Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу

«Операционные системы»

Студент: Велиев Рауф Рамиз оглы
Группа: М8О-209Б-23
Вариант: 4
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2024

Содержание

Репозиторий

https://github.com/velievrauf/OS/tree/main/lab4

Постановка задачи

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 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:

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

Провести анализ двух типов использования библиотек.

Расчет значения числа e(основание натурального логарифма);

Подсчёт количества простых чисел на отрезке [A, B] (A, B - натуральные).

Общий метод и алгоритм решения

Общий метод и алгоритм решения:

1. Созданы функции E(int x) и PrimeCount(int A, int B), реализующие вычисление ряда для числа e и подсчет простых чисел в диапазоне. Они экспортируются через динамическую библиотеку (DLL).
2. Основная программа загружает библиотеку с помощью LoadLibraryA и подключает функции через GetProcAddress.
3. Функции вызываются с заданными аргументами, результаты вычислений выводятся в консоль.
4. Для подсчета простых чисел реализованы оптимизации, такие как решето Эратосфена.
5. Реализована обработка ошибок при загрузке библиотеки и функций для надежности выполнения.

Подход демонстрирует использование динамических библиотек для модульной разработки и эффективной реализации алгоритмов.

Исходный код

main.cpp

```
#include <Windows.h>
#include <string>
#include <iostream>

typedef float (*E)(int);
typedef int (*PrimeCount)(int, int);

int main() {
	HINSTANCE libHandle = nullptr;
	E eFunc = nullptr;
	PrimeCount primeCountFunc = nullptr;

	libHandle = LoadLibraryA("contract_native.dll");

	if (!libHandle) {
		std::cerr << "Failed to include lib file" << std::endl;
		return 1;
	}

	eFunc = (E)GetProcAddress(libHandle, "E");
	primeCountFunc = (PrimeCount)GetProcAddress(libHandle, "PrimeCount");

	if (!eFunc || !primeCountFunc) {
		std::cerr << "Failed to load the functions" << std::endl;
```

```
            FreeLibrary(libHandle);
            return 1;
        }

        std::cout << eFunc(2) << std::endl;

        std::cout << primeCountFunc(3, 100) << std::endl;

        return 0;
}
```

main.cpp

```
#include <contract.h>
#include <iostream>

int main() {
        std::cout << E(2) << std::endl;

        std::cout << PrimeCount(3, 100) << std::endl;

        return 0;
}
```

contract_native.h

```
#include "contract_native.h"
#include <cmath>
#include <vector>

extern "C" {
   float E(int x) {
      if (x == 0) return 1.0f;
      float result = pow(1.0f + 1.0f / x, x);
      return result;
   }

   int PrimeCount(int A, int B)
   {
      if (A > B || A < 2) return 0;

      std::vector<bool> primes(B + 1, true);
      primes[0] = primes[1] = false;

      for (int i = 2; i * i <= B; ++i) {
         if (primes[i]) {
            for (int j = i * i; j <= B; j += i) {
               primes[j] = false;
            }
         }
      }

      int count = 0;
      for (int i = A; i <= B; ++i) {
         if (primes[i]) ++count;
      }

      return count;
   }
}
```

contract_native.cpp

```
#include "contract_native.h"
#include <cmath>
#include <vector>

extern "C" {
    float E(int x) {
        if (x == 0) return 1.0f;
        float result = pow(1.0f + 1.0f / x, x);
        return result;
    }

    int PrimeCount(int A, int B)
    {
        if (A > B || A < 2) return 0;

        std::vector<bool> primes(B + 1, true);
        primes[0] = primes[1] = false;

        for (int i = 2; i * i <= B; ++i) {
            if (primes[i]) {
                for (int j = i * i; j <= B; j += i) {
                    primes[j] = false;
                }
            }
        }

        int count = 0;
        for (int i = A; i <= B; ++i) {
            if (primes[i]) ++count;
        }

        return count;
    }
}
```

contract.h

```
#pragma once

#define API _declspec(dllexport)

extern "C" {
            API float E(int x);
            API int PrimeCount(int A, int B);
}
```

contract.cpp

```
#include "contract.h"
#include <cmath>

extern "C" {
    float E(int x) {
        float sum = 0.0f;
        int factorial = 1;
        for (int n = 0; n <= x; ++n) {
            if (n > 0) {
```

```
            factorial *= n;
        }
        sum += 1.0f / factorial;
    }
    return sum;
}

int PrimeCount(int A, int B)
{
    if (A > B) return 0;

    int count = 0;
    for (int i = A; i <= B; ++i) {
        if (i < 2) {
            continue;
        }

        bool isPrime = true;;
        for (int j = 2; j * j <= i; ++j) {
            if (i % j == 0) {
                isPrime = false;
                break;
            }
        }
        if (isPrime) ++count;
    }

    return count;
}
}
```

Демонстрация работы программы

2.25 (число e во 2 степени)

24 (количество простых чисел в диапазоне от 3 до 100)

2.25 (число e во 2 степени)

7 (количество простых чисел в диапазоне от 3 до 20)

Выводы

В ходе лабораторной работы были приобретены навыки создания динамических библиотек и их использования двумя способами: на этапе компиляции и во время выполнения программы. Реализованные функции E и PrimeCount продемонстрировали корректность и эффективность, включая оптимизацию подсчёта простых чисел с помощью решета Эратосфена.

Работа с интерфейсом ОС для динамической загрузки библиотек показала гибкость данного подхода, что позволяет удобно подключать модули и расширять функциональность программ.

Лабораторная работа подчеркнула важность модульной разработки и оптимизации алгоритмов для решения прикладных задач.