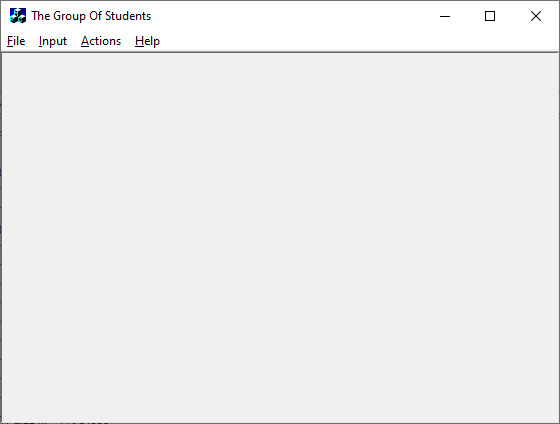
Data arrays lab functional specification

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comment** |
| **1** | 11.20.19 | Miroshnichenko Denys | Version 1.0 |

**TARGET WORKFLOW (Data arrays lab)**

UI of an app should look like this:

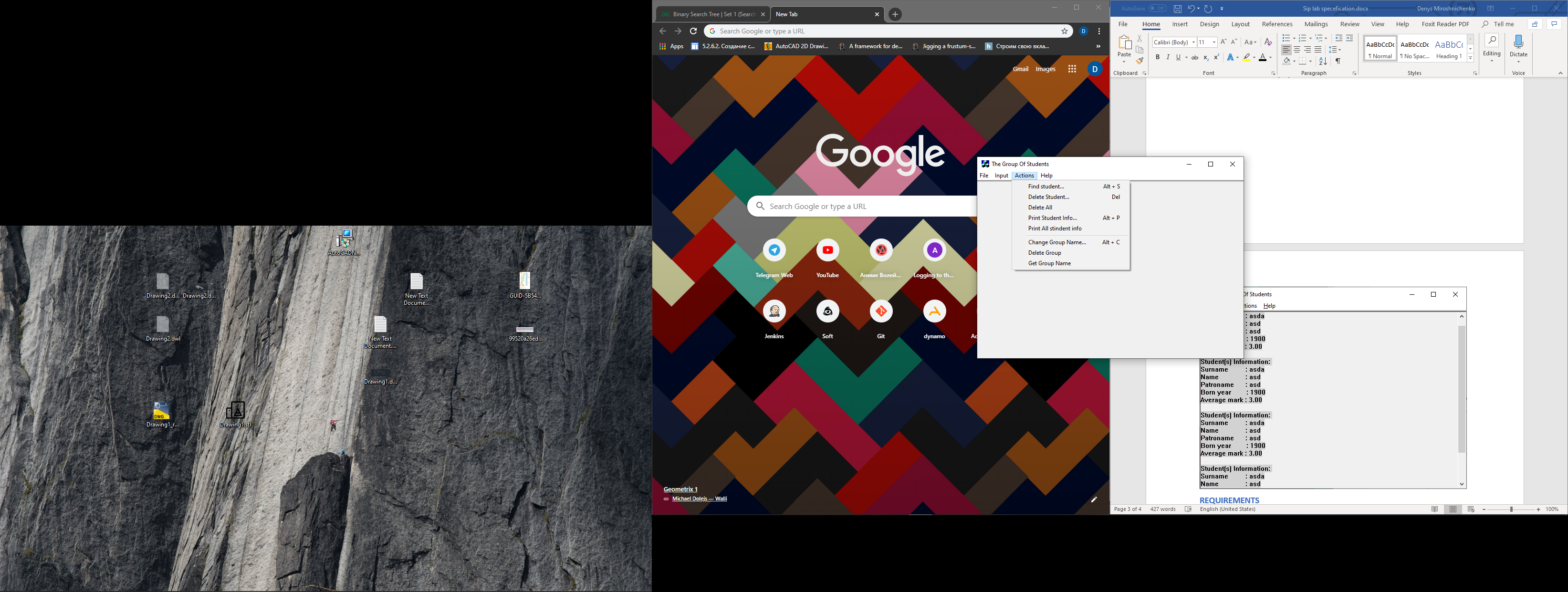


User can choose external file (txt/json/xml) as data source, or add data manually, using interface

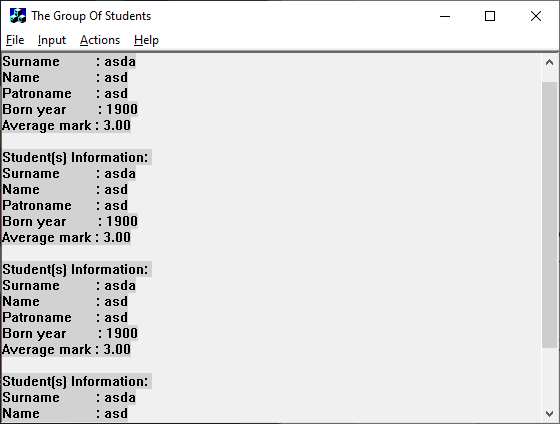
If data will be inputted manually, user firstly must enter group name, then add amount of students in this group (one by one)

|  |  |
| --- | --- |
|  |  |

Then user can choose from proposed features:



Output info for all student should look like this:



**REQUIREMENTS**

Selected Data Structure type: **binary tree**;

1. Separate key value from data value (tree is built by the key values)
2. Implement the following functionality:
   1. Add element to structure (by key)
   2. Find element (by key)
   3. Find element (by value, moreover: user must choose between wide search and deep search)
   4. Delete element (by key)
   5. Delete element (by value, moreover: user must choose between wide search and deep search)
   6. Delete all elements from structure
   7. Copy structure, using overloaded assignment operator
   8. Iterate over all structure elements and apply some operation to them (using wide and deep iteration methods)
3. Moreover class must contain pointers, which is necessary for work with data structure and which must be inaccessible for users.

Element is set by itself value part. During work class mustn’t give access to element of data structure, only to its value part (use copy of it, or pointer to it).

Structure of value part is considered unknown. (recommended to use **typedef** directive for data value type)

DO NOT use console input/output in class methods

1. Implement a class, which will be used as value part of element/ This class should contain the following information for every student:
   1. Surname
   2. Name
   3. Patronymic
   4. Birth year
   5. Average mark
2. Implementation of class should contain the following features:
   1. Lack of direct access to class members
   2. Safe setting of field value (with memory allocation, if necessary)
   3. Safe memory deallocation (when destroying student data)
   4. Properties through which access to the class members is implemented
   5. Overload operators
      * =
      * == (based on surname, name and patronymic)
      * != (based on surname, name and patronymic)
      * >= (based on surname, name and patronymic)
      * <= (based on surname, name and patronymic)
      * < (based on surname, name and patronymic)
      * > (based on surname, name and patronymic)
      * Input/output, using streams
3. The data structure should be implemented, using template
4. Implement ‘group’ and ‘stream’ (we are talking about groups of students) classes
5. Both of this classes should contain:
   1. Group/Faculty name (is not accessible directly to the user)
   2. Pointer(s) required(e) for working with data structure, that stores information about a group of students (is not accessible directly to the user)

\*Information about group is deleted from the student data

1. Develop UI for project, using WinForms, UI should contain: menu, dialog boxes and accelerators

**DEVELOPMENT**

**Stages**

Stage\_1

Implementation features 1-3

**Recommendations for stage 1:**

Collect items from each level into two-direction linear list to implement wide search. To implement this recommended to add pointers on left/right element (from one level) and store one-direction linear list of start element of each level.

**Requierements for stage 1:**

1. Implement tree structure, which is optimized for wide search (it means, that structure (tree) must have connection not only from parent to left and right children, but between nodes in one layer (each tree level will be represented as double linked list))
2. Moreover, we will store first (leftmost) node from each level to another list
3. Requirements for list of leftmost elements:
   1. Values, that stored in the list must be any integer non-zero type (int, long, smallint)
   2. Implementation of **AddToEnd**, **Replace**, **RemoveLast** and **Replace** functions

Stage\_2

Implementation features 4-6

Stage\_3

Implementation features 7-8

Stage\_4

Implement feature 9

**TECHNOLOGIES**

**Development tools**

Visual Studio Community 2019