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.