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-214БВ-24

Студент: Шитов Никита Владиславович

Преподаватель: Бахарев В.Д. (ФИИТ)

Оценка: _____

Дата: 14.12.25

Москва, 2025

Постановка задачи

Вариант 14.

Задание

Требуется создать динамические библиотеки, которые реализуют функции вычисления производной косинуса в точке с приращением двумя вариантами и сортировку массива двумя видами. Далее использовать данные библиотеки 2-мя способами:

1. Во время компиляции (на этапе линковки)

2. Во время исполнения программы. Библиотеки загружаются в память с помощью интерфейса ОС для работы с динамическими библиотеками.

Провести анализ двух типов использования библиотек.

2. **Расчет производной функции cos(x) в точке a с приращением dx:**
   Сигнатура функции:
   ```
   float cos_derivative(float a, float dx);
   ```
   - Реализация №1: f'(x) = (f(a + dx) - f(a)) / dx
   - Реализация №2: f'(x) = (f(a + dx) - f(a - dx)) / (2dx)

8. **Перевод числа x из десятичной системы счисления в другую:**
   Сигнатура функции: `char *convert(int x);`
   - Реализация №1: Перевод в двоичную
   - Реализация №2: Перевод в троичную

Общий метод и алгоритм решения

Использованные системные вызовы:

- **dlopen()** — загрузка динамической библиотеки (.so) в адресное пространство процесса во время выполнения.
- **dlsym()** — получение указателя на функцию по её имени из уже загруженной библиотеки.
- **dlclose()** — выгрузка библиотеки из памяти и освобождение связанных ресурсов.
- **dlerror()** — проверка ошибок при работе с динамическими библиотеками.
- **fgets() / sscanf()** — чтение и разбор пользовательского ввода.
- **printf() / puts()** — вывод результатов.
- **malloc() / free()** — динамическое выделение и освобождение памяти.

В лабораторной работе реализованы две программы: расчёт производной cos(x) и перевод числа из десятичной системы в другую (двоичную/троичную). Первая программа использует статическую линковку — реализации фиксированы на этапе компиляции, команда «0» только выводит информацию. Вторая программа загружает динамические библиотеки во время выполнения с помощью dlopen, dlsym, dlclose, что позволяет переключать реализации по команде «0» — например, с односторонней разности (менее точной) на центральную (высокая точность) и с двоичного на троичный перевод. Обе программы корректно обрабатывают ввод, управляют памятью и демонстрируют ключевые различия между статическим и динамическим связыванием.

Код программы

contract.h

```c
#ifndef CONTRACT_H
#define CONTRACT_H

float cos_derivative(float a, float dx);
char* convert(int x);

#endif
```

conv1.c

```c
#include <stdlib.h>
#include <string.h>
#include "contract.h"

char* convert(int x) {
    if (x == 0) {
        char* res = malloc(2);
        strcpy(res, "0");
        return res;
    }

    int neg = (x < 0);
    unsigned int v = neg ? -(unsigned int)x : (unsigned int)x;

    char buf[34];
    int i = sizeof(buf) - 1;
    buf[i] = '\0';

    while (v) {
        buf[--i] = '0' + (v & 1);
        v >>= 1;
    }
    if (neg) buf[--i] = '-';

    char* res = malloc(sizeof(buf) - i);
    strcpy(res, buf + i);
    return res;
}
```

conv2.c

```c
#include <stdlib.h>
#include <string.h>
#include "contract.h"

char* convert(int x) {
    if (x == 0) {
        char* res = malloc(2);
        strcpy(res, "0");
        return res;
    }
```

```
    int neg = (x < 0);
    int v = neg ? -x : x;

    char buf[25];
    int i = sizeof(buf) - 1;
    buf[i] = '\0';

    while (v) {
        int r = v % 3;
        buf[--i] = '0' + r;
        v /= 3;
    }
    if (neg) buf[--i] = '-';

    char* res = malloc(sizeof(buf) - i);
    strcpy(res, buf + i);
    return res;
}
```

deriv1.c

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

float cos_derivative(float a, float dx) {
    return (cosf(a + dx) - cosf(a)) / dx;
}
```

deriv2.c

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

float cos_derivative(float a, float dx) {
    return (cosf(a + dx) - cosf(a - dx)) / (2.0f * dx);
}
```

prog1.c

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

int main() {
    printf("Программа №1\n");
    printf("Команды:\n  0 — информация\n  1 a dx — производная\n  2 x — перевод в
систему\n  exit — выход\n");

    char line[128];
    while (fgets(line, sizeof(line), stdin)) {
        if (strncmp(line, "exit", 4) == 0) break;

        int cmd;
        if (sscanf(line, "%d", &cmd) != 1) continue;

        if (cmd == 0) {
```

```
            printf("Используются реализации:\n");
            printf("  cos_derivative: (f(a+dx)-f(a))/dx\n");
            printf("  convert: двоичная система\n");
        } else if (cmd == 1) {
            float a, dx;
            if (sscanf(line, "%*d %f %f", &a, &dx) == 2) {
                float r = cos_derivative(a, dx);
                printf("cos'(%g) при dx=%g = %.8f\n", a, dx, r);
            } else {
                printf("Ошибка: требуется '1 a dx'\n");
            }
        } else if (cmd == 2) {
            int x;
            if (sscanf(line, "%*d %d", &x) == 1) {
                char* s = convert(x);
                printf("convert(%d) = %s\n", x, s);
                free(s);
            } else {
                printf("Ошибка: требуется '2 x'\n");
            }
        } else {
            printf("Неизвестная команда\n");
        }
    }
    return 0;
}
```

prog2.c

```
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dlfcn.h>
#include <string.h>
#include "contract.h"


static float (*cos_derivative_ptr)(float, float) = NULL;
static char* (*convert_ptr)(int) = NULL;

static void *deriv_lib = NULL;
static void *conv_lib = NULL;

const char *deriv_libs[2] = {"./libderiv1.so", "./libderiv2.so"};
const char *conv_libs[2] = {"./libconv1.so", "./libconv2.so"};

int load_version(int version) {
    if (deriv_lib) dlclose(deriv_lib);
    if (conv_lib) dlclose(conv_lib);

    deriv_lib = dlopen(deriv_libs[version], RTLD_LAZY);
    if (!deriv_lib) {
        fprintf(stderr, "Ошибка загрузки %s: %s\n", deriv_libs[version], dlerror());
        return -1;
    }
```

```
    conv_lib = dlopen(conv_libs[version], RTLD_LAZY);
    if (!conv_lib) {
        fprintf(stderr, "Ошибка загрузки %s: %s\n", conv_libs[version], dlerror());
        dlclose(deriv_lib);
        deriv_lib = NULL;
        return -1;
    }

    cos_derivative_ptr = (float (*)(float, float)) dlsym(deriv_lib, "cos_derivative");
    convert_ptr = (char* (*)(int)) dlsym(conv_lib, "convert");

    char *err = dlerror();
    if (err) {
        fprintf(stderr, "Ошибка dlsym: %s\n", err);
        dlclose(deriv_lib); dlclose(conv_lib);
        deriv_lib = conv_lib = NULL;
        return -1;
    }

    printf("Переключено на реализацию №%d\n", version + 1);
    printf("   cos_derivative: %s\n", version == 0 ?
        "(f(a+dx)-f(a))/dx" : "(f(a+dx)-f(a-dx))/(2dx)");
    printf("   convert: %s\n", version == 0 ? "двоичная" : "троичная");
    return 0;
}

int main() {
    printf("Программа №2\n");
    printf("Команды:\n");
    printf("  0            – переключить реализацию (1 | 2)\n");
    printf("  1 a dx       – производная cos(x) в точке a с шагом dx\n");
    printf("  2 x          – перевод числа x в другую систему\n");
    printf("  exit         – выход\n");

    int current_version = 0;
    if (load_version(current_version) != 0) {
        return 1;
    }

    char line[256];
    while (fgets(line, sizeof(line), stdin)) {
        line[strcspn(line, "\n")] = '\0';

        if (strcmp(line, "exit") == 0) {
            break;
        }

        if (strlen(line) == 0) continue;

        int cmd;
        if (sscanf(line, "%d", &cmd) != 1) {
            printf("Ошибка: введите команду (0, 1, 2, exit)\n");
            continue;
        }
```

```
        if (cmd == 0) {
            current_version = 1 - current_version;
            if (load_version(current_version) != 0) {
                printf("Не удалось переключиться на реализацию %d\n", current_version + 1);
            }
        } else if (cmd == 1) {
            if (!cos_derivative_ptr) {
                printf("Ошибка: cos_derivative не загружена\n");
                continue;
            }

            float a, dx;
            if (sscanf(line, "%*d %f %f", &a, &dx) == 2) {
                float res = cos_derivative_ptr(a, dx);
                printf("cos'(%g) при dx=%g = %.8f (реализация №%d)\n", a, dx, res,
current_version + 1);
            } else {
                printf("Ошибка: требуется '1 a dx'\n");
            }
        } else if (cmd == 2) {
            if (!convert_ptr) {
                printf("Ошибка: convert не загружена\n");
                continue;
            }

            int x;
            if (sscanf(line, "%*d %d", &x) == 1) {
                char *s = convert_ptr(x);
                if (s) {
                    printf("convert(%d) = %s (реализация №%d)\n", x, s, current_version +
1);
                    free(s);
                } else {
                    printf("convert(%d) = (NULL)\n", x);
                }
            } else {
                printf("Ошибка: требуется '2 x'\n");
            }
        } else {
            printf("Неизвестная команда. Используйте 0, 1, 2 или exit.\n");
        }
    }

    if (deriv_lib) dlclose(deriv_lib);
    if (conv_lib) dlclose(conv_lib);

    return 0;
```

Протокол работы программы

динамическая загрузка - prog2

```
┌──(flow㉿kali)-[~/Documents/vsc/os/lab4]
└─$ cat input.txt
1 0.0 0.001
2 10
0
1 0.0 0.001
2 10
2 -5
1 1.5708 0.01
exit

┌──(flow㉿kali)-[~/Documents/vsc/os/lab4]
└─$ strace -o strace.log ./prog2 < input.txt
Программа №2
Команды:
  0            – переключить реализацию (1 | 2)
  1 a dx       – производная cos(x) в точке a с шагом dx
  2 x          – перевод числа x в другую систему
  exit         – выход
Переключено на реализацию №1
   cos_derivative: (f(a+dx)-f(a))/dx
   convert: двоичная
cos'(0) при dx=0.001 = -0.00047684 (реализация №1)
convert(10) = 1010 (реализация №1)
Переключено на реализацию №2
   cos_derivative: (f(a+dx)-f(a-dx))/(2dx)
   convert: троичная
cos'(0) при dx=0.001 = 0.00000000 (реализация №2)
convert(10) = 101 (реализация №2)
convert(-5) = -12 (реализация №2)
cos'(1.5708) при dx=0.01 = -0.99998242 (реализация №2)

┌──(flow㉿kali)-[~/Documents/vsc/os/lab4]
└─$
```

статическая линковка - prog1

```
┌──(flow㉿kali)-[~/Documents/vsc/os/lab4]
└─$ cat input_1.txt
1 0.0 0.001
2 10
2 -5
1 1.5708 0.01
exit

┌──(flow㉿kali)-[~/Documents/vsc/os/lab4]
└─$ ./prog1 < input_1.txt
Программа №1
Команды:
  0 – информация
  1 a dx – производная
  2 x – перевод в систему
  exit – выход
cos'(0) при dx=0.001 = -0.00047684
convert(10) = 1010
convert(-5) = -101
cos'(1.5708) при dx=0.01 = -0.99998242

┌──(flow㉿kali)-[~/Documents/vsc/os/lab4]
└─$
```

Strace:

execve("./prog2", ["./prog2"], 0x7ffeb32a7ad0 /* 55 vars */) = 0

brk(NULL) = 0x560830c89000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f5be9405000

access("/etc/ld.so.preload", R_OK) = -1 ENOENT (Нет такого файла или каталога)

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

fstat(3, {st_mode=S_IFREG|0644, st_size=122447, ...}) = 0

mmap(NULL, 122447, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f5be93e7000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0p\236\2\0\0\0\0"..., 832) = 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 840, 64) = 840

fstat(3, {st_mode=S_IFREG|0755, st_size=2003408, ...}) = 0

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 840, 64) = 840

mmap(NULL, 2055800, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f5be91f1000

**mmap(0x7f5be9219000, 1462272, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f5be9219000**

**mmap(0x7f5be937e000, 352256, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x18d000) = 0x7f5be937e000**

**mmap(0x7f5be93d4000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e2000) = 0x7f5be93d4000**

**mmap(0x7f5be93da000, 52856, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f5be93da000**

close(3) = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f5be91ee000

arch_prctl(ARCH_SET_FS, 0x7f5be91ee740) = 0

set_tid_address(0x7f5be91eea10) = 35658

set_robust_list(0x7f5be91eea20, 24) = 0

rseq(0x7f5be91ee680, 0x20, 0, 0x53053053) = 0

mprotect(0x7f5be93d4000, 16384, PROT_READ) = 0

mprotect(0x56081ecaa000, 4096, PROT_READ) = 0

mprotect(0x7f5be9443000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

munmap(0x7f5be93e7000, 122447) = 0

fstat(1, {st_mode=S_IFCHR|0600, st_rdev=makedev(0x88, 0x1), ...}) = 0

getrandom("\x12\x76\xc2\x5d\xb3\x5c\x24\x81", 8, GRND_NONBLOCK) = 8

brk(NULL) = 0x560830c89000

brk(0x560830caa000) = 0x560830caa000

write(1, "\320\237\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260 \342\204\2262\n", 24) = 24

write(1, "\320\232\320\276\320\274\320\260\320\275\320\264\321\213:\n", 16) = 16

write(1, " 0   \342\200\224 \320\277\320\265\321\200\320\265\320\272\320\273\321"..., 71) = 71

write(1, " 1 a dx   \342\200\224 \320\277\321\200\320\276\320\270\320\267\320\262\320"..., 82) = 82

write(1, " 2 x   \342\200\224 \320\277\320\265\321\200\320\265\320\262\320\276\320"..., 78) = 78

write(1, " exit   \342\200\224 \320\262\321\213\321\205\320\276\320\264\n", 30) = 30

openat(AT_FDCWD, "./libderiv1.so", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0775, st_size=15296, ...}) = 0

getcwd("/home/flow/Documents/vsc/os/lab4", 128) = 33

mmap(NULL, 16408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f5be9400000

mmap(0x7f5be9401000, 4096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7f5be9401000

mmap(0x7f5be9402000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f5be9402000

mmap(0x7f5be9403000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f5be9403000

close(3)   = 0

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

fstat(3, {st_mode=S_IFREG|0644, st_size=122447, ...}) = 0

mmap(NULL, 122447, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f5be91d0000

close(3)   = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=977112, ...}) = 0

mmap(NULL, 978968, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f5be90e0000

mmap(0x7f5be90f1000, 512000, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x11000) = 0x7f5be90f1000

mmap(0x7f5be916e000, 393216, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8e000) = 0x7f5be916e000

mmap(0x7f5be91ce000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xed000) = 0x7f5be91ce000

close(3)   = 0

mprotect(0x7f5be91ce000, 4096, PROT_READ) = 0

mprotect(0x7f5be9403000, 4096, PROT_READ) = 0

munmap(0x7f5be91d0000, 122447)   = 0

openat(AT_FDCWD, "./libconv1.so", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0775, st_size=15360, ...}) = 0

getcwd("/home/flow/Documents/vsc/os/lab4", 128) = 33

mmap(NULL, 16416, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f5be93fb000

mmap(0x7f5be93fc000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,

0x1000) = 0x7f5be93fc000

mmap(0x7f5be93fd000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f5be93fd000

mmap(0x7f5be93fe000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f5be93fe000

close(3) = 0

mprotect(0x7f5be93fe000, 4096, PROT_READ) = 0

write(1, "\320\237\320\265\321\200\320\265\320\272\320\273\321\216\321\207\320\265\320\275\320\276 \320\275\320\260 \321\200\320\265"..., 54) = 54

write(1, " cos_derivative: (f(a+dx)-f(a)"..., 37) = 37

write(1, " convert: \320\264\320\262\320\276\320\270\321\207\320\275\320\260\321\217\n", 29) = 29

fstat(0, {st_mode=S_IFREG|0664, st_size=59, ...}) = 0

read(0, "1 0.0 0.001\n2 10\n0\n1 0.0 0.001\n2"..., 4096) = 59

write(1, "cos'(0) \320\277\321\200\320\270 dx=0.001 = -0.000"..., 66) = 66

write(1, "convert(10) = 1010 (\321\200\320\265\320\260\320\273\320\270\320\267"..., 47) = 47

munmap(0x7f5be9400000, 16408) = 0

munmap(0x7f5be90e0000, 978968) = 0

munmap(0x7f5be93fb000, 16416) = 0

openat(AT_FDCWD, "./libderiv2.so", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0775, st_size=15296, ...}) = 0

getcwd("/home/flow/Documents/vsc/os/lab4", 128) = 33

mmap(NULL, 16408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f5be9400000

mmap(0x7f5be9401000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7f5be9401000

mmap(0x7f5be9402000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f5be9402000

mmap(0x7f5be9403000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f5be9403000

close(3) = 0

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

fstat(3, {st_mode=S_IFREG|0644, st_size=122447, ...}) = 0

mmap(NULL, 122447, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f5be91d0000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=977112, ...}) = 0

mmap(NULL, 978968, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f5be90e0000

mmap(0x7f5be90f1000, 512000, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x11000) = 0x7f5be90f1000

mmap(0x7f5be916e000, 393216, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8e000) = 0x7f5be916e000

```
mmap(0x7f5be91ce000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xed000) = 0x7f5be91ce000
close(3)                                = 0
mprotect(0x7f5be91ce000, 4096, PROT_READ) = 0
mprotect(0x7f5be9403000, 4096, PROT_READ) = 0
munmap(0x7f5be91d0000, 122447)          = 0
openat(AT_FDCWD, "./libconv2.so", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0775, st_size=15360, ...}) = 0
getcwd("/home/flow/Documents/vsc/os/lab4", 128) = 33
mmap(NULL, 16416, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f5be93fb000
mmap(0x7f5be93fc000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7f5be93fc000
mmap(0x7f5be93fd000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f5be93fd000
mmap(0x7f5be93fe000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f5be93fe000
close(3)                                = 0
mprotect(0x7f5be93fe000, 4096, PROT_READ) = 0
write(1, "\320\237\320\265\321\200\320\265\320\272\320\273\321\216\321\207\320\265\320\275\320\276
\320\275\320\260 \321\200\320\265"..., 54) = 54
write(1, "    cos_derivative: (f(a+dx)-f(a-"..., 43) = 43
write(1, "    convert: \321\202\321\200\320\276\320\270\321\207\320\275\320\260\321\217\n", 29) = 29
write(1, "cos'(0) \320\277\321\200\320\270 dx=0.001 = 0.0000"..., 65) = 65
write(1, "convert(10) = 101 (\321\200\320\265\320\260\320\273\320\270\320\267\320"..., 46) = 46
write(1, "convert(-5) = -12 (\321\200\320\265\320\260\320\273\320\270\320\267\320"..., 46) = 46
write(1, "cos'(1.5708) \320\277\321\200\320\270 dx=0.01 = -0"..., 70) = 70
read(0, "", 4096)                       = 0
munmap(0x7f5be9400000, 16408)           = 0
munmap(0x7f5be90e0000, 978968)          = 0
munmap(0x7f5be93fb000, 16416)           = 0
exit_group(0)                           = ?
    +++ exited with 0 +++
```

Вывод

В ходе лабораторной работы для варианта 14 были реализованы две программы: первая использует статическую линковку с фиксированными реализациями производной cos(x) и перевода чисел в двоичную систему, вторая — динамическую загрузку библиотек, позволяющую переключать реализации на лету, что наглядно демонстрирует повышение точности и гибкость подхода.