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: Ветошкина София Владимировна

Группа: М8О-203Б-23

Вариант: 19

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2024

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Метод и алгоритм решения задачи
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

<https://github.com/sofiavetoshkina/os_labs/tree/main>

**Постановка задачи**

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:

1. 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»;
2. «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 её выполнения;
3. «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 её выполнения.

**Общие сведения о программе**

|  |  |  |  |
| --- | --- | --- | --- |
| **Описание** | **Сигнатура** | **Реализация 1** | **Реализация 2** |
| Подсчет площади плоской геометрической фигуры по  двум сторонам | Float Square(float A, float B) | Фигура прямоугольник | Фигура прямоугольный треугольник |
| Подсчёт количества простых чисел на отрезке [A, B] (A, B - натуральные) | Int PrimeCount(int A, int B) | Наивный алгоритм. Проверить делимость текущего числа на все предыдущие числа. | Решето Эратосфена |

Программа собирается системой сборки CMake.

Программа демонстрирует два подхода к работе с библиотеками:

1. Линковка на этапе компиляции. Библиотеки подключаются во время компиляции, что позволяет программе заранее знать об их функционале.

2. Динамическая загрузка во время выполнения. Библиотеки загружаются в память с использованием системного интерфейса для работы с динамическими библиотеками.

Реализованы тесты для проверки корректности программы с помощью Google Test.

**Метод и алгоритм решения задачи**

Для вычисления площади используются базовые геометрические формулы. Для подсчета простых чисел: наивный метод проверяет делимость каждого числа; pешето Эратосфена создает массив чисел и исключает кратные для оптимизации.

Сначала происходит чтение путей к библиотекам через переменные виртуального окружения (PATH\_TO\_LIBRARY1, PATH\_TO\_LIBRARY2), затем загрузка с помощью dlopen. Затем определяем функции через dlsym. Реализуем переключение библиотек (для программы с динамической загрузкой).

**Исходный код**

lab4/main\_dynamic.cpp:

#include <iostream>

#include <dlfcn.h>

#include <cstring>

using SquareFunc = float(\*)(float, float);

using PrimeCountFunc = int(\*)(int, int);

int main() {

std::cout << "Динамическая загрузка библиотек\n";

const char\* pathToLib1 = std::getenv("PATH\_TO\_LIBRARY1");

if (!pathToLib1) {

std::cerr << "Переменная PATH\_TO\_LIBRARY1 не задана" << std::endl;

return 1;

}

const char\* pathToLib2 = std::getenv("PATH\_TO\_LIBRARY2");

if (!pathToLib2) {

std::cerr << "Переменная PATH\_TO\_LIBRARY2 не задана" << std::endl;

return 1;

}

const char\* lib\_paths[] = {pathToLib1, pathToLib2};

int current\_lib = 0;

std::cout << "\nКакую библиотеку вы хотели бы загрузить?\n";

std::cout << "1 - implementation1.so (Площадь прямоугольника, Наивный поиск простых чисел)\n";

std::cout << "2 - implementation2.so (Площадь прямоугольного треугольника, Поиск простых чисел при помощи алгоритма решето Эратосфена)\n";

std::cin >> current\_lib;

if (current\_lib != 1 && current\_lib != 2) {

std::cout << "Неверная команда\n";

return 1;

}

current\_lib--;

void\* handle = dlopen(lib\_paths[current\_lib], RTLD\_LAZY);

if (!handle) {

std::cerr << "Ошибка загрузки библиотеки: " << dlerror() << "\n";

return 1;

}

SquareFunc Square = reinterpret\_cast<SquareFunc>(dlsym(handle, "Square"));

PrimeCountFunc PrimeCount = reinterpret\_cast<PrimeCountFunc>(dlsym(handle, "PrimeCount"));

char\* error;

if ((error = dlerror()) != nullptr) {

std::cerr << "Ошибка: " << error << "\n";

dlclose(handle);

return 1;

}

while (true) {

std::cout << "\n---------------------------------------------------------------\n";

std::cout << "Библиотека: " << lib\_paths[current\_lib] << "\n";

std::cout << "0 - Следующая библиотека\n";

std::cout << "1 A B - Вычислить площадь (прямоугольник или треугольник)\n";

std::cout << "2 A B - Вычислить количество простых чисел в диапазоне [A,B]\n";

std::cout << "3 - Выход";

std::cout << "\n---------------------------------------------------------------\n";

int command;

std::cin >> command;

if (command == 0) {

dlclose(handle);

current\_lib = 1 - current\_lib;

handle = dlopen(lib\_paths[current\_lib], RTLD\_LAZY);

if (!handle) {

std::cerr << "Ошибка загрузки библиотеки: " << dlerror() << "\n";

return 1;

}

Square = reinterpret\_cast<SquareFunc>(dlsym(handle, "Square"));

PrimeCount = reinterpret\_cast<PrimeCountFunc>(dlsym(handle, "PrimeCount"));

} else if (command == 1) {

float A, B;

std::cin >> A >> B;

float result = Square(A, B);

std::cout << "Площадь: " << result << "\n";

} else if (command == 2) {

int A, B;

std::cin >> A >> B;

int result = PrimeCount(A, B);

std::cout << "Количество простых чисел в диапазоне [A,B]: " << result << "\n";

} else if (command == 3) {

break;

} else {

std::cout << "Такого пункта меню не существует\n";

}

}

dlclose(handle);

return 0;

}

lab4/main\_linked.cpp:

#include <iostream>

#include "functions.h"

int main() {

std::cout << "Статическая линковка\n";

while (true) {

std::cout << "\n---------------------------------------------------------------\n";

std::cout << "1 A B - Вычислить площадь (прямоугольник или треугольник)\n";

std::cout << "2 A B - Вычислить количество простых чисел в диапазоне [A,B]\n";

std::cout << "3 - Выход";

std::cout << "\n---------------------------------------------------------------\n";

int command;

std::cin >> command;

if (command == 1) {

float A, B;

std::cin >> A >> B;

float result = Square(A, B);

std::cout << "Площадь: " << result << "\n";

} else if (command == 2) {

int A, B;

std::cin >> A >> B;

int result = PrimeCount(A, B);

std::cout << "Количество простых чисел в диапазоне [A,B]: " << result << "\n";

} else if (command == 3) {

break;

} else {

std::cout << "Такого пункта меню не существует\n";

}

}

return 0;

}

lab4/include/functions.h:

#pragma once

#ifdef \_\_cplusplus

extern "C" {

#endif

float Square(float A, float B);

int PrimeCount(int A, int B);

#ifdef \_\_cplusplus

}

#endif

lab4/src/implementation1.cpp:

#include <cmath>

#include "functions.h"

extern "C" {

float Square(float A, float B) {

return A \* B;

}

int PrimeCount(int A, int B) {

int count = 0;

for (int num = A; num <= B; ++num) {

if (num < 2) continue;

bool is\_prime = true;

for (int div = 2; div <= std::sqrt(num); ++div) {

if (num % div == 0) {

is\_prime = false;

break;

}

}

if (is\_prime) ++count;

}

return count;

}

}

lab4/src/implementation2.cpp:

#include <cmath>

#include <vector>

#include "functions.h"

extern "C" {

float Square(float A, float B) {

return 0.5f \* A \* B;

}

int PrimeCount(int A, int B) {

if (B < 2) return 0;

std::vector<bool> is\_prime(B + 1, true);

is\_prime[0] = is\_prime[1] = false;

for (int p = 2; p <= std::sqrt(B); ++p) {

if (is\_prime[p]) {

for (int multiple = p \* p; multiple <= B; multiple += p) {

is\_prime[multiple] = false;

}

}

}

int count = 0;

for (int i = A; i <= B; ++i) {

if (is\_prime[i]) ++count;

}

return count;

}

}

tests/lab4\_1\_test.cpp:

#include <gtest/gtest.h>

#include <functions.h>

TEST(Library1, TestSquare1) {

float result = Square(1.5, 3.0);

EXPECT\_FLOAT\_EQ(result, 4.5);

}

TEST(Library1, TestSquare2) {

float result = Square(4.0, 3.0);

EXPECT\_FLOAT\_EQ(result, 12.0);

}

TEST(Library1, TestPrimeCount1) {

int result = PrimeCount(1, 5);

EXPECT\_EQ(result, 3);

}

TEST(Library1, TestPrimeCount2) {

int result = PrimeCount(2, 11);

EXPECT\_EQ(result, 5);

}

int main(int argc, char \*\*argv) {

testing::InitGoogleTest(&argc, argv);

return RUN\_ALL\_TESTS();

}

tests/lab4\_2\_test.cpp:

#include <gtest/gtest.h>

#include <functions.h>

TEST(Library2, TestSquare1) {

float result = Square(1.5, 3.0);

EXPECT\_FLOAT\_EQ(result, 2.25);

}

TEST(Library2, TestSquare2) {

float result = Square(4.0, 3.0);

EXPECT\_FLOAT\_EQ(result, 6.0);

}

TEST(Library2, TestPrimeCount1) {

int result = PrimeCount(1, 5);

EXPECT\_EQ(result, 3);

}

TEST(Library2, TestPrimeCount2) {

int result = PrimeCount(2, 11);

EXPECT\_EQ(result, 5);

}

int main(int argc, char \*\*argv) {

testing::InitGoogleTest(&argc, argv);

return RUN\_ALL\_TESTS();

}

lab4/CmakeLists.txt:

set(CMAKE\_CXX\_STANDARD 17)

add\_library(implementation1 SHARED src/implementation1.cpp)

set\_target\_properties(implementation1 PROPERTIES OUTPUT\_NAME "implementation1" PREFIX "")

target\_include\_directories(implementation1 PRIVATE include)

add\_library(implementation2 SHARED src/implementation2.cpp)

set\_target\_properties(implementation2 PROPERTIES OUTPUT\_NAME "implementation2" PREFIX "")

target\_include\_directories(implementation2 PRIVATE include)

add\_executable(main\_linked\_imp1 main\_linked.cpp)

target\_include\_directories(main\_linked\_imp1 PRIVATE include)

target\_link\_libraries(main\_linked\_imp1 PRIVATE implementation1 m)

add\_executable(main\_linked\_imp2 main\_linked.cpp)

target\_include\_directories(main\_linked\_imp2 PRIVATE include)

target\_link\_libraries(main\_linked\_imp2 PRIVATE implementation2 m)

add\_executable(main\_dynamic main\_dynamic.cpp)

target\_include\_directories(main\_dynamic PRIVATE include)

target\_link\_libraries(main\_dynamic PRIVATE dl m)

**Демонстрация работы программы**

getz66@getz1165-nettop:~/OS/os\_labs/build$ export PATH\_TO\_LIBRARY2=$(pwd)/lab4/implementation2.so

getz66@getz1165-nettop:~/OS/os\_labs/build$ export PATH\_TO\_LIBRARY1=$(pwd)/lab4/implementation1.so

getz66@getz1165-nettop:~/OS/os\_labs/build$ ./lab4/main\_dynamic

Динамическая загрузка библиотек

Какую библиотеку вы хотели бы загрузить?

1 - implementation1.so (Площадь прямоугольника, Наивный поиск простых чисел)

2 - implementation2.so (Площадь прямоугольного треугольника, Поиск простых чисел при помощи алгоритма решето Эратосфена)

1

---------------------------------------------------------------

Библиотека: /home/getz66/OS/os\_labs/build/lab4/implementation1.so

0 - Следующая библиотека

1 A B - Вычислить площадь (прямоугольник или треугольник)

2 A B - Вычислить количество простых чисел в диапазоне [A,B]

3 - Выход

---------------------------------------------------------------

1 5.0 10.0

Площадь: 50

---------------------------------------------------------------

Библиотека: /home/getz66/OS/os\_labs/build/lab4/implementation1.so

0 - Следующая библиотека

1 A B - Вычислить площадь (прямоугольник или треугольник)

2 A B - Вычислить количество простых чисел в диапазоне [A,B]

3 - Выход

---------------------------------------------------------------

2 2 3

Количество простых чисел в диапазоне [A,B]: 2

---------------------------------------------------------------

Библиотека: /home/getz66/OS/os\_labs/build/lab4/implementation1.so

0 - Следующая библиотека

1 A B - Вычислить площадь (прямоугольник или треугольник)

2 A B - Вычислить количество простых чисел в диапазоне [A,B]

3 - Выход

---------------------------------------------------------------

0

---------------------------------------------------------------

Библиотека: /home/getz66/OS/os\_labs/build/lab4/implementation2.so

0 - Следующая библиотека

1 A B - Вычислить площадь (прямоугольник или треугольник)

2 A B - Вычислить количество простых чисел в диапазоне [A,B]

3 - Выход

---------------------------------------------------------------

1 5.0 10.0

Площадь: 25

---------------------------------------------------------------

Библиотека: /home/getz66/OS/os\_labs/build/lab4/implementation2.so

0 - Следующая библиотека

1 A B - Вычислить площадь (прямоугольник или треугольник)

2 A B - Вычислить количество простых чисел в диапазоне [A,B]

3 - Выход

---------------------------------------------------------------

2 2 5

Количество простых чисел в диапазоне [A,B]: 3

---------------------------------------------------------------

Библиотека: /home/getz66/OS/os\_labs/build/lab4/implementation2.so

0 - Следующая библиотека

1 A B - Вычислить площадь (прямоугольник или треугольник)

2 A B - Вычислить количество простых чисел в диапазоне [A,B]

3 - Выход

---------------------------------------------------------------

3

getz66@getz1165-nettop:~/OS/os\_labs/build$ ./lab4/main\_linked\_imp1

Статическая линковка

---------------------------------------------------------------

1 A B - Вычислить площадь (прямоугольник или треугольник)

2 A B - Вычислить количество простых чисел в диапазоне [A,B]

3 - Выход

---------------------------------------------------------------

1 5.0 10.0

Площадь: 50

---------------------------------------------------------------

1 A B - Вычислить площадь (прямоугольник или треугольник)

2 A B - Вычислить количество простых чисел в диапазоне [A,B]

3 - Выход

---------------------------------------------------------------

2 2 5

Количество простых чисел в диапазоне [A,B]: 3

---------------------------------------------------------------

1 A B - Вычислить площадь (прямоугольник или треугольник)

2 A B - Вычислить количество простых чисел в диапазоне [A,B]

3 - Выход

---------------------------------------------------------------

3

getz66@getz1165-nettop:~/OS/os\_labs/build$ ./lab4/main\_linked\_imp2

Статическая линковка

---------------------------------------------------------------

1 A B - Вычислить площадь (прямоугольник или треугольник)

2 A B - Вычислить количество простых чисел в диапазоне [A,B]

3 - Выход

---------------------------------------------------------------

1 5.0 10.0

Площадь: 25

---------------------------------------------------------------

1 A B - Вычислить площадь (прямоугольник или треугольник)

2 A B - Вычислить количество простых чисел в диапазоне [A,B]

3 - Выход

---------------------------------------------------------------

2 2 5

Количество простых чисел в диапазоне [A,B]: 3

---------------------------------------------------------------

1 A B - Вычислить площадь (прямоугольник или треугольник)

2 A B - Вычислить количество простых чисел в диапазоне [A,B]

3 - Выход

---------------------------------------------------------------

3

**Выводы**

В процессе выполнения лабораторной работы было изучено: cоздание и использование динамических библиотек; преимущества динамической загрузки (уменьшение размера исполняемого файла, упрощение обновления функциональности); различия между подходами линковки во время компиляции и выполнения.

Динамические библиотеки позволяют эффективно переиспользовать код и упрощают поддержку приложений. Однако они требуют дополнительного контроля при загрузке и управлении зависимостями.