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-209Б-23

Студент: Сидоров Д.Д.

Преподаватель: Миронов Е.С.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 18.12.24

Москва, 2024

**Постановка задачи**

**Вариант 28.**

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

В конечном итоге, в лабораторной работе необходимо получить следующие части:

1. Динамические библиотеки, реализующие контракты, которые заданы вариантом;
2. Тестовая программа (программа №1), которая используют одну из библиотек, используя информацию полученные на этапе компиляции;
3. Тестовая программа (программа №2), которая загружает библиотеки, используя только их относительные пути и контракты.
4. 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.

Функции

1. Рассчет значения числа Пи при заданной длине ряда (K) методами Лейбница и Валлиса
2. Подсчет площади плоской геометрической фигуры по двум сторонам. Фигуры прямоугольник и прямоугольный треугольник

**Общий метод и алгоритм решения**

Использованные системные вызовы:

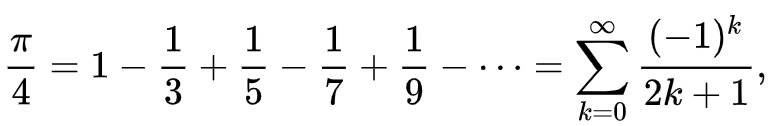
● void\* dlopen(const char\* filename, int flag); – загружает динамическую библиотеку в память.   
● int dlclose(void\* handle); – освобождает ресурсы, связанные с загруженной библиотекой.

**Алгоритм решения**

Программа состоит из нескольких частей, которые стоит рассмотреть по отдельности.

**Функции библиотеки**

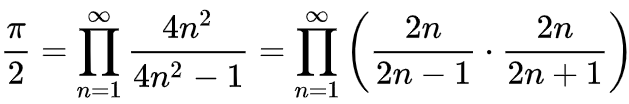
Функция числа Pi. Данную функцию необходимо вычислить двумя реализациями. Через формулу Лейбница и Валлиса. В математике формула Лейбница для π, названная в честь Готфрида Вильгельма Лейбница, утверждает, что:



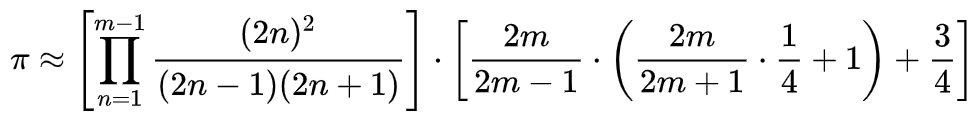
Это значит, что можно написать функцию, которая складывает n дробей и получает приближенное значение числа Пи.

Аналогично математик Валлис вывел формулу, выражающую число

Пи через бесконечное произведение рациональных дробей:



Проблема данной формулы заключается в том, что она медленно сходится, это означает, что только на бесконечности формула получает число Пи. Поэтому стоит рассмотреть отдельно последний член и посчитать его отдельно. Тогда формула будет иметь вид:



**Program 1**

Программа 1 должна использовать линковку динамических библиотек на этапе компиляции соответственно для подключения функций достаточно объявить их перед функцией main. Сама код достаточно прост, в зависимости от ввода пользователя запускаем вычисление либо числа Пи либо площади прямоугольника.

**Program 2**

Програма 2 использует динамическую загрузку библиотек в процессе выполнения, это означает, что при смене варианта реализации функций командой “0” необходимо выгрузить старую библиотеку и загрузить новую, так же необходимо заново получить адреса функций Pi и Square. Обе программы Program 1 и Program 2 работают в цикле, который прерывается после ввода команды “-1”.

**Makefile**

С помощью Makefile выполняется сборка и компиляция программы, makefile состоит из целей, реквизитами и командами. Для успешной компиляции нужно описать все цели, реквизиты и указать команды. Для компиляции С++ используем компилятор g++. Так как мы берем на себя линковку, то при компиляции нужно использовать флаг -c, чтобы получить объектный файл.

**Код программы**

Makefile

all: Program\_1 Program\_2 libPr\_2\_real1.so libPr\_2\_real2.so clean

Program\_1: Program\_1.o libPr\_2\_real1.so

    g++ -o Program\_1 Program\_1.o -L. -lPr\_2\_real1 -Wl,-rpath,.

Program\_1.o: Program\_1.cpp

    g++ -c Program\_1.cpp

Pi\_Leibniz.o: Pi\_Leibniz.cpp

    g++ -c Pi\_Leibniz.cpp

Square\_rect.o: Square\_rect.cpp

    g++ -c Square\_rect.cpp

Program\_2: Program\_2.o

    g++ -o Program\_2 Program\_2.o -ldl

Program\_2.o: Program\_2.cpp

    g++ -c Program\_2.cpp

libPr\_2\_real1.so: Pi\_Leibniz.o Square\_rect.o

    g++ -shared -o libPr\_2\_real1.so Pi\_Leibniz.o Square\_rect.o

libPr\_2\_real2.so: Pi\_Wallis.o Square\_tr.o

    g++ -shared -o libPr\_2\_real2.so Pi\_Wallis.o Square\_tr.o

Pi\_Wallis.o:

    g++ -c Pi\_Wallis.cpp

Square\_tr.o:

    g++ -c Square\_tr.cpp

clean:

    rm -f \*.o

**Pi\_Leibniz.cpp**

extern "C" float Pi(int k)

{

    float res = 0;

    for (int i = 0; i < k; ++i)

    {

        int a = 1;

        if (i % 2 != 0)

            a = -1;

        res += (float(a) / (2.0 \* float(i) + 1.0));

    }

    return 4 \* res;

}

**Pi\_Wallis.cpp**

extern "C" float Pi(int k)

{

    float res = 1;

    for (int i = 1; i < k; ++i)

    {

        res \*= (4.0 \* float(i) \* float(i)) / (4.0 \* float(i) \* float(i) - 1.0);

    }

    float m = float(k);

    float ost\_chlen = ((2.0 \* m) / (2.0 \* m - 1.0)) \* (((2.0 \* m / (2.0 \* m + 1.0)) \* (1.0 / 4.0) + 1) + 3.0 / 4.0);

    return res \* ost\_chlen;

}

**Square\_rect.cpp**

extern "C" float Square(float a, float b)

{

    return a \* b;

}

**Square\_tr.cpp**

extern "C" float Square(float a, float b)

{

    return a \* b / 2;

}

**Program\_1.cpp**

#include <iostream>

extern "C" float Pi(int k);

extern "C" float Square(float a, float b);

int main()

{

    int prog;

    while (true)

    {

        std::cout << "Input program code: -1-exit, 1-calc PI, 2-calc square\n";

        std::cin >> prog;

        switch (prog)

        {

        case 1:

            int k;

            std::cin >> k;

            std::cout << "Pi number: " << Pi(k) << "\n\n";

            break;

        case 2:

            int a, b;

            std::cin >> a >> b;

            std::cout << Square(a, b) << "\n\n";

            break;

        default:

            std::cout << "Exit\n";

            return 0;

        }

    }

}

**Program\_2.cpp**

#include <iostream>

#include <dlfcn.h>

int main()

{

    int prog = 1;

    int real = 1;

    void \*lib = nullptr;

    typedef float (\*PiFunc)(int);

    typedef float (\*SquareFunc)(float, float);

    PiFunc Pi;

    SquareFunc Square;

    lib = dlopen("./libPr\_2\_real1.so", RTLD\_LAZY);

    if (!lib)

    {

        std::cerr << "Error loading initial library: " << dlerror() << std::endl;

        return 1;

    }

    std::cout << "Library is loaded\n";

    Pi = (PiFunc)dlsym(lib, "Pi");

    Square = (SquareFunc)dlsym(lib, "Square");

    if (!Pi || !Square)

    {

        std::cerr << "Failed to load symbols: " << dlerror() << std::endl;

        dlclose(lib);

        return 1;

    }

    while (true)

    {

        std::cout << "Input program code: -1-exit, 0-change realisation, 1-calc PI, 2-calc square\n";

        std::cin >> prog;

        switch (prog)

        {

        case 0:

            dlclose(lib);

            if (real == 1)

            {

                lib = dlopen("./libPr\_2\_real2.so", RTLD\_LAZY);

                real = 2;

            }

            else

            {

                lib = dlopen("./libPr\_2\_real1.so", RTLD\_LAZY);

                real = 1;

            }

            if (!lib)

            {

                std::cerr << "Error loading library: " << dlerror() << std::endl;

                return 1;

            }

            std::cout << "Library is loaded\n";

            Pi = (PiFunc)dlsym(lib, "Pi");

            Square = (SquareFunc)dlsym(lib, "Square");

            if (!Pi || !Square)

            {

                std::cerr << "Failed to load symbols: " << dlerror() << std::endl;

                dlclose(lib);

                return 1;

            }

            break;

        case 1:

            int k;

            std::cin >> k;

            if (real == 1)

                std::cout << "Teck realization of Pi is Leibniz\n";

            else

                std::cout << "Teck realization of Pi is Wallis\n";

            std::cout << "Pi number: " << k <<" " << Pi(k) << "\n\n";

            break;

        case 2:

            int a, b;

            std::cin >> a >> b;

            if (real == 1)

                std::cout << "Teck realization of Square is Rectangle\n";

            else

                std::cout << "Teck realization of Pi is Triangle\n";

            std::cout << "Square is " << Square(a, b) << "\n\n";

            break;

        default:

            std::cout << "Exit\n";

            dlclose(lib);

            return 0;

        }

    }

}

**Протокол работы программы**

**Тестирование:**

**Тест 1:**

./Program\_2

Library is loaded

Input program code: -1-exit, 0-change realisation, 1-calc PI, 2-calc square

1 100

Teck realization of Pi is Leibniz

Pi number: 100 3.1416

Input program code: -1-exit, 0-change realisation, 1-calc PI, 2-calc square

2 10 5

Teck realization of Square is Rectangle

Square is 50

Input program code: -1-exit, 0-change realisation, 1-calc PI, 2-calc square

-1

Exit

**Тест 2:**

./Program\_2

Library is loaded

Input program code: -1-exit, 0-change realisation, 1-calc PI, 2-calc square

1 5

Teck realization of Pi is Leibniz

Pi number: 5 3.1416

Input program code: -1-exit, 0-change realisation, 1-calc PI, 2-calc square

0

Library is loaded

Input program code: -1-exit, 0-change realisation, 1-calc PI, 2-calc square

1 10

Teck realization of Pi is Wallis

Pi number: 10 3.14133

Input program code: -1-exit, 0-change realisation, 1-calc PI, 2-calc square

-1

Exit

**Вывод**

Данная лабораторная работа включает в себя два варианта загрузки библиотек. Динамическую загрузку в процессе выполнения программы и статическую при компиляции и линковки. Программа с линковкой во время компиляции работает быстрее, так как ей не требуется загружать необходимые библиотеки в процессе. Динамическая загрузка позволяет самому принимать решение о том, какие конкретно библиотеки требуются и загружать нужную, это сильно экономит память. Выполнение лабораторной работы помогло понять разницу между статическими и динамическими библиотеками.