ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение высшего

профессионального образования

«Белгородский государственный технологический университет

им. В.Г. Шухова»

Институт ИТУС

Кафедра информационных технологий

**Курсовая работа**

**по дисциплине** «Информатика»

**на тему:**

«Разработка клавиатурного тренажера»

Выполнил: студент группы ИТЗ-221

Дмитриев Дмитрий Анатольевич

Проверил: старший преподаватель

Рога Сергей Николаевич

Белгород 2025

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc191379297)

[1. ОБЩАЯ ИНФОРМАЦИЯ И КРАТКАЯ ОЦЕНКА 3](#_Toc191379298)

[1.1. Обзор и краткая оценка современного состояния вопроса 3](#_Toc191379299)

[1.2. Обзор и краткая оценка современного состояния вопроса 3](#_Toc191379300)

[1.3. Цель и задачи работы 3](#_Toc191379301)

[1.4. Краткое описание вопросов, решенных в ходе выполнения работы 4](#_Toc191379302)

[ОСНОВНАЯ ЧАСТЬ 5](#_Toc191379303)

[2. ПОСТАНОВКА ЗАДАЧИ И ОПИСАНИЕ ИСХОДНЫХ ДАННЫХ 5](#_Toc191379304)

[2.1. Описание задачи 5](#_Toc191379305)

[2.2. Исходные данные 5](#_Toc191379306)

[3. СВЕДЕНИЯ, НЕОБХОДИМЫЕ ДЛЯ РЕШЕНИЯ ЗАДАЧИ 6](#_Toc191379307)

[3.1. Основные принципы слепой печати 6](#_Toc191379308)

[3.2. Принципы работы клавиатурных тренажеров 7](#_Toc191379309)

[3.3. Принципы реализации подсчета скорости и анализа ошибок 7](#_Toc191379310)

[4. ОПИСАНИЕ ПОДХОДОВ К РЕШЕНИЮ ЗАДАЧИ 8](#_Toc191379311)

[4.1. Описание возможных подходов 8](#_Toc191379312)

[4.2. Обоснование выбранного подхода 8](#_Toc191379313)

[4.3. Описание алгоритма решения задачи 9](#_Toc191379314)

[5. ОПИСАНИЕ СТРУКТУРЫ ДАННЫХ И АЛГОРИТМОВ 10](#_Toc191379315)

[5.1. Структура данных 10](#_Toc191379316)

[5.2. Алгоритм решения задачи 10](#_Toc191379317)

[5.3. Спецификация алгоритмов 12](#_Toc191379318)

[6. ТЕСТИРОВАНИЕ И РЕЗУЛЬТАТЫ 13](#_Toc191379319)

[6.1. Подходы к тестированию программы 13](#_Toc191379320)

[6.2. Результаты тестирования 14](#_Toc191379321)

[ЗАКЛЮЧЕНИЕ 15](#_Toc191379322)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ 17](#_Toc191379323)

[Приложение А 18](#_Toc191379324)

[Приложение Б 27](#_Toc191379325)

# ВВЕДЕНИЕ

# ОБЩАЯ ИНФОРМАЦИЯ И КРАТКАЯ ОЦЕНКА

* 1. Обзор и краткая оценка современного состояния вопроса

В современном мире, где технологии стали неотъемлемой частью повседневной жизни, навыки работы с компьютером играют ключевую роль в эффективной профессиональной и личной деятельности. Одним из важнейших элементов компьютерной грамотности является умение быстро и точно печатать на клавиатуре. Это важное качество, которое влияет на производительность и удобство работы, особенно в тех профессиях, где работа с текстами и данными является основной частью деятельности.

Современные тренажеры зачастую предлагают ограниченные функции или не соответствуют требованиям пользователей с различным уровнем подготовки. На данный момент существует множество онлайн-ресурсов и программ, однако они не всегда могут предложить оптимальный способ тренировки для конкретного пользователя, не учитывая его индивидуальные особенности.

В современных условиях важно не только обеспечить доступ к обучению, но и создать инструменты, которые смогут эффективно отслеживать процесс обучения, индивидуализировать задания и предоставлять пользователю возможность самому контролировать прогресс в обучении [1].

Актуальность работы обусловлена необходимостью улучшения навыков печати, которые являются неотъемлемой частью профессиональной деятельности в современном мире. Задача создания клавиатурного тренажера актуальна, поскольку многие существующие решения либо не предлагают достаточной гибкости, либо имеют ограничения по функционалу. Работа актуальна, так как решает важную задачу повышения производительности и эффективности работы с компьютером.

* 1. Обзор и краткая оценка современного состояния вопроса

Сегодня существует множество тренажеров для обучения слепой печати, как в виде десктопных программ, так и онлайн-сервисов. Наиболее известными программами являются Stamina, TypingMaster и онлайн-сервисы как TypingClub. Эти программы предлагают различные подходы к обучению: от стандартных уроков с набором текста до более сложных тренажеров с анализом ошибок и статистикой.

Однако несмотря на свою популярность, эти тренажеры часто имеют ограничения, такие как платные подписки и ограничения по количеству использований. Таким образом, есть необходимость в создании более гибкой и адаптированной программы, которая будет учитывать индивидуальные особенности каждого пользователя. Существующие решения не всегда удовлетворяют потребности пользователей, что подтверждает необходимость создания более гибкого и адаптивного тренажера.

* 1. Цель и задачи работы

Целью данной работы является разработка клавиатурного тренажера, который позволит пользователю эффективно развивать навыки быстрой печати. Такой тренажер должен обеспечивать комплексный подход к обучению, включая оценку скорости печати, подсчет ошибок и предоставление статистики.

Для достижения этой цели необходимо решить следующие задачи:

* разработать интерфейс программы, с возможностью выбора сложности;
* реализовать алгоритмы получения различных источников данных для последующего вывода пользователю для печати;
* реализовать алгоритмы для подсчета скорости печати и ошибок;
* провести тестирование программы с целью оценки её эффективности и удобства.

В ходе работы решались задачи по созданию интерфейса, реализации алгоритмов подсчета скорости и анализа ошибок, а также обеспечению персонализированного подхода к обучению.

* 1. Краткое описание вопросов, решенных в ходе выполнения работы

В ходе выполнения работы была решена задача разработки алгоритмов для реализации различных типов упражнений на печать, анализа ошибок и подсчета скорости ввода. Также был разработан простой и интуитивно понятный интерфейс программы, который позволит пользователю легко начать тренировки. В процессе разработки программы была реализована возможность считывания словарей слов и слогов. Данные словари обеспечивают разнообразие заданий и позволяют тренировать пользователей на разных уровнях сложности. Слова и слоги извлекаются из подготовленных файлов и случайным образом подбираются для отображения на экране, что способствует улучшению скорости и точности печати.

# ОСНОВНАЯ ЧАСТЬ

# ПОСТАНОВКА ЗАДАЧИ И ОПИСАНИЕ ИСХОДНЫХ ДАННЫХ

* 1. Описание задачи

Основная цель разработки программы — создание клавиатурного тренажера, который помогает пользователю улучшить скорость и точность печати. Программа должна случайным образом выводить на экран различные элементы — символы, слоги и слова — которые необходимо набрать на клавиатуре. Каждый элемент будет отображаться на экране в определенной последовательности и с ограничением по времени на ввод. Для каждого символа, слога или слова отводится определенное время в зависимости от уровня сложности.

Программа должна оценивать производительность пользователя по количеству правильно введенных символов, а также учитывать ошибки. Для правильности введенных символов начисляются очки, а за ошибки происходит списание баллов. Оценка производится по количеству правильных нажатий, что позволяет пользователю следить за своим прогрессом и тренироваться.

Задача программы заключается в следующем:

* случайным образом генерировать наборы символов, слогов и слов, которые пользователю необходимо ввести;
* оценивать скорость ввода и точность на основе правильных и неправильных символов;
* отображать результат тренировки с учетом времени, потраченного на ввод и количества ошибок;
* обеспечить три уровня сложности, в которых меняются такие параметры, как время на ввод символа, количество секунд на тест и минимальное количество баллов для прохождения теста.

Ключевая цель программы - помочь пользователю улучшить навыки печати через тренировки с временными ограничениями.

* 1. Исходные данные

Для успешного выполнения задачи требуется несколько исходных данных, которые необходимы для генерации текста и оценки работы программы:

* словари символов, слогов и слов. Программа использует словари для случайной генерации текстов. Словарь символов состоит из всех букв английского алфавита, в словарях слогов и слов хранятся данные о слогах и словах на английском языке. Эти данные используются для создания различных уровней сложности и генерируемых текстов;
* алгоритм генерации символов, слогов и слов. Программа использует функцию генерации, которая случайным образом выбирает элементы из словарей и формирует из них тренировочные задания;
* технические данные. Для корректной работы программы также важны параметры, такие как настройки времени, отведенного на ввод символа (в зависимости от уровня сложности), количество очков, требуемых для прохождения теста, и лимит времени на выполнение теста.

Каждый уровень сложности будет изменять время, отведенное на ввод одного символа:

* легкий уровень (3 секунды на символ, минимум 40 правильных символов);
* средний уровень (2 секунды на символ, минимум 60 правильных символов).
* сложный уровень (1 секунда на символ, минимум 80 правильных символов).

Выбор исходных данных и их организация имеют важное значение для адекватной работы тренажера, обеспечивая его гибкость и функциональность. Таким образом, программа генерирует случайные наборы символов, слогов и слов, учитывая параметры сложности, и оценивает пользователя по количеству правильных вводов, что позволяет тренировать навыки быстрой печати и точности.

# СВЕДЕНИЯ, НЕОБХОДИМЫЕ ДЛЯ РЕШЕНИЯ ЗАДАЧИ

* 1. Основные принципы слепой печати

Слепая печать является ключевым навыком для эффективного взаимодействия с компьютером, особенно в профессиях, где требуется быстрая и точная работа с текстом. Этот процесс заключается в наборе текста без использования визуального контакта с клавишами, что позволяет пользователю сфокусироваться исключительно на экране и минимизировать время на поиски клавиш. Как отмечают исследователи в области компьютерной грамотности, освоение слепой печати значительно увеличивает скорость набора текста и снижает количество ошибок [2].

Основным принципом слепой печати является использование всех десяти пальцев, что повышает общую продуктивность и позволяет достигать высоких результатов в быстром наборе. Существуют различные методики обучения, такие как программированные тренажеры, которые предполагают шаг за шагом изучение расположения клавиш с постепенным увеличением сложности.

К принципам относятся:

* сидеть необходимо прямо, не нагибаясь и не наклоняя голову. Угол между позвоночником и бедром и между бедром и голенью должен составлять 90 градусов. Взгляд должен быть направлен вперёд, в центр экрана монитора. Расстояние от глаз до экрана монитора — от 40 до 70 сантиметров. Правильная поза очень важна для здоровья и способствует быстрому обучению печати слепым методом;
* за каждым пальцем при обучении слепому методу печати закреплены свои клавиши, которые должны нажиматься именно им. Исходное положение левой руки: мизинец на клавише «Ф», безымянный на «Ы», средний на «В», указательный на «А». Исходное положение правой руки: мизинец на «Ж», безымянный на «Д», средний на «Л», указательный на «О». Большие пальцы обеих рук располагаются на клавише пробела;
* «ФЫВА» и «ОЛДЖ» — это так называемые домашние клавиши. Чтобы можно было быстро сориентироваться, как расположить руки в стартовой позиции, на клавишах «А» и «О» делают небольшие выступы;
* заниматься изучением десятипальцевого метода набора вслепую на компьютере желательно два-три раза в день. Для начала необходимо пытаться закрывать глаза и, расположив пальцы на домашней строке, печатать свою фамилию. Перемещать необходимо только пальцы. Взгляд должен быть направлен не на клавиатуру, а на экран [3].

Освоение слепой печати является ключевым навыком для повышения производительности работы с компьютером, что подтверждает актуальность создания тренажеров для ее улучшения.

* 1. Принципы работы клавиатурных тренажеров

Клавиатурные тренажеры создаются для того, чтобы пользователи могли развивать свои навыки печати через целенаправленную практику и обратную связь. Важным аспектом является эффективная оценка результатов, которая позволяет пользователю увидеть свой прогресс, что значительно увеличивает мотивацию к дальнейшему обучению.

Основными характеристиками тренажеров являются:

* оценка скорости печати — это важнейшая метрика, которая измеряется количеством символов, набранных за единицу времени;
* оценка точности — точность набора измеряется количеством правильно введенных символов по сравнению с общим числом символов. Ошибки снижают общий результат;
* сложность упражнений — для более эффективного обучения важно постепенно увеличивать сложность упражнений, начиная с простых слов и заканчивая сложными текстами. Сложность может регулироваться временем на ввод каждого символа, минимальным количеством правильных вводов или количеством ошибок.

Современные тренажеры используют различные подходы для обучения, включая подсчет скорости печати и анализ ошибок. Также важным аспектом является персонализация тренировки, что позволяет адаптировать задания в зависимости от уровня пользователя. Эти принципы являются основой для эффективного обучения и тренировки.

* 1. Принципы реализации подсчета скорости и анализа ошибок

Один из ключевых аспектов работы клавиатурного тренажера заключается в точности подсчета скорости ввода и анализе ошибок. Для оценки скорости можно использовать классическую метрику "скорость печати", которая измеряется в символах в минуту (или в других единицах).

Важным фактором является также учет времени на ввод каждого символа. Программа должна точно отслеживать, сколько времени ушло на ввод каждого символа и сколько времени осталось для всего теста. В зависимости от этого можно корректировать оценку и подсчитывать баллы.

Анализ ошибок является не менее важным аспектом. Ошибки должны фиксироваться и снимать очки в зависимости от уровня сложности. Для более сложных уровней тренажер может предусматривать дополнительные штрафы за каждую ошибку.

Также важно, чтобы программа отслеживала, сколько символов пользователь успел ввести до окончания отведенного времени. Это требует точной синхронизации между временем и введенными символами.

Эти принципы и алгоритмы лежат в основе любого эффективного клавиатурного тренажера и являются основой для реализации функционала, необходимого для выполнения поставленной задачи.

Точная оценка ошибок и времени необходима для мотивации пользователей и корректного анализа их прогресса.

# ОПИСАНИЕ ПОДХОДОВ К РЕШЕНИЮ ЗАДАЧИ

* 1. Описание возможных подходов

Для решения задачи создания клавиатурного тренажера было рассмотрено несколько возможных подходов. Основной целью было создать эффективный и быстрый тренажер, который позволит пользователю тренировать навык слепой печати через текстовый интерфейс. Для этого был выбран следующий подход:

* консольное приложение на C++ — данный подход является оптимальным, так как позволяет реализовать функциональность тренажера с минимальными затратами на ресурсы. Взаимодействие с пользователем происходит через текстовые выводы и ввод в командной строке. Этот метод прост в реализации и предоставляет все необходимые средства для решения задачи;
* использование сторонних библиотек — для реализации функционала, такого как работа с текстом и временем, использовались стандартные библиотеки C++, такие как <chrono>, <vector>, <string>, которые обеспечивают высокую производительность при обработке данных и взаимодействии с пользователем. В рамках этой работы не использовались дополнительные графические библиотеки или фреймворки, так как цель заключалась в создании максимально эффективной программы с простым текстовым интерфейсом.

Для разработки тренажера можно использовать разные подходы, включая использование графических интерфейсов или же создание текстовых приложений. В данном случае был выбран текстовый интерфейс, так как он является наиболее эффективным для решения задач, связанных с быстротой обработки данных и взаимодействием с пользователем.

* 1. Обоснование выбранного подхода

Для реализации задачи был выбран язык C++ в качестве основного инструмента для разработки клавиатурного тренажера. C++ обеспечивает высокую скорость работы программы, что критично при обработке большого объема данных и взаимодействии с пользователем в реальном времени [4]. C++ предоставляет разработчику полный контроль над памятью, что позволяет оптимизировать использование ресурсов и ускорить выполнение программы. Это особенно важно при разработке программ, которые требуют высокой производительности.

C++ — строгий язык, в нём уже на этапе компиляции ясно, что будет делать программа. А значит, на этапе компиляции можно сделать ряд вычислений и оптимизаций, чтобы программа работала быстрее. Это позволяет писать гибкие и быстро работающие программы. В других языках приходится платить скоростью за сложность, универсальность, красоту, понятность. В C++ сложные выразительные конструкции превращаются в простой и быстрый микрокод.

В C++ можно детально регулировать большинство операций. Например, указывать собственный аллокатор — контроллер операций с памятью. Чем больше возможностей, тем легче ошибиться и тем больше нужно знать, чтобы правильно пользоваться этими возможностями [5].

Для реализации был использован Microsoft Visual Studio как среда разработки. Visual Studio является одной из самых популярных и мощных IDE для C++, предоставляя удобные инструменты для написания, отладки и тестирования кода. Visual Studio предоставляет полный набор инструментов для работы с C++, включая поддержку последних стандартов языка, подсветку синтаксиса, авто дополнение и исправление ошибок, что значительно ускоряет процесс разработки [6].

Редактор кода в Visual Studio поддерживает подсветку синтаксиса, авто дополнение, рефакторинг и многое другое. Это позволяет разработчикам писать код быстрее и с меньшим количеством ошибок. Подсветка синтаксиса помогает легко различать различные элементы кода, такие как ключевые слова, переменные и функции. Авто дополнение ускоряет процесс написания кода, предлагая варианты завершения слов и фраз на основе контекста. Рефакторинг позволяет легко изменять структуру кода без изменения его функциональности, что упрощает поддержку и улучшение кода.

Visual Studio предоставляет мощные инструменты для отладки, включая пошаговое выполнение кода, точки останова, просмотр значений переменных и стек вызовов. Это помогает быстро находить и исправлять ошибки в коде. Пошаговое выполнение позволяет разработчикам следить за выполнением кода строка за строкой, что помогает выявить ошибки и понять логику работы программы. Точки останова позволяют приостановить выполнение кода в определенных местах, чтобы проверить значения переменных и состояние программы. Просмотр значений переменных и стек вызовов помогает понять, как данные изменяются в процессе выполнения программы и какие функции вызываются [7].

Таким образом, выбор C++ и Visual Studio был обусловлен необходимостью разработать высокопроизводительную программу с эффективным использованием памяти и простотой тестирования. Использование стандартных библиотек языка C++ позволило значительно упростить реализацию алгоритмов для работы с текстами, анализа ошибок и подсчета времени.

* 1. Описание алгоритма решения задачи

Генерация текста для тренировки осуществляется с помощью функций generate\_letters, generate\_words и generate\_syllables, данные генерируются случайным образом из подготовленных словарей, функции генерации приведены в   
приложении А. Каждый тип данных имеет свой алгоритм, который случайным образом выбирает элементы и создает тренировочный текст для пользователя. Для генерации слов и слогов используются текстовые файлы с предварительно подготовленными данными, данные словарей приведены в приложении В.

Для подсчета времени и анализа ошибок использовался метод, когда на каждый символ или слово отводится определенное время в зависимости от уровня сложности. Время на ввод символа контролируется с помощью библиотеки <chrono>. Это позволяет точно отслеживать, сколько времени пользователь тратит на ввод каждого символа. Ошибки пользователя фиксируются, и программа начисляет штрафные очки в зависимости от уровня сложности.

Для отображения статистики и обратной связи программа выводит пользователю информацию в терминал. Это помогает пользователю контролировать свой прогресс и мотивирует к улучшению результатов.

# ОПИСАНИЕ СТРУКТУРЫ ДАННЫХ И АЛГОРИТМОВ

* 1. Структура данных

Программа использует несколько ключевых структур данных для хранения и обработки информации. Все данные, с которыми работает приложение, организованы с использованием стандартных коллекций C++. Основной структурой данных является вектор строк, который используется для хранения символов, слогов и слов, необходимых для тренировки. Это динамическая структура данных, которая идеально подходит для хранения элементов переменной длины, таких как строки.

Словарь букв представляет собой вектор, содержащий все буквы латинского алфавита, которые используются для тренировки ввода.

Словарь слогов и словарь слов создаются путем считывания данных из текстовых файлов. Эти файлы содержат соответствующие элементы для тренировки, а программа случайным образом выбирает и отображает их пользователю.

Дополнительно используется переменные для статистики, такие как счетчики для правильных и неправильных символов, очков и ошибок. Эти переменные позволяют отслеживать прогресс пользователя и оценивать его результаты в реальном времени.

Кроме того, для работы с временем программа использует структуру данных, основанную на классе chrono, для отслеживания времени, отведенного на ввод каждого символа и на весь тест в целом. Время важно для оценки скорости печати и выполнения теста в рамках заданных ограничений.

* 1. Алгоритм решения задачи

Алгоритм решения задачи заключается в разработке методов, которые обеспечивают выполнение программы в минимальное время, при этом не расходуя ресурсы зря. Алгоритм — это набор последовательных шагов, которые определяют, как данные будут обрабатываться, анализироваться и модифицироваться.

Алгоритм работы программы можно разделить на несколько ключевых этапов, каждый из которых решает конкретную задачу. Cначала пользователь вводит уровень сложности тестирования, в зависимости от сложности пользователь будет получить больше или меньше времени на тест, количество получаемых успешных и ошибочных очков (см. рис. 5.1).

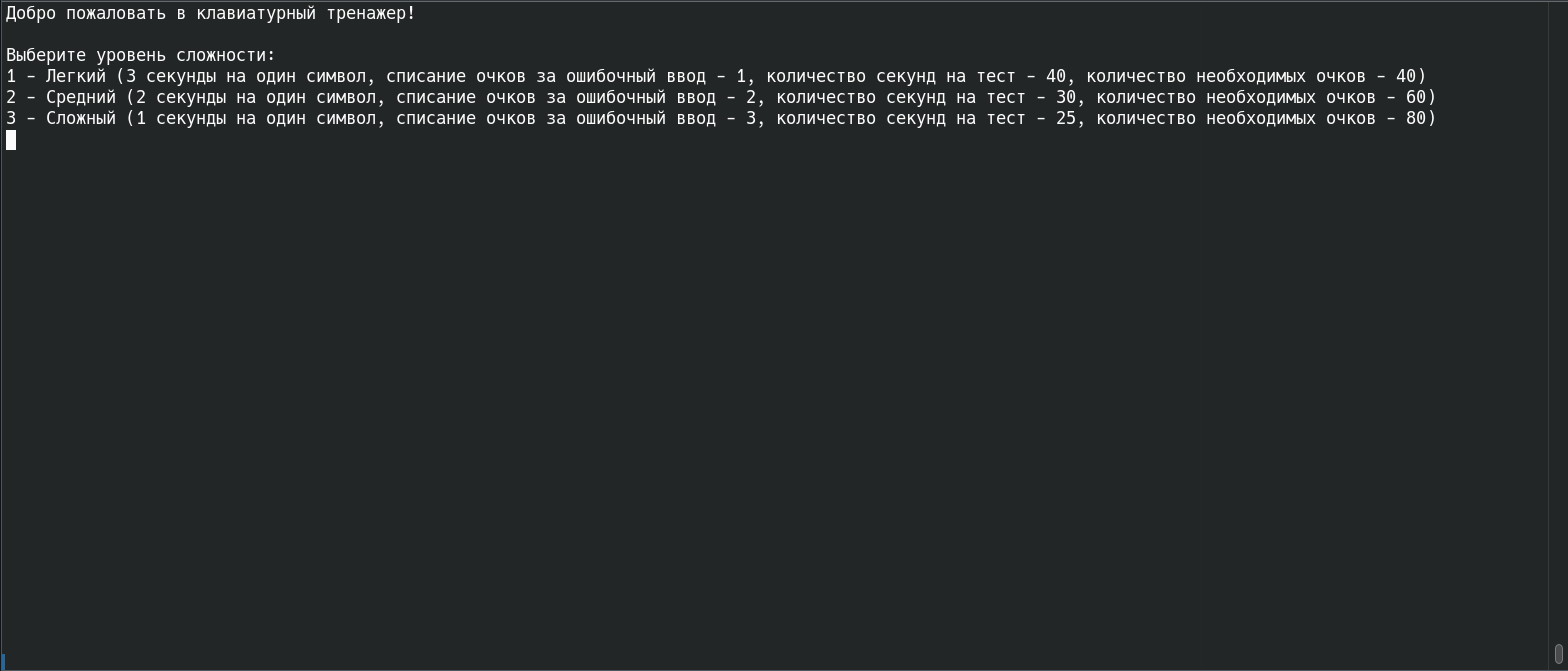


Рисунок 5.1 Выбор сложности тестирования

Программа начинает свою работу с генерации данных для тренировки. Сначала она случайным образом создает набор символов, слогов и слов, которые будет использовать для теста. Для этого используются функции generate\_letters, generate\_words и generate\_syllables, смотреть приложение А. Каждая из этих функций выполняет задачу выбора элементов из предварительно подготовленных словарей и их случайной сортировки. Эта случайность позволяет обеспечивать разнообразие заданий и избежать их повторения.

После генерации данных, программа переходит к следующему этапу — вводу символов пользователем. Каждый символ, слово или слог, отображаемые на экране, требуют от пользователя ввода. Для каждого символа отводится фиксированное количество времени, в зависимости от уровня сложности. Алгоритм проверки ввода очень прост: программа сравнивает введенный символ с правильным и увеличивает счет, если символ введен верно. В противном случае пользователь получает штрафные очки, которые зависят от уровня сложности теста. Символ, который необходимо ввести пользователю подсвечивается красным цветом (см. рис. 5.2).

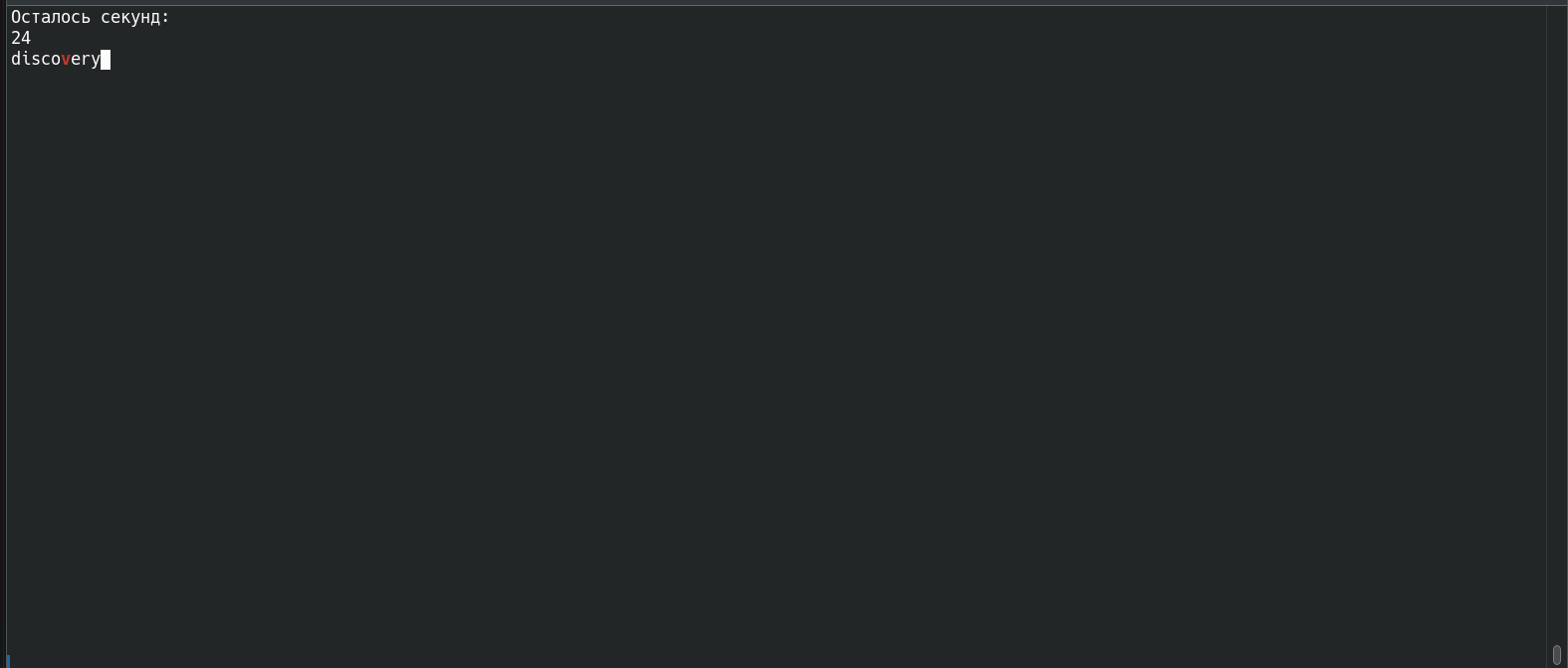


Рисунок 5.2 Пример работы тренажера

Для обеспечения корректного подсчета времени на ввод каждого символа и на весь тест, программа использует встроенные функции работы с временем.

По завершении теста программа анализирует результаты. Если количество ошибок превышает определенный порог для данного уровня сложности, тест считается не пройденным (см. рис. 5.3).

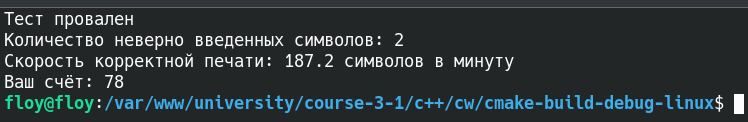


Рисунок 5.3 Пример вывода при неуспешном тестировании

Если же количество правильно введенных символов достигает или превышает необходимое количество, тест считается успешным (см. рис. 5.4).

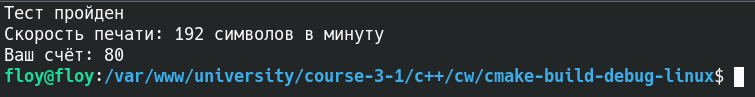


Рисунок 5.4 Пример вывода при успешном тестировании

Пользователю отображается статистика по пройденному тесту, в статистику входит счет, который вычисляется путем вычитания количества неуспешно веденных символов из успешно веденых символов, а также количество корректно введенных символов в минуту.

Оценка результатов в алгоритмах тренажеров и тестов играет важную роль, поскольку она не только позволяет измерить успех пользователя, но и создает мотивацию для улучшения. Алгоритм должен учитывать различные параметры, такие как уровень сложности и максимальное количество очков, необходимое для победы в тесте [4].

* 1. Спецификация алгоритмов

Для того чтобы алгоритм работал корректно, необходимо соблюдать четкие спецификации для каждой задачи. Спецификация включает в себя описание того, что именно алгоритм должен делать, какие данные обрабатывать и какие результаты ожидать.

Функция generate\_letters предназначена для генерации случайных букв из алфавита. Она принимает параметр length, который определяет количество символов, которые нужно сгенерировать. В результате работы функции возвращается вектор строк, содержащий случайные буквы латинского алфавита. Эта функция используется для создания тренировочных наборов для самых простых уровней сложности.

Функция generate\_words генерирует случайные слова, считывая их из внешнего текстового файла. Она принимает параметр length, который определяет количество слов, которые должны быть сгенерированы. Возвращаемое значение — это вектор строк, содержащий случайно выбранные слова из файла. Эта функция используется для тренировки более сложных навыков печати, таких как печать целых слов и фраз.

Функция generate\_syllables работает аналогично функции для слов, но она генерирует слоги, которые также считываются из файла. Слоги могут быть использованы для тренировки печати на более детальном уровне, позволяя пользователю тренироваться на коротких и более сложных фрагментах текста.

Функция typing\_test является основной в программе и отвечает за весь процесс тренировки. Она принимает вектор данных для тренировки и параметр сложности, который определяет количество времени, отведенное на каждый символ. Эта функция запускает сам процесс тестирования, отслеживает ошибки, начисляет очки и выводит результаты на экран.

Функция print\_word\_with\_highlight отображает слово, выделяя текущий символ, который должен быть введен. Это помогает пользователю фокусироваться на правильном вводе символа в реальном времени. Она принимает два параметра: строку с текущим словом и индекс символа, который нужно подсветить.

Функция set\_input\_mode отвечает за включение или отключение режима ввода, при котором символы, введенные пользователем, не отображаются на экране. Это может быть полезно для тренировки, когда нужно сосредоточиться исключительно на самом вводе, не отвлекаясь на видимые символы.

Четкие спецификации алгоритмов являются основой для корректной работы программы. Правильная разработка алгоритмов требует внимания к деталям и способности адаптировать решения под изменяющиеся требования задачи.

# ТЕСТИРОВАНИЕ И РЕЗУЛЬТАТЫ

* 1. Подходы к тестированию программы

Тестирование программы является неотъемлемой частью разработки, поскольку позволяет выявить ошибки, улучшить качество кода и убедиться, что программа работает так, как ожидается. В случае с клавиатурным тренажером, тестирование должно охватывать несколько аспектов: функциональность, производительность, а также правильность обработки ввода и подсчета результатов:

* функциональное тестирование: это основной вид тестирования, направленный на проверку правильности работы программы с точки зрения ее функциональности;
* тестирование на разных уровнях сложности: для тренажера предусмотрено три уровня сложности, где время на ввод символа, количество ошибок и требования к количеству баллов варьируются. Важно было проверить, как программа себя ведет при разных уровнях сложности;
* тестирование на корректность времени: одним из важнейших аспектов программы является точность отслеживания времени на ввод символов и на весь тест в целом. Были проведены тесты, которые проверяли, точно ли программа отслеживает время и правильно ли она реагирует на превышение лимита времени.

Подходы к тестированию программы включают функциональное тестирование, тестирование на корректность времени и проверки для всех уровней сложности. Эти тесты позволяют удостовериться, что программа работает корректно в различных условиях.

* 1. Результаты тестирования

Тестирование программы показало, что она успешно выполняет все функциональные требования. Все функции, включая генерацию случайных символов, слогов и слов, а также проверку ввода и подсчет ошибок, работают корректно. Программа правильно отслеживает время, отведенное на каждый символ и на весь тест, и корректно оценивает результаты.

Уровни сложности были протестированы и соответствуют заявленным параметрам: время на ввод символа, количество ошибок и минимальные баллы для прохождения теста варьируются в зависимости от сложности. Время, отведенное на каждый символ, строго соответствует выбранному уровню сложности.

Таким образом, тестирование подтвердило, что программа работает стабильно, правильно отслеживает время и ошибки, а также предоставляет пользователю понятную обратную связь.

# ЗАКЛЮЧЕНИЕ

В процессе выполнения курсовой работы был успешно разработан клавиатурный тренажер, ориентированный на улучшение навыков быстрой и точной печати. Основная задача заключалась в разработке программы, которая обеспечивала бы комплексный подход к обучению, включая оценку скорости печати, подсчет ошибок и предоставление статистики. Все поставленные цели были успешно реализованы.

Программа имеет функциональный интерфейс, который позволяет пользователю выбрать уровень сложности. В зависимости от выбранной сложности изменяются параметры тренировки, такие как время на ввод символа и количество баллов, необходимых для прохождения теста. Таким образом, интерфейс полностью соответствует требованию персонализации тренировки.

Для решения задачи получения различных источников данных для тренировки была реализована система случайной генерации символов, слогов и слов, которые выводятся пользователю в зависимости от выбранного уровня сложности. Алгоритмы для создания и вывода данных были успешно реализованы, и программа обеспечивает разнообразие тренировок, адаптированных к индивидуальным потребностям пользователя.

Кроме того, алгоритмы подсчета скорости печати и ошибок работают корректно. Программа отслеживает время, отведенное на ввод каждого символа, и точно вычисляет количество правильных вводов, а также подсчитывает ошибки. Это позволяет пользователю получать объективную оценку своих результатов и эффективно работать над улучшением своих навыков.

Тестирование программы подтвердило её работоспособность и удобство в использовании. Все функциональные элементы работают согласно заявленным требованиям, и программа предоставляет точную информацию о времени, ошибках и скорости печати. В будущем возможно расширение функционала, улучшение интерфейса и добавление дополнительных настроек для более точной адаптации тренировки под нужды пользователя.

Таким образом, все цели и задачи работы были выполнены, и разработанная программа является эффективным инструментом для тренировки навыков слепой печати.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Розенберг, А. М. Компьютерная грамотность: учебное пособие / А. М. Розенберг. – 2-е изд., перераб. и доп. – Москва, 2012. – 352 с. – (Библиотека информационных технологий). – ISBN 978-5-9727-1085-6 (дата обращения: 21.02.2025).
2. Филиппов, Ю. В. Основы компьютерной грамотности: Учебное пособие / Ю. В. Филиппов, В. В. Груздев. – М.: Высшая школа, 2018. – 256 с. – ISBN 978-5-06-015307-4 (дата обращения: 21.02.2025).
3. Метод слепой печати. Foxford: <https://externat.foxford.ru/polezno-znat/metod-slepoy-pechati> (дата обращения: 25.02.2025).
4. Станкевич, В. И. Программирование на C++: Теория и практика / В. И. Станкевич. – М.: ДМК Пресс, 2017. – 512 с. – ISBN 978-5-94074-375-7 (дата обращения: 25.02.2025).
5. Гид по языку программирования: <https://practicum.yandex.ru/blog/yazyk-c-plus-dlya-chego-nuzhen/> (дата обращения: 25.02.2025).
6. Тимофеев, В. П. Практическое руководство по Visual Studio для разработчиков на C++ / В. П. Тимофеев. – Санкт-Петербург: Питер, 2015. – 352 с. – ISBN 978-5-4461-0875-1 (дата обращения: 25.02.2025).
7. Обзор Visual Studio: возможности и особенности <https://sky.pro/wiki/javascript/obzor-visual-studio-vozmozhnosti-i-osobennosti/> (дата обращения: 26.02.2025)

# Приложение А

Текст программы

#include <iostream>

#include <vector>

#include <string>

#include <chrono>

#include <unistd.h>

#include <clocale>

#include <random>

#include <filesystem>

#include <fstream>

#ifdef \_\_linux\_\_

#include <termios.h>

#endif

using namespace std;

using namespace chrono;

const vector<string> LETTERS\_DICTIONARY\_EN = {"a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z"};

vector<string> shuffle\_vector(vector<string> data) {

std::random\_device rd;

std::mt19937 g(rd());

std::shuffle(data.begin(), data.end(), g);

return data;

}

void set\_input\_mode(const bool enable) {

#ifdef \_\_linux\_\_

static termios oldt, newt;

if (enable) {

tcgetattr(STDIN\_FILENO, &oldt);

newt = oldt;

// Выключаем буферизацию и отображение символов

newt.c\_lflag &= ~(ICANON | ECHO);

tcsetattr(STDIN\_FILENO, TCSANOW, &newt);

} else {

tcsetattr(STDIN\_FILENO, TCSANOW, &oldt);

}

#endif

}

void clear\_terminal() {

#ifdef \_\_linux\_\_

std::cout << "\x1B[2J\x1B[H";

#endif

}

vector<string> generate\_letters(const int length) {

vector<string> result;

vector<string> dictionary = shuffle\_vector(LETTERS\_DICTIONARY\_EN);

result.reserve(length);

for (int i = 0; i < length; ++i) {

result.push\_back(dictionary[i]);

}

return result;

}

vector<string> generate\_words(const int length) {

ifstream input\_file\_stream("library/words.en.dic");

if (!input\_file\_stream.is\_open()) {

throw std::invalid\_argument("Словарь с словами не найден");

}

vector<string> words;

while (!input\_file\_stream.eof()) {

string word;

input\_file\_stream >> word;

if (!word.empty()) {

words.push\_back(word);

}

}

words = shuffle\_vector(words);

words.resize(length);

return words;

}

vector<string> generate\_syllables(const int length) {

ifstream input\_file\_stream("library/syllables.en.dic");

if (!input\_file\_stream.is\_open()) {

throw std::invalid\_argument("Словарь с слогами не найден");

}

vector<vector<string>> words\_syllables;

vector<string> word\_syllables;

while (!input\_file\_stream.eof()) {

string syllable;

input\_file\_stream >> syllable;

if (syllable != "-") {

word\_syllables.push\_back(syllable);

}

else {

words\_syllables.push\_back(word\_syllables);

word\_syllables.clear();

}

}

std::random\_device rd;

std::mt19937 g(rd());

std::shuffle(words\_syllables.begin(), words\_syllables.end(), g);

words\_syllables.resize(length);

vector<string> syllables;

for (vector<string>& word\_syllables : words\_syllables) {

for (string& syllable : word\_syllables) {

syllables.push\_back(syllable);

}

}

return syllables;

}

void print\_word\_with\_highlight(const string& word, int pos) {

// Ставим курсор в начало строки

cout << "\r";

for (int i = 0; i < word.size(); i++) {

if (i == pos) {

#ifdef \_WIN32

cout << word[i];

#else

cout << "\033[1;31m" << word[i] << "\033[0m";

#endif

}

else {

cout << word[i];

}

}

cout.flush();

}

vector<string> get\_test\_data()

{

vector<string> data;

vector<string> letters = generate\_letters(25);

vector<string> syllables = generate\_syllables(25);

vector<string> words = generate\_words(25);

data.insert(data.end(), letters.begin(), letters.end());

data.insert(data.end(), syllables.begin(), syllables.end());

data.insert(data.end(), words.begin(), words.end());

data = shuffle\_vector(data);

return data;

}

int get\_points\_for\_win(const int difficulty) {

if (difficulty == 1) {

return 40;

}

if (difficulty == 2) {

return 60;

}

if (difficulty == 3) {

return 80;

}

throw runtime\_error("Сложность не найдена");

}

int get\_time\_limit\_on\_symbol(const int difficulty) {

if (difficulty == 1) {

return 3;

}

if (difficulty == 2) {

return 2;

}

if (difficulty == 3) {

return 1;

}

throw runtime\_error("Сложность не найдена");

}

int get\_time\_for\_test(const int difficulty) {

if (difficulty == 1) {

return 5;

}

if (difficulty == 2) {

return 30;

}

if (difficulty == 3) {

return 25;

}

throw runtime\_error("Сложность не найдена");

}

bool is\_still\_ongoing(auto start\_time, auto symbol\_start\_time, int limit\_on\_symbol, int limit\_for\_test) {

if (duration\_cast<seconds>(steady\_clock::now() - symbol\_start\_time).count() > limit\_on\_symbol) {

return false;

}

if (duration\_cast<seconds>(steady\_clock::now() - start\_time).count() > limit\_for\_test) {

return false;

}

return true;

}

void print\_time\_left(auto start\_time, int limit\_for\_test) {

int seconds\_left = limit\_for\_test - duration\_cast<seconds>(steady\_clock::now() - start\_time).count();

if (seconds\_left > 0) {

#ifdef \_WIN32

cout << "Time left:" << endl;

#else

cout << "Осталось секунд:" << endl;

#endif

cout << seconds\_left << endl;

}

else {

#ifdef \_WIN32

cout << "Time end" << endl;

#else

cout << "Время окончено" << endl;

#endif

}

}

void typing\_test(const vector<string>& data, int difficulty) {

int score = 0;

int decrease\_score = 0;

int target\_index = 0;

int limit\_on\_symbol = get\_time\_limit\_on\_symbol(difficulty);

int limit\_for\_test = get\_time\_for\_test(difficulty);

auto symbol\_start\_time = steady\_clock::now();

auto start\_time = steady\_clock::now();

while (duration\_cast<seconds>(steady\_clock::now() - start\_time).count() < limit\_for\_test) {

const string& target = data[target\_index];

int pos = 0;

while (pos < target.size()) {

// Проверяем время до ввода символа

if (!is\_still\_ongoing(start\_time, symbol\_start\_time, limit\_on\_symbol, limit\_for\_test)) {

break;

}

print\_time\_left(start\_time, limit\_for\_test);

print\_word\_with\_highlight(target, pos);

// Ввод символа без энтера

char cin\_char;

read(STDIN\_FILENO, &cin\_char, 1);

// Проверяем время после ввода символа

if (!is\_still\_ongoing(start\_time, symbol\_start\_time, limit\_on\_symbol, limit\_for\_test)) {

break;

}

if (cin\_char == target[pos]) {

symbol\_start\_time = steady\_clock::now();

pos++;

score++;

} else {

decrease\_score += difficulty;

}

clear\_terminal();

}

if (duration\_cast<seconds>(steady\_clock::now() - symbol\_start\_time).count() > limit\_on\_symbol) {

break;

}

target\_index++;

}

cout << endl;

clear\_terminal();

if (score - decrease\_score < get\_points\_for\_win(difficulty)) {

#ifdef \_WIN32

cout << "Test failed" << endl;

cout << "Number of incorrectly entered characters: " << decrease\_score / difficulty << endl;

#else

cout << "Тест провален" << endl;

cout << "Количество неверно введенных символов: " << decrease\_score / difficulty << endl;

#endif

}

else {

#ifdef \_WIN32

cout << "Test pass" << endl;

#else

cout << "Тест пройден" << endl;

#endif

}

float actual\_score = static\_cast<float>(score - decrease\_score);

if (actual\_score < 0) {

actual\_score = 0.0;

}

#ifdef \_WIN32

cout << "Your score: " << score - decrease\_score << endl;

cout << "Correct type speed: " actual\_score / static\_cast<float>(limit\_for\_test) \* 60.0 << " characters per minute" << endl;

#else

cout << "Скорость корректной печати: " << actual\_score / static\_cast<float>(limit\_for\_test) \* 60.0 << " символов в минуту" << endl;

cout << "Ваш счёт: " << score - decrease\_score << endl;

#endif

}

int main() {

setlocale(LC\_ALL, "Russian");

setlocale(LC\_CTYPE, "Russian");

clear\_terminal();

#ifdef \_WIN32

cout << "Welcome to the keyboard trainer!" << endl << endl;

cout << "Select difficulty level:" << endl;

cout << "1 - Easy (3 seconds per character, points deduction for incorrect input - 1, number of seconds for test - 40, number of points required - 40)" << endl;

cout << "2 - Average (2 seconds per character, points deduction for incorrect input - 2, number of seconds for test - 30, number of points required - 60)" << endl;

cout << "3 - Complex (1 second per character, points deduction for incorrect input - 3, number of seconds for test - 25, number of points required - 80)" << endl;

#else

cout << "Добро пожаловать в клавиатурный тренажер!" << endl << endl;

cout << "Выберите уровень сложности:" << endl;

cout << "1 - Легкий (3 секунды на один символ, списание очков за ошибочный ввод - 1, количество секунд на тест - 40, количество необходимых очков - 40)" << endl;

cout << "2 - Средний (2 секунды на один символ, списание очков за ошибочный ввод - 2, количество секунд на тест - 30, количество необходимых очков - 60)" << endl;

cout << "3 - Сложный (1 секунды на один символ, списание очков за ошибочный ввод - 3, количество секунд на тест - 25, количество необходимых очков - 80)" << endl;

#endif

int difficulty = 1;

cin >> difficulty;

clear\_terminal();

set\_input\_mode(true);

typing\_test(get\_test\_data(), difficulty);

set\_input\_mode(false);

#ifdef \_WIN32

getchar();

system("pause");

#endif

return 0;

}

# Приложение Б

Блок схема приложения



Блок схема Б.1 Точка входа приложения



Блок схема Б.2 Метод очистки терминала



Блок схема Б.3 Метод отключения буферизации символов



Блок схема Б.4 Метод проведения тестирования



Блок схема Б.5 Метод вывода оставшегося времени



Блок схема Б.6 Метод вывода текущего символа с подсветкой



Блок схема Б.7 Метод генерации слов



Блок схема Б.8 Метод генерации букв



Блок схема Б.9 Метод получения тестовых данных