Laboratory 13 (8 points)

Implement an **Array** class that holds an array of values of different numeric types. The class dynamically allocates the memory needed to hold a values of a given type.

After completing Task 1 you can solve other tasks in any order.

Task 1 (2p)

Modify the given Array class so that the class can hold values of any data type.
Overload operator<< () for the class, add the virtual display() method.

Task 2 (2p)

Define `T accumulate() const`, method that returns the sum of the values of the elements stored in the member array field of **Array** class. Define a specialized accumulate () method for the class instantiated for `char` so that only the characters representing digits '0' to '9' contained in the array are summed up.

Task 3 (1p)

Declare and define a template function `bool cmp()` that compare two values.
Declare and define a template function `void` sort (), that sorts the values in the **Array** template class in descending order.

Task 4 (2p)

Define a template class `WeightedArray` derived from **Array** class, that will contain an array with the corresponding weights. The class template should take 2 parameters - *data type* and weights array *size* – allocated as static array. The size of this array is equal to the size of the value array.

Task 5 (1p)

Throw and catch an exception `"Index out of array bound"` for invalid operator[] index. Write test code in the Part 5 section of main() function.

Example output:

```
---------- Part 1 (2 point) ----------

-3 -1 -2 0 -5 2 7

2.5 0.7 0.4 -0.1 1.8 3.3 -5

a n a r r a y

-1

1.8

a

 ---------- Part 2 (2 point) ----------

-2
```

```
3.6
0
9
 ---------- Part 3 (1 point) ----------
cmp() test :  cmp(1,2) = true ( = true)
7 2 0 -1 -2 -3 -5

3.3 2.5 1.8 0.7 0.4 -0.1 -5

y r r n a a a

 ---------- Part 4 (2 point) ----------
-3:2.5 -1:0.7 -2:0.4 0:-0.1 -5:1.8 2:3.3 7:-5
a:2.5 n:0.7 a:0.4 r:-0.1 r:1.8 a:3.3 y:-5
 ---------- Part 5 (1 point) ----------
Invalid index
 ---------- Part 5 (end) ----------
```