

Министерство науки и высшего образования РФ  
Пензенский государственный университет  
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
к курсовой работе  
по курсу «Программирование»  
на тему «Разработка программы сложной структуры методом нисходящего  
программирования. Игра "Тетрис"»

10.06.26  
хр  
Выполнил: ст. гр. 25BVB4  
Жмуркина А. С.  
Принял:  
Кандидат т.н. Токарев А. Н.

Пенза 2026

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
Факультет Вычислительной техники  
Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"



Зав. кафедрой ВТ

М.А. Митрохин

« \_\_\_\_ » \_\_\_\_ 20\_\_ г

**ЗАДАНИЕ**

на курсовое проектирование по курсу

**Программирование**

Студенту Митрохину Андрею Сергеевичу Группа 25 ВВВ4

Тема проекта: **Разработка программы сложной структуры методом нисходящего программирования.**

Исходные данные (технические требования) на проектирование  
Игра "Жетрис".

Требования:

Управление фигурами с помощью клавиатуры;

Система подсчета очков;

Увеличение скорости игры;

Постепенное ускорение игрового процесса;

Сохранение текущего результата в текстовый файл.

**Обязательные требования к программе:**

1. Многомодульность.
2. Использование сложных типов данных – структур, списков, файлов.
3. Режим работы видеосистемы – текстовый / графический.
4. Устройство ввода информации – клавиатура, мышь.
5. Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система MS Windows.
8. Язык программирования – Си и Ассемблер.
9. Среда разработки ПО – Microsoft Visual Studio.

## 1. Расчетная часть

### Разработка программы.

## 2. Графическая часть

Схема данных, схема ресурсов системы, схема работы системы, иерархическая структура программы, схема взаимодействия программы.

### 3. Экспериментальная часть

## Отладка программы.

### Срок выполнения проекта по разделам

В соответствии с графиком.

Дата выдачи задания "18" февраля 2026 г.

Дата защиты проекта " \_\_\_\_\_ "

Руководитель Кры

Задание получил "25" февраля 2026г.

Студент Ильинский Андрей Сергеевич *AK*

## Содержание

|                                      |    |
|--------------------------------------|----|
| Введение.....                        | 5  |
| Постановка задачи.....               | 6  |
| Выбор решения.....                   | 7  |
| Описание разработки программы .....  | 8  |
| Структура программы .....            | 9  |
| Алгоритм работы.....                 | 9  |
| Отладка и тестирование.....          | 11 |
| Описание программы.....              | 12 |
| Руководство пользователя .....       | 16 |
| Заключение .....                     | 19 |
| Список используемых источников ..... | 20 |

## Введение

Компьютерные игры остаются одним из наиболее наглядных способов изучения алгоритмов, структур данных и организации интерактивных программ. Даже простая на первый взгляд игра требует продуманной обработки ввода, хранения состояния, вывода графики, проверки столкновений и реакции на изменение игровых условий.

В данной курсовой работе рассматривается разработка консольного приложения Tetris на языке C/C++ для операционной системы Windows. Программа реализует классическую механику падающих фигур: пользователь перемещает и поворачивает тетромино, заполняет горизонтальные линии, получает очки и переходит на новые уровни сложности.

Проект построен как программа сложной структуры методом нисходящего программирования. Главная функция управляет общим сценарием работы, а отдельные модули отвечают за игровую логику, состояние поля, проверку столкновений, отрисовку консоли, цветовое оформление и сохранение результатов.

Особенностью реализации является использование консольных средств Windows API, функций `conio.h` для обработки клавиатуры без ожидания Enter, а также ассемблерных вставок в отдельных низкоуровневых операциях. Такой подход позволяет показать связь между алгоритмической частью программы и системными механизмами вывода.

## Постановка задачи

Необходимо разработать консольное приложение Tetris, предназначенное для запуска в среде Windows и управления с клавиатуры. Программа должна предоставлять пользователю меню, игровой режим, настройку цвета интерфейса, просмотр сохранённых результатов и возможность выхода.

Игровое поле должно иметь размеры 10 на 20 клеток. В программе необходимо реализовать семь классических видов фигур, хранить текущую и следующую фигуру, отображать их на экране и обеспечивать корректное появление новой фигуры после фиксации предыдущей.

Пользователь должен иметь возможность перемещать фигуру влево и вправо, ускорять падение вниз, поворачивать фигуру, выполнять мгновенное падение пробелом и завершать игру клавишей Esc. При проигрыше программа должна показать сообщение об окончании игры и позволить перезапустить партию клавишей R.

В ходе игры необходимо проверять столкновения фигуры со стенками, дном поля и уже установленными блоками. Заполненные горизонтальные линии должны удаляться, поле должно сдвигаться вниз, а счёт, количество линий, уровень и скорость падения должны обновляться.

Дополнительно требуется сохранять результаты в файл scores.txt с датой, количеством очков, уровнем и числом очищенных линий. Программа должна быть разделена на несколько исходных файлов и заголовков для повышения читаемости и удобства сопровождения.

## Выбор решения

В качестве языка реализации выбран C/C++ с использованием стандартных библиотек C и системных функций Windows. Такой выбор подходит для учебной задачи, поскольку позволяет явно работать с массивами, файлами, временем, клавиатурным вводом и консольным выводом.

Для организации проекта использованы три основных исходных файла. Файл TETR.cpp содержит функцию main, стартовый экран, главное меню и основной игровой цикл. Файл game.cpp содержит состояние игры и алгоритмы работы с фигурами, линиями, очками и результатами. Файл graphics.cpp отвечает за позиционирование курсора, установку цвета и отрисовку интерфейса.

Игровое поле представлено двумерным массивом field[H][W], где H равно 20, а W равно 10. Каждая фигура хранится в матрице 4 на 4. Набор фигур задан константным массивом F, что упрощает выбор случайного тетромينو и копирование формы в текущую или следующую фигуру.

Для консольной графики используется функция SetConsoleCursorPosition, позволяющая перерисовывать экран без постоянного прокручивания строк. Цвет текста задаётся через SetConsoleTextAttribute. Ввод обрабатывается функциями \_kbhit и \_getch, поэтому игра реагирует на клавиши сразу после нажатия.

Скорость игры регулируется значением speed. При увеличении количества очищенных линий растёт уровень, а задержка между автоматическими смещениями фигуры уменьшается. Это создаёт постепенное усложнение игрового процесса.

## Описание разработки программы

Разработка выполнялась сверху вниз. Сначала был определён общий сценарий: запуск программы, отображение заставки, переход в главное меню, выбор режима, игровой цикл, завершение партии и сохранение результата. После этого общий сценарий был разделён на независимые функции.

Главное меню реализовано в `TETR.cpp`. Пользователь выбирает начало игры, настройку цвета, просмотр результатов или выход. Для настройки цвета применяется переменная `game_color`, значения которой соответствуют атрибутам консольного текста `Windows`.

Перед началом партии вызывается функция `reset`. Она очищает поле, сбрасывает счёт, уровень, количество линий и признак окончания игры, а также выбирает следующую случайную фигуру. Затем функция `drop` подготавливает текущую фигуру и размещает её в верхней части поля.

Основной игровой цикл постоянно проверяет состояние клавиатуры и таймер. Если нажата стрелка влево или вправо, выполняется проверка столкновения и изменение координаты `fx`. Стрелка вверх поворачивает фигуру, стрелка вниз ускоряет падение, пробел выполняет мгновенное опускание до препятствия.

Поворот фигуры выполняется в функции `rotate`. Сначала создаётся временная копия текущей матрицы, затем элементы переносятся в новую матрицу с поворотом на 90 градусов. Если после поворота возникает столкновение, исходная форма восстанавливается.

Функция `col` проверяет, можно ли переместить текущую фигуру на заданное смещение. Для каждой заполненной клетки фигуры вычисляются координаты на игровом поле. Если клетка выходит за границы, достигает дна или пересекается с установленным блоком, функция сообщает о столкновении.

Когда фигура больше не может двигаться вниз, вызывается `drop`. Фигура фиксируется в массиве `field`, после чего программа ищет полностью

заполненные строки. Заполненная строка удаляется, все строки выше сдвигаются вниз, счёт увеличивается, а игра выбирает новую фигуру.

Отрисовка выполняется функцией `draw`. Она выводит рамку поля, установленные блоки, текущую фигуру, счёт, уровень, число линий и предпросмотр следующей фигуры. При завершении игры поверх поля выводится сообщение «ИГРА ОКОНЧЕНА» и подсказка для перезапуска.

## **Структура программы**

- `TETR.cpp` - главный модуль, меню, обработка выбора пользователя и игровой цикл;
- `game.h` - объявления размеров поля, глобальных переменных и функций игровой логики;
- `game.cpp` - массив фигур, состояние поля, проверка столкновений, поворот, фиксация фигуры, подсчёт очков и сохранение результатов;
- `graphics.h` - объявления функций и переменных, связанных с консольной графикой;
- `graphics.cpp` - позиционирование курсора, установка цвета, вывод поля, интерфейса и стартовой заставки;
- `scores.txt` - файл, создаваемый программой для хранения истории результатов.

## **Алгоритм работы**

После запуска программа устанавливает русскую локаль, получает дескриптор консоли, скрывает курсор и инициализирует генератор случайных чисел. Затем выводится стартовая заставка, ожидающая нажатия `Enter`.

В главном меню пользователь выбирает действие. При запуске игры выполняется сброс состояния и начинается цикл партии. Внутри цикла программа обрабатывает клавиши, автоматически опускает фигуру по таймеру, перерисовывает поле и проверяет признак `gameover`.

Если фигура достигла препятствия, она переносится в массив игрового поля. Далее проверяются все строки от нижней к верхней. Полностью заполненная строка удаляется, а игрок получает очки. За каждую очищенную линию начисляется 100 очков, а за каждую десятую линию предусмотрен бонус 1000 очков.

Уровень вычисляется как  $\text{lines\_total} / 10 + 1$ . Скорость падения уменьшается на 30 единиц на каждом уровне, но не становится меньше 50. Благодаря этому игра ускоряется постепенно и остаётся управляемой.

Проигрыш определяется в момент появления новой фигуры. Если новая фигура сразу пересекается с уже занятыми клетками, устанавливается `gameover`, результат записывается в файл, а игрок может начать новую партию клавишей R.

## Отладка и тестирование

Отладка программы выполнялась по отдельным функциональным блокам. Сначала проверялась корректность вывода поля и позиционирования курсора, затем обработка клавиш и перемещение фигуры по границам поля.

Особое внимание уделялось функции `col`, так как от неё зависит запрет выхода фигуры за пределы поля, корректное падение и невозможность пересечения с уже установленными блоками. Проверялись ситуации у левой и правой стенки, у нижней границы, а также при повороте рядом с препятствиями.

Функция `rotate` проверялась на всех семи фигурах. Важным условием является восстановление исходной матрицы, если поворот приводит к столкновению. Это предотвращает повреждение состояния фигуры.

Функция `drop` тестировалась при фиксации фигуры, удалении одной и нескольких строк, начислении очков, повышении уровня и появлении следующей фигуры. Также проверялось сохранение результата в `scores.txt` после завершения партии.

Пользовательские сценарии включали запуск новой игры, смену цвета интерфейса, просмотр пустого и заполненного списка результатов, перезапуск после проигрыша и выход из игрового цикла клавишей `Esc`.

## Описание программы

Основными данными программы являются массив `field`, матрица текущей фигуры `fig`, матрица следующей фигуры `next_fig`, координаты `fx` и `fy`, а также переменные `score`, `level`, `lines_total`, `speed` и `gameover`.

Массив `F` содержит семь фигур. Единица в матрице означает заполненную клетку, ноль - пустую. При появлении новой фигуры программа копирует одну из матриц в `fig`, а затем заранее выбирает следующую фигуру для предпросмотра.

Функция `gotoxy` получает координаты и передаёт их в `SetConsoleCursorPosition`. Благодаря этому `draw` каждый раз начинает вывод с позиции 0,0 и обновляет уже существующее изображение поля.

Функция `save` открывает файл `scores.txt` в режиме добавления, получает текущее время, форматирует дату и записывает строку результата. Если файл недоступен, программа продолжает работу без аварийного завершения.

Главный цикл использует `clock` для контроля времени падения фигуры. При превышении задержки `speed` фигура смещается вниз, а если смещение невозможно, фиксируется на поле. Небольшая пауза `Sleep(30)` снижает нагрузку на процессор и делает игру визуально стабильнее.

Ассемблерные вставки используются в функциях `col`, `gotoxy` и `drop`. Они демонстрируют низкоуровневую работу с регистрами при вычислении координат, подготовке структуры `COORD` и увеличении счёта.

На рисунке 1 представлена схема взаимодействия программы.

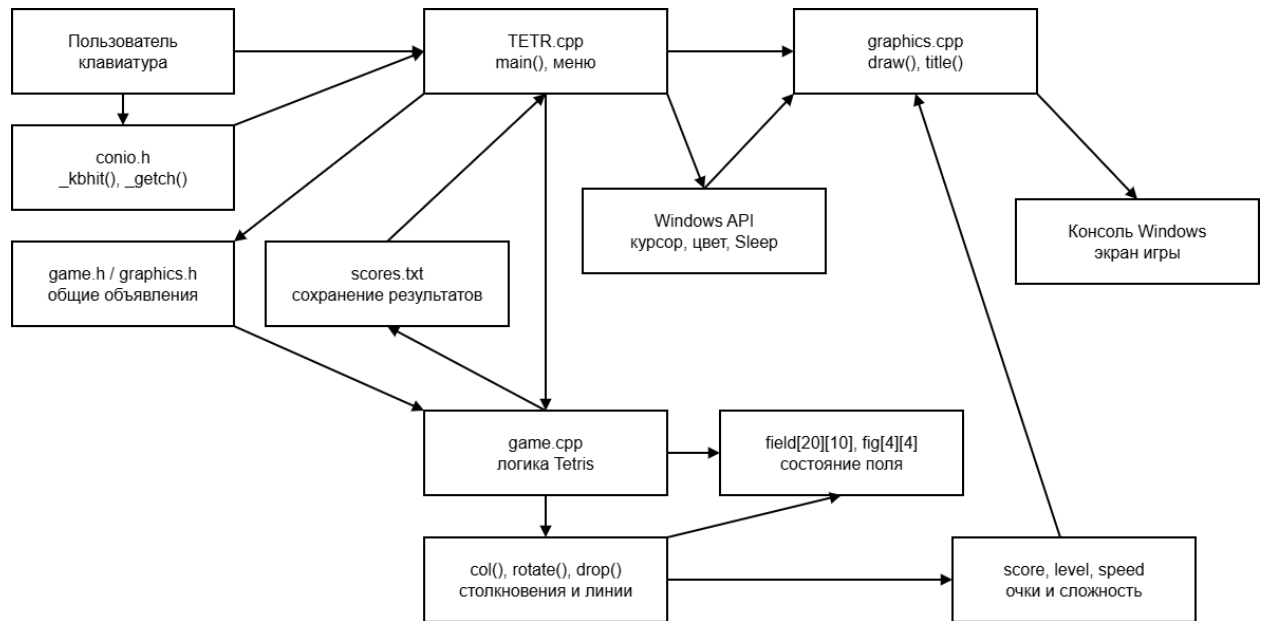


Рисунок 1 - Схема взаимодействия программы

Блок-схема main представлена на рисунке 2.

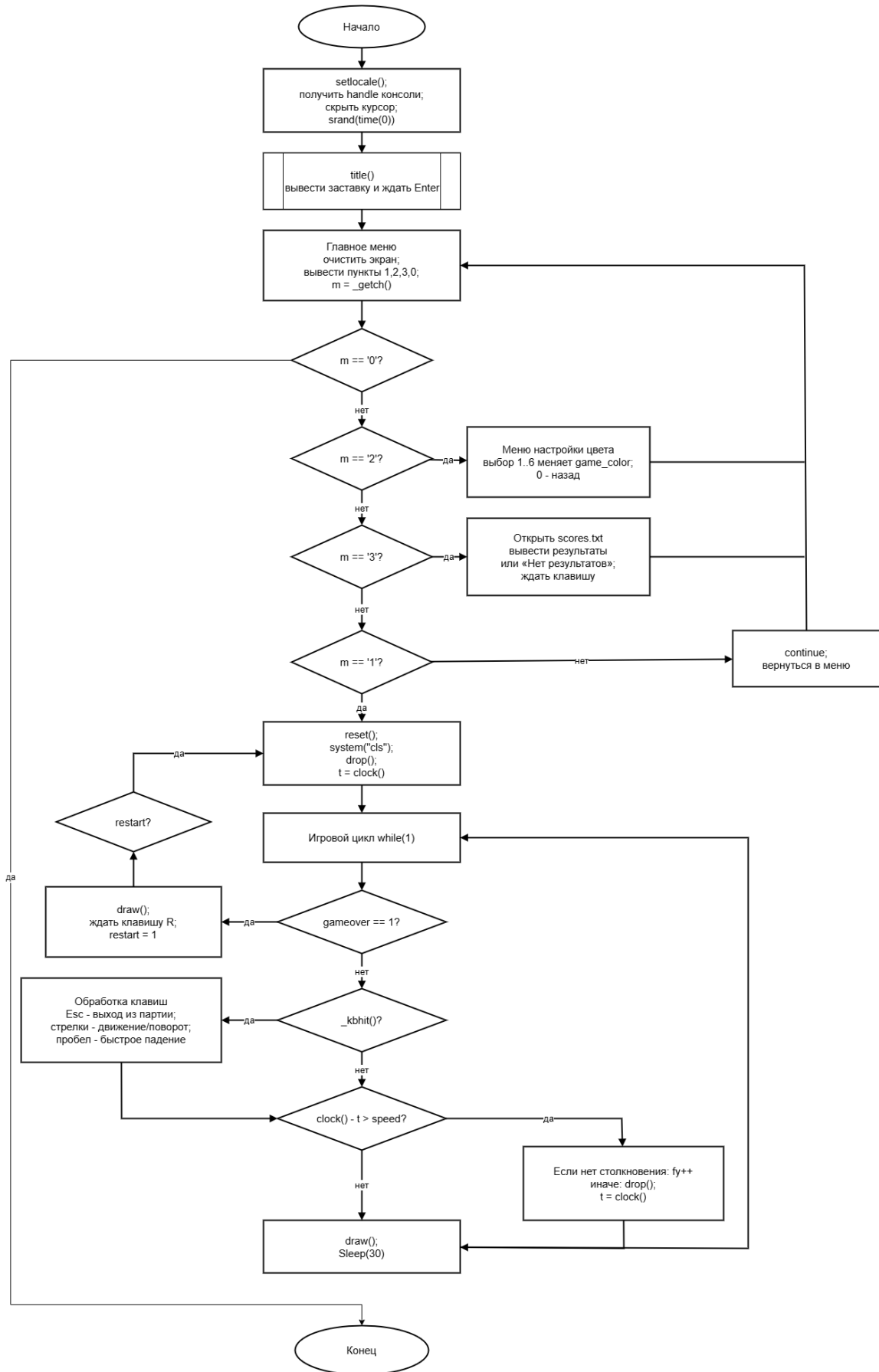


Рисунок 2 - Блок-схема метода main()

На рисунке 3 представлена блок-схема ассемблерной вставки.

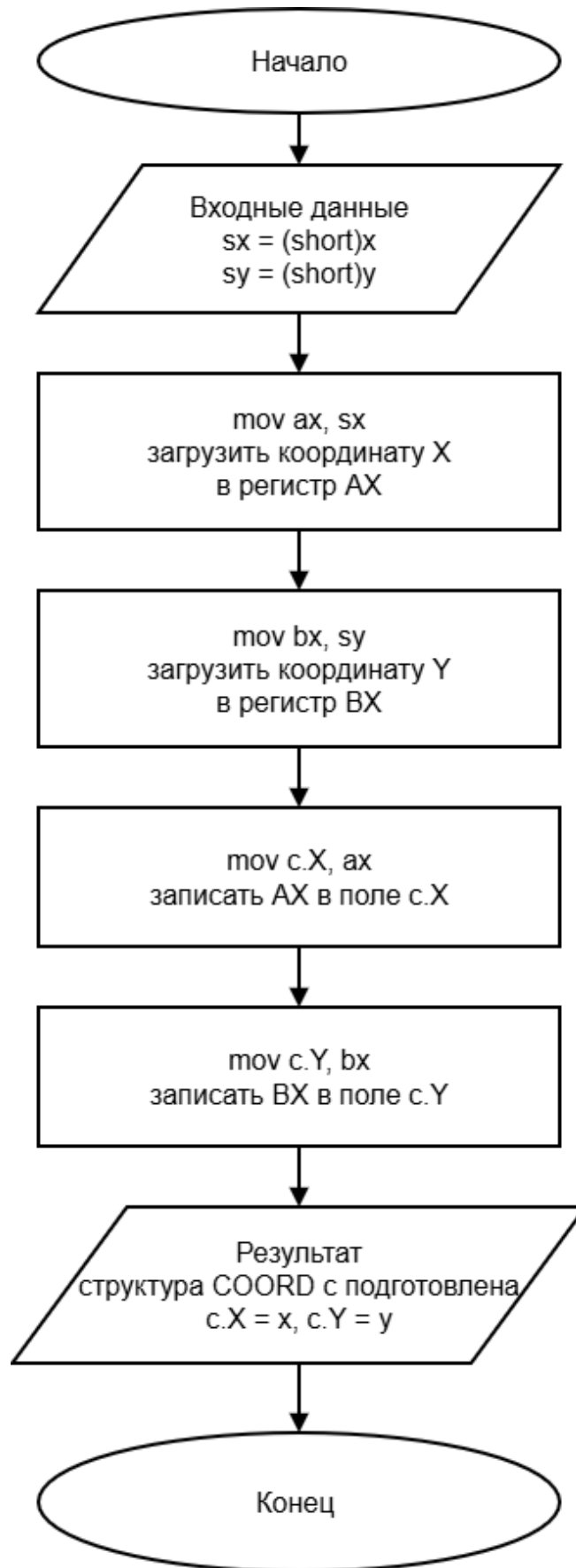


Рисунок 3 - Ассемблерная вставка

## Руководство пользователя

Для запуска программы необходимо скомпилировать проект в среде, поддерживающей Windows API, conio.h и встроенные ассемблерные вставки. Подходящим вариантом является Visual Studio с конфигурацией Win32/x86.

После запуска появляется заставка Tetris (Рисунок 4). Для перехода в главное меню (Рисунок 5) нужно нажать Enter. В меню доступны пункты: 1 - начать игру, 2 - настройки цвета, 3 - результаты, 0 - выход.



Рисунок 4 - Заставка Tetris

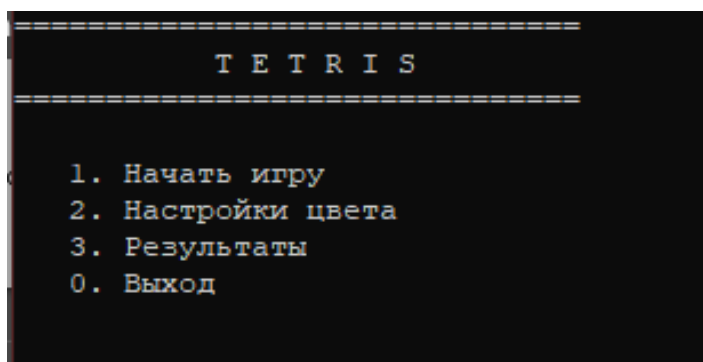


Рисунок 5—Главное меню

В режиме настройки цвета (Рисунок 6) пользователь выбирает один из доступных цветов: белый, зелёный, голубой, красный, фиолетовый или жёлтый. Выбранный цвет применяется к игровому интерфейсу.

```

=== НАСТРОЙКИ ЦВЕТА ===

1. Белый
2. Зеленый
3. Голубой
4. Красный
5. Фиолетовый
6. Желтый

0. Назад

```

Рисунок 6 - Настройки цвета

Во время игры (Рисунок 7) используются клавиши:

- стрелка влево - переместить фигуру влево;
- стрелка вправо - переместить вправо;
- стрелка вверх - повернуть фигуру;
- стрелка вниз - опустить на одну клетку;
- пробел - мгновенно опустить фигуру;
- Esc - выйти из партии.

```

#####
#                                     #
#                                     # СЧЁТ: 0
#                                     # УРОВЕНЬ: 1
#                                     # ЛИНИИ: 0
#                                     #
#                                     # СЛЕДУЮЩАЯ:
#                                     #
#      [] []                          #      []
#      [] []                          #      [] [] []
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#####

```

Рисунок 7 - Игровое поле

При окончании игры (Рисунок 8) на поле появляется сообщение. Для перезапуска партии нужно нажать R. Результат автоматически сохраняется в scores.txt и затем доступен через пункт «Результаты» в главном меню.

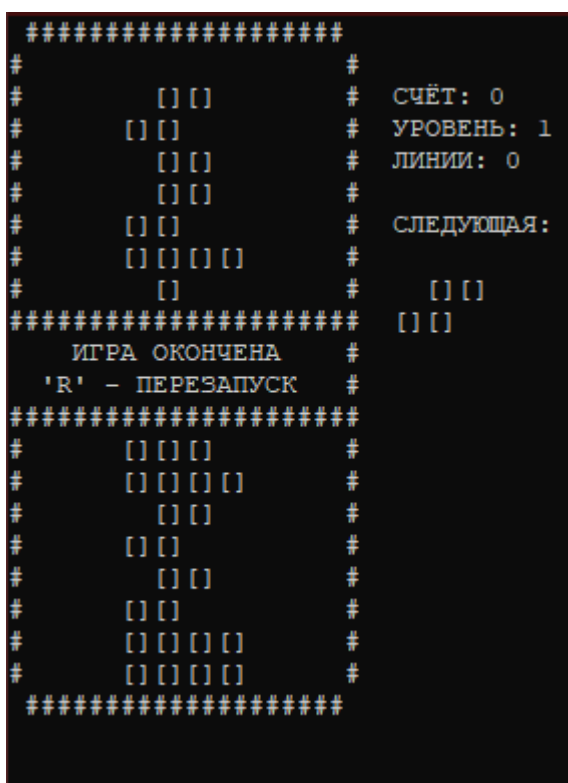


Рисунок 8 - Конец игры

## **Заключение**

В ходе курсовой работы было разработано консольное приложение Tetris на языке C/C++. Программа реализует основные правила классической игры: падение фигур, перемещение, поворот, проверку столкновений, удаление заполненных линий, подсчёт очков и постепенное повышение сложности.

Проект выполнен методом нисходящего программирования. Общая задача была разделена на модули главного управления, игровой логики и консольной графики. Такое разделение сделало программу более понятной и упростило отладку отдельных функций.

Реализация показала практическое применение двумерных массивов, файлового ввода-вывода, работы с таймером, обработки клавиатуры, функций Windows API и ассемблерных вставок. Полученное приложение можно расширять: добавить паузу, разные режимы сложности, таблицу рекордов с именами игроков и более детальное цветовое оформление фигур.

## Список используемых источников

1. Шилдт, Г. С: полное руководство / Г. Шилдт. — 4-е изд. — Москва: Вильямс, 2018. — 832 с. — ISBN 978-5-8459-1709-6.
2. Павловская, Т. А. С/С++. Программирование на языке высокого уровня: учебник для вузов / Т. А. Павловская. — Санкт-Петербург: Питер, 2007. — 460 с. — ISBN 978-5-94723-568-5.
3. ConsoleFunctions [Электронный ресурс] // MicrosoftLearn. — URL: <https://learn.microsoft.com/en-us/windows/console/console-functions> (дата обращения: 02.04.2026).
4. Files and Streams; Time Management [Электронныйресурс] // Microsoft Learn. — URL: <https://learn.microsoft.com/en-us/cpp/c-runtime-library/files-and-streams?view=msvc-170> ; <https://learn.microsoft.com/en-us/cpp/c-runtime-library/time-management?view=msvc-170> (датаобращения: 02.04.2026).
5. GameplayofTetris [Электронныйресурс] // TetrisWiki. — URL: [https://tetris.wiki/Gameplay\\_of\\_Tetris](https://tetris.wiki/Gameplay_of_Tetris) (датаобращения: 02.04.2026).