

Selection sort

Software Documentation

Author: matiwa

Table of contents

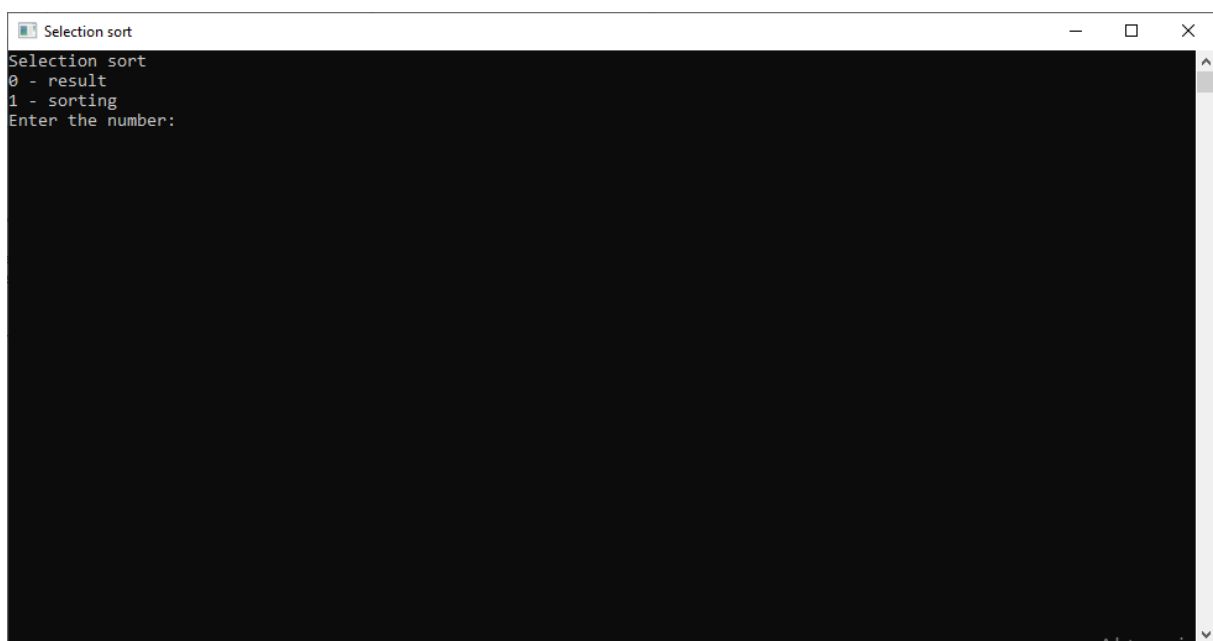
| | |
|--|----|
| Table of contents..... | 2 |
| Introduction..... | 3 |
| Describing of the application's operation..... | 3 |
| What is needed for use?..... | 7 |
| Algorithms used..... | 7 |
| Interface description..... | 8 |
| Source code description..... | 8 |
| List of drawings..... | 11 |
| List 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 selection sort was used.

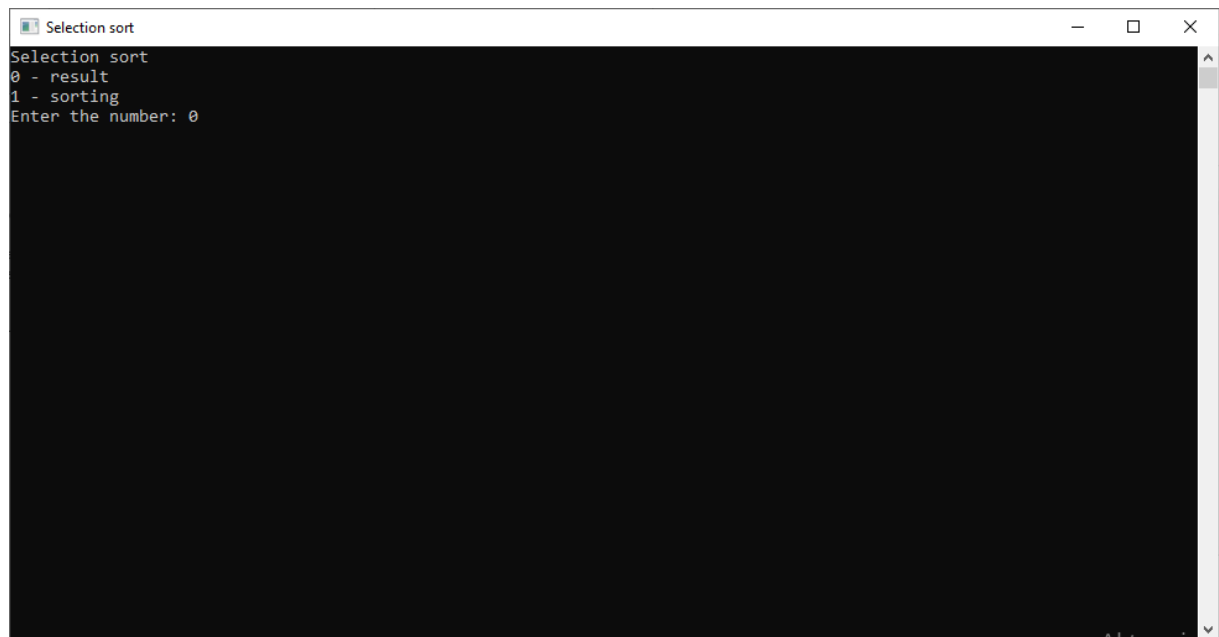
Describing of the application's operation

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.

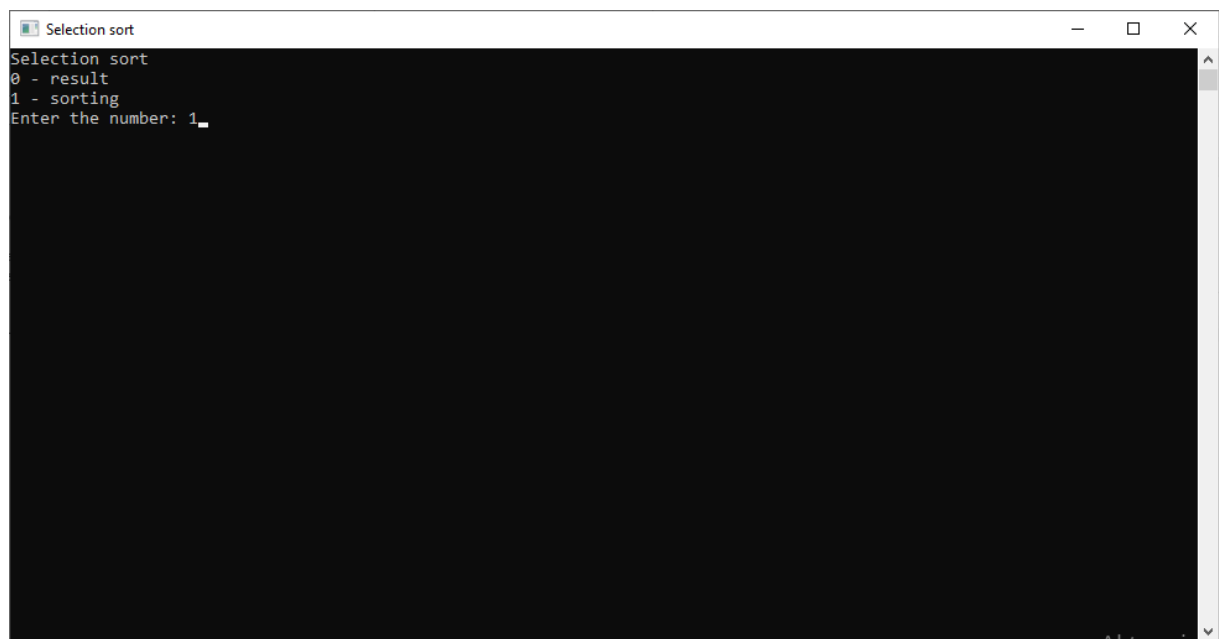


Drawing 1: The beginning of the application's operation [own study]

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.

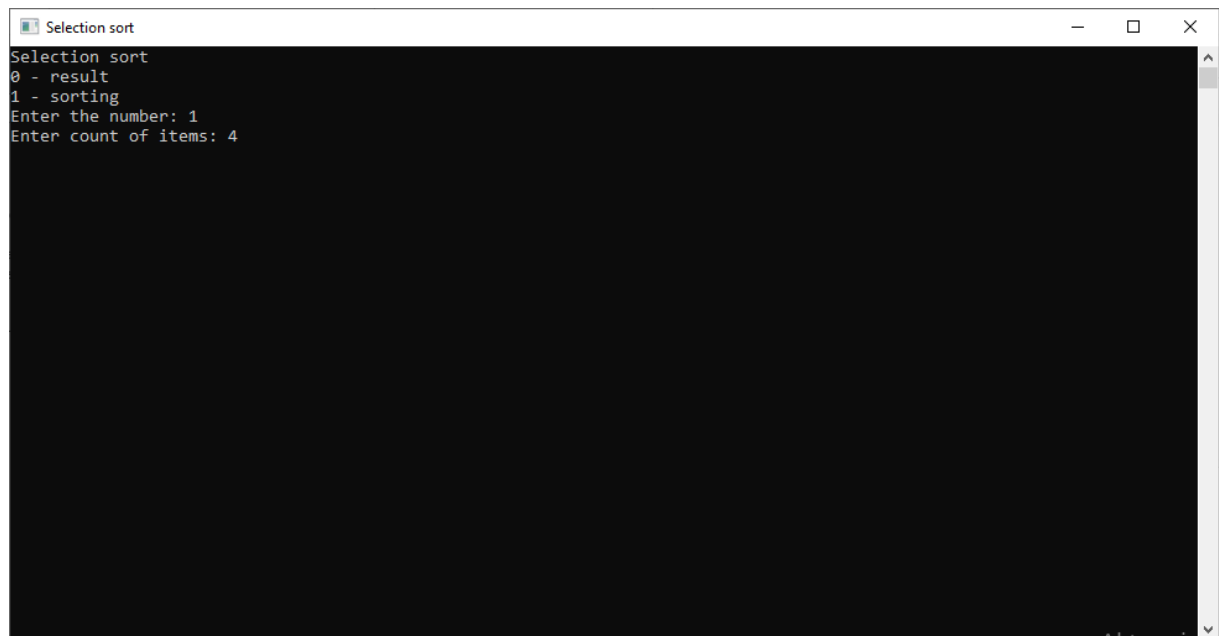


Drawing 2: The user wants to see the result alone [own study]



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

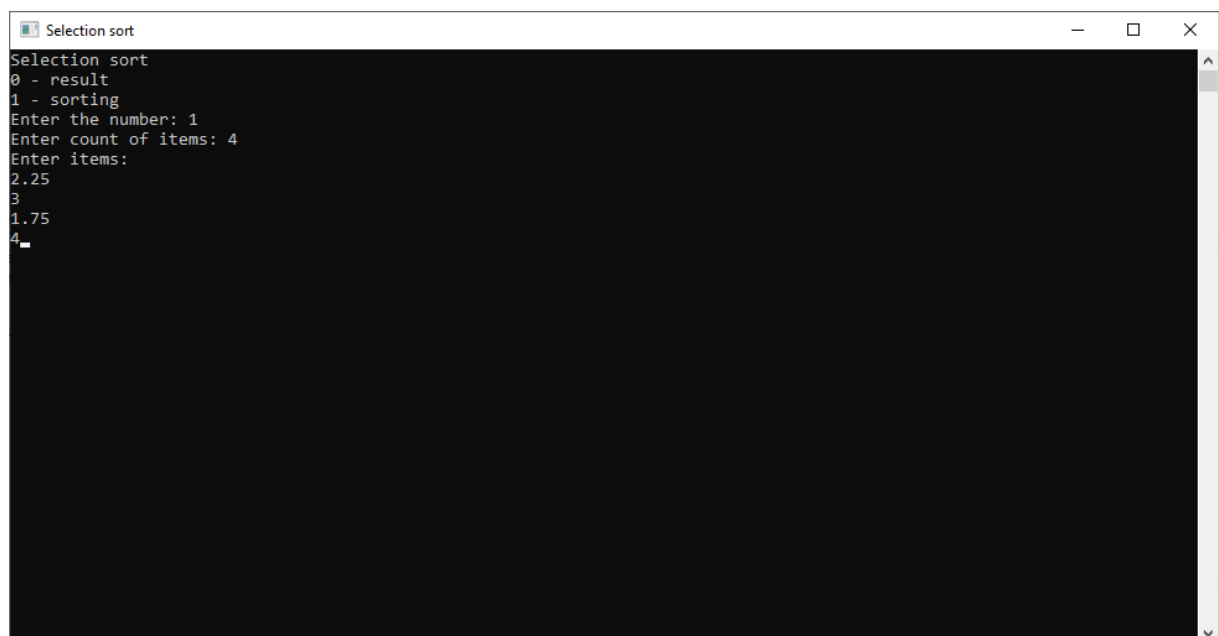
After selecting the option, the user enters the number of elements in the set.



```
Selection sort
0 - result
1 - sorting
Enter the number: 1
Enter count of items: 4
```

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

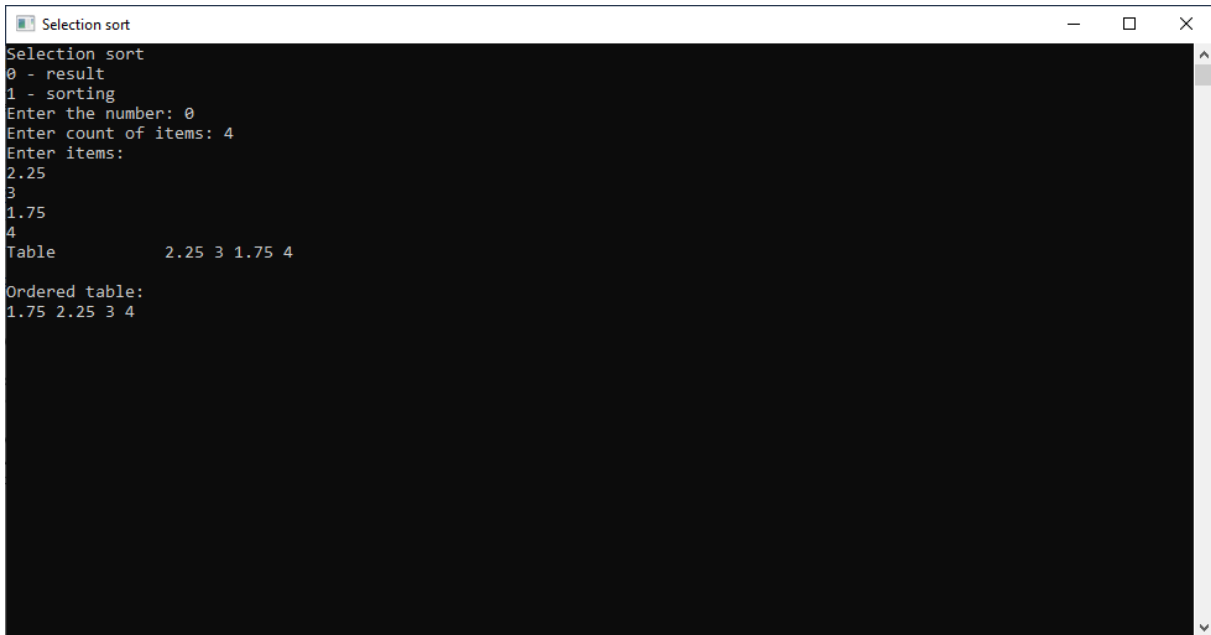
Then enter the values of all elements. The value can be positive, negative and with a decimal point. It is important to use a period instead of a comma when entering a number with a comma.



```
Selection sort
0 - result
1 - sorting
Enter the number: 1
Enter count of items: 4
Enter items:
2.25
3
1.75
4.5
```

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

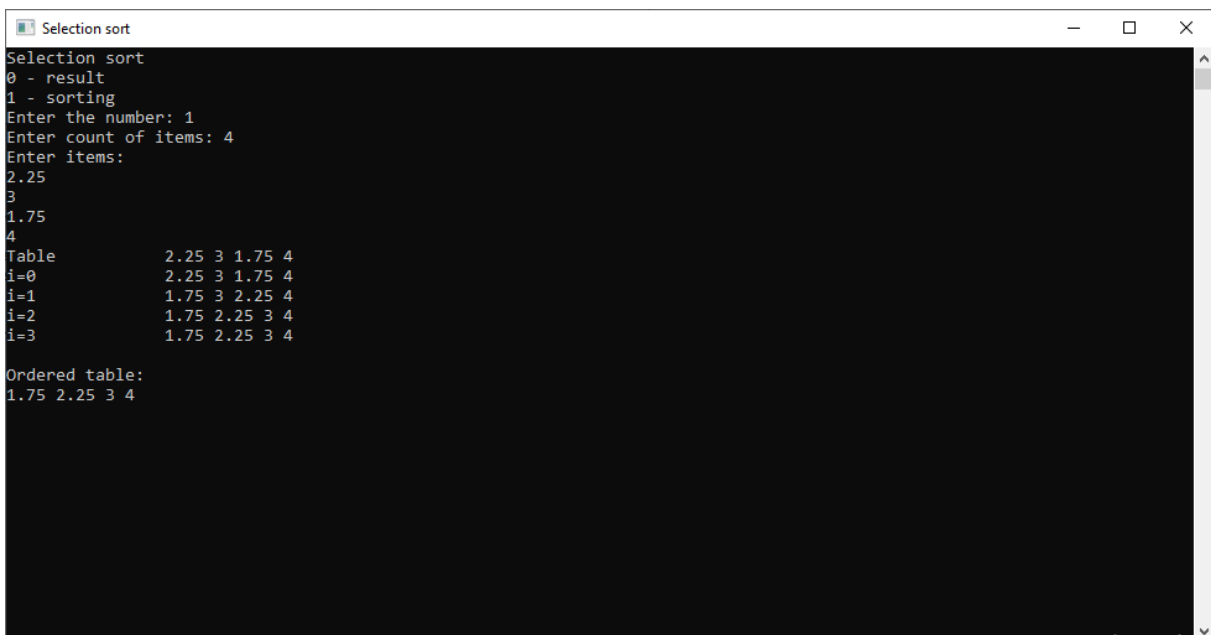
After entering the numbers, the application sorts the set after clicking enter. It is worth noting the differences in the presentation of the results for options 0 and 1. Finally, the application waits for any key to be clicked.



```
Selection sort
0 - result
1 - sorting
Enter the number: 0
Enter count of items: 4
Enter items:
2.25
3
1.75
4
Table          2.25 3 1.75 4

Ordered table:
1.75 2.25 3 4
```

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



```
Selection sort
0 - result
1 - sorting
Enter the number: 1
Enter count of items: 4
Enter items:
2.25
3
1.75
4
Table          2.25 3 1.75 4
i=0            2.25 3 1.75 4
i=1            1.75 3 2.25 4
i=2            1.75 2.25 3 4
i=3            1.75 2.25 3 4

Ordered table:
1.75 2.25 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.

```
Selection sort
0 - result
1 - sorting
Enter the number: 1
Enter count of items: 4
Enter items:
6,6
Table          6 0 2.122e-314 0
i=0            6 0 2.122e-314 0
i=1            0 6 2.122e-314 0
i=2            0 0 2.122e-314 6
i=3            0 0 2.122e-314 6

Ordered table:
0 0 2.122e-314 6
```

Drawing 8: User error [own study]

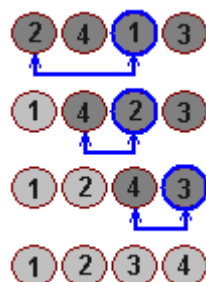
What is needed for use?

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

Algorithms used

This method is called swap sort because it searches for the smallest element first, and when found, it replaces it with the first element in the array.

Then the smallest element is searched again, but starting from the second element (the first - the smallest is already inserted in the appropriate place), after finding it, it is replaced with the second element. We repeat this operation successively on elements from the third, fourth, to the nth..[1]



Drawing 9: An example of the algorithm's operation [1]

Another example of an array to be sorted is the following set. The 8-element array will be sorted [9,1,6,8,4,3,2,0]. The elements among which the minimum value is searched for will be bold in the table.[2]

| | | |
|---|----------------------------|---|
| 0 | [9,1,6,8,4,3,2,0] | 0 |
| 1 | [0, 1 ,6,8,4,3,2,9] | 1 |
| 2 | [0,1,6,8,4,3, 2 ,9] | 2 |
| 3 | [0,1,2,8,4, 3 ,6,9] | 3 |
| 4 | [0,1,2,3, 4 ,8,6,9] | 4 |
| 5 | [0,1,2,3,4,8, 6 ,9] | 6 |
| 6 | [0,1,2,3,4,6, 8 ,9] | 8 |

Drawing 10: An example of the algorithm's operation [2]

The first column shows the iteration number, the second column shows the state of the table, and the third column shows the current minimum. The algorithm can be speeded up a bit when the table is filled from both ends, i.e. the minimum and maximum are searched for at the same time.

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;

void selection_sort(int n, double t[],int option) {
    int i, j, k;
```



```

        for(i=0; i<n; i++) {
            k=i;
            if(option==1){
                cout<<"i="<<i<<"\t\t";
                for(int l=0;l<n;l++) cout<<t[l]<<" ";
                cout<<endl;
            }
            for(j=i+1; j<n; j++) if(t[j]<t[k]) k=j;
            double tmp=t[k];
            t[k]=t[i];
            t[i]=tmp;
        }
        cout<<endl;
    }
}

int main(){
    SetConsoleTitleA("Selection sort");
    int option;
    cout<<"Selection 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<<"\t"<<endl;

        selection_sort(sizet,numbers,option);

        cout<<"Ordered table: "<<endl;
        for(int i=0;i<sizet;i++) cout<<numbers[i]<<" ";

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

```

Listing 1: 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 size 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: An example of the algorithm's operation [1]..... | 7 |
| Drawing 10: An example of the algorithm's operation [2]..... | 8 |

List of listings

| | |
|---|---|
| Listing 1: Source code [own study]..... | 8 |
|---|---|

Bibliography

[1] <http://www.algorytm.org/algorytmy-sortowania/sortowanie-przez-wymiane-wybor-selectionsor.html>

[2] https://pl.wikipedia.org/wiki/Sortowanie_przez_wybieranie