Bogusław Malewski
PROI Projekt 1
Temat: Samochód

Rozdział 1 - Wprowadzenie

Program składa się z klasy głównej
samochod
i 5 klas pomocniczych które wchodzą w skład klasy głównej:
kierowca
pasazer
bagażnik
bak
silnik

Każda klasa posiada swój plik .cpp i .h

W programie są jeszcze 3 pary plików .cpp i .h:
DEBUG
DEMO
Funkcje_pomocnicze

Jest jeszcze plik main.cpp zawierający funkcję main programu.

Rozdział 2 – Opis interfejsu klas

```
class kierowca
{
private:
    string imie;
    string nazwisko;
    int wiek = 0;
    bool prawo_jazdy=false;

public:

    int in_kierowca();//wprowadzenie danych do klasy
    int out_kierowca();//wypisywanie danych
    bool czy_prawo_jazdy();//zwraca true jeśli kierowca ma
prawo jazdy
    bool operator++();//zwiększa wiek kierowcy o 1
    bool operator--();//zmniejsza wiek kierowcy o 1
    bool operator<=(kierowca kier1);//porównuje wieki
kierowców
    bool operator<(kierowca kier1);//porównuje wieki kierowców
    bool operator>=(kierowca kier1);//porównuje wieki
kierowców
    bool operator>(kierowca kier1);//porównuje wieki kierowców
    bool operator==(kierowca kier1);//porównuje wieki
kierowców
    friend ostream & operator<<(ostream & os,const kierowca
&kier1);//wypisywanie danych klasy
    friend istream & operator>>(istream & is, kierowca
&kier1);//wprowadzenie danych do klasy
};
```

```
class pasazer
{
private:
    bool obecny = false;
    string imie;
    string nazwisko;
    int wiek = 0;

public:

    int in_pasazer(); //wprowadza dane do klasy
    int out_pasazer(); //wypisuje dane z klasy
    int usun_pasazer();//usuwa pasazera
    bool operator++(); //zwiększa wiek pasażera
    bool operator--(); //zmniejsza wiek pasażera
    bool operator<=(pasazer pas1); //porównuje wieki pasażerów
    bool operator<(pasazer pas1); //porównuje wieki pasażerów
    bool operator>=(pasazer pas1); //porównuje wieki pasażerów
    bool operator>(pasazer pas1); //porównuje wieki pasażerów
    bool operator==(pasazer pas1); //porównuje wieki pasażerów
    friend ostream & operator<<(ostream & os, const pasazer
&pas1); //wypisuje dane z klasy
    friend istream & operator>>(istream & is, pasazer &pas1);
//wprowadza dane do klasy

};
```

```
class bak
{
private:

    enum skrajne_wartosci
    {
        pusty = 0, pelny = 50
    };
    double stan_bak = 25;


public:

    int zatankuj(double litry);//wlewa do baku paliwo
    double spal(double litry);//spala paliwo z baku
    double info_bak();//wypisuje i zwraca ile jest litrów
paliwa w baku
    bool operator++();//dodaje litr paliwa do baku
    bool operator--();//usuwa litr paliwa z baku
    bool operator<=(bak bak1);//porównuje ilość paliwa w
bakach
    bool operator<(bak bak1);//porównuje ilość paliwa w bakach
    bool operator>=(bak bak1);//porównuje ilość paliwa w
bakach
    bool operator>(bak bak1);//porównuje ilość paliwa w bakach
    bool operator==(bak bak1);//porównuje ilość paliwa w
bakach
    const bak & operator+(bak bak1);//dodaje ilość paliwa z
baku drugiego do pierwszego UWAGA modyfikuje tylko lewy
operand
    const bak & operator+(double litry);//dodaje podaną ilość
paliwa do pierwszego baku UWAGA modyfikuje tylko lewy operand
    const bak & operator-(bak bak1);//odejmuje ilość paliwa z
baku drugiego do pierwszego UWAGA modyfikuje tylko lewy
operand
    const bak & operator-(double litry);//odejmuje ilość
paliwa z baku drugiego do pierwszego UWAGA modyfikuje tylko
lewy operand
    friend ostream & operator<<(ostream & os, bak
&bak1);//wlewa do baku paliwo
    friend istream & operator>>(istream & is, bak
&bak1);//spala paliwo z baku

};
```

```
class bagaznik
{
private:
    bool otwarty = false;
    bool zaladowany = false;
public:
    int otworz(); //otwiera bagażnik
    int zamknij(); //zamyka bagażnik
    int zaladuj(); //ładuje bagażnik
    int rozladuj(); //rozładowuje bagażnik
    bool czy_otwarty(); //zwraca true jeśli otwarty bagażnik
    friend ostream & operator<<(ostream & os, bagaznik &bag1);
//wypisuje stan bagażnika
};
```

```
class silnik
{
private:
    bool wlaczony = false;
public:
    int wlacz();//włącza silnik
    int wylacz();//wyłącza silnik
    bool operator++();//włącza silnik
    bool operator--();//wyłącza silnik
    friend ostream & operator<<(ostream & os,const silnik
&silnik1);//wypisuje stan silnika
};
```

```
class samochod
{
private:
    string marka;
    string model;
    int rocznik = 0;
    int miejsca_dla_pasazerow = 0;
    double spalanie_na_100km = 9;
    kierowca * kierowca_1 = nullptr;
    pasazer * pasazerowie = nullptr;
    bagaznik bagaznik_1;
    bak bak_1;
    silnik silnik_1;
public:
    samochod(); //konstruktor
    ~samochod(); //destruktor
    samochod(const samochod & sam1); //konstruktor kopiujący
    samochod &operator=(const samochod & sam1); //operator
przypisania
    int in_info(); //wpisuje podstawowe dane o samochodzie
    int out_info(); //wypisuje podstawowe dane o samochodzie

    int in_kierowca(); //wprowadza kierowcę do auta
    int out_kierowca(); //wypisuje informacje o kierowcy
    int usun_kierowca(); //usuwa kierowcę z auta
    bool out_prawo_jazdy(); //zwraca true jeśli kierowca ma
prawo jazdy

    int in_pasazer(int miejsce); //wprowadza pasażera na dane
miejsce
    int out_pasazer(int miejsce); //wypisuje dane pasażera z
danego miejsca
    int out_all_pasazer(); //wypisuje dane wszystkich
pasażerów z auta
    int usun_pasazer(int miejsce); //usuwa pasażera z danego
miejsca
    int usun_all_pasazer();//usuwa wszystkich pasażerów z auta

    int open_bagaznik(); //otwiera bagażnik
    int close_bagaznik(); //zamyka bagażnik
    int zaladuj_bagaznik(); //ładuje bagażnik
    int rozladuj_bagaznik(); //rozładowuje bagżnik

    int in_bak(); //wpisuje ilość paliwa
    int out_bak(); //zwraca i wypisuje ilość paliwa w baku

    int on_silnik(); //włącza silnik
    int off_silnik(); //wyłącza silnik

    int przejedz(double kilometry); //przejeżdża daną liczbę
kilometrów
```

```
    bool operator++(); //włącza silnik
    bool operator--(); //wyłącza silnik
    friend ostream & operator<<(ostream & os, samochod
&samochod1); //wypisuje podstawowe informacje
};
```

Rozdział 3 – Funkcja pomocnicza

Na potrzeby programu stworzyłem odporną na nieprawidłowe dane funkcję wczytującą liczby double. Funkcja ta znajduję się w pliku funkcje_pomocnicze.cpp.

Oto jak zostałą zaimplementowana ta funkcja:

```
bool wczytaj_double(double &liczba) {
    string temp_str;
    getline(cin, temp_str);
    if (temp_str.length() == 0) {
        cout << "Podano nieprawidłową liczbę" << endl;
        return false;
    }
    if (temp_str.length() == 1 and !isdigit(temp_str[0])) {
        cout << "Podano nieprawidłową liczbę" << endl;
        return false;
    }
    if (!isdigit(temp_str[0])) {
        if (temp_str[0] != '-') {
            if (temp_str[0] != '+') {
                cout << "Podano nieprawidłową liczbę" << endl;
                return false;
            }
        }
    }
    if (temp_str[0] == '-' or temp_str[0] == '+') {
        if (!isdigit(temp_str[1])) {
            cout << "Podano nieprawidłową liczbę" << endl;
            return false;
        }
    }

    liczba = stod(temp_str);

    return true;
};
```

Rozdział 4 – Plik DEBUG

Plik DEBUG.cpp zawiera funkcje pozwalające testować wszystkie klasy jak i funkcję uruchamiającą interfejs użytkownika pozwalający na wybór klasy do testów użytkownikowi.

```
int wybor_testu();
int testuj_all();
int testuj_kierowca();
int testuj_pasazer();
int testuj_bagaznik();
int testuj_bak();
int testuj_silnik();
int testuj_samochod();
```

Rozdział 5 – Plik DEMO

Plik demo.cpp zawiera funkcję:
```
int demo_taxi();
```

która przeprowadza symulację taksówki.
Taksówka zabiera 2 pasażerów i ich bagaże, przejeżdża pewien dystans po czym taksówka się zatrzymuję by zatankować, po czym rusza dalej w podróż, następnie pasażerowie wysiadają i ich bagaż jest rozładowywany

Rozdiał 6 – Plik main

W pliku main jest kompilowany kod funkcji demonstrującej możliwości klasy samochod lub kod umożlwiający testowanie wybranej przez siebie klasy w zależności od tego czy jest zdefiniowana zmienna kompilacji DEBUG czy DEMO.