Министерство науки и высшего образования Российской Федерации
Мытищинский филиал
Федерального государственного автономного образовательного учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Космический

КАФЕДРА К3

отчет

К ЛАБОРАТОРНОЙ РАБОТЕ

№ 1

по ДИСЦИПЛИНЕ

«ОПЕРАЦИОННЫЕ СИСТЕМЫ»

Студент К3-76Б — В. Д. Чернов
(Группа) (И.О.Фамилия)

Преподаватель — А. В. Чернышов
(И.О.Фамилия)

2025 г.

Задание к лабораторной работе

Написать на языке Си программу. Демонстрирующую способ разделения памяти со стратегией размещения следующий подходящий, структурой данных блоки фиксированного размера и избегающего малых дыр.

РЕШЕНИЯ ЗАДАНИЙ

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>

#define POOL_SIZE 1024 * 64
#define BLOCK_HEADER_SIZE sizeof(BlockHeader)

typedef struct BlockHeader {
    size_t size;
    bool free;
    struct BlockHeader* next;
} BlockHeader;

static void* memory_pool = NULL;
static BlockHeader* free_list = NULL;
static BlockHeader* last_alloc = NULL;

void init_memory_pool() {
    memory_pool = malloc(POOL_SIZE);
    if (!memory_pool) {
        printf("Ошибка: не удалось выделить память под пул.\n");
        exit(1);
    }

    free_list = (BlockHeader*)memory_pool;
    free_list->size = POOL_SIZE - BLOCK_HEADER_SIZE;
    free_list->free = true;
    free_list->next = NULL;

    last_alloc = free_list;

    printf("Память инициализирована: %zu байт\n", POOL_SIZE);
}
```

```
void* my_malloc(size_t size) {
    if (size == 0) return NULL;

    BlockHeader* current = last_alloc;
    BlockHeader* start = current;

    do {
        if (current->free && current->size >= size) {
            size_t remaining = current->size - size;

            if (remaining > BLOCK_HEADER_SIZE) {
                BlockHeader* new_block = (BlockHeader*)((char*)current +
BLOCK_HEADER_SIZE + size);
                new_block->size = remaining - BLOCK_HEADER_SIZE;
                new_block->free = true;
                new_block->next = current->next;
                current->next = new_block;
                current->size = size;
            }

            current->free = false;
            last_alloc = current;

            printf("Выделено %zu байт по адресу %p\n", size, (void*)((char*)current +
BLOCK_HEADER_SIZE));
            return (char*)current + BLOCK_HEADER_SIZE;
        }

        current = current->next ? current->next : free_list;

    } while (current != start);

    printf("Ошибка: недостаточно памяти для %zu байт.\n", size);
    return NULL;
}
```

```
void merge_free_blocks() {
    BlockHeader* current = free_list;
    while (current && current->next) {
        if (current->free && current->next->free) {
            current->size += BLOCK_HEADER_SIZE + current->next->size;
            current->next = current->next->next;
        } else {
            current = current->next;
        }
    }
}

void my_free(void* ptr) {
    if (!ptr) return;

    BlockHeader* block = (BlockHeader*)((char*)ptr - BLOCK_HEADER_SIZE);
    block->free = true;
    merge_free_blocks();

    printf("Освобожден блок по адресу %p (%zu байт)\n", ptr, block->size);
}

void print_memory_state() {
    BlockHeader* current = free_list;
    while (current) {
        printf("Блок %p | размер: %zu | %s\n",
               (void*)current, current->size,
               current->free ? "свободен" : "занят");
        current = current->next;
    }
    printf("\n");
}

int main() {
    init_memory_pool();

    void* a = my_malloc(1000);
```

```
    void* b = my_malloc(2000);
    void* c = my_malloc(3000);

    print_memory_state();

    my_free(b);
    print_memory_state();

    void* d = my_malloc(1500);
    print_memory_state();

    my_free(a);
    my_free(c);
    my_free(d);

    print_memory_state();

    free(memory_pool);
    return 0;
}
```

Результат работы

Память инициализирована: 65536 байт
Выделено 1000 байт по адресу 0x620623cc5928
Выделено 2000 байт по адресу 0x620623cc5d28
Выделено 3000 байт по адресу 0x620623cc6510

Блок 0x620623cc5910 | размер: 1000 | занят
Блок 0x620623cc5d10 | размер: 2000 | занят
Блок 0x620623cc64f8 | размер: 3000 | занят
Блок 0x620623cc70c8 | размер: 59440 | свободен

Освобожден блок по адресу 0x620623cc5d28 (2000 байт)

Блок 0x620623cc5910 | размер: 1000 | занят
Блок 0x620623cc5d10 | размер: 2000 | свободен
Блок 0x620623cc64f8 | размер: 3000 | занят
Блок 0x620623cc70c8 | размер: 59440 | свободен

Выделено 1500 байт по адресу 0x620623cc70e0

Блок 0x620623cc5910 | размер: 1000 | занят
Блок 0x620623cc5d10 | размер: 2000 | свободен
Блок 0x620623cc64f8 | размер: 3000 | занят
Блок 0x620623cc70c8 | размер: 1500 | занят
Блок 0x620623cc76bc | размер: 57916 | свободен

Освобожден блок по адресу 0x620623cc5928 (3024 байт)
Освобожден блок по адресу 0x620623cc6510 (3000 байт)
Освобожден блок по адресу 0x620623cc70e0 (1500 байт)

Блок 0x620623cc5910 | размер: 65512 | свободен