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-203Б-23

Студент: Никитцев А.В.

Преподаватель: Миронов Е.С.

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

Дата: 25.12.24

Москва, 2024

**Цель работы**

Целью является приобретение практических навыков в:

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

**Постановка задачи (Вариант 30)**

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 No1), которая используют одну из библиотек, используя  
информацию полученные на этапе компиляции;  
• Тестовая программа (программа No2), которая загружает библиотеки, используя только их  
относительные пути 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 No2). 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.

Рассчет значения числа Пи при заданной длине ряда (K) (float Pi(int K)) / Ряд Лейбница (Формула Валлиса)

Отсортировать целочисленный массив (Int \* Sort(int \*array)) / Пузырьковая сортировка (Сортировка Хоара)

**Код**

**F1.cpp**

*//  
// Created by nikit on 12/4/2024.  
//*#include "f1.h"  
  
float CalcPiLeibniz(int k) {  
 float res = 0.0f;  
 float one = 1.0f;  
  
 for (int i = 0; i < k; i++) {  
 res += one / (2.0f \* i + 1.0f);  
 one \*= -1;  
 }  
  
 return res \* 4.0f;  
}  
  
std::vector<int> SortBubble(std::vector<int> array) {  
 auto swp = true;  
  
 while (swp) {  
 swp = false;  
 for (auto j = 0; j < array.size() - 1; ++j) {  
 auto curVal1 = array[j];  
 auto curVal2 = array[j + 1];  
 if (curVal2 < curVal1) {  
 swp = true;  
 array[j] = curVal2;  
 array[j + 1] = curVal1;  
 }  
 }  
 }  
  
 return array;  
}

**F1.h**

#pragma once  
#include <vector>  
extern "C" {  
 float CalcPiLeibniz(int k);  
 std::vector<int> SortBubble(std::vector<int> array);  
}

**F2.cpp**

*//  
// Created by nikit on 12/4/2024.  
//*#include "f2.h"  
  
float CalcPiVallis(int k) {  
 float result = 1;  
  
 for (auto n = 1; n <= k; ++n){  
 result \*= (4.0f \* n \* n) / (4.0f \* n \* n - 1.0f);  
 }  
  
 return 2 \* result;  
}  
  
std::vector<int> SortHoar(std::vector<int> array){  
 auto size = array.size();  
 if (size < 2) return array;  
  
 auto idxCenter = size / 2;  
 auto mid = array[idxCenter];  
  
 std::vector<int> temp1;  
 std::vector<int> temp2;  
  
 for (int i = 0; i < size; ++i){  
 if (i == idxCenter) continue;  
 if (array[i] < mid) temp1.push\_back(array[i]);  
 else temp2.push\_back(array[i]);  
 }  
  
 temp1 = SortHoar(temp1);  
 temp2 = SortHoar(temp2);  
  
 std::vector<int> result;  
  
 for (auto& item: temp1) result.push\_back(item);  
 result.push\_back(mid);  
 for (auto& item: temp2) result.push\_back(item);  
  
 return result;  
  
}

**F2.h**

#pragma once  
#include <vector>  
extern "C" {  
 float CalcPiVallis(int k);  
 std::vector<int> SortHoar(std::vector<int> array);  
}

**Test.cpp**

#include <gtest/gtest.h>  
#include <random>  
#include "../src/f1.h"  
#include "../src/f2.h"  
  
constexpr int Size = 10;  
  
std::vector<int> generateVector(const size\_t vecSize){  
 std::vector<int> vec(vecSize);  
  
 std::random\_device dev;  
 std::mt19937 rng(dev());  
 std::uniform\_int\_distribution<> rand(0,100);  
  
  
 for (auto i = 0; i < vecSize; ++i){  
 vec[i] = rand(rng);  
 }  
  
 return vec;  
}  
  
TEST(BubbleSortTests, ShouldSortCorrectly){  
 auto vec1 = generateVector(Size);  
  
 auto result = SortBubble(vec1);  
  
 std::sort(vec1.begin(), vec1.end());  
 EXPECT\_EQ(result, vec1);  
}  
  
TEST(HoarSortTests, ShouldSortCorrectly){  
 auto vec1 = generateVector(Size);  
  
 auto res = SortHoar(vec1);  
  
 std::sort(vec1.begin(), vec1.end());  
 EXPECT\_EQ(res, vec1);  
}  
  
TEST(PiTests, ShouldCalculatePiCorrectlyWithValles){  
 EXPECT\_NEAR(std::numbers::pi, CalcPiVallis(1e6), 3e-4);  
}  
  
TEST(PiTests, ShouldCalculatePiCorrectlyWithLeibniz){  
 EXPECT\_NEAR(std::numbers::pi, CalcPiLeibniz(1e6), 1e-5);  
}  
  
int main(int argc, char \*\*argv) {  
 testing::InitGoogleTest(&argc, argv);  
 return RUN\_ALL\_TESTS();  
}

**Main.cpp**

#include <istream>  
#include <vector>  
#include <iostream>  
#include <functional>  
#include "src/f1.h"  
#include "src/f2.h"  
  
using PiFunc = std::function<float (int)>;  
using Sorter = std::function<std::vector<int> (std::vector<int>)>;  
  
void handlePi(const PiFunc& CalcPi, std::istream& is, std::ostream& os){  
 int n;  
  
 os << "Enter n: ";  
 is >> n;  
  
 auto result = CalcPi(n);  
 os << "Result: " << result << std::endl;  
}  
  
void handleSort(const Sorter& Sorter, std::istream& is, std::ostream& os){  
 std::vector<int> vec;  
 size\_t size;  
  
 os << "Enter size: ";  
 is >> size;  
  
 os << "Enter values separated by space: ";  
 int val;  
 for (auto i = 0; i < size; ++i){  
 is >> val;  
 vec.push\_back(val);  
 }  
  
 os << "Sorting..." << std::endl;  
 auto result = Sorter(vec);  
 os << "Result: ";  
 for (auto& item: result) os << item << " ";  
 os << std::endl;  
}  
  
int main(int argv, char\*\* argc){  
 int command = 0;  
 std::istream& is = std::cin;  
 std::ostream& os = std::cout;  
 while (command != 5) {  
 os << "Enter command: ";  
 is >> command;  
  
 switch (command) {  
 case 1:  
 handlePi(CalcPiVallis, is, os);  
 break;  
 case 2:  
 handlePi(CalcPiLeibniz, is, os);  
 break;  
 case 3:  
 handleSort(SortHoar, is, os);  
 break;  
 case 4:  
 handleSort(SortBubble, is, os);  
 break;  
 default:  
 break;  
 }  
  
 if (command == 5) return 0;  
 }  
}

**mainDynamic.cpp**

*//  
// Created by nikit on 12/4/2024.  
//*#include <functional>  
#include <istream>  
#include <iostream>  
#include <dlfcn.h>  
  
const inline std::string pathToF1 = "./libf1.so";  
const inline std::string pathToF2 = "./libf2.so";  
  
using PiFuncPointer = float (\*)(int a);  
using SortFuncPointer = std::vector<int> (\*)(std::vector<int> a);  
  
using PiFunc = std::function<float (int)>;  
using Sorter = std::function<std::vector<int> (std::vector<int>)>;  
  
void handlePi(const PiFunc& CalcPi){  
 int n;  
  
 std::cout << "Enter n: ";  
 std::cin >> n;  
  
 auto result = CalcPi(n);  
 std::cout << "Result: " << result << std::endl;  
}  
  
void handleSort(const Sorter& Sorter){  
 std::vector<int> vec;  
 size\_t size;  
  
 std::cout << "Enter size: ";  
 std::cin >> size;  
  
 std::cout << "Enter values separated by space: ";  
 int val;  
 for (auto i = 0; i < size; ++i){  
 std::cin >> val;  
 vec.push\_back(val);  
 }  
  
 std::cout << "Sorting..." << std::endl;  
 auto result = Sorter(vec);  
 std::cout << "Result: ";  
 for (auto& item: result) std::cout << item << " ";  
 std::cout << std::endl;  
}  
  
void handlePiDynamic(void\* libHandle, const std::string& funcName) {  
 auto pi = reinterpret\_cast<PiFuncPointer>(dlsym(libHandle, funcName.c\_str()));  
  
 if (!pi) {  
 std::cout << "Error while loading library: Can't use function " << funcName << std::endl;  
 return;  
 }  
  
 handlePi(pi);  
}  
  
void handleSortDynamic(void\* libHandle, const std::string& funcName) {  
 auto pi = reinterpret\_cast<SortFuncPointer>(dlsym(libHandle, funcName.c\_str()));  
  
 if (!pi) {  
 std::cout << "Error while loading library: Can't use function " << funcName << std::endl;  
 return;  
 }  
  
 handleSort(pi);  
}  
  
void\* loadLib(const std::string& path){  
 void\* libHandle = dlopen(path.c\_str(), RTLD\_LAZY);  
 if (!libHandle) {  
 std::cout << "Error while loading library: " << dlerror() << std::endl;  
 return nullptr;  
 }  
 return libHandle;  
}  
  
int main(int argv, char\*\* argc){  
 auto command = 0;  
  
 auto libHandle = dlopen(pathToF1.c\_str(), RTLD\_LAZY);  
 auto isF1 = true;  
   
 while (command != 5) {  
 std::cout << "Enter command: ";  
 std::cin >> command;  
  
 switch (command) {  
 case 0:  
 if (isF1) {  
 libHandle = loadLib(pathToF2);  
 isF1 = false;  
 }  
 else {  
 libHandle = loadLib(pathToF1);  
 isF1 = true;  
 }  
 break;  
 case 1:  
 handlePiDynamic(libHandle, isF1 ? "CalcPiLeibniz" : "CalcPiVallis");  
 break;  
 case 2:  
 handleSortDynamic(libHandle, isF1 ? "SortBubble" : "SortHoar");  
 break;  
 default:  
 break;  
 }  
  
 if (command == 3) return 0;  
 }  
}

**Заключение**

Выполнив лабораторную работу, я приобрел навыки создания и вызова динамических библиотек.