# 

Documentation of the student project

**ALICE’s subsystem**

z przedmiotu

**Design Laboratory**

Elektronika PL, III year

*Stanisław Zachorowski*

*Katarzyna Pióro*

Supervisor: mgr inż. Sebastian Koryciak

04.02.2022

1. **The aim of the project**The aim of project was to create some solution for bigger project(ALICE)'s needs.

These needs were cooling and remote turning on/off system.

1. **Project description**

Project presents simple city bus schedule managing system designed in text-only compter interface. It offers possibility of adding new buses, bus stops and hours schedules assigned to them by the day of the week. Created objects are editable and can be showed in the same or another program session, because all the data is stored in .txt files at corresponding directory.

1. **Instrukcja użytkownika**

### Struktura menu

Po uruchomieniu programu w konsoli zostaje wyświetlone menu, po którym można nawigować wybierając z klawiatury numerycznej przypisany do danej opcji numer..

* **[1] Przeglądaj rozkład jazdy.**
  + **[1] Przeglądaj przystanki.**Wyświetla listę zapisanych w systemie przystanków. Żeby wybrać przystanek należy wpisać jego nazwę z klawiatury. Następnie wyświetli się lista przypisanych do tego przystanku autobusów. Żeby wybrać autobus należy wpisać z klawiatury jego numer. Teraz wyświetli się wybór dni tygodnia, do których przypisane są klawisze klawiatury numerycznej. Po zatwierdzeniu poprawnie wszystkich wyborów wyświetli się rozkład godzinowy.
  + **[2] Przeglądaj autobusy.**Wyświetla listę zapisanych w systemie autobusów.
  + **[3] Powrót.**Umożliwia powrót do menu głównego.
* **[2] Zarządzaj rozkładami jazdy.**
  + **[1] Dodaj przystanek.**Dodawanie nowego przystanku lub nadpisywanie istniejącego o tej samej nazwie.
  + **[2] Dodaj autobus.**
  + Dodawanie autobusu o nowym numerze lub nadpisywanie istniejącego o tym samym numerze.
  + **[3] Dodaj rozkład.**Przypisywanie nowego rozkładu godzinowego do wybranego istniejącego przystanku i wybranego przypisanego mu autobusu.
  + **[4] Powrót.**Umożliwia powrót do menu głównego.
* **[3] Wyjdź.**

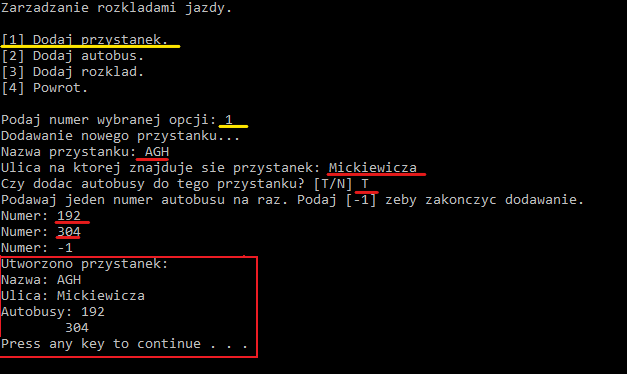
### Zarządzanie rozkładami jazdy

#### Opcja **[1] Dodaj przystanek.**

Po wybraniu tej opcji można dodać nowy przystanek i zapisać:

* nazwę (tekst bez spacji),
* ulicę na której się znajduje (tekst bez spacji),
* jeden lub więcej numerów autobusów, które odjeżdżają z tego przystanku (liczba od 1 do 1000).

**Żeby zakończyć proces należy wpisać -1.**

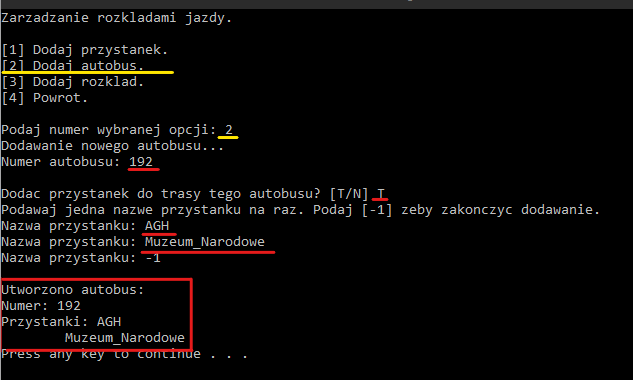
****

#### Opcja **[2] Dodaj autobus.**

Po wybraniu tej opcji można dodać nowy autobus i zapisać:

* numer (liczba w zakresie od 1 do 1000),
* jeden lub więcej przystanków, które znajdują się na trasie tego autobusu (tekst bez spacji).

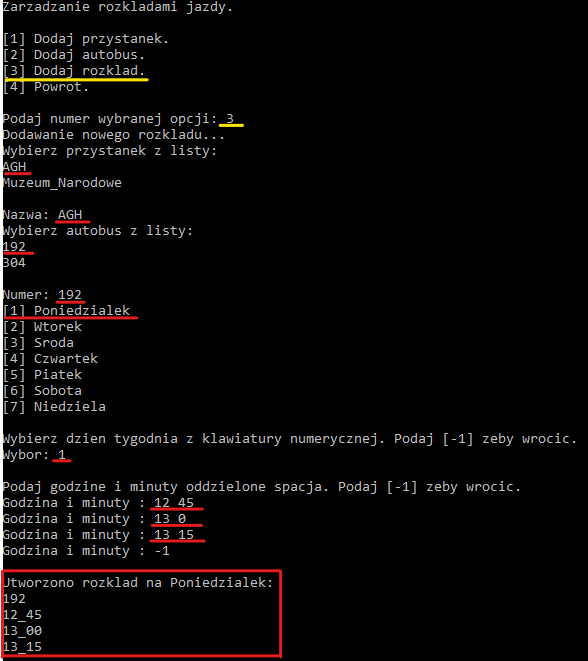
**Żeby zakończyć proces należy wpisać -1.**

****

#### Opcja **[3] Dodaj rozkład.**

Jeżeli lista zapisanych przystanków i autobusów jest niepusta, to możliwy jest:

* wybór przystanku (tekst bez spacji),
* wybór przypisanego mu autobusu (liczba od 1 do 1000),
* wybór dnia tygodnia (liczba od 1 do 7),
* dodanie godzin w formacie godzina minuta (liczba godzin od 0 do 23, liczba minut od 0 do 60).

**Żeby zakończyć proces należy wpisać -1.  
**

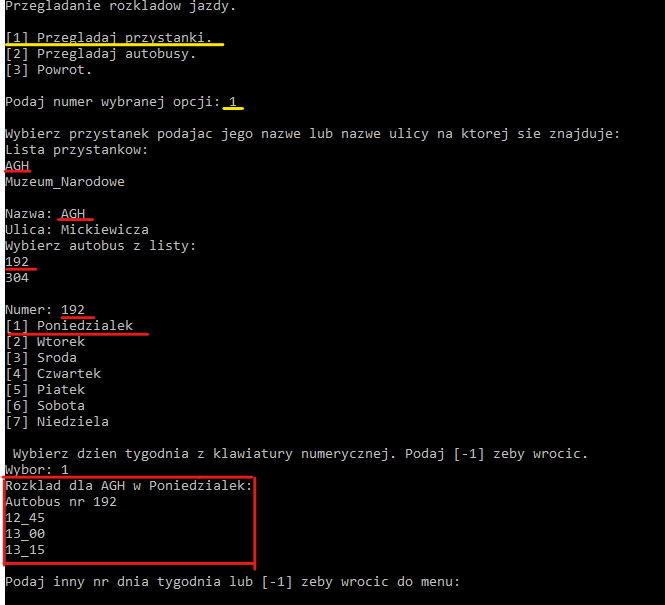
### Przeglądanie rozkładów jazdy

#### Opcja **[1] Przeglądaj przystanki.**

Po wybraniu tej opcji można wybrać rozkład, wybierając po kolei z wyświetlanych list:

* nazwę przystanku (tekst bez spacji) ,
  + alternatywnie można też wyszukać przystanek wpisując nazwę ulicy, na której on się znajduje, np. dla wpisanie “Mickiewicza” spowodowałoby wybranie przystanku “AGH”, który w systemie jest zapisany jako przystanek znajdujący się na tej ulicy.
* numer jednego z autobusów, które są przypisane do wybranego przystanku (liczba od 1 do 1000),
* dzień tygodnia (liczba od 1 do 7).

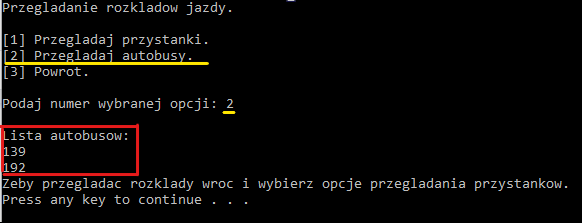
**Żeby zakończyć proces należy wpisać -1.**

****

#### Opcja **[2] Przeglądaj autobusy.**

Wybrana opcja powoduje wyświetlenie listy zapisanych w systemie autobusów. Nie ma możliwości wyszukiwania przystanków według autobusów.

**Żeby wrócić należy wcisnąć dowolny klawisz.**



1. **Kompilacja**

Żeby uruchomić program wystarczy standardowa kompilacja.

Projekt był testowany w systemie Windows 10, ale powinien działać również na innych systemach.

1. **Pliki źródłowe**

Projekt składa się z następujących plików źródłowych:

* *P67\_KP\_bus.cpp* - deklaracja oraz implementacja: funkcji main, klas x, Menu oraz y

1. **Zależności**

Brak.

1. **Opis klas**

W projekcie utworzono 6 klas:

* Przystanek - reprezentuje przystanek.
  + string m\_nazwa - pole nazwy przystanku,
  + string m\_ulica - pole ulicy - adresu przystanku,
  + vector<int> m\_bus\_list - pole na vector, który przechowuje listę numerów autobusów przypisanych do obiektu,
  + string get\_nazwa(void) - getter dla pola nazwy,
  + void set\_nazwa(string nazwa) - setter dla pola nazwy,
  + string get\_ulica(void) - getter dla pola ulicy,
  + void set\_ulica(string ulica) - setter dla pola ulicy,
  + vector<int> get\_bus\_list(void) - getter dla pola listy numerów autobusów,
  + void set\_bus\_list(vector<int> bus\_list) - setter dla pola listy numerów autobusów,
  + void SavePrzystanekTxt() - metoda zapisująca dane przystanku do pliku .txt,
  + void Print(bool showBuses = true) - metoda wypisująca na ekranie dane przystanku, dla parametru showBuses równego false nie wypisuje listy autobusów,
  + void SetValuesFromFile(bool ulica = false)- metoda-setter ustawiający wartości obiektu w klasie na podstawie pliku tekstowego.
* Autobus - reprezentuje autobus.
  + int m\_bus\_nr - pole numeru autobusu,
  + vector<string> m\_przystanek\_list - pole na vector, który przechowuje listę przystanków,
  + int get\_bus\_nr(void) - getter dla pola numeru autobusu,
  + void set\_bus\_nr(int bus\_nr) - setter dla pola numeru autobusu,
  + vector<string> get\_przystanek\_list(void) - getter dla pola listy przystanków,
  + void set\_przystanek\_list(vector<string> przystanek\_list) - setter dla pola listy przystanków,
  + void SaveAutobusTxt() - metoda zapisująca dane autobusu do pliku .txt,
  + void Print(bool showPrzystanki = true) - metoda wypisująca na ekranie dane autobusu
  + void SetValuesFromFile() - metoda-setter ustawiający wartości obiektu w klasie na podstawie pliku tekstowego.
* Rozklad - reprezentuje rozkład jazdy autobusowej.
  + map <string, vector<pair<string, string>> > m\_rozklad - pole-kontener map, który przyporządkowuje vector z rozkładem jazdy dla danego dnia tygodnia(string),
  + int m\_bus\_nr - pole na numer autobusu,
  + string m\_bus\_stop - pole na nazwę przystanku,
  + void set\_bus\_nr(int bus\_nr) - setter dla pola numeru autobusu,
  + int get\_bus\_nr(void) - getter dla pola numeru autobusu,
  + void set\_bus\_stop(string bus\_stop) - setter dla pola nazwy przystanku,
  + string get\_bus\_stop(void) - getter dla pola nazwy przystanku,
  + vector<pair<string, string>> get\_rozklad\_dzien(int dzien\_tyg\_nr) - setter dla elementu pola map z rozkładem(vector z listą godzin),
  + void set\_rozklad\_dzien(vector<pair<string, string>> rozklad\_dzien, int dzien\_tyg\_nr) - getter dla elementu pola map z rozkładem(vector z listą godzin),
  + void SaveRozkladDzienTxt() - metoda zapisująca rozkład do pliku .txt,
  + string SelectDay(int dzien\_tyg\_nr) - metoda zwracająca nazwę dnia tygodnia jako tekst, na podstawie odpowiedniego parametru liczbowego int,
  + void NewHoursForDay(int dzien\_tyg\_nr, string godzina, string minuta) - metoda-setter dzięki której można dodać nowe godziny do vectora z rozkladem dla danego dnia tygodnia.
* Manager - reprezentuje możliwości zarządzania rozkładami jazdy.
  + Przystanek m\_przystanek - pole reprezentujące obiekt-przystanek,
  + Autobus m\_autobus - pole reprezentujące obiekt-autobus,
  + Rozklad m\_rozklad - pole reprezentujące obiekt-rozkład,
  + vector<string> m\_all\_przystanek\_list - pole z listą wszystkich zapisanych przystanków,
  + bool m\_stopEdit - pole-flaga przechowująca informację czy przerwać tryb edycji w programie,
  + bool get\_stopEdit(void) - getter dla pola stopEdit,
  + void set\_stopEdit(bool stopEdit) - setter dla pola stopEdit,
  + void AddPrzystanek() - metoda zapisująca nowy przystanek w programie,
  + void AddBus() - metoda zapisująca nowy autobus w programie,
  + void AddRozklad() - metoda zapisująca nowy rozkład w programie,
  + void Edit() - metoda umożliwiająca edycję-zarządzanie rozkładami jazdy w programie.
* Browser - reprezentuje przeglądarkę zapisanych rozkładów jazdy.
  + Rozklad m\_rozklad - pole reprezentujące obiekt klasy Rozklad,
  + bool m\_stopBrowse - pole-flaga czy zamknąć tryb przeglądania,
  + bool get\_stopBrowse(void) - getter dla pola stopBrowse,
  + void set\_stopBrowse(bool stopBrowse) - setter dla pola stopBrowse,
  + void LoadPrzystanki() - metoda załadowująca obiekty klasy Przystanek z pliku .txt,
  + void LoadAutobusy() - metoda wyświetlająca listę autobusów,
  + void Show() - wyświetlenie menu wyboru w przeglądarce.
* Menu - nawigacja po programie.
  + Manager Menedzer - pole reprezentujące obiekt klasy Manager,
  + Browser Przegladarka - pole reprezentująće obiekt klasy Browser,
  + bool get\_programClosed(void) - getter dla pola programClosed,
  + void set\_programClosed(bool programClosed) - setter dla pola programClosed,
  + void Start() - metoda, która uruchamia wszystkie główne procesy programu,
  + void Update() - metoda aktualizająca program i ekran.
* Inne elementy:
  + int main() - funkcja główna,
  + bool FindElem(string el, vector<string> el\_list) - globalna funkcja wyszukująca podany przez parametr el tekst w podanym przez parametr el\_list vectorze,
  + void PrintVector(vector<string> el\_list) - metoda wypisująca na ekranie zawartość vectora podanego przez parametr el\_list,
  + vector<string> ReadListTxt(string file\_name, int countStart = 1) - metoda zwracająca vector, do którego zostały zapisane elementy z pliku tekstowego, parametr countStart reprezentuje numer linii pliku od której ma być zapisywany vector,
  + void SaveListTxt(string file\_name, string str\_element) - metoda zapisująca nowy element podawany przez parametr str\_element do danego pliku tekstowego.

1. **Zasoby**

W projekcie wykorzystywane są następujące pliki zasobów:

* S\_list.txt - plik zawierający listę utworzonych przystanków w programie. Plik stworzony przez klasę Przystanek.
* Bus\_list.txt - plik zawierający listę utworzonych autobusów w programie. Plik stworzony przez klasę Autobus.
* U\_list.txt - plik zawierający listę ulic przypisanych do utworzonych w programie przystanków. Plik zapisany przez klasę Przystanek.
* S\_nazwa.txt - plik zawierający dane utworzonego w programie przystanku. Plik zapisany przez klasę Przystanek.
  + pierwszy wiersz: nazwa przystanku
  + drugi wiersz: adres - nazwa ulicy
  + od trzeciego wiersza: numery autobusów przypisanych do tego przystanku
* S\_nazwa\_numer\_1.txt, S\_nazwa\_numer\_2.txt, S\_nazwa\_numer\_3.txt, S\_nazwa\_numer\_4.txt, S\_nazwa\_numer\_5.txt, S\_nazwa\_numer\_6.txt, S\_nazwa\_numer\_7.txt - seria 7 plików zawierających rozkład jazdy dla danego autobusu, gdzie numer na końcu odpowiada dniowi tygodnia: poniedziałek - 1, wtorek - 2, środa - 3, czwartek - 4, piątek - 5, sobota - 6, niedziela - 7. Pliki są tworzone przez klasę Rozklad.
  + pierwszy wiersz: numer autobusu
  + od drugiego wiersza: godziny
* Bus\_numer.txt - plik zawierający dane utworzonego w programie autobusu. Zapisany przez klasę Autobus.
  + pierwszy wiersz: numer autobusu
  + od drugiego wiersza: nazwy przystanków przypisanych do trasy tego autobusu

1. **Dalszy rozwój i ulepszenia**

Projekt można rozwinąć w następujący sposób:

* **Wyświetlanie rozkładów,**
  + możliwość wyświetlenia rozkładów dla wszystkich dni na raz na ekranie, coś w rodzaju arkusza, tabeli.
* **Automatyczne tworzenie obiektów,**
  + zautomatyzowane tworzenie przystanku kiedy stworzymy autobus z przypisanym przystankiem o takiej samej nazwie
  + zautomatyzowane tworzenie autobusu kiedy stworzymy przystanek z przypisanym autobusem o takim samym numerze.
* **Wyszukiwanie rozkładu za pomocą numeru autobusu,**
* **Dodatkowe częściowe czyszczenie ekranu z pośrednich procesów.**

1. **Inne**

Brak