BTH001: Assignment 1

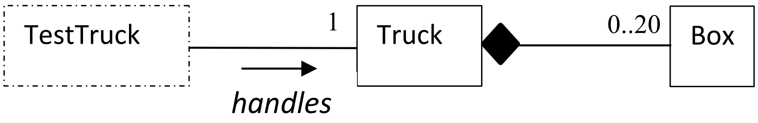
This assignment focuses on:

- Composition
- Deep copying and dynamic memory allocation
  - copy constructor
  - assignment operator
  - destructor
- Operator overloading

Loading boxes on a Truck

Your task is to write an object-oriented program for handling the loading of boxes on a truck. What needs to be fulfilled concerning the Truck and Box is covered in the TestTruck part (the file TestTruck.cpp is already implemented).

Your part of the program (class Box and class Truck) shall be designed and implemented according to the descriptions and the class sketch below.



The class declaration for the class Box is

```
class Box
{
private:
		double length;
		double width;
		double height;
		double weight;
public:
		Box();
		Box(double length, double width, double height, double weight);
		virtual ~Box();

		double volume() const;
		double getWeight() const;
		bool operator==(const Box& otherBox) const;
};
```

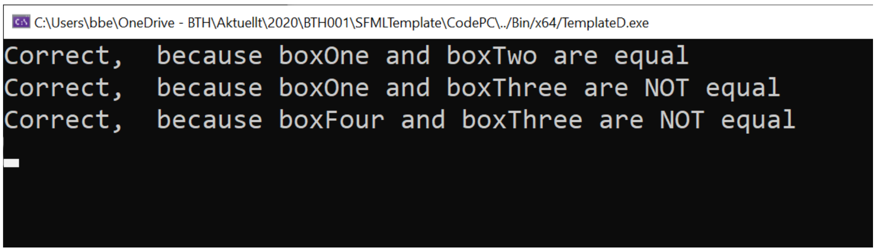
For the class Box your task is to define the constructor and the member functions including the operator ==. This is done in a file named Box.cpp.

volume(): calculates the volume and returns the result.

getWeight(): returns the weight.

operator==(…): returns true or false based on the value of all member variables.

When you have defined the class Box, run the program TestBox.cpp which will result in the output


C:\Users\bbe\OneDrive - BTH\Aktuellt\2020\BTH001\SFMLTemplate\CodePC\../Bin/x64/TemplateD.exe

```
Correct,  because boxOne and boxTwo are equal
Correct,  because boxOne and boxThree are NOT equal
Correct,  because boxFour and boxThree are NOT equal
```

The class declaration for the class Truck is

```
class Truck
{
public:
            static const int CAPACITY = 20;
            static const int MAX_WEIGHT = 2000;
private:
            Box* boxes[CAPACITY];
            int currentNrOfBoxes;
            double currentWeight;
public:
            Truck();
            virtual ~Truck();

            bool loadBox(double length, double width, double height, double weight);
            void removeBoxAt(int index);
            void removeAllBoxesWithWeightAbove(double theWeight);
            double getCurrentWeight() const;
            int getCurrentNrOfBoxes() const;
            double totalVolumeOfCurrentBoxes() const;
};
```

A truck can at most load 20 (`CAPACITY`) boxes but it also has a limit concerning the weight which is 2000 (`MAX_WEIGHT`) kilos. This means that it is not possible to load more than 2000 kg.
The class constants declared in the class Truck shall be used to handle these restrictions.

For the class Truck your first task is to define the constructor, destructor and the member functions. This is done in a file named Truck.cpp.

loadBox(…): if it is possible (depending on current number of boxes and current weight), loads a box with the given values contained in the parameters and returns true. If it is not possible to load a box it returns false.

removeBoxAt(…): removes the Box contained in the array at the subscript contained in the parameter *index*. The order of the boxes isn't relevant.

removeAllBoxesWithWeightAbove(…): removes all boxes that has a weight above the weight contained in the parameter *theWeight*.

getCurrentWeight(): returns current weight.

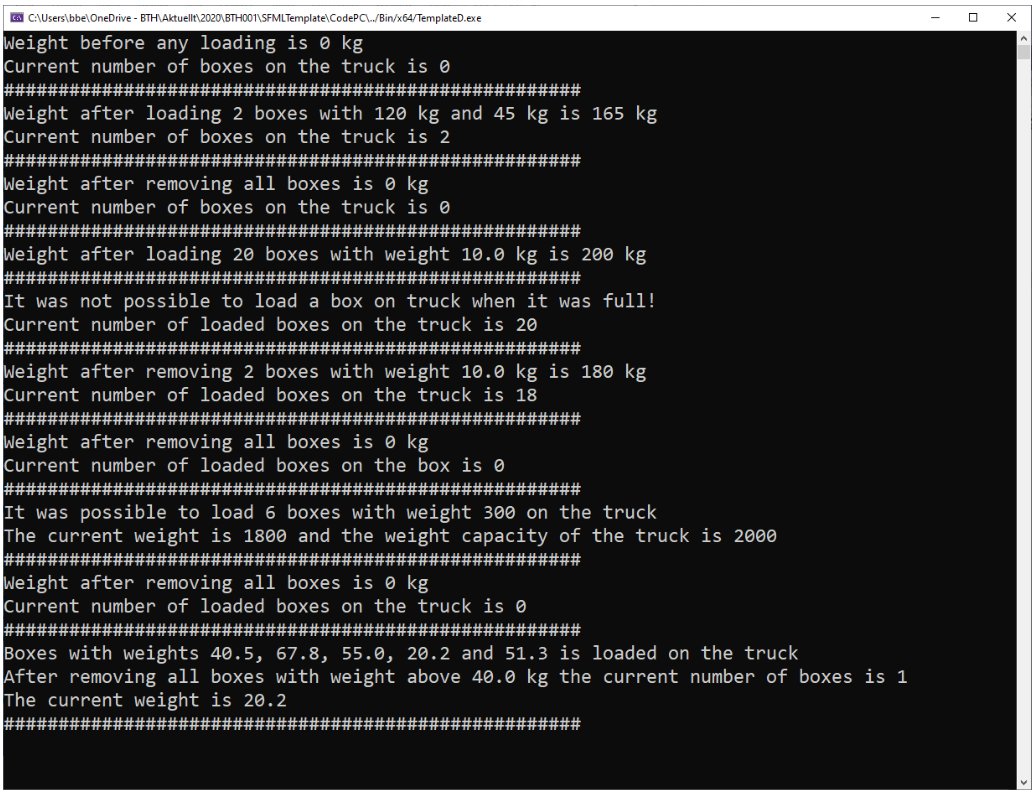
getCurrentNrOfBoxes(): returns current number of boxes.

totalVolumeOfCurrentBoxes(…): summarizes and returns the total volume of all boxes.

The relationship between Truck and Box is implemented using a **statically allocated array of pointers** of type Box. The capacity is given by the class constant CAPACITY.
All boxes must always be found consecutive in the array (no "holes").

Keyboard input and screen output in the member functions are not allowed.

Test the class Truck with the test program that you find in TestTruck.cpp. The output will be


C:\Users\bbe\OneDrive - BTH\Aktuellt\2020\BTH001\SFMLTemplate\CodePC\../Bin/x64/TemplateD.exe

```
Weight before any loading is 0 kg
Current number of boxes on the truck is 0
#####################################################
Weight after loading 2 boxes with 120 kg and 45 kg is 165 kg
Current number of boxes on the truck is 2
#####################################################
Weight after removing all boxes is 0 kg
Current number of boxes on the truck is 0
#####################################################
Weight after loading 20 boxes with weight 10.0 kg is 200 kg
#####################################################
It was not possible to load a box on truck when it was full!
Current number of loaded boxes on the truck is 20
#####################################################
Weight after removing 2 boxes with weight 10.0 kg is 180 kg
Current number of loaded boxes on the truck is 18
#####################################################
Weight after removing all boxes is 0 kg
Current number of loaded boxes on the box is 0
#####################################################
It was possible to load 6 boxes with weight 300 on the truck
The current weight is 1800 and the weight capacity of the truck is 2000
#####################################################
Weight after removing all boxes is 0 kg
Current number of loaded boxes on the truck is 0
#####################################################
Boxes with weights 40.5, 67.8, 55.0, 20.2 and 51.3 is loaded on the truck
After removing all boxes with weight above 40.0 kg the current number of boxes is 1
The current weight is 20.2
#####################################################
```

When the program works as expected it is time to declare and define the copy constructor and the assignment operator in class Truck.

Finally, you shall create a test-program named TestCopyAndAssignment.cpp in which you carefully, for the class Truck, **test the copy constructor and the assignment operator**.

Other requirements:

- No memory leaks are allowed! Use: _CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF | _CRTDBG_LEAK_CHECK_DF); at the beginning of the main-function in the test-programs and run in debug-mode to detect memory leaks.
- Only private member variables are allowed
- All classes shall be divided into header (.h) - and cpp-files where the h-files contains declarations and the cpp-files contains definitions.
- The possibility to implement constant member functions shall be used
- Global variables are not allowed