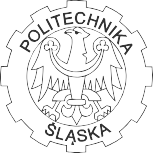
### Politechnika Śląska w Gliwicach

Wydział Informatyki, Elektroniki i Informatyki



**PODSTAWY PROGRAMOWANIA KOMPUTERÓW**

Zarządzanie flotą pojazdów i kierowców

Autor Bartłomiej Kucharski

prowadzący prof. dr hab. inż. Wojciech Sułek  
rok akademicki 2020/2021

kierunek teleinformatyka

rodzaj studiów SSI

semestr 1

termin laboratorium / ćwiczeń wtorek i czwartek 8:00 – 10:15 grupa 4

sekcja 10

termin oddania sprawozdania 13.12.2020

data oddania sprawozdania 13.12.2020

# Teść zadania.

# Proszę napisać program umożliwiający zapisywanie, organizowanie oraz zarządzanie flotą pojazdów i kierowców. Pojazdy są zapisane w pliku w następujący sposób:

# *Przykładowy plik:* *osobowy Audi A6 B osobowy Seat Leon B ciezarowy Renault Magnum C*

# *motocykl honda CBR1000RR*

# A Kierowcy są zapisani w pliku w następujący sposób:

# *Przykładowy plik:*

# *Jan Kowalski A*

# *Adam Nowak AB*

# *Zenon Iksiński BCDT*

# Program ma umożliwiać:

# • dodanie nowego pojazdu,

# • usunięcie istniejącego pojazdu,

# • dodanie pojazdu,

# • usunięcie pojazdu,

# • wyświetlenie listy pojazdów (posortowanych wg wszystkich możliwych kryteriów),

# • filtrowanie listy pojazdów (wg wszystkich możliwych kryteriów),

# • wyświetlenie listy kierowców (posortowanych wg wszystkich możliwych kryteriów),

# • filtrowanie listy kierowców (wg wszystkich możliwych kryteriów),

# • wygenerowanie raportu zawierającego (także uwzględniające filtrowanie) kierowców i samochodów którymi mogą podróżować.

# Zakończenie pracy z programem ma powodować zapisanie na dysku aktualnych danych (kontakty i zadania). Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników:

# *-k plik wejściowy z kierowcami*

# *-p plik wejściowy z pojazdami*

## Analiza zadania.

Zadanie przedstawia manipulacje na tablicach, sortowanie tablic, zapis i odczyt plików tekstowych oraz transformacje na zmiennych łańcuchowych.

### Struktury danych.

W programie wykorzystano stringi, inty, listy oraz manipulatory z biblioteki *<iomanip>.* Stringi są to typy danych służący do przechowywania ciągi znaków (zmiennych łańcuchowych). Inty są to typy danych definiujące liczbę całkowitą. Lista jest dynamiczną strukturą danych, co oznacza, że jej rozmiar (ilość zawartych na liście elementów) może się dowolnie (z ograniczeniem pamięci komputera) zmieniać w czasie działania programu.

### Algorytmy

Program stosuje algorytmy, które odczytują i zapisują do list podane dane w odpowiednim formacie z plików zewnętrznych. Przy pomocy iteratorów „it” stworzone listy są przetwarzane „it”-razy. Dodawanie   
oraz usuwanie, jak i wyszukiwanie elementów z list są zabezpieczone przed wprowadzeniem małych liter na poczatku. Sortowanie wykorzystuje wartość zwracanej funkcji bool jako funkcji porównania binarnego przy czym funkcja już określa sorowanie nie malejące. Dodatkowo po przez funkcje switch są odróżniane kategorie którymi ma się kierować program sortując elementy. W każdym momencie moża zapisać aktualny stan list pojazdów i kierowców, nadpisując aktualne pliki które były odczytywane. Natomiast funkcja generująca raport przydzielający auta odpowiednim osobom, wykorzustuje wcześniej wspomniane iteratory i po przez odpowiednie wyodrebnione stringi sa wypisywane pod warunkiem posiadania danej kategorii prawa jazdy.

## Specyfikacja zewnętrzna.

Program jest uruchamiany przy pomocy linii poleceń. Dany plik jest przekazywane przez odpowiedni przełącznik:

*„- k” dla pliku z kierowcami   
lub  
 „–p” dla pliku z pojazdami*

W przypadku uruchomienia programu bez parametrów lub z błędnymi parametrami zwróci instrukcję informującą o ponownym podaniu parametrów do plików.

# Specyfikacja wewnętrzna.

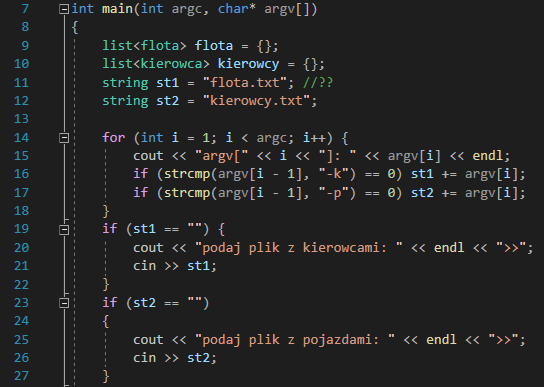
Program składa się z trzech plików:

***main.cpp*** – w tym pliku są wywoływane funkcje za pomocą switcha oraz sprawdzane są przełączniki plików zewnętrznych.

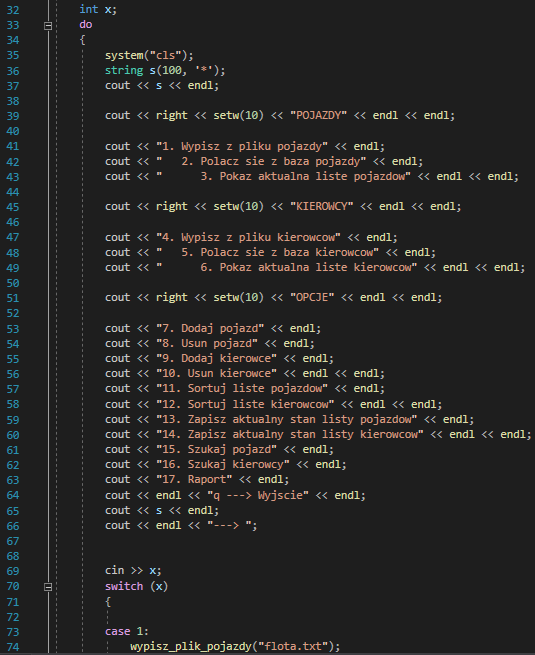
***navi.h*** – jest to plik nagłówkowy ze wszystkimi nagłówkami funkcji zdefiniowanymi w programie.  
***navi.cpp*** - plik w którym zostały zdefiniowane wszystkie funkcje które można było wyłączyć poza plik ***main.cpp***

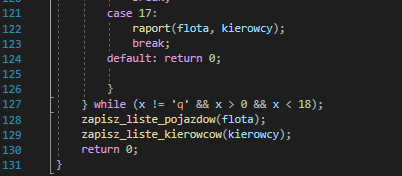
### Ogólna struktura programu.

W funkcji głównej ***main*** zostały wywołane funkcje sprawdzające przełączniki   
oraz instrukcja wyboru danej funkcji:

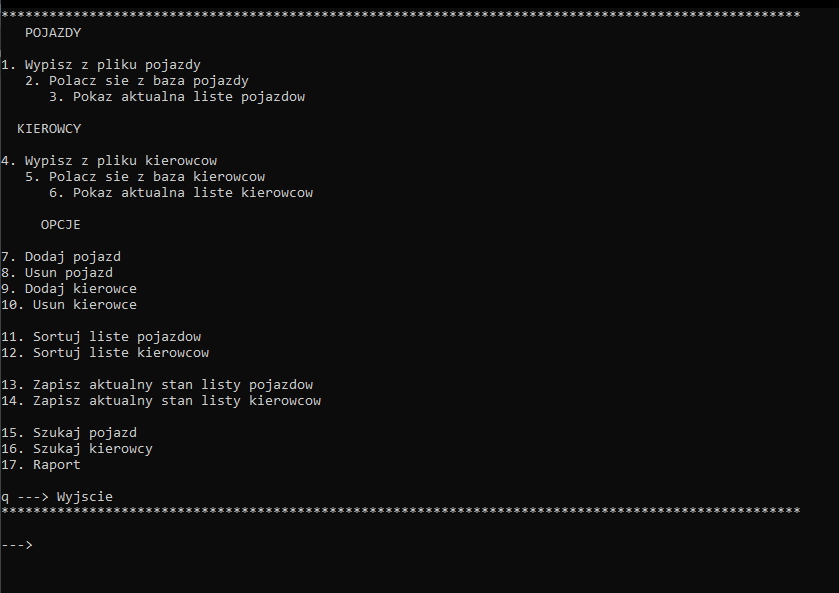


Dodatkowo zostały zimplementowane listy dla kierowców i samochodów.  
Natomiast przełączniki mają charakter trybu w jakim otwieramy pliki.

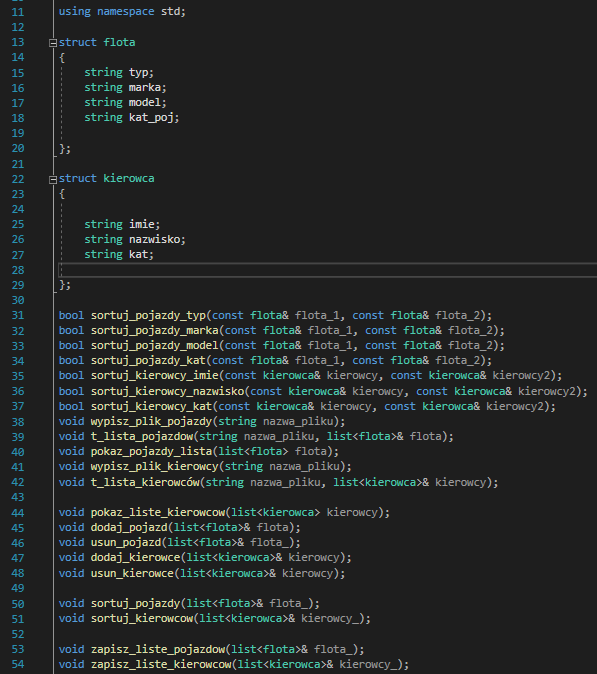




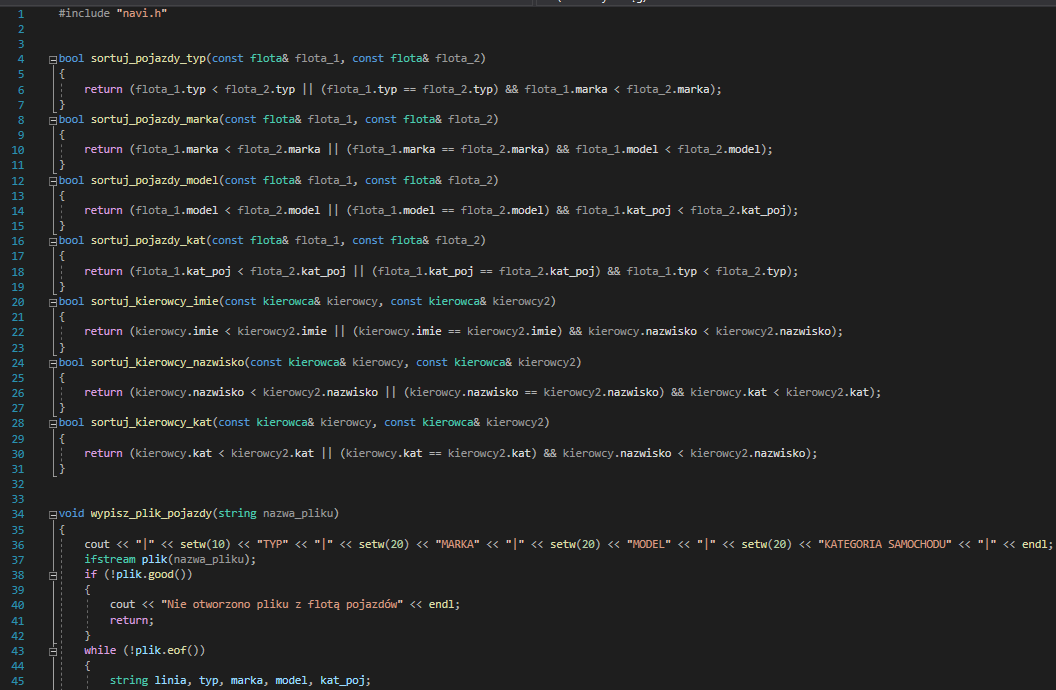
Program w dalszej części ukazuję „menu” gdzie użytkownik może wybrać wypunktowany wcześniej podprogram wykonawczy. Cała funkcja jest wykonywana dopóki użytkownik wyjdzie poza ramy funkcji while, zakańczając pętle (do-while) i zapisuje stan list do plików.



W pliku ***navi.h*** ukazane są użyte struktury wykorzystywane do list i sortowania. Poniżej znjadują się wszystkie zdefiniowane funkcje użyte w programie.

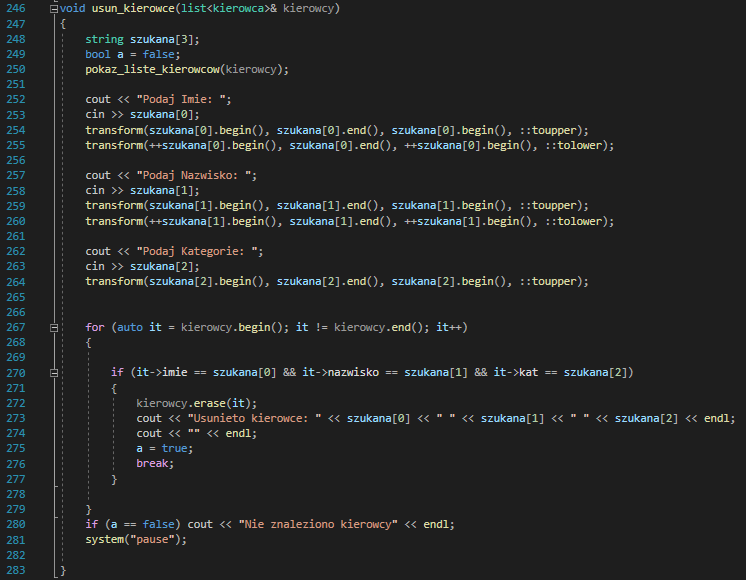


Natomiast w pliku ***navi.cpp*** znajdują się wszystkie funkcje wczesniej zdefiniowane w pliku nagłówkowym, m.in: sortowanie list, wypisywanie elementow z listy, tworzenie listy, usuwanie lub też dodawanie elementow listy, zapisywanie z listy do pliku, wyświetlanie raportu przy pomocy list oraz szukanie w listach elementów.



### Szczegółowa opis implementacji.

Poniżej ukazana jest funkcja usuwająca kierowcę z listy. Do funkcji zostaję przekazana lista przy pomocy referencji, zostają także zdefiniowane dodatkowe elementy takie jak: tablica typu string (ciągów łańcuchowych), zmienna typu bool, oraz funkcja pomocnicza zatrzymujaca program po wywołaniu warunku nie znalezienia kierowcy. Użytkownik po kolei wpisując parametry przypisywane są do tablicy. Same elementy w tablicy są modyfikowane, aby w razie pomyłki uzytkownika imię   
i nazwisko zaczynały się z dużej litery, natomiast wszytskie kategorie były dużymi literami. Po przygotowaniu parametrów do wyszukiwania, w pętli dzięki iteratorowi „it” poruszamy się po elementach iterowanych w liście kierowców. Dzięki warunku w pętli po przez iterator wskazujemy po kolei na elementy listy (imie, nazwisko, kategoria prawa jazdy), jeżeli wszystkie warunki są spełnione, elementy wcześniej wyszukiwane zostaną usunięte za pomocą funkcji ***.erase(it)*** gdzie „***it***” to iterator wskazujacy na danego kierowce, dodatkowo zmieniamy wartość zmiennej „***a”*** na true, aby program wiedzial czy skonczylismy wyszukiwanie pomyślnie i kończąc dalsze szukanie ***break’em***. Na sam koniec po wyjściu z petli, program sprawdza czy parametr „***a”*** wskazuje czy program znalazł kierowce i go usunął, jeżeli program nie znalazł kierowcy po wcześniej podanych parametrach, na ekranie program pokaże odpowiednią informacje o tym.



### Testowanie

Funkcja została przetestowana pod kątem wpisywania niepoprawnych prametrów przełączników. Po wprowadzeniu złych danych lub gdy użytkownik nie wpisze nic, zostaje wyświetlony odpowiedni komunikat. Wszystkie pliki z zewnatrz są sprawdzane pod kątem istnienia, jeżeli taki plik nie istnieje to wyswietli komunikat o tym. Wprowadzanie parametrów do programu zabezpiecza się przed złą wielkością wprowadzanych znaków tak aby format był zgodny zapisanymi danymi, tak jak np. przy dodawaniu czy usuwaniu pojazdu z listy.

# Wnioski

Problemem okazała się liczne założenia do programu aby był on spójny i aby dane prowadzane dane do listy mogły być posortowane, według zamysłu programu. Optymalizowanie niektórych funkcji było problematyczne aby nie powtarzać wczesniej wykorzystanych funkcji w programie. Program ma ustandaryzowane nazwy do kazdych z funkcji, lecz można go dopracować po przez zautomatyzowanie wprowadzenia danych z pliku do listy. Także estetyka i wystepowanie niektórych funkcji w kodzie programu mogłaby być w innym miejscu, celem lepszego poruszania się po kodzie. Cała reszta kodu okazała się być mało skomplikowana.