

Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Пензенский государственный университет»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по курсу «Программирование»

на тему: «Разработка программы сложной структуры методом нисходящего программирования. Игра «2048»»

*21.05.2026
Отмечено
Г. Генералова.*

Выполнил: ст. гр. 25BBB1

Какулин В.С.

Проверил: к.т.н. доцент

Генералова А.А.

Пенза, 2026

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники
Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

М.А. Митрохин

« 18 » февраля 2026 г

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Какулину Владиславу Сергеевичу Группа 25ВВВ1

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

Исходные данные (технические требования) на проектирование

Игра "2048"

Генерацию пиктограмм осуществляет компьютер
Строк питается получить число 2048, путем соедине
ния пиктограмм.

В файл загружается и сохраняется таблица
рекордов.

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.

Объем работы по курсу

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 " феврале 2026г.

Дата защиты проекта " 21 " мар 2026

Руководитель Григорьев А.П.

Задание получил " 18 " феврале 2026г.

Студент Самуил В.С.

Содержание

Введение.....	5
1. Постановка задачи.....	6
2. Выбор решения.....	7
3. Описание разработки программы.....	8
4. Отладка и тестирование.....	10
5. Описание программы.....	11
5.1 Описание модулей.....	11
5.2 Разработка функции на языке Assembler.....	12
6. Руководство пользователя.....	17
6.1 Заставка.....	17
6.2 Игровое поле.....	19
6.3 Победа и поражение.....	20
6.4 Таблица рекордов.....	22
Заключение	23
Список используемых материалов	25
Приложение А. Листинг программы.....	26
Файл main.py.....	26
Файл logic.py.....	31
Файл drawing.py.....	35
Файл scores.py.....	39
Файл constans.py.....	40
Файл stack_asm.dll.....	41

Введение

Цифровые развлечения прочно вошли в повседневную жизнь современного человека, охватывая не только сферу досуга, но и образование. Среди множества логических игр особое место занимает «2048» — простая на первый взгляд, но требующая стратегического мышления и математической интуиции головоломка. Несмотря на минималистичный интерфейс, за ней скрывается богатая алгоритмическая основа, изучение которой даёт ценный практический опыт в разработке программного обеспечения.

Актуальность данной работы определяется тем, что реализация игры «2048» охватывает широкий спектр фундаментальных задач программирования: работу с двумерными структурами данных, проектирование игровой логики, обработку пользовательского ввода и построение графического интерфейса. Всё это в совокупности формирует системный подход к разработке — от архитектуры модулей до взаимодействия отдельных компонентов программы.

В качестве основного языка разработки выбран Python с библиотекой `pygame`, поскольку данный стек обеспечивает быструю разработку графических приложений и широкие возможности для работы с игровым циклом и пользовательским вводом. Для реализации низкоуровневой оптимизации ключевой операции сдвига плиток в работе применяется ассемблерная вставка через механизм взаимодействия с динамической библиотекой посредством `ctypes`.

1. Постановка задачи.

Целью данной работы является разработка графической версии классической игры «2048» с использованием языка программирования Python, сложные структуры которого были изучены благодаря литературы [1], а также ассемблерной вставки для низкоуровневой оптимизации ключевой операции. В ходе выполнения работы необходимо реализовать следующий функционал:

1. Генерация игрового поля 4×4 с начальным размещением двух случайных плиток.
2. Управление сдвигом плиток при помощи клавиатуры.
3. Автоматическое слияние одинаковых соседних плиток с подсчётом и отображением набранных очков.
4. Проверка условий завершения игры — победа при достижении плитки 2048, поражение при отсутствии доступных ходов.
5. Сохранение результатов игры в файл таблицы рекордов с отображением лучшего результата.

В процессе реализации проекта должны быть использованы сложные типы данных — двумерные списки, файлы, а также принципы модульного программирования. Ассемблерная вставка выполнена в стиле MASM и скомпилирована в динамическую библиотеку через Visual Studio. Методика совместного использования языка ассемблера и языка Си в рамках одного исходного файла была освоена на основе учебного пособия [2].

Режим работы видеосистемы — графический. Пользовательский интерфейс построен на основе графического окна с кнопками управления и панелью счёта.

Устройство ввода информации — клавиатура и мышь.

Обязательно наличие экрана приветствия с полем ввода имени игрока.

2. Выбор решения.

В процессе разработки данной игры была выбрана среда разработки Visual Studio Code. Она предоставляет все необходимые инструменты для комфортной и эффективной разработки игры «2048» на языке Python, делая процесс написания кода, отладки и тестирования максимально удобным.

Visual Studio Code подходит для работы с Python благодаря расширению Python Extension, которое обеспечивает подсветку синтаксиса, автодополнение и встроенный отладчик. Для компиляции ассемблерной вставки использовался отдельный инструмент — Visual Studio 2022 Community, поскольку именно она поддерживает синтаксис `__asm {}` и сборку динамической библиотеки `.dll`. Скомпилированная библиотека подключается к Python-коду посредством модуля `ctypes`, изучив его из литературы [3]. Ассемблерная вставка реализует функцию `stack_row()` — операцию сдвига ненулевых элементов строки влево. Именно этот шаг является основой игровой механики — без корректного сдвига слияние плиток работать не будет.

Проект структурирован с использованием модулей, что позволяет логически разделить код на функциональные блоки, которые легко сопровождать.

Результаты игры сохраняются в текстовый файл `'highscores.txt'` с помощью функции `save_score()`. При завершении партии в файл записываются имя игрока и набранный счёт, таблица хранит до 25 лучших результатов, отсортированных по убыванию.

3. Описание разработки программы.

В основу разработки положен нисходящий подход — сначала определяется общая архитектура программы, после чего каждый компонент последовательно детализируется и реализуется.

Запуск программы сопровождается отображением экрана приветствия, где игрок вводит своё имя. После перехода в игру на экране появляется поле 4×4, панель с текущим счётом и личным рекордом, а также кнопки возврата в меню и перезапуска. Управление осуществляется стрелками клавиатуры — плитки сдвигаются в выбранном направлении, одинаковые объединяются, а на освободившееся место добавляется новая плитка со значением 2 или 4.

Код разбит на 6 независимых модулей — такая структура даёт возможность дорабатывать любую часть программы, не затрагивая остальные:

1. *main.py*;
2. *logic.py*;
3. *drawing.py*;
4. *scores.py*;
5. *constants.py*;
6. *stack_asm.dll*.

Модуль *Main.py* является точкой входа — здесь инициализируется библиотека *pygame*, объявляются все переменные состояния и запускается основной цикл обработки событий.

Модуль *Logic.py* содержит всю игровую механику: функции сдвига, слияния, транспонирования и разворота матрицы, а также проверку условий победы и поражения. Здесь же подключается динамическая библиотека *stack_asm.dll* через модуль *ctypes*.

Модуль *Drawing.py* берёт на себя всю визуальную часть — отрисовку плиток, меню, панели счёта и финальных экранов.

Модуль *Scores.py* отвечает за работу с файлом рекордов: чтение существующих результатов и запись нового по завершении партии.

Модуль *Constants.py* централизованно хранит все настройки программы — цвета, размеры, отступы и путь к файлу рекордов.

Модуль *Stack_asm.dll* — динамическая библиотека, скомпилированная из С-файла с ассемблерной вставкой в Visual Studio. Реализует функцию *stack_row()*, которая на уровне машинного кода сдвигает ненулевые элементы строки матрицы влево.

4. Отладка и тестирование.

В качестве среды разработки был выбран Visual Studio Code. Он оптимален для работы с Python благодаря удобному отладчику и встроенному терминалу. В ходе отладки применялись такие возможности, как точка остановки, выполнение кода по шагам, анализ содержимого локальных и глобальных переменных.

Пошаговое выполнение кода позволяло отслеживать состояние матрицы на каждом ходу — как изменяются значения плиток после операций сдвига и слияния. Точки останова использовались для проверки корректности работы ключевых функций.

Отдельно тестировалась ассемблерная функция `stack_row()` — для каждого варианта входных данных результат сравнивался с ожидаемым вручную:

- строка с нулями в начале;
- строка с нулями в середине;
- строка с нулями в конце;
- полностью нулевая строка;
- строка без нулей.

Тестирование проводилось в рабочем порядке — в процессе разработки каждого модуля и после завершения написания программы в целом.

5. Описание программы.

5.1 Описание модулей.

Программа состоит из 6 основных модулей, которые упрощают процесс разработки и отладки, улучшают читаемость и ускоряют сборку проекта.

Модуль *main.py* является главным файлом программы. После запуска инициализируется *pygame*, загружается иконка и шрифты, после чего запускается основной цикл обработки событий.

start_new_game() — создаёт новую матрицу, сбрасывает счёт и все флаги состояния игры, переключает экран в режим игры.

Модуль *logic.py* отвечает за всю игровую механику. Содержит функции работы с матрицей, проверки условий игры и подключение ассемблерной библиотеки.

create_matrix() — инициализирует игровое поле: создаёт матрицу 4×4, заполненную нулями, и добавляет две начальные плитки.

add_new_tile() — находит все свободные клетки матрицы и помещает на случайную из них плитку со значением 2 или 4.

stack() — сдвигает ненулевые элементы каждой строки влево. Вызывает ассемблерную функцию *stack_row()* из динамической библиотеки *stack_asm.dll* через модуль *ctypes*.

combine() — проходит по каждой строке и сливает соседние одинаковые элементы. Возвращает изменённую матрицу и количество набранных за ход очков.

reverse() — разворачивает каждую строку матрицы в обратном порядке. Используется для приведения ходов «Вправо» и «Вниз» к операции сдвига влево.

transpose() — транспонирует матрицу: строки становятся столбцами. Используется для операций «Вверх» и «Вниз».

move() — выполняет полный ход в заданном направлении, последовательно применяя *stack*, *combine*, *stack* с необходимыми преобразованиями. Возвращает новую матрицу и набранные очки.

check_win() — проверяет наличие плитки со значением 2048 на поле. Возвращает True при победе.

check_game_over() — проверяет наличие свободных клеток и доступных ходов. Возвращает True если ходов больше нет.

Модуль *scores.py* отвечает за работу с файлом таблицы рекордов.

load_scores() — загружает таблицу рекордов из файла *highscores.txt* и возвращает список кортежей (имя, счёт).

save_score() — добавляет результат текущей партии в таблицу, сортирует по убыванию и перезаписывает файл.

Модуль *drawing.py* отвечает за отрисовку всех элементов графического интерфейса.

draw_text_centered() — универсальная функция вывода текста точно по центру переданного прямоугольника.

draw_menu() — отрисовывает экран главного меню: заголовок, поле ввода имени, кнопку «ИГРАТЬ» и плашку ошибки при пустом имени.

draw_board() — отрисовывает игровое поле: для каждой клетки матрицы рисует прямоугольник нужного цвета и выводит значение плитки по центру.

draw_hud() — отрисовывает верхнюю панель: имя игрока, текущий счёт, лучший результат и кнопки управления.

draw_game() — объединяет вызовы *draw_hud()* и *draw_board()*, а также отображает финальный экран победы или поражения.

Модуль *constants.py* централизованно хранит все настройки программы — цвета плиток, размеры окна и ячеек, отступы и путь к файлу рекордов. Не содержит функций.

5.2 Разработка функции на языке Assembler.

Функции, реализованные на языке ассемблера, выполняются быстрее и работают напрямую с регистрами процессора. В качестве разрабатываемой функции была выбрана операция сдвига элементов строки — *stack_row()*. Описание функции выполнено на языке C, а вычисления представляют собой ассемблерную вставку в стиле MASM, при изучении которого использовалась

литература [4]. Для реализации используется ключевое слово `__asm`, а сам код располагается в фигурных скобках.

Ассемблерная вставка находится в функции `stack_row()` файла `stack_asm.c`. Она проходит по массиву `row`, собирает все ненулевые элементы во временный буфер, после чего С-код заполняет остаток нулями и копирует результат обратно:

```
#include <string.h>

__declspec(dllexport) void stack_row(int* row, int size) {
    int temp[16] = {0};
    int pos = 0;
    int i;
    __asm {
        mov esi, row
        lea edi, temp
        xor ebx, ebx
        xor ecx, ecx

loop_start:
        cmp ecx, size
        jge loop_end
        mov eax, [esi + ecx * 4]
        test eax, eax
        jz skip
        mov [edi + ebx * 4], eax
        inc ebx

skip:
        inc ecx
        jmp loop_start
loop_end:
        mov pos, ebx
    }
```

```

    }
    for (i = pos; i < size; i++) {
        temp[i] = 0;
    }
    memcpy(row, temp, size * sizeof(int));
}

```

На рисунке 1 представлена схема данных.

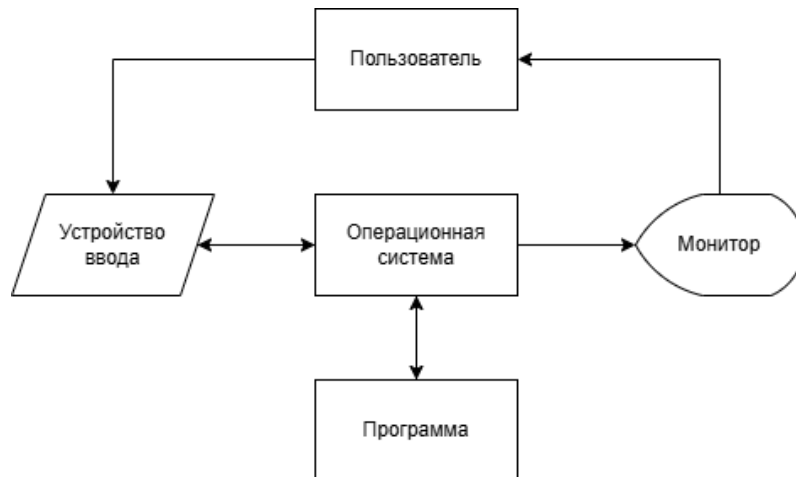


Рисунок 1 – Схема данных

На рисунке 2 представлена схема ресурсов системы – аппаратные и программные ресурсы.

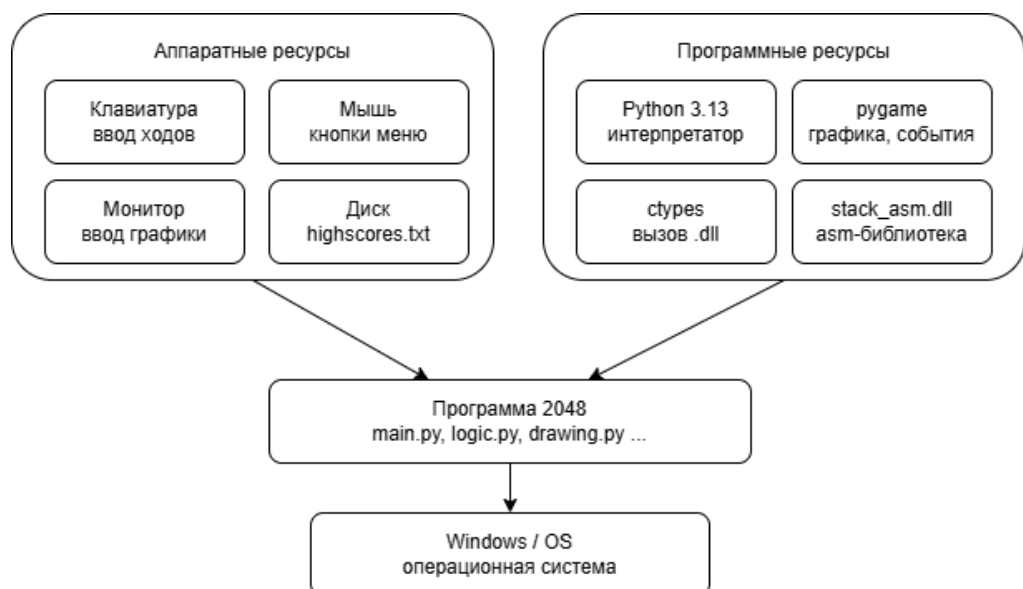


Рисунок 2 – Схема ресурсов системы

На рисунке 3 представлена схема работы системы – игровой цикл от запуска до завершения.

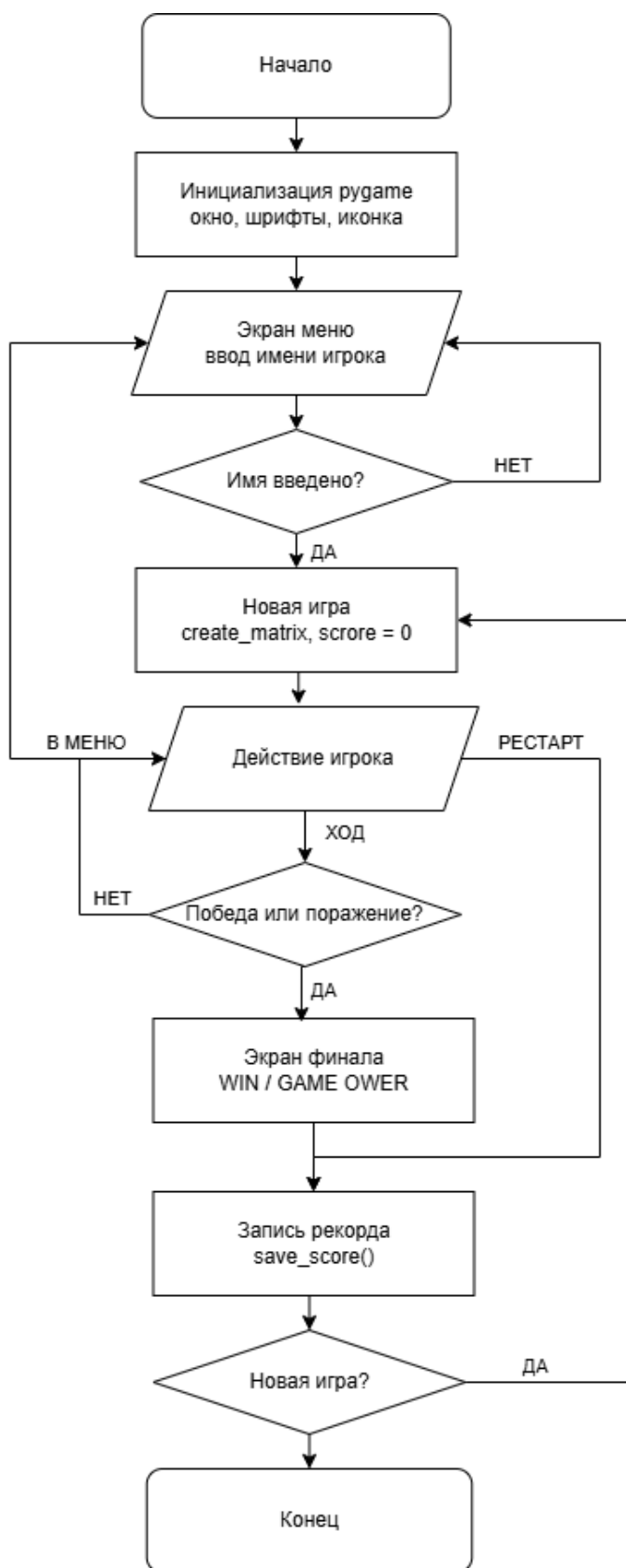


Рисунок 3 – Схема работы системы

На рисунке 4 представлена иерархическая структура программы.



Рисунок 4 – Иерархическая структура программы

На рисунке 5 представлена схема взаимодействия программы — как модули обмениваются данными между собой.

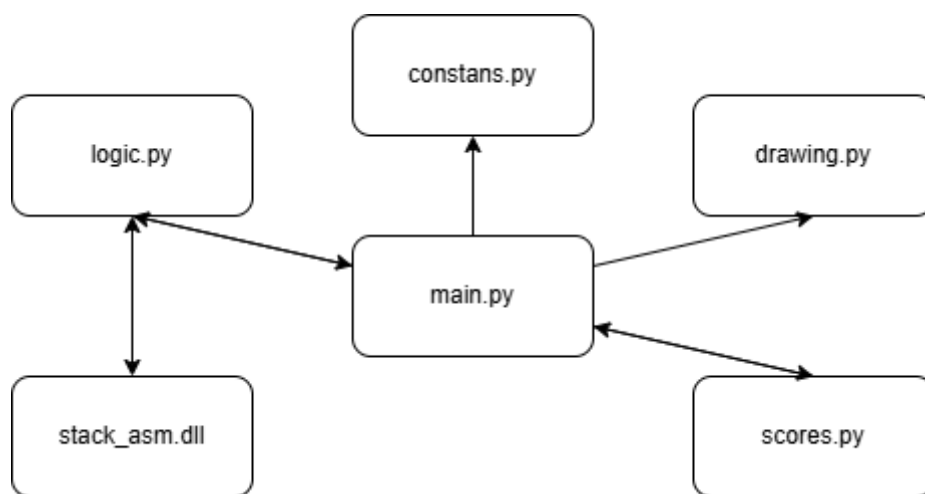


Рисунок 5 – Схема взаимодействия программы

Получается, что каждые модули игры взаимосвязаны между собой и образуют полноценную программу.

6. Руководство пользователя.

Программа main.py предназначена для игры в «2048». Управление интуитивно понятно и не требует специальных навыков.

«2048» — классическая логическая головоломка, где игрок сдвигает плитки на поле 4×4 , добиваясь их слияния. Игровое поле представляет собой сетку из шестнадцати клеток, часть которых занята плитками с числовыми значениями. При сдвиге в любом направлении все плитки смещаются к выбранному краю поля, а одинаковые соседние плитки сливаются в одну с удвоенным значением — при этом игроку начисляются очки. После каждого хода на случайную свободную клетку добавляется новая плитка со значением 2 или 4. Если на поле появляется плитка со значением 2048 — игрок побеждает. Поражение наступает тогда, когда на поле не остаётся свободных клеток и ни один ход невозможен.

6.1 Заставка.

При запуске программы открывается экран приветствия. Игрок вводит своё имя в поле ввода — до 12 символов. Если поле оставить пустым и нажать кнопку «ИГРАТЬ» или клавишу Enter, появится красная надпись «Имя не может быть пустым!». Для начала игры необходимо ввести имя и нажать кнопку «ИГРАТЬ» или клавишу Enter. На рисунке 6 представлен экран приветствия.

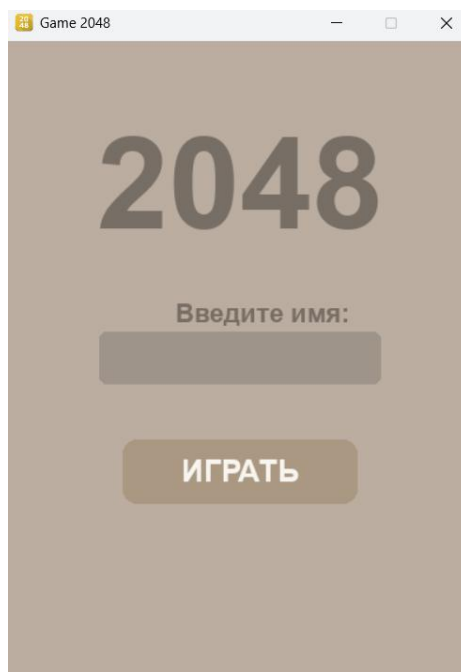


Рисунок 6 – Главное меню с полем ввода имени

На рисунке 7 представлен пример появления плашки при пустом вводе имени.

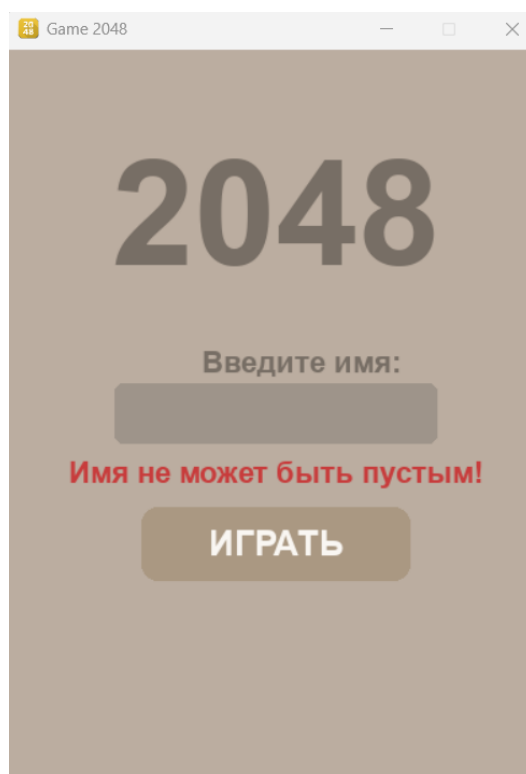


Рисунок 7 – Плашка ошибки при пустом имени

Таким образом, представленные на рисунках 6 и 7 скриншоты демонстрируют наличие полноценной заставки в разработанном игровом приложении.

6.2 Игровое поле.

После ввода имени и нажатия кнопки «ИГРАТЬ» открывается игровое поле 4×4. В верхней части экрана отображаются имя игрока, текущий рекорд, текущий счёт и кнопки «В МЕНЮ» и «РЕСТАРТ». На рисунке 8 представлено игровое поле в начале партии.



Рисунок 8 – Игровое поле в начале игры

Управление осуществляется стрелками клавиатуры. При каждом нажатии все плитки сдвигаются в выбранном направлении, одинаковые соседние сливаются, а на свободное место добавляется новая плитка. На рисунке 9 представлено игровое поле в процессе игры.



Рисунок 9 – Игровое поле в процессе игр

Кнопка «РЕСТАРТ» сбрасывает текущую партию и начинает новую. Кнопка «В МЕНЮ» возвращает на экран приветствия, при этом текущий результат сохраняется в таблицу рекордов.

6.3 Победа и поражение

Если в процессе игры на поле появляется плитка со значением 2048, игра завершается победой. На экран выводится сообщение «ПОБЕДА!» зелёного цвета с итоговым счётом и кнопкой «НОВАЯ ИГРА». Результат автоматически сохраняется в таблицу рекордов. На рисунке 10 представлен экран победы.



Рисунок 10 – Окно победы

Если на поле не остаётся свободных клеток и ни один ход невозможен, игра завершается поражением. На экран выводится сообщение «GAME OVER» красного цвета с итоговым счётом и кнопкой «НОВАЯ ИГРА». На рисунке 11 представлен экран поражения.

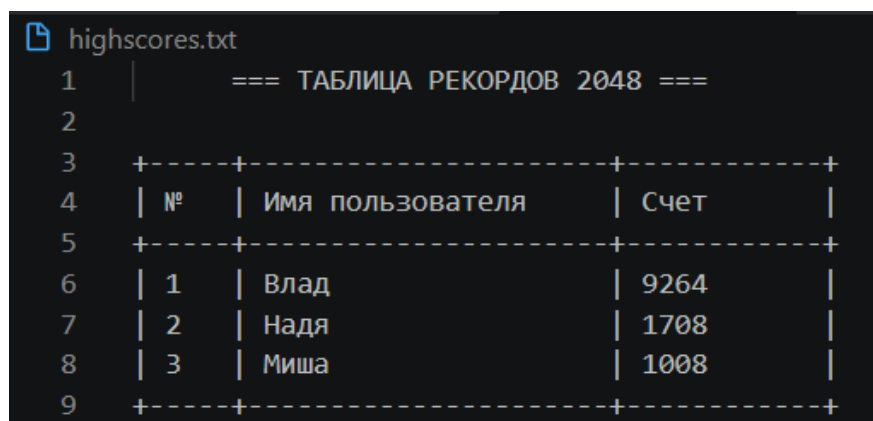


Рисунок 11 – Окно поражения

Экраны победы и поражения реализованы в виде полноценных финальных заставок, логически завершающих игровой цикл.

6.4 Таблица рекордов.

При победе, поражении или перехода в меню результат игры с именем пользователя заносится в таблицу рекордов, отсортированная по убыванию очков игры хранящая топ 25 игроков. На рисунке 12 представлена таблица рекордов.



```
highscores.txt
1      === ТАБЛИЦА РЕКОРДОВ 2048 ===
2
3      +-----+-----+-----+
4      | №   | Имя пользователя | Счет   |
5      +-----+-----+-----+
6      | 1   | Влад             | 9264   |
7      | 2   | Надя             | 1708   |
8      | 3   | Миша             | 1008   |
9      +-----+-----+-----+
```

Рисунок 11 – Таблица рекордов

Таблица рекордсменов – ключевой аспект игры, добавляющий чувство азарта и стремления набрать как можно больше очков.

Заключение

В ходе выполнения курсовой работы была разработана программа — компьютерная игра «2048», реализованная на языке программирования Python с использованием библиотеки `pygame` и ассемблерной вставки. Данный проект позволил закрепить и углубить ключевые навыки разработки программного обеспечения, а также продемонстрировал эффективное применение современных инструментов для решения практических задач.

Основные достижения и реализованные функции:

1. Реализована полная игровая механика, включающая:

- алгоритм сдвига плиток во всех четырёх направлениях;
- слияние одинаковых плиток с подсчётом очков;
- генерацию новых плиток после каждого хода.

2. Применены низкоуровневые оптимизации:

- реализация операции сдвига строки на уровне ассемблера;
- подключение динамической библиотеки через `ctypes`;
- работа с регистрами процессора напрямую.

3. Разработан графический пользовательский интерфейс:

- экран приветствия с вводом имени игрока;
- наглядное отображение игрового поля с цветовой индикацией плиток;
- панель текущего счёта и личного рекорда.

4. Реализована система сохранения результатов:

- запись результатов в файл таблицы рекордов;
- хранение и отображение лучшего результата сессии.

Выполненная работа продемонстрировала эффективность связки Python и ассемблера для разработки логических игр. Программа успешно реализует все основные функции классической игры «2048», сочетая при этом высокоуровневую логику на Python с низкоуровневой оптимизацией на уровне машинного кода. Приобретённые в ходе работы навыки имеют важное

значение для дальнейшего профессионального роста в области разработки программного обеспечения.

Список используемых материалов

1. Лутц, М. Изучаем Python : в 2 т., Т. 1 : пер. с англ. / М. Лутц. – 5-е изд. – Санкт-Петербург : Диалектика, 2019. – 832 с. – ISBN 978-5-907144-06-4. Т. 2 : пер. с англ. / М. Лутц. – 5-е изд. – Санкт-Петербург : Диалектика, 2020. – 1072 с. – ISBN 978-5-907144-07-1.
2. Поляков, А. К. Языки ассемблера для процессоров x86 и инлайн-ассемблер C/C++ / А. К. Поляков. – Москва : ИНФРА-М, 2022. – 315 с. – ISBN 978-5-16-017122-3.
3. Рамальо, Л. Python. К вершинам мастерства : пер. с англ. / Л. Рамальо. – Москва : ДМК Пресс, 2016. – 506 с. – ISBN 978-5-97060-384-0.
4. Ирвин, К. Язык ассемблера для процессоров x86 : пер. с англ. / К. Ирвин. – 7-е изд. – Москва : Вильямс, 2016. – 1008 с. – ISBN 978-5-8459-2045-4.

Приложение А. Листинг программы.

Файл main.py

```
import pygame
import os

from constants import *
from scores import *
from logic import *
from drawing import *

def main():
    pygame.init()
    screen = pygame.display.set_mode((SCREEN_WIDTH,
SCREEN_HEIGHT))
    pygame.display.set_caption("Game 2048")

    icon_path = os.path.join(os.path.dirname(__file__),
"icon.png")
    if os.path.exists(icon_path):
        pygame.display.set_icon(pygame.image.load(icon_
path))

    clock = pygame.time.Clock()

    fonts = {
        "xl": pygame.font.SysFont("Arial", 110,
bold=True),
        "lg": pygame.font.SysFont("Arial", 50,
bold=True),
```

```

        "md": pygame.font.SysFont("Arial", 26,
bold=True),
        "sm": pygame.font.SysFont("Arial", 22,
bold=True),
    }

    btn_start      = pygame.Rect(100, 340, 200, 55)
    btn_menu       = pygame.Rect(270, 10, 120, 35)
    btn_restart    = pygame.Rect(270, 52, 120, 35)
    btn_new_game   = pygame.Rect(100, 310, 200, 50)

    state          = "MENU"
    user_name      = ""
    show_error     = False
    matrix          = create_matrix()
    score           = 0
    best_score     = max((s for _, s in load_scores()),
default=0)
    game_over      = False
    won             = False
    score_saved    = False

    running = True
    while running:
        screen.fill(BG_COLOR)
        mouse_pos = pygame.mouse.get_pos()

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                if not score_saved and score > 0:

```

```

        save_score(user_name, score)
    running = False

    if state == "MENU":
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_RETURN:
                if user_name.strip():
                    matrix = create_matrix()
                    score = 0
                    game_over = won =
score_saved = False

                    show_error = False
                    state = "PLAYING"
                else:
                    show_error = True
            elif event.key ==
pygame.K_BACKSPACE:
                user_name = user_name[:-1]
                show_error = False
            elif event.unicode.isprintable()
and len(user_name) < 12:
                user_name += event.unicode
                show_error = False

        if event.type == pygame.MOUSEBUTTONDOWN
and btn_start.collidepoint(mouse_pos):
            if user_name.strip():
                matrix = create_matrix()
                score = 0

```

```

game_over = won = score_saved =
False

show_error = False
state = "PLAYING"
else:
    show_error = True

elif state == "PLAYING":
    if not game_over and not won and
event.type == pygame.KEYDOWN:
        direction = {
            pygame.K_UP:    "Up",
            pygame.K_DOWN:  "Down",
            pygame.K_LEFT:  "Left",
            pygame.K_RIGHT: "Right",
        }.get(event.key)

        if direction:
            matrix, gained = move(matrix,
direction)

            score += gained
            if score > best_score:
                best_score = score
            if not score_saved:
                if check_win(matrix):
                    won = True
                    save_score(user_name,
score)

                    score_saved = True

```

```

elif
check_game_over(matrix):
    game_over = True
    save_score(user_name,
score)
    score_saved = True

    if event.type ==
pygame.MOUSEBUTTONDOWN:
        if
btn_menu.collidepoint(mouse_pos):
            if not score_saved and score >
0:
                save_score(user_name,
score)
                state = "MENU"
            elif
btn_restart.collidepoint(mouse_pos):
                if not score_saved and score >
0:
                    save_score(user_name,
score)
                    matrix = create_matrix()
                    score = 0
                    game_over = won = score_saved =
False
            elif (game_over or won) and
btn_new_game.collidepoint(mouse_pos):
                matrix = create_matrix()
                score = 0

```

```

        game_over = won = score_saved =
False

        if state == "MENU":
            draw_menu(screen, fonts, user_name,
btn_start, show_error)
        elif state == "PLAYING":
            draw_game(screen, matrix, score,
best_score, user_name,
                        fonts, btn_menu, btn_restart,
btn_new_game, game_over, won)

        pygame.display.flip()
        clock.tick(60)

    pygame.quit()

if __name__ == "__main__":
    main()

```

Файл logic.py

```

import ctypes
import os
import random
from constants import GRID_SIZE

_lib_name = "stack_asm.dll" if os.name == "nt" else
"stack_asm.so"

```

```

_lib_path =
os.path.join(os.path.dirname(os.path.abspath(__file__)))
, _lib_name)
_lib = ctypes.CDLL(_lib_path)
_lib.stack_row.argtypes =
[ctypes.POINTER(ctypes.c_int), ctypes.c_int]
_lib.stack_row.restype = None

def add_new_tile(matrix):
    empty_cells = [(r, c) for r in range(GRID_SIZE) for
c in range(GRID_SIZE) if matrix[r][c] == 0]
    if empty_cells:
        r, c = random.choice(empty_cells)
        matrix[r][c] = random.choice([2, 4])

def create_matrix():
    matrix = [[0] * GRID_SIZE for _ in
range(GRID_SIZE)]
    add_new_tile(matrix)
    add_new_tile(matrix)
    return matrix

def stack(m):
    result = [row[:] for row in m]
    for row in result:
        c_row = (ctypes.c_int * GRID_SIZE)(*row)
        _lib.stack_row(c_row, GRID_SIZE)
        row[:] = list(c_row)
    return result

```

```

def combine(m):
    gained = 0
    for i in range(GRID_SIZE):
        for j in range(GRID_SIZE - 1):
            if m[i][j] != 0 and m[i][j] == m[i][j + 1]:
                m[i][j] *= 2
                m[i][j + 1] = 0
                gained += m[i][j]
    return m, gained

def reverse(m):
    return [row[::-1] for row in m]

def transpose(m):
    return [list(row) for row in zip(*m)]

def check_game_over(matrix):
    if any(0 in row for row in matrix):
        return False
    for i in range(GRID_SIZE):
        for j in range(GRID_SIZE - 1):
            if matrix[i][j] == matrix[i][j + 1] or
matrix[j][i] == matrix[j + 1][i]:
                return False
    return True

def check_win(matrix):
    return any(matrix[r][c] == 2048 for r in
range(GRID_SIZE) for c in range(GRID_SIZE))

```

```

def move(matrix, direction):
    old = [row[:] for row in matrix]
    total_gained = 0

    if direction == "Left":
        matrix = stack(matrix); matrix, g =
combine(matrix); total_gained += g; matrix =
stack(matrix)
    elif direction == "Right":
        matrix = reverse(matrix); matrix =
stack(matrix); matrix, g = combine(matrix);
total_gained += g; matrix = stack(matrix); matrix =
reverse(matrix)
    elif direction == "Up":
        matrix = transpose(matrix); matrix =
stack(matrix); matrix, g = combine(matrix);
total_gained += g; matrix = stack(matrix); matrix =
transpose(matrix)
    elif direction == "Down":
        matrix = transpose(matrix); matrix =
reverse(matrix); matrix = stack(matrix); matrix, g =
combine(matrix); total_gained += g; matrix =
stack(matrix); matrix = reverse(matrix); matrix =
transpose(matrix)

    if matrix == old:
        return matrix, 0

    add_new_tile(matrix)
    return matrix, total_gained

```

Файл drawing.py

```
import pygame
from constants import *

def draw_text_centered(surface, font, text, color,
rect):
    rendered = font.render(text, True, color)
    x = rect.centerx - rendered.get_width() // 2
    y = rect.centery - rendered.get_height() // 2
    surface.blit(rendered, (x, y))

def draw_menu(surface, fonts, user_name, btn_start,
show_error):
    title = fonts["xl"].render("2048", True, TEXT_DARK)
    surface.blit(title, (SCREEN_WIDTH // 2 -
title.get_width() // 2, 60))

    surface.blit(fonts["sm"].render("Введите имя:",
True, TEXT_DARK),
                (SCREEN_WIDTH // 2 - 55, 220))

    pygame.draw.rect(surface, COLORS[0], (80, 248, 240,
45), border_radius=6)
    name_surface = fonts["md"].render(user_name, True,
TEXT_LIGHT)
    surface.blit(name_surface, (SCREEN_WIDTH // 2 -
name_surface.get_width() // 2, 256))
```

```

    if show_error:
        err = fonts["sm"].render("Имя не может быть
пустым!", True, ERROR_COLOR)
        surface.blit(err, (SCREEN_WIDTH // 2 -
err.get_width() // 2, 302))

    btn_color = (170, 152, 130) if not
user_name.strip() else PANEL_COLOR
    pygame.draw.rect(surface, btn_color, btn_start,
border_radius=12)
    draw_text_centered(surface, fonts["md"], "ИГРАТЬ",
TEXT_LIGHT, btn_start)

def draw_board(surface, matrix, fonts):
    for row in range(GRID_SIZE):
        for col in range(GRID_SIZE):
            value = matrix[row][col]
            rect = pygame.Rect(
                col * CELL_STEP + GRID_OFFSET_X,
                row * CELL_STEP + TOP_RESERVE,
                CELL_SIZE,
                CELL_SIZE,
            )
            color = COLORS.get(value, TILE_BIG)
            pygame.draw.rect(surface, color, rect,
border_radius=6)

            if value != 0:
                font = fonts["sm"] if value >=
1024 else fonts["md"]

```

```

        text_color = TEXT_DARK if value <= 4
    else TEXT_LIGHT

        draw_text_centered(surface, font,
str(value), text_color, rect)

def draw_hud(surface, score, best_score, user_name,
fonts, btn_menu, btn_restart):

    surface.blit(fonts["sm"].render(f"Игрок:
{user_name}", True, TEXT_DARK), (10, 5))

    surface.blit(fonts["sm"].render(f"Рекорд:
{best_score}", True, TEXT_DARK), (10, 28))

    surface.blit(fonts["sm"].render(f"Счёт: {score}",
True, TEXT_DARK), (10, 51))

    for button, label in [(btn_menu, "В МЕНЮ"),
(btn_restart, "РЕСТАРТ")]:
        pygame.draw.rect(surface, PANEL_COLOR, button,
border_radius=6)
        draw_text_centered(surface, fonts["sm"], label,
TEXT_LIGHT, button)

def draw_end_overlay(surface, fonts, title,
title_color, subtitle, btn_rect):
    overlay = pygame.Surface((SCREEN_WIDTH,
SCREEN_HEIGHT), pygame.SRCALPHA)
    overlay.fill((255, 255, 255, 180))
    surface.blit(overlay, (0, 0))

```

```

        title_surface = fonts["lg"].render(title, True,
title_color)
        surface.blit(title_surface, (SCREEN_WIDTH // 2 -
title_surface.get_width() // 2, 200))

        sub = fonts["sm"].render(subtitle, True, TEXT_DARK)
        surface.blit(sub, (SCREEN_WIDTH // 2 -
sub.get_width() // 2, 265))

        pygame.draw.rect(surface, PANEL_COLOR, btn_rect,
border_radius=10)
        draw_text_centered(surface, fonts["md"], "ХОБАЯ
ИГПА", TEXT_LIGHT, btn_rect)

def draw_game(surface, matrix, score, best_score,
user_name, fonts,
                btn_menu, btn_restart, btn_new_game,
game_over, won):
    draw_hud(surface, score, best_score, user_name,
fonts, btn_menu, btn_restart)
    draw_board(surface, matrix, fonts)

    if won:
        draw_end_overlay(surface, fonts, "ПОБЕДА!", (0,
140, 0),
                        f"Счет: {score}",
btn_new_game)
    elif game_over:

```

```

        draw_end_overlay(surface, fonts, "GAME OVER",
(160, 0, 0),
                                f"Счёт: {score}",
btn_new_game)

```

Файл scores.py

```

import os
from constants import HIGHSCORES_FILE

def load_scores():
    scores = []
    if not os.path.exists(HIGHSCORES_FILE):
        return scores
    try:
        with open(HIGHSCORES_FILE, "r", encoding="utf-
8") as file:
            for line in file:
                if "|" in line and "Имя" not in line:
                    parts = line.split("|")
                    if len(parts) >= 4:
                        name  = parts[2].strip()
                        value = parts[3].strip()
                        if value.isdigit():
                            scores.append((name,
int(value)))
    except Exception:
        pass
    return scores

def save_score(user_name, score):

```

```

        if score == 0: return
        all_scores = load_scores()
        all_scores.append((user_name, score))
        all_scores.sort(key=lambda item: item[1],
reverse=True)

        with open(HIGHSCORES_FILE, "w", encoding="utf-8")
as file:
            file.write("      === ТАБЛИЦА РЕКОРДОВ 2048
===\n\n")
            separator = f"+{'-' * 5}+{'-' * 22}+{'-' *
12}+\n"
            file.write(separator)
            file.write(f"| {'№':<3} | {'Имя
пользователя':<20} | {'Счет':<10} |\n")
            file.write(separator)
            for index, (name, value) in
enumerate(all_scores[:25], 1):
                file.write(f"| {index:<3} | {name[:20]:<20}
| {value:<10} |\n")
            file.write(separator)

```

Файл constans.py

```

SCREEN_WIDTH   = 400
SCREEN_HEIGHT  = 550
GRID_SIZE      = 4
CELL_SIZE      = 80
CELL_STEP      = 85
GRID_OFFSET_X  = 35

```

```
TOP_RESERVE    = 160
HIGHSCORES_FILE = "highscores.txt"
```

```
COLORS = {
    0:    (158, 148, 138),
    2:    (238, 228, 218),
    4:    (237, 224, 200),
    8:    (242, 177, 121),
    16:   (245, 149, 99),
    32:   (246, 124, 95),
    64:   (246, 94, 59),
    128:  (237, 207, 114),
    256:  (237, 204, 97),
    512:  (237, 200, 80),
    1024: (237, 197, 63),
    2048: (237, 194, 46),
}
```

```
BG_COLOR      = (187, 173, 160)
PANEL_COLOR   = (143, 122, 102)
TEXT_DARK     = (119, 110, 101)
TEXT_LIGHT    = (249, 246, 242)
TILE_BIG      = (60, 58, 50)
ERROR_COLOR   = (200, 60, 60)
BEST_COLOR    = (98, 175, 98)
```

Файл `stack_asm.dll`

```
#include <string.h>
```

```

__declspec(dllexport) void stack_row(int* row, int
size) {
    int temp[16] = {0};
    int pos = 0;
    int i;

    __asm {
        mov     esi, row
        lea     edi, temp
        xor     ebx, ebx
        xor     ecx, ecx

loop_start:
        cmp     ecx, size
        jge     loop_end

        mov     eax, [esi + ecx * 4]
        test    eax, eax
        jz      skip

        mov     [edi + ebx * 4], eax
        inc     ebx

skip:
        inc     ecx
        jmp     loop_start

loop_end:
        mov     pos, ebx
    }

```

```
    for (i = pos; i < size; i++) {  
        temp[i] = 0;  
    }  
    memcpy(row, temp, size * sizeof(int));  
}
```