Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №4 по курсу**

**«Операционные системы»**

**ДИНАМИЧЕСКИЕ БИБЛИОТЕКИ**

Студент: Корнев Максим Сергеевич

Группа: М8О–212Б–22

Вариант: 26

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2023.

**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 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. 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 её 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 её выполнения.

**Функция 1:**

Описание: Подсчёт наибольшего общего делителя для двух натуральных чисел

Сигнатура: Int GCF(int A, int B)

Реализация 1: Алгоритм Евклида

Реализация 2: Наивный алгоритм. Пытаться разделить числа на все числа, что меньше A и B.

**Функция 2:**

Описание: Отсортировать целочисленный массив

Сигнатура: int \*sort(int\* array)

Реализация 1: пузырьковая сортировка

Реализация 2: сортировка Хоара

**Общие сведения о программе**

*func.hpp* – Заголовочный файл для функций;

*real1.cpp* – Содержит реализацию 1, данных функций;

*real2.cpp* – Содержит реализацию 2, данных функций;

*main1.cpp* - Тестовая программа (программа №1), которая используют одну из библиотек, используя знания полученные на этапе компиляции;

*main2.cpp* - Тестовая программа (программа №2), которая загружает библиотеки, используя только их местоположение и контракты;

CMakeLists.txt– Компиляция программ main\_1.c и main\_2.c;

**Общий метод и алгоритм решения**.

Для реализации, поставленной задачи необходимо:

1. Реализовать заданные функции двумя способами, в соответствии с заданием
2. Написать тестовую программу №1 использовать данные библиотеки во время компиляции
3. Написать тестовую программу №2 использовать данные библиотеки Во время исполнения программы.
4. Скомпилировать и запустить программу.

**Основные файлы программы**

**func.hpp**

#pragma once

extern "C" {

int gcf(int A, int B);

int \*sort(int \*array, int size);

}

**real1.cpp**

#include "func.hpp"

#include <iostream>

extern "C" {

int gcf(int A, int B) {

    int result;

    if(B != 0) {

        result = gcf(B, A % B);

    } else {

        result = A;

    }

    return result;

}

int \*sort(int \*array, int size) {

    bool swapped;

    do {

        swapped = false;

        for (int i = 0; i < size - 1; ++i) {

            if (array[i] > array[i + 1]) {

                int tmp = array[i];

                array[i] = array[i + 1];

                array[i + 1] = tmp;

                swapped = true;

            }

        }

    } while (swapped);

    return array;

}

}

**real2.cpp**

#include "func.hpp"

extern "C" {

int gcf(int A, int B) {

    int res = 1;

    int k;

    if( A <= B) {

        k = A;

    } else {

        k = B;

    }

    for(int i = 2; i < k; ++i) {

        if(A % i == 0 and B % i == 0) {

            return i;

        }

    }

}

void swap(int &a, int &b) {

    int temp = a;

    a = b;

    b = temp;

}

int partition(int \*array, int low, int high) {

    int pivot = array[low];

    int left = low + 1;

    int right = high;

    while (true) {

        while (left <= right && array[left] < pivot) {

            ++left;

        }

        while (right >= left && array[right] > pivot) {

            --right;

        }

        if (left <= right) {

            swap(array[left], array[right]);

        } else {

            break;

        }

    }

    swap(array[low], array[right]);

    return right;

}

void quicksort(int \*array, int low, int high) {

    if (low < high) {

        int pivotIndex = partition(array, low, high);

        quicksort(array, low, pivotIndex - 1);

        quicksort(array, pivotIndex + 1, high);

    }

}

int \*sort(int \*array, int size) {

    quicksort(array, 0, size - 1);

    return array;

}

}

**main1.cpp**

#include <iostream>

#include "func.hpp"

int main(int argc, char const \*argv[])

{

    std::cout << "1 arg1 arg2 - GCF function\n2 - Sort function\n3 exit" << std::endl;

    int com;

    int a, b;

    int size;

    int \*array = nullptr;

    std::cout << "Input the command: ";

    std::cin >> com;

    while(com != 3) {

        switch (com)

        {

        case 1:

            std::cout << "Input a and b: ";

            std::cin >> a >> b;

            std::cout << "GCF(" << a << ',' << b << ") = " << gcf(a, b) << std::endl;

            std::cout << "Input the command: ";

            std::cin >> com;

            break;

        case 2:

            std::cout << "Input the size of your array: ";

            std::cin >> size;

            array = new int[size];

            std::cout << "Input your array: ";

            for(int i = 0; i < size; ++i) {

                std::cin >> array[i];

            }

            sort(array, size);

            for(int i = 0; i < size; ++i) {

                std::cout << array[i] << std::endl;

            }

            delete[] array;

            std::cout << "Input the command: ";

            std::cin >> com;

            break;

        case 3:

            break;

        default:

            std::cout << "Incorreact input\n";

            exit(1);

            break;

        }

    }

    return 0;

}

**main2.cpp**

#include <iostream>

#include <dlfcn.h>

#include "func.hpp"

#include <string>

int main(int argc, char const \*argv[])

{

    std::cout << "0 - change realisation\n1 arg1 arg2 - GCF function\n2 your array - Sort function\n3 exit\n" << std::endl;

    int com;

    int a, b;

    int size;

    int \*array = nullptr;

    int lib = 1;

    void \*library;

    char name1[] = "gcf";

    char name2[] = "sort";

    std::cout << "Input your command: ";

    std::cin >> com;

    library = dlopen("libreal1.so", RTLD\_NOW);

    if(!library) {

        std::cerr << "dlopen error: " << dlerror() << std::endl;

        exit(1);

    }

    while(com != 3) {

        switch (com) {

        case 0:

            dlclose(library);

            if(lib == 1) {

                library = dlopen("libreal1.so", RTLD\_NOW);

                lib = 2;

            } else {

                library = dlopen("libreal2.so", RTLD\_NOW);

                lib = 1;

            }

            std::cout << "Realisation was changed.\n";

            std::cout << "Input your command: ";

            std::cin >> com;

            break;

        case 1:

            std::cout << "input a and b: ";

            std::cin >> a >> b;

            int (\*func\_gcf)(int, int); // указатель на GCF

            func\_gcf = reinterpret\_cast<int (\*)(int, int)>(dlsym(library, name1));

            if(!func\_gcf) {

                std::cerr << "GCF error:" << dlerror() << "\n";

                exit(1);

            }

            if(lib == 1) {

                std::cout << "GCF realisation 1\n";

            } else {

                std::cout << "GCF realisation 2\n";

            }

            std::cout << (\*func\_gcf)(a, b) << std::endl;

            std::cout << "Input your command: ";

            std::cin >> com;

            break;

        case 2:

            std::cout << "Input the size of your array: ";

            std::cin >> size;

            array = new int[size];

            std::cout << "Input your array: ";

            for(int i = 0; i < size; ++i) {

                std::cin >> array[i];

            }

            void (\*func\_sort)(int\*, int);

            func\_sort = reinterpret\_cast<void (\*)(int\*, int)>(dlsym(library, name2));

            if (!func\_sort) {

                std::cerr << "sort function error: " << dlerror() << "\n";

                exit(1);

            }

            if(lib == 1) {

                std::cout << "GCF realisation 1\n";

            } else {

                std::cout << "GCF realisation 2\n";

            }

            (\*func\_sort)(array, size);

            std::cout << "Sorted array: ";

            for (int i = 0; i < size; ++i) {

                std::cout << array[i] << " ";

            }

            std::cout << std::endl;

            delete[] array;

            std::cout << "Input your command: ";

            std::cin >> com;

            break;

        case 3:

            break;

        default:

            std::cout << "Incorreact input\n";

            exit(1);

            break;

        }

    }

    dlclose(library);

}

**CMakeLists.txt**

cmake\_minimum\_required(VERSION 3.10)

project(OS\_lab4)

set(CMAKE\_CXX\_STANDARD 20)

add\_library(real1 SHARED real1.cpp)

add\_library(real2 SHARED real2.cpp)

add\_executable(first main1.cpp)

target\_link\_libraries(first PRIVATE real1)

add\_executable(second main2.cpp)

target\_link\_libraries(second PRIVATE real2)

**Пример работы**

**radioactive@DESKTOP-RNP2IGB:/mnt/d/labs/os\_lab\_4$ ./first**

**1 arg1 arg2 - GCF function**

**2 - Sort function**

**3 exit**

**Input the command: 1**

**Input a and b: 15 3**

**GCF(15,3) = 3**

**Input the command: 3**

**radioactive@DESKTOP-RNP2IGB:/mnt/d/labs/os\_lab\_4$ ./second**

**0 - change realisation**

**1 arg1 arg2 - GCF function**

**2 your array - Sort function**

**3 exit**

**Input your command: 0**

**Realisation was changed.**

**Input your command: 1**

**input a and b: 15 3**

**GCF realisation 2**

**3**

**Input your command: 0**

**Realisation was changed.**

**Input your command: 1**

**input a and b: 15 3**

**GCF realisation 1**

**3**

**Input your command: 2**

**Input the size of your array: 4**

**Input your array: 4 3 2 1**

**GCF realisation 1**

**Sorted array: 1 2 3 4**

**Input your command: 0**

**Realisation was changed.**

**Input your command: 2**

**Input the size of your array: 4**

**Input your array: 4 3 2 1**

**GCF realisation 2**

**Sorted array: 1 2 3 4**

**Input your command: 3**

**radioactive@DESKTOP-RNP2IGB:/mnt/d/labs/os\_lab\_4$ ./second**

**0 - change realisation**

**1 arg1 arg2 - GCF function**

**2 your array - Sort function**

**3 exit**

**Input your command: 1**

**input a and b: 192 53**

**GCF realisation 1**

**1**

**Input your command: 1**

**input a and b: 500 50**

**GCF realisation 1**

**50**

**Input your command: 0**

**Realisation was changed.**

**Input your command: 1**

**input a and b: 600 92**

**GCF realisation 2**

**4**

**Input your command: 2**

**Input the size of your array: 5**

**Input your array: 90 500 1 4 5**

**GCF realisation 2**

**Sorted array: 1 4 5 90 500**

**Input your command: 0**

**Realisation was changed.**

**Input your command: 2**

**Input the size of your array: 10**

**Input your array: 10 9 8 7 6 5 4 3 2 1**

**GCF realisation 1**

**Sorted array: 1 2 3 4 5 6 7 8 9 10**

**Input your command: 3**

**radioactive@DESKTOP-RNP2IGB:/mnt/d/labs/os\_lab\_4$ ./first**

**1 arg1 arg2 - GCF function**

**2 - Sort function**

**3 exit**

**Input the command: 1**

**Input a and b: 500 10**

**GCF(500,10) = 10**

**Input the command: 2**

**Input the size of your array: 6**

**Input your array: 4 2 1 3 5 6**

**1**

**2**

**3**

**4**

**5**

**6**

**Input the command: 3**

**radioactive@DESKTOP-RNP2IGB:/mnt/d/labs/os\_lab\_4$**

**Вывод**

В результате выполнения лабораторной работы мной были приобретены практические навыки в создании динамических библиотек и программ, которые используют функции этих библиотек.

В рамках задания было создано несколько динамических библиотек, реализующих определенный функционал. После этого были использованы данные библиотеки двумя способами: на этапе компиляции, при линковке программы, и во время исполнения, загружая библиотеки в память с помощью интерфейса ОС.

В результате лабораторной работы мной были получены следующие основные части: динамические библиотеки, реализующие заданные контракты, тестовая программа №1, которая использует одну из библиотек на этапе компиляции, и тестовая программа №2, которая загружает библиотеки только по их местоположению.

Использование динамической библиотеки во время компиляции означает, что код библиотеки подключается к программе во время ее компиляции. Это позволяет компилятору проверить правильность использования функций и переменных из библиотеки и включить необходимые зависимости в исполняемый файл.

С другой стороны, использование динамической библиотеки во время исполнения программы означает, что код библиотеки загружается и подключается к программе только во время ее выполнения. Это позволяет сократить размер исходного кода программы и обеспечить более гибкую работу с библиотеками, поскольку можно обновлять их без перекомпиляции программы.

Таким образом, выбор между использованием библиотек на этапе компиляции или на этапе исполнения зависит от требований конкретного проекта и наличия нужного функционала в библиотеках.