

---

## *Programowanie obiektowe*

---

### Zajęcia 4. Przeciążanie operatorów

#### Zadanie 1. Projekt przeciążanie operatorów

1. Utwórz w wybranym miejscu na dysku folder **OperatorCpp**.
2. Utwórz klasę **Samochod**, która będzie zawierać informacje o pojemności baku (float), poziomie paliwa (float), liczbie przejechanych kilometrów (unsigned int), model (string)

```
1 class Samochod{  
2 private:  
3     float pojemnosc_baku;  
4     float poziom_paliwa;  
5     unsigned int liczba_kilometrow;  
6     string model;  
7 };  
8
```

3. Stwórz konstruktor bezparametrowy ustawiający domyślne wartości parametrów oraz konstruktor przyjmujący jako argument model samochodu oraz pojemność baku.
4. Stwórz kilka metod dla tej klasy.
5. Przeciąż operatory `bool()` oraz `!` aby obiekt zwracał `true` tylko wtedy, gdy model nie jest pustym stringiem oraz pojemność baku jest większa od 0.

```
1 operator bool () const {  
2     ...  
3 }  
4  
5 bool operator !() const {  
6     ...  
7 }
```

6. Przetestuj kod:

```
1 Samochod s1(123, "audi");  
2 Samochod s2(0, "bmw");  
3 Samochod s3(200, "");  
4 if(s1){  
5     ...  
6 }  
7 if(!s1){  
8     ...  
9 }  
10 ...
```

7. Przeciąż operator << tak, aby cout wypisywało model oraz poziom paliwa. (UWAGA, jeśli nie zastosowano w kodzie using namespace std należy przed składowymi biblioteki standardowej dodać std::)

```
1 friend ostream & operator << (ostream &os, const Samochod &  
    samochod) {  
2     os << samochod.model <<" " <<samochod.poziom_paliwa<<"\n";  
3     return os;  
4 }
```

8. Powyższy kod wewnątrz klasy używa słowa kluczowe friend, napisz krótki komentarz dlaczego.

9. Przeciąż operator < tak, aby podczas użycia funkcji sort samochody zostały posortowane leksykograficznie.

```
1 bool operator < (const Samochod &samochod) const {  
2     return this->model < samochod.model ;  
3 }
```

10. Dlaczego użyto słowa kluczowego const za funkcją? Napisz krótki komentarz.

11. Przetestuj powyższy kod.

```
1 #include <iostream>
2 #include <algorithm>
3 #include <vector>
4 using namespace std;
5
6 ...
7
8 vector <Samochod> v;
9 v.push_back(s1);
10 v.push_back(s2);
11 v.push_back(s3);
12 sort(v.begin(), v.end());
13 for(auto it=v.begin(); it!=v.end(); ++it){
14     cout<<*it<<endl;
15 }
```

12. Stwórz strukturę służącą do porównywania obiektów w zbiorze `std::set`.

```
1 struct cmp {
2     bool operator() (const Samochod &a, const Samochod &b)
3     const {
4         return a.odczytaj_pojemnosc_baku() < b.
5         odczytaj_pojemnosc_baku();
6     }
7 };
```

13. Przetestuj kod.

```
1 set <Samochod, cmp> s;
2 s.insert(s1);
3 s.insert(s2);
4 s.insert(s3);
5 for(auto it=s.begin(); it!=s.end(); ++it){
6     cout<<*it<<endl;
7 }
```

14. Zapoznaj się z innymi sposobami na rozwiązanie tego problemu (wykorzystującymi możliwości z nowszych standardów języka C++).

<https://stackoverflow.com/questions/2620862/using-custom-stdset-comparator/46128321#46128321>

15. Napisz własne przeciążenia dla 3 poniższych operatorów:

++

,

| =