МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ

ФЕДЕРАЦИИ

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»

Студент: Кириллова Е.К.

Группа: М8О-208Б-23

Вариант: 25

Преподаватель: Миронов Е. С.

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2024

**Содержание**

1. Репозиторий

2. Постановка задачи

3. Общие сведения о программе

4. Общий метод и алгоритм решения

5. Исходный код

6. Выводы

**Репозиторий**

https://github.com/ElenaKirillova05/osLabs/tree/main

**Постановка задачи**

**Цель работы**

Целью является приобретение практических навыков в:

Создание динамических библиотек

Создание программ, которые используют функции динамических библиотек

**Задание**

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.

Контракты и реализации функций(**мой вариант**):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | Подсчёт наибольшего общего делителя для двух натуральных чисел  Int GCF(int A, int B) | Int GCF(int A, int B) | Алгоритм Евклида | Наивный алгоритм. Пытаться разделить числа на все числа, что меньше A и B. |
| 8 | Перевод числа x из десятичной системы счисления в другую | Char\* translation(long x) | Другая система счисления двоичная | Другая система счисления троичная |

**Общие сведения о программе**

Код реализует вычисление НОД (наибольшего общего делителя) двумя методами (Евклида и наивным) и перевод числа в разные системы счисления (двоичную и троичную) с использованием динамической загрузки библиотек. Есть две статические библиотеки: одна содержит алгоритмы Евклида и перевода в двоичную систему, другая – наивный алгоритм НОД и перевод в троичную систему. Две программы используют эти библиотеки: первая (program1.c) статически подключает библиотеку с Евклидовым алгоритмом, вторая (program2.c) динамически подгружает библиотеки и переключается между ними в процессе работы. Тесты на GTest проверяют корректность вычислений для обеих реализаций.

**Общий метод и алгоритм решения**

Метод решения заключается в разделении функционала на две библиотеки с одинаковыми именами функций, что позволяет легко переключаться между реализациями с помощью динамической загрузки (dlopen, dlsym). program2.c загружает библиотеку по умолчанию, а затем может переключаться на другую по запросу пользователя. Алгоритм НОД реализован как Евклидов (цикл с остатком) или наивный (перебор делителей), а перевод чисел в строку выполняется путем деления на основание системы счисления и сохранения остатков. Тесты проверяют оба алгоритма и оба варианта перевода.

**Исходный код**

gcd\_euclid\_and\_binary.h:

```

#ifndef GCD\_EUCLID\_AND\_BINARY\_H

#define GCD\_EUCLID\_AND\_BINARY\_H

#ifdef \_\_cplusplus

extern "C" {

#endif

int GCD(int A, int B);

char \*translation(long x);

#ifdef \_\_cplusplus

}

#endif

#endif

```

gcd\_naive\_and\_ternary.h:

```

#ifndef GCD\_NAIVE\_AND\_TERNARY\_H

#define GCD\_NAIVE\_AND\_TERNARY\_H

#ifdef \_\_cplusplus

extern "C" {

#endif

int GCD(int A, int B);

char \*translation(long x);

#ifdef \_\_cplusplus

}

#endif

#endif

```

gcd\_euclid\_and\_binary.cpp:

```

#include <stdio.h>

#include <stdlib.h>

int GCD(int A, int B)

{

while (B != 0)

{

int temp = B;

B = A % B;

A = temp;

}

return A;

}

char \*translation(long x)

{

static char result[64];

int i = 63;

result[i--] = '\0';

do

{

result[i--] = (x % 2) + '0';

x /= 2;

} while (x > 0);

return &result[i + 1];

}

```

gcd\_naive\_and\_ternary.cpp:

```

#include <stdio.h>

#include <stdlib.h>

int GCD(int A, int B)

{

int gcd = 1;

for (int i = 1; i <= A && i <= B; i++)

{

if (A % i == 0 && B % i == 0)

gcd = i;

}

return gcd;

}

char \*translation(long x)

{

static char result[64];

int i = 63;

result[i--] = '\0';

do

{

result[i--] = (x % 3) + '0';

x /= 3;

} while (x > 0);

return &result[i + 1];

}

```

program1.c:

```

#include <stdio.h>

#include "gcd\_euclid\_and\_binary.h"

int main()

{

int command;

printf("Введите команду: ");

while (scanf("%d", &command) != EOF)

{

if (command == 1)

{

int a, b;

printf("Введите два числа: ");

scanf("%d %d", &a, &b);

printf("НОД: %d\n", GCD(a, b));

}

else if (command == 2)

{

long x;

printf("Введите число: ");

scanf("%ld", &x);

printf("Результат перевода: %s\n", translation(x));

}

else

{

printf("Неверная команда.\n");

}

printf("Введите команду: ");

}

return 0;

}

```

program2.c:

```

#include <stdio.h>

#include <dlfcn.h>

#include <stdlib.h>

#include <string.h>

void \*lib\_handle = NULL;

char \*(\*translation)(long) = NULL;

int (\*GCD)(int, int) = NULL;

char current\_system[20] = "";

void load\_library(const char \*lib\_path)

{

if (lib\_handle != NULL)

{

dlclose(lib\_handle);

}

lib\_handle = dlopen(lib\_path, RTLD\_LAZY);

if (!lib\_handle)

{

fprintf(stderr, "Ошибка загрузки библиотеки: %s\n", dlerror());

exit(1);

}

GCD = dlsym(lib\_handle, "GCD");

translation = dlsym(lib\_handle, "translation");

char \*error = dlerror();

if (error != NULL)

{

fprintf(stderr, "Ошибка загрузки функций: %s\n", error);

dlclose(lib\_handle);

exit(1);

}

if (strstr(lib\_path, "binary") != NULL)

{

strcpy(current\_system, "двоичной");

}

else if (strstr(lib\_path, "ternary") != NULL)

{

strcpy(current\_system, "троичной");

}

}

int main()

{

load\_library("./libgcd\_euclid\_and\_binary.so");

char command[100];

int running = 1;

while (running)

{

printf("\nВведите команду (1 для НОД, 2 для перевода, 0 для смены библиотеки, q для выхода): ");

fgets(command, sizeof(command), stdin);

if (command[0] == '1')

{

int a, b;

printf("Введите два числа для НОД: ");

scanf("%d %d", &a, &b);

getchar();

printf("НОД: %d\n", GCD(a, b));

}

else if (command[0] == '2')

{

long x;

printf("Введите число для перевода: ");

scanf("%ld", &x);

getchar();

printf("Число %ld в %s системе: %s\n", x, current\_system, translation(x));

}

else if (command[0] == '0')

{

printf("Введите путь к новой библиотеке: ");

char lib\_path[256];

fgets(lib\_path, sizeof(lib\_path), stdin);

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

load\_library(lib\_path);

printf("Библиотека переключена.\n", current\_system);

}

else if (command[0] == 'q' || command[0] == 'Q')

{

printf("Завершение программы.\n");

running = 0;

}

else

{

printf("Неверная команда. Попробуйте снова.\n");

}

}

dlclose(lib\_handle);

return 0;

}

```

ТЕСТЫ

test.cpp:

```

#include <gtest/gtest.h>

#include "include/gcd\_euclid\_and\_binary.h"

#include "include/gcd\_naive\_and\_ternary.h"

#include <dlfcn.h>

typedef int (\*GCD\_Function)(int, int);

typedef char \*(\*Translation\_Function)(long);

class GCDTest : public ::testing::Test {

protected:

void \*lib\_handle;

GCD\_Function GCD;

Translation\_Function translation;

void LoadLibrary(const char \*lib\_path) {

lib\_handle = dlopen(lib\_path, RTLD\_LAZY);

ASSERT\_NE(lib\_handle, nullptr) << "Ошибка загрузки библиотеки: " << dlerror();

GCD = (GCD\_Function)dlsym(lib\_handle, "GCD");

translation = (Translation\_Function)dlsym(lib\_handle, "translation");

ASSERT\_NE(GCD, nullptr) << "Ошибка загрузки функции GCD: " << dlerror();

ASSERT\_NE(translation, nullptr) << "Ошибка загрузки функции translation: " << dlerror();

}

void UnloadLibrary() {

if (lib\_handle) dlclose(lib\_handle);

}

void SetUp() override {

LoadLibrary("./libgcd\_euclid\_and\_binary.so");

}

void TearDown() override {

UnloadLibrary();

}

};

TEST\_F(GCDTest, GCD\_Euclid\_BasicCases) {

EXPECT\_EQ(GCD(48, 18), 6);

EXPECT\_EQ(GCD(56, 98), 14);

EXPECT\_EQ(GCD(101, 103), 1);

EXPECT\_EQ(GCD(36, 60), 12);

}

TEST\_F(GCDTest, BinaryTranslation\_BasicCases) {

EXPECT\_STREQ(translation(10), "1010");

EXPECT\_STREQ(translation(255), "11111111");

EXPECT\_STREQ(translation(1), "1");

}

TEST\_F(GCDTest, SwitchToNaiveAndTest) {

UnloadLibrary();

LoadLibrary("./libgcd\_naive\_and\_ternary.so");

EXPECT\_EQ(GCD(48, 18), 6);

EXPECT\_EQ(GCD(56, 98), 14);

EXPECT\_EQ(GCD(101, 103), 1);

EXPECT\_EQ(GCD(36, 60), 12);

EXPECT\_STREQ(translation(10), "101");

EXPECT\_STREQ(translation(255), "100110");

EXPECT\_STREQ(translation(1), "1");

}

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

::testing::InitGoogleTest(&argc, argv);

return RUN\_ALL\_TESTS();

}

```

**Выводы**

В процессе работы с программой были изучены ключевые концепции программирования на языке C++ и работы с динамическими библиотеками. Я научилась использовать функции **dlopen**, **dlsym** и **dlclose** для динамической загрузки и работы с библиотеками, что позволяет гибко выбирать реализации функций во время выполнения. Этот опыт помог мне лучше понять принципы работы с динамическими библиотеками.