**Piotr Małecki**

**Michał Daniluk**

**Zaawansowane Programowanie w C++**

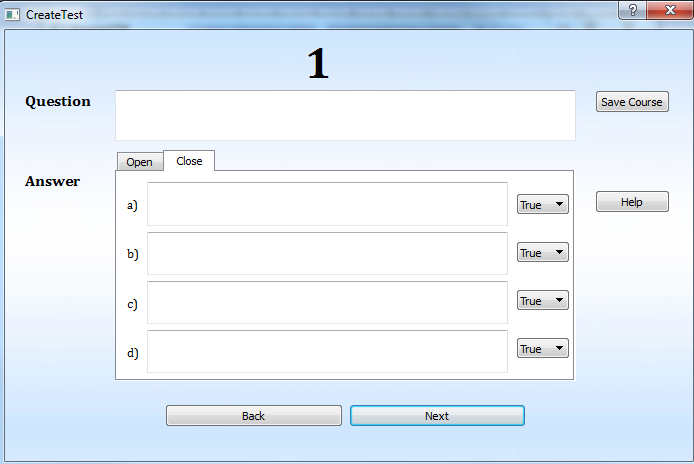
**Dokumentacja Końcowa**

**Temat projektu:** Program wspomagający proces uczenia się w stylu pytanie-odpowiedź.

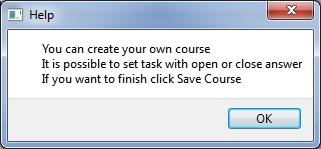
1. **Dokumentacja funkcjonalna**

Program wspomaga proces uczenia się poprzez odpowiednie dobieranie dat powtórzenia odpowiednich pytań. Założeniem programu jest regularne z niego korzystanie, w przeciwnym wypadku pytania nieodrobione kumulują się. Program wylicza datę powtórki każdego z pytań według algorytmu SuperMemo 2, w którym następna data powtórki jest wyliczana na podstawie współczynnika Easiness Factor (domyślnie ustawiony na 2,5), ilości powtórek danego pytania oraz oceny użytkownika na dane pytanie w skali od 0 do 5. Zgodnie z tym algorytmem po pierwszym wyświetleniu danego pytania pojawi się ono ponownie po jednym lub 6 dniach.

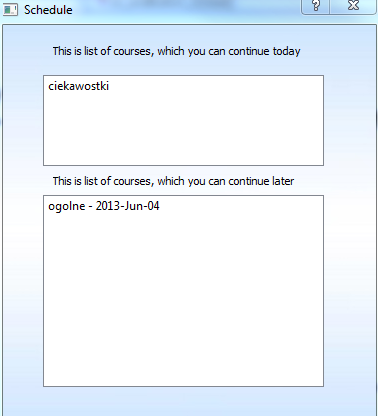
Aplikacja została wyposażona w graficzny interfejs użytkownika. Program został zrealizowany wykorzystując wzorzec projektowy Model Widok Kontroler. Dzięki niemu udało się oddzielić logikę aplikacji od GUI, więc program niezależnie od intensywności wykonywanych obliczeń pozostaje wrażliwy na akcje użytkownika.



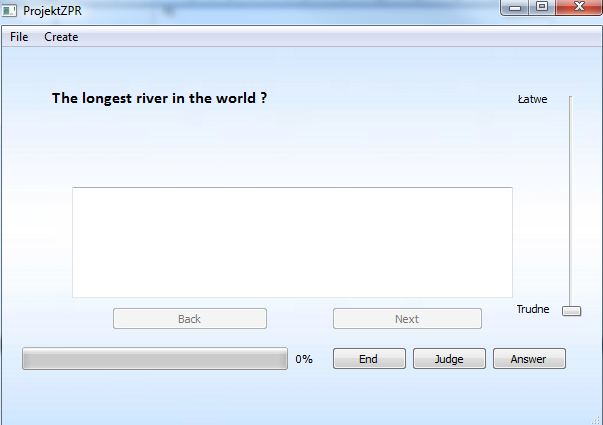
Na rysunku powyżej widzimy, że użytkownik ma możliwość stworzenia własnego kursu poprzez tworzenie zadań mając do wyboru odpowiedź otwartą lub zamknięta. Po wybraniu opcji *Save Course* kurs zapisuje się do pliku XML.



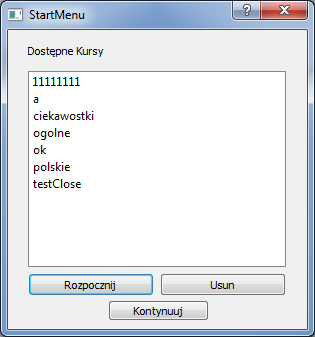
**Plan kursów do przerobienia**

****

Użytkownik po wybraniu odpowiedniej opcji (File ->Schedule) ma możliwość przeglądania listy kursów do przerobienia. Wybierając *Start*  z menu głównego użytkownik ma opcję zaczęcia kursu od nowa lub kontynuacji.

**Rozpoczęcie nowego kursu**

Podczas robienia kursu, użytkownikowi prezentowane jest pytanie oraz miejsce na udzielenie odpowiedzi w przypadku pytania otwartego lub zestaw odpowiedzi zamkniętych z możliwością wyboru poprawnych. Po udzieleniu odpowiedzi użytkownik powinien nacisnąć przycisk Answer. Pojawi się wtedy prawidłowa odpowiedź oraz propozycja oceny, jednak użytkownik ma możliwość zmiany tej propozycji. Jest to uzasadnione w przypadku jeżeli błąd użytkownika to zwykła literówka. Zmianę oceny zatwierdza przyciskiem Judge. Po zakończeniu kursu i kliknięciu End wyświetlane są statystyki przerobionego kursu i zostaje on poddany serializacji. Zastosowaliśmy serializację do pliku tekstowego, dzięki czemu użytkownik może podejrzeć ten plik, (jest to również przydatne przy testowaniu aplikacji gdzie w konkretnym pliku tekstowym można zmienić datę następnej powtórki i tak oszukać aplikację, w ten sam sposób sprawdzając poprawność odpowiednich pytań w danym dniu). Nazwa pliku jest w formie [nazwa\_kursu].txt.



Po naciśnięciu przycisku Rozpocznij na nazwie konkretnego kursu, aplikacja przenosi nas do okna z pytaniami. Mamy również możliwość usunięcia konkretnego kursu. Jeśli wybierzemy opcję kontynuuj na dowolnym kursie to mogą stać się 3 rzeczy. Pierwszą z nich jest poinformowanie użytkownika, że nie rozpoczął jeszcze danego kursu więc oczywiście nie może go kontynuować (po naciśnięciu tego przycisku deserializujemy wybrany kurs). Drugą z nich jest informacja że w danym kursie, choć już wcześniej zaczętym, nie ma pytań do powtórki i informacja kiedy powtórzyć te pytania. Trzecią opcją jest wyświetlenie odpowiednich pytań dla użytkownika. Po tym jak użytkownik odpowie na wszystkie pytania, po naciśnięciu END ponownie przeliczana jest data następnej powtórki.

1. **Budowa aplikacji**

Program jest zrealizowany w architekturze Model Widok Kontroler. Do stworzenia interfejsu graficznego wykorzystaliśmy bibliotekę QT. Wielokrotnie wykorzystywaliśmy bibliotekę boost. Między innymi do serializacji, odczytu plików XML oraz wykorzystywaliśmy sprytne wskaźniki.

Zestaw użytych klas należących do Widoku:

* + - **View** – reprezentuje główne okno programu
    - **CreateTest** – reprezentuje okno tworzenia kursu
    - **Schedule**– reprezentuje okno zaplanowanych kursów
    - **StartMenu**– reprezentuje okno wyboru kursu
    - **Statistic**– reprezentuje okno statystyk

Zestaw użytych klas należących do Modelu:

* **Model** – klasa reprezentuje całą logikę aplikacji.
* **Course** – klasa reprezentująca logikę tworzenia nowego kursu.
* **Deck** – klasa reprezentuje zbiór pytań do wyświetlenia podczas kursu. Udostępnia ona wektor sprytnych wskaźników do obiektów klasy QuestionCard
* **Mark ­**– pomocnicza klasa obliczająca proponowaną ocenę użytkownikowi po udzieleniu odpowiedzi.
* **QuaestionCard** – reprezentuje pytanie i odpowiedź, które są wyświetlane w danej

chwili

* **Start –** klasa reprezentuje logikę rozpoczęcia lub kontynuacji kursu

**Controller** – jest to klasa Kontrolera, która umożliwia połączenia widoku oraz kontrolera. W aplikacji wykorzystano mechanizm sygnałów i slotów wykorzystywany w QT. W celu komunikacji z Kontrolerem, Widok emituje sygnały a Kontroler tworzy sloty. Pozostałe okna dialogowe również emitują sygnały, ale wywołują przy tym sygnału Widoku, które następnie wywołują sloty kontrolera.

Dostarczoną dokumentację kodu wygenerowaliśmy używając narzędzia Doxygen.

1. **Napotkane trudności**

Pierwszą napotkaną trudnością było przenoszenie kodu źródłowego pomiędzy członkami zespołu. Aby to rozwiązać zapoznaliśmy się z systemem kontroli wersji GIT i go używaliśmy wykorzystując serwis github.

Następnym problemem było zapisanie obiektu przerobionego kursu ze wszystkimi ocenami w celu późniejszej kontynuacji. Wykorzystaliśmy do tego mechanizm serializacji. Ponadto należało zorganizować tak przyciski, aby użytkownik nie wykonał nieodpowiedniej akcji. Zrobiliśmy to przez komunikaty wyświetlane (warning), gdy akcja jest nieodpowiednia.

Aplikacje pisaliśmy korzystając z Visual Studio. Brakło nam niestety czasu na uruchomienie aplikacji na Linuxie.

**Źródła:**

1. <http://www.supermemo.com/english/ol/sm2.htm-> opis zaimplementowanego algorytmu.
2. Slajdy z przedmiotu ZPR – wykorzystywane w trakcie wykonywania projektu.