- Solution

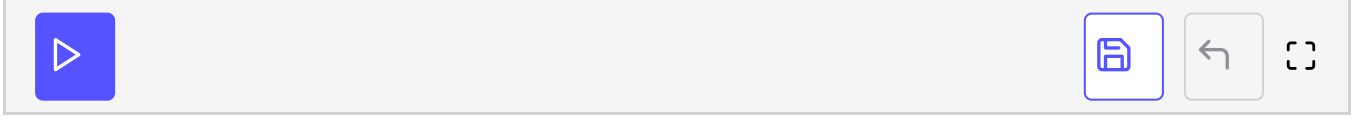
In the lesson, we will take a look at the solution to the tasks in the previous lesson.

WE'LL COVER THE FOLLOWING

Solution

```
//userDefinedLiteralExtended.cpp
#include <iostream>
#include <ostream>

namespace Distance{
  class MyDistance{
    public:
      explicit MyDistance(double i):m(i){}

      friend MyDistance operator +(const MyDistance& a, const MyDistance& b){
        return MyDistance(a.m + b.m);
      }
      friend MyDistance operator -(const MyDistance& a,const MyDistance& b){
        return MyDistance(a.m - b.m);
      }

      friend MyDistance operator*(double m, const MyDistance& a){
        return MyDistance(m*a.m);
      }

      friend std::ostream& operator<< (std::ostream &out, const MyDistance& myDist){
        out << myDist.m << " m";
         return out;
      }
    private:
      double m;

  };

  namespace Unit{
    MyDistance operator "" _mi(long double d){
        return MyDistance(1609.344 *d);
      }
    MyDistance operator "" _km(long double d){
      return MyDistance(1000*d);
```

```
    }
    MyDistance operator "" _m(long double m){
      return MyDistance(m);
    }
    MyDistance operator "" _ft(long double d){
      return MyDistance(0.348*d);
    }
    MyDistance operator "" _dm(long double d){
      return MyDistance(d/10);
    }
    MyDistance operator "" _cm(long double c){
      return MyDistance(c/100);
    }
  }
}

using namespace Distance::Unit;

int main(){

  std:: cout << std::endl;

  std::cout << "1.0_mi: " << 1.0_mi << std::endl;
  std::cout << "1.0_km: " << 1.0_km << std::endl;
  std::cout << "1.0_m: " << 1.0_m << std::endl;
  std::cout << "1.0_ft: " << 1.0_ft << std::endl;
  std::cout << "1.0_dm: " << 1.0_dm << std::endl;
  std::cout << "1.0_cm: " << 1.0_cm << std::endl;

  std::cout << std::endl;

  std::cout << "0.001 * 1.0_km: " << 0.001 * 1.0_km << std::endl;
  std::cout << "10 * 1_dm: " << 10 * 1.0_dm << std::endl;
  std::cout << "100 * 1.0cm: " << 100 * 1.0_cm << std::endl;

  std::cout << std::endl;
  std::cout << "1.0_km + 2.0_dm +  3.0_dm + 4.0_cm: " << 1.0_km + 2.0_dm +  3.0_dm + 4.0_cm <
  std::cout << std::endl;

  Distance::MyDistance work= 63.0_km;
  Distance::MyDistance workPerDay= 2 * work;
  Distance::MyDistance abbrevationToWork= 5400.0_m;
  Distance::MyDistance workout= 2 * 1600.0_m;
  Distance::MyDistance shopping= 2 * 1200.0_m;

  Distance::MyDistance myDistancePerWeek= 4 * workPerDay - 3 * abbrevationToWork + workout +

  std::cout << "4 * workPerDay - 3 * abbrevationToWork + workout + shopping: " << myDistanceP

  std::cout << "\n\n";

}
```



Explanation

- The mile unit is implemented in lines 31-32.

- The feet unit is implemented in lines 40-41.
- The components `work`, `workPerDay`, `workout`, `abbrevationToWork`, and `shopping` are defined in lines 75-79.
- In line 81, we have computed `myDistancePerWeek` using the components we have defined above.

In the next lesson, we will learn about the raw and cooked forms of user-defined literals.