0.1 Określanie interfejsu szablonu

```
template<typename S, typename T>
    requires Sequence<S> &&
    Equality_comparable<Value_type<S>, T>
Iterator_of<S> szukaj(S &seq, const T &value);
```

Powyższy szablon przyjmuje dwa argumenty typu szablonu. Pierwszy argument typu musi być typu `Sequence` i musimy być w stanie porównywać elementy sekwencji ze zmienną `value` używając operatora `==` (stąd `Equality_comparable<Value_type<S>, T>`). Funkcja `szukaj` przyjmuje sekwencję przez referencję i `value` do znalezienia jako referencję `const`. Zwraca iterator.

Sekwencja musi posiadać `begin()` i `end()`. Koncept `Equality_comparable` jest zaproponowany jako koncept standardowej biblioteki. Wymaga by jego argument dostarczał operatory `==` i `!=`. Ten koncept przyjmuje dwa argumenty. Wiele konceptów przyjmuje więcej niż jeden argument. Koncepty mogą opisywać nie tylko typy, ale również związki między typami.

Użycie funkcji `szukaj`:

```
void test(vector<string> &v, list<double> &list){
    auto a0 = szukaj(v, "test");(1)
    auto p1 = szukaj(v, 0.7);(2)
    auto p2 = szukaj(list, 0.7);(3)
    auto p3 = szukaj(list, "test");(4)

    if(a0 != v.end()){
      //Znaleziono "test"
    }
}
```

1) OK 2) Błąd: nie można porównać string do double 3) OK 4) Błąd: nie można porównać double ze string