схема начисления очков. Далее были созданы 7 фигур – «Тетрамино», случайная смена цвета, проверка на координатное движение фигур, а также условие для окончания игры. Следом было написано для них движение (влево, вправо, ускорение вниз), а также повороты самой фигуры. Работа с окном: размер игровой области, её заголовок, закрытие окна и т.д. Создано управление с помощью стрелок. Потом решено было добавить, как и в оригинальной игре, уровень сложности, то есть с набором очков у вас увеличивается скорость падения фигуры, по умолчанию стоит первый уровень сложности, второй уровень начинается с 30 очков, третий с 60, четвёртый с 90, а максимальный из них – пятый, который достигается при наличии 130 очков. За каждую собранную линию даётся 10 очков. После этого была доделана визуальная часть игры: цвет фона и букв, их стиль и шрифт, используемых в счёте и уровне игры, а также цвет клеток была изменен с однотонного на цвет с текстурой и бликами. В конце было добавлено меню из 2 пунктов: 1 - запуск игры, 2 - выход из меню. В процессе работы меню было решено вынести в отдельную программу.

# Демонстрация работы

На рисунках 1 и 2 изображены меню программы и сама игра в нескольких стадиях.

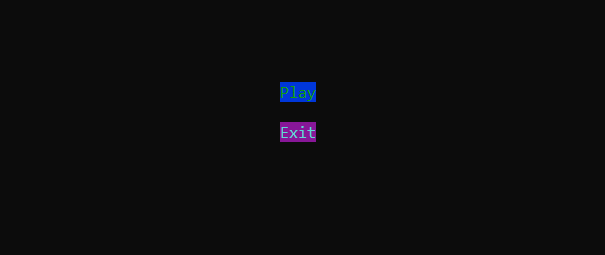
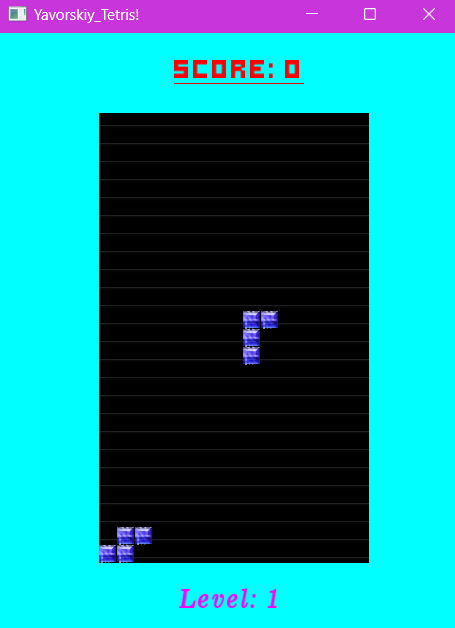
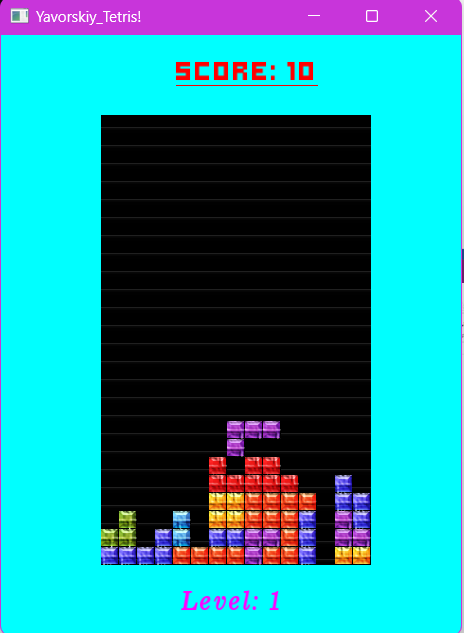
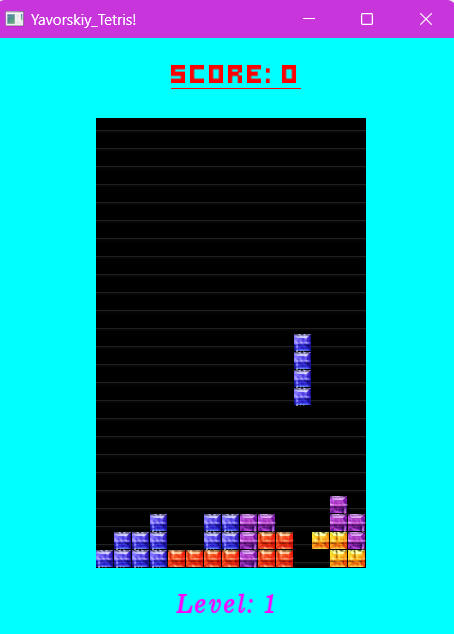
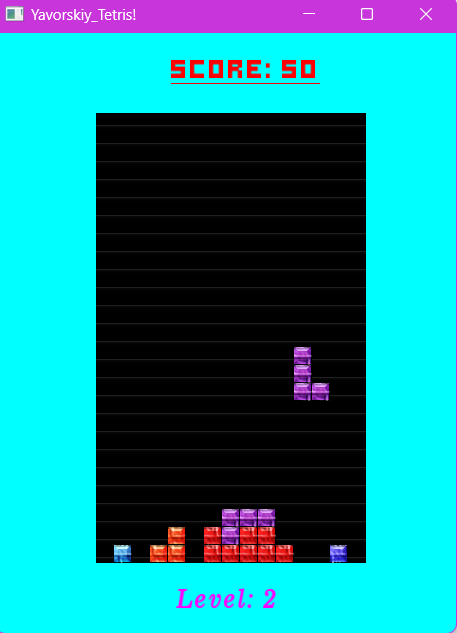
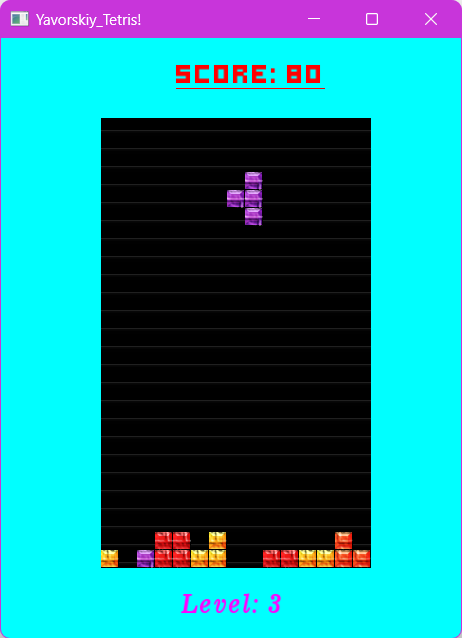


Рисунок 2. Демонстрация меню

Рисунки 3 - 6. Несколько фигур «Тетриса»



Рисунки 7 - 11. Демонстрация движения объекта

# Вывод

В результате выполнения данной курсовой работы я смог написать свою игру на С++ - «Тетрис», с оригинальным геймплеем, а также сделав счётчик очков и уровней.

# Литература

1. Васильев А. Н., Программирование на С++ в примерах и задачах: учебное пособие, 2016. 369 с.

2. Земсков Ю.В., Программирование на языке C/C++: учебное пособие, 2012. 151 с.

# Приложение

## Листинг 1 – menu.cpp

|  |
| --- |
| #include <stdio.h>  #include <conio.h>  #include <graphics.h>  #include "altconio.h"  int main(){  int ch,i;  for (i = 1; i <= 3; i++) {  if (i == 1) settextcolor (2, 17);  else settextcolor (11, 5);  gotoxy(55, 5 + i \* 2);  if(i == 1) {  printf("Play");  }  if(i == 2) {  printf("Exit");  }  }  i=1;  do {  ch=\_getch();  gotoxy(55, 5 + i \* 2);  settextcolor (11, 5);  if(i == 1) {  printf("Play");  }  if(i == 2) {  printf("Exit");  }  if (ch == 72) i--;  if (ch == 80) i++;  if (i > 2) i = 1;  if (i < 1) i = 2;  gotoxy (55, 5 + i \* 2);  settextcolor (2, 17);  if(i == 1) {  printf("Play");  }  if(i == 2) {  printf("Exit");  }  } while(ch != 13);  if (i == 1){ system("C:\\Users\\79133\\source\\repos\\ConsoleApplication3\\Debug\\ConsoleApplication3.exe");  closegraph();  return 0;  }  if (i == 2)  return 0;  \_getch();  return 0;  } |

## Листинг 2 – ConsoleApplication3.cpp

|  |
| --- |
| #include <SFML/Graphics.hpp>  #include <iostream>  #include <time.h>  #include <string>  const int field\_height = 25;  // высота игрового поля  const int field\_width = 15;  // ширина игрового поля  int field[field\_height][field\_width]{ }; // игровое поле  // Массив фигурок-тетрамино  int figures[7][4] =  {  1,3,5,7, // I  2,4,5,7, // S  3,5,4,6, // Z  3,5,4,7, // T  2,3,5,7, // L  3,5,7,6, // J  2,3,4,5, // O  };  int checkLine(int& userPoints)  {  //----ПРОВЕРКА ЛИНИИ----//  int k = field\_height - 1; // строка k - самая нижняя  for (int i = field\_height - 1; i > 0; i--)  // i пробегает все строки снизу вверх.  // в начале k и i совпадают  {  int count = 0;  for (int j = 0; j < field\_width; j++)  {  if (field[i][j])  {  count++;  }  field[k][j] = field[i][j];  }  if (count < field\_width)  {  k--;  }  else  {  userPoints += 10; //сколько очков даётся за каждую линию  }  // k уменьшится только если линия не занята полностью  // если линия занята полностью, то следующее i  // уже соответствует более высокой строке  // а k всё ещё на полной строке  // строка k будет перезаписана следующей строкой i  }  return userPoints;  }  int Level  {  Level = 1  };  class Tetramino  {  public:  int color{ };  int type{ };  int blockPositions[4]{ };  struct Coordinates  {  int x;  int y;  };  Coordinates coordinates[4]{ };  int figures[7][4] =  {  1,3,5,7, // I  2,4,5,7, // S  3,5,4,6, // Z  3,5,4,7, // T  2,3,5,7, // L  3,5,7,6, // J  2,3,4,5, // O  };  void create()  {  std::cout << "\n Show what you are capable!!" << std::endl;  color = 1 + rand() % 7;  type = rand() % 7;  for (int i = 0; i < 4; i++)  {  blockPositions[i] = figures[type][i];  }  for (int i = 0; i < 4; i++)  {  coordinates[i].x = blockPositions[i] % 2 + 7;  // сдвигаем стартовую позицию ближе к центру с помощью + 7  coordinates[i].y = blockPositions[i] / 2;  }  };  void draw(sf::RenderWindow& thatWindow,  sf::Sprite& thatSprite,  int marginLeft,  int marginTop)  {  // Рисуем тетрамино из класса Tetramino  for (int i = 0; i < 4; i++)  {  // Разукрашиваем тетрамино  thatSprite.setTextureRect(sf::IntRect(color \* 18, 0, 18, 18));  // Устанавливаем позицию каждого кусочка тетрамино  thatSprite.setPosition(coordinates[i].x \* 18, coordinates[i].y \* 18);  thatSprite.move(marginLeft, marginTop); // смещение  // Отрисовка спрайта  thatWindow.draw(thatSprite);  }  };  bool check()  {  for (int i = 0; i < 4; i++)  {  if (coordinates[i].x < 0 || coordinates[i].x >= field\_width || coordinates[i].y >= field\_height)  {  std::cout << std::endl << "field X" << coordinates[i].x << "Y " << coordinates[i].y << "is out of board" << std::endl;  return false;  }  else if (field[coordinates[i].y][coordinates[i].x])  {  std::cout << std::endl << "field X" << coordinates[i].x << "Y " << coordinates[i].y << "is occupied" << std::endl;  return false;  }  }  return true;  }  void moveDown(sf::RenderWindow& thatWindow)  {  int x[4]{};  int y[4]{};  for (int i = 0; i < 4; i++)  {  y[i] = coordinates[i].y;  x[i] = coordinates[i].x;  }  for (int i = 0; i < 4; i++)  {  coordinates[i].y += 1;  }  if (check() == false)  {  std::cout << "stop here" << std::endl;  for (int i = 0; i < 4; i++)  {  coordinates[i].y = y[i];  std::cout << "coloring the field" << std::endl;  field[y[i]][x[i]] = color;  }  create();  if (check() == false)  {  std::cout << "GAME OVER!!" << std::endl;  thatWindow.close();  }  }  }  void moveHoriz(sf::RenderWindow& thatWindow, int direction)  {  int x[4]{};  int y[4]{};  for (int i = 0; i < 4; i++)  {  y[i] = coordinates[i].y;  x[i] = coordinates[i].x;  }  for (int i = 0; i < 4; i++)  {  coordinates[i].x += direction;  }  if (check() == false)  {  for (int i = 0; i < 4; i++)  {  coordinates[i].x = x[i];  }  }  }  void rotate()  {  int x[4]{};  int y[4]{};  for (int i = 0; i < 4; i++)  {  y[i] = coordinates[i].y;  x[i] = coordinates[i].x;  }  int originX = coordinates[1].x; // указываем X центра вращения  int originY = coordinates[1].y; // указываем Y центра вращения  for (int i = 0; i < 4; i++)  {  int dy = coordinates[i].y - originY; // y - y0  int dx = coordinates[i].x - originX; // x - x0  coordinates[i].x = originX - dy; // x\_2 = x\_0 - dy  coordinates[i].y = originY + dx; // y\_2 = y\_0 + dx  }  if (check() == false)  {  for (int i = 0; i < 4; i++)  {  coordinates[i].x = x[i];  }  }  }  };  int main()  {  int window\_width {460};  int window\_height {600};  int userPoints {0};  Tetramino tetramino;  tetramino.create();  sf::RenderWindow window(sf::VideoMode(window\_width, window\_height), "Yavorskiy\_Tetris!");  // Создание и загрузка текстуры  sf::Texture texture;  texture.loadFromFile("images/tiles.png");  // Создание спрайта  sf::Sprite sprite(texture);  // Вырезаем из спрайта отдельный квадратик размером 18х18 пикселей  sprite.setTextureRect(sf::IntRect(0, 0, 18, 18));  // Переменные для таймера и задержки  float timer = 0;  float delay = 0.05;  // Часы (таймер)  sf::Clock clock;  // Главный цикл приложения: выполняется, пока открыто окно  while (window.isOpen())  {  // Получаем время, прошедшее с начала отсчёта, и конвертируем его в секунды  float time = clock.getElapsedTime().asSeconds();  clock.restart();  timer += time;  // Обрабатываем события в цикле  sf::Event event;  while (window.pollEvent(event))  {  // Пользователь нажал на «крестик» и хочет закрыть окно?  if (event.type == sf::Event::Closed)  // тогда закрываем его  window.close();  // Была ли нажата клавиша на клавиатуре?  if (event.type == sf::Event::KeyPressed)  {  // Эта кнопка – стрелка влево  if (event.key.code == sf::Keyboard::Left)  {  tetramino.moveHoriz(window, -1);  }  // Эта кнопка – стрелка вправо  else if (event.key.code == sf::Keyboard::Right)  {  tetramino.moveHoriz(window, 1);  }  // Эта кнопка – стрелка вверх  else if (event.key.code == sf::Keyboard::Up)  {  tetramino.rotate();  }  }  // Нажали на кнопку "Вниз" ? Ускоряем падение тетрамино  if (sf::Keyboard::isKeyPressed(sf::Keyboard::Down)) delay = 0.1;  }  // Движение тетрамино вниз («тик» таймера)  if (timer > delay)  {  tetramino.moveDown(window);  timer = 0;  }  checkLine(userPoints);  //пять уровней сложности  Level = 1; delay = 0.8; //(минимальная) 1-ый уровень скорости падения фигуры  if (userPoints > 20)  {  Level = 2;  delay = 0.6; //2-ой уровень скорости падения фигуры  }  if (userPoints > 50)  {  Level = 3;  delay = 0.4; //3-ий уровень скорости падения фигуры  }  if (userPoints > 80)  {  Level = 4;  delay = 0.2; //4-ый уровень скорости падения фигуры  }  if (userPoints > 120)  {  Level = 5;  delay = 0.05; //(максимальная) 5-ый уровень скорости падения фигуры  }  //----ОТРИСОВКА----//  // цвет фона - голубой  window.clear(sf::Color::Cyan);  // Отрисовкна фона  int tetrisFeildTop{ 80 };  int tetrisFeildLeft{ 100 };  for (int varHeight = 0; varHeight < field\_height; varHeight++)  {  for (int j = 0; j < field\_width; j++)  {  if (field[varHeight][j] == 0)  {  sprite.setTextureRect(sf::IntRect(8 \* 18, 0, 18, 18));  sprite.setPosition(tetrisFeildLeft + j \* 18, tetrisFeildTop + varHeight \* 18);  }  else  {  sprite.setTextureRect(sf::IntRect(field[varHeight][j] \* 18, 0, 18, 18));  // значение каждого поля field[varHeight][j] это цвет, его мы используем для выбора цветного квадратика из спрайта  }  sprite.setPosition(tetrisFeildLeft + j \* 18, tetrisFeildTop + varHeight \* 18);  window.draw(sprite);  }  }  // Рисуем тетрамино с помощью метода draw из класса Tetramino  tetramino.draw(window, sprite, tetrisFeildLeft, tetrisFeildTop);  //надпись Score  sf::Font font;  font.loadFromFile("Font/Samson.ttf");    sf::Text text;  // выбор шрифта  text.setFont(font); // шрифт - sf::Font  std::string userPointsString = "Score: " + std::to\_string(userPoints);  // установка строки для отображения  text.setString(userPointsString);  // установка размера символов  text.setCharacterSize(35); // в пикселях  // установка цвета символов  text.setFillColor(sf::Color::Red);  // установка стиля текста и расположения  text.setStyle(sf::Text::Bold | sf::Text::Underlined);  text.setPosition(175.f, 10.f);  window.draw(text);  //надпись Level  sf::Font font1;  font1.loadFromFile("Font/News 705 Italic BT.ttf");  sf::Text text1;  // выбор шрифта  text1.setFont(font1); // шрифт - sf::Font  std::string LevelString = "Level: " + std::to\_string(Level);  // установка строки для отображения  text1.setString(LevelString);  // установка размера символов  text1.setCharacterSize(25); // в пикселях  // установка цвета символов  text1.setFillColor(sf::Color::Magenta);  // установка стиля текста и расположения  text1.setStyle(sf::Text::Bold);  text1.setPosition(180.f, 550.f);  window.draw(text1);  // Отрисовка окна  window.display();  }  return 0;  } |