B. Tasks for laboratory work 2 semester

Alexei Martynov

September 9, 2019

version 3.0

1 vectors

must perform *all* tasks.

The work must be performed as a one executable file receiving parameters as follows:

```
$ ./Lab number [args]
```

number represents point number, such as 1 or 2. As args The additional parameters are transmitted, depending on the item.

1. Write a sorting algorithm (any simple) collection of integers obtained from of standard input, so that:

   (A) vector Sorting was performed using an operator **operator [];**

   (B) sorting the vector was carried out using the method std :: vector <> :: at ().

   (C) sorting linked list iterators performed using [March, Sec. 16.3.1]. The end of the input is necessary to consider the state End Of File (EOF, end of the file). The program must be run with an additional parameter, depending on the value of which, sorting is done in ascending or descending order:

   ascending Ascending;

   descending descending.

   Display the standard output of sorted collection, sharing elements of space. Kol lecture sorted every type of access, to be output on a separate line. For instance,

   ```
   $ Cat data 4 3 2 1

   $ ./Lab 1 ascending <data 1 2 3 4 1 2 3 4 1 2 3
   4
   ```

2. Read in the built-in array C contents of a text file whose name is passed in the parameter. Copy the data in the vector on one line of code (without loops and algorithms Standard Template Library (STL)).

   Display the contents of the standard output vector.

3. Write a program is stored in the vector integers derived from the standard input (input end is a number 0). Using iterators, remove all items, which are divided into 2 (without using STL algorithms), if the last number 1. If the last number 2 add after each number that is divisible by 3 three units, also using iterators. All changes must be made using iterators without using indexes. Display the standard output obtained after transformation collection of numbers, separated by space pure la.

4. Write a function **void** fillRandom (**double *** array, **int** size), fills an array of random GOVERNMENTAL values in the range of from - 1.0 to + 1.0. Fill using a predetermined function vector of a predetermined size and sort the contents (by any previously developed an algorithm modified to sort both integer and real numbers). Display the standard output of the source and sorted collection on a separate line, separated by spaces elements.

   The program should be run with two additional parameters:

   - sorting direction (ascending or descending);
   - vector resolution. For

   instance,

   ```
   $. / Lab ascending 3 4
   0.5 0.8 0.4
   0.4 0.5 0.8
   ```

2 sequences

must perform *all* tasks.
As well as job 1, the job must be done in a console program that takes as its only parameter setting item number.

1

Below is an interface class queue with priorities, which operates in the following ob- time:

1. In turn, may be added elements, each element adding assigned one of three priority levels (low, normal, high).

2. The elements of the queue are retrieved in accordance with their priorities (first retrieve items with a priority of high, then normal, then low), elements with the same priority are retrieved from the queue in the order they are received.

3. The queue can also be accelerated operation - all the elements of low priority, are at the time of acceleration in turn, increase its priority to high and "overtake" elements with priority normal. Below is the interface of this class:

```
typedef enum
{
        LOW,
        NORMAL,
        HIGH
} ElementPriority;

typedef struct
{
        std :: string name;
} QueueElement;

class QueueWithPriority
{
        QueueWithPriority ();

        ~ QueueWithPriority ();

        void PutElementToQueue ( const QueueElement & element,
                ElementPriority priority);

        QueueElement GetElementFromQueue ();

        void Accelerate ();
};
```

1. Recycle class so that it can process the elements of any type.

2. Correct the error in the interface, ensuring a secure interface for use in industrially large project.

3. **Implement revised class using `std :: list <>` or `std :: deque <>`. Explain the choice.**

4. Implement program that handles all of the strings to commands received from the standard input Nogo. Each row contains exactly one command. The following commands should be supported:

   - add <priority> <data>
     Adding an element to the queue with priority <priority> (low, normal or high).

- get

  Preparation of the next element in the queue in accordance with priority, are printed to standard output data element. If the queue is empty, it contains the string <EMPTY>.

- accelerate

  Changing the priority queue elements.

In the case of an unsupported or incorrectly formed command in *standard out* printed <INVALID COMMAND>.

2

Develop a program that:

1. Fills std :: list < **int**> values of 1 before 20 from the standard input, the list may contain from 0 before 20 values.
2. Displays the contents of the list in the following order: the first element, the last element, the second element, the penultimate element, the third element, etc.
3. In the case of incorrect data, the program should display an error message to standard error and exit with code 1.

For example, if the list contains:

```
1 2 3 4 5 6 7 8
```

then the output will look like

```
1 8 2 7 3 6 4 5
```

Tip: You can use recursion and bidirectional iterators.

Iteration 3

follow *all* job as a program that takes as its first parameter item number:

1

Write programmu- "phonebook", consisting of 2 components:

1. Component-book.

   Record (name and phone) must be stored in any container from the STL. The program should support the following operations:

   - Viewing the current record.
   - Go to the next record.
   - Go to the previous record.
   - Inserting records before / after viewing.
   - Replacing viewing records.
   - Inserting a record in the database end.
   - Forward / backward through • records.

   Please note that client code may be necessary to make reference to the different entries in the book at the same time.

   Remember that after the insertion and removal of the element iterators may be invalidated. Telephone represented as a sequence of numbers with no spacers.

2. The user interface that accepts commands from standard input one per line and outputting the results to standard output. The following commands should be supported:

- add number "name"

  Adding to the end of the recording. The quotation marks are not part of the name. It takes into account that the name can contain quotation marks and backslash (preceded by a backslash, as in the literals C ++ [Mar Table. 3.2]), but it can not contain a new line (for example, "Name \" Nick \ "Surname") .

- store mark-name new-mark-name

  Saves the current position of the tab with the name mark-name as a new tab with the name of new- mark-name. Name contains only characters of the English alphabet, numbers, and "minus" sign. Po- follows the launch of the program is available 1 tab with the name of current.

- insert before mark-name number "name" Adding entries before

  setting mark-name.

- insert after mark-name number "name" Adding an entry after the

  laying of mark-name.

- delete mark-name

  Deleting an entry pointed to by the bookmark mark-name. After removing the tab it is indicated to the next item.

- show mark-name

  Showing records pointed to a bookmark mark-name. If there are no entries (empty book) is displayed <EMPTY>. The output should be performed without overhead sequences, in particular, a string of Example add command to be outputted as a Name "Nick" Surname.

- move mark-name steps

  Move bookmarks mark-name elements by steps. If steps positive, forcibly relocated forward tab, otherwise - back. Also in keywords first and last can be used as steps, meaning the first and the last record, respectively. If the setting is not the number of steps and a reserved keyword in *standard out* a message is displayed <INVALID STEP>.

UI job ends when the EOF or IO errors. In the event of an error, the return code must be equal to 2. In the case of an incorrect command, you must bring in *standard out* line <INVALID COMMAND> and continue. If you pass the bookmark name does not exist in *standard out* output line <INVALID BOOKMARK>.

2

Implement the following classes:

- "Container" which contains the values of the factorial 1! before 10!.

  class interface should include as a minimum:

  - The default constructor.
  - A function of acquiring iterator pointing to the first element of the container - begin ().
  - A function of acquiring an iterator pointing to the element following the last - end ().

  Access to the elements of the container is only possible using iterators returned by the function tions begin () and end ().

  The container should not be stored in its memory elements, they must be calculated when accessed through an iterator.

- Iterator class for listing of this container, this class of objects are returned to the functions begin () and end (). Iterator must be bidirectional. Iterator must be compatible with the STL.

  iterator category requirements must be met.

- Print content "container" in two rows in the standard output: the first line in the forward directivity, the second - in reverse using std: copy (). It must be remembered that prevent the use of std :: reverse_iterator <> in the client code is not possible.

4 Algorithms I

Write a program that performs the following actions:

1. Fills std :: vector <DataStruct> structures DataStruct, Read from standard input. Each line comprises successively key1, key2 and str, separated by commas, str It continues to the end line. key1 and key2 in the range of - 5 to + 5.
2. Sort the vector as follows: (a) Ascending key1.

   (B) If key1 are the same, ascending key2.

   (C) Where key1 and key2 identical, the ascending line length str.
3. Outputs the resulting vector printing.

DataStruct defined as follows: 1 **struct** DataStruct

```
{2 3
        int              key1;
4       int              key2;
5       std :: string str;
6};
```

5 Algorithms II

This task must be done in a single console program. Selection of reference etsya continued the implementation using the first parameter at startup.

1

Run:

1. Reading the contents of a text file, transmitted via standard input.
2. Isolation of words, a word is a sequence of characters separated by spaces and / or knowledge kami tabs and / or newline characters.
3. Make a list of words (one word per line) that are present in the text without repetition (meaning that the same word can be listed only once).

2

Using standard algorithms to perform the following steps:

1. Fill a container geometric shapes by reading them from stdin. This point should be made a separate action, further actions with the data reading can not be combined.
2. Count the total number of vertices of all shapes (triangle so it adds to the total number 3 square 4 etc.).
3. Calculate the number of triangles, squares and rectangles.
4. Remove all pentagons.
5. On the basis of the remaining data to create std :: vector <Point>, which comprises coordinates one of the vertices (of any) of each figure, i.e. the first element of this vector contains the coordinates of a vertex of the first shape, the second element of this vector contains the coordinates of a vertex of the second figure, etc.
6. Change the container so that it is contained in the beginning of all the triangles, then all the squares, and then rectangles.

7. Derive the standard output results of the form:

```
Vertices: 16 Triangles: 1
Squares: 1 Rectangles:
2

Points: (5, 5) (10; 10) (1; 1) Shapes:

3 (1: 1) (2, 2) (3: 1)
4 (10; 10) (10; 11) (11; 11) (11; 10) 4 (5, 5) (7, 5), (7, 6) (5, 6)
```

EXAMPLE 1: Example for output 5.2

Geometrical figure defined by the following structure: 1 **struct** Point

```
{2 3
        int x, y;
4 }; 56 using Shape = std :: vector <Point>;
```

Each point is defined by a coordinate pair (*•; •),* figures indicate one per line, the first number sets the number of vertices, and then specify vertex points:

```
3 (1, 3) (23; 3) (15; 8)
```

The figures do not include self-intersections, checking the condition is not required. If all the points of the figures are the same, the figure should be regarded as the most stringent version of the polygon.

Tip: In addition to the algorithms discussed in this paper can be used all the means described in previous studies, including sorting algorithms.

6 Functors I

Develop a functor, which allows to collect statistics about the sequence of integers. Functor sequence after processing algorithm std :: for_each must provide the following statistics:

1. Maximum number in the sequence.
2. The minimum number in the sequence.
3. Mean numbers in sequence.
4. The number of positive numbers.
5. Negative numbers.
6. The sum of the odd elements of the sequence.
7. The sum of the elements of even order.
8. coincide if the first and last elements of the sequence.

Check the work, writing a program that takes a set of numbers from standard input and prints out the statistics in the following format:

```
Max: 124 Min: -784365 Mean:
765,65 Positive: 15 Negative: 24
Odd Sum: 123 Even Sum: 87464
First / Last Equal: no
```

EXAMPLE 2: Example for output 6

If not in the list of values that you want to display «No Data» line.

7 Functors II

This task must be done in a single console program. Selection of reference etsya continued the implementation using the first parameter at startup.

1

Using only standard algorithms and functor, multiply each element of a list of numbers in floating point number π. The numbers must be read from the standard input, the results are derived for the standard output.

2

1. To implement a hierarchy of geometric shapes consisting of: (a) Class Shape, comprising:

   - information on the situation shapes the center (coordinates • and •);
   - method isMoreLeft (), allows to determine whether this is located to the left of the figure (determined by the position of the center) than the figure passed as an argument;
   - method isUpper, allows to determine whether this figure is higher (determined by the position of the center) than the figure passed as an argument;
   - pure virtual function draw draw () (Each figure in the implementation of this function should output tion in the transmitted stream of

   its name and the center position). (B) Class Circle, derived from the class Shape.

   (C) Class Triangle, derived from the class Shape.

   (D) Class Square, derived from the class Shape.
2. Fill a list of pointers to the various figures, read them from standard input.
3. Using standard algorithms and adapters withdraw all figures.
4. Use of standard algorithms and adapters, sort the list by the center position of the left- to right (meaning that at the beginning of the list should go to the left of the figure are) and bring the figure result.
5. Using standard algorithms and sort the list of adapters on the position of the center and left sprava- display figures.
6. Using standard algorithms and sort the list of adapters on the top-center position and to bring down the figure.
7. Using standard algorithms and sort the list of adapters on the position of the center and bottom-up displays figures.

Figures data entry are set to one row, the figure type is determined keyword (CIRCLE, TRIANGLE and SQUARE), after which the coordinates of the center with brackets. Conclusion GOVERNMENTAL lists various note strings «Original:», «Left-Right:», «Right-Left:», «Top-Bottom:» and «Bottom-Top». Shapes are displayed one per line in the same format as that of input ()

```
CIRCLE (1; 2) TRIANGLE
(-5; 10) SQUARE (15; 5)
CIRCLE (-3; -3)
```

Example 3: Example I for 7.2

```
Original: CIRCLE (1; 2)
TRIANGLE (-5; 10) SQUARE
(15; 5) CIRCLE (-3; -3)
Left-Right: TRIANGLE (-5;
10)
```

```
CIRCLE (-3; -3) CIRCLE (1;
2) SQUARE (15; 5)
Right-Left: SQUARE (15; 5)
CIRCLE (1; 2) CIRCLE (-3;
-3) TRIANGLE (-5; 10 )
Top-Bottom: CIRCLE (-3; -3)
CIRCLE (1; 2) SQUARE (15;
5) TRIANGLE (-5; 10)
Bottom-Top: TRIANGLE (-5;
10) SQUARE (15; 5) CIRCLE
(1; 2) CIRCLE (-3; -3)
```

EXAMPLE 4: Example for output 7.2

8 text

Develop a program that needs to do the following:

1. Read the text from standard input, which may contain:

   (A) The words - are made up of lowercase and uppercase letters, and hyphen "-", the word length must be no more than 20 characters.

   (B) The words can end with a hyphen, but can not start with him.

   (C) Punctuation -, ",", ":", ";", "---" (3 dashes, hyphens equivalent) "." "!" "?". signs pre-kicking can not go one by one, except for a dash: can occur comma before the dash. *Capable of processing and other punctuation marks, in accordance with the way they understand user preferences.*

   (D) The numbers composed of the digits and the decimal point and optional sign of ("+" And "-"), separation of groups of digits are not used, the length of - less than 20 characters. (E) Whitespace - space, tab, newline. (F) text *not* It may start with punctuation.

2. Format the text as follows:

   Not (a) must be whitespace characters other than a space. (B) It should not go more than one space in a row.

   (C) between the word and punctuation mark should be no space except a dash which allocated space on both sides. (D) must always keep a space after punctuation. (E) is not allowed to transfer the dash to the next line.

3. Convert the resulting text strings in a set, each of which contains an integer number of words and numbers (each element must be entirely in a row) and its length does not exceed the number of symbols defined parameter. Thus each line should contain the maximum number of words that should not end by whitespace and start with punctuation.

4. Print the resulting set of rows on the standard output.

The program should take an optional argument --line-width with a parameter indicating a long string in characters well. If not specified, the length of the string is taken 40 characters. Thus, the program can be started:

```
$ ./Lab --line-width 70
```

This gives the line length 70 characters. Or this:

```
$ ./lab
```

What gives the default value 40 characters.

All additional information, such as the decimal separator, you must obtain from the user settings using the standard library [1].

[1] When using [MinGW] support options, other than the «C» and «POSIX» is required.