

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 "Змейка"»

Выполнил: ст. гр. 25ВВВ4

Пантюхин З. С.

Принял:

Кандидат т.н. Токарев А. Н.

Пенза 2026

Содержание

Введение	5
Постановка задачи.....	6
Выбор решения.....	8
Описание разработки программы.....	10
Модуль game.c	10
Модуль graphics.c	11
Модуль input.c	11
Модуль score.c	11
Отладка и тестирование	12
Описание программы	13
Руководство пользователя	19
Заключение	23
Список используемых источников.....	24
Приложение А. Листинги программы.....	Error! Bookmark not defined.
snake.h	Error! Bookmark not defined.
main.c.....	Error! Bookmark not defined.
game.c	Error! Bookmark not defined.
graphics.c	Error! Bookmark not defined.
input.c	Error! Bookmark not defined.
score.c	Error! Bookmark not defined.
Приложение В. Результаты работы программы	Error! Bookmark not defined.
Главное меню	Error! Bookmark not defined.
past_games.txt.....	Error! Bookmark not defined.
difficulty.txt	Error! Bookmark not defined.
colors.txt	Error! Bookmark not defined.

Введение

Компьютерные игры являются одним из наглядных способов изучения программирования, так как объединяют работу с данными, обработку ввода, построение интерфейса и организацию непрерывного цикла выполнения программы. Даже простая игра требует продуманной структуры: необходимо хранить состояние игрового поля, реагировать на действия пользователя, изменять положение объектов и своевременно выводить результат на экран.

В данной курсовой работе рассматривается разработка консольного приложения Snake на языке C. Игра «Змейка» выбрана как классический пример программы сложной структуры: в ней присутствует главный цикл, обработка клавиатуры, логика движения, столкновения, система очков, настройки и сохранение результатов в файлы.

Разработанная программа запускается в консоли Windows и использует стандартные средства языка C, библиотеку `conio.h` для чтения нажатий клавиш, функции Windows API для изменения цвета текста и функцию `Sleep` для регулирования скорости игры. Такой подход позволяет создать интерактивное приложение без использования графических библиотек и при этом показать основные принципы модульного программирования.

Актуальность работы заключается в том, что консольные приложения остаются удобной учебной средой для освоения алгоритмов, структур данных и организации проекта. На примере Snake можно проследить полный цикл разработки: от постановки задачи и проектирования модулей до тестирования, сохранения пользовательских настроек и подготовки руководства пользователя.

Цель курсовой работы — разработать консольную игру Snake, реализующую управление змейкой, появление фрукта, подсчёт очков, выбор сложности, настройку цветовой палитры, сохранение рекорда и просмотр истории последних игр.

Постановка задачи

Необходимо разработать консольное приложение Snake, в котором пользователь управляет змейкой на прямоугольном игровом поле. Программа должна обеспечивать движение змейки, обработку столкновений с собственным хвостом, появление фрукта в случайной клетке и увеличение длины змейки после его поедания.

Игровое поле должно иметь фиксированные размеры 20 на 20 клеток. По краям поля необходимо выводить рамку, отделяющую игровую область от остальной части консольного окна. Голова змейки, элементы хвоста и фрукт должны отображаться различными символами, чтобы пользователь мог легко различать игровые объекты.

Управление должно выполняться с клавиатуры. Для движения используются клавиши W, A, S, D, а также клавиши со стрелками. Программа должна запрещать мгновенный разворот змейки в противоположную сторону, так как такой разворот приводит к некорректному самопересечению.

При выходе за границы поля змейка должна появляться с противоположной стороны. Такое решение делает игровой процесс более динамичным и позволяет не завершать игру при касании рамки. Завершение игры происходит при столкновении головы змейки с элементом собственного хвоста.

В программе необходимо реализовать систему очков. За каждый съеденный фрукт игрок получает 10 очков. Лучший результат должен сохраняться в отдельном файле и загружаться при следующем запуске приложения. Кроме того, программа должна записывать историю завершённых игр с датой, временем и итоговым счётом.

Также требуется предусмотреть меню с пунктами запуска игры, настроек, просмотра последних игр и выхода. В настройках пользователь

должен иметь возможность выбрать цветовую палитру и уровень сложности. Сложность определяет скорость обновления игрового цикла.

Программа должна быть разбита на несколько модулей, каждый из которых отвечает за отдельную часть функциональности:

- точку входа;
- игровую логику;
- отображение;
- ввод;
- работу со счётом;
- общие объявления.

Выбор решения

Для решения задачи выбран язык программирования C, так как он позволяет явно работать с массивами, файлами, функциями и глобальным состоянием программы. Проект создан как консольное приложение Visual Studio. Такой формат соответствует учебной задаче и не требует дополнительных графических библиотек.

Основой программы является бесконечный игровой цикл. На каждой итерации выполняются три ключевых действия:

- отрисовка текущего состояния поля;
- обработка пользовательского ввода;
- пересчёт логики игры.

После этого программа делает паузу с помощью Sleep, длительность которой зависит от выбранного уровня сложности.

Состояние игры хранится в наборе глобальных переменных, объявленных в заголовочном файле snake.h. Координаты головы змейки задаются переменными x и y. Координаты фрукта хранятся в fruitX и fruitY. Хвост представлен двумя массивами tailX и tailY, где элементы с одинаковыми индексами образуют координаты одной части хвоста. Количество элементов хвоста хранится в nTail.

Для направления движения используется перечисление enum eDirection со значениями STOP, LEFT, RIGHT, UP и DOWN. Это делает код более понятным по сравнению с хранением направления в виде произвольного числа.

Отображение реализовано в консоли. Функция Draw очищает экран, выводит верхнюю рамку, затем построчно строит игровое поле и в конце показывает текущий счёт и лучший результат. Для изменения цвета используется функция SetConsoleTextAttribute, что позволяет выделять змейку, фрукт и границы разными цветами.

Ввод пользователя выполняется без ожидания нажатия Enter. Для проверки наличия нажатой клавиши используется `_kbhit`, а для чтения символа — `_getch`. В проекте проверка нажатия вынесена в отдельную функцию `check_keypress`.

Для хранения настроек и результатов применяются текстовые файлы. Файл `save.txt` содержит лучший результат. Файл `past_games.txt` используется для истории завершённых игр. Файл `colors.txt` хранит выбранные цвета, а `difficulty.txt` — текущую скорость игры. Использование простых текстовых файлов упрощает отладку и позволяет быстро проверить корректность сохранения данных.

Программа разделена на модули:

- `main.c`;
- `game.c`;
- `graphics.c`;
- `input.c`;
- `score.c`;
- `snake.h`.

Такое разбиение соответствует нисходящему подходу: сначала выделяются основные функции программы, затем каждая крупная задача реализуется в отдельном файле.

Описание разработки программы

Разработка программы выполнялась поэтапно. На первом этапе была создана общая структура проекта и заголовочный файл `snake.h`, в котором определены константы размеров поля, перечисление направлений, внешние переменные и прототипы функций.

На втором этапе была реализована точка входа в `main.c`. Функция `main` вызывает `Menu`, затем `Setup`, после чего запускает основной цикл игры. В цикле последовательно вызываются `Draw`, `Input` и `Logic`. Эта структура делает порядок выполнения программы простым и наглядным.

Модуль `game.c`

Модуль `game.c` содержит функции `Setup`, `Logic`, `ChangeDifficulty`, `LoadDifficulty`, `SaveDifficulty` и `Menu`. Функция `Setup` выполняет начальную настройку: инициализирует генератор случайных чисел, загружает сохранённый рекорд, цвета и сложность, размещает голову змейки по центру поля, создаёт фрукт и обнуляет счёт.

Функция `Logic` отвечает за изменение состояния игры. Сначала она сдвигает элементы хвоста вслед за головой, затем изменяет координаты головы в зависимости от текущего направления. После этого выполняется обработка выхода за границы поля: если координата выходит за допустимый диапазон, она переносится на противоположную сторону.

Проверка проигрыша также выполняется в `Logic`. Если координаты головы совпадают с координатами одного из элементов хвоста, программа сохраняет результат, очищает экран, выводит сообщение `GAME OVER` и завершает работу. Если голова совпадает с координатами фрукта, счёт увеличивается на 10, фрукт получает новые случайные координаты, а длина хвоста увеличивается.

Модуль graphics.c

Модуль graphics.c отвечает за визуальную часть. В нём реализованы функции SetColor, SetColorPalette, Draw, SaveColors, LoadColors и ChangeColors. Пользователь может выбрать одну из трёх палитр: зелёную, синюю или жёлтую. Выбранная палитра сохраняется в файл colors.txt и используется при следующем запуске игры.

Модуль input.c

Модуль input.c отвечает за управление. Функция Input проверяет наличие нажатой клавиши и изменяет направление движения. Для удобства поддерживаются как латинские клавиши W, A, S, D, так и коды клавиш со стрелками. Клавиша X сохраняет результат и завершает программу.

Модуль score.c

Модуль score.c реализует работу с результатами. Функция LoadHighScore считывает лучший результат из save.txt. Функция SaveScore обновляет рекорд, если текущий счёт больше сохранённого, и добавляет запись в past_games.txt с датой, временем и количеством набранных очков.

Меню программы позволяет начать игру, перейти в настройки, просмотреть историю последних игр или выйти из приложения. В разделе настроек пользователь может изменить цветовую палитру и скорость игры. Скорость задаётся числом миллисекунд задержки между кадрами: 150 для лёгкого режима, 100 для обычного и 60 для сложного.

В результате разработки получена программа, состоящая из нескольких связанных модулей. Каждый модуль имеет ограниченную область ответственности, что упрощает понимание исходного кода и дальнейшее расширение проекта.

Отладка и тестирование

Отладка программы проводилась после реализации каждого основного модуля. Сначала проверялась корректность запуска приложения и перехода из меню в игровой режим. Затем тестировалась отрисовка поля, положение головы змейки, вывод фрукта, рамки, текущего счёта и рекорда.

При тестировании управления проверялись все поддерживаемые клавиши: W, A, S, D и стрелки. Отдельно проверялось ограничение на разворот в противоположную сторону. Например, при движении вправо нажатие клавиши A не должно сразу менять направление на LEFT.

Игровая логика проверялась на нескольких ситуациях: движение без нажатия клавиш, проход через границы поля, поедание фрукта, увеличение счёта, увеличение хвоста и столкновение с собственным хвостом. При выходе за правую границу координата x должна становиться равной 0, а при выходе за левую — WIDTH - 1.

Сохранение данных проверялось через файлы save.txt, past_games.txt, colors.txt и difficulty.txt. После завершения игры результат должен добавляться в историю, а при превышении рекорда файл save.txt должен обновляться. После изменения цветовой палитры и сложности выбранные значения должны сохраняться и загружаться при следующем запуске.

Также проверялась реакция меню на неправильный ввод. Если пользователь вводит значение вне допустимого диапазона, программа выводит сообщение об ошибке и предлагает повторить выбор. Это предотвращает переход в неопределённое состояние.

По итогам тестирования основные функции программы работают корректно: приложение запускается, меню обрабатывает команды пользователя, игра отображается в консоли, управление реагирует без необходимости нажимать Enter, очки начисляются, настройки и результаты сохраняются в текстовые файлы.

Описание программы

Программа Snake представляет собой консольное приложение для операционной системы Windows. Она предназначена для интерактивной игры, в которой пользователь управляет змейкой, собирает фрукты и старается набрать максимальное количество очков.

Входными данными программы являются нажатия клавиш пользователем и сохранённые текстовые файлы с настройками. Клавиши используются для выбора пунктов меню и управления движением змейки. Файлы colors.txt, difficulty.txt и save.txt позволяют восстановить выбранные параметры и рекорд при новом запуске.

Выходными данными являются изображение игрового поля в консоли, сообщения меню, текущий счёт, лучший результат, сообщение о завершении игры и записи в файле past_games.txt.

Основные функции программы:

- Menu — выводит главное меню и обрабатывает выбор пользователя; Setup — подготавливает начальное состояние игры;
- Draw — выводит поле и игровые объекты;
- Input — обрабатывает клавиатуру;
- Logic — пересчитывает координаты и проверяет игровые события;
- LoadHighScore и SaveScore — работают с результатами.

Константы WIDTH и HEIGHT задают размеры игрового поля. В текущей версии оба значения равны 20. Такой размер удобен для консольного окна и позволяет сохранять читаемость изображения.

Для хранения хвоста используются массивы tailX и tailY размером 100 элементов. При каждом обновлении игры координаты элементов хвоста сдвигаются: первая часть хвоста занимает предыдущее положение головы, следующая часть занимает предыдущее положение первой части и так далее.

Фрукт появляется в случайной позиции с помощью функции rand. После поедания фрукта координаты fruitX и fruitY обновляются, а переменная nTail увеличивается на единицу. Счёт увеличивается на 10 очков.

Цветовое оформление реализовано через числовые коды цветов консоли Windows. В программе доступны три палитры, каждая из которых задаёт отдельные цвета для змейки, фрукта и рамки.

Программа не требует установки дополнительных библиотек. Для запуска достаточно исполняемого файла snake.exe и файлов настроек, которые создаются и обновляются во время работы приложения.

На рисунке 1 представлена схема взаимодействия программы.

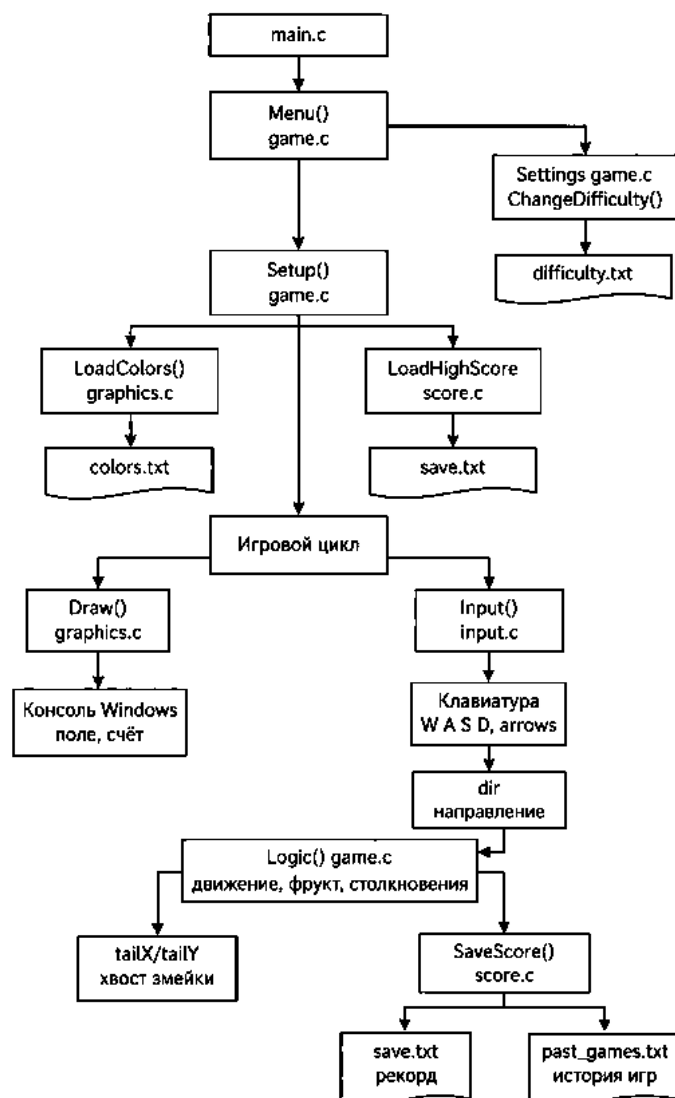


Рисунок 1 - Схема взаимодействия программы

На рисунке 2, рисунке3, рисунке4 представлена блок-схема функции Logic().

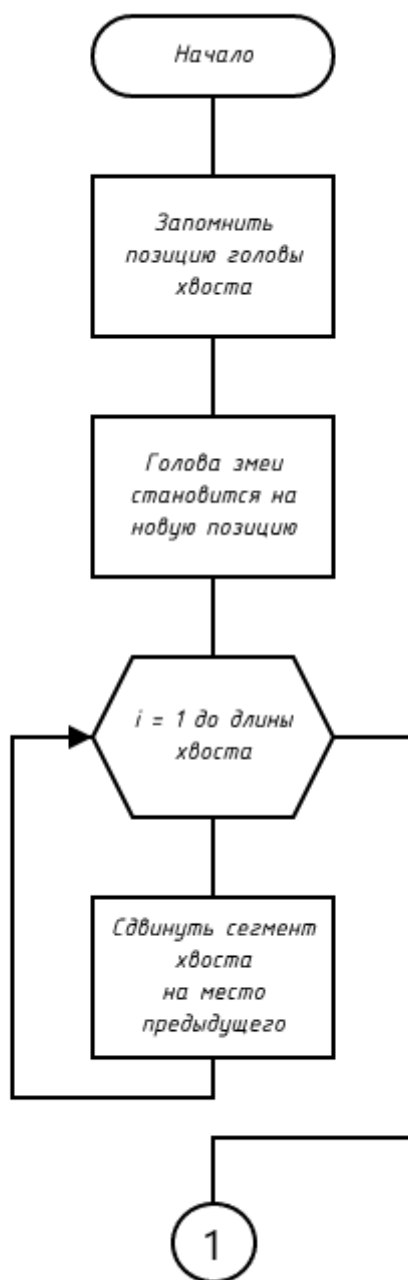


Рисунок 2 - Блок-схема функции Logic()

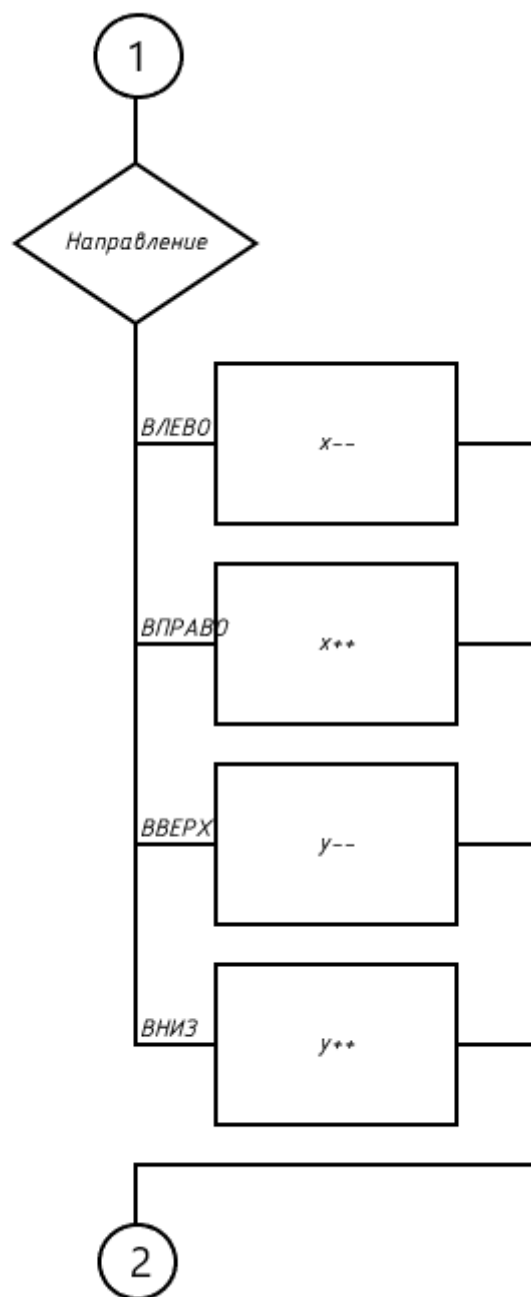


Рисунок 3 - Блок-схема функции Logic()

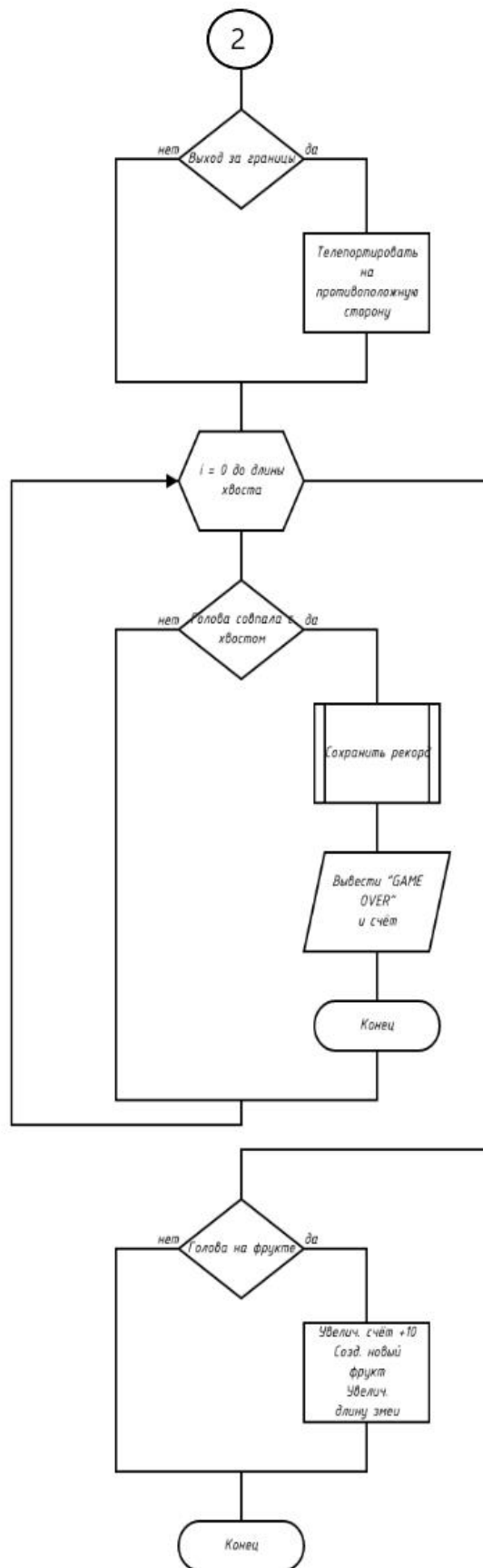


Рисунок 4 - Блок-схема функции Logic()

На рисунке 5 представлена блок-схема ассемблерской вставки из input.c.



Рисунок 5 - Блок-схема ассемблерской вставки

Руководство пользователя

Для запуска программы необходимо открыть файл `snake.exe`. После запуска на экране появляется главное меню (Рисунок 6) с четырьмя пунктами: Start Game, Settings, Watch last games и Exit.

Чтобы начать игру, нужно ввести 1 и подтвердить выбор. После этого откроется игровое поле (Рисунок 7). Голова змейки обозначается символом O, хвост — символом o, фрукт — символом F, а границы поля — символом #.

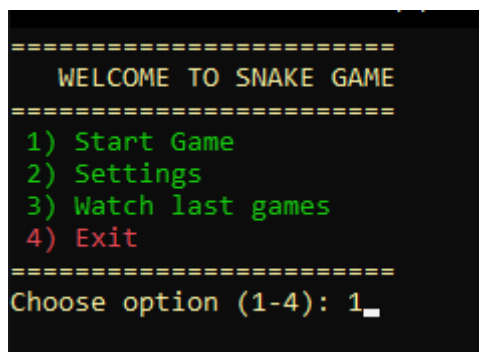


Рисунок 6 - Главное меню

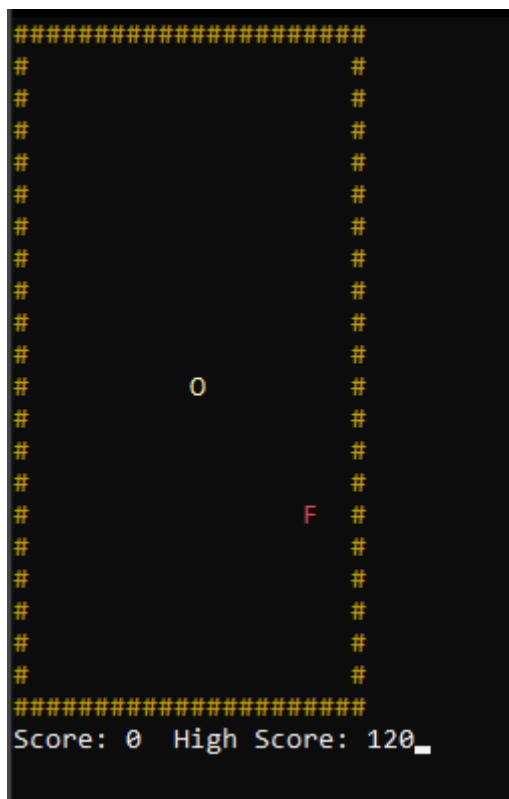


Рисунок 7 - Игровое поле

Управление змейкой выполняется клавишами W, A, S, D или клавишами со стрелками. W и стрелка вверх перемещают змейку вверх, S и стрелка вниз — вниз, A и стрелка влево — влево, D и стрелка вправо — вправо. Для выхода из игры можно нажать X.

Цель игры — собирать фрукты и не сталкиваться с собственным хвостом. За каждый фрукт начисляется 10 очков. После поедания фрукта змейка становится длиннее, поэтому управлять ей постепенно становится сложнее.

Если змейка выходит за границу поля, она появляется с противоположной стороны. Например, при выходе за правый край она появляется слева. Это правило позволяет продолжать игру и использовать всё поле.

Для изменения настроек в главном меню нужно выбрать пункт 2 (Рисунок 8). В открывшемся меню (Рисунок 9) доступны изменение цветовой палитры и изменение сложности. Цветовая палитра (Рисунок 10) влияет на цвета змейки, фрукта и рамки. Сложность влияет на скорость движения змейки.

```
=====
  WELCOME TO SNAKE GAME
=====
1) Start Game
2) Settings
3) Watch last games
4) Exit
=====
Choose option (1-4): 2
```

Рисунок 8 – Открытие меню настроек

```
=== SETTINGS ===
1) Change Color Palette
2) Change Difficulty
3) Back to Main Menu
Choose option (1-3): 1
```

Рисунок 9 - Меню настроек

```
=== Color Palettes ===  
  
1) Green Palette (Snake: Green, Fruit: Red, Borders: White)  
2) Blue Palette (Snake: Cyan, Fruit: Purple, Borders: Blue)  
3) Yellow Palette (Snake: Yellow, Fruit: Red, Borders: Yellow)  
  
Choose palette (1-3): _
```

Рисунок 10 - Выбор цветовой палитры

В меню сложности (Рисунок 11) доступны три режима: Easy, Normal и Hard. В лёгком режиме задержка между кадрами больше, поэтому змейка движется медленнее. В сложном режиме задержка меньше, поэтому требуется быстрее реагировать на изменение ситуации.

```
=== Difficulty Settings ===  
  
1) Easy (slow)  
2) Normal (medium)  
3) Hard (fast)  
  
Choose option (1-3): _
```

Рисунок 11 - Меню сложности

Для просмотра истории игр (Рисунок 12) нужно выбрать пункт Watch last games. Программа откроет список завершённых игр (Рисунок 13), в котором указаны дата, время и набранный счёт. Для возврата в меню нужно нажать любую клавишу.

```
=====
      WELCOME TO SNAKE GAME
=====
1) Start Game
2) Settings
3) Watch last games
4) Exit
=====
Choose option (1-4): 3_
```

Рисунок 12 - Открытие истории игр

```
=== LAST GAMES ===  
  
08-05-2026 13:37:35 - 120  
08-05-2026 14:25:07 - 10  
08-05-2026 14:25:42 - 10  
08-05-2026 14:25:48 - 0  
08-05-2026 14:26:42 - 10  
08-05-2026 14:27:19 - 0  
  
Press any key to return to menu...
```

Рисунок 13 - Список завершённых игр

Для завершения работы из главного меню нужно выбрать пункт Exit. Программа очистит экран, выведет сообщение Thanks for playing! и завершит выполнение.

Заключение

В ходе курсовой работы было разработано консольное приложение Snake на языке C и ASM. Программа реализует основные правила классической игры: движение змейки, сбор фрукта, увеличение хвоста, подсчёт очков и завершение игры при столкновении с собственным хвостом.

В процессе разработки были применены принципы модульного программирования. Исходный код разделён на несколько файлов, каждый из которых отвечает за конкретную часть программы: главный цикл, игровую логику, ввод, вывод, настройки и сохранение результатов.

Дополнительно реализованы пользовательские настройки: выбор цветовой палитры и уровня сложности. Настройки сохраняются в текстовые файлы, поэтому программа может восстанавливать их при последующих запусках. Также реализовано сохранение лучшего результата и истории последних игр.

Разработанная программа соответствует поставленной задаче и демонстрирует применение нисходящего подхода к созданию программы сложной структуры. Полученный проект может быть расширен: например, можно добавить паузу, препятствия на поле, таблицу рекордов, несколько режимов игры или проверку появления фрукта внутри хвоста.

Список используемых источников

1. Язык программирования С / Б. У. Керниган, Д. М. Ритчи. – М.: Вильямс, 2019. – 304 с.
2. С/С++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб.: Питер, 2021. – 432 с.
3. MicrosoftLearn. ConsoleFunctions [Электронный ресурс]. – URL: <https://learn.microsoft.com/en-us/windows/console/console-functions> (дата обращения: 01.05.2026).
4. MicrosoftLearn. SetConsoleTextAttributefunction [Электронный ресурс]. – URL: <https://learn.microsoft.com/en-us/windows/console/setconsoletextattribute> (дата обращения: 01.05.2026).
5. MicrosoftVisualStudio. Создание консольных приложений на С/С++ [Электронный ресурс]. – URL: <https://learn.microsoft.com/ru-ru/cpp/build/vscpp-step-1-create?view=msvc-170> (дата обращения: 01.05.2026).
6. ISO/IEC 9899:2018. Programming languages — C. – Geneva: International Organization for Standardization, 2018.

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 "Змейка"»

Выполнил: ст. гр. 25BVB4


07.8.
19.08.26

Пантюхин З. 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 Захару Сергеевичу Группа 25ВВВ

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

Исходные данные (технические требования) на проектирование

Игра "Змейка". Требования:

- Управление движением змейки осуществляется с помощью клавиатуры.
- Система набора очков.
- Должна быть предусмотрена рост и уменьшение размеров змейки.
- Изменение звуковой панели.
- Сохранение результатов предыдущих игр в 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

2. Графическая часть

3. Экспериментальная часть

Срок выполнения проекта по разделам

Дата выдачи задания "18" с. 26

Дата защиты проекта " " _____

Руководитель _____

Задание получил "25" февраля 2026 г.

Студент Звонков Владимир З.С.