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)

```
class Samochod{
private:
  float pojemnosc_baku;
  float poziom_paliwa;
  unsigned int liczba_kilometrow;
  string model;
};

```

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.

```
operator bool () const {
    ...
}

bool operator !() const {
    ...
}
```

6. Przetestuj kod:

```
Samochod s1(123, "audi");
Samochod s2(0, "bmw");
Samochod s3(200, "");
if(s1){
  ...
}
if(!s1){
  ...
}
...
```

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::)

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

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

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

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

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

11. Przetestuj powyższy kod.

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;

...

vector <Samochod> v;
v.push_back(s1);
v.push_back(s2);
v.push_back(s3);
sort(v.begin(), v.end());
for(auto it=v.begin(); it!=v.end(); ++it){
  cout<<*it<<endl;
}
```

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

```
struct cmp {
    bool operator() (const Samochod &a, const Samochod &b)
    const {
        return a.odczytaj_pojemnosc_baku() < b.
    odczytaj_pojemnosc_baku();
    }
};
```

13. Przetestuj kod.

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

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:

    ++

    ,

    |=