C++ - Samochody (obiektowe)

Ogólne

Program prezentuje atrybuty samochodów i sprawdza, który z nich jest ekologiczny (spala najmniej paliwa).

Uwaga: Nigdy nie wolno kopiować kodu z PDF-a, gdyż zawiera on niewidoczne znaki końca linii, twarde odstępy, odmienne cudzysłowy, odmienne apostrofy, odmienne myślniki. To godziny dodatkowej pracy na wykrywanie błędów i ich poprawianie. Kod należy przepisać ze zrozumieniem.

Kod - wersja podstawowa (*Windows*)

```
#include <iostream>

using namespace std;

class Samochod {
   public:
      string marka_samochodu;
      string model_samochodu;
      string kolor_samochodu;
      float spalanie_miasto;
      float spalanie_autostrada;

      // Konstruktor, czyli metoda inicjalizująca zmienne / atrybuty:
      Samochod(string marka, string model, string kolor, float po_miescie, float
      na_autostradzie) {
         this->marka_samochodu = marka;
         this->model_samochodu = model;
         this->kolor_samochodu = kolor;
         this->spalanie_miasto = po_miescie;
         this->spalanie_autostrada = na_autostradzie;
      }

      void czy_ekologiczny() {
         cout << "\n\nDane samochodu: " << marka_samochodu << " " << model_samochodu
         << " " << kolor_samochodu << "." << endl;
         if ((spalanie_miasto <= 6.0) && (spalanie_autostrada <= 8.0)) {
            cout << "Rewelacja! Ten model jest ekologiczny zarówno w mieście, jak i na
            autostradzie!" << endl;
         }
          else if ((spalanie_miasto <= 6.0) && (spalanie_autostrada > 8.0)) {
            cout << "Ten model jest idealny na jazdę po mieście. Mało spala!" << endl;
         }
          else if ((spalanie_miasto > 6.0) && (spalanie_autostrada <= 8.0)) {
```

```
            cout << "Ten model jest ekologiczny podczas dużych prędkości na autostradzie!"
            << endl;
        }
        else if ((spalanie_miasto > 6.0) && (spalanie_autostrada > 8.0)) {
            cout << "Tragedia! Ten model NIE jest ekologiczny ani podczas jazdy w mieście, ani
            na autostradzie. Ale blachary na to lecą!" << endl;
        }
    }
};

int main() {
    cout << "\n===== EKO SAMOCHODY =====" << endl;
    Samochod s1("Volvo", "XC90", "Czerwony", 2.5, 14.0);
    Samochod s2("Peugeot", "308", "Czarny", 3.0, 5.9);
    Samochod s3("Toyota", "Land Cruiser", "Niebieski", 11.90, 18.20);
    Samochod s4("Seat", "Ibiza", "Czerwony", 9.70, 8.0);
    s1.czy_ekologiczny();
    s2.czy_ekologiczny();
    s3.czy_ekologiczny();
    s4.czy_ekologiczny();

    cout << "\nNaciśnij ENTER, aby zakończyć..." << endl;
    cin.get();
    return 0;
}
```

Kod - wersja rozbudowana (Windows)

```
#include <iostream>
#include <windows.h> // Potrzebne do kolorowania napisów; działa tylko w Windows.
// #include <locale.h> // Biblioteka do znaków Polskich

using namespace std;

class Samochod {
    public:
        // Dlaczego nie deklaruję tych zmiennych / atrybutów w części prywatnej?
        string marka_samochodu;
        string model_samochodu;
        string kolor_samochodu;
        double spalanie_miasto;
        double spalanie_autostrada;

        // Konstruktor, czyli metoda inicjalizująca zmienne / atrybuty:
        Samochod(string marka, string model, string kolor, double po_miescie, double
        na_autostradzie) {
            // Tworzymy zmienne wewnętrzne oparte o parametry. Chcemy przejąć argumenty z
            // funkcji main() i chcemy, aby były one dostępne wewnątrz całej klasy (także wewnątrz
```

```
        // metody czy_ekologiczny())
        // Od tej pory, atrybuty zainicjowane w klasie będą miały wartość argumentów
        // przekazanych do konstruktora.
        this->marka_samochodu = marka;
        this->model_samochodu = model;
        this->kolor_samochodu = kolor;
        this->spalanie_miasto = po_miescie;
        this->spalanie_autostrada = na_autostradzie;
    }

    void czy_ekologiczny() {
        cout << "\n\nDane samochodu: \033[1;36;40m" << marka_samochodu << " " <<
        model_samochodu << " " << kolor_samochodu << "\033[0m." << endl;
        if ((spalanie_miasto <= 6.0) && (spalanie_autostrada <= 8.0)) {
            cout << "Rewelacja! Ten model jest ekologiczny zarówno w mieście, jak i na
            autostradzie!" << endl;
        }
        else if ((spalanie_miasto <= 6.0) && (spalanie_autostrada > 8.0)) {
            cout << "Ten model jest idealny na jazdę po mieście. Mało spala!" << endl;
        }
        else if ((spalanie_miasto > 6.0) && (spalanie_autostrada <= 8.0)) {
            cout << "Ten model jest ekologiczny podczas dużych prędkości na autostradzie!"
            << endl;
        }
        else if ((spalanie_miasto > 6.0) && (spalanie_autostrada > 8.0)) {
            cout << "Tragedia! Ten model NIE jest ekologiczny ani podczas jazdy w mieście, ani
            na autostradzie. Ale blachary na niego lecą!" << endl;
        }
    }
};


int main() {
    // Jeśli Twój Windows obsługuje kolorowe znaki ANSI w Wierszu poleceń, możesz poniższy
    // blok kodu pominąć:
    // ============== KOLOROWE ZNAKI ANSI =============
    // Na początek naprawiamy Windowsa, aby kolorował czcionki w konsoli.
    // Tworzymy tzw. uchwyt do tego, co będzie pojawiać się na konsoli (do bufora konsoli):
    HANDLE konsola = GetStdHandle(STD_OUTPUT_HANDLE);
    // Aktywujemy virtualny terminal i to, co będzie się na nim pojawiać:
    #ifndef ENABLE_VIRTUAL_TERMINAL_PROCESSING // Jeśli nie jest zdefiniowany, to:
    #define ENABLE_VIRTUAL_TERMINAL_PROCESSING 0x0004
    #endif

    // Aby powyższe działało, musimy aktywować i to, co poniżej.
    // Jeśli nie jest zdefiniowany, to:
    #ifndef ENABLE_PROCESSED_OUTPUT
    #define ENABLE_PROCESSED_OUTPUT 0x0001
```

```
#endif

// Wartość trybu (intup lub output). Słowo "dw" to skrót od "Display Window",
// jest to jednak nazwa zmiennej, i może być inna:
DWORD dwMode = 0;
dwMode |= ENABLE_PROCESSED_OUTPUT | ENABLE_VIRTUAL_TERMINAL_PROCESSING;
SetConsoleMode(konsola, dwMode);
// ==================== KONIEC BLOKU ================

string marka_uzytkownika;
string model_uzytkownika;
string kolor_uzytkownika;
double spalanie_miasto_uzytkownika;
double spalanie_autostrada_uzytkownika;
string wartoscUzytkownika;
bool wprowadzonoLiczbe1 = false; // W C++ zmienne logiczne nie mają domyślnej wartości
bool wprowadzonoLiczbe2 = false;

cout << "\n\033[1;34;40m===== EKO SAMOCHODY =====\033[0m" << endl;
// Dzięki konstruktorowi, zamiast robić to tak:
// s1.marka = "Volvo";
// s1.model = "XC90";
// s1.kolor = "Czerwony";
// s1.spalanie_miasto = 2.5;
// s1.spalanie_autostrada = 14.0;
// robimy to tak:
Samochod s1("Volvo", "XC90", "Czerwony", 2.5, 14.0);
Samochod s2("Peugeot", "308", "Czarny", 3.0, 5.9);
Samochod s3("Toyota", "Land Cruiser", "Niebieski", 11.90, 18.20);
Samochod s4("Seat", "Ibiza", "Czerwony", 9.70, 8.0);
s1.czy_ekologiczny();
s2.czy_ekologiczny();
s3.czy_ekologiczny();
s4.czy_ekologiczny();

cout << "\nWprowadź swój własny model samochodu." << endl;
cout << "Podaj markę (np. Opel, Fiat, itp.): ";
cin >> marka_uzytkownika;
cout << "Podaj model: ";
cin >> model_uzytkownika;
cout << "Podaj kolor: ";
cin >> kolor_uzytkownika;

// Uwaga: Przypisanie wartości typu "string" do zmiennej "double", daje 0.
while(!wprowadzonoLiczbe1) {
   cout << "Ile litrów spala w czasie jazdy po mieście (wpisz liczbę): ";
   cin >> wartoscUzytkownika;
   try {
```

```
        // Jeśli użytkownik wprowadził liczbę z przecinkiem, zamieniamy go na kropkę.
        // Typ danych size_t stosowany jest w stosunku do rozmiaru obiektu.
        // Funkcja find() zwraca pozycję pierwszego wystąpienia przecinka.
        size_t pozycjaPrzecinka = wartoscUzytkownika.find(",");
        // npos oznacza brak jakiejkolwiek pozycji zawierającej przecinek.
        // npos w praktyce reprezentuje maksymalną liczbę typu danych size_t, czyli:
        // 18446744073709551615.
        if (pozycjaPrzecinka != string::npos) {
            // Zamiana przecinka na kropkę:
            wartoscUzytkownika.replace(pozycjaPrzecinka, 1, ".");
        }
        // Próbujemy zamienieć stringa na liczbę i przypisać jej typ double.
        // Jeśli się to uda, to znaczy, że wprowadzona wartosc jest typu double.
        spalanie_miasto_uzytkownika = stod(wartoscUzytkownika); // stod = String-to-Double
        wprowadzonoLiczbe1 = true;
    }
    catch (const invalid_argument& exc) {
        cout << "\n\n\033[0;37;41mBŁĄD:\033[0m" << endl;
        cout << "\033[1;31;40mTo nie jest liczba! Spróbuj jeszcze raz.\033[0m\n\n" << endl;
    }
}

while(!wprowadzonoLiczbe2) {
    cout << "Podaj spalanie przy prędkości powyżej 140km/h (wpisz liczbę): ";
    cin >> wartoscUzytkownika;
    try {
        // Jeśli użytkownik wprowadził liczbę z przecinkiem, zamieniamy go na kropkę.
        // Typ danych size_t stosowany jest w stosunku do rozmiaru obiektu.
        // Funkcja find() zwraca pozycję pierwszego wystąpienia przecinka.
        size_t pozycjaPrzecinka = wartoscUzytkownika.find(",");
        // npos oznacza brak jakiejkolwiek pozycji zawierającej przecinek.
        // npos w praktyce reprezentuje maksymalną liczbę typu danych size_t, czyli:
        // 18446744073709551615.
        if (pozycjaPrzecinka != string::npos) {
            // Zamiana przecinka na kropkę:
            wartoscUzytkownika.replace(pozycjaPrzecinka, 1, ".");
        }
        // Próbujemy zamienieć stringa na liczbę i przypisać jej typ double.
        // Jeśli się to uda, to znaczy, że wprowadzona wartosc jest typu double.
        spalanie_autostrada_uzytkownika = stod(wartoscUzytkownika); // stod = String-to-
                                                                    // Double
        wprowadzonoLiczbe2 = true;
    }
     catch (const invalid_argument& exc) {
        cout << "\n\n\033[0;37;41mBŁĄD:\033[0m" << endl;
        cout << "\033[1;31;40mTo nie jest liczba! Spróbuj jeszcze raz.\033[0m\n\n" << endl;
    }
}
```

```
    Samochod samochod_uzytkownika(marka_uzytkownika, model_uzytkownika,
    kolor_uzytkownika, spalanie_miasto_uzytkownika, spalanie_autostrada_uzytkownika);
    samochod_uzytkownika.czy_ekologiczny();
     cout << "\033[1;30;40mW mieście spala: \033[0m" << spalanie_miasto_uzytkownika << "\
033[1;30;40m l na sto kilometrów.\033[0m" << endl;
    cout << "\033[1;30;40mNa autostradzie spala: \033[0m" <<
    spalanie_autostrada_uzytkownika << "\033[1;30;40m l na sto kilometrów.\033[0m" << endl;

    cout << "\nNaciśnij ENTER, aby zakończyć..." << endl;
    cin.ignore(256,'\n');   // Usuwa z bufora znak \n (w ilości 256) pozostawiony przez
                            // wcześniejsze "cin" ("cin" domyślnie zostawia taki znak).
    cin.get();
    return 0;
}
```

Kod - wersja rozbudowana (Linux)

```
#include <iostream>

using namespace std;

class Samochod {
    public:
        // Dlaczego nie deklaruję tych zmiennych / atrybutów w części prywatnej?
        string marka_samochodu;
        string model_samochodu;
        string kolor_samochodu;
        double spalanie_miasto;
        double spalanie_autostrada;

        // Konstruktor, czyli metoda inicjalizująca zmienne / atrybuty:
        Samochod(string marka, string model, string kolor, double po_miescie, double
        na_autostradzie) {
            // Tworzymy zmienne wewnętrzne oparte o parametry. Chcemy przejąć argumenty z
            // funkcji main() i chcemy, aby były one dostępne wewnątrz całej klasy (także wewnątrz
             // metody czy_ekologiczny())
            // Od tej pory, atrybuty zainicjowane w klasie będą miały wartość argumentów
            // przekazanych do konstruktora.
            this->marka_samochodu = marka;
            this->model_samochodu = model;
            this->kolor_samochodu = kolor;
            this->spalanie_miasto = po_miescie;
            this->spalanie_autostrada = na_autostradzie;
        }

        void czy_ekologiczny() {
            cout << "\n\nDane samochodu: \033[1;36;40m" << marka_samochodu << " " <<
```

```
        model_samochodu << " " << kolor_samochodu << "\033[0m." << endl;
        if ((spalanie_miasto <= 6.0) && (spalanie_autostrada <= 8.0)) {
            cout << "Rewelacja! Ten model jest ekologiczny zarówno w mieście, jak i na
            autostradzie!" << endl;
        }
         else if ((spalanie_miasto <= 6.0) && (spalanie_autostrada > 8.0)) {
            cout << "Ten model jest idealny na jazdę po mieście. Mało spala!" << endl;
        }
         else if ((spalanie_miasto > 6.0) && (spalanie_autostrada <= 8.0)) {
            cout << "Ten model jest ekologiczny podczas dużych prędkości na autostradzie!"
            << endl;
        }
         else if ((spalanie_miasto > 6.0) && (spalanie_autostrada > 8.0)) {
            cout << "Tragedia! Ten model NIE jest ekologiczny ani podczas jazdy w mieście, ani
            na autostradzie. Ale blachary na niego lecą!" << endl;
        }
    }
};


int main() {
    string marka_uzytkownika;
    string model_uzytkownika;
    string kolor_uzytkownika;
    double spalanie_miasto_uzytkownika;
    double spalanie_autostrada_uzytkownika;
    string wartoscUzytkownika;
    bool wprowadzonoLiczbe1 = false; // W C++ zmienne logiczne nie mają domyślnej wartości
    bool wprowadzonoLiczbe2 = false;

    cout << "\n\033[1;34;40m===== EKO SAMOCHODY =====\033[0m" << endl;
    // Dzięki konstruktorowi, zamiast robić to tak:
    // s1.marka = "Volvo";
    // s1.model = "XC90";
    // s1.kolor = "Czerwony";
    // s1.spalanie_miasto = 2.5;
    // s1.spalanie_autostrada = 14.0;
    // robimy to tak:
    Samochod s1("Volvo", "XC90", "Czerwony", 2.5, 14.0);
    Samochod s2("Peugeot", "308", "Czarny", 3.0, 5.9);
    Samochod s3("Toyota", "Land Cruiser", "Niebieski", 11.90, 18.20);
    Samochod s4("Seat", "Ibiza", "Czerwony", 9.70, 8.0);
    s1.czy_ekologiczny();
    s2.czy_ekologiczny();
    s3.czy_ekologiczny();
    s4.czy_ekologiczny();

    cout << "\nWprowadź swój własny model samochodu." << endl;
```

```
cout << "Podaj markę (np. Opel, Fiat, itp.): ";
cin >> marka_uzytkownika;
cout << "Podaj model: ";
cin >> model_uzytkownika;
cout << "Podaj kolor: ";
cin >> kolor_uzytkownika;

// Uwaga: Przypisanie wartości typu "string" do zmiennej "double", daje 0.
while(!wprowadzonoLiczbe1) {
   cout << "Ile litrów spala w czasie jazdy po mieście (wpisz liczbę): ";
   cin >> wartoscUzytkownika;
   try {
      // Jeśli użytkownik wprowadził liczbę z przecinkiem, zamieniamy go na kropkę.
      // Typ danych size_t stosowany jest w stosunku do rozmiaru obiektu.
      // Funkcja find() zwraca pozycję pierwszego wystąpienia przecinka.
      size_t pozycjaPrzecinka = wartoscUzytkownika.find(",");
      // npos oznacza brak jakiejkolwiek pozycji zawierającej przecinek.
      // npos w praktyce reprezentuje maksymalną liczbę typu danych size_t, czyli:
      // 18446744073709551615.
      if (pozycjaPrzecinka != string::npos) {
         // Zamiana przecinka na kropkę:
         wartoscUzytkownika.replace(pozycjaPrzecinka, 1, ".");
      }
      // Próbujemy zamienieć stringa na liczbę i przypisać jej typ double.
      // Jeśli się to uda, to znaczy, że wprowadzona wartosc jest typu double.
      spalanie_miasto_uzytkownika = stod(wartoscUzytkownika); // stod = String-to-Double
      wprowadzonoLiczbe1 = true;
   }
   catch (const invalid_argument& exc) {
      cout << "\n\n\033[0;37;41mBŁĄD:\033[0m" << endl;
      cout << "\033[1;31;40mTo nie jest liczba! Spróbuj jeszcze raz.\033[0m\n\n" << endl;
   }
}

while(!wprowadzonoLiczbe2) {
   cout << "Podaj spalanie przy prędkości powyżej 140km/h (wpisz liczbę): ";
   cin >> wartoscUzytkownika;
   try {
      // Jeśli użytkownik wprowadził liczbę z przecinkiem, zamieniamy go na kropkę.
      // Typ danych size_t stosowany jest w stosunku do rozmiaru obiektu.
      // Funkcja find() zwraca pozycję pierwszego wystąpienia przecinka.
      size_t pozycjaPrzecinka = wartoscUzytkownika.find(",");
      // npos oznacza brak jakiejkolwiek pozycji zawierającej przecinek.
      // npos w praktyce reprezentuje maksymalną liczbę typu danych size_t, czyli:
      // 18446744073709551615.
      if (pozycjaPrzecinka != string::npos) {
         // Zamiana przecinka na kropkę:
         wartoscUzytkownika.replace(pozycjaPrzecinka, 1, ".");
```

```
            }
            // Próbujemy zamienieć stringa na liczbę i przypisać jej typ double.
            // Jeśli się to uda, to znaczy, że wprowadzona wartosc jest typu double.
            spalanie_autostrada_uzytkownika = stod(wartoscUzytkownika); // stod = String-to-
                                                                        // Double
            wprowadzonoLiczbe2 = true;
        }
         catch (const invalid_argument& exc) {
            cout << "\n\n\033[0;37;41mBŁĄD:\033[0m" << endl;
            cout << "\033[1;31;40mTo nie jest liczba! Spróbuj jeszcze raz.\033[0m\n\n" << endl;
        }
    }

    Samochod samochod_uzytkownika(marka_uzytkownika, model_uzytkownika,
    kolor_uzytkownika, spalanie_miasto_uzytkownika, spalanie_autostrada_uzytkownika);
    samochod_uzytkownika.czy_ekologiczny();
     cout << "\033[1;30;40mW mieście spala: \033[0m" << spalanie_miasto_uzytkownika << "\
033[1;30;40m l na sto kilometrów.\033[0m" << endl;
    cout << "\033[1;30;40mNa autostradzie spala: \033[0m" <<
    spalanie_autostrada_uzytkownika << "\033[1;30;40m l na sto kilometrów.\033[0m" << endl;

    cout << "\nNaciśnij ENTER, aby zakończyć..." << endl;
    cin.ignore(256,'\n');   // Usuwa z bufora znak \n (w ilości 256) pozostawiony przez
                            // wcześniejsze "cin" ("cin" domyślnie zostawia taki znak).
    cin.get();
    return 0;
}
```

Ostatnia aktualizacja: 4 stycznia 2024.