BAZA DANYCH Z INTERFEJSEM GRAFICZNYM

Autor: Eryk Wicher
Akademia Górniczo-Hutnicza

Spis treści

1. Wstęp

Dokument dotyczy projektu bazy danych z interfejsem graficznym z możliwością zapisywania i odczytywania rekordów z pliku .txt. Jest wykonany w QT framework.

- **Wymagania systemowe**

Program może działać na każdym z głównych systemów operacyjnych (windows, linux, macos), jednak trzeba skompilować go na tym systemie.

2. Funkcjonalność

Program posiada takie funkcjonalności jak:

Wyświetlanie poszczególnych rekordów,

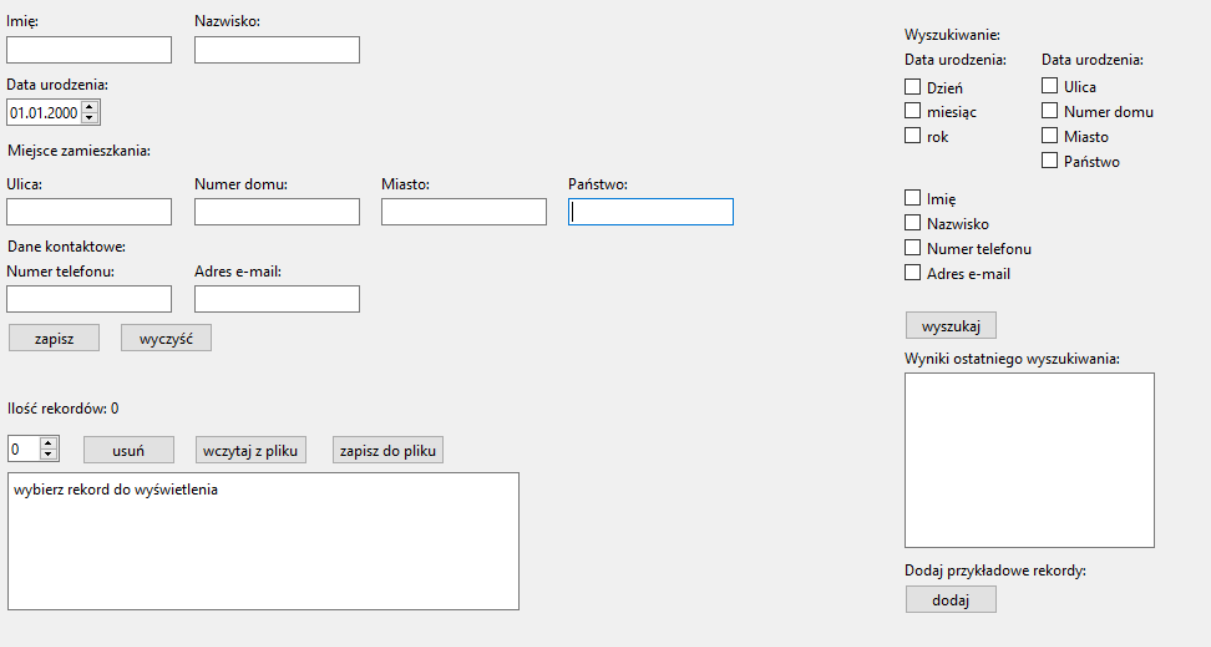
Dodawanie rekordów za pomocą pól tekstowych w programie jak i z plików .txt,

Usuwanie wybranych rekordów,

Zapisywanie rekordów do pliku .txt,

Wyszukiwanie rekordów w zależności od wybranych kryteriów.

User interface programu:



Struktura rekordu:

Nazwa	Typ danej	Dodatkowe informacje
Imię	Tekst	-
Nazwisko	Tekst	-
Data urodzenia		
Dzień	Liczba	Ograniczenie od 1 do 31
Miesiąc	Liczba	Ograniczenie od 1 do 12
Rok	Liczba	
Miejsce zamieszkania		
Ulica	Tekst	
Numer domu	Liczba	
Miasto	Tekst	
Państwo	Tekst	
Dane kontaktowe:		
Numer telefonu	Liczba	Ograniczenie do liczb składających się z 9 cyfr
Adres e-mail	tekst	

Zapisywanie rekordów w programie:

Zapisując rekord w programie należy wypełnić wszystkie pola odpowiednią wartością. Jeżeli jakieś pole będzie miało złą wartość przy próbie zapisu wyskoczy błąd.

Wczytując rekord z pliku program nie sprawdza czy dane mieszczą się w ograniczeniach z powyższej tabeli, ponieważ użytkownik nie powinien ingerować w te pliki. Jednakże jeżeli użytkownik spróbuje otworzyć plik niezgodny z formatem pliku przeznaczonego do odczytywania, program pokaże błąd ale nadal będzie kontynuować działanie.

Zapisywanie rekordów do pliku:

Zapisywany jest aktualnie wybrany rekord w miejscu wybranym przez użytkownika za pomocą okienka dialogowego.

Usuwanie rekordów:

Usuwany jest aktualnie wybrany rekord, a użytkownik jest przenoszony do zerowego rekordu (braku rekordu).

Wyszukiwanie rekordów:

Wyszukując rekord należy wybrać co najmniej jedno kryterium, a poprawność wpisanych danych będzie sprawdzana tylko dla zaznaczonych kryteriów. Wynik jest w postaci numeru indeksu rekordu.

3. Analiza problemu

Najlepszym sposobem na implementację bazy danych z interfejsem graficznym jest podzielenie jej na dwie części. Pierwsza to napisanie bazy danych w czystym c++ najlepiej używając funkcji ze standardowych bibliotek z funkcjami pozwalającymi na pozyskiwanie składników tej bazy. Drugą częścią jest zrobienie projektu za pomocą framework-ów umożliwiających tworzenie interfejsów graficznych takich jak QT.

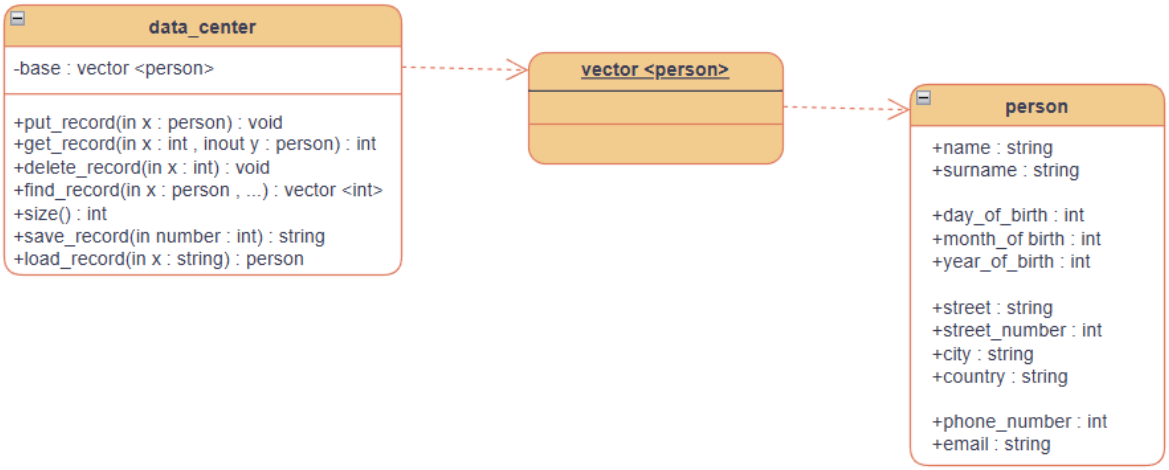
Taki podzielenie projektu sprawia, że przeniesienie projektu na inne platformy lub framework-i nie wymaga tworzenia bazy danych, bo jest ona uniwersalna o ile piszemy w c++.

4. Projekt techniczny

- ## Hierarchia klas

Baza danych jest zaprojektowana w taki sposób aby użytkownik nie miał dostępu do jej składowych. Może się z nimi komunikować tylko za pomocą funkcji.

Diagram UML bazy danych



Funkcja „find_record" była za długa żeby ją wpisać do diagramu.

FIND_RECORD()
vector<int> find_record(person x,int name_i, int surname_i, int day_of_birth_i, int month_of_birth_i, int year_of_birth_i, int street_i, int street_number_i, int city_i, int country_i, int phone_number_i, int email_i)

Niestety baza nie jest uniwersalna, ponieważ nie da się zamienić jej głównej składowej, typu vectora. Trzeba było by wtedy zmieniać funkcje find_ record() , save_ record() i load_record() gdyż bazują one na składowych struktury person.

- ## Opis funkcji

➢ ***Put_record()***

void put_record(person x)

Dopisuje strukturę person do bazy danych.

- ***Get_record()***

```
int get_record(int x, person& y)
```

Funkcja pobiera index x i pobiera referencje do struktury person do której jest zapisywany wybrany rekord. Index x odwołuje się bezpośrednio do vectora więc jego pierwszy element to 0. Zwraca 0 lub 1 w zależności od tego czy dany rekord istnieje, w celu zapobiegania odwoływania się do nieistniejących elementów vectora.

- ***delete_record()***

```
void delete_record(int x)
```

Usuwa rekord o wybranym indexe. Tak samo jak w get_record() index należy zmniejszyć o 1.

- ***find_record()***

```
vector<int> find_record(person x,int name_i, int surname_i, int day_of_birth_i,
                        int month_of_birth_i, int year_of_birth_i, int street_i, int street_number_i,
                        int city_i, int country_i, int phone_number_i, int email_i)
```

Funkcja pobiera strukturę person do której będzie porównywana cała baza oraz wartości 0 lub 1 w każdej innej zmiennej. Decydują one, który składnik struktury będzie brany pod uwagę przy szukaniu rekordów. 1 oznacza że pędzie przeszukany. Zwraca vector z indexami rekordów, które wyszukał. Te indexy są numerowane od 0.

- ***size()***

```
int size(void)
```

Zwraca ilość elementów.

- ***Save_record()***

```
string save_record( int number)
```

Pobiera index numerowany od 1, a zwraca string zawierający wybrany rekord w formie odpowiedniej do zapisania do pliku.

- ***Load_record()***

```
person load_record(string x)
```

Funkcja zaprojektowana do odczytywania rekordów z pliku. Pobiera stringa, którego trzeba wyekstraktować z pliku. Zwraca strukturę person zawartą w stringu.

Funkcję save_record() i load_record() nie używają żadnych funkcji piszących do plików aby można było tych funkcji używać w wielu frameworkach.

- **Obsługa wyjątków**

W programie jest parę funkcji, które mają potencjał do jego awaryjnego wyłączenia. Głównie są to vectory, zapisywanie i odczytywanie plików.

W przypadku wektora Program jest zabezpieczony aby niemożliwe było wywoływanie funkcji, które odnoszą się do nieistniejącego indexu albo w funkcji albo w kodzie programu.

Drugim przypadkiem jest zapisywanie i odczytywanie plików. Jest tu zastosowany mechanizm sprawdzający czy ścieżka to nullptr aby nie używać funkcji ifstream lub ofstream na nieistniejącym pliku, co mogło by wyłączyć program. Mogło to wystąpić gdy użytkownik kliknął zapisz do pliku, a następnie rozmyślił się i zamknął okno dialogowe.

W przypadku wczytywania z pliku trzeba było zastosować mechanizm try catch na wypadek niepoprawnego tekstu w wczytywanym pliku.

5. Używane narzędzia

Pogram został wykonany w QT framework, który umożliwia proste tworzenie interfejsów graficznych. Dodatkowo została do niego dodana klasa data_base.

Program był tworzony w środowisku windows 10 w programie Qt creator 15.0.0 (Community). Używanym kompilatorem był gcc, natomiast wersja języka c++ to 20.

Natomiast do testowania bazy danych zostały użyte Gtesty w programie Microsoft visual studio.

6. Wykonane testy

Wykonane Gtesty są załączone w tym projekcie oraz odnoszą się do tylko do bazy danych. Mają za zadanie sprawdzić czy funkcje bazy danych działają poprawnie i zawszę zwracają oczekiwany wynik.

Ich opis znajduje się poniżej oraz wszystkie odnoszą się do klasy data_center.

- ***Size_test:***

```
TEST(TestCaseName, size_test) {
        base.put_record(per1);
        base.put_record(per2);
        EXPECT_EQ(base.size(), 2);
```

Ma za zadanie sprawdzić czy funkcja .size() zwraca poprawny wynik.

- ***Put_record_test:***

```
TEST(TestCaseName, put_record_test) {
        base.put_record(per3);
        EXPECT_EQ(base.size(), 3);
```

Sprawdzanie czy dodanie 1 rekordu powiększy bazę.

- ***Get_record_test:***

```
TEST(TestCaseName, get_record_test1) {
        person y;
        int x = base.get_record(0, y);
        EXPECT_EQ(x, 0);
        EXPECT_EQ(y.name, per1.name);
        EXPECT_EQ(y.surname, per1.surname);
```

Sprawdzanie czy get_record zwraca prawidłową strukturę zawartą w podanym indexe. Struktura per1 została wpisana wcześniej na 0 indexe.

- ***Get_record_test2:***

```
TEST(TestCaseName, get_record_test2) {
        person y;
        int x = base.get_record(3, y);
        EXPECT_EQ(x, 1);
        std::cout << y.name;
```

Test próbuje dostać się do indexu, który nie istnieje więc funkcja powinna zwrócić błąd, który objawia się 1 na wyjściu.

- ***Delete_record_test:***

```
TEST(TestCaseName, delete_record_test) {
        base.delete_record(2);
        EXPECT_EQ(base.size(), 2);
```

Test sprawdza czy usuwanie rekordu działa.

- ***Find_record_test:***

```
TEST(TestCaseName, find_record_test) {
        base.put_record(per3);
        vector <int> k = base.find_record(per2, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0);
        EXPECT_EQ(k[0], 1);
```

Wprowadzamy z powrotem strukturę per3 i wyszukujemy strukturę per2 która została wprowadzona do bazy i ma index 1.

- ***Load_record_test:***

```
TEST(TestCaseName, load_record_test) {
        person u =
        base.load_record("Ula,Nowak,30,9,1986,balicka,87,Mydlniki,Polska,444555666,ula@hotmail
        .com");
        EXPECT_EQ(u.name, per1.name);
        EXPECT_EQ(u.surname, per1.surname);
```

Test sprawdza czy baza poprawnie odczytuje rekordy z zmiennej string.

- ***Save_record_test:***

```
TEST(TestCaseName, save_record_test) {
        string st1 = base.save_record(1);
        string st2 = "Ula,Nowak,30,9,1986,balicka,87,Mydlniki,Polska,444555666,ula@hotmail.com
        EXPECT_EQ(st1, st2);
```

Sprawdzanie czy funkcja zwracająca rekord w postaci string zwraca go poprawnie.

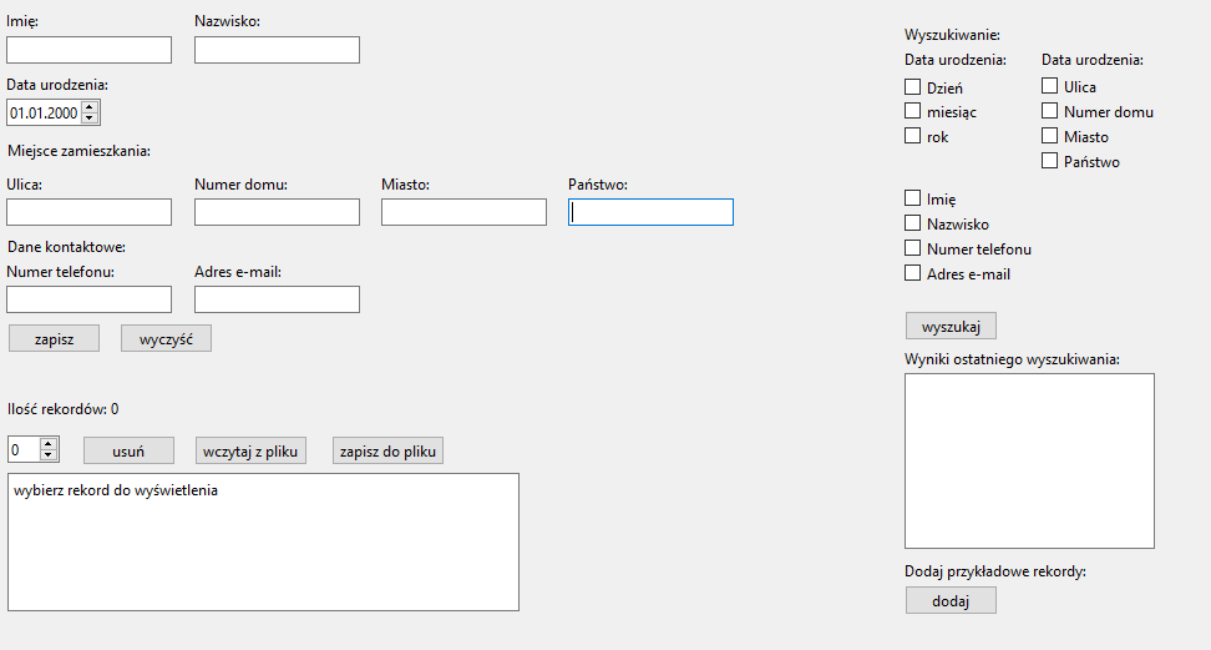
• Testy całego programu

Testy działania całego programu były wykonywane ręcznie. Były przetestowane między innymi:

- wprowadzanie danych do pól tekstowych i sprawdzanie czy program wpisze rekord z błędnymi danymi wejściowymi.
- w przypadku odczytywania i zapisywania rekordów z plików testowane były głównie sytuacje w których użytkownik wybrał niepoprawny plik bądź zamknął okienko dialogowe. W pierwszych wersjach programu ponieważ po zamknięciu okienka program nie otrzymuje poprawnej ścieżki do pliku mógł się awaryjnie wyłączyć. To samo odnosi się do próby odczytania niepoprawnie zapisanego pliku.

7. Podręcznik użytkownika

User interface programu:



Po włączeniu programu pojawi się interfejs pokazany powyżej. Standardowo nie ma zapisanych żadnych rekordów. W prawym dolnym rogu jest opcja dodania 5 przykładowych rekordów oraz dodatkowo w folderze projektu są przykładowe rekordy, które można wczytać.

Zmienianie aktualnie oglądanego rekordu odbywa się za pomocą strzałeczek na lewo od przycisku usuń. Zerowy rekord zawsze wyświetla się jako „wybierz rekord do wyświetlenia”.

Usuwanie rekordu odbywa się poprzez naciśnięcie przycisku „usuń”. Wtedy zostanie usunięty rekord znajdujący się pod aktualnie wybranym indexem i wyświetlający się pod przyciskiem.

Aby zapisać rekordy za pomocą formularzu w lewym górnym rogu należy wypełnić wszystkie jego pozycje. Należy jednak pamiętać, że do pola „numer domu” oraz „numer telefonu” można wpisać tylko liczbę dodatkowo do tego drugiego liczba musi być 3-cyfrowa. Wszystkie pola muszą być wypełnione aby zapisać rekord.

Wyszukiwanie rekordów odbywa się za pomocą formularza znajdującego się w prawym górnym rogu w którym wybiera się kryteria, według których program przeszukuję bazę. Wtedy nie muszą być wypełnione wszystkie pola, a jedynie te które są zaznaczone jako kryteria do wyszukiwania.

Bibliografia

[1] Dokumentacja techniczna QT Qt Documentation | Home
[2] Dokumentacja techniczna cmake CMake - Cross Platform Make

Dokumentacja została stworzona na bazie szkicu zapewnionego przez Dr hab. inż. Bogusław Cyganek