Workshop #6: Classes and resources, IO operators
Learning Outcomes
Submission Policy
Due Dates
Late penalties
Citation
For work that is done entirely by you (ONLY YOU)
For work that is done partially by you.
If you have helped someone with your code
Compiling and Testing Your Program
Custom code submission
LAB (100%) The Numbers Module
streaming on ostream
The Numbers class
Constructors, Destructor and Copy Operations (Construction and Assignment)
One argument constructor
Default constructor
Destructor
Copy Constructor
Copy Assignment Operator
Copying procedure

Workshop #6: Classes and resources, IO operators

- Version 0.9 (Submission is not open yet)

In this workshop, you will implement copy constructor and copy assignment to prevent memory-leak and resize allocated memory.

Learning Outcomes

Upon successful completion of this workshop, you will have demonstrated the abilities to:

- define and create copy constructors (Rule of three)
- define and create copy assignment (Rule of three)
- Read and write from and to text files
- Resize dynamically allocated memory.
- Overload insertion operator so the class can be printed using ostream
- Overload extraction operator so the class can be read using istream

Submission Policy

The workshop is divided into one coding part and one non-coding part:

- Part 1 (**LAB**): A step-by-step guided workshop, worth 100% of the workshop's total mark
  > Please note that the part 1 section is **not to be started in your first session of the week**. You should start it on your own before the day of your class and join the first session of the week to ask for help and correct your mistakes (if there are any).
- Part 2 (reflection): non-coding part. The reflection doesn't have marks associated with it but can incur a **penalty of max 40% of the whole workshop's mark** if your professor deems it insufficient (you make your marks from the code, but you can lose some on the reflection).

Due Dates

Workshop #6: Classes and resources, IO operators
Learning Outcomes
Submission Policy
Due Dates
Late penalties
Citation
For work that is done entirely by you (ONLY YOU)
For work that is done partially by you.
If you have helped someone with your code
Compiling and Testing Your Program
Custom code submission
LAB (100%) The Numbers Module
streaming on ostream
The Numbers class
Constructors, Destructor and Copy Operations (Construction and Assignment)
One argument constructor
Default constructor
Destructor
Copy Constructor
Copy Assignment Operator
Copying procedure

The Due dates depend on your section. Please choose the "-due" option of the submitter program to see the exact due date of your section:

```
~profname.proflastname/submit 2??/wX/pY_sss -due<ENTER>
```

- Replace **??** with your subject code (`00 or 44`)
- Replace **X** with Workshop number: [`1 to 10`]
- Replace **Y** with the part number: [`1 or 2`]
- Replace **sss** with the section: [`naa, nbb, nra, zaa, etc...`]

Late penalties

You are allowed to submit your work up to 2 days after the due date with a 30% penalty for each day. After that, the submission will be closed and the mark will be zero.

Citation

Every file that you submit must contain (as a comment) at the top:
your name, **your Seneca email**, **Seneca Student ID** and the **date** when you completed the work.

For work that is done entirely by you (ONLY YOU)

If the file contains only your work or the work provided to you by your professor, add the following message as a comment at the top of the file:

> I have done all the coding by myself and only copied the code that my professor provided to complete my workshops and assignments.

For work that is done partially by you.

If the file contains work that is not yours (you found it online or somebody provided it to you), **write exactly which part of the assignment is given to you as help, who gave it to you, or which source you received it from.** By doing this you will only lose the mark for the parts you got help for, and the person helping you will be clear of any wrongdoing.

> - Add the citation to the file in which you have the borrowed code

Workshop #6: Classes and resources, IO operators
- Learning Outcomes
- Submission Policy
- Due Dates
- Late penalties
- Citation
  - For work that is done entirely by you (ONLY YOU)
  - For work that is done partially by you.
  - If you have helped someone with your code
- Compiling and Testing Your Program
  - Custom code submission

LAB (100%) The Numbers Module
- streaming on ostream

The Numbers class
- Constructors, Destructor and Copy Operations (Construction and Assignment)
  - One argument constructor
  - Default constructor
  - Destructor
  - Copy Constructor
  - Copy Assignment Operator
    - Copying procedure

> - In the 'reflect.txt` submission of part 2 (DIY), add exactly what is added to which file and from where (or whom).

> ⚠️ This Submission Policy only applies to the workshops. All other assessments in this subject have their own submission policies.

If you have helped someone with your code

If you have helped someone with your code. Let them know of these regulations and in your 'reflect.txt' of part 2 (DIY), write exactly which part of your code was copied and who was the recipient of this code.
By doing this you will be clear of any wrongdoing if the recipient of the code does not honour these regulations.

Compiling and Testing Your Program

All your code should be compiled using this command on `matrix`:

```
g++ -Wall -std=c++11 -g -o ws file1.cpp file2.cpp ...
```

- `-Wall`: the compiler will report all warnings
- `-std=c++11`: the code will be compiled using the C++11 standard
- `-g`: the executable file will contain debugging symbols, allowing *valgrind* to create better reports
- `-o ws`: the compiled application will be named `ws`

After compiling and testing your code, run your program as follows to check for possible memory leaks (assuming your executable name is `ws`):

```
valgrind --show-error-list=yes --leak-check=full --show-leak-kinds=all --track-origin
```

- `--show-error-list=yes`: show the list of detected errors
- `--leak-check=full`: check for all types of memory problems
- `--show-leak-kinds=all`: show all types of memory leaks identified (enabled by the previous flag)
- `--track-origins=yes`: tracks the origin of uninitialized values (`g++` must use `-g` flag for compilation, so the information displayed here is meaningful).

Workshop #6: Classes and resources, IO operators

Learning Outcomes

Submission Policy

Due Dates

Late penalties

Citation

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

Compiling and Testing Your Program

Custom code submission

LAB (100%) The Numbers Module

streaming on ostream

The Numbers class

Constructors, Destructor and Copy Operations (Construction and Assignment)

One argument constructor

Default constructor

Destructor

Copy Constructor

Copy Assignment Operator

Copying procedure

To check the output, use a program that can compare text files. Search online for such a program for your platform, or use *diff* available on `matrix`.

> Note: All the code written in workshops and the project must be implemented in the **sdds** namespace, unless instructed otherwise.

Custom code submission

If you have any additional custom code, (i.e. functions, classes etc) that you want to reuse in the workshop save them under a module called Utils (`Utils.cpp and Utils.h`) and submit them with your workshop using the instructions in the "Submitting Utils Module" section.

LAB (100%) The Numbers Module

Your task for this lab is to complete the implementation of the **Numbers** class. This class, when created using a filename as its constructor argument, reads several double values from the file (assuming there is one number per line in the file) and holds them dynamically in memory. Then the caller application has the option of adding numbers to the collection.

streaming on ostream

If inserted into ostream this class, provides the following information about the numbers kept in the object:

- The Module can stream the values in ascending order (if sorted).
- The Module streams the largest, the smallest and the average of the values in the collection

Also, The **Numbers** class has the following capabilities:

- A **Numbers** object can safely be copied or assigned to another **Numbers** object; however, if copied, then the values kept in the copied object will not be written back into the file.
- A **Numbers** object can be displayed or streamed into an ostream object.
- A **Numbers** object can be extracted or read from an istream object.

When the **Numbers** object goes out of scope the double values kept in the object should overwrite the old values in the file, ONLY IF the object is the original object and not a copy.

Workshop #6: Classes and resources, IO operators
Learning Outcomes
Submission Policy
Due Dates
Late penalties
Citation
For work that is done entirely by you (ONLY YOU)
For work that is done partially by you.
If you have helped someone with your code
Compiling and Testing Your Program
Custom code submission
LAB (100%) The Numbers Module
streaming on ostream
The Numbers class
Constructors, Destructor and Copy Operations (Construction and Assignment)
One argument constructor
Default constructor
Destructor
Copy Constructor
Copy Assignment Operator
Copying procedure

The Numbers class

The Numbers class has at least the following five attributes:

- a double pointer to hold the address of the dynamic array of doubles that keeps the number read from the file
  *(we will call this array, the **Collection** from now on)*
- a 255-character long cString to hold the name of the file associated with the class
  *(we will call this cString, the **file name** from now on)*
- an integer (preferably unsigned) to hold the number of double numbers in the **Collection**.
  *(We will call this **Collection Size** from now on)*
- a Boolean flag to keep track if this object is the original or a copy.
  *(We will call this the **original flag** from now on)*
- a Boolean flag to keep track if any number was added to the collection.
  *(We will call this the **added flag** from now on)*

Constructors, Destructor and Copy Operations (Construction and Assignment)

The **Numbers** class can be instantiated in three different ways:

One argument constructor

Creates an original instance of the Numbers class by receiving the data file name (a Cstring) as an argument

- Sets the **file name** to the argument value
- calls the load() function

Default constructor

- Sets the object to a safe empty state.

Destructor

- Saves the file by calling the save() function
- deletes the **Collection** pointer

Workshop #6: Classes and resources, IO operators
Learning Outcomes
Submission Policy
Due Dates
Late penalties
Citation
For work that is done entirely by you (ONLY YOU)
For work that is done partially by you.
If you have helped someone with your code
Compiling and Testing Your Program
Custom code submission
LAB (100%) The Numbers Module
streaming on ostream
The Numbers class
Constructors, Destructor and Copy Operations (Construction and Assignment)
One argument constructor
Default constructor
Destructor
Copy Constructor
Copy Assignment Operator
Copying procedure

Copy Constructor

Creates a copy out of an already existing **Numbers** object.
The standard way of doing this is as follows:

- Make sure the class is in a safe empty state
- call the copy assignment operator to reuse your logic for copying the object.

Copy Assignment Operator

If this is not a self-copy, proceed with the copying procedure, otherwise, skip the whole process and end the operation

Copying procedure

- save the **Collection** calling the save() function. This will save the original values that are about to be replaced.
- delete the current **Collection**
- set the current object to the empty state; this will make sure the object remains empty if the source (right operand) is empty.
- if the right operand is in a good state (not empty)
  - sets the **original flag** to false (not original)
  - Allocates new memory for the **Collection** to the size of the right operand's **Collection size**
  - Copies all the double values of the right operand **Collection** to the newly allocated **Collection**.
  - updates the **Collection size**

Ending the operation

- in any case, return the reference of the current object.

Member Functions and Member operator+= overload

Private methods:

void sort(double* array, unsigned int size) (implemented)

sorts an array of doubles in ascending order.

> This could be moved to the Utils module if you have one!

- Workshop #6: Classes and resources, IO operators
  - Learning Outcomes
  - Submission Policy
  - Due Dates
  - Late penalties
  - Citation
    - For work that is done entirely by you (ONLY YOU)
    - For work that is done partially by you.
    - If you have helped someone with your code
  - Compiling and Testing Your Program
    - Custom code submission
- LAB (100%) The Numbers Module
  - streaming on ostream
- The Numbers class
  - Constructors, Destructor and Copy Operations (Construction and Assignment)
    - One argument constructor
    - Default constructor
    - Destructor
    - Copy Constructor
    - Copy Assignment Operator
      - Copying procedure

countLines() (implemented)

This function returns the number of lines in the **data file** (hence returning the number of double values in the file).

> Note that if the number returned by this method and the actual number of successful reads from the file don't match, the **data file** is corrupted and the object must be set back to an empty state (see the load() function) This method can not change the state of the current object.

setEmpty()

Sets all attributes to null, zero, false or empty Cstring.

load()

This function returns true if all the double values are read from the **data file** and stored in the **Collection**, otherwise returns false.

- Delete the current **Collection**
- Call the countLines() and get the number of lines in the file.
- If the number of lines is greater than zero (the file is not empty)
  - Allocate memory to the number of lines (that is what the number of double values in the file should be) and keep the address in the **Collection** pointer.
  - Create an instance of ifstream for reading the data file using the **file name**
  - While the ifstream object is in a good state, keep reading double values from the file into the elements of the **Collection** (as you do with cin) and count the number of reads
  - If the number of lines and the number of successful reads do not match discard all the read and set the object back to an empty state as mentioned in countLines() function. Otherwise set the **Collection size** to the number of double values read and set the **original flag** to true.
- At the end return true if at least one number was read successfully

save()

- If the current object is an original and new values are added to it, then
  - Create an instance of ofstream to overwrite the data file (using the **file name**)
  - Set the precision of the floating point numbers to 2 digits after the decimal point when writing the number into the file.
  - Write all the elements of the **Collection** using the ofstream object (as you do with cout).
  - Write a newline after each double value.

- Workshop #6: Classes and resources, IO operators
  - Learning Outcomes
  - Submission Policy
  - Due Dates
  - Late penalties
  - Citation
    - For work that is done entirely by you (ONLY YOU)
    - For work that is done partially by you.
    - If you have helped someone with your code
  - Compiling and Testing Your Program
    - Custom code submission
- LAB (100%) The Numbers Module
  - streaming on ostream
- The Numbers class
  - Constructors, Destructor and Copy Operations (Construction and Assignment)
    - One argument constructor
    - Default constructor
    - Destructor
    - Copy Constructor
    - Copy Assignment Operator
      - Copying procedure

max()

Returns the largest double number in the **Collection**

> This method can not change the state of the current object

min()

Returns the smallest double number in the **Collection**

> This method can not change the state of the current object

average()

Returns the average of the double numbers in the **Collection**.

> This method can not change the state of the current object

Pubic:

Boolean type conversion operator overload (bool cast overload)

Returns true if the **Numbers** object is in a good state (not empty) and false if it is not
(i.e. returns true if **Collection** pointer is not nullptr)

> This method can not change the state of the current object

sort()

calls the provided sort function to sort the **Collection** double array.

Operator +=

Overload the += operator to add a single double value to the list of numbers in the array and then return the reference of the current object, only if the **Numbers** object is not empty.

You need to increase the size of the allocated memory by one (add one double to the **Collection**), to be able to do this.

Here is the sequence of the actions to be taken to resize memory:

- if the current object is not empty

Workshop #6: Classes and resources, IO operators

Learning Outcomes

Submission Policy

Due Dates

Late penalties

Citation

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

Compiling and Testing Your Program

Custom code submission

LAB (100%) The Numbers Module

streaming on ostream

The Numbers class

Constructors, Destructor and Copy Operations (Construction and Assignment)

One argument constructor

Default constructor

Destructor

Copy Constructor

Copy Assignment Operator

Copying procedure

- Create a temporary local double-pointer and allocate memory with the increased size
- Copy all the current values from the **Collection** to the newly allocated memory.
- Update the size of the data to the new size
- Now that all the values are copied to the new memory, delete the original **Collection** pointer
- Set the original **Collection** pointer to point to newly allocated memory
- Update the **Collection size**

- return the reference of the current object.

View the Slides

display function. (to be implemented)

```
std::ostream& display(std::ostream& ostr = std::cout) const
```

Inserts the **file name**, the **Collection** numbers and the largest, smallest and average values into the `ostream` reference argument in the following format:

- if the object is empty, insert: `"Empty list"` .
- if the object is not empty
  - set the precision to print two digits after the decimal point.
  - if the **Numbers** object is not original insert `"Copy Of "`
  - insert the **file name**
  - New line
  - insert all the number in the **Collection** comma separated like the following example: `1.99, 99.99, 5.12`
  - New line
  - insert a line with 76 dashes (`-`).
  - insert the **Collection** information exactly as the following example: `Total of 99 number(s), Largest: 99.99, Smallest: 99.99, Average: 99.99`
  - New line
  - insert a line with 76 assignment signs (`=`).
- return the reference of the ostream.

Helper Functions

Workshop #6: Classes and resources, IO operators

- Learning Outcomes
- Submission Policy
- Due Dates
- Late penalties
- Citation
  - For work that is done entirely by you (ONLY YOU)
  - For work that is done partially by you.
  - If you have helped someone with your code
- Compiling and Testing Your Program
  - Custom code submission

LAB (100%) The Numbers Module

- streaming on ostream

The Numbers class

- Constructors, Destructor and Copy Operations (Construction and Assignment)
  - One argument constructor
  - Default constructor
  - Destructor
  - Copy Constructor
  - Copy Assignment Operator
    - Copying procedure

- Overload the insertion operator so a **Numbers** object can be inserted into an ostream object.
- Overload the extraction operator so a **Numbers** object can be extracted an istream object. (without using `freind` statement)
  > hint: you can use the operator+= overload

Tester Program

main.cpp

Execution sample

correct_output.txt

Files to submit:

```
Numbers.h
Numbers.cpp
main.cpp
add.txt
badFileFormat.txt
number.txt
```

Data Entry

Follow the instruction in the tester program.

Submission Process:

Upload the files listed above to your `matrix` account. Compile and run your code using the `g++` compiler as shown in Compiling and Testing Your Program and make sure that everything works properly.

Then, run the following command from your matrix account

```
~profname.proflastname/submit 2??/wX/pY_sss  <ENTER>
```

- Replace **??** with your subject code (`00 or 44`)

Workshop #6: Classes and resources, IO operators
Learning Outcomes
Submission Policy
Due Dates
Late penalties
Citation
For work that is done entirely by you (ONLY YOU)
For work that is done partially by you.
If you have helped someone with your code
Compiling and Testing Your Program
Custom code submission
LAB (100%) The Numbers Module
streaming on ostream
The Numbers class
Constructors, Destructor and Copy Operations (Construction and Assignment)
One argument constructor
Default constructor
Destructor
Copy Constructor
Copy Assignment Operator
Copying procedure

- Replace **X** with Workshop number: [`1 to 10`]
- Replace **Y** with the part number: [`1 or 2`]
- Replace **sss** with the section: [`naa, nbb, nra, zaa, etc...`]

and follow the instructions.

Submitting Utils Module

To have your custom Utils module compiled with your workshop and submitted, add a **u** to the part number of your workshop (i.e **u**p1 for part one and **u**p2 for part two) and issue the following submission command instead of the above:

```
~profname.proflastname/submit 2??/w#/upX
```

See Custom Code Submission section for more detail

> ⚠**Important:** Please note that a successful submission does not guarantee full credit for this workshop. If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Re-submissions will attract a penalty

Part 2: Reflection

Study your final solutions for each deliverable of the workshop **and the most recent milestones of the project**, reread the related parts of the course notes, and make sure that you have understood the concepts covered by this workshop. **This should take no less than 30 minutes of your time and the result is suggested to be at least 150 words in length.**

Create a file named `reflect.txt` that contains your detailed description of the topics that you have learned in completing this workshop and **the project milestones** and mention any issues that caused you difficulty.

Submission Process:

Upload the files listed above to your `matrix` account. Compile and run your code using the `g++` compiler as shown in Compiling and Testing Your Program and make sure that everything works properly.

Then, run the following command from your matrix account

Workshop #6: Classes and resources, IO operators

Learning Outcomes

Submission Policy

Due Dates

Late penalties

Citation

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

Compiling and Testing Your Program

Custom code submission

LAB (100%) The Numbers Module

streaming on ostream

The Numbers class

Constructors, Destructor and Copy Operations (Construction and Assignment)

One argument constructor

Default constructor

Destructor

Copy Constructor

Copy Assignment Operator

Copying procedure

```
~profname.proflastname/submit 2??/wX/pY_sss  <ENTER>
```

- Replace **??** with your subject code (`00 or 44`)
- Replace **X** with Workshop number: [`1 to 10`]
- Replace **Y** with the part number: [`1 or 2`]
- Replace **sss** with the section: [`naa, nbb, nra, zaa, etc...`]

and follow the instructions.