

**PROJECT REPORT**

FIRST YEAR

COMPUTER SCIENCE

BATCH: 2024 - 28

PROGRAMMING FUNDAMENTALS (CT-175)

**TYPE DASH**

**(TYPING SPEED & ACCURACY TESTER)**

**NAME:**

SMOOTH OPERATORS

**MEMBERS:**

* SYED HUZAIFA BUKHARI (CT-24029)
* ZORAIZ AFRIDI (CT-24042)
* IZHAN KHAN (CT-24047)
* HAMMAD AHMED SIDDIQUI (CT-24022)

**DISCIPLINE:** FSCS (COMPUTER SCIENCE)

**SECTION:** A

**SEMESTER:** FALL (2024-25)

**PROJECT DESCRIPTION:**

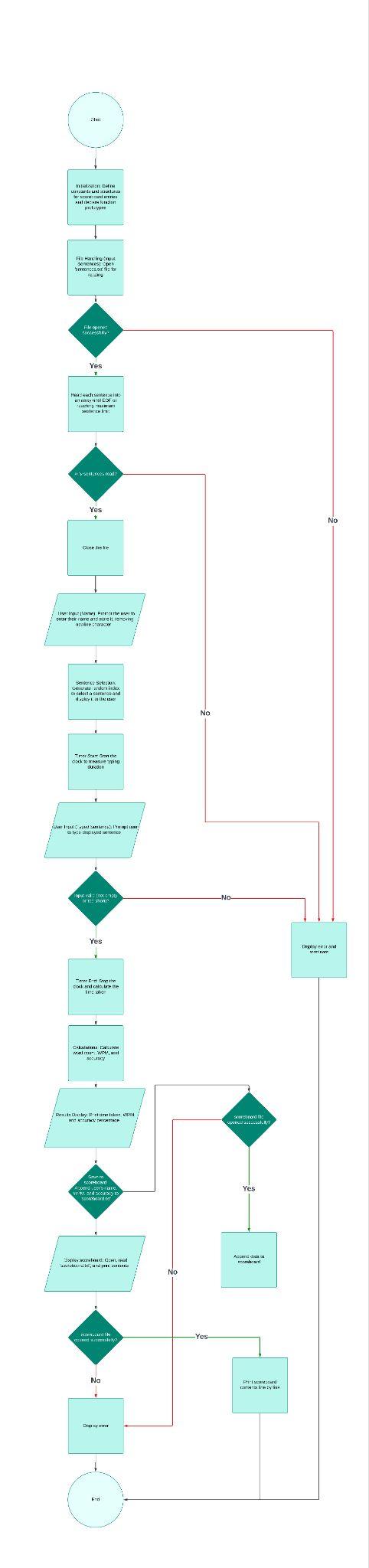
**Type Dash** is a **Typing Speed and Accuracy Tester** designed to assess and improve a user's typing proficiency. It randomly selects a sentence from a file, measures how quickly and accurately the user can type the sentence, and records the results in a scoreboard.

The program incorporates fundamental programming concepts such as file handling, string manipulation, and arithmetic calculations, providing an engaging way for users to interact with these concepts. It also implements randomization and persistent data storage to enhance usability and functionality.

**FEATURES DESCRIPTION:**

1. **Sentence Selection**
   * Randomly selects a sentence from a file (sentences.txt) to ensure variety in the typing challenges.
   * Handles up to 100 sentences, each with a maximum length of 200 characters.
2. **User Input and Timing**
   * Captures the user’s name and typed sentence.
   * Measures the time taken to type the sentence using system clock functions.
3. **Performance Metrics**
   * Words Per Minute (WPM): Calculates typing speed based on the time taken and the number of words in the sentence.
   * Accuracy: Compares the typed sentence with the original sentence and calculates a percentage score based on the number of matching characters.
4. **Scoreboard Management**
   * Records the user’s name, WPM, and accuracy into a text file (scoreboard.txt).
   * Displays the scoreboard with all previous entries, sorted by the order of addition.
5. **Error Handling**
   * Handles cases where the sentence file is missing or empty.
   * Ensures valid input by checking the length and content of the user’s typed text.
   * Provides user-friendly error messages for common issues.
6. **Data Persistence**
   * Saves the scoreboard data in a file for later reference, ensuring scores are not lost between program runs.
7. **Dynamic Feedback**
   * Displays detailed results, including time taken, WPM, and accuracy, immediately after the typing test.

**PROJECT FLOWCHART:**

****

**DATA DESCRIPTION:**

#### **Global Constants**

* MAX\_LENGTH (int): Sets the maximum length for the typed text input (1000 characters).
* MAX\_SENTENCES (int): Limits the number of sentences that can be loaded from the file (100 sentences).
* MAX\_SENTENCE\_LENGTH (int): Sets the maximum length of a single sentence (200 characters).
* MAX\_SCOREBOARD\_ENTRIES (int): Limits the number of scoreboard entries (100 entries).
* SCOREBOARD\_FILE (string): Name of the file storing the scoreboard ("scoreboard.txt")

##### **Main Function Variables**

* char sentences[MAX\_SENTENCES][MAX\_SENTENCE\_LENGTH]:
  + Stores the array of sentences read from the sentences.txt file.
  + Purpose: Provides a pool of sentences for the typing test.
* int sentence\_count:
  + Tracks the number of sentences successfully loaded from the file.
  + Purpose: Ensures sentences do not exceed the set limit.
* FILE \*file:
  + Pointer to a file object used to open and read the sentences file.
  + Purpose: Handles file operations.
* char name[50]:
  + Stores the user's name for the scoreboard entry.
  + Purpose: Identifies the user in the scoreboard.
* int random\_index:
  + Stores a randomly generated index to select a sentence from sentences[].
  + Purpose: Ensures randomness in the sentence selection process.
* char \*selected\_sentence:
  + Points to the randomly selected sentence from the sentences[] array.
  + Purpose: Supplies the target sentence for the typing test.
* char typed\_text[MAX\_LENGTH]:
  + Stores the user's input text during the typing test.
  + Purpose: Captures the text for comparison with the target sentence.
* clock\_t start\_time, end\_time:
  + Holds the time values at the start and end of the typing test.
  + Purpose: Measures how long the user takes to type the sentence.
* float time\_taken:
  + Calculates the duration of the typing test in seconds.
  + Purpose: Used in calculating words per minute (WPM).
* int word\_count:
  + Stores the count of words in the selected sentence.
  + Purpose: Used to compute the WPM.
* float wpm:
  + Holds the calculated words per minute.
  + Purpose: Measures typing speed.
* float accuracy:
  + Stores the calculated typing accuracy as a percentage.
  + Purpose: Measures how accurately the user typed the sentence.

**Function-Specific Variables**

1. int count\_words(const char \*text)
   * const char \*text: Input text to count words in.
   * int word\_count: Tracks the total number of words.
   * int in\_word: Flags whether the current character is part of a word.
2. float calculate\_wpm(float time\_taken, int word\_count)
   * float time\_taken: Time taken to type the sentence.
   * int word\_count: Total number of words in the sentence.
   * Returns: Words per minute as a floating-point value.
3. float calculate\_accuracy(const char \*typed\_text, const char \*original\_text)
   * const char \*typed\_text: User-typed sentence.
   * const char \*original\_text: Original sentence for comparison.
   * int correct\_chars: Number of characters typed correctly.
   * int total\_chars: Length of the original sentence.
   * Returns: Accuracy as a floating-point percentage.
4. void save\_scoreboard(const char \*name, float wpm, float accuracy)
   * const char \*name: Name of the user.
   * float wpm: Calculated words per minute.
   * float accuracy: Calculated accuracy.
5. void display\_scoreboard()
   * FILE \*file: File pointer to read the scoreboard data.
   * char line[100]: Temporary buffer to hold each line from the file.

**FUNCTION DESCRIPTION:**

1. **int main()**
   * Manages overall flow: reads sentences from a file, prompts the user for input, calculates results, and updates the leaderboard.
2. **int count\_words(const char \*text)**
   * Counts the number of words in a given string by checking spaces and word boundaries.
3. **float calculate\_wpm(float time\_taken, int word\_count)**
   * Calculates words per minute based on the time taken and the number of words in the sentence.
4. **float calculate\_accuracy(const char \*typed\_text, const char \*original\_text)**
   * Compares the typed text with the original sentence and calculates accuracy as a percentage.
5. **void save\_scoreboard(const char \*name, float wpm, float accuracy)**
   * Appends the user's name, WPM, and accuracy to the scoreboard file.
6. **void display\_scoreboard()**
   * Reads and displays the scoreboard from the file.

**SOURCE CODE:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#include <ctype.h>

#define MAX\_LENGTH 1000

#define MAX\_SENTENCES 100

#define MAX\_SENTENCE\_LENGTH 200

#define MAX\_SCOREBOARD\_ENTRIES 100

#define SCOREBOARD\_FILE "scoreboard.txt"

// Structure for scoreboard

typedef struct {

char name[50];

float wpm;

float accuracy;

} scoreboardEntry;

// Function prototypes

int count\_words(const char \*text);

float calculate\_wpm(float time\_taken, int word\_count);

float calculate\_accuracy(const char \*typed\_text, const char \*original\_text);

void save\_scoreboard(const char \*name, float wpm, float accuracy);

void display\_scoreboard();

int main() {

char sentences[MAX\_SENTENCES][MAX\_SENTENCE\_LENGTH];

int sentence\_count = 0;

FILE \*file;

file = fopen("sentences.txt", "r");

if (file == NULL) {

printf("Error: Could not open sentences.txt.\n");

return 1;

}

while (fgets(sentences[sentence\_count], MAX\_SENTENCE\_LENGTH, file) != NULL) {

int new\_line = strcspn(sentences[sentence\_count], "\n");

sentences[sentence\_count][new\_line] = '\0';

sentence\_count++;

if (sentence\_count >= MAX\_SENTENCES)

break;

}

fclose(file);

if (sentence\_count == 0) {

printf("Error: No sentences found in sentences.txt.\n");

return 1;

}

char name[50];

printf("Enter your name for the scoreboard: ");

fgets(name, sizeof(name), stdin);

name[strcspn(name, "\n")] = '\0';

srand(time(NULL));

int random\_index = rand() % sentence\_count;

char \*selected\_sentence = sentences[random\_index];

printf("Type the following sentence:\n\n%s\n\n", selected\_sentence);

clock\_t start\_time = clock();

char typed\_text[MAX\_LENGTH];

printf("Start typing: ");

fgets(typed\_text, MAX\_LENGTH, stdin);

typed\_text[strcspn(typed\_text, "\n")] = '\0';

if (strlen(typed\_text) == 0) {

printf("No input detected. Please try again.\n");

return 1;

}

if (strlen(typed\_text) < 5) {

printf("Input is too short. Please type a longer sentence.\n");

return 1;

}

clock\_t end\_time = clock();

float time\_taken = (float)(end\_time - start\_time) / CLOCKS\_PER\_SEC;

int word\_count = count\_words(selected\_sentence);

float wpm = calculate\_wpm(time\_taken, word\_count);

float accuracy = calculate\_accuracy(typed\_text, selected\_sentence);

printf("\nResults:\n");

printf("Time taken: %.2f seconds\n", time\_taken);

printf("Words per minute: %.2f WPM\n", wpm);

printf("Accuracy: %.2f%%\n", accuracy);

save\_scoreboard(name, wpm, accuracy);

display\_scoreboard();

return 0;

}

int count\_words(const char \*text) {

int word\_count = 0;

int in\_word = 0;

for (int i = 0; text[i] != '\0'; i++) {

if (isspace(text[i])) {

in\_word = 0;

} else if (in\_word == 0) {

in\_word = 1;

word\_count++;

}

}

return word\_count;

}

float calculate\_wpm(float time\_taken, int word\_count) {

return (word\_count / time\_taken) \* 60;

}

float calculate\_accuracy(const char \*typed\_text, const char \*original\_text) {

int correct\_chars = 0;

int total\_chars = strlen(original\_text);

for (int i = 0; i < total\_chars; i++) {

if (typed\_text[i] == original\_text[i]) {

correct\_chars++;

}

}

return (float) correct\_chars / total\_chars \* 100;

}

void save\_scoreboard(const char \*name, float wpm, float accuracy) {

FILE \*file = fopen(SCOREBOARD\_FILE, "a");

if (file == NULL) {

printf("Error: Could not open Scoreboard file.\n");

return;

}

fprintf(file, "%s, %.2f wpm, %.2f%%\n", name, wpm, accuracy);

fclose(file);

}

void display\_scoreboard() {

FILE \*file = fopen(SCOREBOARD\_FILE, "r");

if (file == NULL) {

printf("Error: Could not open scoreboard file.\n");

return;

}

printf("\nScoreboard:\n");

char line[100];

while (fgets(line, sizeof(line), file) != NULL) {

printf("%s", line);

}

fclose(file);

}

**MEMBERS CONTRIBUTION:**

Izhan Khan (CT-24047):

* Responsible for the **main function** and overall program flow.
* Implemented the logic for loading sentences from sentences.txt and developed the random sentence selection for typing practice.
* Sections of code:
  + main() function up to clock\_t start\_time = clock();

Hammad Ahmed Siddiqui (CT-24022):

* Focused on calculating Words Per Minute (WPM) and accuracy.
* Sections of code:
  + float calculate\_wpm(float time\_taken, int word\_count);
  + float calculate\_accuracy(const char \*typed\_text, const char \*original\_text);.

Zoraiz Afridi (CT-24042):

* Handled edge cases regarding user input and file handling (e.g., short input, empty file)
* Focused on handling the leaderboard functionality.
* Sections of code:
  + void save\_leaderboard(const char \*name, float wpm, float accuracy);
  + void display\_leaderboard();.

Syed Huzaifa Bukhari (CT-24029):

* Performed repetitive test runs of the entire program to identify issues and improve performance by handling edge cases across functions.
* Implemented and improved the count\_words function, refining the in\_word logic for better word-count accuracy.
* Validated outputs for all major functions to ensure they generated the right results (e.g., accuracy and leaderboard functions).
* Sections of code:
  + int count\_words(const char \*text);.
  + Edge-case handling and refinements in main(), calculate\_accuracy, and display\_leaderboard.