Лабораторная работа №4: Динамические библиотеки

Цель работы:

- 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:

- 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»;
- “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 ее выполнения;
- “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 ее выполнения.

Контракты и реализации функций

1. **Расчет интеграла функции $\sin(x)$ на отрезке $[A, B]$ с шагом e:**
   Сигнатура функции:
   ```
   float sin_integral(float a, float b, float e);
   ```
   - Реализация №1: Подсчет интеграла методом прямоугольников;
   - Реализация №2: Подсчет интеграла методом трапеций.

2. **Расчет производной функции $\cos(x)$ в точке a с приращением dx:**
   Сигнатура функции:
   ```
   float cos_derivative(float a, float dx);
   ```
   - Реализация №1: $f'(x) = (f(a + dx) - f(a)) / dx$
   - Реализация №2: $f'(x) = (f(a + dx) - f(a - dx)) / (2dx)$

3. **Подсчёт количества простых чисел на отрезке $[a, b]$ (a, b – натуральные):**
   Сигнатура функции: `int prime_count(int a, int b);`
   - Реализация №1: Наивный алгоритм. Проверить делимость текущего числа на все предыдущие числа.
   - Реализация №2: Решето Эратосфена

4. **Подсчёт наибольшего общего делителя для двух натуральных чисел:**
   Сигнатура функции: `int gcd(int a, int b);`
   - Реализация №1: Алгоритм Евклида
   - Реализация №2: Наивный алгоритм: пытаться разделить числа на все числа, что меньше a и b.

5. **Расчет значения числа π при заданной длине ряда (k):**
   Сигнатура функции: `float pi(int k);`
   - Реализация №1: Ряд Лейбница
   - Реализация №2: Формула Валлиса

6. **Расчет значения числа e (основание натурального логарифма):**
   Сигнатура функции: `float e(int x);`
   - Реализация №1: $(1 + 1 / x)^x$
   - Реализация №2: Сумма ряда по n от 0 до x, где элементы ряда равны: $(1 / (n!))$

7. **Подсчет площади плоской геометрической фигуры по двум сторонам:**
   Сигнатура функции: `float area(float a, float b);`
   - Реализация №1: Фигура прямоугольник
   - Реализация №2: Фигура прямоугольный треугольник

8. **Перевод числа x из десятичной системы счисления в другую:**
   Сигнатура функции: `char *convert(int x);`
   - Реализация №1: Перевод в двоичную
   - Реализация №2: Перевод в троичную

9. **Отсортировать целочисленный массив:**
   Сигнатура функции: `int *sort(int *array, size_t n);`
   - Реализация №1: Пузырьковая сортировка :^)
   - Реализация №2: Сортировка Хоара (за использование `qsort` из `libc` или `std::sort` и его варианты из `libstdc++` будет бан; реализуйте его самостоятельно)

Варианты:

№	Функция 1	Функция 2
1	1	2
2	1	3
3	1	4
4	1	5
5	1	6
6	1	7
7	1	8
8	1	9
9	2	3
10	2	4
11	2	5
12	2	6
13	2	7
14	2	8
15	2	9

16	3	4
17	3	5
18	3	6
19	3	7
20	3	8
21	3	9
22	4	5
23	4	6
24	4	7
25	4	8
26	4	9
27	5	6
28	5	7
29	5	8
30	5	9
31	6	7
32	6	8
33	6	9
34	7	8
35	7	9
36	8	9