Politechnika Śląska

Wydział Matematyki Stosowanej

Kierunek Informatyka

Gliwice, 01.10.2020

Programowanie I

**Projekt zaliczeniowy**

**"Nauka szybkiego pisania"**

**Marcin Hajdecki gr. lab. 3**

**1. Opis projektu.**

Program pozwala doskonalić umiejętność szybkiego pisania na klawiaturze.

**2. Wymagania**

Lista funkcjonalności:

* generuje tekst składający się ze słów oddzielonych spacjami
* wyświetla, w jakich miejscach użytkownik popełnił błędy oraz ich liczbę
* oblicza i wyświetla prędkość pisania
* oblicza i wyświetla uzyskany wynik
* wyświetla najlepszy wynik
* umożliwia kontynuację ćwiczenia

**3. Przebieg realizacji**

Program składa się z 4 plików:

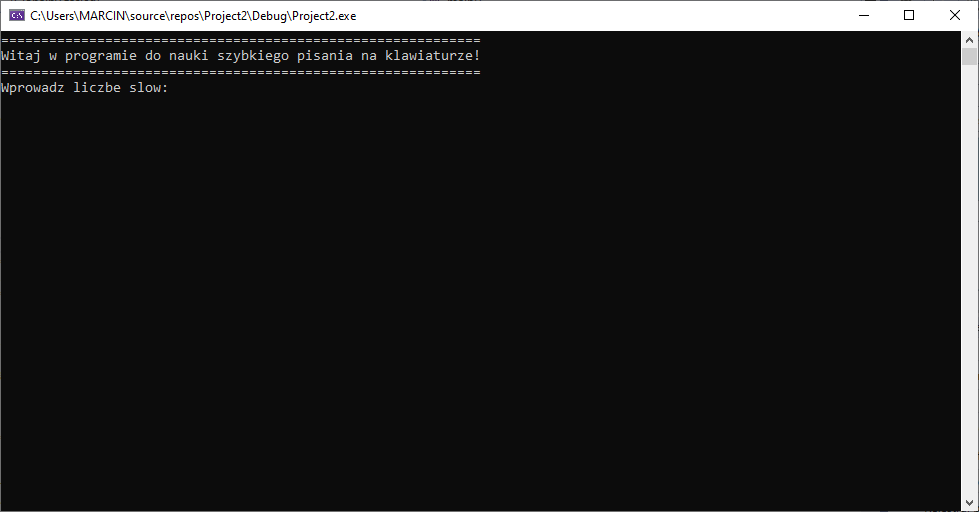
1. main.cpp – uruchamia pierwszą lekcję, wyświetla użytkownikowi informacje zwrotne i umożliwia uruchomienie następnej lekcji
2. generateText.cpp – generuje tekst do przepisania przez użytkownika
3. getUserInput.cpp – pobiera tekst od użytkownika i oblicza prędkość pisania tekstu
4. correctErrors.cpp – implementuje mechanizm liczenia błędów i wyświetlania, gdzie użytkownik popełnił błąd

Wykorzystane biblioteki:

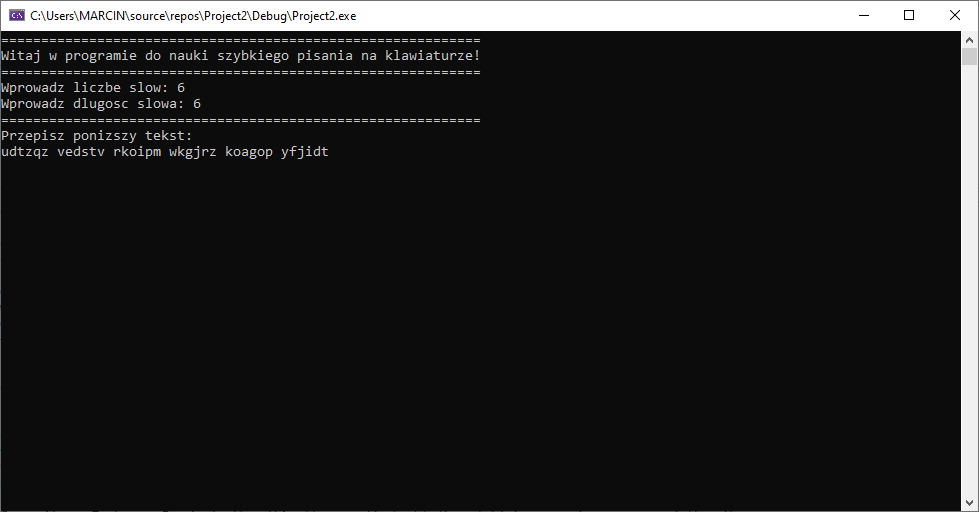
* iostream i chrono należą do biblioteki standardowej
* windows.h – pozwala korzystać z funkcji WinAPI , w tym zmieniać kolor tekstu w konsoli (tworzę obiekt typu HANDLE za pomocą GetStdHandle(), aby korzystać z funkcji SetConsoleTextAttribute())

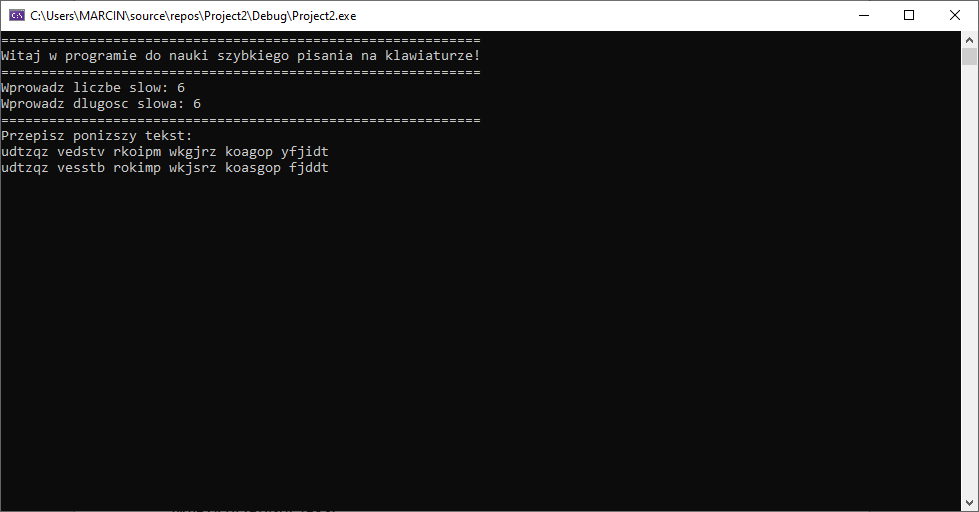
**4. Instrukcja użytkownika**

Program po uruchomieniu wyświetla ekran powitalny i prosi o podanie parametrów: długości słów w tekście i ilości słów w tekście. Wartości te obowiązują przez cały czas działania programu.

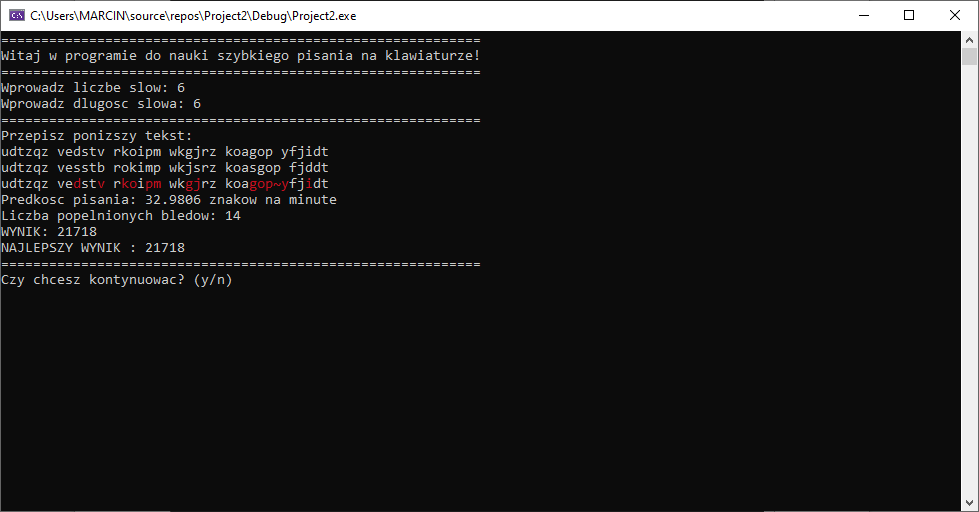


Po wpisaniu wartości generowany jest tekst do przepisania

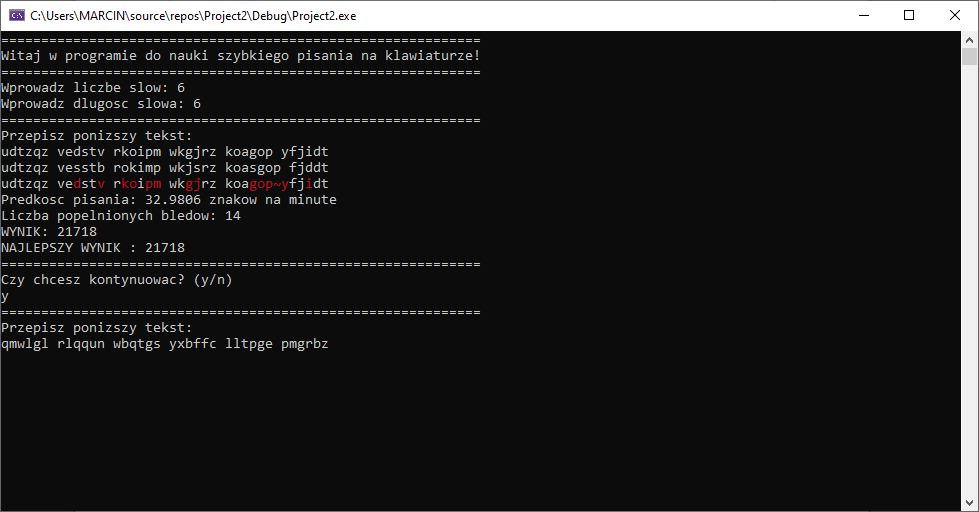


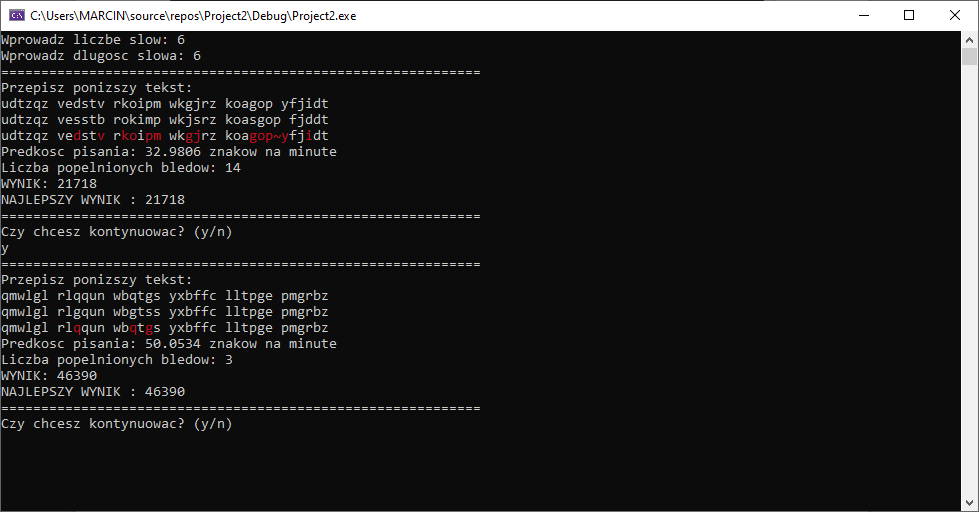


Po przepisaniu tekstu i naciśnięciu ENTER pojawiają się informacje zwrotne, kolejno: tekst z błędami zaznaczonymi na czerwono, prędkość pisania, liczba popełnionych błędów. Program wyświetla wynik i najlepszy wynik uzyskany w ciągu działania programu. Wynik jest tym wyższy, im większa prędkość pisania i tym niższy, im więcej błędów popełniono proporcjonalnie do ilości znaków w tekście. Przy połowie znaków wpisanych błędnie wynik będzie o połowę niższy niż przy wpisaniu wszystkich znaków poprawnie z tą samą prędkością. Przy wszystkich znakach wpisanych błędnie wynik zawsze wynosi 0.

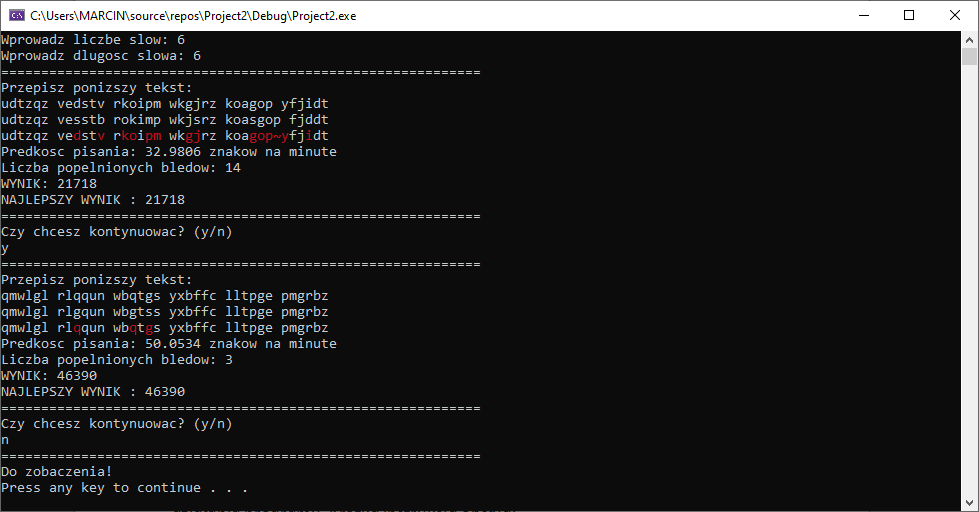


Program pyta nas, czy kontynuować. Możemy uruchomić następną lekcję wpisując „y”. Wtedy program wygeneruje kolejny tekst do przepisania i powtarza poprzednio wyjaśnione instrukcje.

Poniżej widać, jak program zarejestrował nowy najlepszy wynik.



Możemy także zakończyć działanie programu wpisując „n”



**5. Podsumowanie i wnioski.**

Największą napotkaną trudnością okazało się wprowadzone przez użytkownika dane pozostające w buforze strumienia wejściowego, które wywoływały lawinowe wykonanie się kolejnych instrukcji. Problem udało się wyeliminować poprzez wywołanie w odpowiednich miejscach funkcji cin.ignore(). Trochę trudności sprawiła implementacja funkcji generująca tekst składający się z słów oddzielonych spacjami. Poza tym tworzenie programu przebiegało raczej płynnie.

W następnej wersji programu można by dodać następujące funkcjonalności:

* wyświetlanie użytkownikowi czasu wpisywania tekstu
* generowanie tekstu z gotowych słów znajdujących się w osobnym pliku „słownika” (tekst wyglądałby bardziej naturalnie)