Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

# Лабораторная работа №4 по курсу

# «Операционные системы»

Студент: Дудовцев Андрей Андреевич
Группа: М8О-208Б-22
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2024

**Содержание**

## Репозиторий

AndreyDdvts/OS_LABS (github.com)

## Цель работы

Приобретение практических навыков в:

- Создании динамических библиотек
- Создании программ, которые используют функции динамических библиотек

## Задание

Требуется создать динамические библиотеки, которые реализуют определенный 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 знания полученные на этапе компиляции;
- Тестовая программа (программа №2), которая загружает библиотеки, используя только их местоположение и контракты.

## Описание работы программы

Функции, написанные в результате выполнения лабораторной работы:

- Рассчет производной функции cos(x) в точке A с приращением deltaX двумя реализациями

- Подсчёт количества простых чисел на отрезке [A, B] (A, B - натуральные) наивным алгоритмом и решетом Эратосфена

В ходе выполнения лабораторной работы я использовала следующие системные вызовы:

- dlopen - открытие динамического объекта
- dlclose - закрытие динамического объекта

## Исходный код

```
========================== realizations.hpp ========================

#pragma once

#include <iostream>
#include <cmath>

#ifdef __cplusplus
extern "C" {
#endif

constexpr int NUM_POINTS = 3000; // число разбиений
const float PI = 3.1415926535;

float SinIntegral(float a, float b, float e);
int PrimeCount(int a, int b);

#ifdef __cplusplus
}
#endif

========================== utils.hpp ==========================

#include <iostream>
#include <cstdlib>
```

```
#include <dlfcn.h>

using SinIntegralFunc = float (*)(float, float, float);
using PrimeCountFunc = int (*)(int, int);

void* LoadLibrary(const char *libraryName);
void UnloadLibrary(void* handle);

======================== realization1.cpp ========================

#include "realizations.hpp"

extern "C" float SinIntegral(float a, float b, float e) {
    float integral = 0.0;
    e = (b - a) / NUM_POINTS;
    for (int i = 0; i <= NUM_POINTS; ++i) {
        integral = integral + e * sin(a + e * (i - 0.5));
    }
    return integral;
}

extern "C" int PrimeCount(int a, int b) {
    int count = 0;
    bool flag = true;
    for (int i = a; i <= b; ++i) {
        if (i <= 1) {
            continue;
        }
        for (int j = 2; j < i; ++j) {
            if (i % j == 0) {
                flag = false;
                break;
            }
        }
        if (flag) {
```

```
            ++count;
        }
        flag = true;
    }
    return count;
}

========================== realization2.cpp ==========================
#include "realizations.hpp"
#include <vector>

extern "C" float SinIntegral(float a, float b, float e) {
    float integral = 0.0;
    e = (b - a) / NUM_POINTS;
    for (int i = 1; i < NUM_POINTS; ++i) {
        float x1 = a + i * e;
        float x2 = a + (i + 1) * e;
        integral += 0.5 * e * (sin(x1) + sin(x2));
    }
    return integral;
}

extern "C" int PrimeCount(int a, int b) {
    int count = 0;
    std::vector<int> numbers;
    numbers.reserve(b + 1);
    for (int i = 0; i <= b; ++i) {
        numbers.push_back(i);
    }
    for (int i = 2; i <= b; ++i) {
        if (numbers[i] != 0) {
            if (numbers[i] >= a && numbers[i] <= b) {
                ++count;
            }
```

```
            for (int j = i * i; j <= b; j += i) {
                numbers[j] = 0;
            }
        }
    }
    return count;
}

========================== utils.cpp ==========================
#include "utils.hpp"

void* LoadLibrary(const char *libraryName) {
    void* handle = dlopen(libraryName, RTLD_LAZY);
    if (!handle) {
        std::cerr << "Couldn't load the library: " << dlerror() <<
std::endl;
        exit(EXIT_FAILURE);
    }
    return handle;
}

void UnloadLibrary(void* handle) {
    if (dlclose(handle) != 0) {
        std::cerr << "Couldn't unload the library: " << dlerror() <<
std::endl;
        exit(EXIT_FAILURE);
    }
}



========================== dynamic_main.cpp ========================

#include "utils.hpp"
```

```
int main() {
    const char *pathToLib1 = getenv("PATH_TO_LIB1");
    const char *pathToLib2 = getenv("PATH_TO_LIB2");
    // bash: export
PATH_TO_LIB1="/home/qwz/OS_LABS/build/lab4/librealization1.so"
    // bash: export
PATH_TO_LIB2="/home/qwz/OS_LABS/build/lab4/librealization2.so"

    void* libraryHandle = LoadLibrary(pathToLib1);
    SinIntegralFunc SinIntegral =
(SinIntegralFunc)dlsym(libraryHandle, "SinIntegral");
    PrimeCountFunc PrimeCount = (PrimeCountFunc)dlsym(libraryHandle,
"PrimeCount");

    std::string command;
    while(true) {
        std::cout << "Enter the command (0 - switch realization, e -
exit): ";
        std::cin >> command;
        if (command == "e") {
            break;
        } else if (command == "0") {
            std::cout << "Enter the library (1 or 2): ";
            std::cin >> command;
            if (command == "1") {
                libraryHandle = LoadLibrary(pathToLib1);
            } else if (command == "2") {
                libraryHandle = LoadLibrary(pathToLib2);
            } else {
                std::cout << "Invalid library" << std::endl;
            }
            SinIntegral = (SinIntegralFunc)dlsym(libraryHandle,
"SinIntegral");
```

```
            PrimeCount = (PrimeCountFunc)dlsym(libraryHandle,
"PrimeCount");
        } else {
            if (command == "1") {
                std::cout << "SinIntegral function:" << std::endl;
                float arg1, arg2, arg3;
                std::cin >> arg1 >> arg2 >> arg3;
                float result = SinIntegral(arg1, arg2, arg3);
                std::cout << "Result of integral = " << result <<
std::endl;
            } else if (command == "2") {
                std::cout << "PrimeCount function:" << std::endl;
                int arg1, arg2;
                std::cin >> arg1 >> arg2;
                int result = PrimeCount(arg1, arg2);
                std::cout << "Count of prime numbers = " << result <<
std::endl;
            } else {
                std::cout << "Invalid command" << std::endl;
            }
        }
    }
    UnloadLibrary(libraryHandle);
    return 0;
}
```

```
========================= static_main.cpp ========================

#include "realizations.hpp"

#include <iostream>

void Task(const std::string& command) {
```

```
    if (command == "1") {
        float arg1, arg2, arg3;
        std::cin >> arg1 >> arg2 >> arg3;
        float result = SinIntegral(arg1, arg2, arg3);
        std::cout << "Result of integral = " << result << std::endl;
    } else if (command == "2") {
        int arg1, arg2;
        std::cin >> arg1 >> arg2;
        int result = PrimeCount(arg1, arg2);
        std::cout << "Count of prime numbers = " << result <<
std::endl;
    } else {
        std::cout << "Invalid command" << std::endl;
    }
}

int main() {
    std::string command;
    while(true) {
        std::cout << "Enter the command (0 - exit): ";
        std::cin >> command;
        if (command == "0") {
            break;
        }
        Task(command);
    }
    return 0;
}
```

## Тесты

```
#include "gtest/gtest.h"
#include "realizations.hpp"
```

```
TEST(fourthLabTest, SinIntegralStaticTest1) {
    float result = SinIntegral(0, PI, 0.01);
    EXPECT_FLOAT_EQ(result, 2);
}

TEST(fourthLabTest, SinIntegralStaticTest2) {
    float result = SinIntegral(0, PI/2, 0.01);
    EXPECT_FLOAT_EQ(result, 1);
}

TEST(fourthLabTest, SinIntegralStaticTest3) {
    float result = SinIntegral(0, PI/3, 0.01);
    EXPECT_FLOAT_EQ(result, 0.49999982);
}

TEST(fourthLabTest, PrimeCountStaticTest) {
    float result = PrimeCount(3, 15);
    EXPECT_FLOAT_EQ(result, 5);
}

int main(int argc, char **argv) {
    testing::InitGoogleTest(&argc, argv);
    return RUN_ALL_TESTS();
}
```

```
#include "gtest/gtest.h"
#include "realizations.hpp"

TEST(fourthLabTest, SinIntegralStaticTest1) {
```

```
    float result = SinIntegral(0, PI, 0.01);
    EXPECT_FLOAT_EQ(result, 2);
}

TEST(fourthLabTest, SinIntegralStaticTest2) {
    float result = SinIntegral(0, PI/2, 0.01);
    EXPECT_FLOAT_EQ(result, 0.9999997);
}

TEST(fourthLabTest, SinIntegralStaticTest3) {
    float result = SinIntegral(0, PI/3, 0.01);
    EXPECT_FLOAT_EQ(result, 0.49999979);
}

TEST(fourthLabTest, PrimeCountStaticTest) {
    float result = PrimeCount(0, 10);
    EXPECT_FLOAT_EQ(result, 4);
}

int main(int argc, char **argv) {
    testing::InitGoogleTest(&argc, argv);
    return RUN_ALL_TESTS();
}
```

**Консоль**

```
Enter the command (0 - exit): 1
0 3.14 0.01
Result of integral = 2
Enter the command (0 - exit): 2
3 15
Count of prime numbers = 5
Enter the command (0 - exit): 0
```

```
qwz@qwz-VirtualBox:~/OS_LABS/build/lab4$ ./dynamic_main
Enter the command (0 - switch realization, e - exit): 0
Enter the library (1 or 2): 1
Enter the command (0 - switch realization, e - exit): 1
SinIntegral function:
0 3.14 0.01
Result of integral = 2
Enter the command (0 - switch realization, e - exit): 2
PrimeCount function:
0 10
Count of prime numbers = 4
Enter the command (0 - switch realization, e - exit): 0
Enter the library (1 or 2): 2
Enter the command (0 - switch realization, e - exit): 1
SinIntegral function:
0 3.14 0.01
Result of integral = 2
Enter the command (0 - switch realization, e - exit): 2
PrimeCount function:
0 10
Count of prime numbers = 4
Enter the command (0 - switch realization, e - exit): e
qwz@qwz-VirtualBox:~/OS_LABS/build/lab4$
```

## Запуск тестов

```
[==========] Running 4 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 4 tests from fourthLabTest
[ RUN      ] fourthLabTest.SinIntegralStaticTest1
[       OK ] fourthLabTest.SinIntegralStaticTest1 (0 ms)
[ RUN      ] fourthLabTest.SinIntegralStaticTest2
[       OK ] fourthLabTest.SinIntegralStaticTest2 (0 ms)
```

```
[ RUN      ] fourthLabTest.SinIntegralStaticTest3
[       OK ] fourthLabTest.SinIntegralStaticTest3 (0 ms)
[ RUN      ] fourthLabTest.PrimeCountStaticTest
[       OK ] fourthLabTest.PrimeCountStaticTest (0 ms)
[----------] 4 tests from fourthLabTest (0 ms total)

[----------] Global test environment tear-down
[==========] 4 tests from 1 test suite ran. (0 ms total)
[  PASSED  ] 4 tests.
qwz@qwz-VirtualBox:~/OS_LABS/build/tests$ ./lab4_test2
[==========] Running 4 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 4 tests from fourthLabTest
[ RUN      ] fourthLabTest.SinIntegralStaticTest1
[       OK ] fourthLabTest.SinIntegralStaticTest1 (0 ms)
[ RUN      ] fourthLabTest.SinIntegralStaticTest2
[       OK ] fourthLabTest.SinIntegralStaticTest2 (0 ms)
[ RUN      ] fourthLabTest.SinIntegralStaticTest3
[       OK ] fourthLabTest.SinIntegralStaticTest3 (0 ms)
[ RUN      ] fourthLabTest.PrimeCountStaticTest
[       OK ] fourthLabTest.PrimeCountStaticTest (0 ms)
[----------] 4 tests from fourthLabTest (0 ms total)

[----------] Global test environment tear-down
[==========] 4 tests from 1 test suite ran. (0 ms total)
[  PASSED  ] 4 tests.
```

## Выводы

В результате выполнения данной лабораторной работы были созданы динамические библиотеки, которые реализуют функционал в соответствие с вариантом задания на C++. Я приобрел практические навыки в создании программ, которые используют функции динамических библиотек.