Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №4 по курсу

«Операционные системы»

Группа: М8О-210БВ-24

Студент: Дмитренко Я.С.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 26.11.25

Москва, 2025

Постановка задачи

Вариант 3.

1. Расчет интеграла функции sin(x) на отрезке [A, B] с шагом e: Сигнатура функции: float sin_integral(float a, float b, float e);

● Реализация №1: Подсчет интеграла методом прямоугольников;

● Реализация №2: Подсчет интеграла методом трапеций.

4. Подсчёт наибольшего общего делителя для двух натуральных чисел:

Сигнатура функции: int gcd(int a, int b);

● Реализация №1: Алгоритм Евклида

● Реализация №2: Наивный алгоритм: пытаться разделить числа на все числа, что меньше a и b.

Общий метод и алгоритм решения

Использованные системные вызовы:

- void *dlopen(const char *filename, int flags); - загружает в память и открывает динамическую библиотеку.
- void *dlsym(void *handle, const char *symbol); - возвращает адрес функции или переменной из загруженной библиотеки.
- int dlclose(void *handle); - выгружает из памяти ранее загруженную динамическую библиотеку.

Я создал две динамические библиотеки (libmy1.so и libmy2.so), которые реализуют контракты двух функций: sin_integral(float a, float b, float e) для вычисления интеграла функции sin(x) на отрезке [a, b] с шагом e и gcd(int a, int b) для вычисления наибольшего общего делителя двух чисел. Первая библиотека использует метод прямоугольников для вычисления интеграла и алгоритм Евклида для вычисления НОД. Вторая библиотека использует метод трапеций для вычисления интеграла и наивный алгоритм (перебор делителей) для вычисления НОД.

Я реализовал два способа использования этих библиотек.

Первый способ - статическая линковка**.** Исходный код библиотечных функций (lib1.c) компилируется непосредственно в исполняемый файл программы static с использованием флага -static. Математические функции из системной библиотеки также включаются статически с помощью флага -lm. В результате создается полностью самодостаточный исполняемый файл, который не зависит от внешних динамических библиотек во время выполнения. Реализация фиксирована на этапе компиляции (используется только код из lib1.c), поэтому невозможно переключить реализацию во время выполнения программы.

Второй способ - загрузка во время выполнения. Библиотеки загружаются программно через интерфейс операционной системы для работы с динамическими библиотеками с использованием функций dlopen(), dlsym() и dlclose(). Это позволяет переключаться между реализациями libmy1.so и libmy2.so во время работы программы dynamic по команде пользователя (команда 0). Такой подход предоставляет возможность переключать реализации на лету, обновлять библиотеки без перекомпиляции основной программы и более гибко управлять ресурсами, поскольку библиотеки можно загружать и выгружать по необходимости.

Код программы

lib.h:

```
#pragma once
#include <stdlib.h>

float sin_integral(float a, float b, float e);
int gcd(int a, int b);
```

lib1.c:

```
#include "lib.h"
#include <math.h>

// Реализация №1: Метод прямоугольников
float sin_integral(float a, float b, float e) {
    float integral = 0.0;
    for (float x = a; x < b; x += e) {
        integral += sin(x) * e;
    }
    return integral;
}

/* Алгоритм Евклида */
int gcd(int a, int b) {
    while (b != 0) {
        int t = b;
        b = a % b;
        a = t;
    }
    return a;
}
```

lib2.c:

```
#include "lib.h"
#include <math.h>

// Реализация №2: Метод трапеций
float sin_integral(float a, float b, float e) {
    float integral = 0.0;
    float n = (b - a) / e;
    for (int i = 0; i < n; i++) {
        float x1 = a + i * e;
        float x2 = a + (i + 1) * e;
        integral += (sin(x1) + sin(x2)) * e / 2;
    }
    return integral;
}

/* Наивный алгоритм: пытаться разделить числа на
```

```
все числа, что меньше a и b.  */
int gcd(int a, int b) {
    int min = (a < b) ? a : b;
    for (int i = min; i >= 1; i--) {
        if (a % i == 0 && b % i == 0)
            return i;
    }
    return 1;
}
```

static_part.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "lib.h"

int main() {

    printf("  1 a b e - интеграл sin(x) от a до b с шагом e\n");
    printf("  2 a b - НОД чисел a и b\n");
    printf("  exit - выход\n");

    char input[256];

    while (1) {
        printf("\n> ");
        fflush(stdout);

        if (!fgets(input, sizeof(input), stdin)) {
            break;
        }

        // Убираем символ новой строки
        input[strcspn(input, "\n")] = '\0';

        if (strcmp(input, "exit") == 0) {
            break;
        } else if (input[0] == '1') {
            // Интеграл sin(x)
            float a, b, e;
            if (sscanf(input + 1, "%f %f %f", &a, &b, &e) == 3) {
                float result = sin_integral(a, b, e);
                printf("Интеграл sin(x) на [%.2f, %.2f] с шагом %.4f = %.6f\n",
                       a, b, e, result);
            } else {
                printf("Ошибка: нужно 3 аргумента: a b e\n");
            }
        } else if (input[0] == '2') {
            // НОД
            int a, b;
            if (sscanf(input + 1, "%d %d", &a, &b) == 2) {
                int result = gcd(a, b);
```

```
                printf("НОД(%d, %d) = %d\n", a, b, result);
            } else {
                printf("Ошибка: нужно 2 аргумента: a b\n");
            }
        } else {
            printf("Неизвестная команда\n");
        }
    }

    return 0;
}
```

dynamic_part.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dlfcn.h>
#include <stdbool.h>
#include "lib.h"

// Объявления типов функций
typedef float (*sin_integral_func)(float, float, float);
typedef int (*gcd_func)(int, int);

int main() {
    void *lib_handle = NULL;
    sin_integral_func sin_integral = NULL;
    gcd_func gcd = NULL;

    // Загружаем первую реализацию по умолчанию
    lib_handle = dlopen("./libmy1.so", RTLD_LAZY);
    if (!lib_handle) {
        fprintf(stderr, "Ошибка загрузки библиотеки: %s\n", dlerror());
        return 1;
    }

    // Получаем указатели на функции
    sin_integral = (sin_integral_func)dlsym(lib_handle, "sin_integral");
    gcd = (gcd_func)dlsym(lib_handle, "gcd");

    if (!sin_integral || !gcd) {
        fprintf(stderr, "Ошибка получения функций: %s\n", dlerror());
        dlclose(lib_handle);
        return 1;
    }

    printf("  0 - переключить реализацию (1 или 2)\n");
    printf("  1 a b e - интеграл sin(x) от a до b с шагом e\n");
    printf("  2 a b - НОД чисел a и b\n");
    printf("  exit - выход\n");

    char input[256];
```

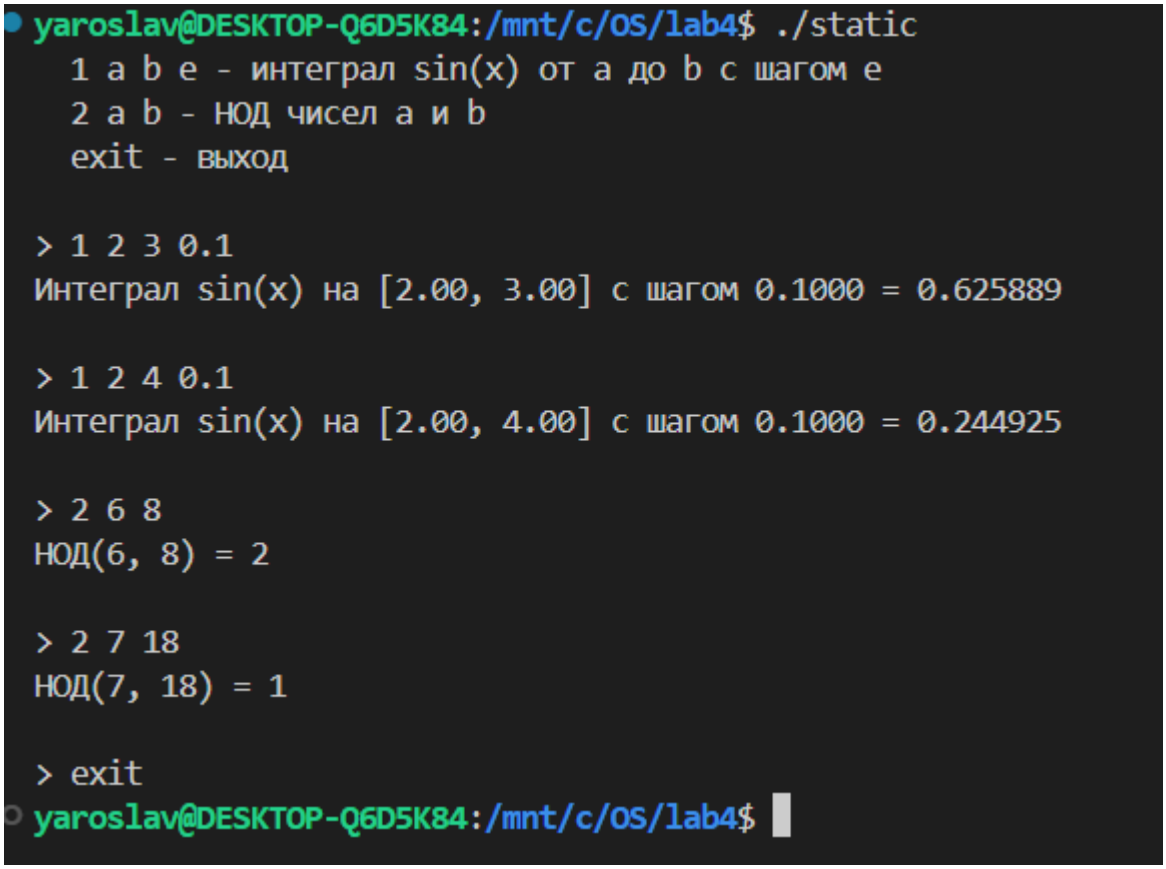
```
    bool using_lib1 = true;  // true = lib1, false = lib2

    while (1) {
        printf("\n> ");
        fflush(stdout);

        if (!fgets(input, sizeof(input), stdin)) {
            break;
        }

        input[strcspn(input, "\n")] = '\0';

        if (strcmp(input, "exit") == 0) {
            break;
        } else if (strcmp(input, "0") == 0) {
            // Переключение реализации
            dlclose(lib_handle);

            if (using_lib1) {
                // Переключаемся на lib2
                lib_handle = dlopen("./libmy2.so", RTLD_LAZY);
                printf("Переключено на реализацию 2 (трапеции + наивный алгоритм)\n");
            } else {
                // Переключаемся на lib1
                lib_handle = dlopen("./libmy1.so", RTLD_LAZY);
                printf("Переключено на реализацию 1 (прямоугольники + Евклид)\n");
            }

            if (!lib_handle) {
                fprintf(stderr, "Ошибка загрузки библиотеки: %s\n", dlerror());
                return 1;
            }

            // Обновляем указатели на функции
            sin_integral = (sin_integral_func)dlsym(lib_handle, "sin_integral");
            gcd = (gcd_func)dlsym(lib_handle, "gcd");

            if (!sin_integral || !gcd) {
                fprintf(stderr, "Ошибка получения функций: %s\n", dlerror());
                dlclose(lib_handle);
                return 1;
            }

            using_lib1 = !using_lib1;

        } else if (input[0] == '1') {
            // Интеграл sin(x)
            float a, b, e;
            if (sscanf(input + 1, "%f %f %f", &a, &b, &e) == 3) {
                float result = sin_integral(a, b, e);
                printf("Интеграл sin(x) на [%.2f, %.2f] с шагом %.4f = %.6f\n",
                       a, b, e, result);
            } else {
                printf("Ошибка: нужно 3 аргумента: a b e\n");
```

```
            }
        } else if (input[0] == '2') {
            // НОД
            int a, b;
            if (sscanf(input + 1, "%d %d", &a, &b) == 2) {
                int result = gcd(a, b);
                printf("НОД(%d, %d) = %d\n", a, b, result);
            } else {
                printf("Ошибка: нужно 2 аргумента: a b\n");
            }
        } else {
            printf("Неизвестная команда\n");
        }
    }

    dlclose(lib_handle);
    return 0;
}
```

Протокол работы программы

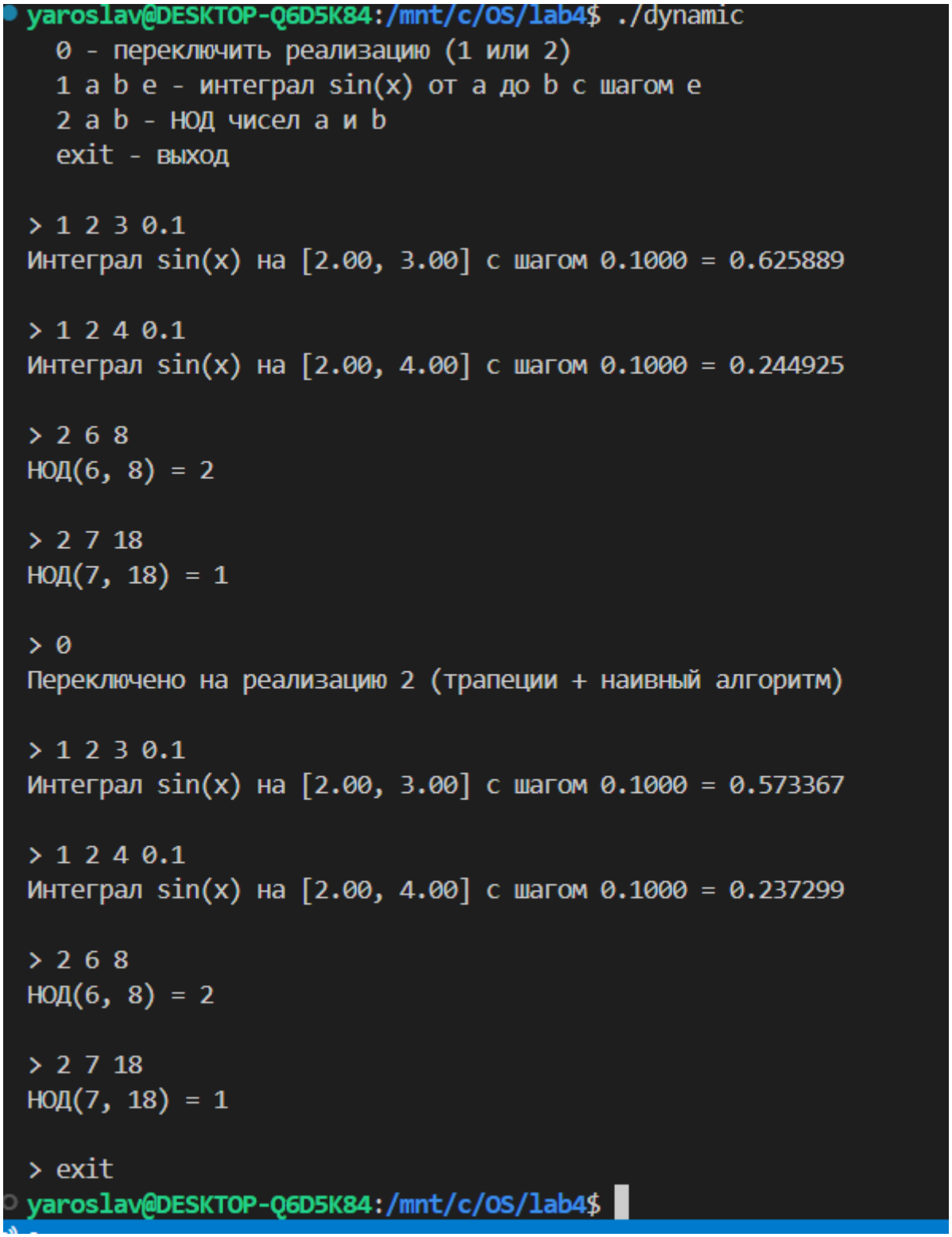
Тестирование:

```
yaroslav@DESKTOP-Q6D5K84:/mnt/c/OS/lab4$ ./static
  1 a b e - интеграл sin(x) от a до b с шагом e
  2 a b - НОД чисел a и b
  exit - выход

> 1 2 3 0.1
Интеграл sin(x) на [2.00, 3.00] с шагом 0.1000 = 0.625889

> 1 2 4 0.1
Интеграл sin(x) на [2.00, 4.00] с шагом 0.1000 = 0.244925

> 2 6 8
НОД(6, 8) = 2

> 2 7 18
НОД(7, 18) = 1

> exit
yaroslav@DESKTOP-Q6D5K84:/mnt/c/OS/lab4$
```




```
yaroslav@DESKTOP-Q6D5K84:/mnt/c/OS/lab4$ ./dynamic
  0 - переключить реализацию (1 или 2)
  1 a b e - интеграл sin(x) от a до b с шагом e
  2 a b - НОД чисел a и b
  exit - выход

> 1 2 3 0.1
Интеграл sin(x) на [2.00, 3.00] с шагом 0.1000 = 0.625889

> 1 2 4 0.1
Интеграл sin(x) на [2.00, 4.00] с шагом 0.1000 = 0.244925

> 2 6 8
НОД(6, 8) = 2

> 2 7 18
НОД(7, 18) = 1

> 0
Переключено на реализацию 2 (трапеции + наивный алгоритм)

> 1 2 3 0.1
Интеграл sin(x) на [2.00, 3.00] с шагом 0.1000 = 0.573367

> 1 2 4 0.1
Интеграл sin(x) на [2.00, 4.00] с шагом 0.1000 = 0.237299
```

```
> 2 6 8
НОД(6, 8) = 2

> 2 7 18
НОД(7, 18) = 1

> exit
yaroslav@DESKTOP-Q6D5K84:/mnt/c/OS/lab4$
```



Вывод

В ходе работы я освоил создание динамических библиотек и их использование двумя способами: через линковку при компиляции и через динамическую загрузку во время выполнения.