Stanisław Dutkiewicz 263509

Object Oriented Programming Project:

NATO Bases

Description:

My idea to create this project was to create a program that holds the database of NATO bases placed in the Eastern Flank that would be used in the NATO headquarters. The program can manipulate those bases; therefore, NATO employees can use this program to check their information and its facilities. It could also be used if the bases are enough to be useful in actions against the enemy. Especially when now, we are experiencing the war in Ukraine.

The databases include details such as name, country, city, number of rifles, tanks, planes, and rocket launchers. Whenever a user wants to add the base, the program asks for the input of the given attributes and attaches the unique identification number to the database. With inputted databases, the program can display all of them, remove and edit them using their ID number. Additionally, all bases can be saved and opened using the special txt file made by the program. Finally, each base can be compared using its number of resources to the given number of Russian divisions that could attack the chosen base and display whether the base would survive or not.

The program is written in C++ language.

Structure:

The program has the header file with two classes: Database and Natobase.

Natobase class consists of private attributes such as: name(string), country(string), city(string), rifles(integer), tanks(integer), planes(integer), rocket launchers (integer), ID (integer), counter (static integer). Counter is incremented each time the class is initialised, and it attaches to the number for ID attribute. Also, it has:

- Default constructor
- Parametrized constructor, that initializes class attributes to values provided by the user.
- Constructor that defines implicit conversion
- Initializer list. Converting constructor should delegate the initialization of class attributes to the parameterized constructor for the class.

The Natobase class has input and output overloaded operators that are used to collect data and display databases with its attributes. Functions such as 'tosave' and 'toim' are used to import and

export databases that use input and output operators. Moreover, there is a function called “printnames” that prints IDs and Names of each base. It is used to classify Databases for Natobase functions (edit, remove, compare) All these functions will be further used for functions included in the database function.

The functions 'tosave', 'toim', and 'printnames' are assigned as friend functions for the Natobase class. Other Natobase functions use the public interface to be capable of accessing each of the class attributes as read-only.

The Database class only has one attribute, which is a private vector DbMain that is responsible for storing Databases in vector form. On top of that, it has eight functions that are used in the main file.

Read function uses the input overloaded function from Database to ask the user to input attributes. In addition, the iterator is initialised in a loop over the vector. Inside the loop, the function vector 'push_back' is used to save the inputted data in the DbMain vector.

The Print function uses the same special loop with iterator that overloads the output of every Database that is stored in DbMain.

The Save function with the use of function 'tosave' collects databases and uses ofstream along with the iterator to overloop the elements of the vector to save databases in the txt file called 'Database.'

Load function opens collects the content of the txt file “Database” by ifstream, according to the input operator and attaches to them DbMain vector.

For the last three functions a special auto iterator is declared using function “find_if” that searches up the whole vector to match the ID number with the number that user inputted.

Remove function firstly prints out all Names of Bases along with their IDs. Later, it asks the user which base user wants to remove. The inputted number is used in special auto iterator to match it with the ID of Database. After that, the part of the desired vector is removed by “vector.erase’ function. At the end, the static integer of the counter is decremented.

The Edit function works similarly to the Remove function. It also displays all numbers and IDs of bases and asks user to choose the number of the base. Next, after matching the ID with the auto iterator, the input overload is used to ask the user to input the attributes of the chosen database.

The Compare function is used to perform a mini comparison between a base facility (rifles, tanks, planes, rocket launchers) and the strength of the Russian divisions. A user is asked how many of the Russian division would like to be compared with the chosen base by the user by auto iterator. A Russian army is equivalent to the 1300 number. The function declares a special integer that sums of all facilities that the base has., it uses a public interface to attain the numbers of facilities.

The equation for this is: sum = 1 * rifle + 0.9 * tanks + 2 * planes + 3* launchers

If the sum of facilities exceeds that number of Russian divisions, the system shows that the base will survive the attack. If not, the system will say that the base will not oppose that force.

Finally, a Clear Function removes all bases stored in vector by the function “vector. clear”.

Interface design:

The main file consists of the infinity loop that asks user to enter the certain to initiate all the Database class functions. The loop will stop working until number 9 is clicked (which is exit).

UML diagram: