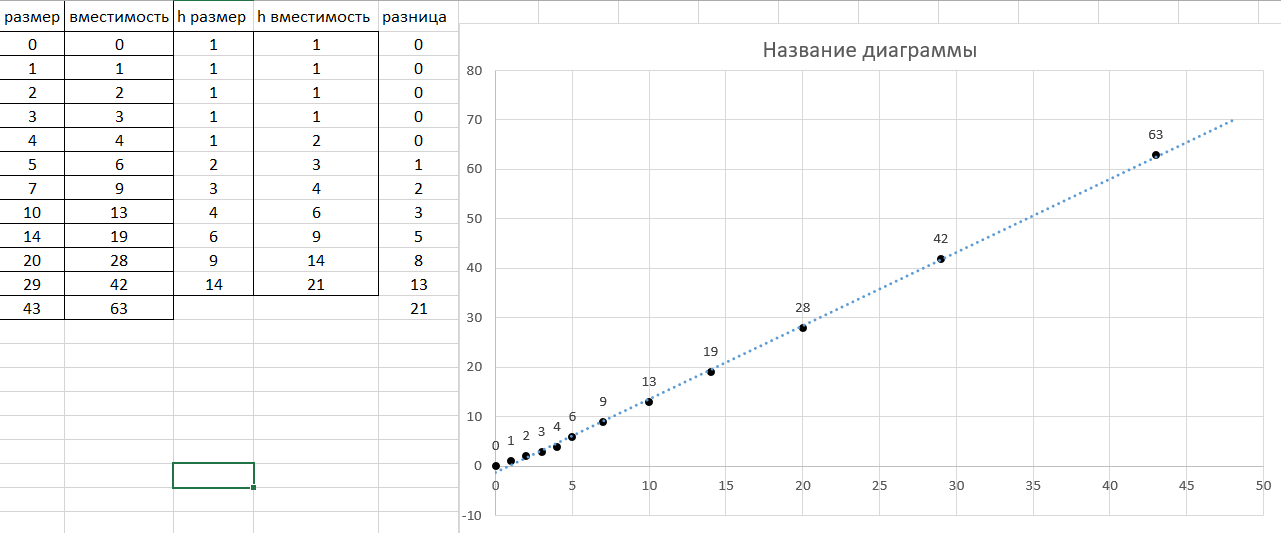
# Отчет “Задание 7. STL” Вариант-2.

Знакомство с контейнерами STL. Приемы управления памятью. Изучение поведения умных указателей shared\_ptr.

* 1 Часть.

В процессе выполнения данного задания был создан динамический массив целых чисел. По заданию необходимо было 32 раза добавить в конец массива элемент равный 1, при этом размер массива никогда не превосходил его вместимости. Вместимость массива увеличивается в тот момент, когда размер превосходит ее, и увеличивается на половину вместимости в настоящий момент времени (при этом округляется в меньшую сторону).

6. “К чему приводит такой способ управления динамической памятью? Например, сколько раз происходило бы перевыделение памяти, если бы vector был реализован “наивно” – не хватает, увеличим на 1, а мы бы выполнили push\_back 1000 раз? А сколько должно получиться в случае применения обнаруженного нами правила?”

В случае «наивной» реализации перевыделение памяти происходило бы 999 раз, в нашем случае же это произойдет лишь 18 раз.

* 2 Часть

В процессе выполнения был создан умный указатель shared\_ptr, а также объявлены две функции foo и goo, которые возвращают умный и простой указатели соответсвенно, далее поочередно присвоили результат этих функций переменной типа shared\_ptr<Test>. В случае вызова функции foo, по завершении блока вызывался деструктор. В случае goo в отличии от умного указателя, по завершению кода у нас не был вызван деструктор. Далее нужно было заключить строки с вызовом функции в дополнительный блок. У функции foo при выделении в отдельный блок, получилось так, что и конструктор, и деструктор были вызваны внутри созданного блока.

Следующим пунктом было: «Добавьте перед блоком объявление shared\_ptr<Test> t; а в блоке присвойте переменной t значение p.   
Снова выполните программу и объясните результат.»

В этом же случае деструктор вызван только по завершении выполнения программы. Можно предположить, что связанно это с тем, что мы объявили объект t вне блока.