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: Абдыкалыков Нурсултан Абдыкалыкович

Группа: М8О-206Б-23

Вариант: 1

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

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 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.

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.

Контракты и реализации функций:

1. Рассчет интеграла функции sin(x) на отрезке [A, B] с шагом e
2. Рассчет производной функции cos(x) в точке A с приращением deltaX

**Общий метод и алгоритм решения**

1. Реализация математических функций:

Созданы две версии математические функции с разной реализацией в каждой библиотеке

* SinIntegral(float A, float B, float e)
* Derivative(float A, float deltaX)

1. Модульная архитектура

Функции реализованы в виде двух динамических библиотек (libmath1.so и libmath2.so), что позволяет:

* Легко переключаться между разными методами вычислений
* Изолировать реализации друг от друга
* Загружать нужную версию во время выполнения

1. Два способа использования библиотек:

* Статическая линковка: функции жестко линкуются на этапе компиляции (программа **static\_program**)
* Динамическая загрузка: библиотеки загружаются во время выполнения с возможностью переключения (программа **dynamic\_program**)

1. Обработка ошибок.

**Исходный код**

**Mathlib1.c:**

|  |
| --- |
| #include <math.h>  float SinIntegral(float *A*, float *B*, float *e*) {      float integral = 0.0;      for(float x = *A*; x < *B*; x += *e*) {          integral += sin(x) \* *e*;      }      return integral;  }  float Derivative(float *A*, float *deltaX*) {      return (cos(*A* + *deltaX*) - cos(*A*)) / *deltaX*;  } |

**Mathlib2.c:**

|  |
| --- |
| #include <math.h>  float SinIntegral(float *A*, float *B*, float *e*) {      float integral = 0.0;      for(float x = *A*; x < *B*; x += *e*) {          integral += (sin(x) + sin(x + *e*)) \* *e* / 2;      }      return integral;  }  float Derivative(float *A*, float *deltaX*) {      return (cos(*A* + *deltaX*) - cos(*A* - *deltaX*)) / (2 \* *deltaX*);  } |

**Main\_dynamic.c:**

|  |
| --- |
| **#include <stdio.h>**  **#include <stdlib.h>**  **#include <dlfcn.h>**  **#include <string.h>**  **#include <ctype.h>**  **#include <stdbool.h>**  **typedef float (\*func\_float3)(float, float, float);**  **typedef float (\*func\_float2)(float, float);**  **bool is\_number(const char\* *str*) {**  **if (\**str* == '-' || \**str* == '+') *str*++;**  **bool has\_dot = false;**  **while (\**str*) {**  **if (\**str* == '.') {**  **if (has\_dot) return false;**  **has\_dot = true;**  **}**  **else if (!isdigit(\**str*)) {**  **return false;**  **}**  ***str*++;**  **}**  **return true;**  **}**  **bool validate\_integral\_params(float *A*, float *B*, float *e*) {**  **if (*e* <= 0.0f) {**  **printf("Ошибка: шаг 'e' должен быть > 0\n");**  **return false;**  **}**  **if (*A* >= *B*) {**  **printf("Ошибка: A должно быть меньше B\n");**  **return false;**  **}**  **return true;**  **}**  **bool validate\_derivative\_params(float *deltaX*) {**  **if (*deltaX* == 0.0f) {**  **printf("Ошибка: deltaX не может быть нулём\n");**  **return false;**  **}**  **return true;**  **}**  **int main() {**  **int current\_lib = 1;**  **char lib\_path[256];**  **void \*handle = NULL;**  **func\_float3 SinIntegral = NULL;**  **func\_float2 Derivative = NULL;**    **while(1) {**  **snprintf(lib\_path, sizeof(lib\_path), "./lib/libmath%d.so", current\_lib);**  **handle = dlopen(lib\_path, RTLD\_LAZY);**    **if(!handle) {**  **fprintf(stderr, "Ошибка загрузки %s: %s\n", lib\_path, dlerror());**  **return 1;**  **}**    **SinIntegral = (func\_float3)dlsym(handle, "SinIntegral");**  **Derivative = (func\_float2)dlsym(handle, "Derivative");**    **if(!SinIntegral || !Derivative) {**  **fprintf(stderr, "Ошибка загрузки функций: %s\n", dlerror());**  **dlclose(handle);**  **return 1;**  **}**    **printf("Используется libmath%d.so\n", current\_lib);**  **printf("Команды:\n1 A B e - интеграл sin(x)\n2 A dx - производная cos(x)\n0 - переключить метод\n> ");**    **char cmd[256];**  **while(fgets(cmd, sizeof(cmd), stdin)) {**  **cmd[strcspn(cmd, "\n")] = '\0';**    **if(cmd[0] == '0') {**  **dlclose(handle);**  **current\_lib = (current\_lib == 1) ? 2 : 1;**  **printf("Переключено на libmath%d.so\n", current\_lib);**  **break;**  **}**  **else if(cmd[0] == '1') {**  **char\* endptr;**  **float params[3];**  **char\* token = strtok(cmd + 1, " ");**  **int i = 0;**  **bool valid = true;**    **while (token && i < 3) {**  **if (!is\_number(token)) {**  **printf("Ошибка: '%s' не является числом\n", token);**  **valid = false;**  **break;**  **}**  **params[i++] = strtof(token, &endptr);**  **token = strtok(NULL, " ");**  **}**    **if (valid && i == 3) {**  **if (validate\_integral\_params(params[0], params[1], params[2])) {**  **printf("Результат интеграла: %f\n", SinIntegral(params[0], params[1], params[2]));**  **}**  **} else if (i != 3) {**  **printf("Ошибка! Требуется 3 числа: 1 A B e\n");**  **}**  **}**  **else if(cmd[0] == '2') {**  **char\* endptr;**  **float params[2];**  **char\* token = strtok(cmd + 1, " ");**  **int i = 0;**  **bool valid = true;**    **while (token && i < 2) {**  **if (!is\_number(token)) {**  **printf("Ошибка: '%s' не является числом\n", token);**  **valid = false;**  **break;**  **}**  **params[i++] = strtof(token, &endptr);**  **token = strtok(NULL, " ");**  **}**    **if (valid && i == 2) {**  **if (validate\_derivative\_params(params[1])) {**  **printf("Результат производной: %f\n", Derivative(params[0], params[1]));**  **}**  **} else if (i != 2) {**  **printf("Ошибка! Требуется 2 числа: 2 A dx\n");**  **}**  **}**  **else {**  **printf("Неизвестная команда! Доступные: 1/2/0\n");**  **}**  **printf("> ");**  **}**  **}**    **return 0;**  **}** |

**Main\_static.c:**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <math.h>  #include <ctype.h>  #include <stdbool.h>  #include <string.h>  extern float SinIntegral(float *A*, float *B*, float *e*);  extern float Derivative(float *A*, float *deltaX*);  bool is\_number(const char\* *str*) {      if (\**str* == '-' || \**str* == '+') *str*++;      bool has\_dot = false;      while (\**str*) {          if (\**str* == '.') {              if (has\_dot) return false;              has\_dot = true;          }          else if (!isdigit(\**str*)) {              return false;          }  *str*++;      }      return true;  }  int main() {      char input[100];        printf("Калькулятор:\n1 A B e - интеграл sin(x)\n2 A dx - производная cos(x)\n0 - выход\n");        while(1) {          printf("> ");          if (!fgets(input, sizeof(input), stdin)) {              printf("Ошибка чтения ввода\n");              continue;          }          input[strcspn(input, "\n")] = '\0';            if(input[0] == '1') {              char\* endptr;              float params[3];              char\* token = strtok(input + 1, " ");              int i = 0;              bool valid = true;                while (token && i < 3) {                  if (!is\_number(token)) {                      printf("Ошибка: '%s' не является числом\n", token);                      valid = false;                      break;                  }                  params[i++] = strtof(token, &endptr);                  token = strtok(NULL, " ");              }                if (valid && i == 3) {                  if (params[2] <= 0.0f) {                      printf("Ошибка: шаг 'e' должен быть > 0\n");                  }                  else if (params[0] >= params[1]) {                      printf("Ошибка: A должно быть меньше B\n");                  }                  else {                      printf("Результат: %f\n", SinIntegral(params[0], params[1], params[2]));                  }              }              else if (i != 3) {                  printf("Ошибка! Нужно 3 числа: 1 A B e\n");              }          }          else if(input[0] == '2') {              char\* endptr;              float params[2];              char\* token = strtok(input + 1, " ");              int i = 0;              bool valid = true;                while (token && i < 2) {                  if (!is\_number(token)) {                      printf("Ошибка: '%s' не является числом\n", token);                      valid = false;                      break;                  }                  params[i++] = strtof(token, &endptr);                  token = strtok(NULL, " ");              }                if (valid && i == 2) {                  if (params[1] == 0.0f) {                      printf("Ошибка: deltaX не может быть нулём\n");                  }                  else {                      printf("Результат: %f\n", Derivative(params[0], params[1]));                  }              }              else if (i != 2) {                  printf("Ошибка! Нужно 2 числа: 2 A dx\n");              }          }          else if(input[0] == '0') {              printf("Выход...\n");              break;          }          else {              printf("Неизвестная команда! Доступно: 1/2/0\n");          }      }        return 0;  } |

**Демонстрация работы программы:**

>make

>bin/static\_program или bin/dynamic\_program

>Далее будет диалоговое окно в терминале

**Для автоматизации запуска использовался Makefile, в нём следующий код:**

|  |
| --- |
| CC = gcc  CFLAGS = -shared -fPIC  SRC\_DIR = src  LIB\_DIR = lib  BIN\_DIR = bin  all: libs programs  # Сборка библиотек  libs:      mkdir -p $(LIB\_DIR)      $(CC) $(CFLAGS) $(SRC\_DIR)/mathlib1.c -o $(LIB\_DIR)/libmath1.so -lm      $(CC) $(CFLAGS) $(SRC\_DIR)/mathlib2.c -o $(LIB\_DIR)/libmath2.so -lm  # Сборка программ  programs:      mkdir -p $(BIN\_DIR)      $(CC) $(SRC\_DIR)/main\_static.c -L$(LIB\_DIR) -lmath1 -o $(BIN\_DIR)/static\_program -Wl,-rpath=$(LIB\_DIR)      $(CC) $(SRC\_DIR)/main\_dynamic.c -ldl -o $(BIN\_DIR)/dynamic\_program  # Очистка  clean:      rm -rf $(LIB\_DIR) $(BIN\_DIR) |

**Тестирование:**

|  |
| --- |
| **bin/static\_program**  **Калькулятор:**  **1 A B e - интеграл sin(x)**  **2 A dx - производная cos(x)**  **0 - выход**  **> 1 1 4 0.001**  **Результат: 1.194035**  **> 2 5 0.01**  **Результат: 0.957512**  **> 0**  **Выход...** |

|  |
| --- |
| **bin/dynamic\_program**  **Используется libmath1.so**  **Команды:**  **1 A B e - интеграл sin(x)**  **2 A dx - производная cos(x)**  **0 - переключить метод**  **> 1 1 4 0.001**  **Результат интеграла: 1.194035**  **> 2 5 0.01**  **Результат производной: 0.957512**  **> 0**  **Переключено на libmath2.so**  **Используется libmath2.so**  **Команды:**  **1 A B e - интеграл sin(x)**  **2 A dx - производная cos(x)**  **0 - переключить метод**  **> 1 1 4 0.001**  **Результат интеграла: 1.193238**  **> 2 5 0.01**  **Результат производной: 0.958930**  **> ^C** |

**Выводы**

В ходе лабораторной работы я освоил создание и применение динамических библиотек в C, реализовав два способа их подключения - статическую линковку и динамическую загрузку во время выполнения. В результате я получил ценный опыт модульной разработки, научился создавать переключаемые алгоритмы и автоматизировать сборку проектов с помощью CMake.