Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Системное программирование**

Студент: Козека Е. М.

ФИТ 3 курс 4 группа

Преподаватель: Бернацкий П. В.

Минск 2024

**Лабораторная работа №5. Управление памятью**

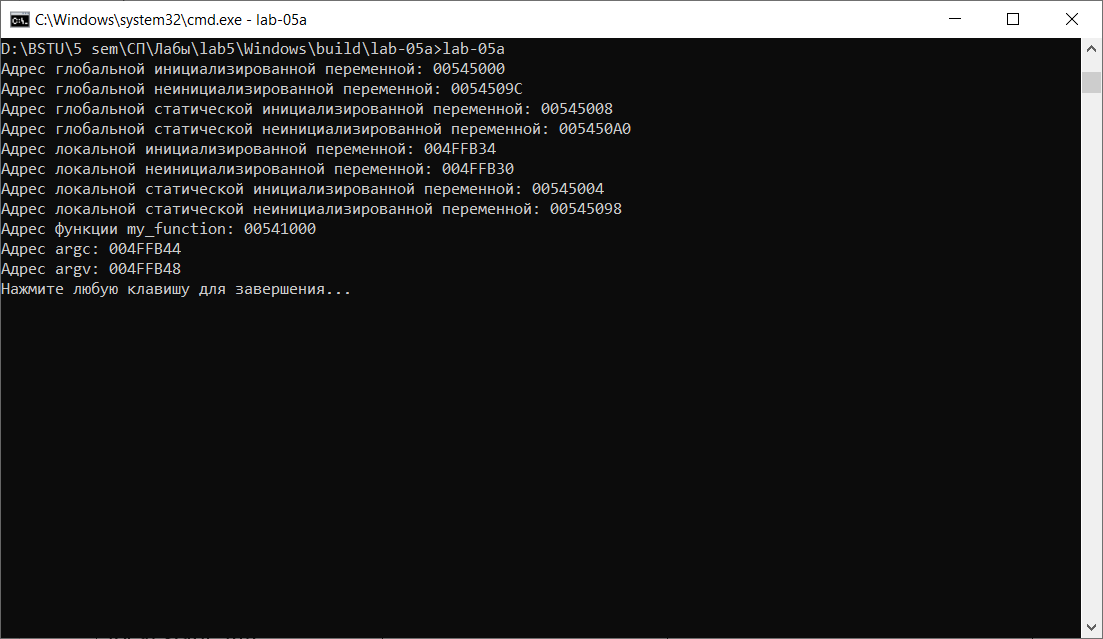
**Задание 1.** Разработайте консольное Windows-приложение **Lab-05a** в котором создайте следующее:

* Любую функцию;
* Глобальные инициализированную и неинициализированную переменные;
* Глобальные статические инициализированную и неинициализированную переменные;
* Локальные инициализированную и неинициализированную переменные;
* Локальные статические инициализированную и неинициализированную переменные;
* Объявите функцию main cо стандартными аргументами **argc** и **argv**.

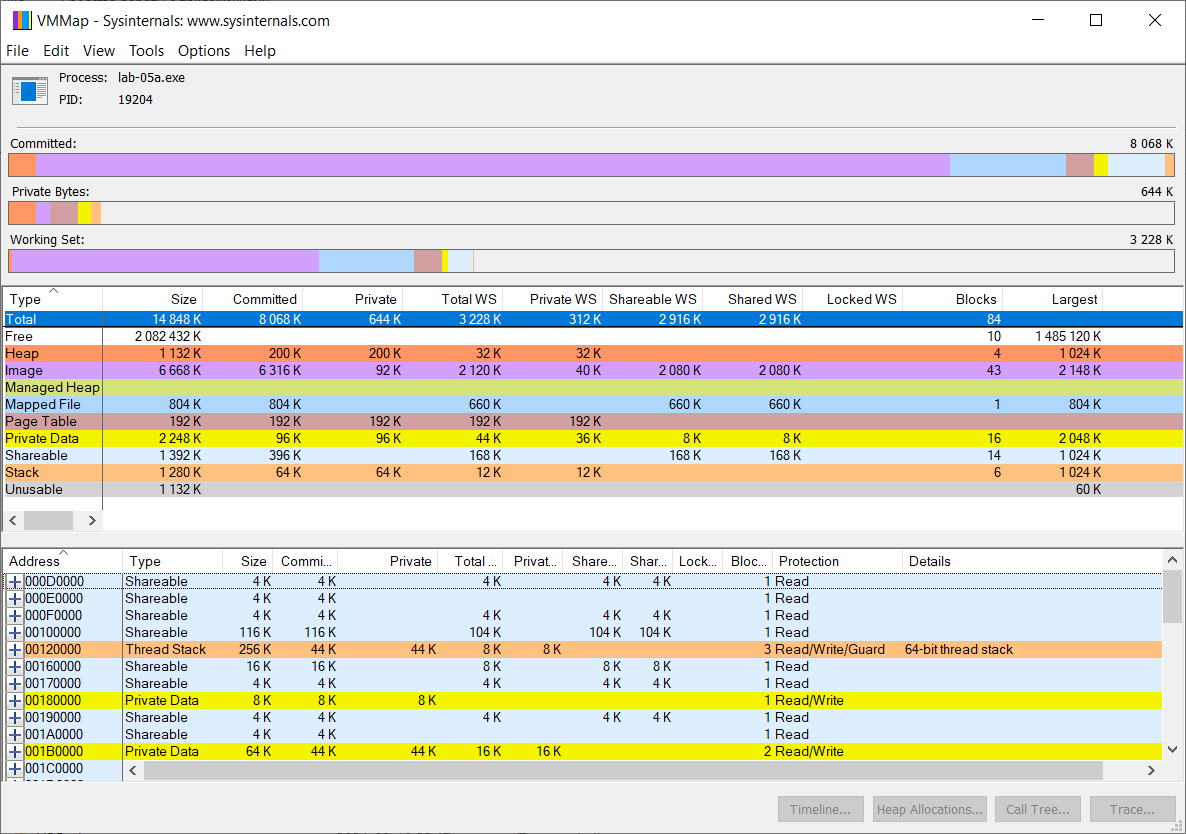
Приложение должно выводить на консоль адреса всех объявленных переменных и функций, а также адреса переменных **argc** и **argv** и перейти в режим ожидания ввода символа на консоль.

С помощью утилиты VMMap найти секции памяти, в которых расположены все эти переменные и функции и заполнить таблицу.

Результат выполнения в консоли:



|  |  |  |
| --- | --- | --- |
| **Вид переменной** | **Адрес** | **Название секции** |
| global init | 00545000 | .data |
| global uninit | 0054509C | .bss |
| global static init | 00545008 | .data |
| global static uninit | 005450A0 | .bss |
| local init | 004FFB34 | thread stack |
| local uninit | 004FFB30 | thread stack |
| local static init | 00545004 | .data |
| local static uninit | 00545098 | .bss |
| function | 00541000 | .text |
| argc | 004FFB44 | thread stack |
| argv | 004FFB48 | thread stack |



**Задание 2.** Разработайте консольное приложение **Lab-05b** в котором реализуйте код, который делится на следующие этапы:

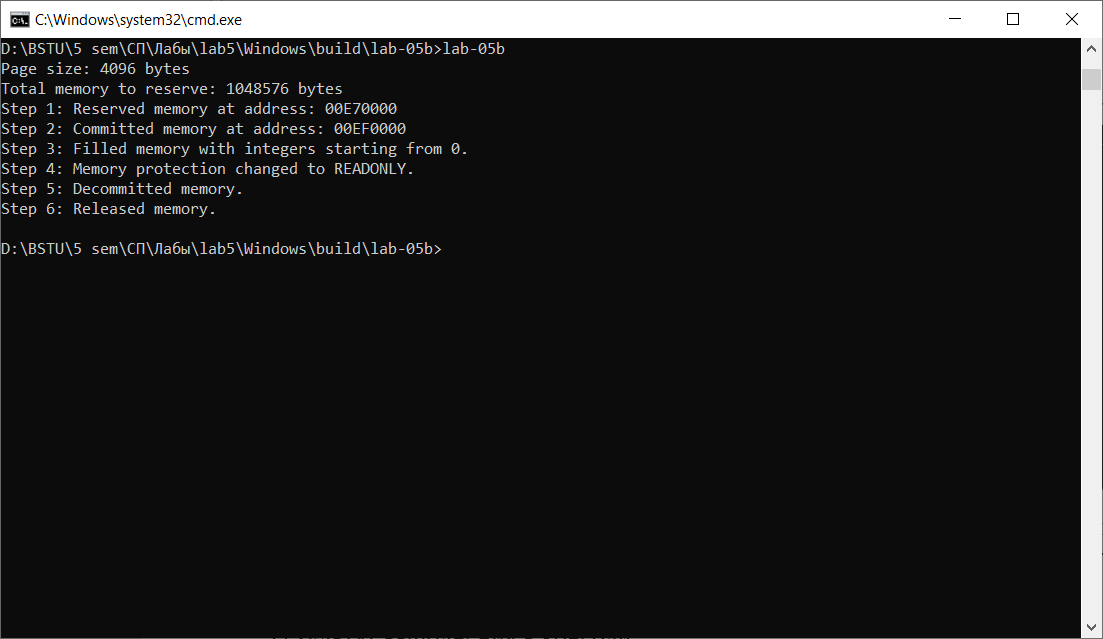
* Зарезервировать виртуальной памяти процесса на 256 страниц (размер страницы определить через вызов соответствующей функции) и вывести адрес области памяти на консоль;
* Для второй половины (128 страниц) зарезервированной области выделить физическую память;
* Заполнить данную половину памяти последовательностью целых чисел начиная от 0 с шагом 1;
* Изменить атрибуты защиты страниц памяти на «только для чтения»;
* Освободить выделенную физическую память для 128 виртуальных страниц;
* Освободить 256 страниц виртуальной памяти.

После каждого этапа программа должна ожидать ввода в консоль. В это время требуется найти в программе **VMMap** выделенную область виртуальной памяти и изучить её характеристики. Также на этапах 1, 2, 3 и 5 в программе **RAMMap** проверить выделены ли физические страницы памяти.

В программе **RAMMap** выполняется это на вкладке «Physical Pages», а также используя фильтр внизу окна по значению «Process».

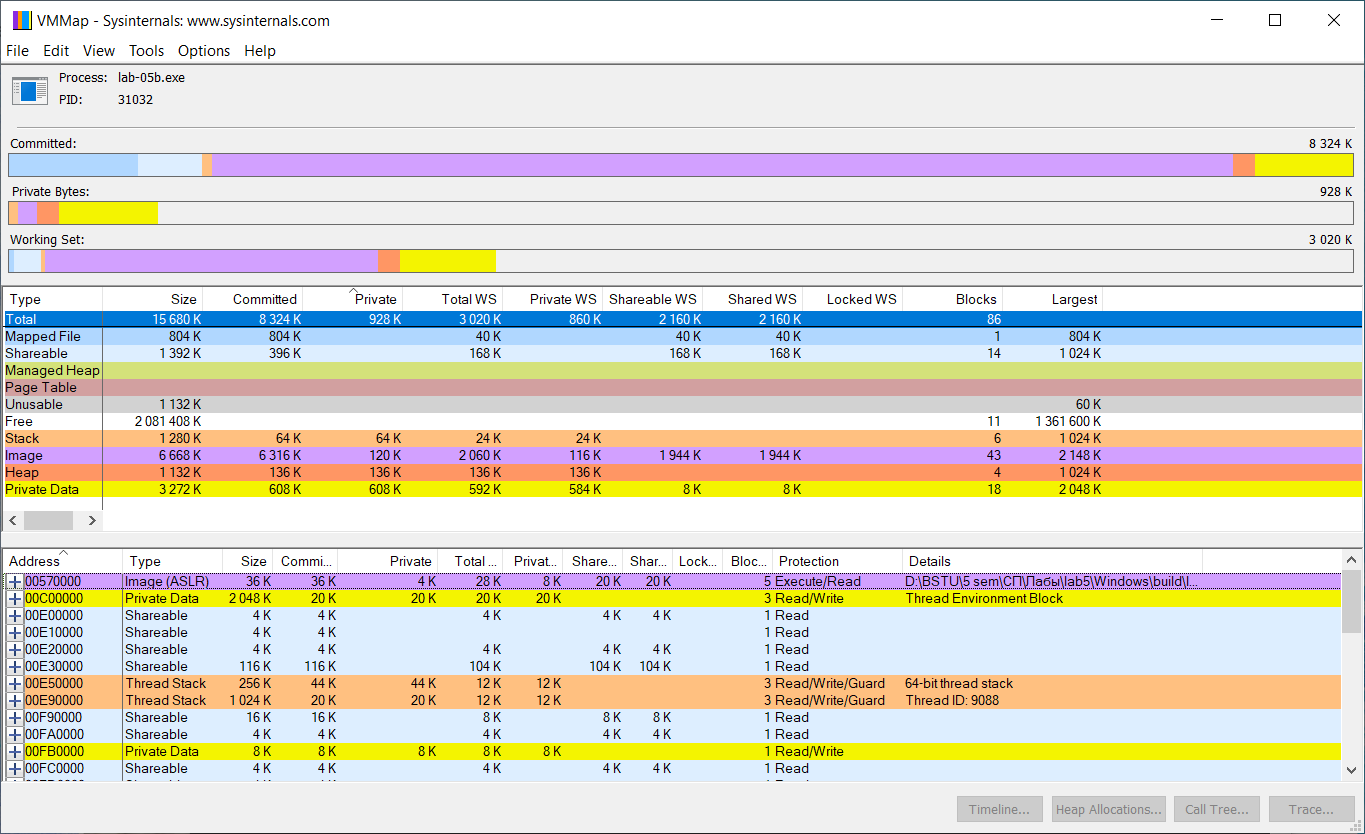
*На каком этапе фактически были выделены физические блоки памяти?*

Результат выполнения в консоли:

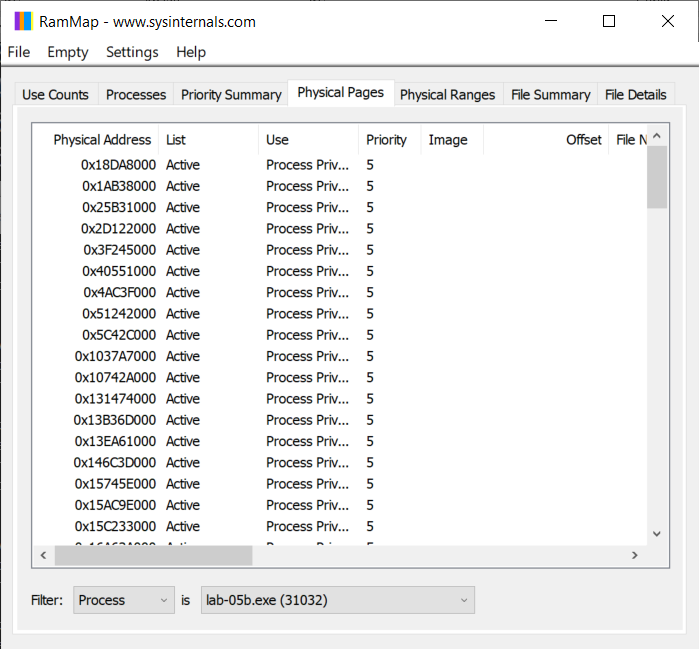


Физические блоки памяти выделяются на втором этапе при вызове MEM\_COMMIT.

VMMap:



RAMMap:



**Задание 3.** Разработайте консольное приложение **Lab-05с** в котором реализуйте код, который делится на следующие этапы:

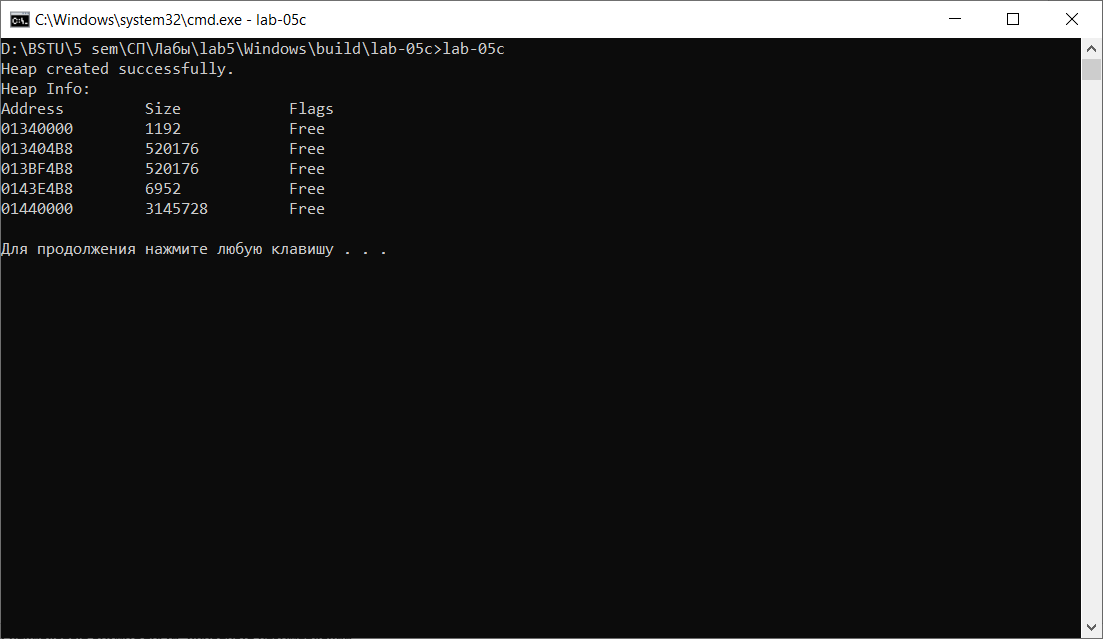
* Создание пользовательской кучи со стартовым размером 1 МиБ и максимальным 8 МиБ;
* В цикле выделить 10 блоков памяти из кучи размером 512 КиБ каждый;
* Заполнить каждый блок массивом целых чисел соответствующей длины (512 КиБ / 4 Б);
* В цикле освободить память, выделенную для всех блоков;
* Уничтожить кучу.

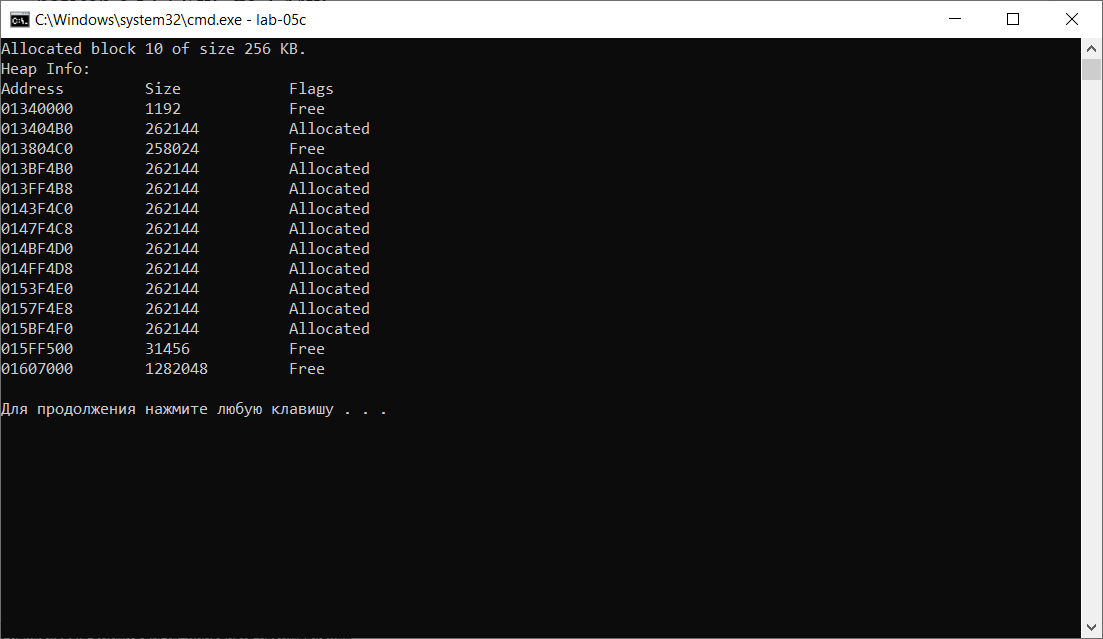
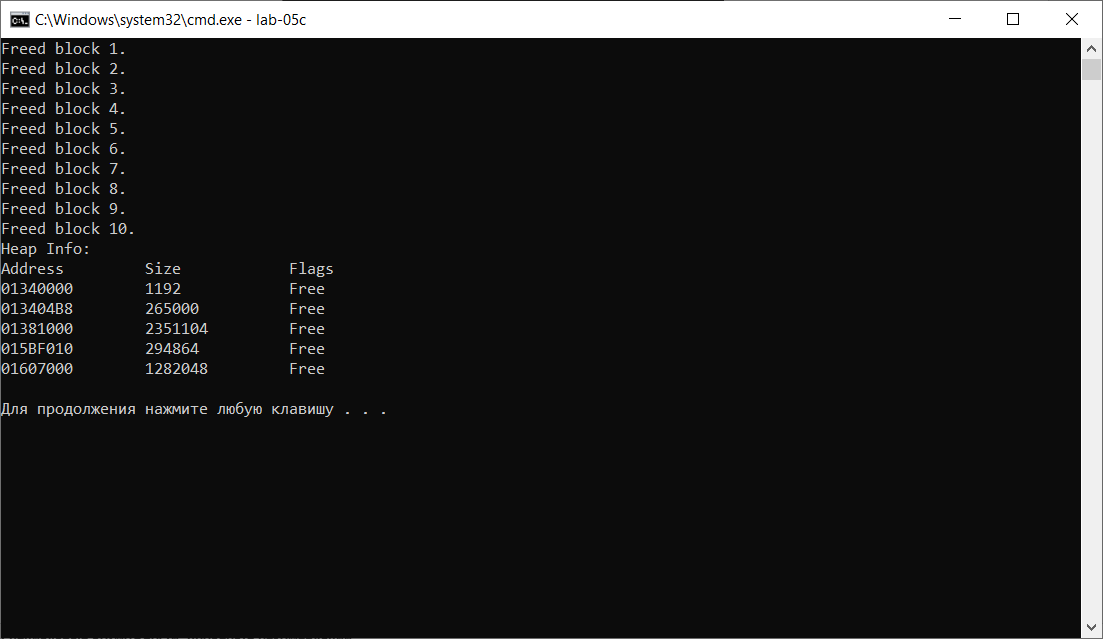
Написать функцию **HeapInfo**, которая выводит на консоль информацию о куче, а именно общий размер, размеры областей памяти кучи с указанием адреса начала области, размера области и типа области. Вызов данной функции должен производиться после каждого этапа кроме заполнения массивами (на этапе 2 после каждой итерации).

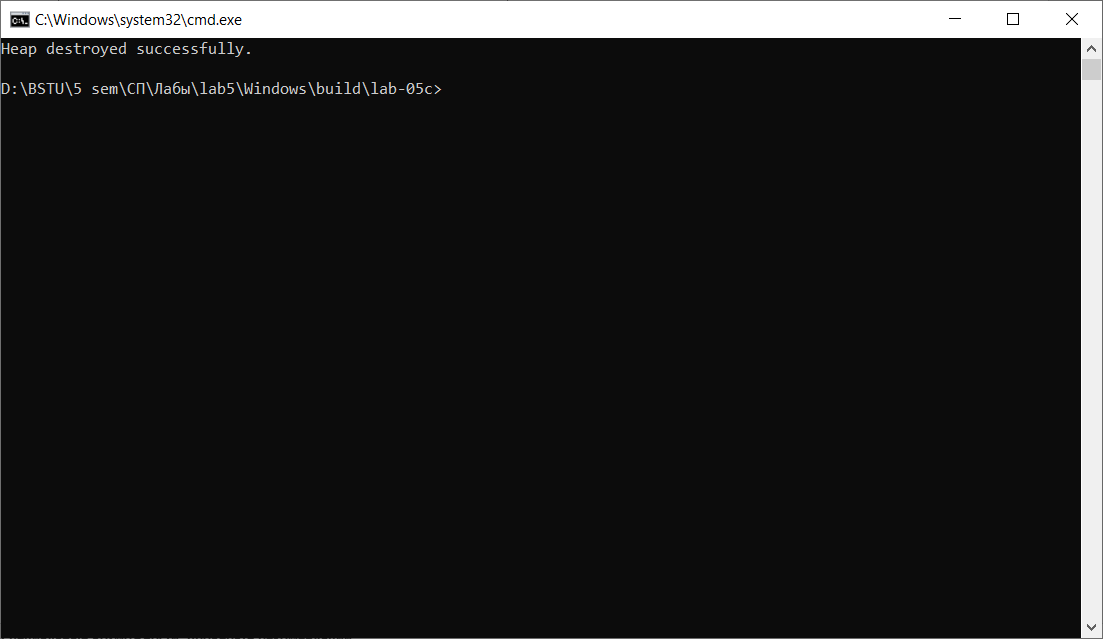
Для красоты вывода на консоль можно вызывать функцию в связке с паузой следующим образом: HeapInfo(heap); system("pause & cls");.

Попытаться изменить количество выделяемых блоков с 10 до 5 и изменить размер с 512 КиБ до 1 МиБ.

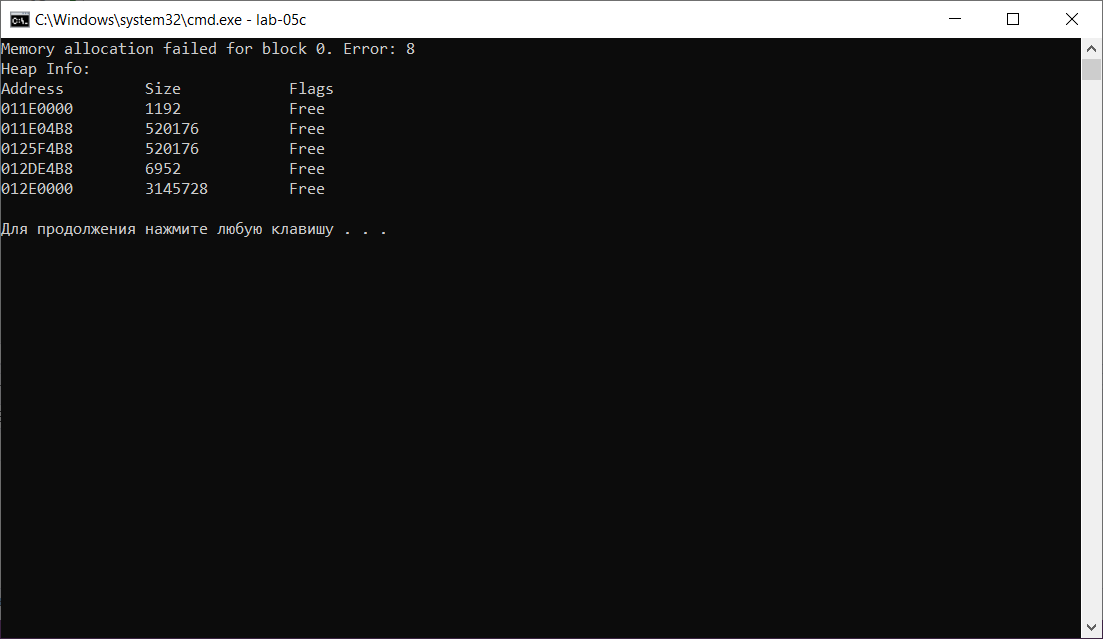
Результат выполнения в консоли:





Попытка изменить количество выделяемых блоков и их размер вызывает ошибку:



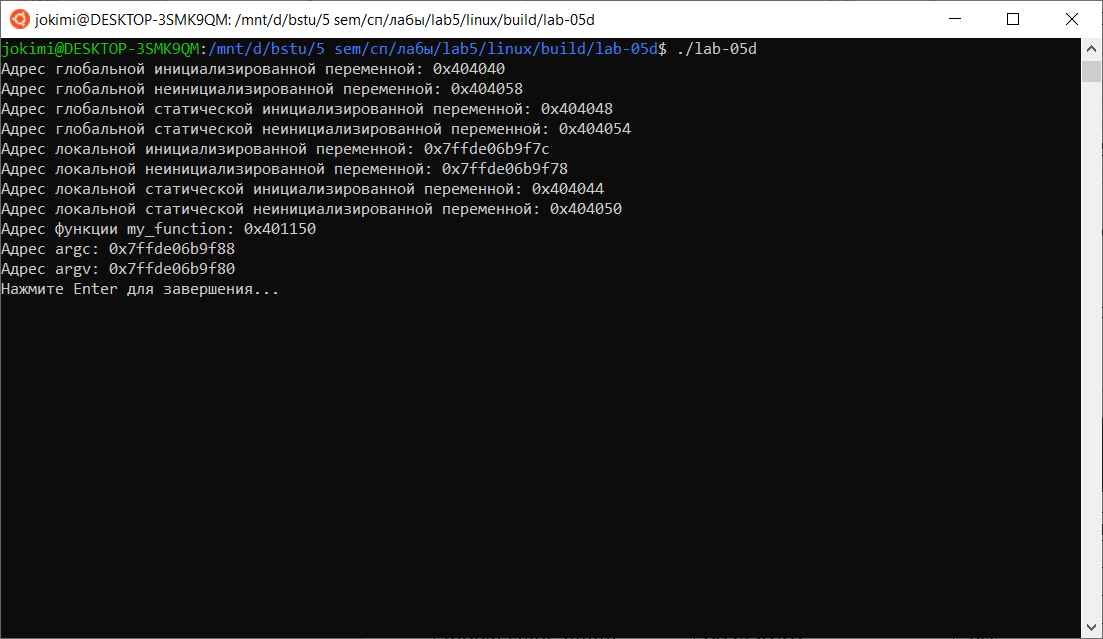
**Задание 4.** Разработайте консольное Linux-приложение, являющееся полным аналогом **Lab-05а.**

Для заполнения таблицы требуется сделать следующее:

* Запустить процесс;
* Вызвать утилиту «**pmap -X <PID> | awk '{ print $1, $2, $6, $20 }'**» и выписать стартовый адрес вашего образа исполняемого файла (обычно это самая первая строка в таблице)
* Вызвать утилиту «**objdump --adjust-vma <адрес с прошлого шага> -x ./<image-file>**» (здесь вас интересует раздел Sections)

На основе полученной информации из двух утилит найти секции памяти, в которых расположены все эти переменны и функции и заполнить таблицу.

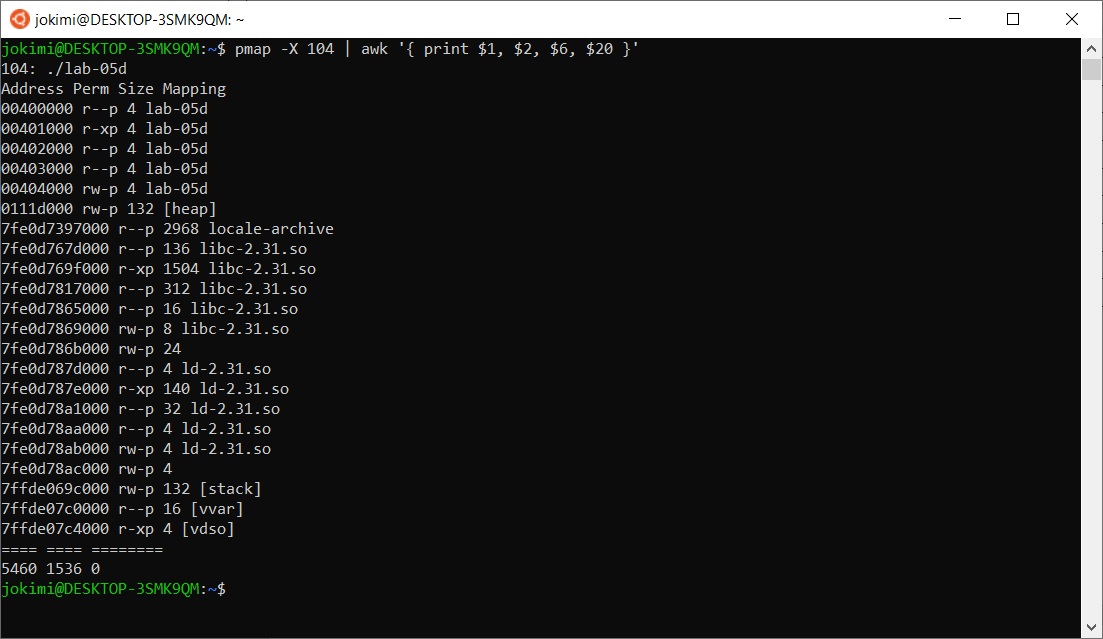
Результат выполнения в консоли:



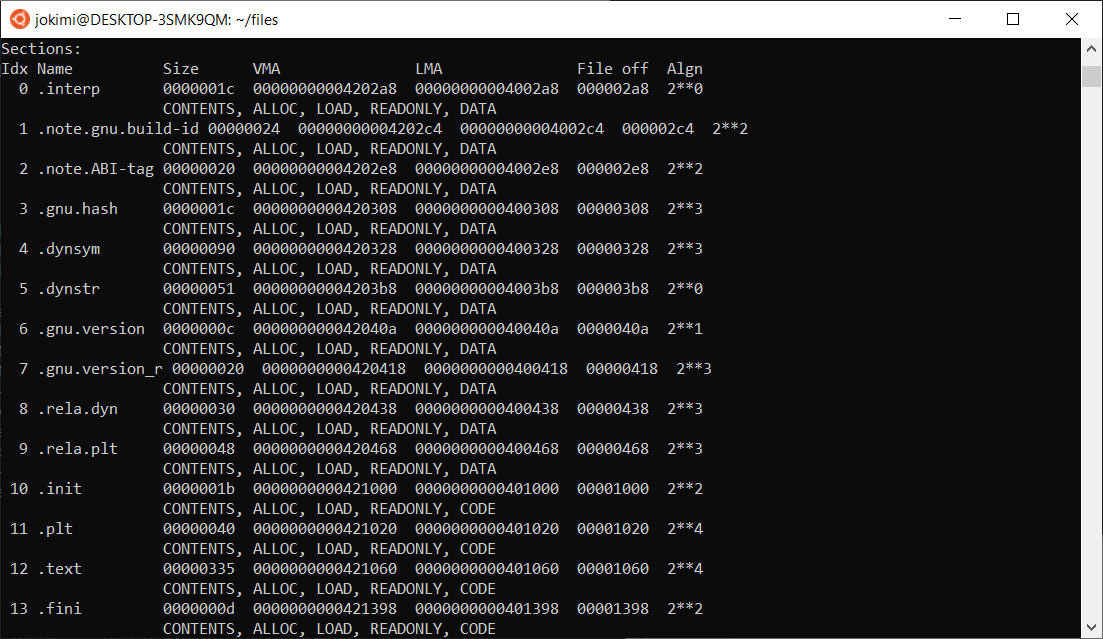
|  |  |  |
| --- | --- | --- |
| **Вид переменной** | **Адрес** | **Название секции** |
| global init | 0x404040 | .data |
| global uninit | 0x404058 | .bss |
| global static init | 0x404048 | .data |
| global static uninit | 0x404054 | .bss |
| local init | 0x7FFDE06B9F7C | stack |
| local uninit | 0x7FFDE06B9F78 | stack |
| local static init | 0x404044 | .data |
| local static uninit | 0x404050 | .bss |
| function | 0x401150 | .text |
| argc | 0x7FFDE06B9F88 | stack |
| argv | 0x7FFDE06B9F80 | stack |

Стартовый адрес: 00400000

pmap:



objdump:

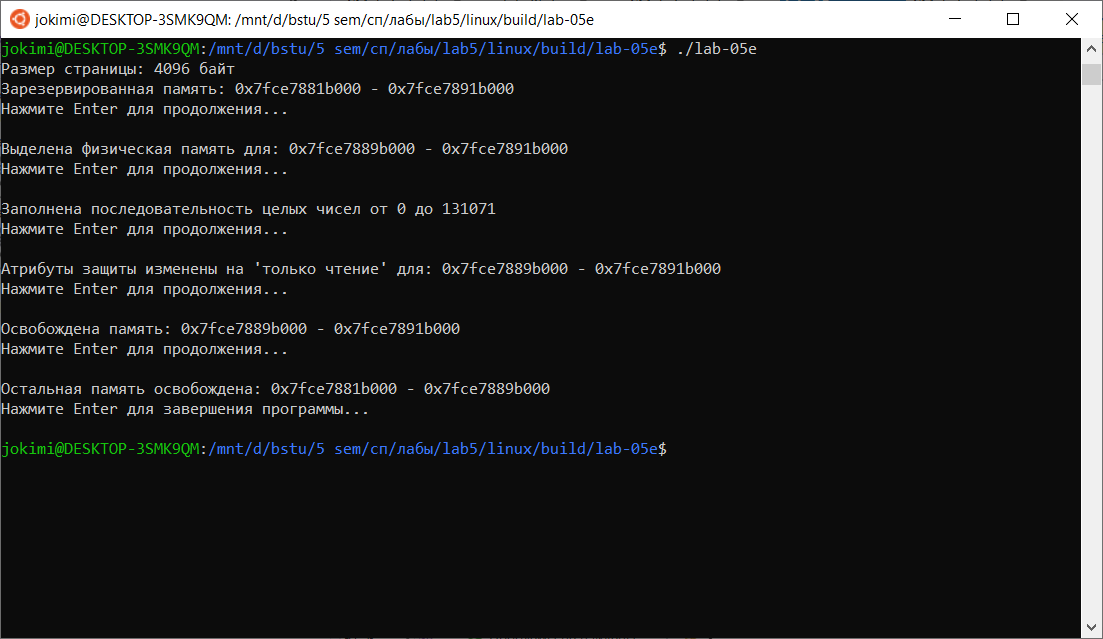


**Задание 5.** Разработайте консольное приложение **Lab-05e** в котором реализуйте код, который делится на следующие этапы:

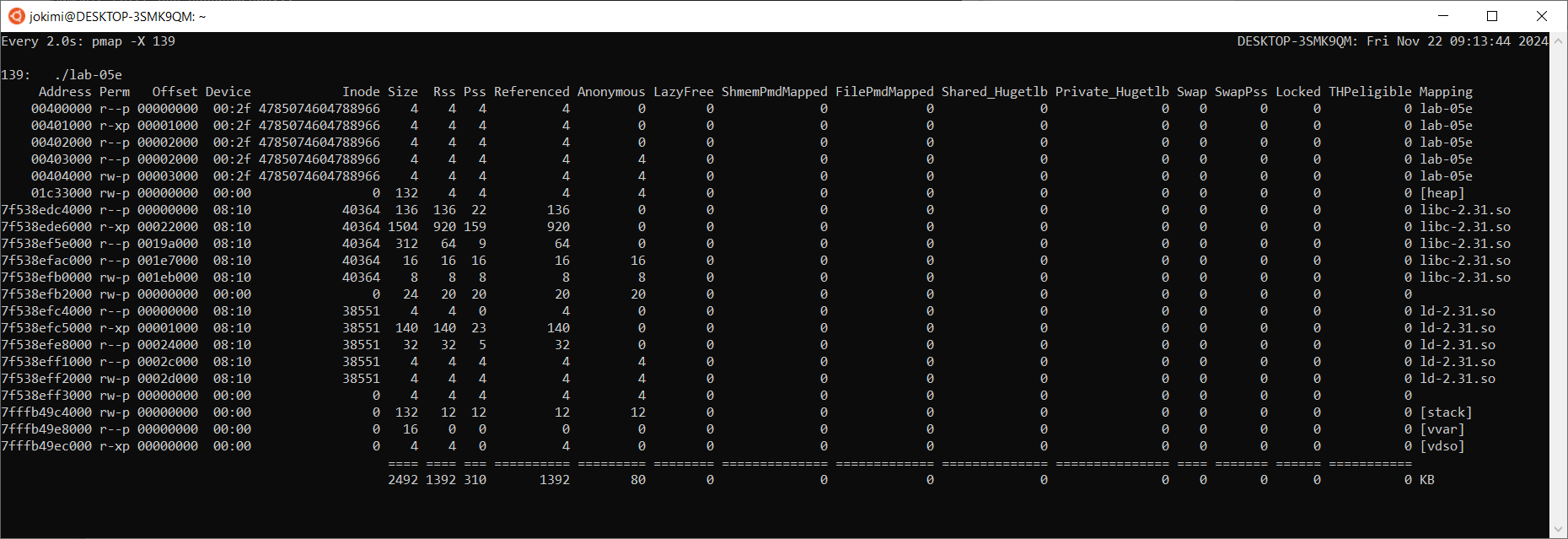
* Зарезервировать виртуальной памяти процесса на 256 страниц (размер страницы определить через вызов соответствующей функции) и вывести адрес области памяти на консоль;
* Для второй половины (128 страниц) зарезервированной области выделить физическую память (сменить с PROT\_NONE на PROT\_READ и PROT\_WRITE);
* Заполнить данную половину памяти последовательностью целых чисел начиная от 0 с шагом 1;
* Изменить атрибуты защиты страниц памяти на «только для чтения»;
* Освободить 128 виртуальных страниц в которые записан массив;
* Освободить оставшиеся страницы виртуальной памяти.

После каждого этапа программа должна ожидать ввода в консоль. В это время требуется просматривать информацию об областях памяти через утилиту **pmap**. Для удобства можно использовать: **watch pmap -X <PID>**.

Результат выполнения в консоли:



pmap:



**Резервирование памяти.** В программе используется mmap с флагами PROT\_NONE, чтобы зарезервировать виртуальную память (без привязки к физическим страницам). Команда pmap покажет новую область памяти, обозначенную как неактивную (---).

**Выделение физической памяти.** Использование mprotect для второй половины памяти (128 страниц) с флагами PROT\_READ | PROT\_WRITE приводит к созданию физических страниц и доступу на запись.

**Заполнение памяти.** Этот этап не меняет состояния памяти, так как мы только записываем данные в ранее выделенные страницы. pmap останется таким же, как на предыдущем этапе.

**Изменение защиты памяти на «только чтение».** После вызова mprotect с флагом PROT\_READ для второй половины (128 страниц), pmap отобразит доступ на чтение (r--).

**Освобождение второй половины.** При освобождении второй половины с помощью munmap, pmap покажет уменьшение размера региона. Освобождённая память исчезнет из таблицы.

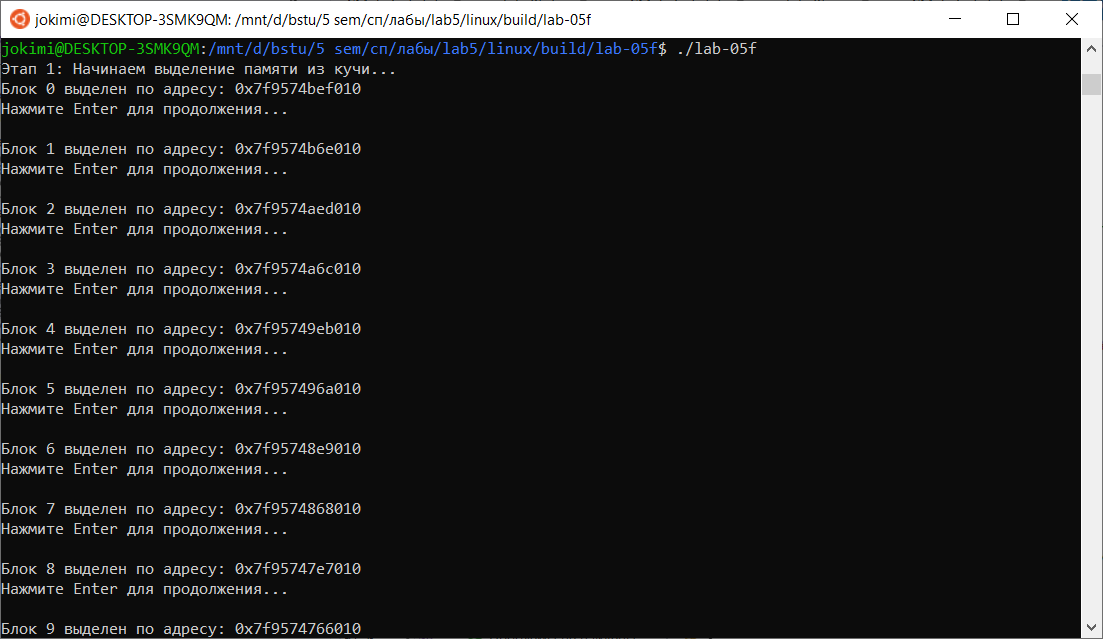
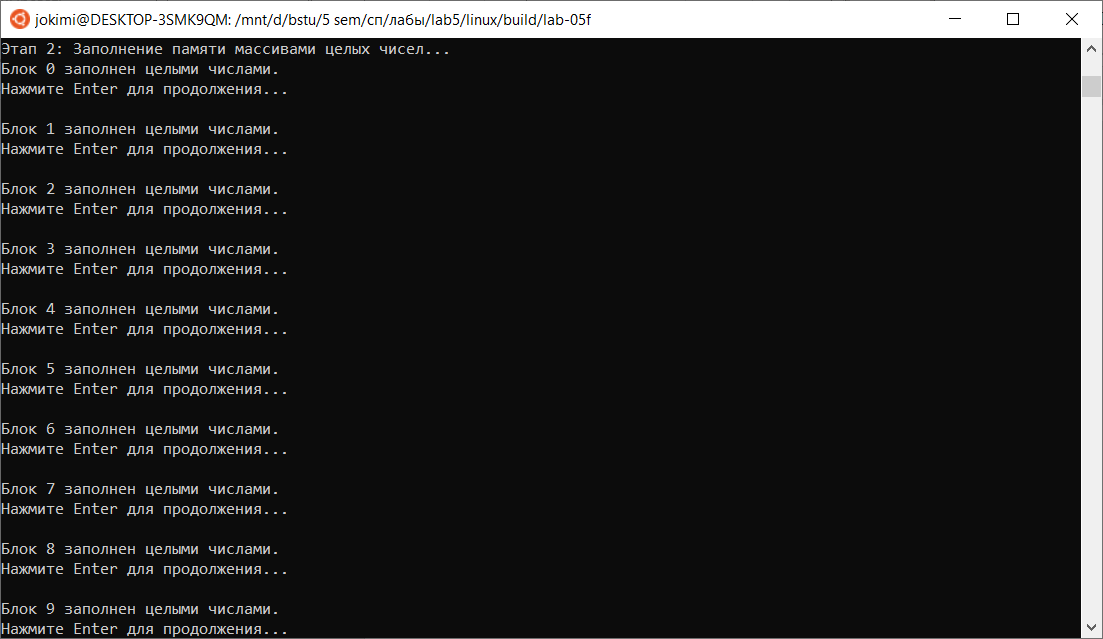
**Освобождение оставшихся страниц.** После полного освобождения памяти (первой половины) команда pmap больше не будет показывать выделенные области.

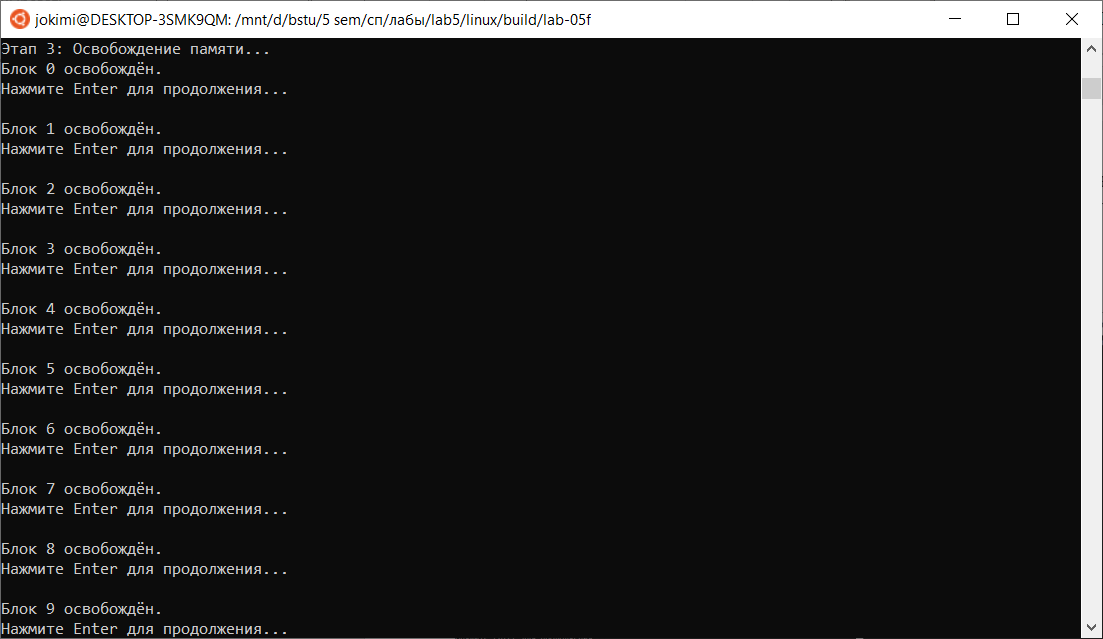
**Задание 6.** Разработайте консольное приложение **Lab-05f** в котором реализуйте код, который делится на следующие этапы:

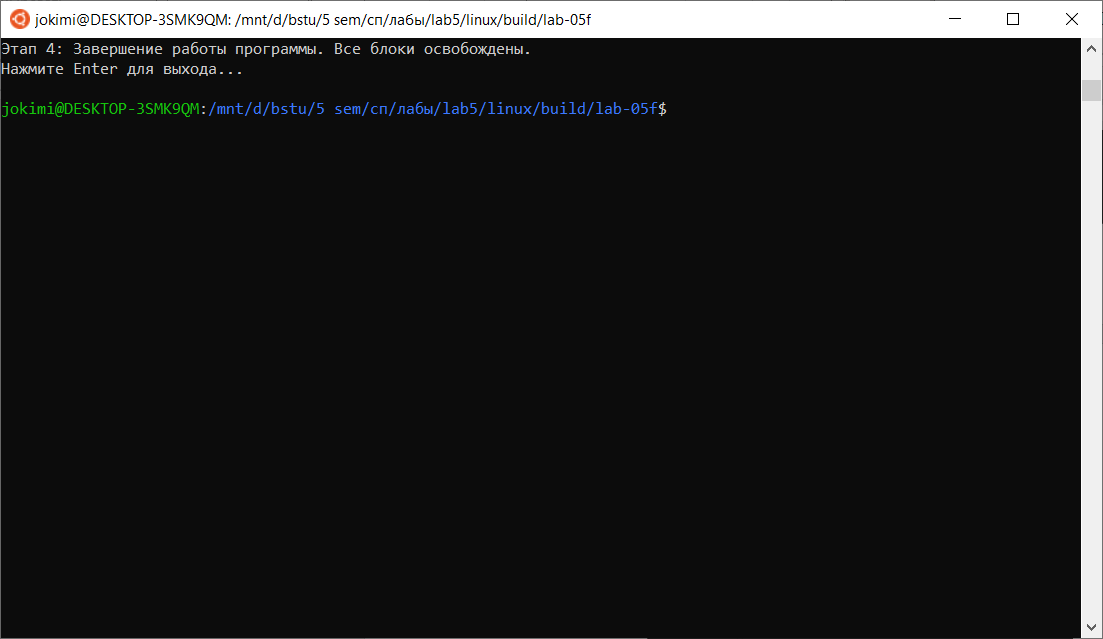
* В цикле выделить 10 блоков памяти из кучи размером 512 КиБ каждый;
* Заполнить каждый блок массивом целых чисел соответствующей длины (512 КиБ / 4 Б);
* В цикле освободить память, выделенную для всех блоков;
* Уничтожить кучу.

После каждого этапа программа должна ожидать ввода в консоль. В это время требуется просматривать информацию об областях памяти через утилиту **pmap**. Для удобства можно использовать: **watch pmap -X <PID>**.

Результат выполнения в консоли:





pmap:

