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: Татаркин И. В.

Группа: М8О-208Б-23

Преподаватель: Живалев Е. А.

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

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

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

Москва, 2024

**Тема:** Динамические библиотеки

**Цель работы:** Приобретение практических навыков в:

* Создании динамических библиотек.
* Создании программ, использующих функции динамических библиотек.

**Вариант: 9**.

**Задачи:**

1. Создать динамические библиотеки, реализующие два контракта:
   * Расчет интеграла двумя способами: методом прямоугольника и методом Трапеции.
   * Отсортировать целочисленный массив двумя способами: Пузырьковой сортировкой и сортировкой Хоара
2. Реализовать две программы для работы с библиотеками:
   * Первая программа использует функции библиотек на этапе компиляции.
   * Вторая программа загружает библиотеки динамически во время исполнения и позволяет переключаться между их реализациями.
3. Реализовать возможность переключения реализаций библиотек во время выполнения программы.

**Ход работы:**

**1. Создание динамических библиотек** Были реализованы две динамические библиотеки с различными реализациями функций для вычисления интегралов и сортировки массивов.

*Функции библиотеки №1:*

* float SinIntegral\_rectangle(float A, float B, float e): Вычисление интеграла методом прямоугольника.
* float BubbleSort(int \*array, int size): Сортировка методом пузырька.

*Функции библиотеки №2:*

* float SinIntegral\_trapezoid(float A, float B, float e): Вычисление интеграла методом трапеции.
* void QuickSortRecursive(int \*array, int left, int right): Сортировка Хоара.

**Репозиторий:** https://github.com/IvanTatarkin/OS-Labs/blob/LW1/LW4

**Код библиотеки №1:**

#include <math.h>

float SinIntegral\_rectangle(float A, float B, float e) {

float sum = 0.0;

for(float x = A; x < B; x += e) {

sum += sin(x) \* e;

}

return sum;

}

void BubbleSort(int \*array, int size) {

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

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

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

int temp = array[j];

array[j] = array[j + 1];

array[j + 1] = temp;

}

}

}

}

**Код библиотеки №2:**

float SinIntegral\_trapezoid(float A, float B, float e) {

float sum = 0.0;

for(float x = A; x < B; x += e) {

sum += (sin(x) + sin(x + e)) / 2 \* e;

}

return sum;

}

void QuickSortRecursive(int \*array, int left, int right) {

if (left < right) {

int pivot = array[right];

int i = left - 1;

for (int j = left; j < right; ++j) {

if (array[j] <= pivot) {

++i;

int temp = array[i];

array[i] = array[j];

array[j] = temp;

}

}

int temp = array[i + 1];

array[i + 1] = array[right];

array[right] = temp;

QuickSortRecursive(array, left, i);

QuickSortRecursive(array, i + 2, right);

}

}

void QuickSort(int \*array, int size) { QuickSortRecursive(array, 0, size - 1); }

**2. Первая программа** использует функции динамических библиотек на этапе компиляции. Пользователь может вызывать функции для вычисления интеграла и сортировки массива.

**3. Вторая программа** загружает динамические библиотеки во время выполнения. Пользователь может переключаться между реализациями библиотек и вызывать функции вычисления интеграла и сортировки массива.

**Пример работы:**Режимы:

0 - Смена реализаций (между lib1 и lib2),

1 - Расчет интеграла sin(x),

2 - Сортировка массива,

3 - Выход.

Введите команду: 2

Введите размер массива: 3

Введите 3 целых чисел: 2 1 4

Отсортированный массив: 1 2 4

Введите команду: 0

Библиотека переключена.

Введите команду: 1

Введите A B e через пробел: 3 4 0.11

Результат интеграла sin(x) от 3.000000 до 4.000000 с шагом 0.110000: -0.414749

Введите команду:

**Выводы:** В ходе лабораторной работы были разработаны две динамические библиотеки с разными подходами к реализации функций для вычисления интеграла и сортировки массива. В процессе работы удалось изучить особенности использования динамических библиотек. Статическая линковка обеспечивает высокую производительность программы, так как подключение библиотек происходит на этапе компиляции. Динамическая загрузка, в свою очередь, позволяет изменять функционал программы без необходимости её перекомпиляции, что делает её более гибкой и удобной для расширяемых приложений. Возможность переключения между различными реализациями функций в динамически загружаемых библиотеках дает удобный способ для анализа производительности и функциональности разных подходов. Проведенные эксперименты подтвердили основные преимущества работы с динамическими библиотеками, а результаты соответствуют ожидаемым теоретическим выводам.