

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-212БВ-24

Студент: Мышакин М.С.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 29.11.25

Москва, 2025

Постановка задачи

Вариант 13.

• Требуется создать динамические библиотеки, которые реализуют заданный вариантом функционал. Далее использовать данные библиотеки 2-мя способами:

1. Во время компиляции (на этапе линковки/linking)
2. Во время исполнения программы. Библиотеки загружаются в память с помощью интерфейса ОС для работы с динамическими библиотеками

В конечном итоге, в лабораторной работе необходимо получить следующие части:

- Динамические библиотеки, реализующие контракты, которые заданы вариантом;
- Тестовая программа (программа №1), которая используют одну из библиотек, используя информацию полученные на этапе компиляции;
- Тестовая программа (программа №2), которая загружает библиотеки, используя только их относительные пути и контракты.

Провести анализ двух типов использования библиотек. Пользовательский ввод для обоих программ должен быть организован следующим образом:

- Если пользователь вводит команду «0», то программа переключает одну реализацию контрактов на другую (необходимо только для программы №2). Можно реализовать лабораторную работу без данной функции, но максимальная оценка в этом случае будет «хорошо»;
- “1 arg1 arg2 ... argN”, где после “1” идут аргументы для первой функции, предусмотренной контрактами. После ввода команды происходит вызов первой функции, и на экране появляется результат ее выполнения;
- “2 arg1 arg2 ... argM”, где после “2” идут аргументы для второй функции, предусмотренной контрактами. После ввода команды происходит вызов второй функции, и на экране появляется результат ее выполнения.

Задание для 13 варианта

Функция №1:

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: $f'(x) = (f(a + dx) - f(a - dx)) / (2dx)$.

Функция №2:

7. Подсчет площади плоской геометрической фигуры по двум сторонам: Сигнатура функции: `float area(float a, float b);`

- Реализация №1: Фигура прямоугольник
- Реализация №2: Фигура прямоугольный треугольник.

Общий метод и алгоритм решения

Создание контрактов

- Определён общий интерфейс для функций `cos_derivative` и `area`.
- Контракт оформлен в отдельном заголовочном файле `contract.h`.

Создание двух реализаций динамических библиотек

- **impl1.dll** — первая реализация:
 - $\text{cos_derivative}(a, dx) = (\cos(a+dx) - \cos(a)) / dx$
 - $\text{area}(a, b) = a * b$ (прямоугольник)
- **impl2.dll** — вторая реализация:
 - $\text{cos_derivative}(a, dx) = (\cos(a+dx) - \cos(a-dx)) / (2*dx)$
 - $\text{area}(a, b) = a * b / 2$ (прямоугольный треугольник)

Создание тестовых программ

- **Program 1** — статическая линковка с одной библиотекой (использование на этапе компиляции).
- **Program 2** — динамическая загрузка библиотек с возможностью переключения реализации в runtime (`LoadLibrary` / `FreeLibrary`).

Организация пользовательского ввода

- 0 — переключение библиотеки (только для программы №2).
- 1 arg1 arg2 — вызов функции `cos_derivative`.
- 2 arg1 arg2 — вызов функции `area`.

Описание работы программы

Программа №1 (статическая линковка)

- Загружает одну из библиотек при компиляции.
- При вводе команды 1 а dx вычисляет приближённую производную функции $\cos(x)$ в точке а с шагом dx.
- При вводе команды 2 а b вычисляет площадь геометрической фигуры по сторонам а и b.
- Переключение библиотек не поддерживается.

Программа №2 (динамическая загрузка)

- Загружает библиотеку в runtime по относительному пути.
- Поддерживает переключение между реализациями командой 0.
- Функции вызываются через указатели, полученные через GetProcAddress.
- При каждой загрузке DLL создаётся новая копия библиотеки (все статические переменные обнуляются).

Код программы

contracts/contract.h

```
#pragma once  
  
#ifdef _WIN32  
  
#define API __declspec(dllexport)  
  
#endif
```

```
API float cos_derivative(float a, float dx);
```

```
API float area(float a, float b);
```

lib_impl1/impl1.h

```
#pragma once
```

```
float cos_derivative(float a, float dx);
```

```
float area(float a, float b);
```

lib_impl1/impl1.c

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

```
#include "contract.h"
```

```
float cos_derivative(float a, float dx)
```

```
{
```

```
    return (cosf(a + dx) - cosf(a)) / dx;
```

```
}
```

```
float area(float a, float b)
```

```
{
```

```
    return a * b; // прямоугольник
```

```
}
```

lib_impl2/impl2.h

```
#pragma once
```

```
float cos_derivative(float a, float dx);
```

```
float area(float a, float b);
```

lib_impl2/impl2.c

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

```
#include "contract.h"
```

```
float cos_derivative(float a, float dx)
```

```
{
```

```
    return (cosf(a + dx) - cosf(a - dx)) / (2 * dx);
```

```
}
```

```
float area(float a, float b)
```

```
{
```

```
    return (a * b) / 2.0f; // прямоугольный треугольник
```

```
}
```

program1_static/main_static.c

```
#include <stdio.h>
```

```
#include "contract.h"
```

```
int main()
```

```
{
```

```
    int cmd;
```

```
    printf("Commands:\n"
```

```
        "1 a dx -> cos_derivative(a, dx)\n"
```

```
        "2 a b -> area(a, b)\n"
```

```
        "0 -> exit\n");
```

```
    while (scanf("%d", &cmd) == 1)
```

```
    {
```

```
        if (cmd == 0)
```

```
            break;
```

```

        if (cmd == 1)
        {
            float a, dx;
            scanf("%f %f", &a, &dx);
            printf("Result = %f\n", cos_derivative(a, dx));
        }
        else if (cmd == 2)
        {
            float a, b;
            scanf("%f %f", &a, &b);
            printf("Result = %f\n", area(a, b));
        }
    }
}

```

program2_dynamic/main_dynamic.c

```

#include <stdio.h>
#include <windows.h>

typedef float (*fn_derivative)(float, float);
typedef float (*fn_area)(float, float);

HMODULE lib = NULL;
fn_derivative fder = NULL;
fn_area farea = NULL;

void load_library(const char *name)
{

```

```

    if (lib)
        FreeLibrary(lib);

lib = LoadLibraryA(name);

if (!lib)
{
    printf("Error loading %s\n", name);
    exit(1);
}

fder = (fn_derivative)GetProcAddress(lib, "cos_derivative");
farea = (fn_area)GetProcAddress(lib, "area");

if (!fder || !farea)
{
    printf("Invalid library: missing contracts\n");
    exit(1);
}

printf("Loaded: %s\n", name);
}

int main()
{
    int state = 0;
    load_library("libimpl1.dll");

    printf("Commands:\n"

```

```

        "0 -> switch implementation\n"
        "1 a dx -> cos_derivative(a, dx)\n"
        "2 a b -> area(a, b)\n");

int cmd;
while (scanf("%d", &cmd) == 1)
{
    if (cmd == 0)
    {
        state ^= 1;
        load_library(state ? "libimpl2.dll" : "libimpl1.dll");
    }
    else if (cmd == 1)
    {
        float a, dx;
        scanf("%f %f", &a, &dx);
        printf("Result = %f\n", fder(a, dx));
    }
    else if (cmd == 2)
    {
        float a, b;
        scanf("%f %f", &a, &b);
        printf("Result = %f\n", fareal(a, b));
    }
}
}

```

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.10)
```

```
project(DynamicLab C)
```

```
set(CMAKE_C_STANDARD 11)
```

```
include_directories(${CMAKE_SOURCE_DIR}/contracts)
```

```
include_directories(${CMAKE_SOURCE_DIR}/lib_impl1)
```

```
include_directories(${CMAKE_SOURCE_DIR}/lib_impl2)
```

```
add_library(impl1 SHARED
```

```
lib_impl1/impl1.c
```

```
)
```

```
set_target_properties(impl1 PROPERTIES OUTPUT_NAME "impl1")
```

```
add_library(impl2 SHARED
```

```
lib_impl2/impl2.c
```

```
)
```

```
set_target_properties(impl2 PROPERTIES OUTPUT_NAME "impl2")
```

```
add_executable(program1_static program1_static/main_static.c)
```

```
target_link_libraries(program1_static impl1)
```

```
add_executable(program2_dynamic program2_dynamic/main_dynamic.c)
```

Протокол работы программы

```
PS C:\Users\Public\VsCodePrograms\C\OS-LABS\laba4\build> ./program1_static.exe
```

Commands:

```
1 a dx -> cos_derivative(a, dx)
```

```
2 a b -> area(a, b)
```

```
0 -> exit
```

```
2 3 4
```

```
Result = 12.000000
```

```
1 0 0.001
```

```
Result = -0.000477
```

```
0
```

```
S C:\Users\Public\VsCodePrograms\C\OS-LABS\laba4\build> ./program2_dynamic.exe
```

```
Loaded: libimpl1.dll
```

Commands:

```
0 -> switch implementation
```

```
1 a dx -> cos_derivative(a, dx)
```

```
2 a b -> area(a, b)
```

```
1 0 0.001
```

```
Result = -0.000477
```

```
2 3 4
```

```
Result = 12.000000
```

```
0
```

```
Loaded: libimpl2.dll
```

```
1 0 0.001
```

```
Result = 0.000000
```

```
2 3 4
```

```
Result = 6.000000
```

```
1 1 0.01
```

```
2 3 4
```

```
Result = 6.000000
```

```
1 1 0.01
```

```
Result = 6.000000
```

```
1 1 0.01
```

```
1 1 0.01
```

```
Result = -0.841457
```

```
0
```

```
Loaded: libimpl1.dll
```

```
1 1 0.01
```

```
Result = -0.844157
```

```
2 10 10
```

```
Result = 100.000000  
0  
Loaded: libimpl2.dll  
2 10 10  
Result = 50.000000
```

Вывод

В ходе работы были созданы две динамические библиотеки с разными реализациями функций. Созданы две программы: одна использует библиотеку на этапе компиляции, другая — динамически загружает библиотеку в runtime с возможностью переключения. Реализован пользовательский интерфейс для вызова функций библиотек. Проверено корректное вычисление производной функции $\cos(x)$ и площади геометрических фигур. Лабораторная работа выполнена успешно, цель достигнута. Программы корректно используют динамические библиотеки и позволяют сравнивать разные реализации функций.