<https://github.com/Milonov-Grigory> - ссылка на репозиторий

1 лаба (опсание в ридми):  
<https://github.com/Milonov-Grigory/kmbo-19>

2 лаба (перегрука нью и делит + аллокатор):  
<https://github.com/Milonov-Grigory/operators_new_delete>1. Реализовать свой аллокатор, который ограничивает количество объектов некоторым максимальным числом, задаваемым в конструкторе аллокатора.

2. Перегрузить для некоторого класса оператор new и delete

3 лаба (парсер)

<https://github.com/Milonov-Grigory/inheritance>

- Разработать иерархию полиморфных классов — парсеров для разных типов данных. (целое число, строковая константа, идентификатор, арифметический оператор) Парсер должен выделять лексему из входного потока и с помощью **полиморфной** функции проверять ее на соответствие типу. Если тип соответствует, то лексема запоминается и выдается по соответствующему аксессору.

- Разработать класс «парсер выражения», как производный от класса «парсер», являющийся контейнером для вложенных парсеров.

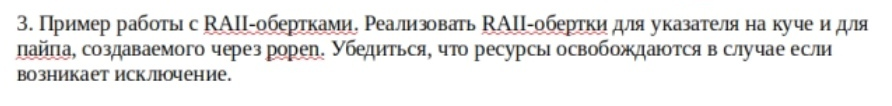
- Продемонстрировать работу парсеров выражений, например, типа арифметический многочлен. В результате работы должен выводиться вердикт о соответствии входного потока шаблону.

4 лаба(обработка исключений) дедлайн 18 апреля 00:00

<https://online-edu.mirea.ru/mod/assign/view.php?id=37037>

(зайти на портал oline-edu.mirea.ru в раздел МиС, там найдёте материалы лекции)

***ВНИМАНИЕ!!!***



Милонов дал 3-ий пункт л.р. в конце семинара, здесь привожу то, что успел записать:

“3. Сделать RAII-обёртки для указателя char на куче, для объекта pipe - примеры приведены в лекции. Для соответствующий примеров доделать следующий функционал:

* Управление владением(запрет копирования - реализуется путём добавления конструктора копирования в private класса; совместное владение - передача содержимого от одного объекта - инциализированного - не иниц-ому)
* Реализовать для объектов с совместным владением ресурсами возможность очистки памяти по указателю тогда и только тогда, когда удалён последний объект из тех, которые имеют доступ к ресурсу(в класс достаточно добавить счётчик, реализуемый на куче, и при копировании достаточно просто копировать его адрес, при удалении объекта достаточно заглядывать по указателю в этот счётчик и проверять, последний ли это объект с управлением данным ресурсом или нет)

5 лаба(Move-семантика)

<http://vega.fcyb.mirea.ru/dist/docs/92i298t-rvalues_and_move_semantic.pdf>

1. В файле, прикреплённом выше, есть код, к которому ЖЕЛАТЕЛЬНО подописывать конструкторы копирования перемещением и посмотреть, что будет
2. **ОБЯЗАТЕЛЬНО:** дописать конструктор и оператор для поддержки MOVE-семантики к односвязному списку(л.р.№1), придумать сценарий, где НАГЛЯДНО будет видна реализация MOVE-семантики и отличия в использовании памяти от Copy-семантики. Наглядность обеспечивается использованием ресурса valgrind massif(valgrind.org - здесь лежит вся документация с примерами), с помощью которого строятся графики по потреблению памяти программой при выполнении придуманного сценария. Пример работы с valgrind-massif см. В записях практики от 18.04
3. СПОСОБ СДАЧИ обновлённого связанного списка: ещё раз форкнуть репозиторий со связанным списком на гитхабе, сохранить как “Лабораторная работа по MOVE-семантике”(или что-нибудь в этом духе), там сделать все грязные дела и отправить от неё pull-request преподавателю.

6 лаба(hash-table - шаблоны)

<https://github.com/Milonov-Grigory/cxx_labs/tree/master/hash_table>

Реализация chained hash table с итератором внутри.