Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Організація обчислювальних процесів

Лабораторна робота №2

**«Аллокатор пам’яті загального призначення (частина 2)»**

Виконала:

студентка групи ІВ-71

Молчанова В.С.

Перевірив:

Сімоненко А.В.

Київ

2020 р.

### Опис розробленого алгоритму

1. Виділено декілька сторінок та кожна сторінка виділеної пам’яті містить опис:

// Сторінки розміром від 16 до 2048 + сторінка усієї пам'яті

struct memoryHeader

{

    void\* page16;

    size\_t page16Count;

    void\* page32;

    size\_t page32Count;

    void\* page64;

    size\_t page64Count;

    void\* page128;

    size\_t page128Count;

    void\* page256;

    size\_t page256Count;

    void\* page512;

    size\_t page512Count;

    void\* page1024;

    size\_t page1024Count;

    void\* page2048;

    size\_t page2048Count;

    void\* pageFull;

    size\_t pageFullCount;

};

/\*

\* Заголовок сторінки.

\* status – стан: вільний чи ні.

\* blockSize – розмір блоку.

\* freeBlockCount – кількість вільних блоків.

\* freeBlock – вільний блок.

\* nextPage – наступна сторінка.

\*/

struct pageHeader

{

    char status;

    size\_t blockSize;

    size\_t freeBlocksCount;

    void\* freeBlock;

    void\* nextPage;

};

2. Перевиділення пам’яті функцією mem\_realloc(void \*ptr, size\_t size):

a. Якщо ptr == NULL, то виконується виділення пам’яті заданого розміру за допомогою функції mem\_alloc(size).

b. Якщо розмір сторінки, що виділена під дані вказівника ptr, більший за size, то перевиділяємо пам’ять за допомогою функції mem\_free(ptr).

c. Якщо цього виявилося недостатньо, то знаходимо вільне місце для даних за допомогою mem\_alloc(size), переносимо дані й звільняємо необхідну сторінку.

3. Звільнення пам’яті mem\_free(void \*ptr) відбувається так, що алокатор рахує номер сторінки та помічає їй як + 1 вільний блок, якщо у результаті всі блоки у сторінці вільні, то й сама сторінка помічається як вільна.

У функції mem\_alloc() звертаємося до сторінок. У подальшому використовуються лише ті сторінки, до яких було хоча б одне звернення. При першому зверненні створюється масив з вільними сторінками. У даний список не входять сторінки, у яких нема вільних блоків. Далі звертаємося до пам’яті допоки не знайдемо сторінку, розмір якої більше заданого блоку. Але спочатку перевіряється, чи не є заданий розмір більшим за половину виділеної пам’яті. Надалі розмір ділиться на 2 та знов проходить перевірку, пошук необхідної сторінки.

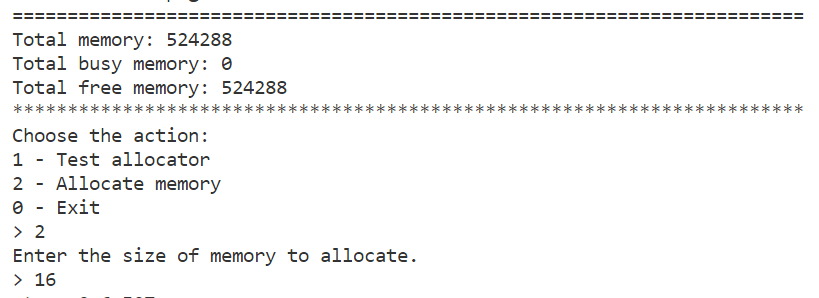
Оскільки всі сторінки вирівнени, при звертанні до інформації за адресою сторінки, звертання відбувається лише до однієї сторінки.

Таким чином, алокатор звертається до меншої кількості сторінок, а отже працює ефективно.

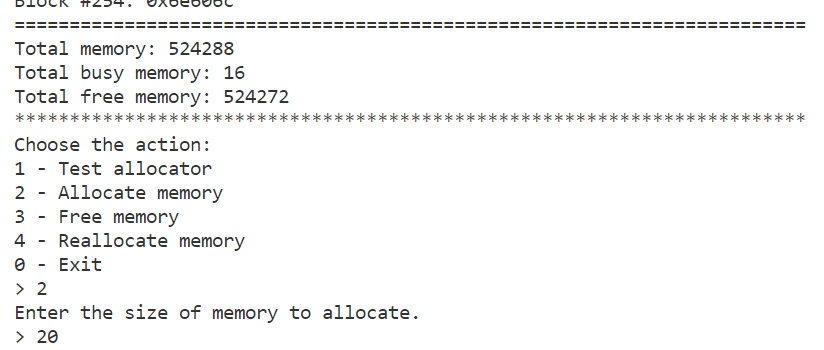
### Результати роботи програми

#### Алокація

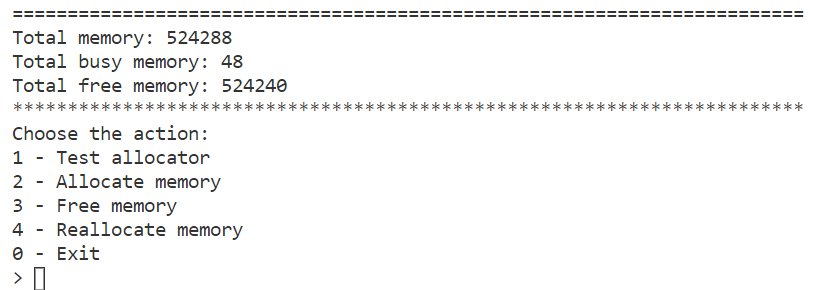
1. Початковий стан системи: наявна пам’ять, вся вільна, оскільки ще не було алоковано жодного блоку. Алокуємо перший блок на 16 байт



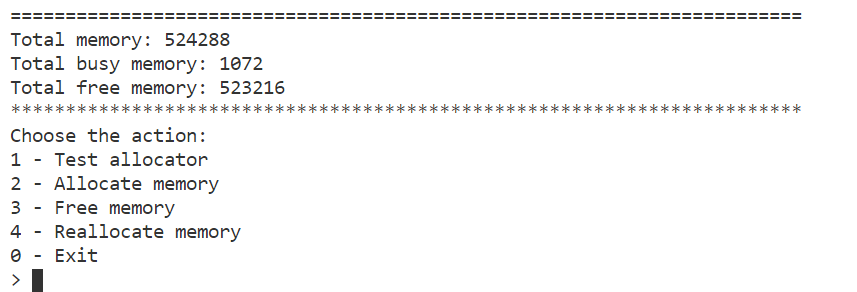
1. Алокуємо другий блок на 20 байт



1. Алоковано два блоки пам’яті. Перший блок на 16 байт, другий – 20. Оскільки перший блок займає комірку на 16 байт, а другий – на 32, загалом зайнято 16+32=48 байт пам’ті.



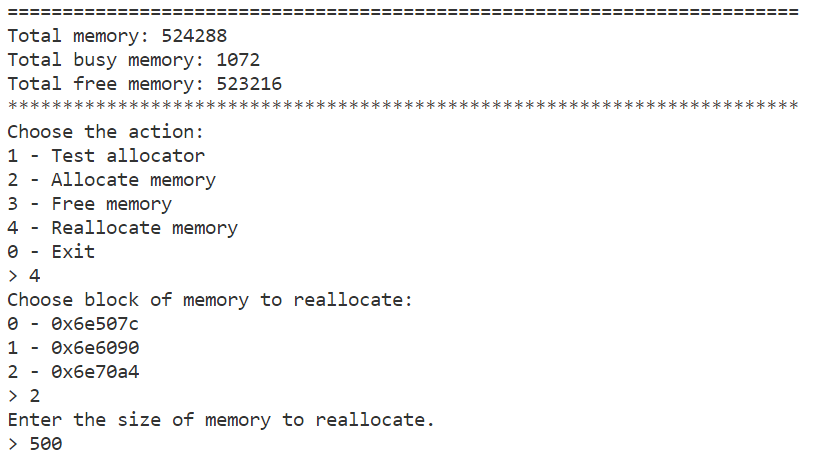
1. Алоковано ща один блок на 1000 байт, що займає 1024 байти, та загалом 1024+48=1072.



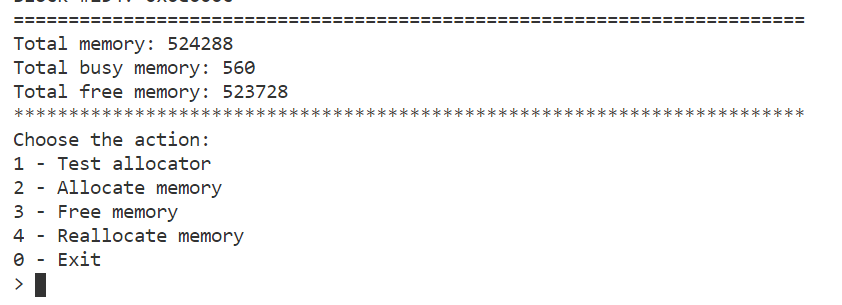
#### Реалокація

Після алокації першого блоку з’являється можливість реалоковувати блок.

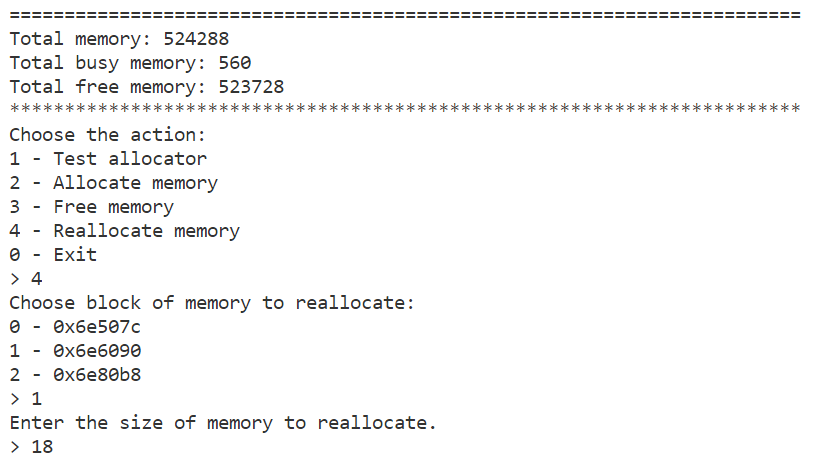
1. Реалокуємо останній блок до 500 байт



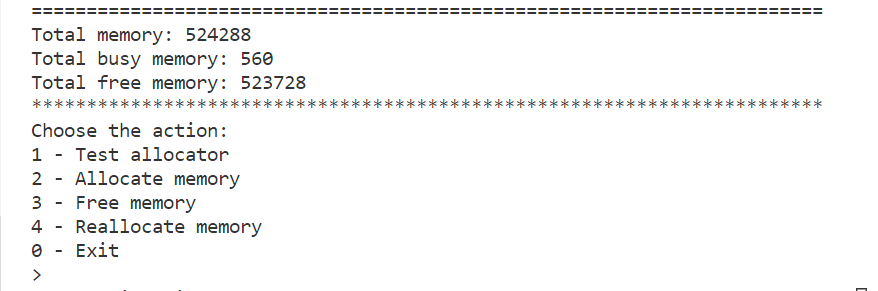
1. Останній алокований блок на 1000 байт було реалоковано до 500, тому останній (третій) блок став займати 512 байт, а в загальному обсяг зайнятої пам’яті становить 512 + 16 + 32 = 560 байт.



1. Спробуємо реалокувати другий блок до 18 байт:

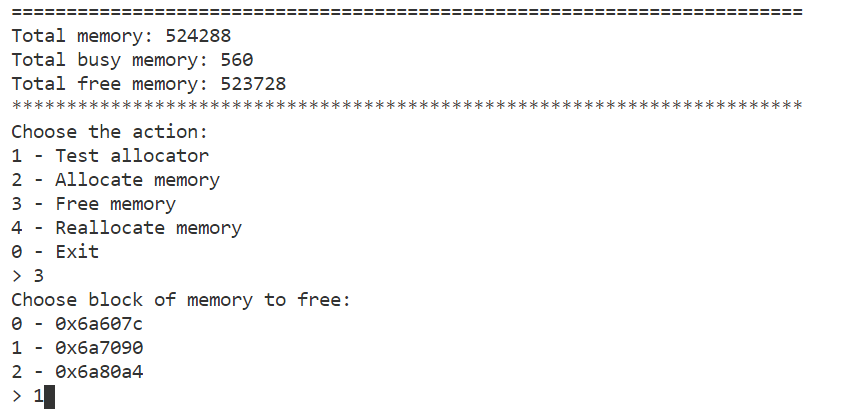


1. Загальний розмір не змінився, адже блок все одно завеликий для комірки в 16 байт і продовжує займати комірку в 32 байти.



#### Звільнення пам’яті

1. Звільнимо другий блок, який займає комірку у 32 байти:



1. Тепер загалом зайнято 560 – 32 = 528 байтів пам’яті:

