Operator Overloading

- Operators can be implemented for user-defined data types.

- The number of operands depends on two facts
  - Kind of operator
  - Should the operator be a member functionor a function.

```
struct Account{
  Account& operator += (double b){
      balance += b;
      return *this;
  } . . . .
};

Account a;
a += 100.0;
```

Operator Overloading

- Operator as function
  - Requires all arguments
  - Will be declared as (`friend`)

- Rules:
  1. You can not change the precedence of an operator.
  2. Derived Classes inherit all operators with the exception of the assignment operator.
  3. Operators with the exception of the call operator can not have default arguments.
  4. Operators can explicitly be called: `a.operator += (b).`

`operatorOverloading.cpp`

Assignment Operator

- Can be implemented as copy or move assignment operator.
- Has to be a member function.
- The (Copy/Move) assignment operator behave similar to (Copy/Move) constructor.
- The assignment operator will automatically be created if all base classes and all members have an assignment operator.

`operatorOverloadingAssignment.cpp`

Call Operator

- Objects of class, having a call operator can be just as functions.
- These special objects are called function objects.
- Function objects can have in contrast to functions state.
- Function objects are quite similar to lambda functions.

Function objects are often used in algorithms of the Standard Template Library.

operatorOverloadingCall.cpp