Języki Programowania Obiektowego - Laboratorium

Sprawozdanie z projektu

**BAZA DANYCH Z INTERFEJSEM GRAFICZNYM**

Projekt wykonał:

Huber Pałka  
Akademia Górniczo-Hutnicza  
Elektronika i Telekomunikacja

Spis treści

[1. Wstęp 3](#_Toc188111478)

[2. Obsługa Aplikacji 3](#_Toc188111479)

[3. Opis realizacji 4](#_Toc188111480)

[4. Funkcjonalność 4](#_Toc188111481)

[5. Napotkane problemy 7](#_Toc188111482)

[6. Możliwości rozbudowy 7](#_Toc188111483)

[7. Podsumowanie 7](#_Toc188111484)

[8. Bibliografia 8](#_Toc188111485)

# Wstęp

Poniższy dokument stanowi opracowanie projektu bazy danych z interfejsem graficznym, przygotowanego na zajęcia z języków programowania obiektowego. Program ten pozwala użytkownikowi na stworzenie bazy danych samochodów.

# Obsługa Aplikacji

Do podstawowych funkcji programu należy możliwość dodawania, usuwania oraz wyszukiwania samochodów w bazie danych. Dodatkowo istnieje funkcjonalność wyczyszczenia bazy danych za pomocą jednego przycisku.

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

W przypadku dodawania oraz usuwania należy do pól Car Brand, Model, Year oraz Color wpisać wszystkie dane pojazdu, a następnie w celu dodania należy wcisnąć przycisk SUBMIT, a w celu dodania przycisk DELETE. W przypadku wyszukiwania, można podać tylko część danych, a po wciśnięciu przycisku SEARCH program pokaże wszystkie samochody o pasujących cechach. W sytuacji, w której nie podane zostaną żadne argumenty, program pokaże wszystkie samochody znajdujące się w bazie. W przypadku czyszczenia bazy danych, nie jest wymagane podawanie jakichkolwiek danych.

# Opis realizacji

Celem projektu było stworzenie aplikacji MFC umożliwiającej zarządzanie bazą danych samochodów. W tym celu skorzystano z poniższych narzędzi oraz technologii:

* Microsoft Visual Studio 2022 – środowisko programistyczne,
* MFC (Microsoft Foundation Class) – wykorzystane do stworzenia interfejsu graficznego,
* CMake – do zarządzania budowaniem projektu,
* GitHub – do kontroli wersji i przechowywania kodu
* Google Test – do testowania kodu
* Pliki tekstowe txt – do przechowywania danych

Projekt rozpoczęto od stworzenia wszystkich elementów interfejsu MFC. Następnie przystąpiono do tworzenia każdej z funkcji.

# Funkcjonalność

Funkcjonalność programu opiera się na tworzeniu dwóch plików tekstowych: cars.txt oraz cars\_temp.txt. Pierwszy plik wykorzystywany jest przy każdej z operacji, natomiast drugi tylko do usuwania.

**4.1 Dodawanie pojazdów**

Pierwszą funkcjonalnością jest dodawanie pojazdów do bazy danych.

void CMFCApplication2Dlg::OnBnClickedButton1()

{

// Aktualizuj zmienne z GUI

UpdateData(TRUE);

if (Manufacturer.IsEmpty() || Model.IsEmpty() || Year.IsEmpty() || Color.IsEmpty())

{

AfxMessageBox(\_T("All fields must be filled before adding data!"));

return;

}

Korzystając z funkcji OnBnClickedButton1(), po wciśnięciu przycisku SUBMIT, włączana jest możliwość modyfikacji danych w oknach do wpisywania. Następnie sprawdzane jest, czy wszystkie pola są uzupełnione.

// Otwórz plik w trybie dodawania danych

CStdioFile file;

CString filePath = \_T("cars.txt");

if (file.Open(filePath, CFile::modeCreate | CFile::modeNoTruncate | CFile::modeWrite))

Następnie przy użyciu CStdioFile, funkcji wbudowanej w MFC tworzymy nowy plik o nazwie cars.txt, bądź go otwieramy jeżeli taki plik już istnieje. Opcja CFile::modeNoTruncate, umożliwia otwieranie pliku bez usuwania jego zawartości.

{

// Ustaw wskaźnik pliku na koniec, aby dopisywać nowe dane

file.SeekToEnd();

// Formatowanie danych do zapisu

CString carData;

carData.Format(\_T("Manufacturer: %s, Model: %s, Year: %s, Color: %s\n"),

Manufacturer, Model, Year, Color);

// Zapisz dane do pliku

file.WriteString(carData);

file.Close();

Kolejnym krokiem jest ustawienie wskaźniku w pliku na sam koniec, po czym dane z pól do wpisywania są formatowane oraz zapisywane w pliku cars.txt. Na koniec plik jest zamykany.

// Pokaż komunikat o sukcesie

AfxMessageBox(\_T("Car data has been saved!"));

Manufacturer = (\_T(""));

Model = (\_T(""));

Year = (\_T(""));

Color = (\_T(""));

UpdateData(FALSE);

}

else

{

AfxMessageBox(\_T("ERROR : Unable to access the database!"));

}

}

Ostatnią częścią jest wyczyszczenie pól służących do wpisywania oraz wypisanie komunikatu w oknie o powodzeniu lub niepowodzeniu zadania.

**4.2 Wyszukiwanie pojazdów**

Drugą funkcjonalnością jest wyszukiwanie pojazdów w bazie danych. Ze względu na powtarzające się formuły z funkcjonalności pierwszej, w tym i pozostałych punktach opisane zostaną tylko elementy unikatowe dla tych części.

W momencie wciśnięcia przycisku SEARCH wykonuje się kod odpowiedzialny za wyszukiwanie pojazdów.

CStdioFile file;

if (file.Open(\_T("cars.txt"), CFile::modeRead))

{

CString line;

CString matchingCars;

bool found = false;

while (file.ReadString(line))

{

// Sprawdź po dokładnym dopasowaniu wszystkich wypełnionych pól

bool match = true;

if (!Manufacturer.IsEmpty() && line.Find(\_T("Manufacturer: ") + Manufacturer) == -1)

match = false;

if (!Model.IsEmpty() && line.Find(\_T("Model: ") + Model) == -1)

match = false;

if (!Year.IsEmpty() && line.Find(\_T("Year: ") + Year) == -1)

match = false;

if (!Color.IsEmpty() && line.Find(\_T("Color: ") + Color) == -1)

match = false;

if (match)

{

matchingCars += line + \_T("\n");

found = true;

}

}

Plik zostaje otwarty przy użyciu file.Open a następnie tworzone są trzy zmienne pomocnicze : stringi line oraz matchingCars, a także bool found o wartości false. W pętli while, plik cars.txt jest skanowany linijka po linijce w celu znalezienia pasujących samochodów w bazie danych. Tworzona jest następna zmienna pomocnicza bool match o wartości true. Następnie sprawdzane jest po kolei każde z pól używanych do wpisywania i w przypadku kiedy nie jest ono puste przechodzi dalej i porównuje wartość wpisaną wraz z bazą danych. Jeśli się nie zgadza, wartość match zmienia się na false. W przeciwnym razie, w przypadku zgodności wyszukiwania z bazą danych, linijka jest dodawana do zmiennej matchingCars w celu wyświetlenia tego wyniku. Dodatkowo zmienia zmienną found na true, w celu wykonania funkcjonalności wyświetlenia.

**4.3 Czyszczenie bazy danych**

Kolejną funkcjonalnością jest usuwanie całej bazy danych przy użyciu przycisku CLEAR.

CStdioFile file;

if (file.Open(filePath, CFile::modeCreate | CFile::modeWrite))

W tym celu plik cars.txt jest otwierany, jednak bez opcji CFile::modeNoTruncate, co powoduje nadpisanie starego pliku nowym, pustym plikiem.

**4.4 Usuwanie pojazdów**

Ostatnią funkcjonalnością jest usuwanie elementów z bazy danych przy użyciu przycisku DELETE. Ta opcja jest najbardziej skomplikowana i wykorzystuje dodatkowy plik jakim jest cars\_temp.txt.

Podobnie jak opcja dodawania, opcja usuwania wymaga od użytkownika podania wszystkich danych pojazdu w polach do wpisywania. Następnie otwierane są pliki cars.txt oraz cars\_temp.txt oraz kopiowana jest zawartość cars.txt do pliku tymczasowego. Kolejnym krokiem jest znalezienie w pliku cars.txt linijki z podanym samochodem.

if (line.Find(\_T("Manufacturer: ") + Manufacturer) != -1 &&

line.Find(\_T("Model: ") + Model) != -1 &&

line.Find(\_T("Year: ") + Year) != -1 &&

line.Find(\_T("Color: ") + Color) != -1)

{

deleted = true;

}

else

{

tempFile.WriteString(line + \_T("\n"));

}

Jeżeli zostanie znaleziony pojazd, wartość zmiennej lokalnej delete jest ustawiana na true, a następnie wykonywana usunięcia oryginalnego pliku z bazą danych, oraz zmiana nazwy pliku tymczasowego na cars.txt, tym samym tworząc nową bazę danych.

if (deleted)

{

if (\_tremove(filePath) != 0 || \_trename(tempFilePath, filePath) != 0)

{

AfxMessageBox(\_T("ERROR : Unable to update the database!"));

}

else

{

AfxMessageBox(\_T("Car was successfuly deleted from the database"));

Manufacturer = (\_T(""));

Model = (\_T(""));

Year = (\_T(""));

Color = (\_T(""));

}

}

# Napotkane problemy

W trakcie tworzenia aplikacji napotkano różne problemy. Jednym z nich było poprawne tworzenie oraz zapisywanie plików. Błąd został naprawiony poprzez użycie odpowiednich funkcji CFile:: w funkcji open. Pojawił się także problem ze zmianą nazwy projektu. Niestety ze względu na mnogość występowań nazwy projektu w plikach aplikacji, zmienione zostały tylko podstawowe nazwy, tak aby nie utracić działania aplikacji.

# Możliwości rozbudowy

Projekt można rozbudować o kolejne dane opisujące pojazd, jak na przykład numer rejestracyjny, właściciel czy chociażby kraj pochodzenia samochodu. Dodatkowo, można także rozbudować program poprzez dodanie funkcjonalności importowania gotowych baz danych z pluku CSV lub JSON.

# Podsumowanie

Projekt można uznać za zakończony sukcesem, wszystkie założenia początkowe projektu zostały Realizacja projektu umożliwiła zdobycie wiedzy z zakresu projektowania aplikacji z interfejsem wizualnym. Aplikacja działa zgodnie z założeniami, umożliwiając tworzenie użytkownikowi bazy danych z interfejsem graficznym oraz wykorzystuje wymagane metody.

# Bibliografia

<https://learn.microsoft.com/en-us/cpp/mfc/mfc-desktop-applications?view=msvc-170>

<https://learn.microsoft.com/en-us/cpp/mfc/reference/cstdiofile-class?view=msvc-170>

<https://stackoverflow.com/questions/11580748/using-cmake-for-making-a-project-which-includes-mfc>

<https://github.com/google/googletest/blob/main/googletest/README.md>

Programowanie w języku C++ Wprowadzenie dla inżynierów – Dr hab. inż. Bogusław Cyganek