Лабораторная работа 06

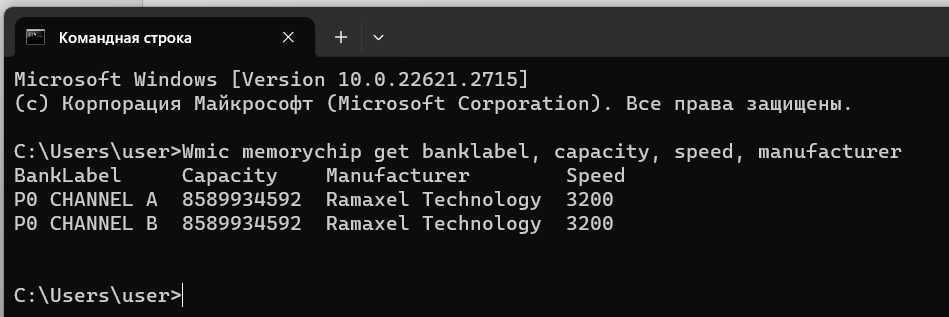
Управление памятью

OC, ПОИТ-3

**Задание 01. Windows**

1. Получите с помощью утилиты **wmic** информации об физической оперативной памяти компьютера, поясните эту информацию.

Wmic memorychip get banklabel, capacity, speed, manufacturer



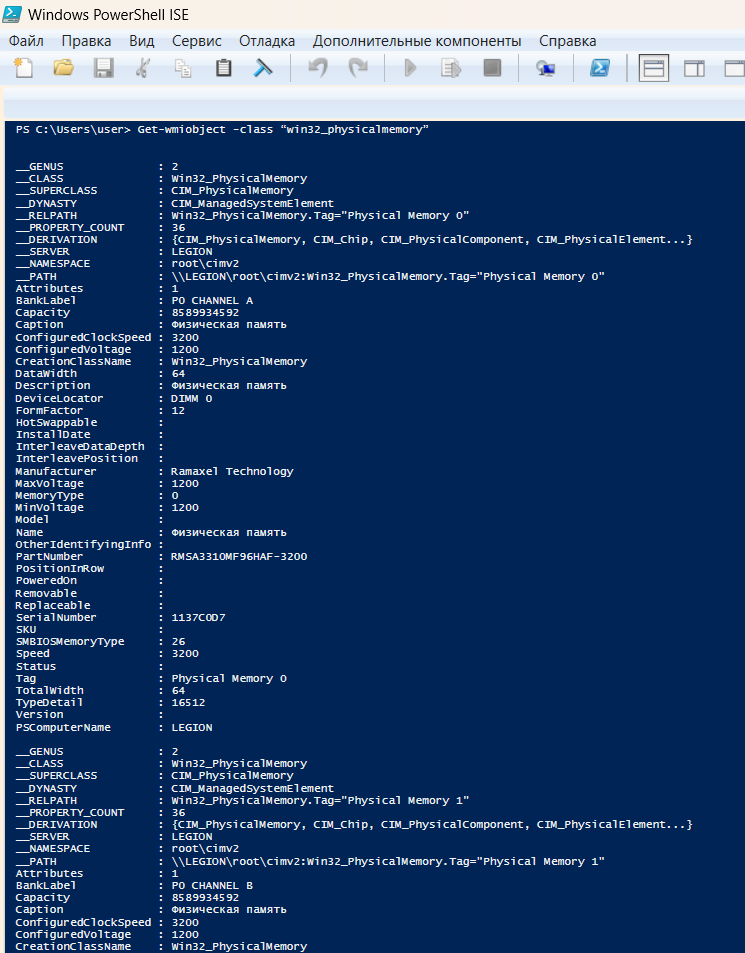
1. **BankLabel:** Это метка банка памяти. Обычно оперативная память разбита на банки, и каждый банк имеет свою метку. Например, "P0 CHANNEL A" и "P0 CHANNEL B" указывают на два банка памяти на канале A и канале B соответственно.
2. **Capacity:** Это емкость оперативной памяти в байтах. В вашем случае значение "8589934592" байт, что соответствует примерно 8 гигабайтам (1 байт = 8 бит).
3. **Manufacturer:** Производитель оперативной памяти. В вашем случае, Ramaxel Technology.
4. **Speed:** Это частота работы оперативной памяти, измеренная в мегагерцах (МГц). Значение "3200" указывает на скорость работы 3200 МГц.

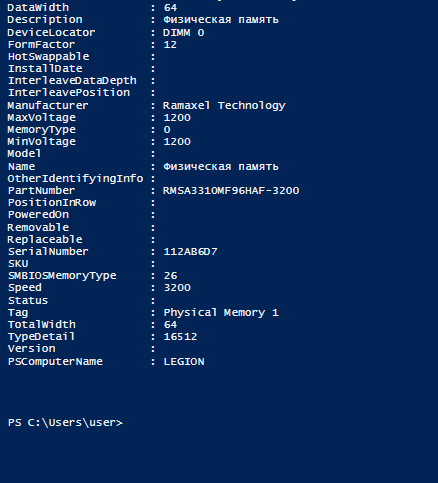
Таким образом, вы видите информацию о двух банках оперативной памяти, каждый объемом 8 гигабайт, производства Ramaxel Technology, с частотой работы 3200 МГц.

1. Получите с помощью утилиты **powershell** информации об физической оперативной памяти компьютера, поясните эту информацию.

Get-wmiobject -class “win32\_physicalmemory”

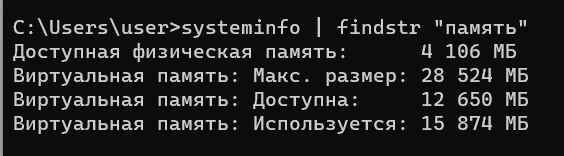
Get-CimInstance -ClassName Win32\_PhysicalMemory | Select-Object Manufacturer, Capacity, DeviceLocator, Speed





1. **\_\_GENUS (2):** Это число указывает на общий тип объекта. В данном случае, значение "2" указывает на объект типа "Экземпляр".
2. **\_\_CLASS (Win32\_PhysicalMemory):** Это имя класса объекта. В данном случае, объект принадлежит классу "Win32\_PhysicalMemory", что означает физическую оперативную память.
3. **\_\_SUPERCLASS (CIM\_PhysicalMemory):** Это имя родительского класса объекта. В данном случае, объект принадлежит классу "CIM\_PhysicalMemory", который является базовым классом для объектов, представляющих физическую оперативную память.
4. **\_\_DYNASTY (CIM\_ManagedSystemElement):** Это имя династии объекта, которая представляет собой последовательность классов, которые порождаются друг из друга. В данном случае, объект принадлежит динстии "CIM\_ManagedSystemElement".
5. **\_\_RELPATH (Win32\_PhysicalMemory.Tag="Physical Memory 0"):** Это путь к объекту в рамках пространства имен. В данном случае, объект имеет тег "Physical Memory 0" и принадлежит классу "Win32\_PhysicalMemory".
6. **\_\_PROPERTY\_COUNT (36):** Это количество свойств (полей) объекта. В данном случае, у объекта 36 свойств.
7. **\_\_DERIVATION ({CIM\_PhysicalMemory, CIM\_Chip, CIM\_PhysicalComponent, CIM\_PhysicalElement...}):** Это последовательность классов, от которых происходит текущий класс. В данном случае, объект произошел от нескольких классов, включая "CIM\_PhysicalMemory" и другие.
8. **\_\_SERVER (LEGION):** Это имя компьютера (сервера), на котором выполняется запрос.
9. **\_\_NAMESPACE (root\cimv2):** Это пространство имен WMI, в котором находится класс объекта. В данном случае, объект находится в пространстве имен "root\cimv2".
10. **\_\_PATH (\LEGION\root\cimv2:Win32\_PhysicalMemory.Tag="Physical Memory 0"):** Это путь к объекту в виде строки, который может использоваться для идентификации объекта в рамках WMI.
11. **Attributes (1):** Это количество атрибутов объекта. В данном случае, у объекта 1 атрибут.
12. **BankLabel (P0 CHANNEL A):** Это метка банка памяти. В данном случае, "P0 CHANNEL A" указывает на банк памяти в канале A.
13. **Capacity (8589934592):** Это общая емкость памяти в байтах. Здесь "8589934592" байта, что равно примерно 8 гигабайтам.
14. **Caption (Физическая память):** Это краткое описание объекта. В данном случае, "Физическая память".
15. **ConfiguredClockSpeed (3200):** Это настроенная частота работы памяти в МГц. Здесь "3200" МГц.
16. **ConfiguredVoltage (1200):** Это настроенное напряжение памяти.
17. **CreationClassName (Win32\_PhysicalMemory):** Это имя класса, создавшего объект.
18. **DataWidth (64):** Это ширина шины данных памяти в битах.
19. **Description (Физическая память):** Это описание объекта. В данном случае, "Физическая память".
20. **DeviceLocator (DIMM 0):** Это идентификатор устройства (DIMM), указывающий, где физически расположен модуль в системе.
21. **FormFactor (12):** Это форм-фактор модуля памяти.
22. **HotSwappable:** Это свойство, указывающее, может ли модуль быть извлечен (горячая замена).
23. **InstallDate:** Это дата установки.
24. **InterleaveDataDepth:** Глубина интерливинга данных. Если глубина интерливинга данных равна 2, то данные могут быть записаны (или считаны) чередуясь на каждом втором блоке памяти.
25. **InterleavePosition:** Позиция интерливинга. Если позиция интерливинга равна 0, то интерливинг может начинаться с самого начала памяти.
26. **Manufacturer (Ramaxel Technology):** Производитель памяти.
27. **MaxVoltage (1200):** Максимальное напряжение памяти.
28. **MemoryType (0):** Тип памяти.
29. **MinVoltage (1200):** Минимальное напряжение памяти.
30. **Model:** Модель памяти.
31. **Name (Физическая память):** Имя объекта. В данном случае, "Физическая память".
32. **OtherIdentifyingInfo:** Дополнительная идентификационная информация.
33. **PartNumber (RMSA3310MF96HAF-3200):** Номер модели памяти.
34. **PositionInRow:** Позиция в ряду.
35. **PoweredOn:** Включено ли устройство.
36. **Removable:** Может ли быть извлечено.
37. **Replaceable:** Может ли быть заменено.
38. **SerialNumber (1137C0D7):** Серийный номер памяти.
39. **SKU:** Stock Keeping Unit (единица учета запасов).
40. **SMBIOSMemoryType (26):** Тип памяти согласно стандарту SMBIOS.
41. **Speed (3200):** Фактическая частота работы памяти.
42. **Status:** Статус объекта.
43. **Tag (Physical Memory 0):** Тег объекта. В данном случае, "Physical Memory 0".
44. **TotalWidth (64):** Общая ширина шины данных памяти в битах.
45. **TypeDetail (16512):** Детали типа памяти.
46. **Version:** Версия объекта.
47. **PSComputerName (LEGION):** Имя компьютера, от которого получена информация.
48. Получите с помощью утилиты **systeminfo** информации об оперативной памяти компьютера, поясните эту информацию.

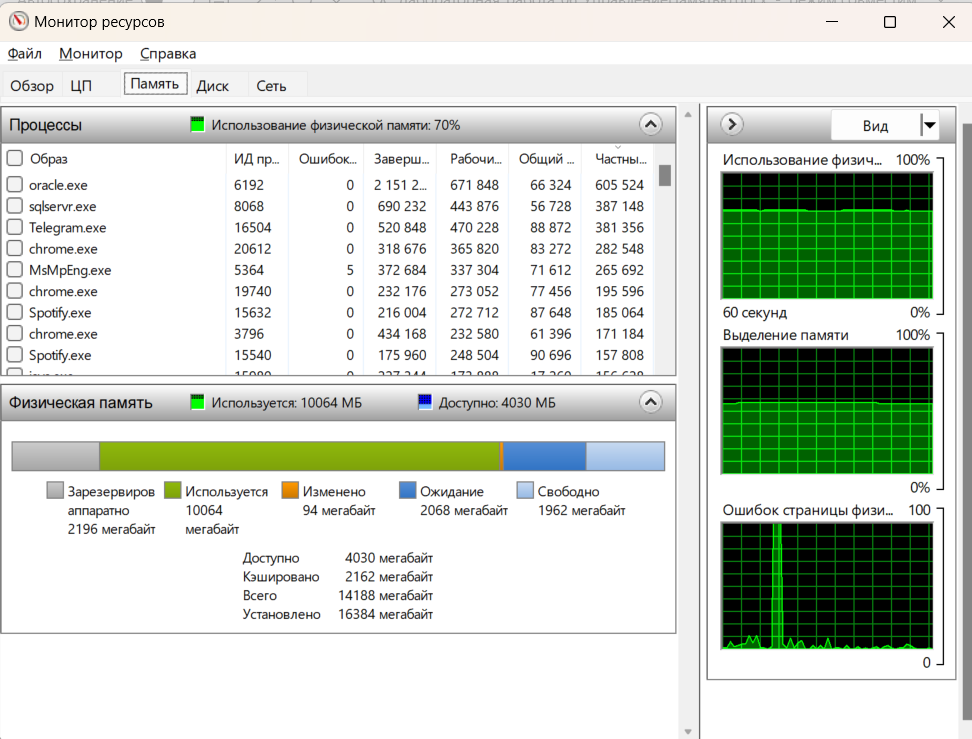
systeminfo | findstr "память"



Физическая память - это физически установленные модули оперативной памяти, доступные прямо процессору.

Таким образом, виртуальная память - это расширение физической памяти, предоставляющее дополнительное пространство для хранения данных на диске в тех случаях, когда физическая память оказывается недостаточной. (файл подкачки)

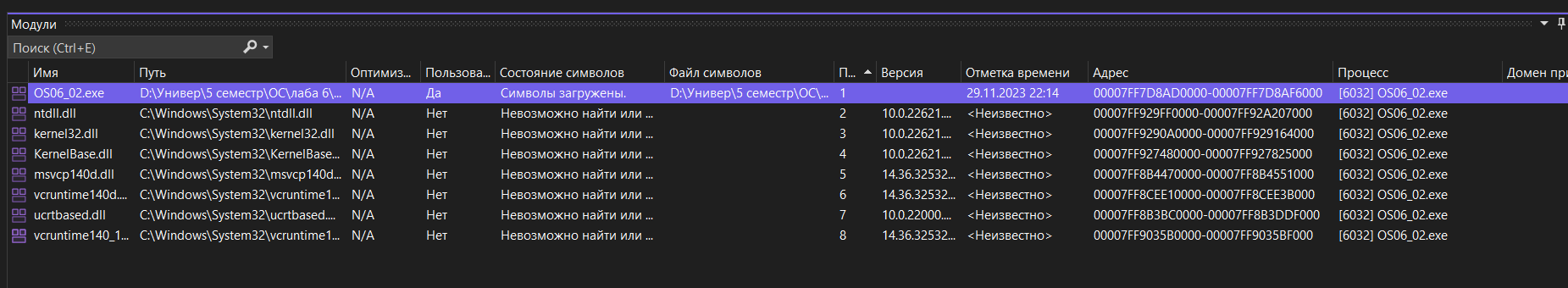
1. Получите с помощью утилиты **performance monitor** информации об оперативной памяти компьютера, поясните эту информацию.

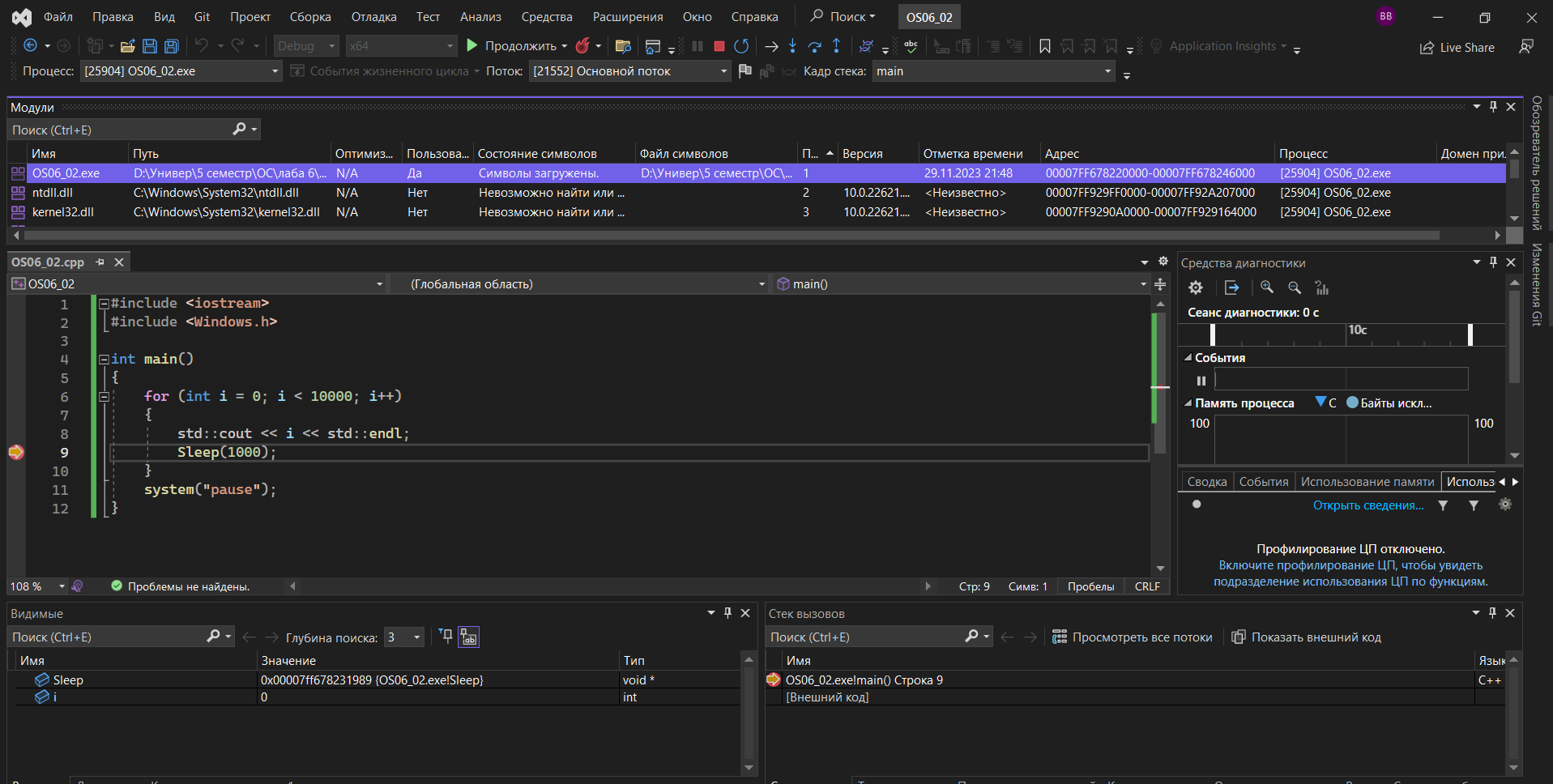


**Задание 02.Windows**

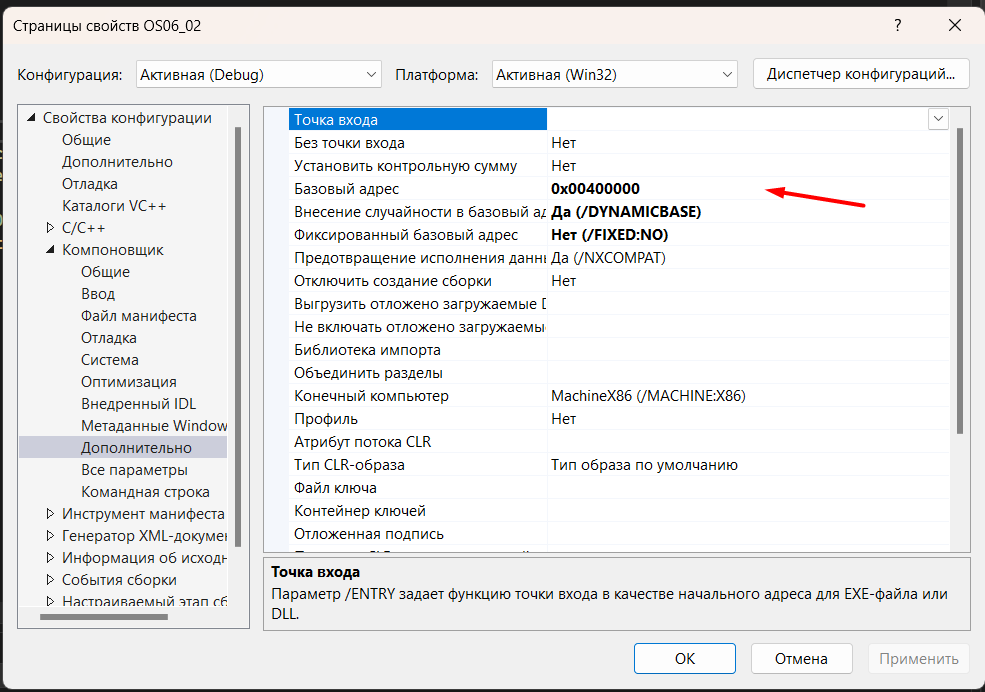
1. Разработайте консольное приложение **OS06\_02**, выполняющее длинный цикл.
2. Продемонстрируйте с помощью отладчика адреса расположения модулей приложения **OS06\_02**.

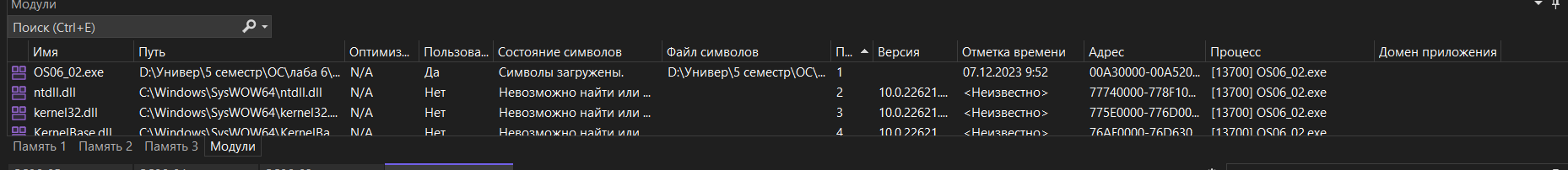
F5-Отладка-окна-модули





1. Установите для приложения **OS06\_02** стандартный адрес загрузки в память.
2. Продемонстрируйте с помощью отладчика стандартный адрес расположения модулей приложения **OS06\_02**





**Задание 03.Windows**

1. Разработайте консольное приложение **OS06\_03**, выполняющее получение 256 страниц оперативной памяти.
2. Разместите в этой памяти массив типа **int,** полностью занимающее все 256 страниц.
3. Заполните этот массив нарастающей последовательностью чисел с шагом 1.
4. Запишите 3 первых буквы своей фамилии в 16-ричными числами в кодировке Windows-1251.
5. Найдите в полученной области памяти с помощью отладчика значение в байте, имеющем адрес вычисленный по следующему принципу: номер страницы = число в нулевом байте, смещение в странице = число 12 бит в 1ом и втором байтах.

*Пример: Иванов*

*И = C8 В = c2*

*в = E2 а = e0*

*а = E0 к = ea*

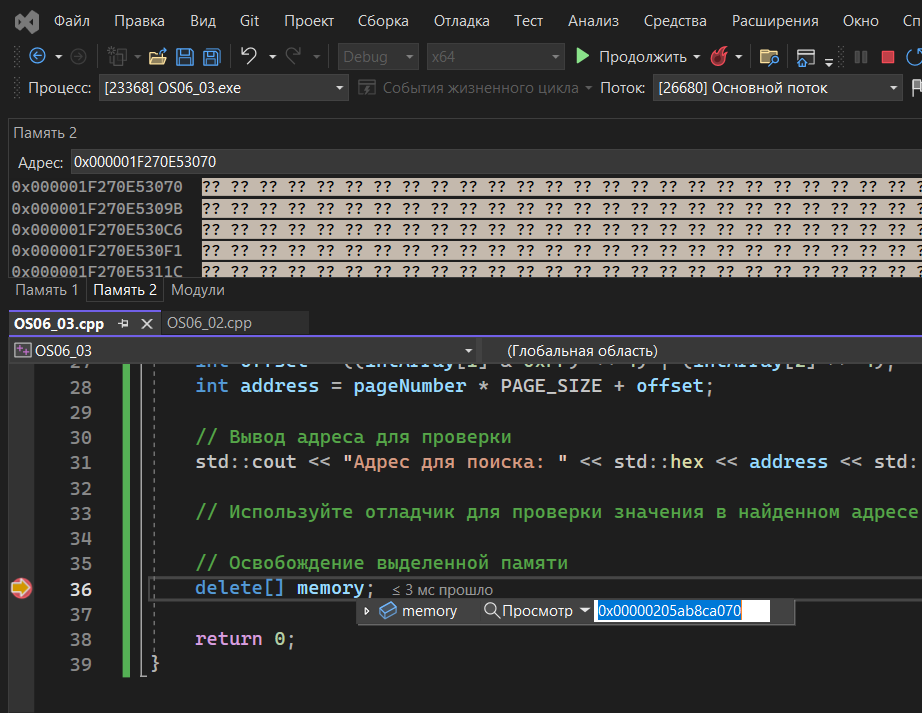
*Страница C8 = 200, смещение E2E = 3630*

*Меняем инициалы фамилии*

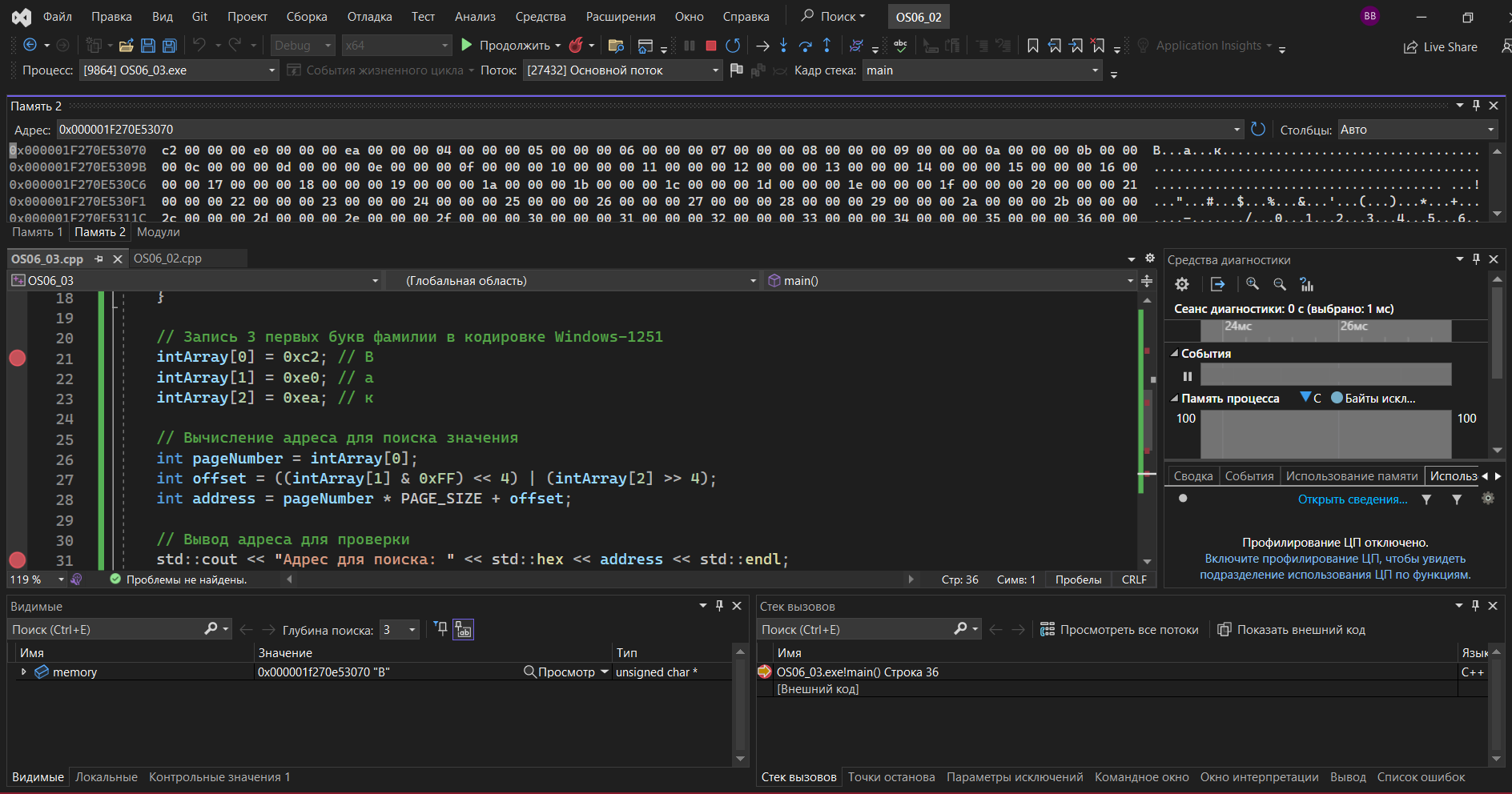
*Breakpoint у delete 36 строка*

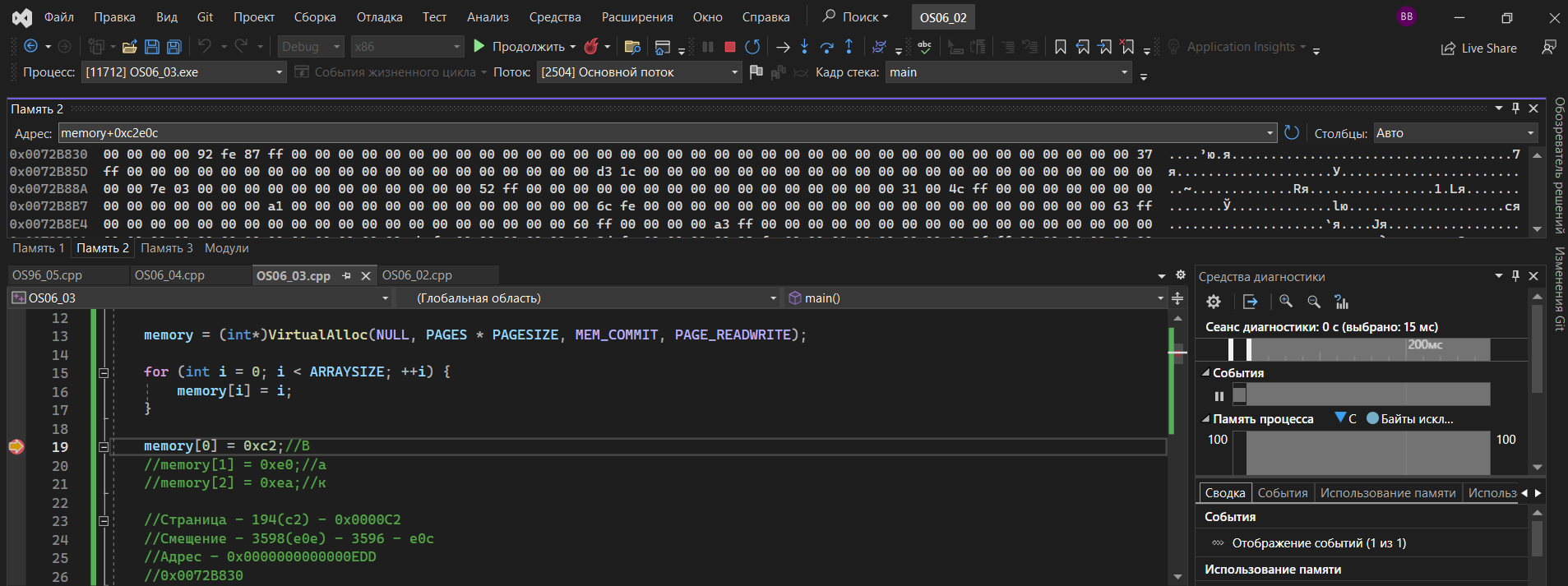
*-окна-память-память 1 или память 2*

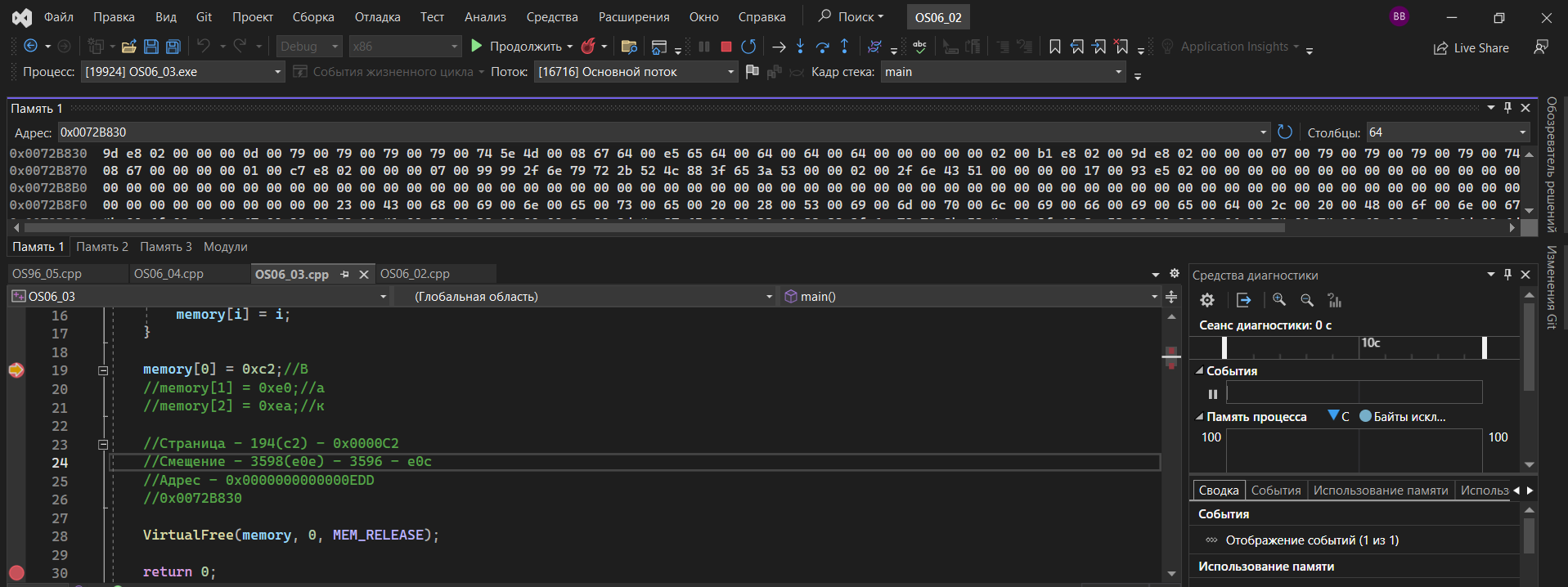
*Ждем пока прога дойдет до этой строки и чекаем адрес*

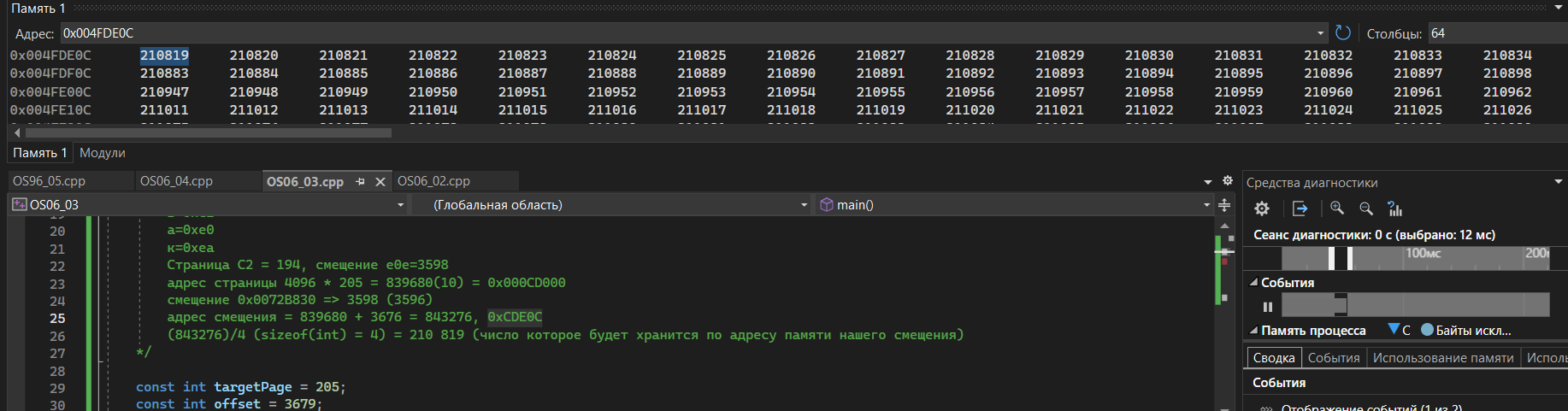


*Закидываем его в наше окно памяти и смотрим в первой строке наши символы:* c2, e0, ea.



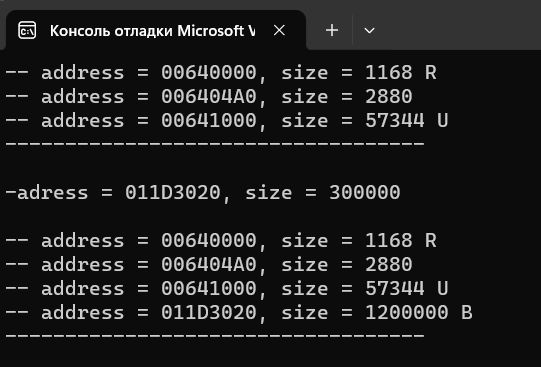






**Задание 04.Windows**

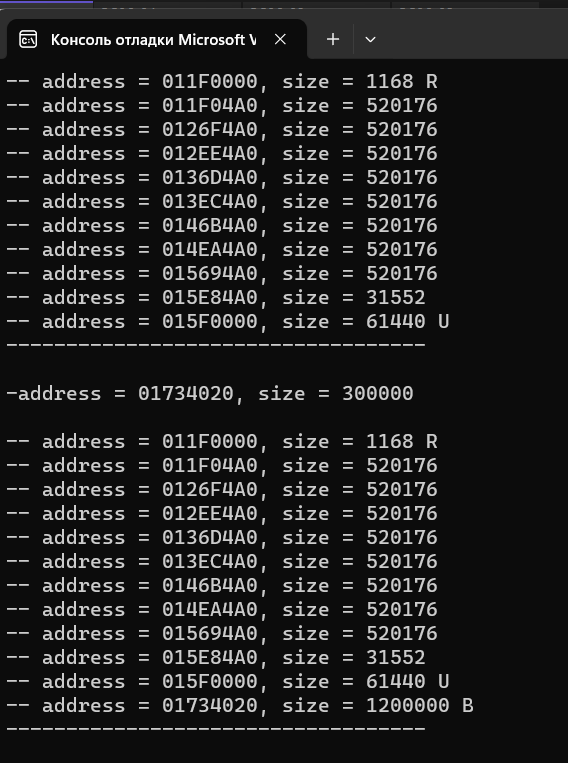
1. Разработайте консольное приложение **OS06\_04,** которое включает функцию **sh**, принимающую 1 параметр: дескриптор (HANDLE) heap.
2. Функция **sh** выводит на консоль, общий размер heap, размеры распределенной и нераспределенных областей памяти heap.
3. Приложение **OS06\_04** размещает в стандартной heap процесса int-массив размерности 300000.
4. Выведите с помощью функции **sh** информацию до размещения массива и после.
5. Объясните результат.



Видим что после размещения массива добавилось 4 поле с флагом B

**Задание 05.Windows**

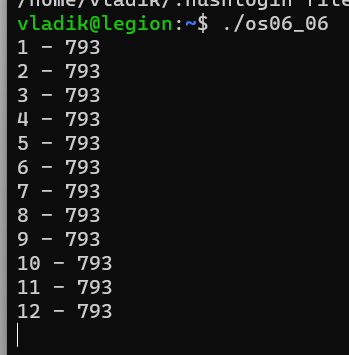
1. Разработайте консольное приложение **OS06\_05** аналогичное приложению **OS06\_05,** но использующее пользовательскую heap, которая имеет первоначальный размер 4MB.
2. Объясните результат.



В результате появился новый распределительный блок памяти

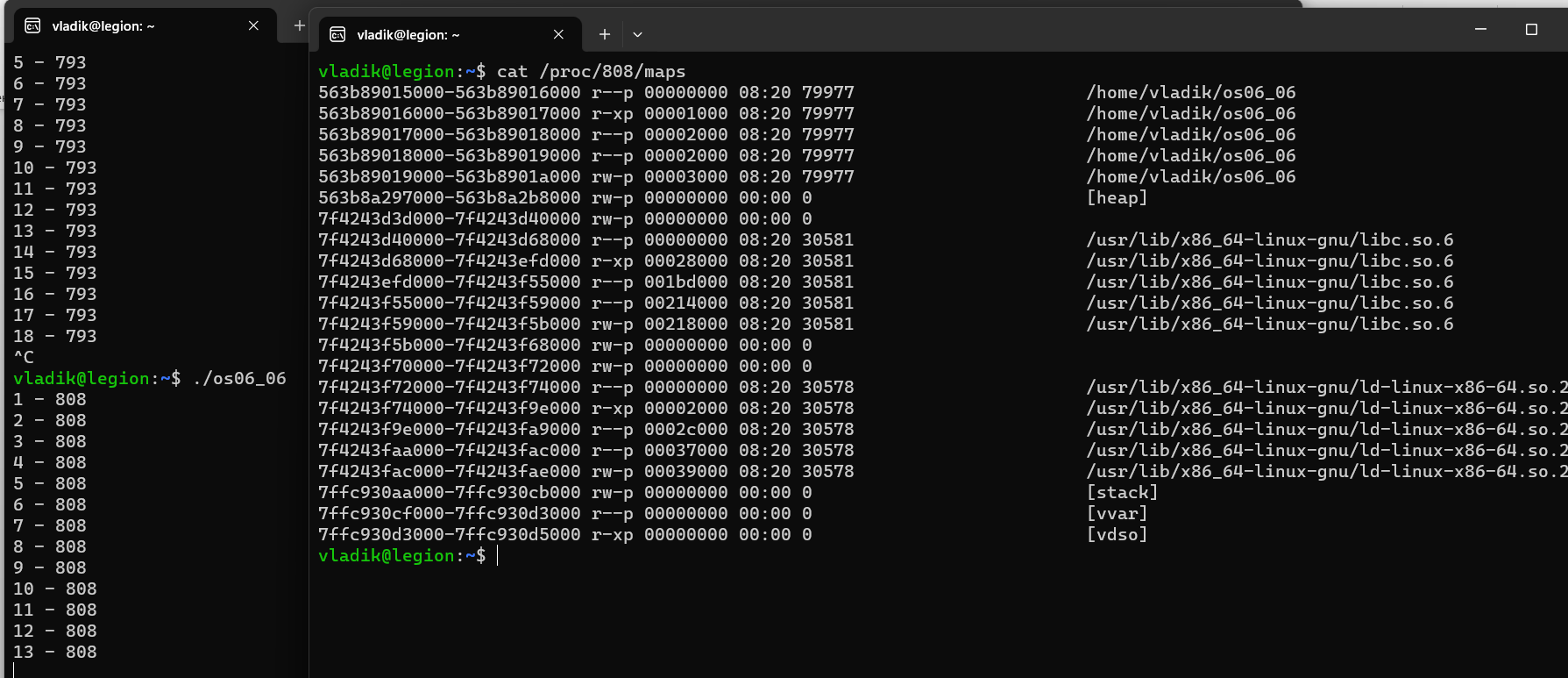
**Задание 06.Linux**

1. Разработайте консольное приложение **OS06\_06**, выполняющее длинный цикл.
2. Продемонстрируйте с помощью файловой системы **/proc** структуру адресного пространства.
3. Продемонстрируйте с помощью **pmap**  структуру адресного пространства.
4. Определите с помощью утилиты objdump адрес загрузки main-модуля, секций с кодом, данными, неинициализированными глобальными переменными.



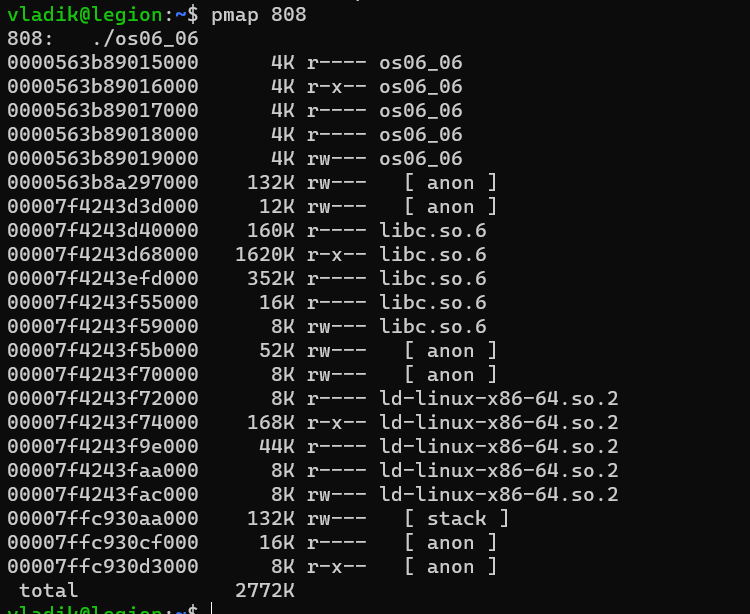
cat /proc/793/maps

Показывает отображение виртуальной памяти процесса, включая диапазоны адресов, права доступа, смещения файлов (если применимо) и другую информацию.



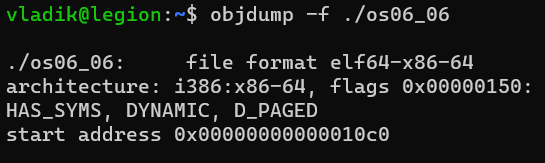
Pmap

Показывает распределение использования памяти внутри процесса, включая адреса, разрешенные права доступа и размеры секций памяти.



objdump -f ./os06\_06

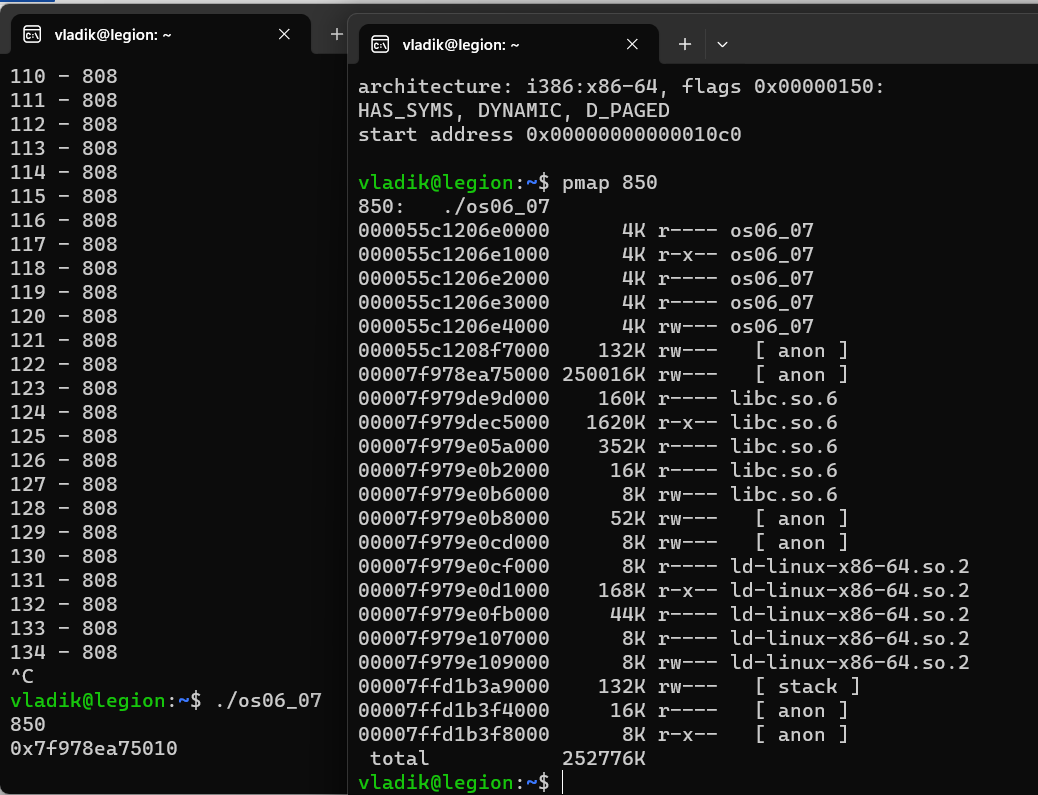
Показывает различную информацию о формате и структуре исполняемого файла, такую как архитектура, флаги, размер секций и другие атрибуты.



C кодом данными неинициализированными и глобальными переменными

**Задание 07.Linux**

1. Разработайте консольное приложение **OS06\_07**, которое динамически выделяет 256 МБ памяти.
2. В выделенной памяти разместите int-массив максимальной размерности. Проинициализируйте массив последовательными значениями с шагом 1.
3. Выведите на консоль адрес выделенной памяти.
4. После инициализации приложение должно приостановить свое выполнение на длительный интервал времени.
5. С помощью утилиты pmap определите область памяти в которой выделена память.



**Задание 08** Ответьте на следующие вопросы

1. Поясните понятие «виртуальная память».
2. Поясните понятие «свопинг».
3. Поясните понятие «страничная память».
4. Поясните понятие MMU.
5. Поясните понятие TLB.
6. Какая информация содержится в строке таблицы страниц
7. Поясните принцип применения хэш-таблиц.
8. Поясните применение «инвертированной таблицы физических» страничной памяти.
9. Поясните понятие «рабочий набор страниц».
10. Поясните принцип работы алгоритма LRU.
11. Windows: поясните назначение сервиса SysMain.
12. Windows: поясните назначение файла hiberfil.sys.
13. Windows: поясните назначение файла pagefile.sys.
14. Windows: поясните назначение файла swapfile.sys.
15. Windows: перечислите области адресного пространства (от младших к старшим адресам) и поясните их назначения.
16. Windows: какой стандартный начальный размер области heap?
17. Windows: каким образом можно изменить начальный размер области памяти heap приложения?
18. Windows: какой стандартный размер области памяти stack?
19. Windows: каким образом можно изменить размер области памяти stack приложения?
20. Windows: поясните назначение функции Windows API: GlobalMemoryStatus.
21. Windows: поясните назначение функции Windows API: VirtualQuery; перечислите значения атрибутов Protect, State и Type.
22. Windows: что такое «рабочее множество»? поясните принцип управления рабочим множеством с помощью OS API.
23. Windows: что означает «страница заблокирована»? с помощью каких функций OS API можно установить блокировку страниц и снять блокировку? Какое максимальное количество страниц можно заблокировать?
24. Windows: что такое «heap»? Что такое «heap процесса»? Что такое «пользовательская heap»? Поясните принцип устройства heap.
25. Linux: перечислите области адресного пространства (от младших к старшим адресам) и поясните их назначения.
26. Linux: в какой части адресного пространства выделяется памяти с помощью функций malloc, calloc?

ghp\_UiP1dFUGPRJE81DceswyhGAoat0OD618kUI2 – github token to download a folder from gt