Program Description

The purpose of this assignment was to use our knowledge of pointers and arrays to create a user-friendly data structure, based on the vector object. Our custom implementation would borrow many features from the STL vector, which is a data container.

Data Structure Description & Runtime

Inside our data structure *My_vec* and *My_vector* there is an integer storing size, which is the number of filled “slots” in our container. There is another integer storing the capacity, or overall size of our container. There is also a pointer to the beginning of an array. As the size of the vector increases, the capacity must also be adjusted. Whenever the capacity is increased, it is doubled from it’s previous capacity. Whenever capacity is decreased, it is half of the previous capacity. Only when size is larger than the capacity, does the vector resize the capacity. When size is less than half of the capacity, the capacity is halved.

Runtime

Compile & Run instructions

to compile, extract all contents to a new folder. Using terminal, navigate to this directory. Then run the command “g++ -std=c++11 *.cpp” to compile. To run use the common “./a.out”

Logical Exceptions

The program may not behave as intended if a variable is given an incompatible type (such as giving “A” when the program is expecting an integer). The program may not also behave as intended when handling data types that do not have fixed sizes. The program encounters bugs in the tempted version, where when searching for max index, may return an index out of bounds warning.

C++ Object Oriented or Generic Programing Features

This assignment does not use object oriented programming, or generic programming features.

Testing results

See next page.

My_vec

```
dhcp-10-202-146-158:programming assignment1 kyle$ ./a.out
Index: 0   Value: B
1

Index: 0   Value: A
Index: 1   Value: B
2

Index: 0   Value: A
Index: 1   Value: B
Index: 2   Value:
Index: 3   Value:
Index: 4   Value:
Index: 5   Value:
Index: 6   Value:
Index: 7   Value:
Index: 8   Value:
Index: 9   Value:
Index: 10   Value: D
Index: 11   Value:
Index: 12   Value:
Index: 13   Value:
Index: 14   Value:
Index: 15   Value:
3



removing at rank 2

Index: 0   Value: A
Index: 1   Value: B
Index: 2   Value:
Index: 3   Value:
Index: 4   Value:
Index: 5   Value:
Index: 6   Value:
Index: 7   Value:
Index: 8   Value:
Index: 9   Value: D
Index: 10   Value:
Index: 11   Value:
Index: 12   Value:
Index: 13   Value:
Index: 14   Value:
Index: 15   Value:
3

Index: 0   Value: A
Index: 1   Value: B
Index: 2   Value: E
Index: 3   Value:
Index: 4   Value:
Index: 5   Value:
Index: 6   Value:
Index: 7   Value:
Index: 8   Value:
Index: 9   Value: D
Index: 10   Value:
Index: 11   Value:
Index: 12   Value:
Index: 13   Value:
Index: 14   Value:
Index: 15   Value:
4



output v1
```

```
output v1, replace E with Y

Index: 0   Value: A
Index: 1   Value: B
Index: 2   Value: Y
Index: 3   Value:
Index: 4   Value:
Index: 5   Value:
Index: 6   Value:
Index: 7   Value:
Index: 8   Value:
Index: 9   Value: D
Index: 10   Value:
Index: 11   Value:
Index: 12   Value:
Index: 13   Value:
Index: 14   Value:
Index: 15   Value:


output v2

Index: 0   Value: K
1



v2 now equals v1

Index: 0   Value: A
Index: 1   Value: B
Index: 2   Value: Y
Index: 3   Value:
Index: 4   Value:
Index: 5   Value:
Index: 6   Value:
Index: 7   Value:
Index: 8   Value:
Index: 9   Value: D
Index: 10   Value:
Index: 11   Value:
Index: 12   Value:
Index: 13   Value:
Index: 14   Value:
Index: 15   Value:
4

max value index: 2
Index out of bounds
Index out of bounds
Index out of bounds
Index out of bounds
Index out of bounds
Index out of bounds
Index out of bounds
Index out of bounds
Index out of bounds


list should be sorted by max to min

Index: 0   Value: Y
Index: 1   Value: D
Index: 2   Value: B
Index: 3   Value: A
Index: 4   Value:
Index: 5   Value:
Index: 6   Value:
Index: 7   Value:
Index: 8   Value:
```

My_vector

```
Index: 11   Value:
Index: 12   Value:
Index: 13   Value:
Index: 14   Value:
Index: 15   Value:
4

max value index: 2
Index out of bounds
Index out of bounds
Index out of bounds
Index out of bounds
Index out of bounds
Index out of bounds
Index out of bounds
Index out of bounds
Index out of bounds


list should be sorted by max to min

Index: 0   Value: Y
Index: 1   Value: D
Index: 2   Value: B
Index: 3   Value: A
Index: 4   Value:
Index: 5   Value:
Index: 6   Value:
Index: 7   Value:
Index: 8   Value:
Index: 9   Value:
Index: 10   Value:
Index: 11   Value:
Index: 12   Value:
Index: 13   Value:
Index: 14   Value:
Index: 15   Value:
Type: int

size : 1
max index: 0
sorting...
Index: 0   Value: 5
Index: 1   Value: 0


Type: char

size : 1
max index: Index out of bounds
0
sorting...
Index: 0   Value: A
Index: 1   Value:


Type: double

size : 1
max index: Index out of bounds
Index out of bounds
Index out of bounds
Index out of bounds
3
sorting...
Index: 0   Value: 1.51
Index: 1   Value: 3.21143e-322


dhcp-10-202-146-158:programming assignment1 kyle$
```