Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа № 5 по курсу**

**«Операционные системы»**

Студент: Постнов Александр Вячеславович

Группа: М8О-201Б-21

Вариант: 5

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/kappaprideonly/mai-os-labs

**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.

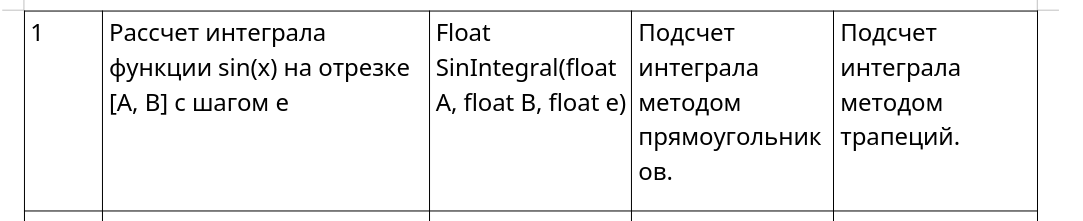
Провести анализ двух типов использования библиотек.

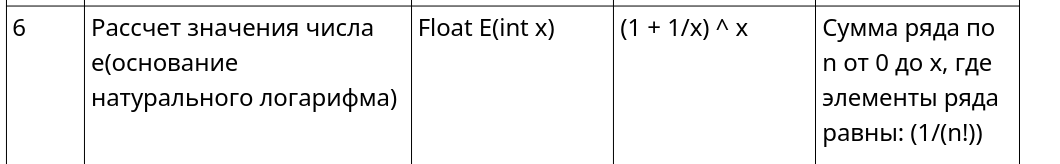
Пользовательский ввод для обоих программ должен быть организован следующим образом:

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

CMakeLists.txt - описание процесса сборки проекта  
  
main\_dynamic.cpp - main, который подгружает библиотеку во время runtime  
  
main\_static.cpp - main, который подключает библиотеку во время компиляции  
  
signature.h - объявления функций

implem1.cpp - 1 реализация функций

implem2.cpp - 2 реализация функций

lab5\_test.cpp - тесты

**Общий метод и алгоритм решения**Для статической реализации необходимо скомпилировать исполняемый файл с библиотекой. Для динамической реализации, небходимо использовать системный вызов dlopen, чтобы подгрузить библиотеку во время runtime.

**Исходный код**

**CMakeLists.txt**

**# Generate shared / static librarry for hello.cpp**

**# STATIC generates `libhello.a`**

**# SHARED geneartes `libhello.so`**

**add\_library(d1\_static STATIC src/implem1.cpp include/signature.h)**

**add\_library(d2\_static STATIC src/implem2.cpp include/signature.h)**

**add\_library(d1\_dynamic SHARED src/implem1.cpp include/signature.h)**

**add\_library(d2\_dynamic SHARED src/implem2.cpp include/signature.h)**

**target\_include\_directories(d1\_static PRIVATE include)**

**target\_include\_directories(d2\_static PRIVATE include)**

**target\_include\_directories(d1\_dynamic PRIVATE include)**

**target\_include\_directories(d2\_dynamic PRIVATE include)**

**# Genarete `main` executable from `main.cpp`**

**add\_executable(main\_static\_1 main\_static.cpp)**

**add\_executable(main\_static\_2 main\_static.cpp)**

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

**target\_include\_directories(main\_static\_1 PRIVATE include)**

**target\_include\_directories(main\_static\_2 PRIVATE include)**

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

**# Link `main` program with library**

**target\_link\_libraries(main\_static\_1 PRIVATE d1\_static)**

**target\_link\_libraries(main\_static\_2 PRIVATE d2\_static)**

**signature.h**

**#ifndef SIGNATURE\_H**

**#define SIGNATURE\_H**

**#include <cstddef>**

**#include <algorithm>**

**#include <cmath>**

**extern "C" {**

**float E(int x);**

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

**}**

**#endif /\* SIGNATURE \*/**

**implem1.cpp**

**#include "signature.h"**

**float E(int x) {**

**if (x < 0) {**

**return -1;**

**}**

**float e = 1.0;**

**for (int i = 0; i < x; ++i) {**

**e \*= 1 + 1 / static\_cast<float>(x);**

**}**

**return e;**

**}**

**float SinIntegral(float a, float b, float e) {**

**if (a > b || e > (b - a)) {**

**return -1;**

**}**

**float h = e;**

**float w = 0;**

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

**w += std::sin(i);**

**}**

**return h \* w;**

**}**

**implem2.cpp**

**#include "signature.h"**

**#include <iostream>**

**float E(int x) {**

**if (x < 0) {**

**return -1;**

**}**

**float e = 1.0;**

**float term = 1.0;**

**for (int i = 1; i <= x; ++i) {**

**term /= i;**

**e += term;**

**}**

**return e;**

**}**

**float SinIntegral(float a, float b, float e) {**

**float value = 0;**

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

**value += (std::sin(i) + std::sin(i + e)) / 2;**

**}**

**return value \* e;**

**}**

**main\_dynamic.cpp**

**#include <array>**

**#include <cstdio>**

**#include <iostream>**

**#include <cstdlib>**

**#include <dlfcn.h>**

**#include <string>**

**#include <vector>**

**int main() {**

**const std::vector <std::string> LIB = {"./libd1\_dynamic.so", "./libd2\_dynamic.so"};**

**const std::vector <std::string> FUNC\_NAME = {"E", "SinIntegral"};**

**int curlib = 0;**

**float (\*e)(int x);**

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

**void\* handle = dlopen(LIB[curlib].c\_str(), RTLD\_LAZY);**

**if (handle == nullptr) {**

**std::cout << "Fail dlopen\n";**

**return EXIT\_FAILURE;**

**}**

**e = (float(\*)(int))dlsym(handle, FUNC\_NAME[0].c\_str());**

**sinIntegral = (float(\*)(float, float, float))dlsym(handle, FUNC\_NAME[1].c\_str());**

**int command;**

**while (std::cin >> command) {**

**if (command == 1) {**

**int num;**

**std::cin >> num;**

**float value = e(num);**

**if (value == -1) {**

**std::cout << "incorrect value of variable" << std::endl;**

**continue;**

**}**

**std::cout << value << std::endl;**

**} else if (command == 2) {**

**float a;**

**float b;**

**float e;**

**std::cin >> a >> b >> e;**

**float value = sinIntegral(a, b, e);**

**if (value == -1) {**

**std::cout << "incorrect values of variables" << std::endl;**

**continue;**

**}**

**std::cout << value << std::endl;**

**} else if (command == 0) {**

**dlclose(handle);**

**curlib ^= 1;**

**void\* handle = dlopen(LIB[curlib].c\_str(), RTLD\_LAZY);**

**if (handle == nullptr) {**

**std::cout << "Fail dlopen\n";**

**return EXIT\_FAILURE;**

**}**

**e = (float(\*)(int))dlsym(handle, FUNC\_NAME[0].c\_str());**

**sinIntegral = (float(\*)(float, float, float))dlsym(handle, FUNC\_NAME[1].c\_str());**

**} else {**

**std::cout << "you have to enter 1 or 2" << std::endl;**

**}**

**}**

**dlclose(handle);**

**}**

**main\_static.cpp**

**#include <iostream>**

**#include "signature.h"**

**int main() {**

**int command;**

**while (std::cin >> command) {**

**if (command == 1) {**

**int num;**

**std::cin >> num;**

**float value = E(num);**

**if (value == -1) {**

**std::cout << "incorrect value of variable" << std::endl;**

**continue;**

**}**

**std::cout << value << std::endl;**

**} else if (command == 2) {**

**float a;**

**float b;**

**float e;**

**std::cin >> a >> b >> e;**

**float value = SinIntegral(a, b, e);**

**if (value == -1) {**

**std::cout << "incorrect values of variables" << std::endl;**

**continue;**

**}**

**std::cout << value << std::endl;**

**} else {**

**std::cout << "you have to enter 1 or 2" << std::endl;**

**}**

**}**

**}**

**lab5\_test.cpp**

**#include <cstddef>**

**#include <cstdio>**

**#include <cstdlib>**

**#include <fstream>**

**#include <gtest/gtest.h>**

**#include <string>**

**#include <dlfcn.h>**

**#include "signature.h"**

**TEST(Lab5Test, DynamicTest) {**

**const std::vector <std::string> FUNC\_NAME = {"E", "SinIntegral"};**

**const float expectE = 2.71;**

**const auto inputParameterE = {500, 300, 200};**

**const std::vector <std::vector <float>> inputParameterSin = {**

**{1, 5, 0.001},**

**{5, 6, 0.001},**

**{1, 100, 0.001}**

**};**

**const std::vector <float> answerSin = {0.2566, -0.676, -0.322};**

**float (\*eOne)(int x);**

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

**void\* handleOne = dlopen(getenv("PATH\_TO\_LIB1"), RTLD\_LAZY);**

**std::cout << getenv("PATH\_TO\_LIB1");**

**ASSERT\_NE(handleOne, nullptr);**

**float (\*eTwo)(int x);**

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

**void\* handleTwo = dlopen(getenv("PATH\_TO\_LIB2"), RTLD\_LAZY);**

**ASSERT\_NE(handleTwo, nullptr);**

**eOne = (float(\*)(int))dlsym(handleOne, FUNC\_NAME[0].c\_str());**

**eTwo = (float(\*)(int))dlsym(handleTwo, FUNC\_NAME[0].c\_str());**

**sinIntegralOne = (float(\*)(float, float, float))dlsym(handleOne, FUNC\_NAME[1].c\_str());**

**sinIntegralTwo = (float(\*)(float, float, float))dlsym(handleTwo, FUNC\_NAME[1].c\_str());**

**for (const auto& elem : inputParameterE) {**

**auto eOutOne = eOne(elem);**

**auto eOutTwo = eTwo(elem);**

**EXPECT\_NEAR(eOutOne, expectE, 0.1);**

**EXPECT\_NEAR(eOutTwo, expectE, 0.1);**

**}**

**for (size\_t i = 0; i < inputParameterSin.size(); ++i) {**

**float a = inputParameterSin[i][0];**

**float b = inputParameterSin[i][1];**

**float e = inputParameterSin[i][2];**

**auto sqOne = sinIntegralOne(a, b, e);**

**auto sqTwo = sinIntegralTwo(a, b, e);**

**EXPECT\_NEAR(sqOne, answerSin[i], 0.1);**

**EXPECT\_NEAR(sqTwo, answerSin[i], 0.1);**

**}**

**}**

**TEST(Lab5Test, StaticOneTest) {**

**const float expectE = 2.71;**

**const auto inputParameterE = {500, 300, 200};**

**const std::vector <std::vector <float>> inputParameterSin = {**

**{1, 5, 0.001},**

**{5, 6, 0.001},**

**{1, 100, 0.001}**

**};**

**const std::vector <float> answerSin = {0.2566, -0.676, -0.322};**

**for (const auto& elem : inputParameterE) {**

**auto eOut = E(elem);**

**EXPECT\_NEAR(eOut, expectE, 0.1);**

**}**

**for (size\_t i = 0; i < inputParameterSin.size(); ++i) {**

**float a = inputParameterSin[i][0];**

**float b = inputParameterSin[i][1];**

**float e = inputParameterSin[i][2];**

**auto sq = SinIntegral(a, b, e);**

**EXPECT\_NEAR(sq, answerSin[i], 0.1);**

**}**

**}**

**Демонстрация работы программы**

➜ lab5 git:(main) ✗ ./main\_dynamic

1 100

2.70481

3

you have to enter 1 or 2

0

1 100

2.71828

2 1 100 0.1

-0.321262

0

2 1 100 0.1

-0.304547

➜ lab5 git:(main) ✗ ./main\_static\_1

1 100

2.70481

2 1 100 0.1

-0.304547

➜ lab5 git:(main) ✗ ./main\_static\_2

1 100

2.71828

2 1 100 0.1

-0.321262

**Выводы:**

Понял различие динамических и статических библиотек. Создал динамическую библиотеку. Создал программы, которые используют функции динамических библиотек. Разобрался как собирается программа(более подробно этапы сборки), узнал различие extern и inline.