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: Мазепа Илья Алексеевич

Группа: М8О-209Б-23

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_

Москва, 2024

**GitHub репозиторий:** https://github.com/Tyhyqo/mai\_oc

## Тема

Динамические библиотеки

## Цель работы

Целью является приобретение практических навыков в:

* Создании динамических библиотек
* Создании программ, которые используют функции динамических библиотек

## Задание

Требуется создать динамические библиотеки, которые реализуют заданный вариантом функционал. Далее использовать данные библиотеки двумя 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 использует одну из библиотек, используя информацию, полученную на этапе компиляции - Тестовая программа (программа №2), которая загружает библиотеки, используя только их относительные пути и контракты
* Провести анализ двух типов использования библиотек.

## Пользовательский ввод

Пользовательский ввод для обеих 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

## Контракты и реализации функций

### Вариант:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Функция № | Описание | Сигнатура | Реализация 1 | Реализация 2 |
| 1 | Рассчет производной функции cos(x) в точке A с приращением deltaX | Derivative(float A, float deltaX) | f’(x) = (f(A + deltaX) – f(A)) / deltaX | f’(x) = (f(A + deltaX) – f(A - deltaX)) / (2 \* deltaX) |
| 2 | Рассчет значения числа Пи при заданной длине ряда (K) | float Pi(int K) | Ряд Лейбница | Формула Валлиса |

## Реализация

### Динамические библиотеки

Derivative1.c

#include <math.h>  
  
float Derivative(float A, float deltaX) {  
 return (cosf(A + deltaX) - cosf(A)) / deltaX;  
}

#### 

Derivative2.c

#include <math.h>  
  
float Derivative(float A, float deltaX) {  
 return (cosf(A + deltaX) - cosf(A - deltaX)) / (2 \* deltaX);  
}

#### 

Pi1.c

float Pi(int K) {  
 float pi = 0.0f;  
 for (int i = 0; i < K; i++) {  
 pi += (i % 2 == 0 ? 1.0f : -1.0f) / (2 \* i + 1);  
 }  
 return pi \* 4;  
}

#### 

Pi2.c

float Pi(int K) {  
 float pi = 1.0f;  
 for (int i = 1; i <= K; i++) {  
 pi \*= ((2.0f \* i) / (2.0f \* i - 1.0f)) \* ((2.0f \* i) / (2.0f \* i + 1.0f));  
 }  
 return pi \* 2;  
}

### Тестовые программы

#### 

Program1.c

#include <stdio.h>  
  
extern float Derivative(float, float);  
extern float Pi(int);  
  
int main() {  
 char command;  
 printf("Введите команду:\n");  
 while (scanf(" %c", &command) != EOF) {  
 if (command == '1') {  
 float A, deltaX;  
 scanf("%f %f", &A, &deltaX);  
 printf("Результат: %f\n", Derivative(A, deltaX));  
 } else if (command == '2') {  
 int K;  
 scanf("%d", &K);  
 printf("Результат: %f\n", Pi(K));  
 } else {  
 printf("Неизвестная команда.\n");  
 }  
 }  
 return 0;  
}

#### 

Program2.c

#include <stdio.h>  
#include <dlfcn.h>  
  
int main() {  
 char command;  
 printf("Введите команду:\n");  
 while (scanf(" %c", &command) != EOF) {  
 if (command == '1') {  
 float A, deltaX;  
 scanf("%f %f", &A, &deltaX);  
  
 void\* handle = dlopen("./libDerivative1.so", RTLD\_LAZY);  
 if (!handle) {  
 fprintf(stderr, "Ошибка загрузки библиотеки: %s\n", dlerror());  
 return 1;  
 }  
 float (\*Derivative)(float, float) = dlsym(handle, "Derivative");  
 char\* error = dlerror();  
 if (error != NULL) {  
 fprintf(stderr, "Ошибка загрузки функции: %s\n", error);  
 dlclose(handle);  
 return 1;  
 }  
 printf("Результат: %f\n", Derivative(A, deltaX));  
 dlclose(handle);  
 } else if (command == '2') {  
 int K;  
 scanf("%d", &K);  
  
 void\* handle = dlopen("./libPi1.so", RTLD\_LAZY);  
 if (!handle) {  
 fprintf(stderr, "Ошибка загрузки библиотеки: %s\n", dlerror());  
 return 1;  
 }  
 float (\*Pi)(int) = dlsym(handle, "Pi");  
 char\* error = dlerror();  
 if (error != NULL) {  
 fprintf(stderr, "Ошибка загрузки функции: %s\n", error);  
 dlclose(handle);  
 return 1;  
 }  
 printf("Результат: %f\n", Pi(K));  
 dlclose(handle);  
 } else {  
 printf("Неизвестная команда.\n");  
 }  
 }  
 return 0;  
}

### 

CMakeLists.txt

cmake\_minimum\_required(VERSION 3.0)  
project(Lab4)  
  
set(CMAKE\_C\_STANDARD 11)  
set(CMAKE\_C\_FLAGS "${CMAKE\_C\_FLAGS} -fPIC")  
  
# Создание динамических библиотек  
add\_library(Derivative1 SHARED Derivative1.c)  
target\_link\_libraries(Derivative1 m)  
  
add\_library(Derivative2 SHARED Derivative2.c)  
target\_link\_libraries(Derivative2 m)  
  
add\_library(Pi1 SHARED Pi1.c)  
  
add\_library(Pi2 SHARED Pi2.c)  
  
# Тестовая программа №1 (линковка на этапе компиляции)  
add\_executable(Program1 Program1.c)  
target\_link\_libraries(Program1 Derivative1 Pi1 m)  
  
# Тестовая программа №2 (динамическая загрузка библиотек)  
add\_executable(Program2 Program2.c)  
target\_link\_libraries(Program2 dl)

## Анализ

### Программа №1

Программа №1 использует динамические библиотеки, которые подключаются на этапе компиляции. Это позволяет компилятору и линковщику проверить наличие всех необходимых функций и их сигнатур. Преимущества такого подхода:

* Простота использования
* Высокая производительность, так как все символы разрешаются на этапе компиляции

Недостатки:

* Невозможность смены реализации функций без перекомпиляции программы

**Пример лога работы программы**

tyhyqo@BOOK-L939VNBBJO:~/Education/MAI/C/mai\_oc/lab\_4/build$ ./Program1

Введите команду:

1 10 20

Результат: 0.049666

2 20

Результат: 3.091624

### Программа №2

Программа №2 загружает динамические библиотеки во время исполнения с помощью интерфейса ОС для работы с динамическими библиотеками. Преимущества такого подхода:

* Гибкость, возможность смены реализации функций без перекомпиляции программы
* Возможность загрузки библиотек по требованию, что может уменьшить использование памяти

Недостатки:

* Более сложная реализация
* Потенциально более низкая производительность из-за необходимости разрешения символов во время исполнения

**Пример лога работы программы**

tyhyqo@BOOK-L939VNBBJO:~/Education/MAI/C/mai\_oc/lab\_4/build$ ./Program2

Введите команду:

1 20 10

Результат: -0.025383

2 10

Результат: 3.041840

## Заключение

В данной лабораторной работе были созданы динамические библиотеки, реализующие заданный функционал, и две тестовые программы, использующие эти библиотеки разными способами. Программа №1 использует библиотеки, подключенные на этапе компиляции, что обеспечивает простоту и высокую производительность. Программа №2 загружает библиотеки во время исполнения, что обеспечивает гибкость и возможность смены реализации функций без перекомпиляции программы.