Министерство образования Республики Беларусь

Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ

Кафедра интеллектуальных информационных технологий

Отчёт по лабораторной работе №3
по курсу «ОС» на тему:
«Изучение алгоритмов распределения памяти»

Выполнил студент группы 921703: Василевский Артемий Дмитриевич

Проверил: Садовский Михаил Ефимович

Минск 2021

Содержание

1. Цель и задача

Цель лабораторной состоит в изучении алгоритмов распределения памяти, а также реализации одного из них.

Вариант 2

Задача: реализовать менеджер памяти с перемещающимися блоками.

2. Использованные методы

2.1. Общая архитектура

Менеджер памяти — программа, которая способна выделять, освобождать память, а также записывать и читать из этой памяти.

Память в менеджере представлена набором блоков. В общем случае, блок — кусок памяти заданного размера. Однако, часть блоков имеют специальный байт заголовка, в котором описывается служебная информация данного блока. Именно такой блок описывается указателем ptr. Служебная информация может быть различной, но каждый блок должен соблюдать следующий контракт:

1. По указателю можно записать максимум столько байт, сколько было выделено данному указателю (то есть, нельзя записать соседний блок).
2. По указателю можно прочесть максимум столько байт, сколько было выделено данному указателю (то есть, нельзя прочесть соседний блок).
3. Указатель при любых изменениях среды будет указывать на один и тот же блок. Однако, реальный адрес памяти (адрес операционной системы) может меняться.
4. Блок должен хранить информацию о правах чтения и записи:
    1. Прочесть можно только из блока, который доступен на чтение
    2. Записать можно только в блок, который доступен на запись

Указатель представлен структурой, которая хранит адрес блока памяти, а также номер ошибки для случаев, если этот указатель больше не может быть использован.

2.2. Особенности реализации

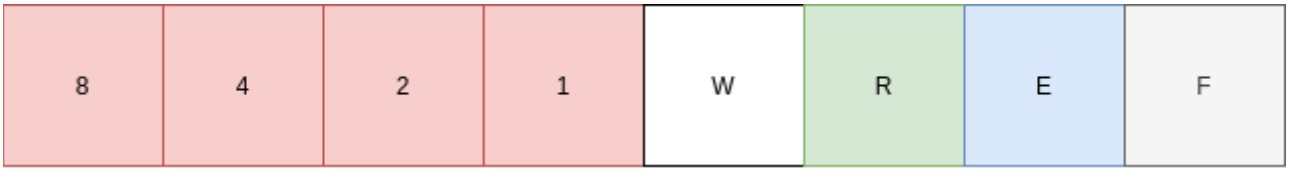
В менеджере памяти хранится указатель на начальный блок, а также указатель на текущий блок. Кроме этого, в менеджере хранится размер одного блока и количество блоков. Данные характеристики задаются при инициализации менеджера.

Указатель состоит из указателя на блок памяти и целого числа, описывающего номер ошибки. Возможные ошибки описаны в файле memager.h. Блок памяти, который хранится в указателе, представлен только заголовком — однобайтовой переменной, которая содержит (начиная с младшего):

1. Бит занятости(F). Описывает, занят ли данный блок: 1 — занят, 0 — свободен.
2. Бит расширенности(E). Описывает, является ли данный блок расширенным. При аллокации указателя можно указать количество байт, которое не поместится в один блок. Тогда реальный размер выделенной памяти будет кратен размеру одного блока. 1 — расширенный блок, 0 — одиночный блок.
3. Бит прав на чтение(R). 1 — из блока можно читать, 0 — из блока читать нельзя.

4. Бит прав на запись(W). 1 — в блок можно писать, 0 — в блок писать нельзя.
5. 4 бита размера расширенного блока. Если было выделено больше байт, чем может поместиться в одном блоке, блок становится расширенным, и представлен в памяти несколькими блоками (заголовок есть только у первого). Как можно заметить, максимально можно аллоцировать объект, который не превышает размера 16 блоков памяти (минус байт заголовка). Если необходимо разместить больше, следует инициализовать память с блоками большего размера.

В результате, байт заголовка выглядит следующим образом:



2.3. Описание работы функций

Менеджер памяти может выполнить следующие функции:

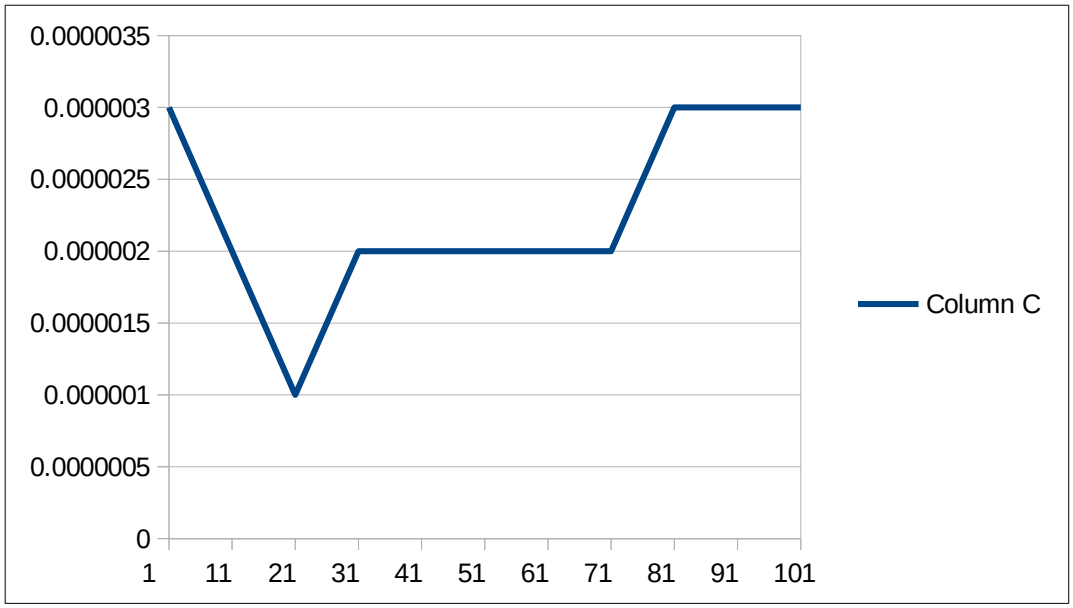
1. Инициализация. Запрос у операционной системы некоторого количества памяти, которой потом будет управлять менеджер. Единственный метод, который работает с операционной системой, вызывая метод calloc. В метод инициализации необходимо передать 2 параметра: размер блока и количество блоков. Также во время инициализации создаются внутренние управляющие структуры (указатель на начальный блок, указатель на текущий блок)
2. Выделение памяти. Менеджер памяти помечает некоторое количество блоков как занятые и возвращает указатель на блок памяти с требуемым объёмом. Стоит заметить, что выделенный объём будет кратен размеру блока, так как атомарной единицей памяти является блок (так, например, если размер блока 32 байта, то при выделении даже одного байта будет использоваться целый блок 32 байта. Однако, затем в программе все 32 байта можно будет использовать (исключая байт заголовка)).
3. Освобождение памяти. Функция освобождения принимает указатель на память, которую нужно освободить. Освобождение проходит в два этапа: сначала блок помечается как свободный, а затем происходит проверка, является ли данный блок последним в памяти. Если блок последний, происходит глубокая очистка — предыдущие блоки проверяются на используемость. Если предыдущий блок пуст, указатель текущего блока смещается. И так пока указатель текущего блока не встретит занятый блок или не переместится в начало. Но в ситуации, когда освобождаемый блок зажат между сверху занятым блоком, глубокая очистка не происходит (в данном случае можно произвести дефрагментацию, чего нет в задании).

4. Запись данных в память. При вызове функции записи данных, необходимо передать ей указатель на память, куда записывать, далее указатель на данные из памяти операционной системы, которые нужно записать. И в конце количество байт, которые нужно записать. При записи происходит побайтовое копирование данных из буфера в память.
5. Чтение данных происходит аналогично.

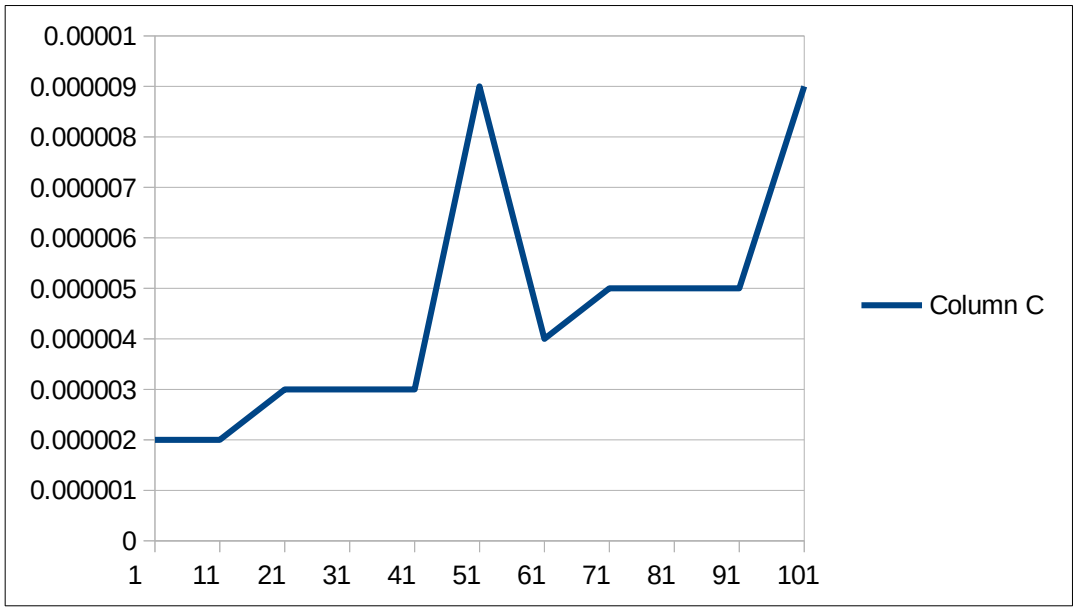
3. Графики

Здесь представлены графики, полученные при нагрузочном тестировании менеджера памяти.

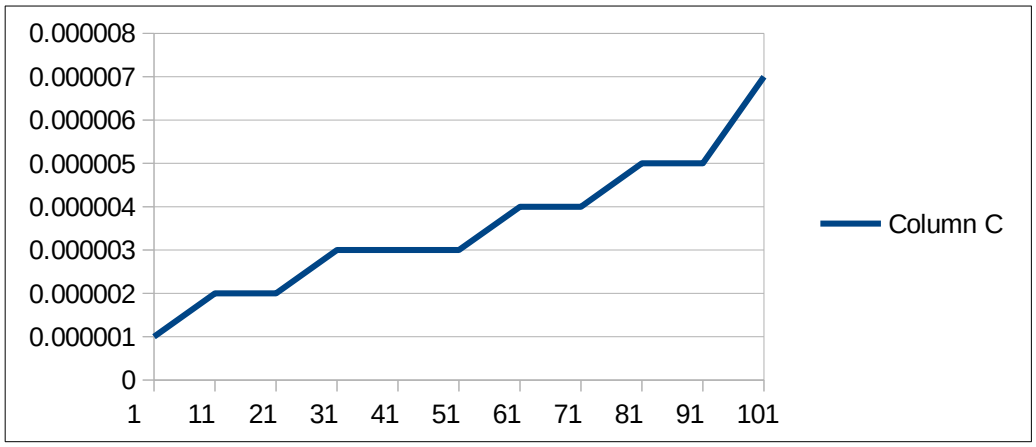
3.1. Размер блока = 1



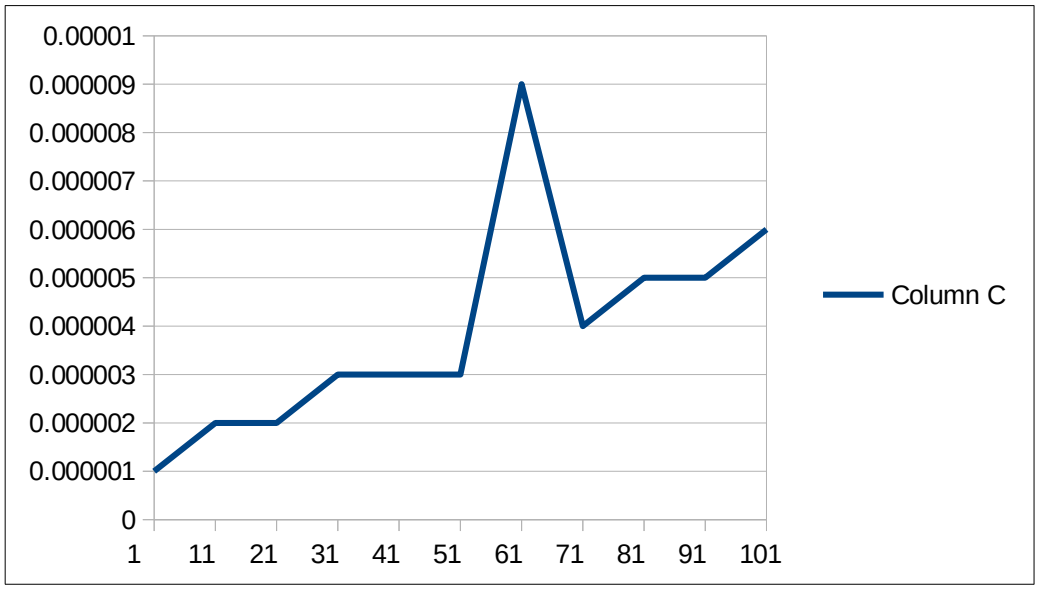
3.2. Размер блока = 11



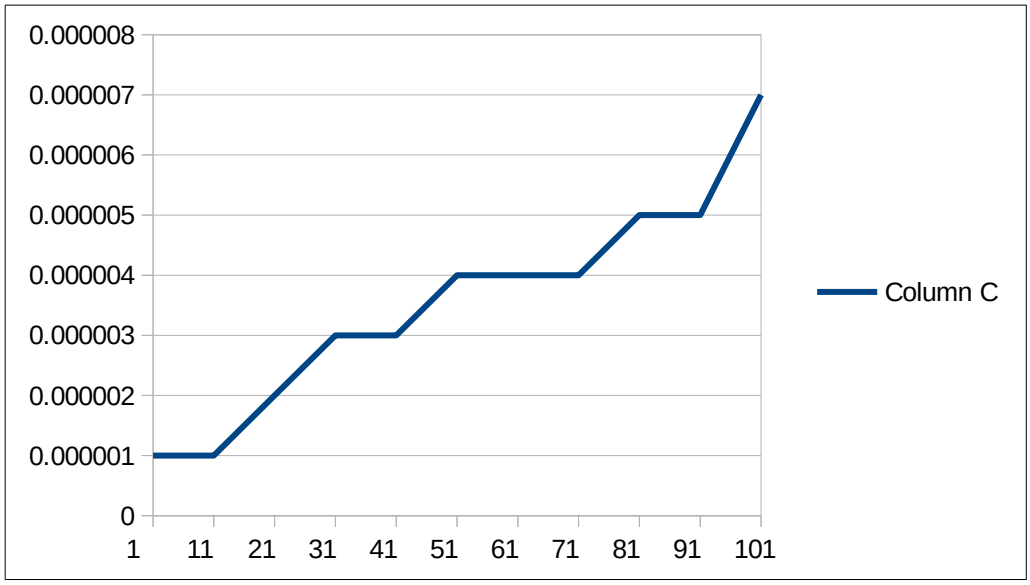
3.3. Размер блока = 21



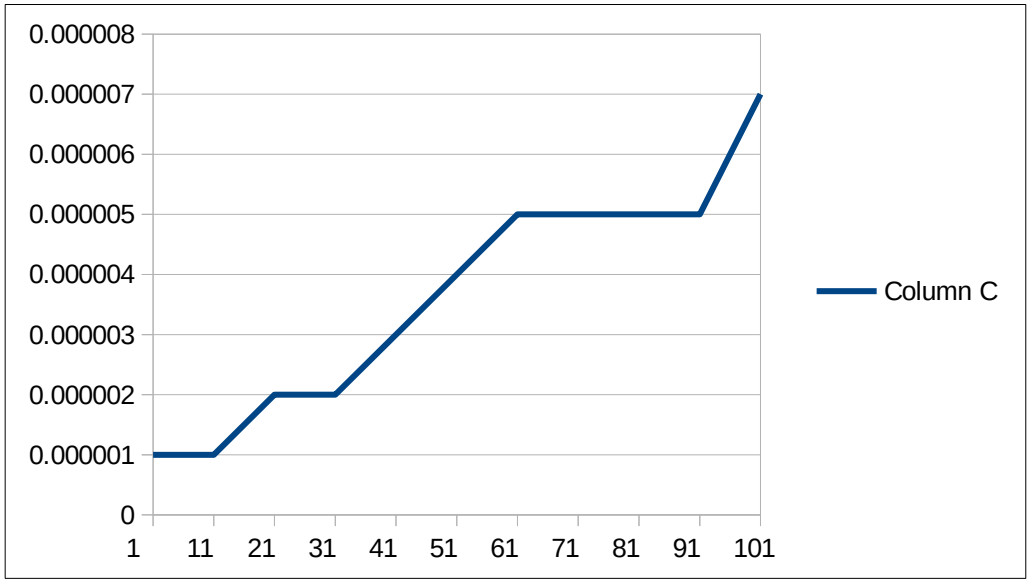
3.4. Размер блока = 31



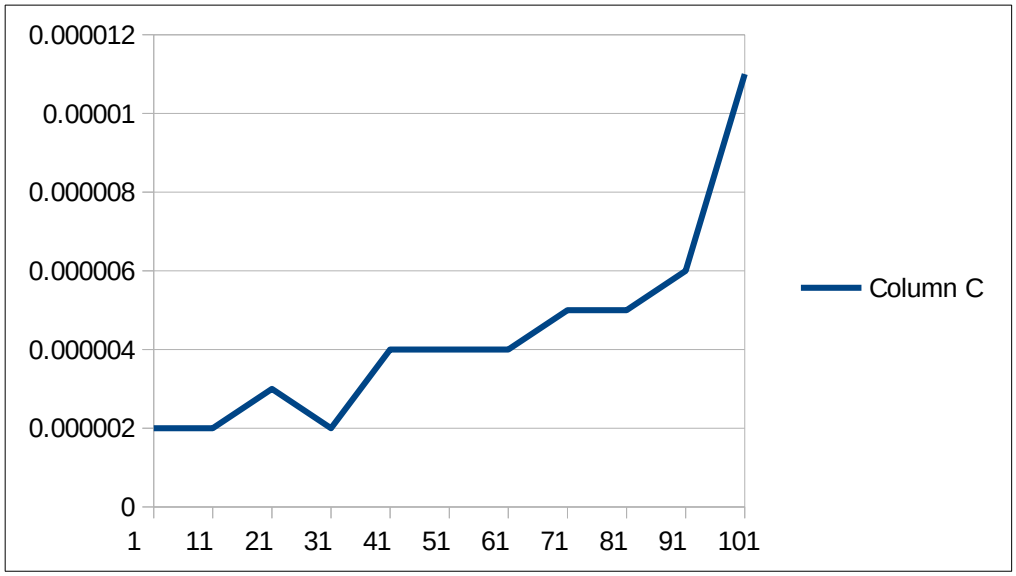
3.5. Размер блока = 41



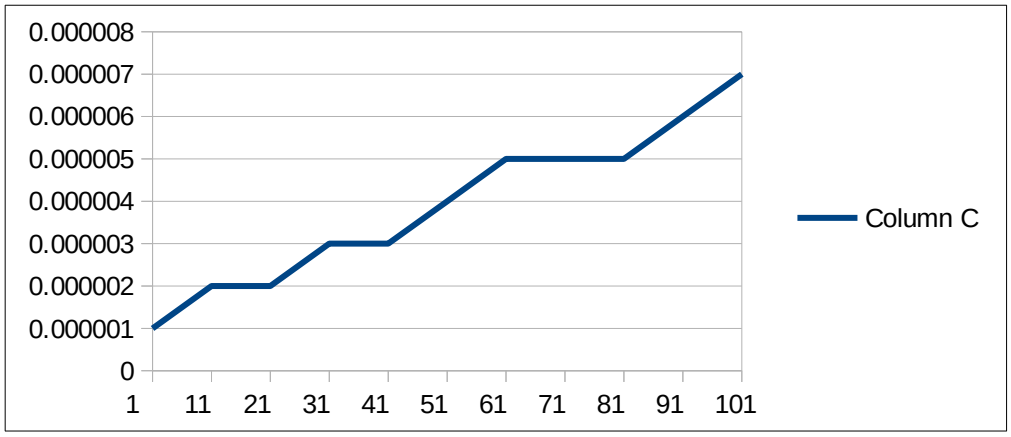
3.6. Размер блока = 51



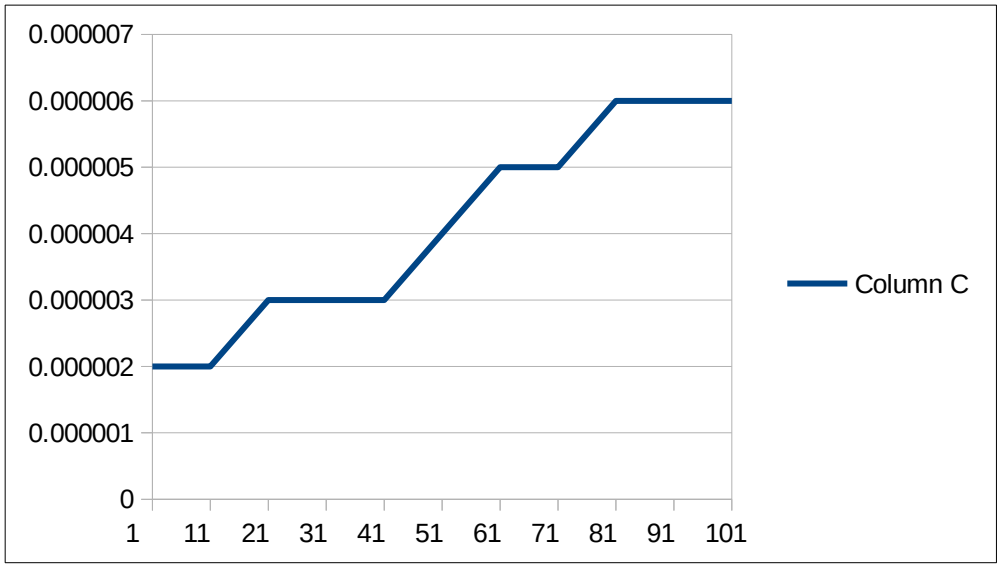
3.7. Размер блока = 61



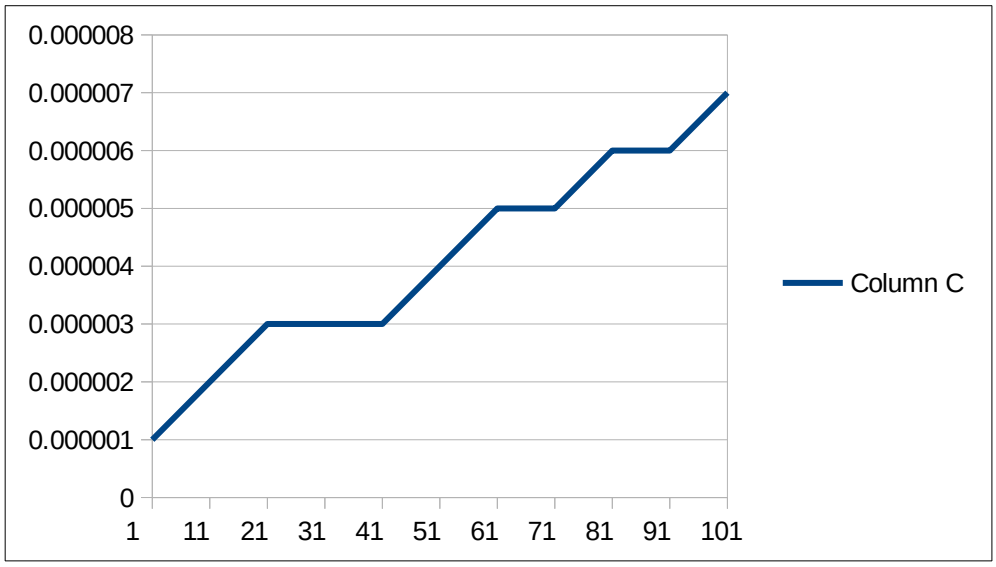
3.8. Размер блока = 71



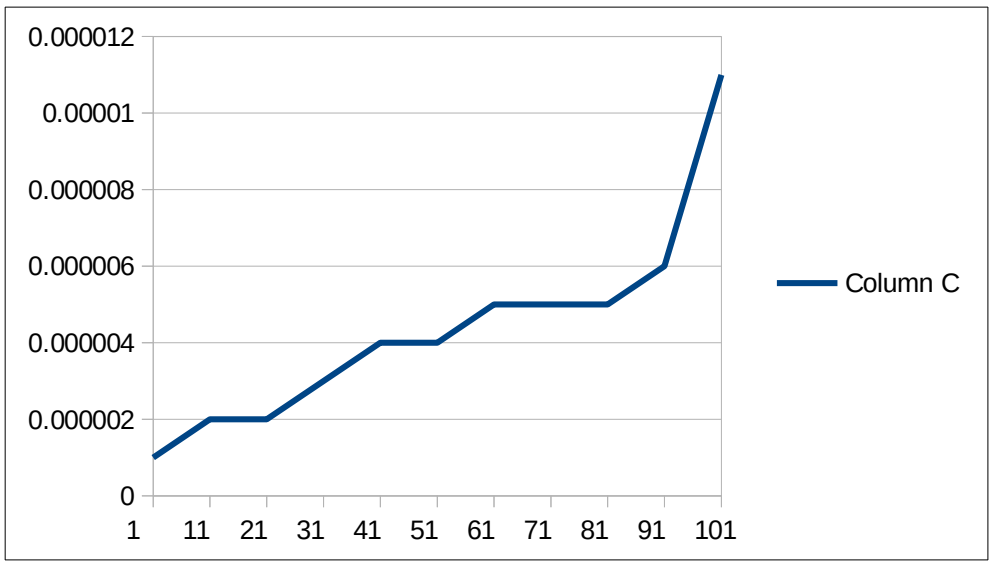
3.9. Размер блока = 81



3.10. Размер блока = 91



3.11. Размер блока = 101



3.12. Результаты исследования

Из графиков можно сделать вывод, что время выделения памяти линейно зависит от количество выделенной памяти.

Недостатком выбранной модели менеджера памяти является необходимость глубокой очистки и другие проблемы, вызванные фрагментацией памяти. Также недостатком является то, что вне зависимости от количества записанных данных, блок будет иметь фиксированный размер, который задан при аллокации. Но данный минус может давать выигрыш в производительности в некоторых ситуациях, когда для программы действительно нужно обсчитать большую пачку данных, которые находятся в одном блоке. Тогда процессору не придётся делать несколько запросов к памяти. Данная оптимизация широко используется и называется выравнивание.

4. Вывод

В данной лабораторной работе были рассмотрены варианты алгоритмов распределения памяти, а также реализован один из вариантов — блочный менеджер памяти. Модель менеджера была покрыта юнит-тестами для демонстрации надёжности программы. Кроме этого, проведено нагрузочное тестирование, по результатам которого построены соответствующие графики и сделаны выводы.