

Bubble sort

Software Documentation

Author: matiwa

Table of contents

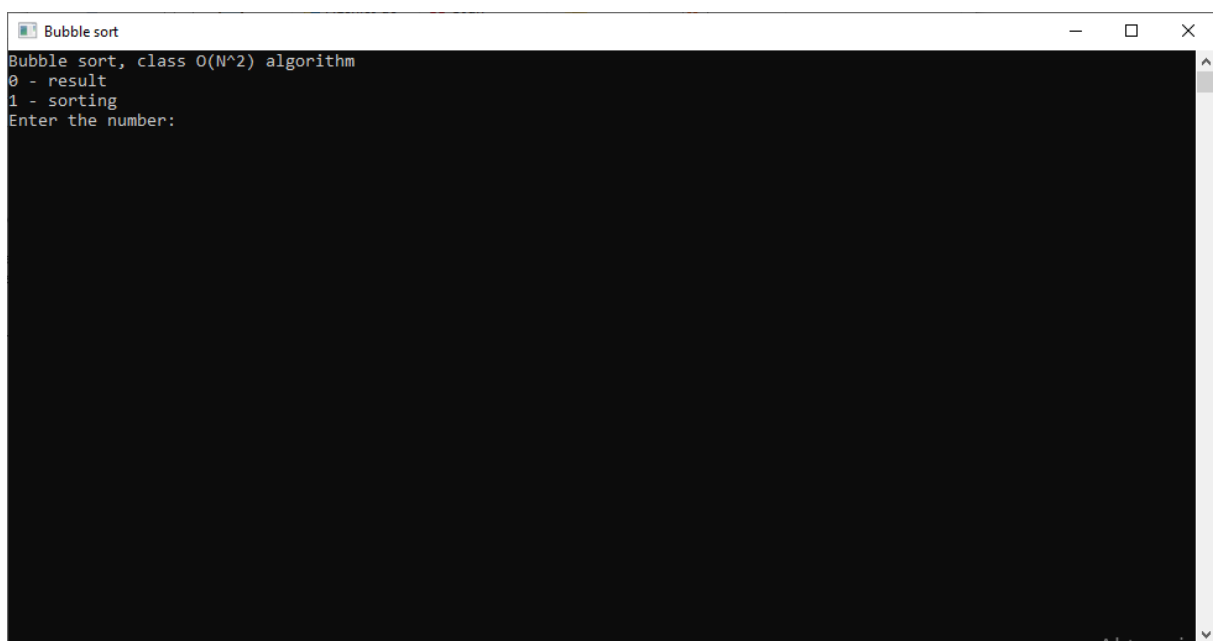
Table of contents.....	2
Introduction.....	3
Describing of the application's operation.....	3
What is needed for use?.....	7
Algorithms used.....	7
Interface description.....	10
Source code description.....	10
List of drawings.....	12
List of listings.....	12
Bibliography.....	12

Introduction

This software documentation includes: description of the application's operation, what is needed for use, algorithms used, interface description and source code description. This application is used to sort numbers from smallest to largest. For this process, a sort algorithm called bubble sort was used.

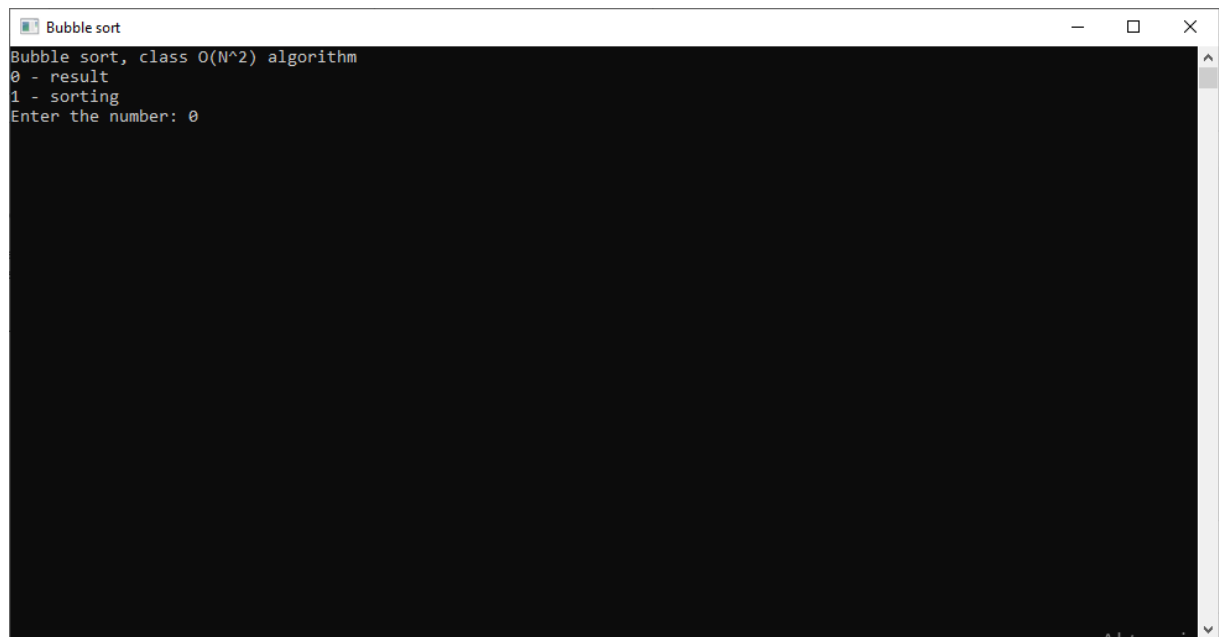
Describing of the application's operation

If the user wants to sort any non-empty set of numbers, run the *.exe file. After this operation, the console window will appear on the screen as below.

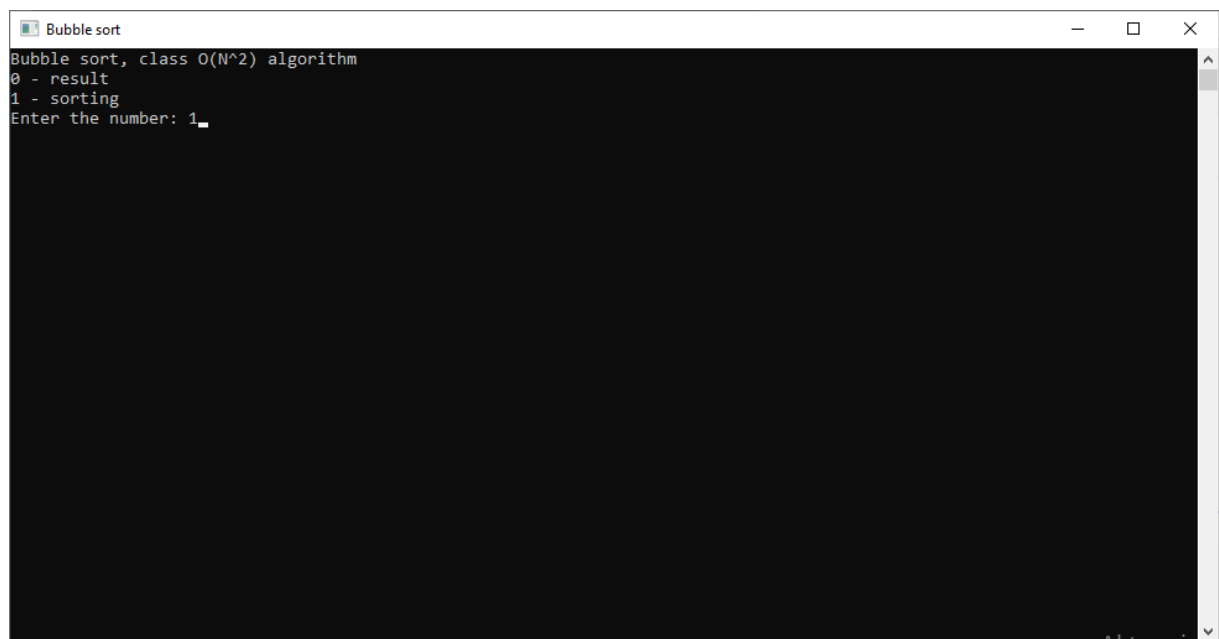


Drawing 1: The beginning of the application's operation [own study]

The user has to choose whether to have the result of the operation displayed alone or with a step-by-step representation of the entire sorting process. For this purpose user must enter a number 0 or 1.

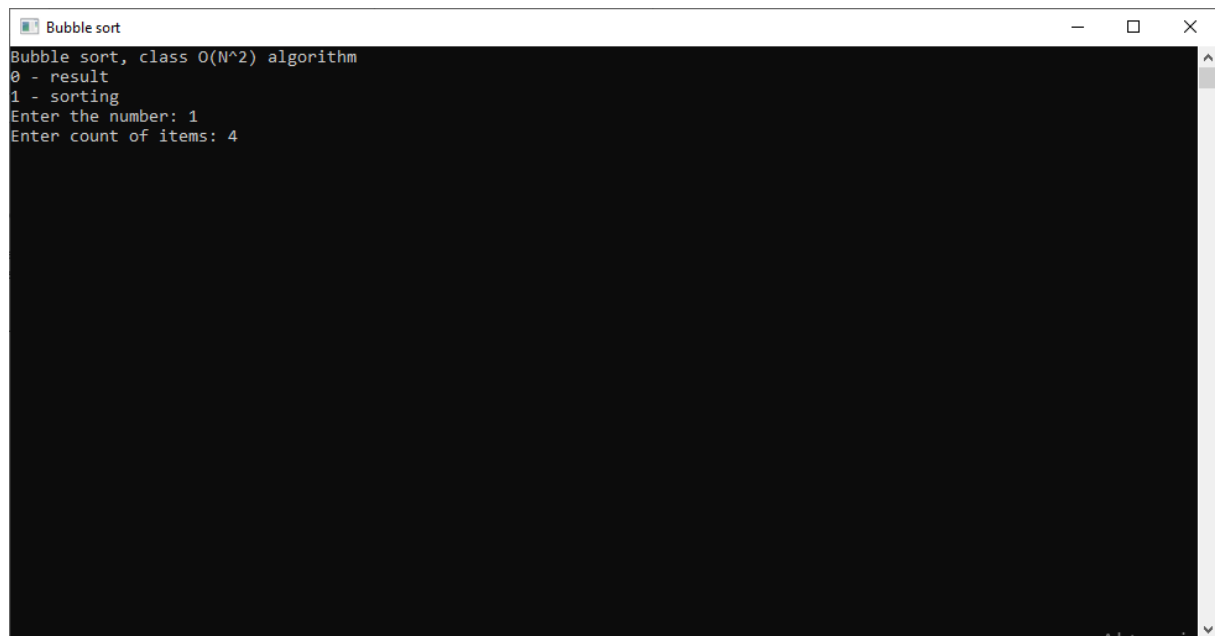


Drawing 2: The user wants to see the result alone [own study]



Drawing 3: The user wants to see the result with the sorting process step by step [own study]

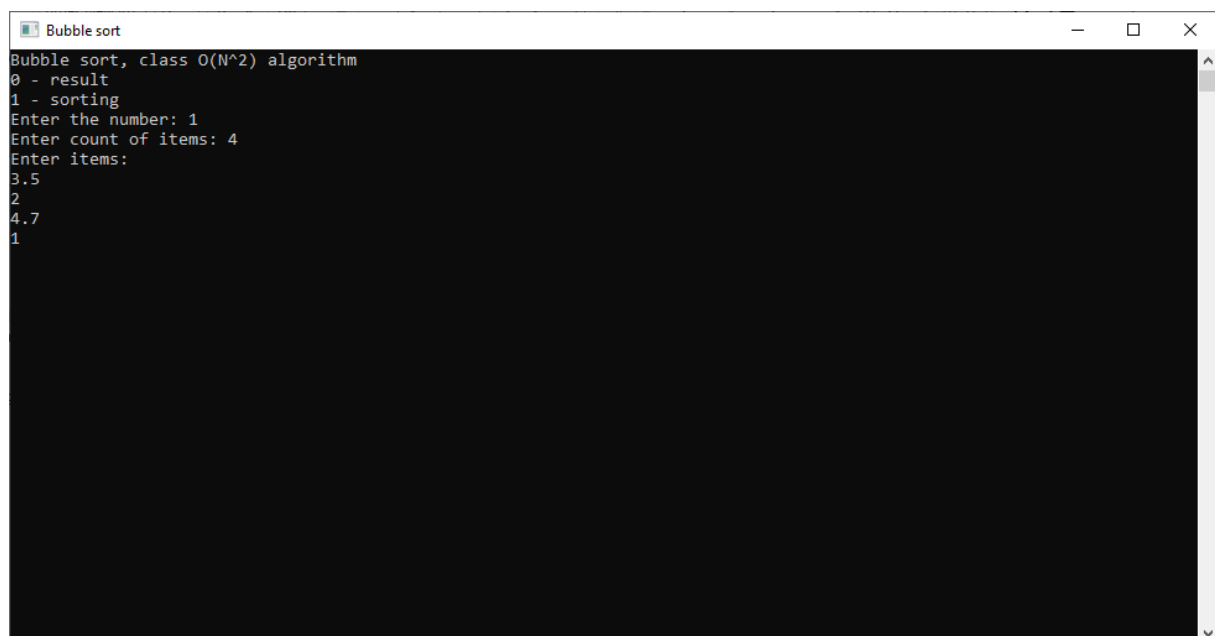
After selecting the option, the user enters the number of elements in the set.



```
Bubble sort, class O(N^2) algorithm
0 - result
1 - sorting
Enter the number: 1
Enter count of items: 4
```

Drawing 4: Entering the number of items [own study]

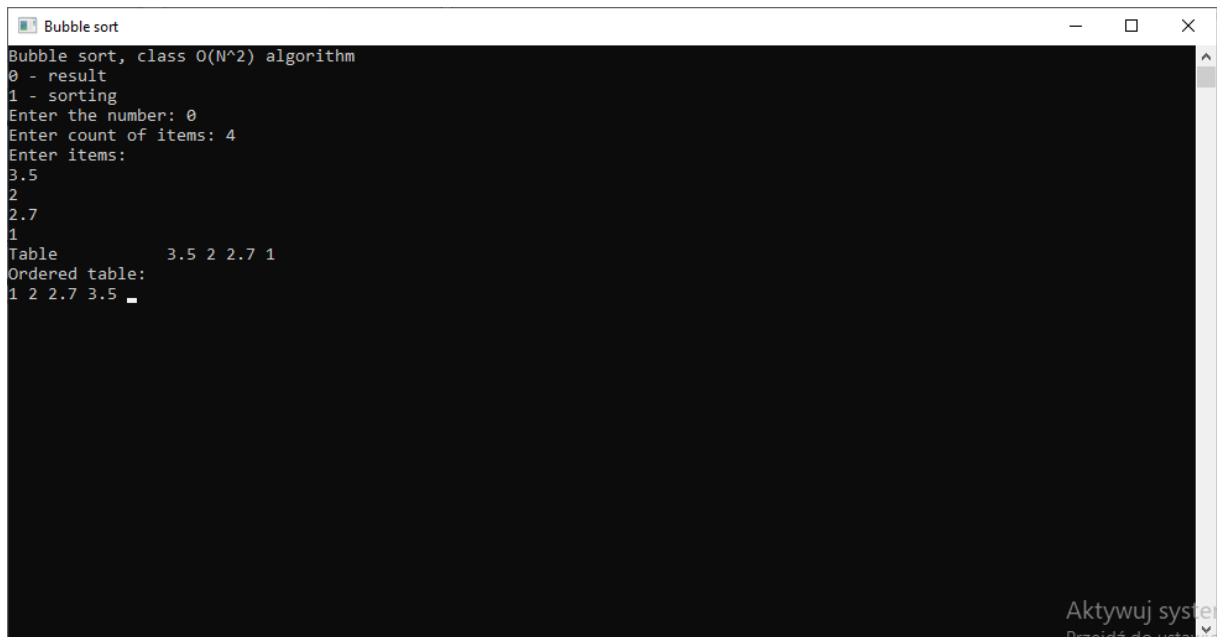
Then enter the values of all elements. The value can be positive, negative and with a decimal point. It is important to use a period instead of a comma when entering a number with a comma.



```
Bubble sort, class O(N^2) algorithm
0 - result
1 - sorting
Enter the number: 1
Enter count of items: 4
Enter items:
3.5
2
4.7
1
```

Drawing 5: Entering the values of all elements [own study]

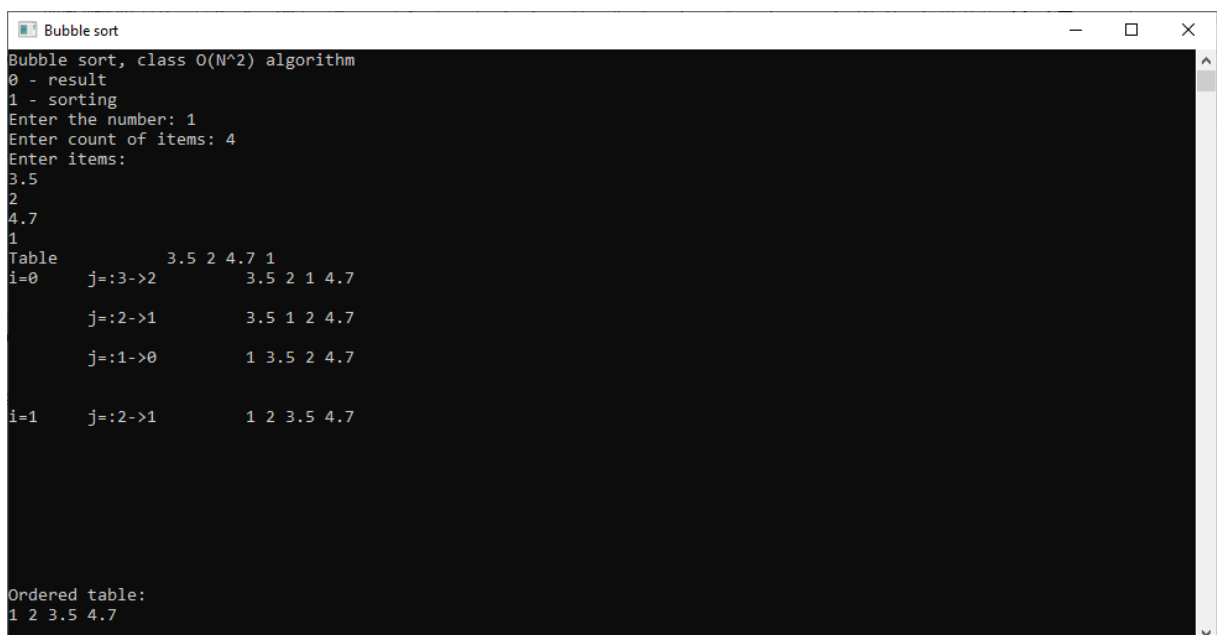
After entering the numbers, the application sorts the set after clicking enter. It is worth noting the differences in the presentation of the results for options 0 and 1. Finally, the application waits for any key to be clicked.



```

Bubble sort, class O(N^2) algorithm
0 - result
1 - sorting
Enter the number: 0
Enter count of items: 4
Enter items:
3.5
2
2.7
1
Table          3.5 2 2.7 1
Ordered table:
1 2 2.7 3.5
  
```

Drawing 6: The final result for option 0 [own study]



```

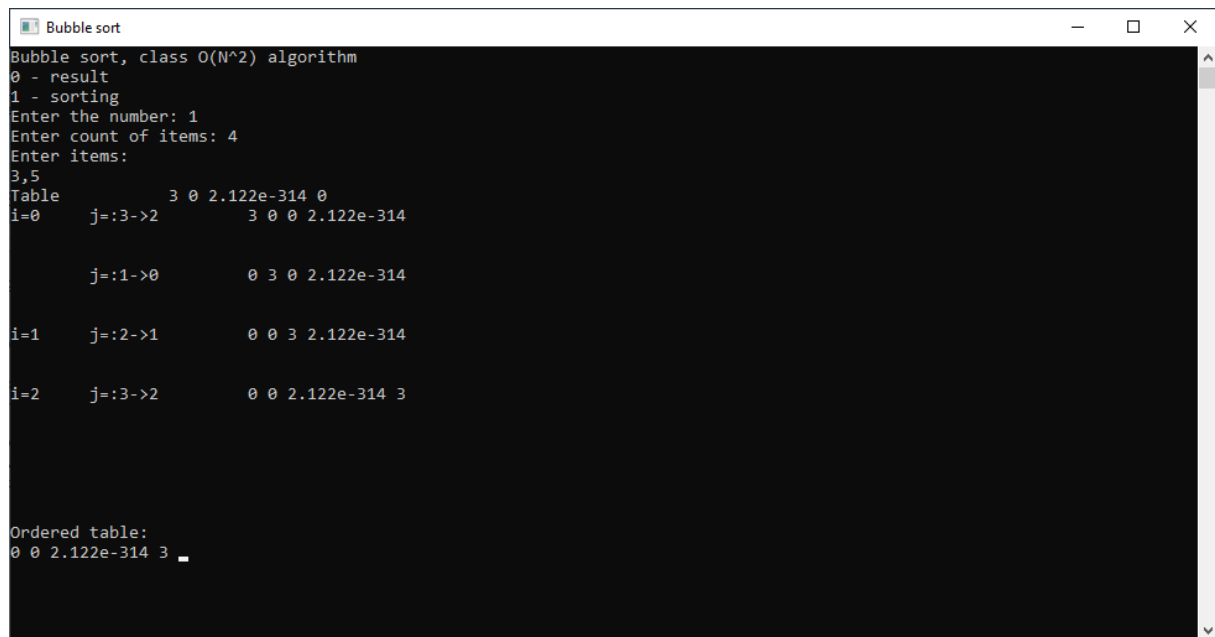
Bubble sort, class O(N^2) algorithm
0 - result
1 - sorting
Enter the number: 1
Enter count of items: 4
Enter items:
3.5
2
4.7
1
Table          3.5 2 4.7 1
i=0      j=:3->2      3.5 2 1 4.7
          j=:2->1      3.5 1 2 4.7
          j=:1->0      1 3.5 2 4.7
i=1      j=:2->1      1 2 3.5 4.7

Ordered table:
1 2 3.5 4.7
  
```

Drawing 7: The final result for option 1 [own study]

If the user presses any key, the console window clears and other values can be sorted. This is the correct use of the program. There may be times when the user makes a mistake. It is important for the user to enter numbers with a comma, using a period instead of a comma. If

he does otherwise, only the value before the decimal point will be entered and further input of the value will be aborted.



```
Bubble sort, class O(N^2) algorithm
0 - result
1 - sorting
Enter the number: 1
Enter count of items: 4
Enter items:
3,5
Table
i=0      j=:3->2      3 0 2.122e-314 0
          j=:1->0      0 3 0 2.122e-314
i=1      j=:2->1      0 0 3 2.122e-314
i=2      j=:3->2      0 0 2.122e-314 3

Ordered table:
0 0 2.122e-314 3
```

Drawing 8: User error [own study]

What is needed for use?

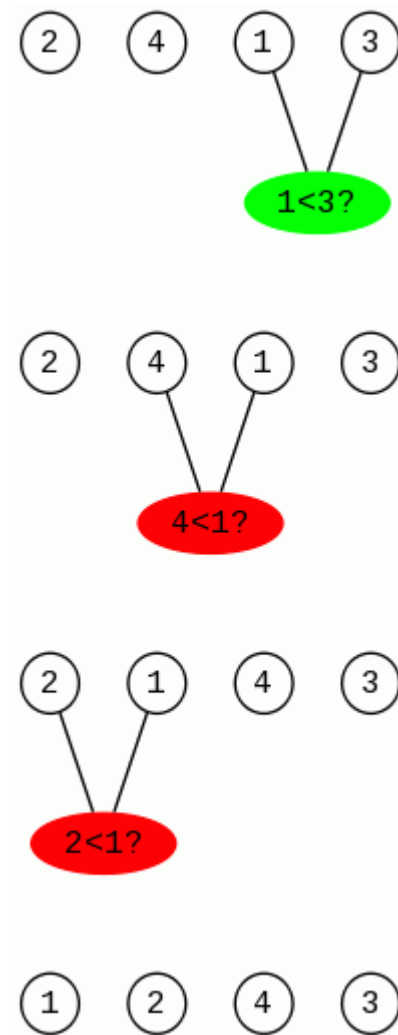
The application does not require installation. It only needs the Windows operating system.

Algorithms used

The only algorithm used for the implementation is a sort algorithm called bubble sort. This is one of the simplest sorting algorithms. We check the whole array from the end, if we find a pair of elements where the larger one precedes the smaller one, we swap them places. After going through the entire array, we start searching this array again from the end. We repeat the activity until when checking the whole array, there is not a single exchange of elements. This is most often done using a logical variable.

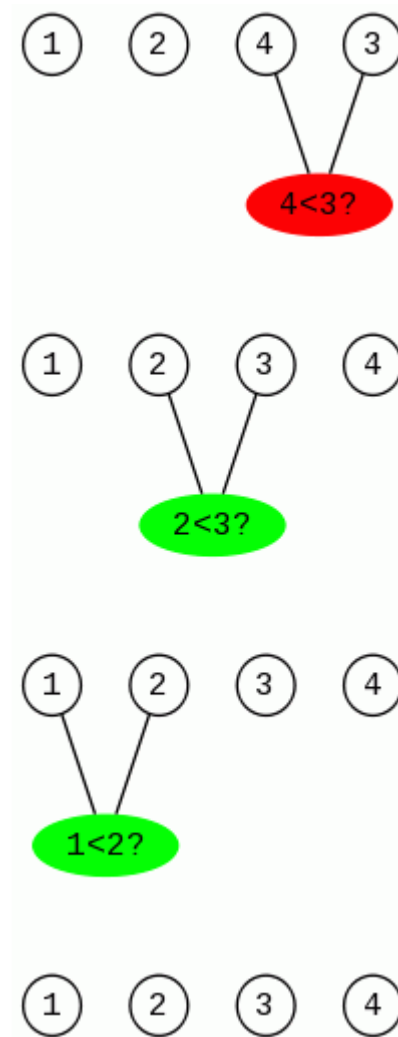
The algorithm is called the bubble algorithm because the smallest numbers "float" from the bottom of the board to the top, just like air bubbles in water.

Here is an example of use for an unordered sequence of numbers $\langle\langle 2, 4, 1, 3 \rangle\rangle$. [1]



Drawing 9: First pass of all elements [1]

After the first pass of all the elements, the number 1 came to the front. We start checking the table again from the end.[1]



Drawing 10: Second pass of all elements [1]

On the next run, no change will occur, a sign that the string is already sorted.[1]

Basic pseudo-code of the algorithm for an array of size n (array elements are numbered from 0 to $n-1$):[2]

```

procedure bubbleSort( A : list of items to sort)
  n = numer_of_elements(A)
  do
    for (i = 0; i < n-1; i++) do:
      if A[i] > A[i+1] then
        swap(A[i], A[i+1])
      end if
    end for
    n = n-1
  while n > 1
end procedure

```

Listing 1: Pseudocode [2]

Interface description

The interface is a console pane. The operation of the program is based on user communication. He gives the needed values on the input. The length of the action depends on how many items are to be sorted. The course of the process and possible operating errors are described in the chapter „Describing of the application’s operation”.

Source code description

The project was made in the C++ programming language, in the Dev-C++ programming environment. All work was done on the Windows 10 operating system. The application's source code looks like this.

```
#include<iostream>
#include<conio.h>
#include<windows.h>
using namespace std;

int main(){
    SetConsoleTitleA("Bubble sort");
    int option;
    cout<<"Bubble sort, class O(N^2) algorithm"<<endl<<"0 -
result"<<endl<<"1 - sorting"<<endl<<"Enter the number: ";
    cin>>option;
    for(;;){
        int sizet;
        double* numbers;
        cout<<"Enter count of items: ";
        cin>>sizet;
        numbers=new double[sizet];
        cout<<"Enter items:"<<endl;
        for(int i=0;i<sizet;i++) cin>>numbers[i];
        cout<<"Table\t\t";
        for(int i=0;i<sizet;i++) cout<<numbers[i]<<" ";
        cout<<"\t"<<endl;
        for(int i=0;i<sizet;i++){
            int swap=0;
            for(int j=sizet-1;j>0;j--){
                if(numbers[j]<numbers[j-1]){
                    if((option==1)&&(swap==0))
                        cout<<"i="<<i<<"\t";

                    double tmp=numbers[j];
                    numbers[j]=numbers[j-1];
                    numbers[j-1]=tmp;
```

```

        if((swap>0)&&(option==1)) cout<<"\t";
        if(option==1) cout<<"j="<<j<<"->"<<j-
1<<"\t\t";
        if(option==1){
            for(int k=0;k<szet;k++)
                cout<<endl;
        }
        swap++;
    }
    if(option==1) cout<<endl;
}
cout<<"Ordered table: "<<endl;
for(int i=0;i<szet;i++) cout<<numbers[i]<<" ";

delete[] numbers;
getch();
system("cls");
}
return 0;
}

```

Listing 2: Source code [own study]

The option variable holds a value that determines whether or not to show sort step by step. The user only decides which option he chooses at the beginning! The szet variable holds the size of the array with the values to be sorted. The numbers array holds values. Due to its flexible nature, it has to be dynamic, so you need to free up some memory at the end. The getch () statement waits for any key to be pressed, and system ("cls") clears the console window. This fact does not change the value of the option variable!

List of drawings

Drawing 1: The beginning of the application's operation [own study].....	3
Drawing 2: The user wants to see the result alone [own study].....	4
Drawing 3: The user wants to see the result with the sorting process step by step [own study].....	4
Drawing 4: Entering the number of items [own study].....	5
Drawing 5: Entering the values of all elements [own study].....	5
Drawing 6: The final result for option 0 [own study].....	6
Drawing 7: The final result for option 1 [own study].....	6
Drawing 8: User error [own study].....	7
Drawing 9: First pass of all elements [1].....	8
Drawing 10: Second pass of all elements [1].....	9

List of listings

Listing 1: Pseudocode [2].....	9
Listing 2: Source code [own study].....	10

Bibliography

- [1] <http://www.algorytm.org/algorytmy-sortowania/sortowanie-babelkowe-bubblesort.html>
- [2] https://pl.wikipedia.org/wiki/Sortowanie_b%C4%85belkowe