МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ

ФЕДЕРАЦИИ

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: Теребаев К. Д.

Группа: М8О-203Б-22

Преподаватель: Миронов Е. С.

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2024

**Цель работы**

Целью является приобретение практических навыков в:

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

**Задание**

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:

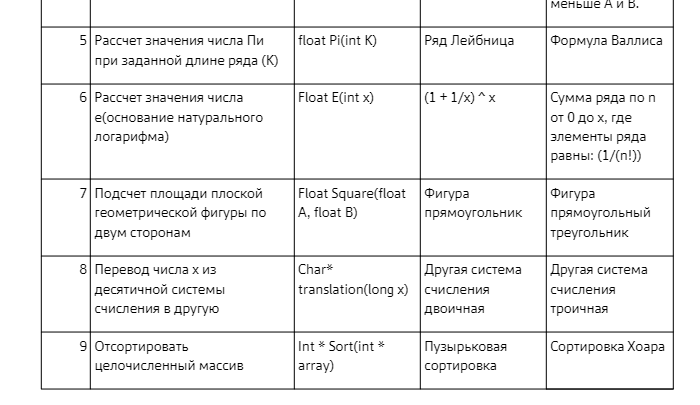
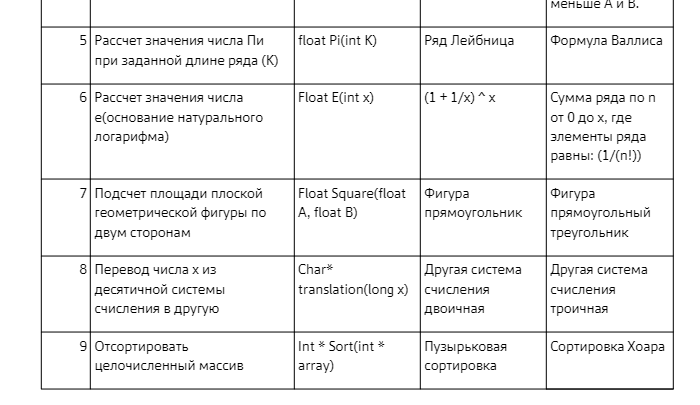
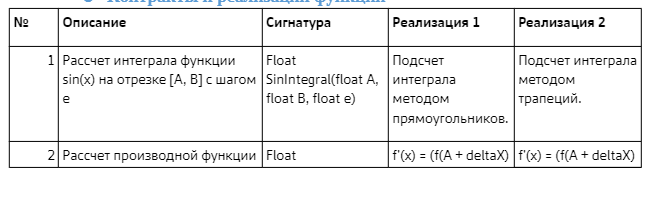
* Динамические библиотеки, реализующие контракты, которые заданы вариантом;
* Тестовая программа (программа №1), которая используют одну из библиотек, используя знания полученные на этапе компиляции;
* Тестовая программа (программа №2), которая загружает библиотеки, используя только их местоположение 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.

30 вариант



**Общие сведения о программе**

1. dlopen () – загружает (открывает) динамическую библиотеку. Возвращает указатель на загруженную библиотеку, в случае ошибки возвращает NULL;
2. dlsym () – получение адреса функции или переменной из библиотеки. Возвращает адрес функции, в случае ошибки возвращает NULL;
3. dlerror () – возвращает понятную человеку строку, описывающую последнюю ошибку, которая произошла при вызове одной из функции dlopen, dlsym, dlclose. Возвращает NULL если не возникло ошибок с момента инициализации или с момента ее последнего вызова;
4. dlclose () – уменьшает на единицу счетчик ссылок на указатель динамической библиотеки. Возвращает 0 при удачном завершении и ненулевой результат при ошибке.

**Общий алгоритм решения**

Описываем решения в библиотечных файлах, создаём общий заголовочный файл. Нам не потребуется два, так как в обеих реализациях одни и те же функции, соответственно, между двумя заголовочными файлами не было бы различия. Далее собираем всё в исполняемый файл.

**Основные файлы программы**

function.hpp  
#ifndef FUNCTION\_HPP

#define FUNCTION\_HPP

extern "C" {

float Pi(int k);

int\* Sort(int\* array, int size);

}

#endif

function1.cpp  
#include <utility>

#include "function.hpp"

float Pi(int k) {

    float result = 0.0;

    for (int i = 0; i < k; ++i) {

        result += (1.0 - i % 2 \* 2.0) / (2.0 \* i + 1.0);

    }

    return result \* 4.0;

}

int\* Sort(int\* array, int size) {

    bool sorted;

    for (int i{0}; i < size; ++i) {

        sorted = true;

        for (int j{0}; j < size - i - 1; ++j) {

            if (array[j] > array[j + 1]) {

                std::swap(array[j], array[j + 1]);

                sorted = false;

            }

        }

        if (sorted) break;

    }

    return array;

}

function2.cpp  
#include <cmath>

#include "function.hpp"

float Pi(int k) {

    float result = 1.0;

    for (int i = 1; i <= k; ++i) {

        result \*= (4.0 \* i \* i) / (4.0 \* i \* i - 1.0);

    }

    return 2.0 \* result;

}

std::pair<int, int> partition(int\* a, int begin, int end, int pivot) {

    int lt\_end{begin}, eq\_end{begin};

    for (int i{begin}; i < end; ++i) {

        if (a[i] < pivot) {

            std::swap(a[eq\_end], a[lt\_end]);

            if (eq\_end != i) {

                std::swap(a[lt\_end], a[i]);

            }

            ++lt\_end;

            ++eq\_end;

        } else if (a[i] == pivot) {

            std::swap(a[eq\_end], a[i]);

            ++eq\_end;

        }

    }

    return std::pair<int, int>(lt\_end, eq\_end);

}

void qsort(int\* a, int begin, int end) {

    if ((end - begin) < 2) return;

    int pivot = a[begin + rand() % (end - begin)];

    std::pair<int, int> i = partition(a, begin, end, pivot);

    qsort(a, begin, i.first);

    qsort(a, i.second, end);

}

int\* Sort(int\* array, int size) {

    qsort(array, 0, size);

    return array;

}

static\_main.cpp  
#include <iostream>

#include "function.hpp"

using namespace std;

int main() {

    int type;

    while (cin >> type) {

        if (type == 1) {

            int k;

            cin >> k;

            cout << Pi(k) << endl;

        } else if (type == 2) {

            int size;

            cin >> size;

            int \*array = new int[size];

            for (int i = 0; i < size; ++i) {

                cin >> array[i];

            }

            Sort(array, size);

            for (int i = 0; i < size; ++i) {

                cout << array[i] << " ";

            }

            cout << endl;

            delete[] array;

        } else {

            break;

        }

    }

}

dynamic\_main.cpp  
#include <dlfcn.h>

#include <iostream>

using namespace std;

typedef float (\*pi\_func)(int);

typedef int\* (\*sort\_func)(int\*, int);

pi\_func Pi = nullptr;

sort\_func Sort = nullptr;

void\* lib\_handle = nullptr;

void load\_lib(const char\* file) {

    lib\_handle = dlopen(file, RTLD\_LAZY);

    if (!lib\_handle) {

        throw runtime\_error(dlerror());

    }

    Pi = (pi\_func)dlsym(lib\_handle, "Pi");

    if (!Pi) {

        throw runtime\_error(dlerror());

    }

    Sort = (sort\_func)dlsym(lib\_handle, "Sort");

    if (!Sort) {

        throw runtime\_error(dlerror());

    }

}

int main() {

    load\_lib("libfunction1.so");

    int type, current\_lib = 1;

    while (cin >> type) {

        if (type == 0) {

            dlclose(lib\_handle);

            switch (current\_lib) {

                case 1:

                    load\_lib("libfunction2.so");

                    current\_lib = 2;

                    break;

                default:

                    load\_lib("libfunction1.so");

                    current\_lib = 1;

                    break;

            }

        } else if (type == 1) {

            int k;

            cin >> k;

            cout << Pi(k) << endl;

        } else if (type == 2) {

            int size;

            cin >> size;

            int\* array = new int[size];

            for (int i = 0; i < size; ++i) {

                cin >> array[i];

            }

            Sort(array, size);

            for (int i = 0; i < size; ++i) {

                cout << array[i] << " ";

            }

            cout << endl;

            delete[] array;

        } else {

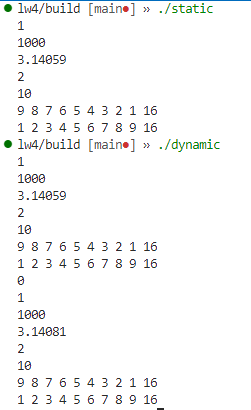
            break;

        }

    }

}

**Пример работы**

****

**Вывод**

В ходе выполнения лабораторной работы я изучил динамические библиотеки. Они определенно имеют ряд преимуществ: отсутствие необходимости перекомпилировать весь проект при изменении только одной библиотеки, возможность смены библиотеки прямо во время работы программы. Эти свойства позволяют программе быть гибкой.