Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №5 по курсу**

**«Операционные системы»**

**Тема работы**

**«Динамические библиотеки»**

Студент: Савинова Екатерина Ильинична

Группа: М8О-207Б-21

Вариант: 3

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Содержание**

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

**Репозиторий**

<https://github.com/savinova-kati/operating-systems/tree/main/lab5>

**Постановка задачи**

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:

1. Если пользователь вводит команду «0», то программа переключает одну реализацию контрактов на другую (необходимо только для *программы №2*).
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.

**Вариант 3**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | Рассчет интеграла функции sin(x) на отрезке [A, B] с шагом e | Float SinIntegral(float A, float B, float e) | Подсчет интеграла методом прямоугольников. | Подсчет интеграла методом трапеций. |
| 4 | Подсчёт наибольшего общего делителя для двух натуральных чисел | Int GCF(int A, int B) | Алгоритм Евклида | Наивный алгоритм. Пытаться разделить числа на все числа, что меньше A и B. |

Система сборки: CMake.

Вариант 2:

При начале работы программа печатает ID и версию компилятора, при помощи которого была собрана, а также дату и время сборки. Программа должна печатать корректные значения при сборке на любой ОС без изменения CMake файлов.

**Общие сведения о программе**

Программа состоит из двух интерфейсов (main1.c и main2.c), каждый из них реализован по-разному, в соответствии с заданием. Также каждая реализация контрактов представляет из себя отдельный файл: lib1.c и lib2.c. Для объявления необходимых функций также используется заголовочный файл lib.h. Так как все собирается с помощью CMake, то в проекте присутствует CMakeLists.txt.

**Общий метод и алгоритм решения**

Внутри файла lib.h мы объявляем функции, которые будут использоваться. Используем спецификатор хранения extern, который сообщает компилятору, что находящиеся за ним типы и имена переменных объявляются где-то в другом месте.

В файлах lib1.c и lib2.c мы прописываем логику работы наших функций. В первом – первую реализацию, во втором – вторую.

Используемые алгоритмы:

* Косинус — сумма ряда Тейлора;
* Факториал — факториал «Деревом»;
* Возведение в степень — алгоритм «бинарного» возведения в степень.

Интерфейс 1:

Подключаем lib.h и пользуемся функциями так, как будто библиотека обычная.

Интерфейс 2:

Воспользуемся системными вызовами из библиотеки <dlfcn.h>.

Функция dlopen открывает динамическую библиотеку (объект .so) по названию.

Функция dlsym - обоаботчик динамически загруженного объекта вызовом dlopen.

Функция dlclose, соответственно, закрывает динамическую библиотеку.

**Исходный код**

**lib.h**

#ifndef \_\_LIB\_H\_\_

#define \_\_LIB\_H\_\_

extern "C" float SinIntegral(float a, float b, float e);

extern "C" int GCF(int a, int b);

#endif

#endif

**lib1.cpp**

#include<iostream>

using namespace std;

// extern позволяет компилятору знать о типах и именах глобальных переменных без действительного создания этих переменных

extern "C" float SinIntegral(float a, float b, float e)

{

float square = 0;

for (float i = a; i <= b; i += e) {

square += e \* sin(i);

}

return square;

}

extern "C" int GCF(int a, int b)

{

while (a != 0 && b != 0) {

if (a > b) {

a = a % b;

} else {

b = b % a;

}

}

return a + b;

}

/\*

int main()

{

int a, b;

float a1, b1, e;

cin >> a;

cout << endl;

cin >> b;

cout << endl;

cin >> a1;

cout << endl;

cin >> b1;

cout << endl;

cin >> e;

cout << endl;

cout << SinIntegral(a1, b1, e) << endl;

cout << GCF(a, b) << endl;

return 0;

}

**lib2.cpp**

#include<iostream>

using namespace std;

extern "C" float SinIntegral(float a, float b, float e)

{

float square = 0;

for (float i = a; i < b; i += e) {

square += e \* ((sin(i) + sin(i + e)) / 2);

}

return square;

}

extern "C" int GCF(int a, int b)

{

int max\_del = 0;

if (a > b) {

for (int i = 1; i <= b; i++) {

if (a % i == 0 && b % i == 0) {

max\_del = i;

}

}

} else {

for (int i = 1; i <= a; i++) {

if (a % i == 0 && b % i == 0) {

max\_del = i;

}

}

}

return max\_del;

}

/\*int main()

{

int a, b;

float a1, b1, e;

cin >> a;

cout << endl;

cin >> b;

cout << endl;

cin >> a1;

cout << endl;

cin >> b1;

cout << endl;

cin >> e;

cout << endl;

cout << SinIntegral(a1, b1, e) << endl;

cout << GCF(a, b) << endl;

return 0;

}

**main1.c**

#include<iostream>

#include<stdio.h>

#include<cmath>

#include"lib.h"

#include"config.h"

using namespace std;

int main()

{

cout << "ID компьютера: " << COMP\_ID << endl;

cout << "Версия компьютера: " << COMP\_VER << endl;

cout << "Дата компиляции: "<< TIME\_NOW << endl;

cout << "Записывайте комманды в виде: <command> <arg1> <arg2> ... <argn>" << endl;

cout << "Если вы хотите посчитать интеграл функции sin(x) на отрезке [a, b] с шагом e, введите: 1 <a> <b> <e> " << endl;

cout << "Если вы хотите найти наибольший общий делитель двух натуральных чисел, введите: 2 <a> <b> " << endl;

int command;

while(cin >> command) {

if (command == 2) {

int a, b;

cin >> a >> b;

int res2 = GCF(a, b);

cout << "Наибольший общий делитель " << a << " и " << b << " - " << res2 << endl;

} else if (command == 1) {

float a1, b1, e;

cin >> a1 >> b1 >> e;

float res1 = SinIntegral(a1, b1, e);

cout << "Интеграл функции sin(x) на отрезке [" << a1 << ", " << b1 << "] с шагом " << e << " - " << res1 << endl;

} else {

cout << "Неверно введенная команда. Повторите ввод" << endl;

}

}

}

**main2.c**

#include<iostream>

#include<stdio.h>

#include<cmath>

#include<dlfcn.h>

#include"lib.h"

#include"config.h"

#include<string>

using namespace std;

int main()

{

cout << "ID компьютера: " << COMP\_ID << endl;

cout << "Версия компьютера: " << COMP\_VER << endl;

cout << "Дата сборки: "<< TIME\_NOW << endl;

cout << "Сейчас вы находитесь в 1 реализации программы " << endl;

cout << "Записывайте комманды в виде: <command> <arg1> <arg2> ... <argn>" << endl;

cout << "Если вы хотите посчитать интеграл функции sin(x) на отрезке [a, b] с шагом e, введите 1 <a> <b> <e> " << endl;

cout << "Если вы хотите найти наибольший общий делитель двух натуральных чисел, введите 2 <a> <b> " << endl;

cout << "Если вы хотите поменять реализацию программы, введите 0 <a> <b> " << endl;

int command;

string lib1 = "./liblib1.dylib"; // хранятся динамические библиотеки

string lib2 = "./liblib2.dylib";

void\* cur\_lib = dlopen(lib1.c\_str(), RTLD\_LAZY); //загружает динамическую библиотеку

//RTLD\_LAZY, подразумевающим разрешение неопределенных символов в виде кода, содержащегося в исполняемой динамической библиотеке

float (\*SinIntegral)(float a, float b, float e);

int (\*GCF)(int a, int b);

SinIntegral = (float(\*)(float, float, float))dlsym(cur\_lib, "SinIntegral");

GCF = (int(\*)(int, int))dlsym(cur\_lib, "GCF");

int id = 1;

while(cin >> command) {

if (command == 0) {

dlclose(cur\_lib);

if (id == 1) {

cur\_lib = dlopen(lib2.c\_str(), RTLD\_LAZY);

id = 2;

cout << "Теперь вы находитесь во 2 реализации программы " << endl;

} else {

cur\_lib = dlopen(lib1.c\_str(), RTLD\_LAZY);

id = 1;

cout << "Теперь вы находитесь в 1 реализации программы " << endl;

}

SinIntegral = (float(\*)(float, float, float))dlsym(cur\_lib, "SinIntegral");

GCF = (int(\*)(int, int))dlsym(cur\_lib, "GCF");

} else if (command == 2) {

int a, b;

cin >> a >> b;

int res2 = GCF(a, b);

cout << "Наибольший общий делитель " << a << " и " << b << " - " << res2 << endl;

} else if (command == 1) {

float a1, b1, e;

cin >> a1 >> b1 >> e;

cout << a1 << " " << b1 << " " << e << endl;

float res1 = SinIntegral(a1, b1, e);

cout << "Интеграл функции sin(x) на отрезке [" << a1 << ", " << b1 << "] с шагом " << e << " - " << res1 << endl;

} else {

cout << "Неверно введенная команда. Повторите ввод" << endl;

}

}

}

**config.h.in**

#ifndef CONFIG\_H\_IN

#define CONFIG\_H\_IN

#define PROJECT\_NAME "@PROJECT\_NAME@"

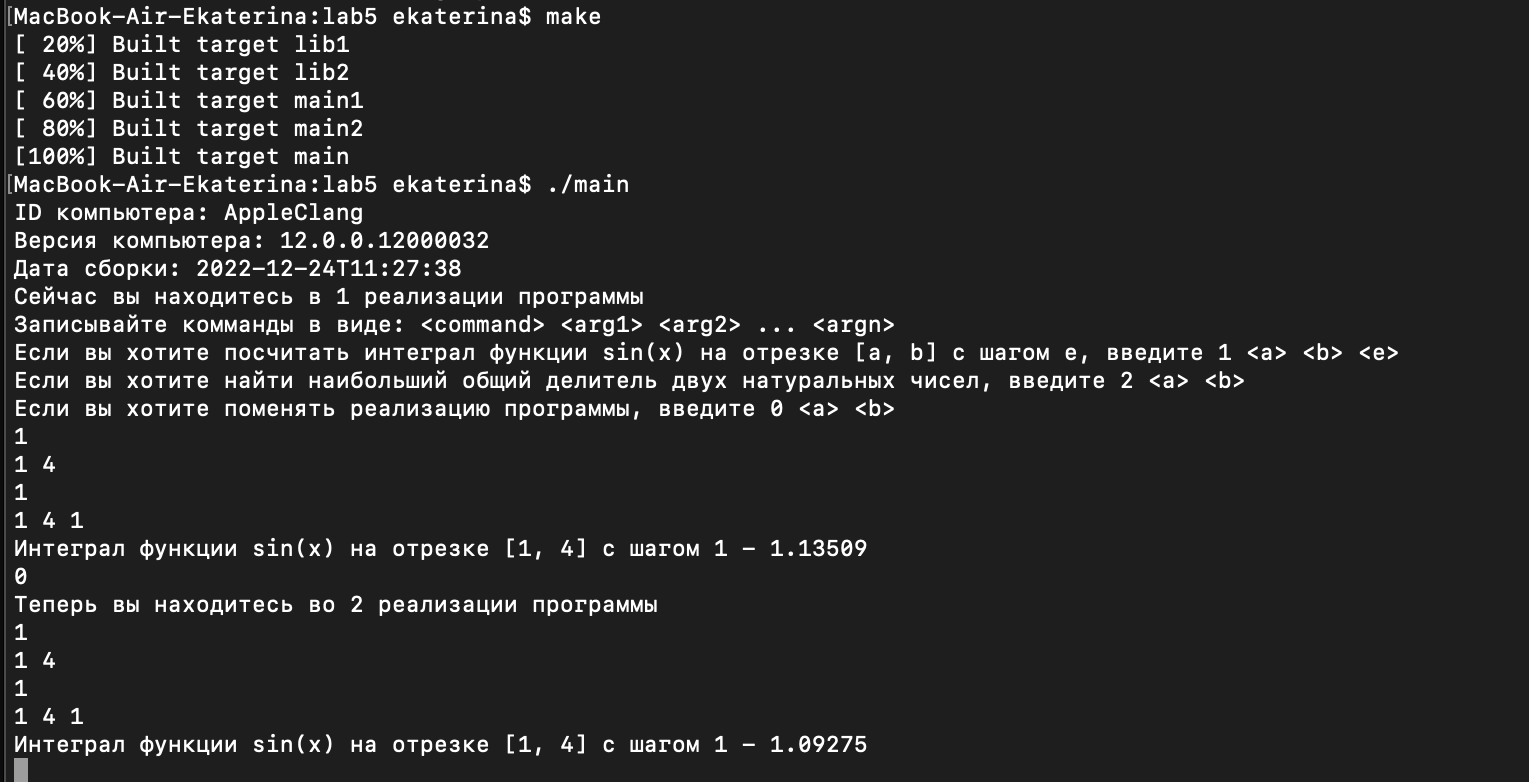
#define COMP\_ID "@COMP\_ID@"

#define COMP\_VER "@COMP\_VER@"

#define TIME\_NOW "@TIME\_NOW@"

#endif // CONFIG\_H\_IN

**Демонстрация работы программы**

****

**Выводы**

**Благодаря данной лабораторной работе я познакомилась с динамическими библиотеками, научилась с ними работать, и теперь могу сильно упрощать себе работу с большими проектами**