

Quicksort

Software Documentation

Author: matiwa

Table of contents

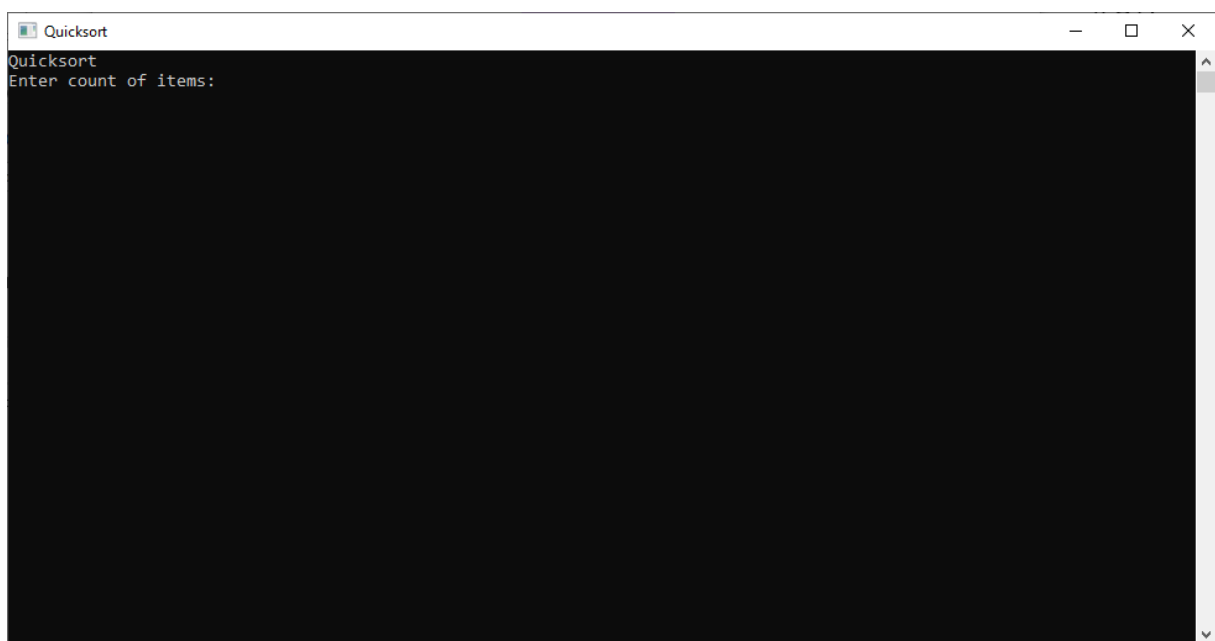
Table of contents.....	2
Introduction.....	3
Describing of the application's operation.....	3
What is needed for use?.....	6
Algorithms used.....	6
Interface description.....	7
Source code description.....	8
List of drawings.....	10
List of listings.....	10
Bibliography.....	10

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 quicksort was used.

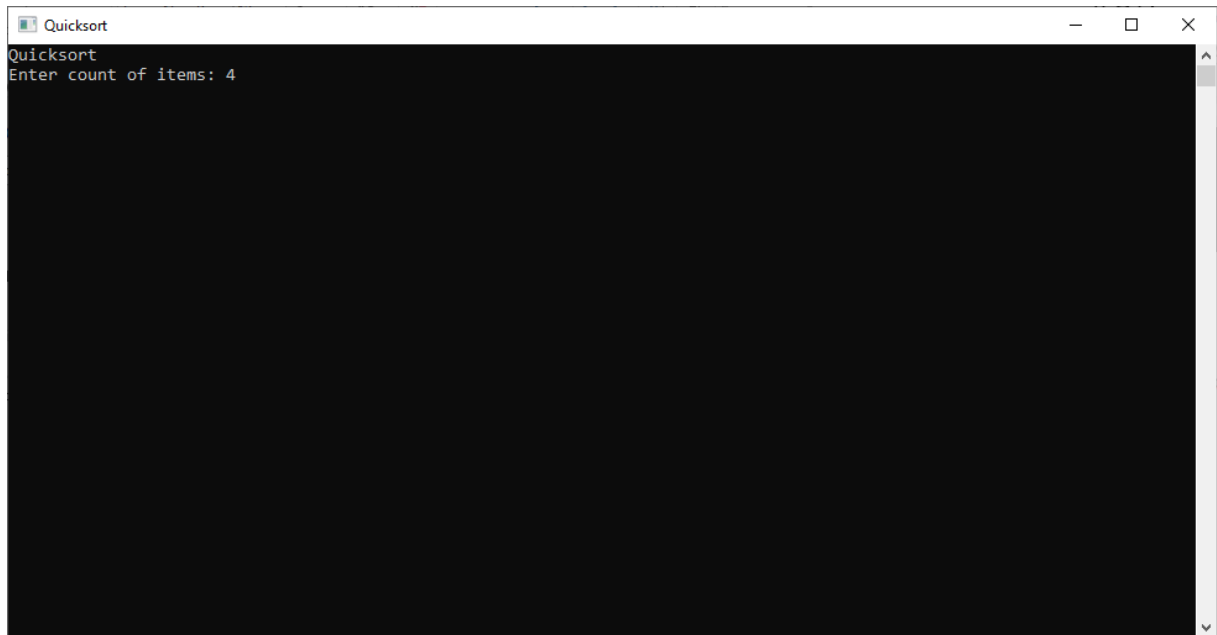
Describing of the application's operation

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.



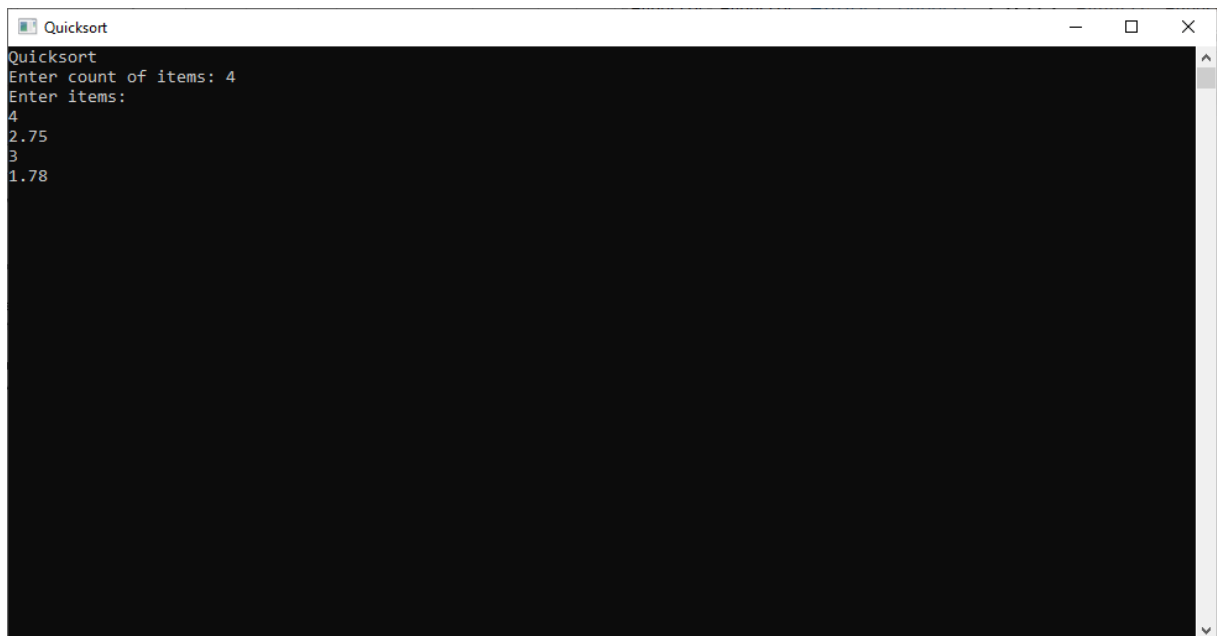
Drawing 1: The beginning of the application's operation [own study]

The user enters the number of elements in the set.



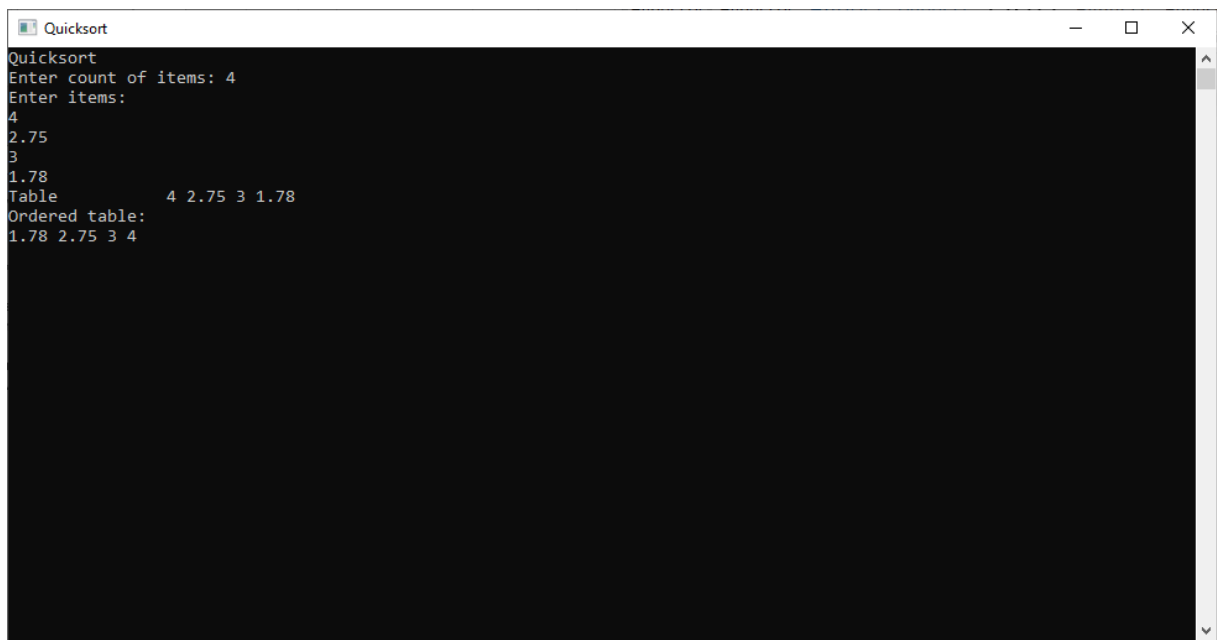
Drawing 2: Entering the number of items [own study]

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 3: Entering the values of all elements [own study]

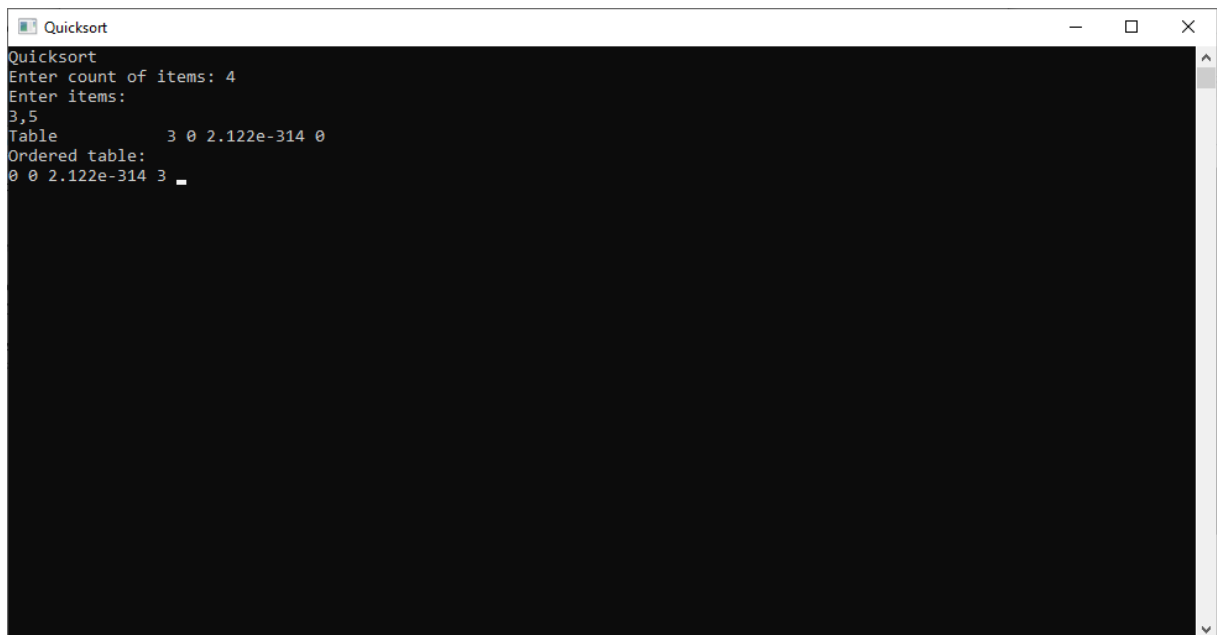
After entering the numbers, the application sorts the set after clicking enter. This is what the result looks like. Finally, the application waits for any key to be clicked.



```
Quicksort
Enter count of items: 4
Enter items:
4
2.75
3
1.78
Table      4 2.75 3 1.78
Ordered table:
1.78 2.75 3 4
```

Drawing 4: The final result [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.



```
Quicksort
Enter count of items: 4
Enter items:
3,5
Table      3 0 2.122e-314 0
Ordered table:
0 0 2.122e-314 3
```

Drawing 5: 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 quicksort. The quicksort algorithm is considered to be the fastest algorithm for random data.

The principle of its operation is based on the method of divide and conquer.[1]

The separator is selected from the array, and then the array is divided into two fragments: all elements not larger than the separator are transferred to the initial one, and all larger ones to the final one. Then, the beginning and the end of the table are sorted separately [1]. Recursion ends when the next fragment obtained from splitting contains a single element, as a single-element array does not require sorting.

If l denotes the index of the first, and r denotes the last element of the sorted array fragment, and i denotes the index of the element on which the array was divided, then the sorting procedure can be (simplified) represented by the following pseudocode.[2]

```
PROCEDURE Quicksort (table, l, r)
```

```
  Begin
```

```
    IF  $l < r$  THEN {if the fragment is longer than 1 element}
```

```
      Begin
```

```
         $i := \text{SplitArrays}(\text{array}, l, r)$ ; {divide and remember the  
split point}
```

```
        Quicksort (array, l,  $i-1$ ); {sort left part}
```

```
        Quicksort (array,  $i + 1$ , r); {sort right part}
```

```
      END
```

```
    END
```

```
{selects the element to be used for split
```

```
and it moves all the smaller items to the left of
```

```
of this element and elements greater than or equal to the right  
from selected item}
```

```
PROCEDURE Split Arrays (array, l, r)
```

```
  Begin
```

```
    DividendIndex: = Select DividendPoint (l, r); {select an  
element that will be used to split the array}
```

```
    dividendvalue: = array [dividendindex]; {remember element  
value}
```

```
    Replace (array, DivisionIndex, r); {move the break element to  
the end of the array so it doesn't participate in the break itself}
```

```

        currentPosition: = 1;
        FOR i: = 1 TO r-1 DO {iterate through all elements if the
element is less than the value of the split element add it to the
"left" side}
        Begin
            IF array [i] <divisionValue THEN
                Begin
                    Replace (array, i, currentPosition);
                    currentLocation: = currentLocation + 1;
                END
            END
        Replace (array, currentPosition, r); {insert the split element
in the right place}
        return current position;
    END

    {basic implementation of split point selection - selects "middle"
element in array}
    PROCEDURE Select the Breakpoint (l, r)
    Begin
        return 1 + (r-1) div 2;
    END

    {swaps items in cells i1, i2}
    PROCEDURE Replace (array, i1, i2)
    Begin
        IF i1 <> i2 THEN
            Begin
                aux: = array [i1];
                array [i1]: = array [i2];
                array [i2]: = aux;
            END
        END
    END

```

Listing 1: Pseudocode [2]

For proper operation of the algorithm, the SplitArray procedure must leave the pivot at the end of the left section obtained from the split. Then this element is in its final position and is not processed further - recursive Quicksort calls do not include the index i. Thus, one less element is passed for further processing, which guarantees the termination of recursion.[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 SelectTheIntervalPoint(int l, int r){
    return l + (r-l) / 2;
}

void Swap(double *table, int i1, int i2){
    if (i1!=i2){
        double aux = table[i1];
        table[i1] = table[i2];
        table[i2] = aux;
    }
}

int SplitArray(double *table, int l, int r){
    int splitindex = SelectTheIntervalPoint(l, r);
    double splitvalue = table[splitindex];
    Swap(table, splitindex, r);

    int currentposition = l;
    for(int i=l;i<=r-1;i++){
        if(table[i] < splitvalue){
            Swap(table, i, currentposition);
            currentposition++;
        }
    }
    Swap(table, currentposition, r);
    return currentposition;
}

void Quicksort(double *table, int l, int r){
    if(l < r){
        int i = SplitArray(table, l, r);
        Quicksort(table, l, i-1);
        Quicksort(table, i+1, r);
    }
}
```



```

    }
}

int main(){
    SetConsoleTitleA("Quicksort");
    //int option;
    cout<<"Quicksort"<<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;

        //sorting
        Quicksort(numbers,0,sizet-1);
        //end
        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 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: Entering the number of items [own study].....	4
Drawing 3: Entering the values of all elements [own study].....	4
Drawing 4: The final result [own study].....	5
Drawing 5: User error [own study].....	5

List of listings

Listing 1: Pseudocode [2].....	6
Listing 2: Source code [own study].....	8

Bibliography

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