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»

Группа: М8О-211БВ-24

Студент: Бахшалиев М.А.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 26.11.25

Москва, 2025

Постановка задачи

Вариант 5.

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.

Вариант 5. Расчет значения числа π при заданной длине ряда (k):

Сигнатура функции: float pi(int k);

● Реализация №1: Ряд Лейбница

● Реализация №2: Формула Валлиса

Общий метод и алгоритм решения

В рамках данной работы была реализована программа для вычисления числа π двумя различными математическими методами: формулой Лейбница и формулой Валлиса. Общий подход к решению задачи заключался в создании модульной архитектуры, позволяющей использовать оба алгоритма как в статическом, так и в динамическом режиме. Основная идея заключалась в разделении реализации алгоритмов в отдельные библиотечные модули с единым интерфейсом, что обеспечивает гибкость использования и легкость расширения.

Для формулы Лейбница был применен алгоритм, основанный на разложении в бесконечный ряд: π/4 = 1 - 1/3 + 1/5 - 1/7 + 1/9 - ... . Алгоритм последовательно вычисляет частичные суммы этого ряда, чередуя знаки слагаемых, и умножает конечный результат на 4 для получения приближенного значения π. Количество итераций k задается пользователем и определяет точность вычисления - чем больше итераций, тем ближе результат к истинному значению π.

Для формулы Валлиса был реализован алгоритм, основанный на бесконечном произведении: π/2 = (2/1)×(2/3)×(4/3)×(4/5)×(6/5)×(6/7)×... . Алгоритм вычисляет произведение дробей вида $(2n)^2/((2n-1)(2n+1))$ для n от 1 до k, после чего умножает результат на 2 для получения итогового значения π. Этот метод также позволяет регулировать точность вычисления через параметр k.

Архитектурное решение предусматривает создание двух раздельных библиотек (libpi_leibniz.so и libpi_wallis.so), экспортирующих единую функцию pi(int k) через общий заголовочный файл pi_lib.h. Для демонстрации работы реализованы две тестовые программы: test_static, статически линкующаяся с одной из библиотек на этапе компиляции, и test_dynamic, динамически загружающая библиотеки во время выполнения с возможностью переключения между ними без перезапуска программы. Динамическая версия использует механизм dlopen/dlsym для загрузки символов из разделяемых библиотек, что обеспечивает инкапсуляцию алгоритмов и возможность их горячей замены. Такой подход демонстрирует принципы модульного программирования и позволяет наглядно сравнить эффективность различных математических методов вычисления π при одинаковых параметрах точности.

Код программы

pi_lib.h:

```
#ifndef PI_LIB_H
#define PI_LIB_H

float pi(int k);

#endif
```

pi_leibniz.c:

```
#include "../include/pi_lib.h"

float pi(int k) {
    float sum = 0.0f;
    for (int i = 0; i < k; i++) {
        float term = 1.0f / (2 * i + 1);
        if (i % 2 == 1) {
            term = -term;
        }
        sum += term;
    }
    return 4.0f * sum;
}
```

pi_wallis.c:

```
#include "../include/pi_lib.h"

float pi(int k) {
    float product = 1.0f;
    for (int n = 1; n <= k; n++) {
        float numerator = (2 * n) * (2 * n);
        float denominator = (2 * n - 1) * (2 * n + 1);
        product *= numerator / denominator;
    }
    return 2.0f * product;
}
```

test_dynamic.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>
#include <string.h>
```

```
typedef float (*pi_func_t)(int);

int main(int argc, char *argv[]) {
    pi_func_t pi_func = NULL;
    void *handle = NULL;

    const char *default_lib = "./libpi_leibniz.so";
    const char *other_lib = "./libpi_wallis.so";

    handle = dlopen(default_lib, RTLD_LAZY);
    if (!handle) {
        fprintf(stderr, "Ошибка загрузки библиотеки %s: %s\n", default_lib, dlerror());
        return 1;
    }
    pi_func = (pi_func_t)dlsym(handle, "pi");
    if (!pi_func) {
        fprintf(stderr, "Ошибка: символ 'pi' не найден в библиотеке\n");
        dlclose(handle);
        return 1;
    }

    char command[100];
    int k;

    while (1) {
        printf("Команда (0 — выйти, 1 k — вычислить pi(k), 2 - сменить библиотеку): ");
        if (fgets(command, sizeof(command), stdin) == NULL) {
            break;
        }

        if (command[0] == '2') {
            dlclose(handle);

            if (strcmp(default_lib, "./libpi_leibniz.so") == 0) {
                default_lib = other_lib;
            } else {
                default_lib = "./libpi_leibniz.so";
            }

            handle = dlopen(default_lib, RTLD_LAZY);
            if (!handle) {
                fprintf(stderr, "Ошибка загрузки %s: %s\n", default_lib, dlerror());
                handle = dlopen("./libpi_leibniz.so", RTLD_LAZY);
```

```
                if (!handle) return 1;
            }
            pi_func = (pi_func_t)dlsym(handle, "pi");
            if (!pi_func) {
                fprintf(stderr, "Ошибка: символ 'pi' не найден\n");
                dlclose(handle);
                return 1;
            }
            printf("Переключено на библиотеку: %s\n", default_lib);
        } else if (command[0] == '1') {
            if (sscanf(command + 2, "%d", &k) == 1 && k > 0) {
                float result = pi_func(k);
                printf("pi(%d) = %.6f (библиотека: %s)\n", k, result, default_lib);
            } else {
                printf("Ошибка: неверный формат аргумента\n");
            }
        } else if (command[0] == '0') {
            break;
        } else {
            printf("Неизвестная команда\n");
        }
    }

    if (handle) {
        dlclose(handle);
    }
    return 0;
}
```

test_static.c:

```
#include <stdio.h>
#include <stdlib.h>
#include "include/pi_lib.h"

int main() {
    char command[100];
    int k;

    while (1) {
        printf("Команда (0 – выйти, 1 k – вычислить pi(k)): ");
        if (fgets(command, sizeof(command), stdin) == NULL) {
            break;
        }
```

```
        if (command[0] == '0') {
            break;
        } else if (command[0] == '1') {
            if (sscanf(command + 2, "%d", &k) == 1 && k > 0) {
                float result = pi(k);
                printf("pi(%d) = %.6f\n", k, result);
            } else {
                printf("Ошибка: неверный формат аргумента\n");
            }
        } else {
            printf("Неизвестная команда\n");
        }
    }

    return 0;
}
```

Протокол работы программы

Тестирование:

```
make all
gcc -Wall -fPIC -Iinclude -c src/pi_leibniz.c -o pi_leibniz.o
gcc -shared -o libpi_leibniz.so pi_leibniz.o
gcc -Wall -fPIC -Iinclude -c src/pi_wallis.c -o pi_wallis.o
gcc -shared -o libpi_wallis.so pi_wallis.o
gcc test_static.c -L. -lpi_leibniz -o test_static -ldl
gcc test_dynamic.c -o test_dynamic -ldl
make run
LD_LIBRARY_PATH=. ./test_static
Команда (0 — выйти, 1 k — вычислить pi(k)): 1 100
pi(100) = 3.131593
Команда (0 — выйти, 1 k — вычислить pi(k)): 1 1000
pi(1000) = 3.140593
Команда (0 — выйти, 1 k — вычислить pi(k)): 1 10000
pi(10000) = 3.141499
Команда (0 — выйти, 1 k — вычислить pi(k)): 0
LD_LIBRARY_PATH=. ./test_dynamic
Команда (0 — выйти, 1 k — вычислить pi(k), 2 - сменить библиотеку): 1 100
```

```
pi(100) = 3.131593 (библиотека: ./libpi_leibniz.so)
Команда (0 — выйти, 1 k — вычислить pi(k), 2 - сменить библиотеку): 1 1000
pi(1000) = 3.140593 (библиотека: ./libpi_leibniz.so)
Команда (0 — выйти, 1 k — вычислить pi(k), 2 - сменить библиотеку): 1 10000
pi(10000) = 3.141499 (библиотека: ./libpi_leibniz.so)
Команда (0 — выйти, 1 k — вычислить pi(k), 2 - сменить библиотеку): 2
Переключено на библиотеку: ./libpi_wallis.so
Команда (0 — выйти, 1 k — вычислить pi(k), 2 - сменить библиотеку): 1 100
pi(100) = 3.133788 (библиотека: ./libpi_wallis.so)
Команда (0 — выйти, 1 k — вычислить pi(k), 2 - сменить библиотеку): 1 1000
pi(1000) = 3.140806 (библиотека: ./libpi_wallis.so)
Команда (0 — выйти, 1 k — вычислить pi(k), 2 - сменить библиотеку): 1 10000
pi(10000) = 3.141350 (библиотека: ./libpi_wallis.so)
Команда (0 — выйти, 1 k — вычислить pi(k), 2 - сменить библиотеку): 0
```

Вывод

В результате выполнения работы была успешно разработана и реализована программа для вычисления числа π двумя различными математическими методами. Программа продемонстрировала эффективность модульного подхода в программировании, позволив создать гибкую архитектуру с возможностью как статической, так и динамической компоновки. Оба алгоритма - Лейбница и Валлиса - показали свою работоспособность, сходясь к ожидаемому значению π с увеличением количества итераций. Динамическая версия программы особенно наглядно продемонстрировала преимущества разделяемых библиотек, обеспечив возможность "горячей" замены алгоритмов без перекомпиляции основной программы. Реализованное решение подтвердило практическую ценность использования динамической загрузки библиотек в языках программирования семейства C для создания гибких и расширяемых приложений. Полученный опыт может быть успешно применен в более сложных проектах, требующих модульности и возможности runtime-конфигурации.