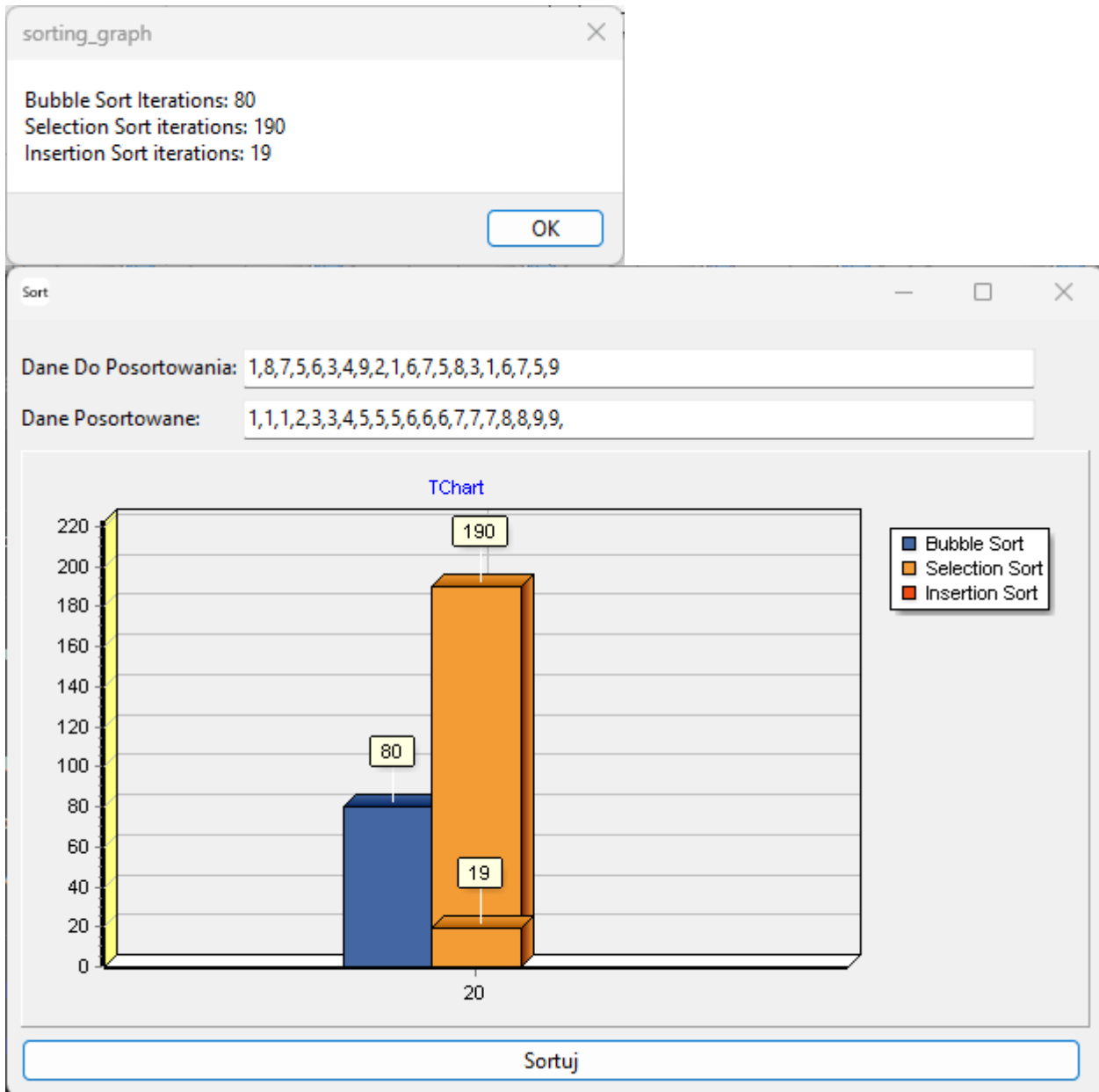


Porównanie wydajności sortowań



pliki:

- [sorting_graphs.zip](#)

kod:

[main.cpp](#)

```
//-----  
-----  
  
#include <vcl.h>  
#pragma hdrstop  
  
#include "Unit2.h"  
#include <string>
```

```
//-----  
-----  
#pragma package(smart_init)  
#pragma resource "*.dfm"  
TForm2 *Form2;  
//-----  
-----  
__fastcall TForm2::TForm2(TComponent* Owner)  
    : TForm(Owner)  
{  
    //AllocConsole();  
    //freopen("CONIN$", "r", stdin);  
    //freopen("CONOUT$", "w", stdout);  
    //freopen("CONOUT$", "w", stderr);  
}  
//-----  
-----  
int iterationsBubble;  
  
int* bubbleSort(int arr[], int size) {  
    iterationsBubble = 0;  
  
    for (int i = 0; i < size; ++i) {  
        for (int j = 0; j < size - i; ++j) {  
            if (arr[j] > arr[j + 1]) {  
                // Swap elements if they are in the wrong order  
                int temp = arr[j];  
                arr[j] = arr[j + 1];  
                arr[j + 1] = temp;  
                iterationsBubble++;  
            }  
        }  
    }  
  
    //ShowMessage("Bubble Sort Iterations: " +  
    IntToStr(iterationsBubble));  
  
    return arr; // Return the sorted array  
}  
  
int iterationsSelection;  
  
int* selectionSort(int arr[], int size) {  
    iterationsSelection = 0;  
    for (int i = 0; i < size - 1; ++i) {  
        int minIndex = i;  
  
        // Find the index of the minimum element in the unsorted part  
        of the array  
        for (int j = i + 1; j < size; ++j) {  
            if (arr[j] < arr[minIndex]) {
```

```
        minIndex = j;
    }
    iterationsSelection ++;
}

// Swap the found minimum element with the first element
int temp = arr[i];
arr[i] = arr[minIndex];
arr[minIndex] = temp;
}

return arr; // Return the sorted array
}

int iterationsInsertion;

int* insertionSort(int arr[], int size) {
    iterationsInsertion = 0;
    for (int i = 1; i < size; ++i) {
        int key = arr[i];
        int j = i - 1;

        // Move elements greater than key to
        //one position ahead of their current position
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }

        arr[j + 1] = key;
        iterationsInsertion++;
    }

    return arr; // Return the sorted array
}

/*
String intArrayToString(int arr[], int size) {
    String result = "";

    for (int i = 0; i < size; ++i) {
        if (i > 0) {
            result += ",";
        }
        result += IntToStr(arr[i]);
    }

    return result;
}
*/
```

```
void __fastcall TForm2::SortButtonClick(TObject *Sender)
{
    String strData = DataTextBox->Text;
    std::cout << AnsiString(strData).c_str() << std::endl;

    int counter = 0;

    for (int i = 1; i <= strData.Length(); i++) {
        if (strData[i] == ',') {
            counter++;
        }
    }

    std::cout << AnsiString(counter+1).c_str() << std::endl;

    // Dynamically allocate memory for intDataArray
    int* intDataArray = new int[counter + 1];

    // Initialize the array elements to avoid garbage values
    for (int i = 0; i <= counter; ++i) {
        intDataArray[i] = 0; // You can use any default value here
    }

    int numElement = 0;
    String element = "";
    int x = 0;
    for (int i = 1; i <= strData.Length(); i++) {
        x = i;
        if (strData[i] == ',') {
            numElement++;
            intDataArray[numElement] = StrToInt(element);

            std::cout << AnsiString("element:"+element).c_str() <<
std::endl;
            element = ""; // Reset element for the next iteration
        }
        else if (i == strData.Length()) {
            element += strData[i];
            numElement++;
            intDataArray[numElement] = StrToInt(element);

            std::cout << AnsiString("element:"+element).c_str() <<
std::endl;
            element = ""; // Reset element for the next iteration
        }
        else {
            element += strData[i];
        }
    }

    int arraySize = counter + 1;
```

```
int* sortedArray = bubbleSort(intDataArray, arraySize);
SortedDataTextBox->Text = "";
for (int i = 1; i <= arraySize; i++) {
    SortedDataTextBox->Text += IntToStr(sortedArray[i]);
    SortedDataTextBox->Text += ",";
}

selectionSort(intDataArray, arraySize);
insertionSort(intDataArray, arraySize);
ShowMessage("Bubble Sort Iterations: " + IntToStr(iterationsBubble)
+ "\nSelection Sort iterations: "+IntToStr(iterationsSelection)+
"\nInsertion Sort iterations: "+IntToStr(iterationsInsertion));

Series1->AddXY(arraySize,iterationsBubble);
Series2->AddXY(arraySize,iterationsSelection);
Series2->AddXY(arraySize,iterationsInsertion);

//String arrayString = intArrayToString(sortedString, arraySize);

//SortedDataTextBox->Text = arrayString;

// Don't forget to release the dynamically allocated memory
delete[] intDataArray;
}
//-----
-----
```