

Insertion sort

Software Documentation

Author: matiwa

Table of contents

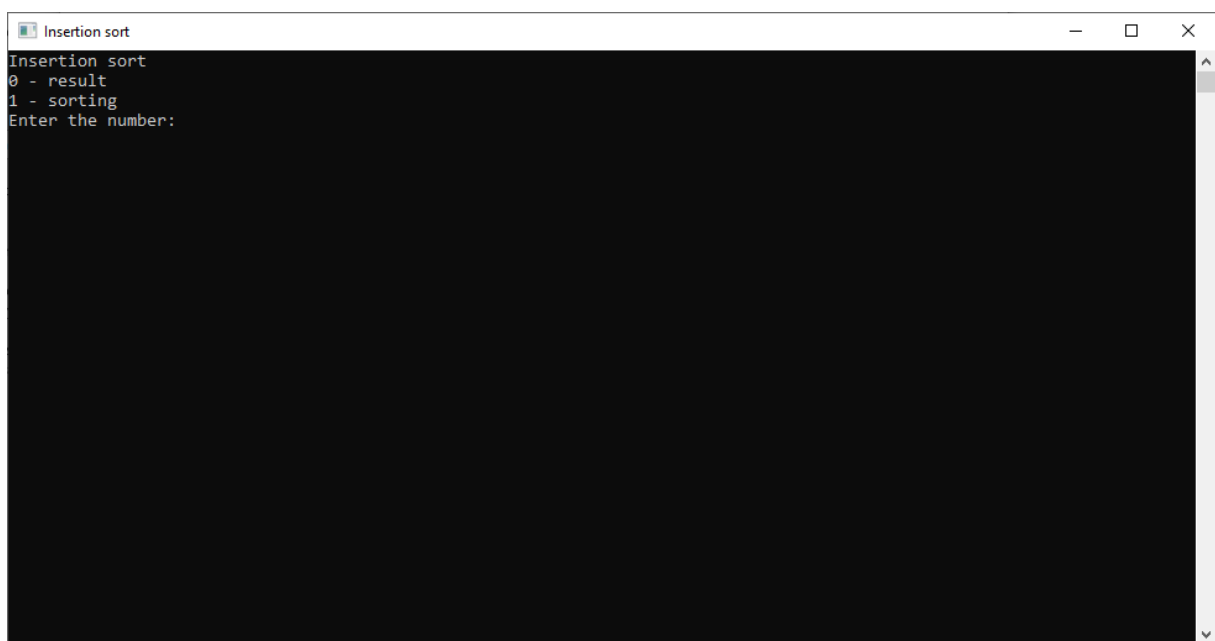
Table of contents.....	2
Introduction.....	3
Describing of the application's operation.....	3
What is needed for use?.....	7
Algorithms used.....	7
Interface description.....	9
Source code description.....	9
List of drawings.....	11
List of listings.....	11
Bibliography.....	11

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 insertion sort was used.

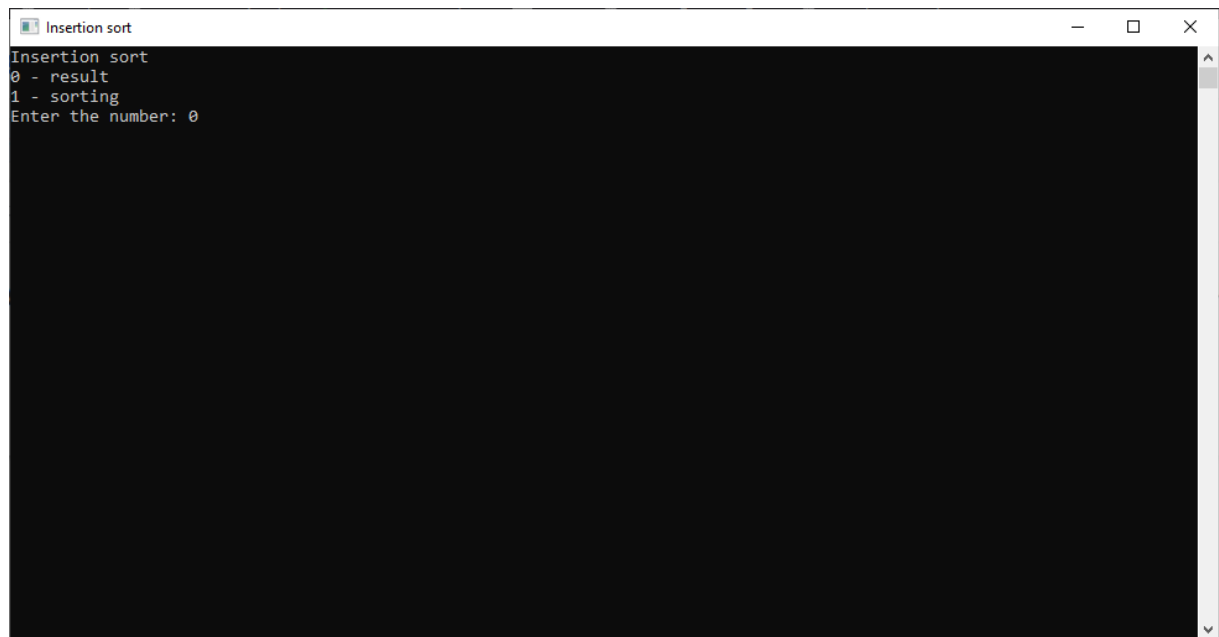
Describing of the application's operation

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.

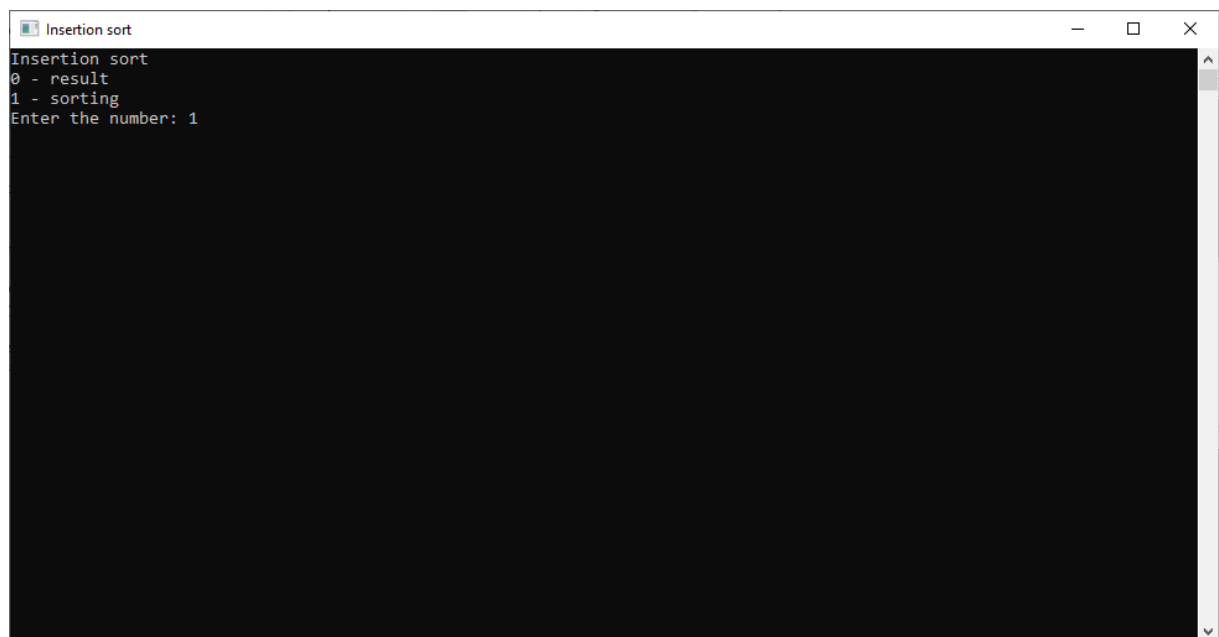


Drawing 1: The beginning of the application's operation [own study]

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.

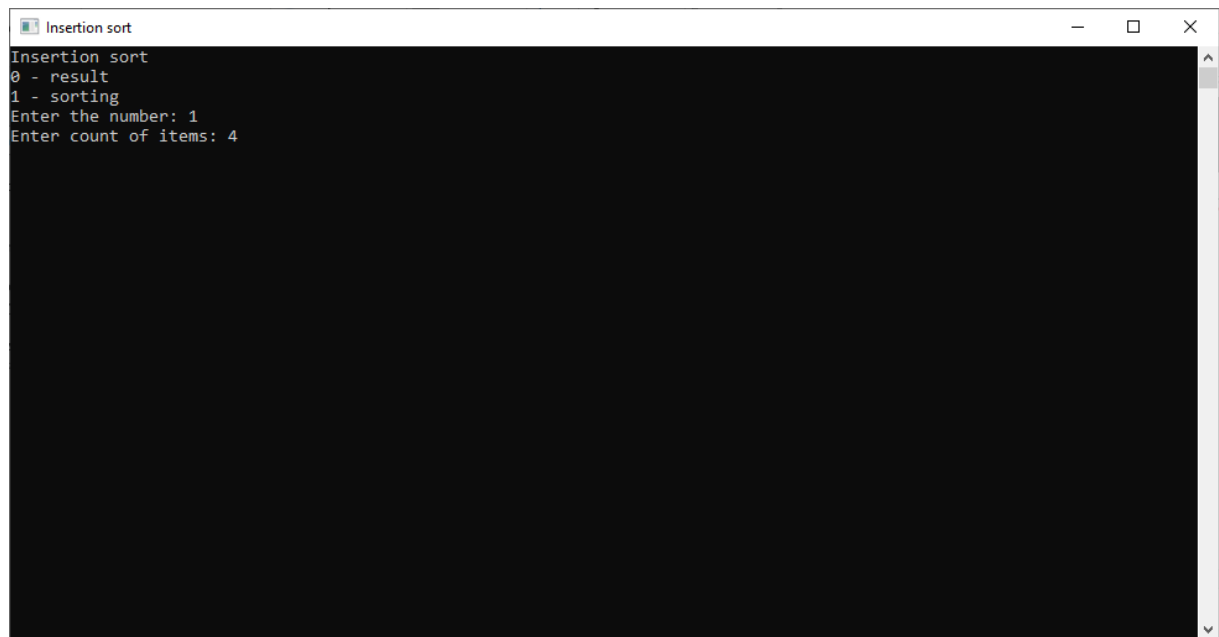


Drawing 2: The user wants to see the result alone [own study]



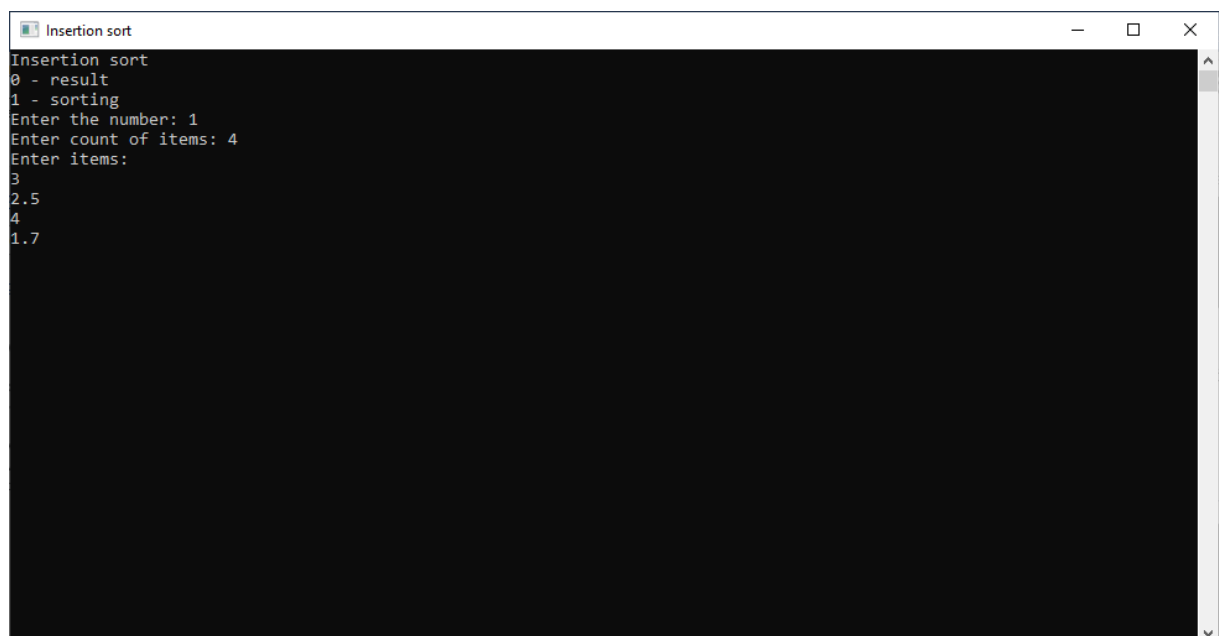
Drawing 3: The user wants to see the result with the sorting process step by step [own study]

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



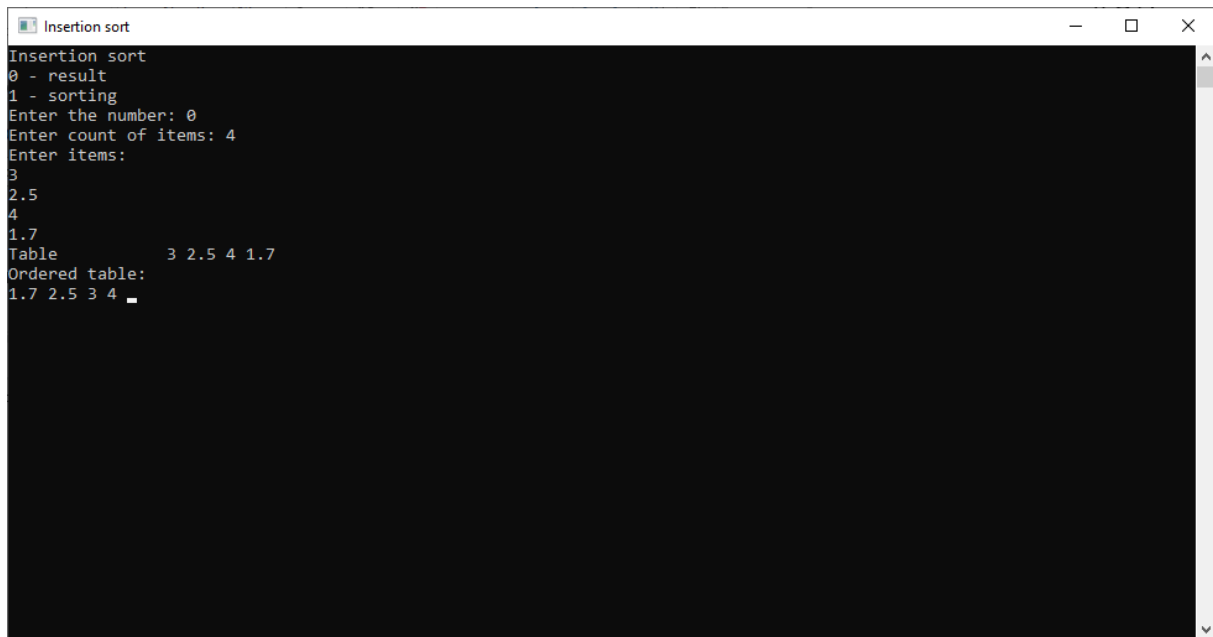
Drawing 4: Entering the number of items [own study]

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.



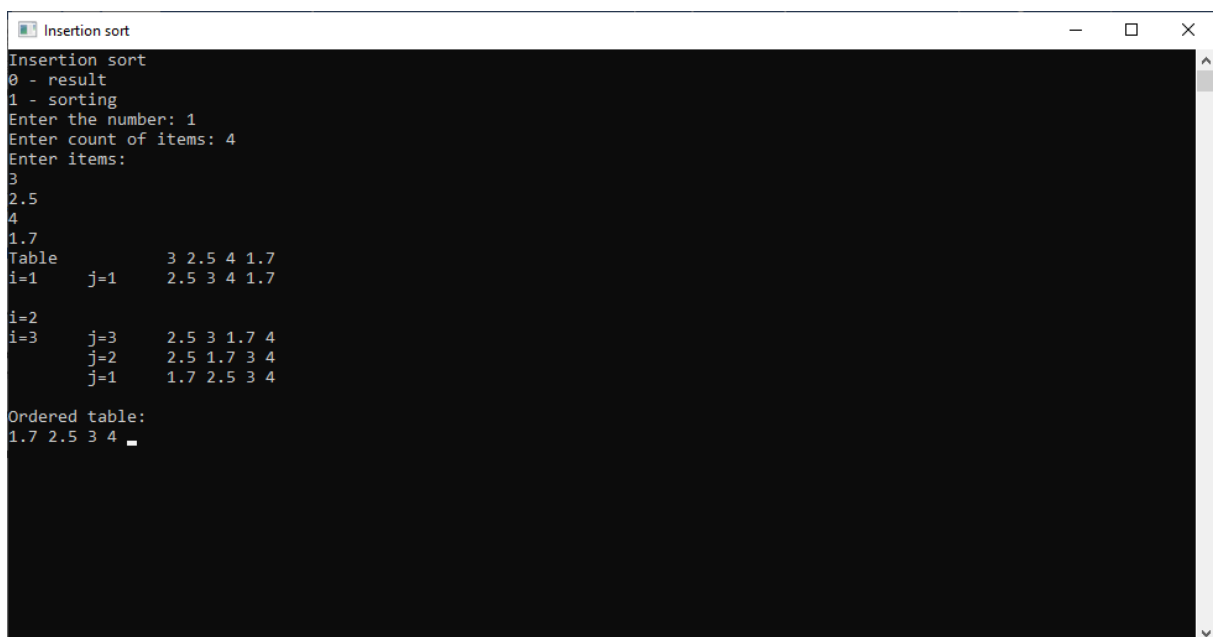
Drawing 5: Entering the values of all elements [own study]

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.



```
Insertion sort
0 - result
1 - sorting
Enter the number: 0
Enter count of items: 4
Enter items:
3
2.5
4
1.7
Table      3 2.5 4 1.7
Ordered table:
1.7 2.5 3 4 _
```

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

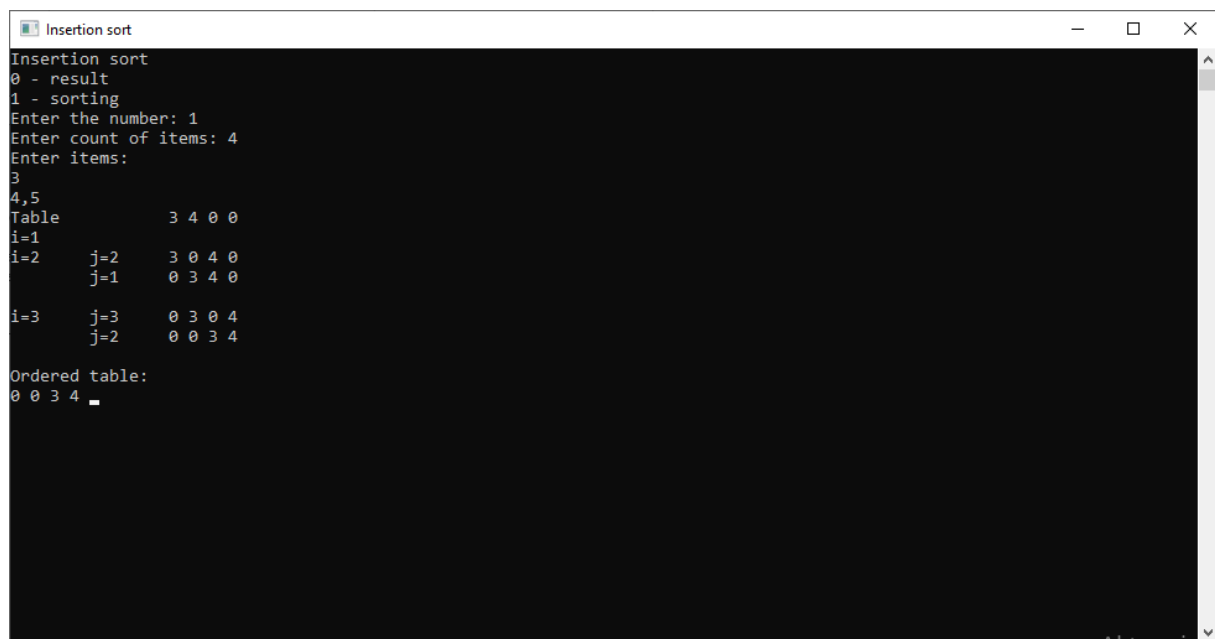


```
Insertion sort
0 - result
1 - sorting
Enter the number: 1
Enter count of items: 4
Enter items:
3
2.5
4
1.7
Table      3 2.5 4 1.7
i=1        j=1    2.5 3 4 1.7
i=2        j=3    2.5 3 1.7 4
           j=2    2.5 1.7 3 4
           j=1    1.7 2.5 3 4
Ordered table:
1.7 2.5 3 4 _
```

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.



```
Insertion sort
0 - result
1 - sorting
Enter the number: 1
Enter count of items: 4
Enter items:
3
4,5
Table      3 4 0 0
i=1
i=2      j=2  3 0 4 0
          j=1  0 3 4 0
i=3      j=3  0 3 0 4
          j=2  0 0 3 4
Ordered table:
0 0 3 4
```

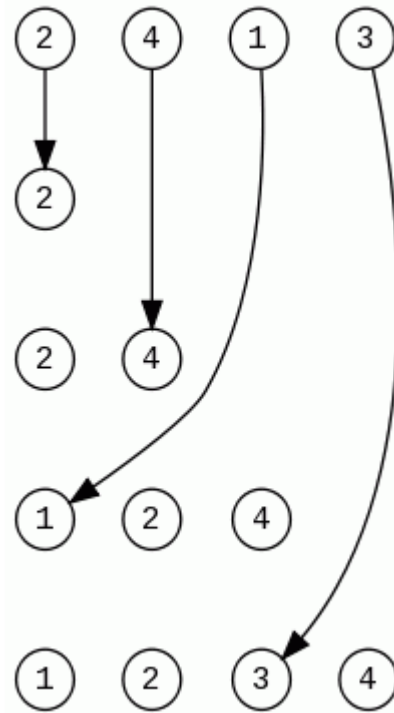
Drawing 8: User error [own study]

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 insertion sort. This is one of the simplest sorting algorithms. Here is an example for the unsorted string <2, 4, 1, 3>.



Drawing 9: Algorithm form [1]

The principle of this algorithm is often compared to arranging cards in a fan during a game. We put each card in the right place, i.e. after the younger one, but before the older one. It is the same with numbers.[1]

The first element remains in place. Then we take the second one and check what relationship it is to with the first. If it is not smaller, it stays in second place, otherwise it goes to first place. Then we check the third element (we compare it to the first two and put it in the right place), the fourth (we compare it to the first three), the fifth, etc. The idea behind the algorithm is to divide the sequence into two parts: the first is sorted, the second is not yet. We choose the next number from the second part and put it in the first. Since it is sorted, we are looking for a place for our number such that the number on the left is not greater and the number on the right is not smaller.[1]

The pseudocode looks like this. The following code shows the pseudo-code algorithm, where:

- A - array of data to be sorted (indexed from 1 to n),
- n - number of elements in table A.[2]

Insert_sort(A, n)

```
for i=2 to n :
# Insert A[i] in sorted string A[1 ... i-1]
element_being_entered = A[i]
j = i - 1
while j>0 and A[j]> element_being_entered:
A[j + 1] = A[j]
```



```
j = j - 1
A[j + 1] = element_being_entered
```

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("Insertion sort");
    int option;
    cout<<"Insertion sort"<<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<<endl;

        int i=1;
        while (i<sizet){
            int j=i;
            if(option==1) cout<<"i="<<i<<"\t";
            int swap=0;
            while((j>0) && (numbers[j-1]>numbers[j])){
```

```

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

cout<<"Ordered table: "<<endl;
for(int i=0;i<sizet;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 sizet 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: Algorithm form [1].....	8

List of listings

Listing 1: Pseudocode [2].....	8
Listing 2: Source code [own study].....	9

Bibliography

- [1] <http://www.algorytm.org/algorytmy-sortowania/sortowanie-przez-wstawianie-insertionsort.html>
- [2] https://pl.wikipedia.org/wiki/Sortowanie_przez_wstawianie