Программирование

Петрусевич Денис Андреевич

Доцент Кафедры Высшей математики

petrusevich@mirea.ru

Аллокаторы памяти

Проблемы выделения памяти с помощью malloc/new:

- Фрагментация памяти
- Относительно плохая производительность
- Ошибки, наподобие heap corrupt
- Как отслеживать, какая память и где используется?
- Каждый раз выделяя память, мы можем получить новый участок, располагающийся «далеко» от уже имеющихся

Аллокаторы

Выделение непрерывного участка памяти, из которого можно выдавать участки памяти для работы

Основные операции

- *create* – создает аллокатор и отдает ему в распоряжение некоторый объем памяти;
- *allocate* – выделяет блок определенного размера из области памяти, которым распоряжается аллокатор;
- *deallocate* – освобождает определенный блок;
- *free* – освобождает все выделенные блоки из памяти аллокатора (память, выделенная аллокатору, не освобождается);
- *destroy* – уничтожает аллокатор с последующим освобождением памяти, выделенной аллокатору.

Аллокаторы

Собственный аллокатор позволяет отловить ошибки:

Выделение/освобождение чужой памяти

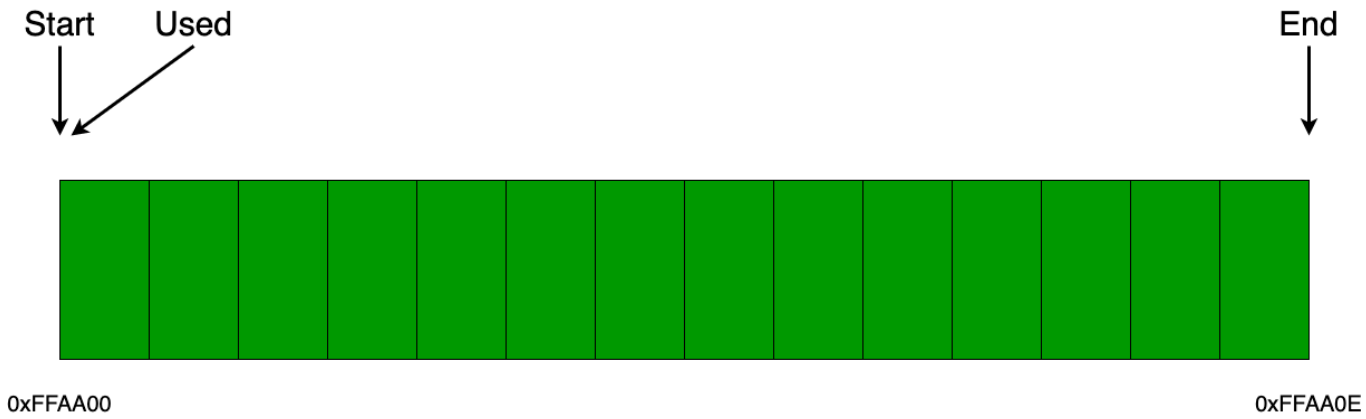
Обращение к чужой памяти

Повторное выделение используемой памяти

Линейный аллокатор

Начало и конец выделенного участка памяти – указатели start, end

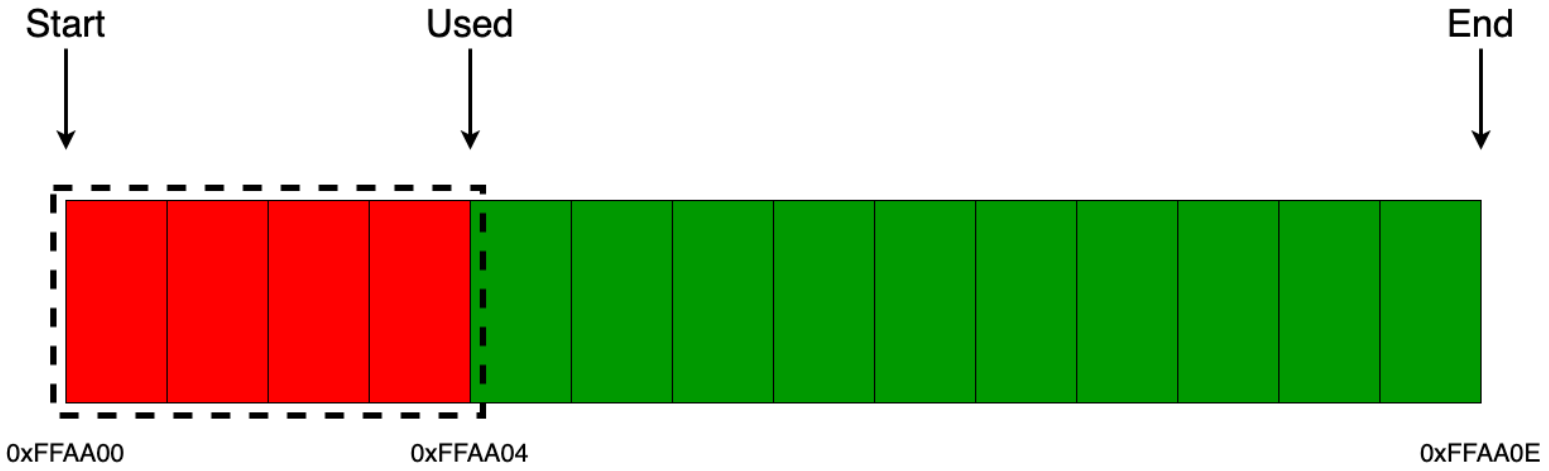
Указатель used даёт понять, сколько памяти используется



Линейный аллокатор

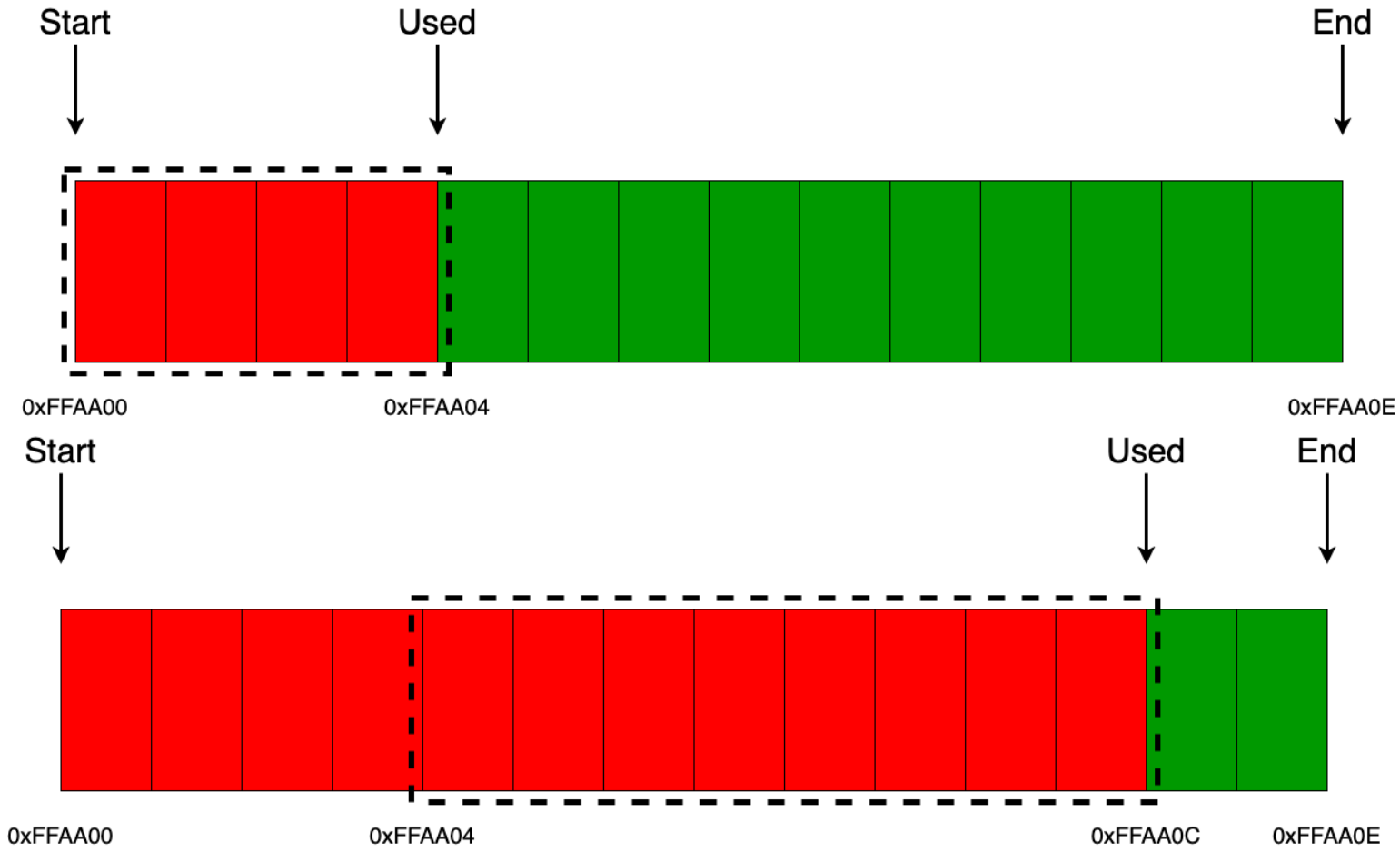
Начало и конец выделенного участка памяти – указатели start, end

Указатель used даёт понять, сколько памяти используется



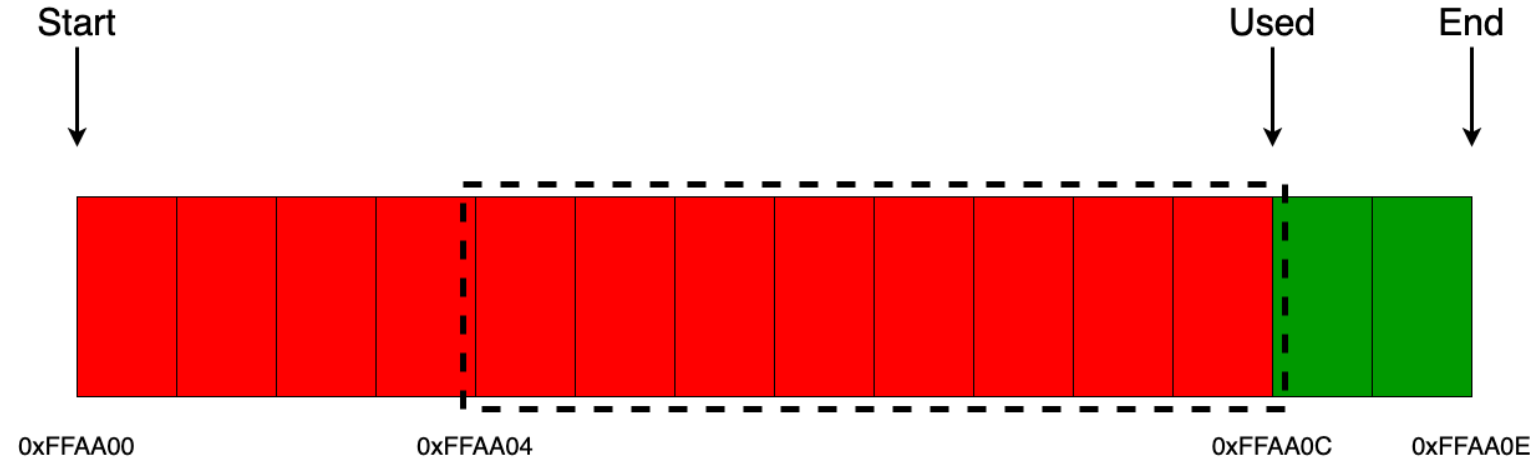
Линейный аллокатор

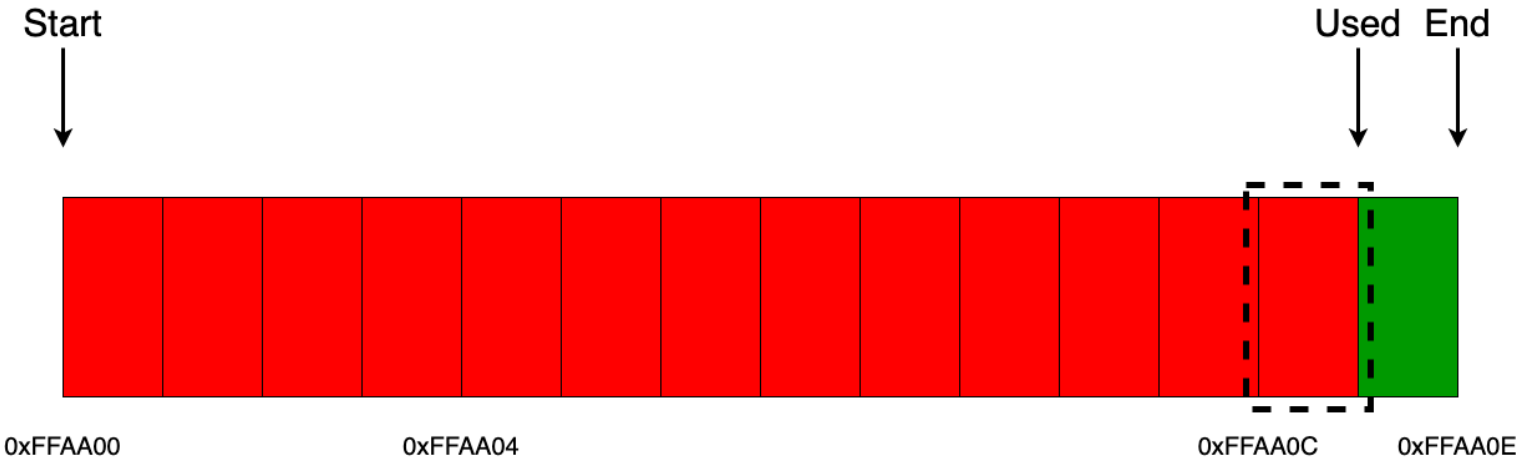
Выделение памяти: запрос на 8 байт



Линейный аллокатор

Выделение памяти: запрос на 1 байт (выравнивания нет)



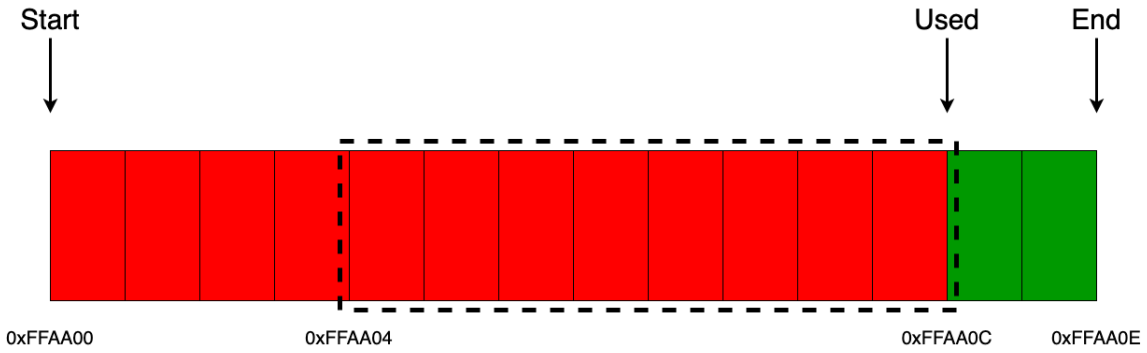


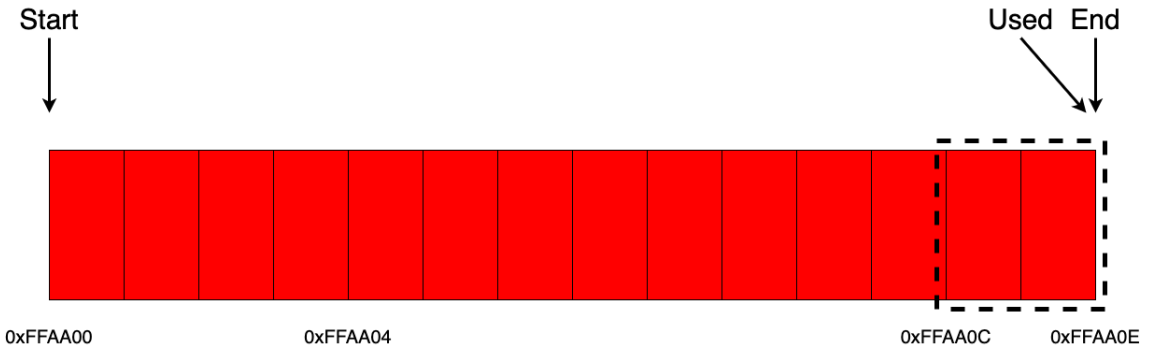
Линейный аллокатор

Выделение памяти: запрос на 1 байт (выравнивание есть)

При выравнивании по адресам, кратным 2, заполняется весь участок

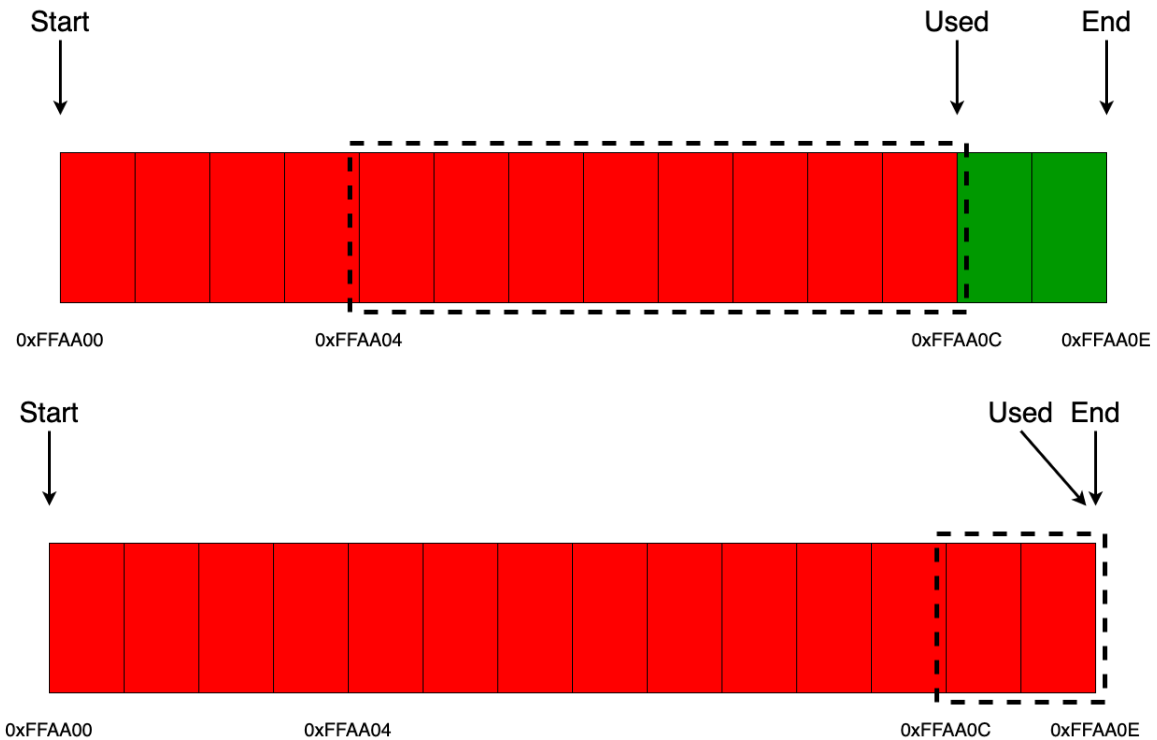
При выравнивании по адресам, кратным числу более, чем 2 (4, 8, …) память не выделена, хотя свободный участок есть





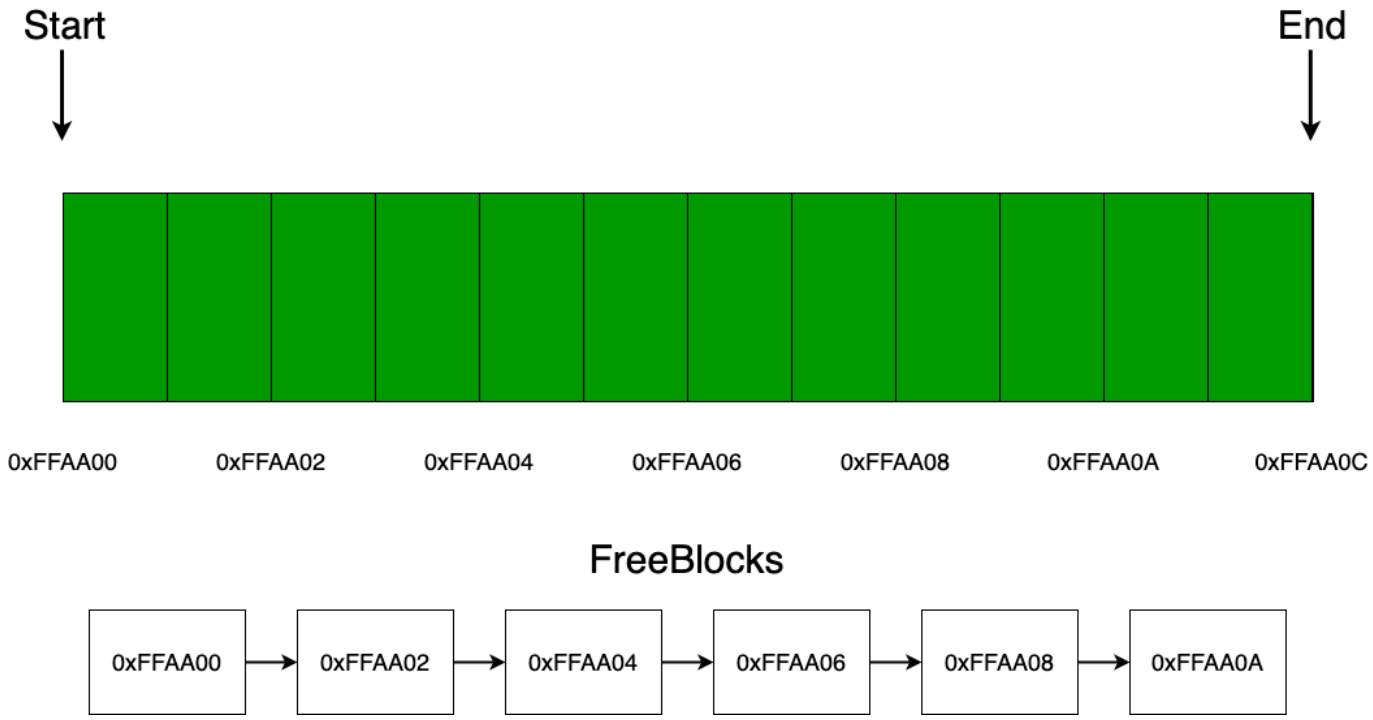
Линейный аллокатор

Освобождение памяти – только целиком



Аллокатор Pool

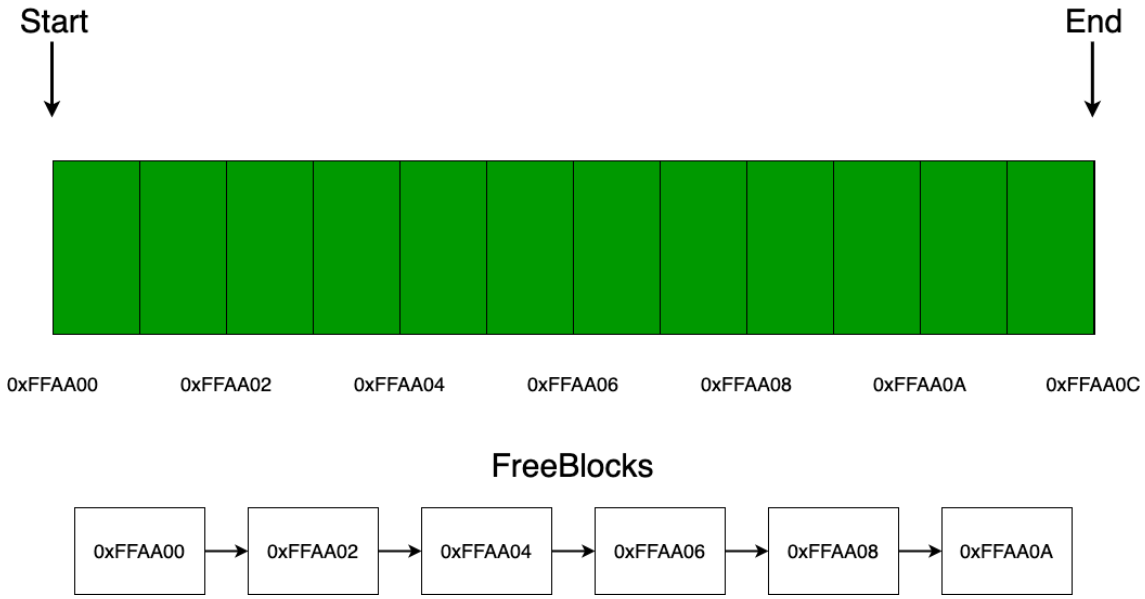
- Деление памяти на сегменты одинакового размера
- Хранятся элементы одинакового типа
- Разрешено выделение и освобождение сегментов
- Можно хранить связный список свободных сегментов прямо в выделенной памяти

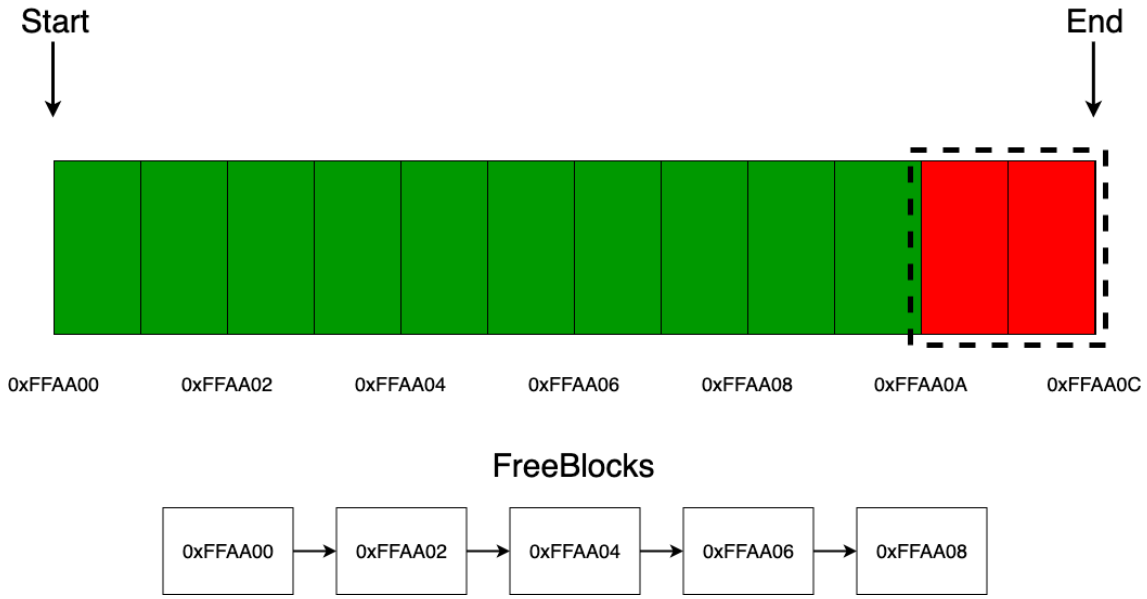


Аллокатор Pool

Запрос на выделение памяти: если есть свободные блоки, достать корневое или хвостовое звено (или несколько звеньев из списка)

Запрос на освобождение: добавить освобожденный сегмент в список





Аллокатор Pool

Потенциальные проблемы:

проверка запросов на соответствие Start/End;

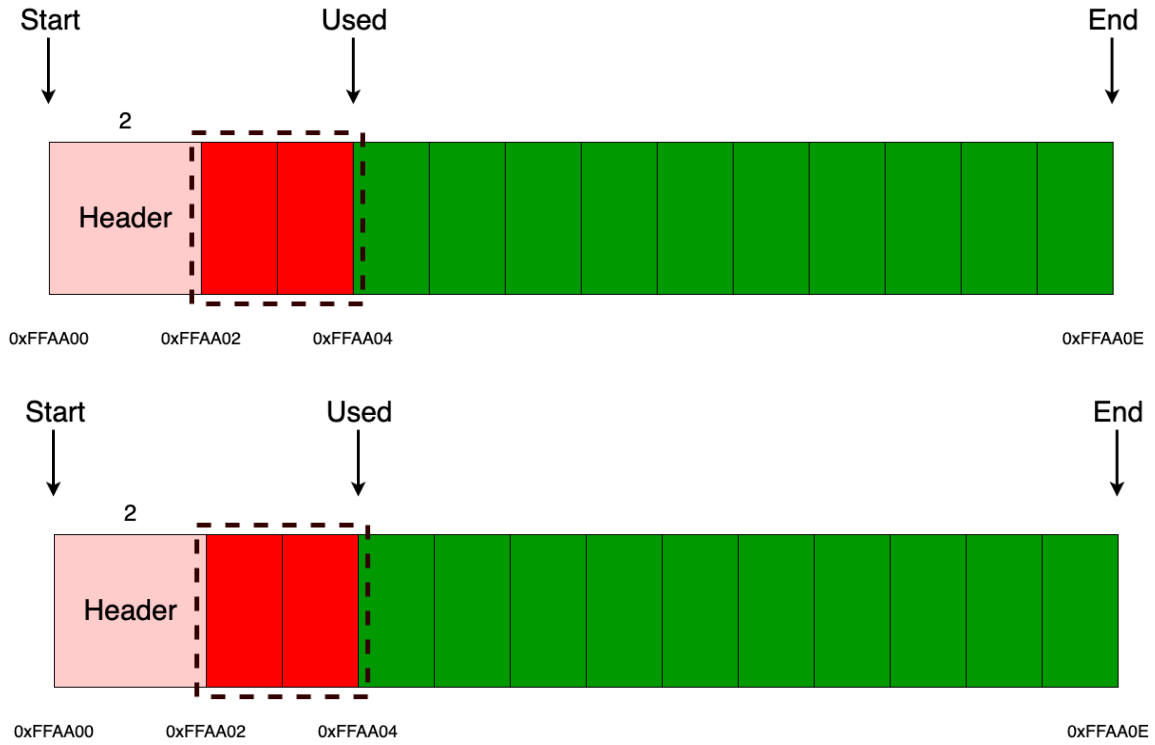
запрос на работу с участком памяти, не совпадающим с сегментом

Стековый аллокатор

В начале выделяемого блока хранится заголовок. Там есть информация о размере выделенного участка

1) запрос на выделение 2 байт
2) запрос на выделение 6 байт

Можно выполнить освобождение участка (deallocate)

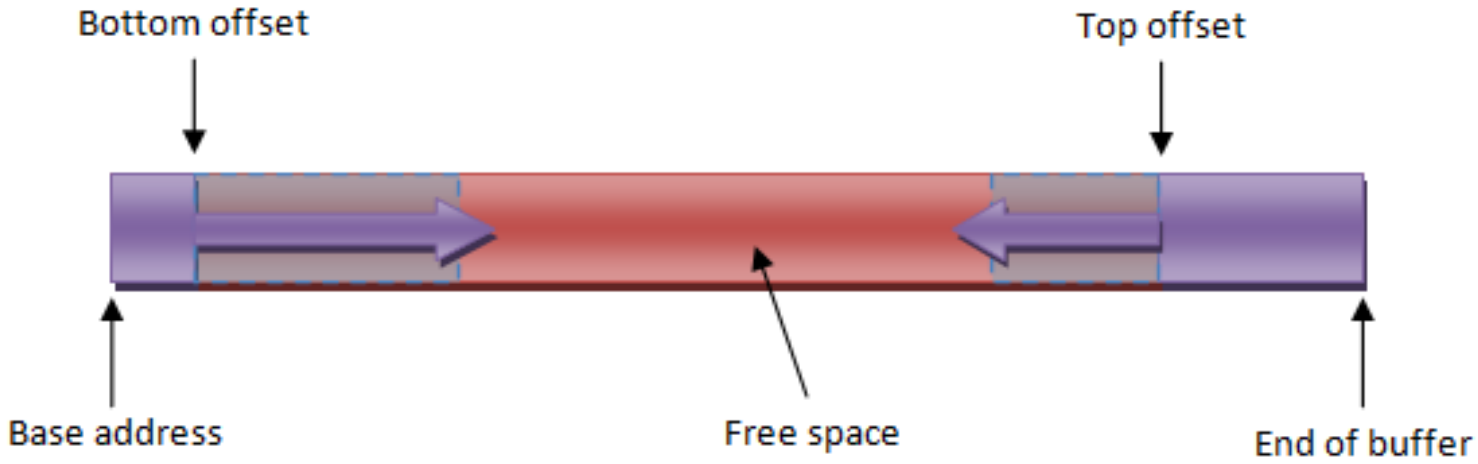


Двусторонний стек

Память выделяется с обеих сторон

С одной – для переменных с низким временем жизни

С другой – для переменных с высоким уровнем жизни



Класс allocator

```
allocator<string> myAllocator;
// allocate space for three strings
string* str = myAllocator.allocate(3);
myAllocator.construct(str, "Geeks");
 myAllocator.destroy(str);
```

Литература (аллокаторы)

https://habr.com/ru/post/505632/

https://habr.com/ru/post/274827/

https://habr.com/ru/post/514404/

https://habr.com/ru/post/435698/

https://habr.com/ru/post/250809/

https://habr.com/ru/company/otus/blog/520502/

https://habr.com/ru/post/515242/

https://www.geeksforgeeks.org/stdallocator-in-cpp-with-examples/