МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

## Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**Отчёт по лабораторной работе № 23(9)**

по дисциплине: ”Системное программирование”

на тему: «***Многозадачность потоков,***

***автосборка сложных проектов»***

Вариант -

Выполнил**:** студент группы 10701222 Зухта

Принял**:** пр. Давыденко Н.В.

# Лабораторная работа № 23.

## Задание 1.

Проведите рефакторинг программы, разработанной ранее по заданию №3 ла-

бораторной работы №21 «Концепция «Файл» и операции ввода-вывода, созда-

ние программы», руководствуясь принципом Single Responsibility Principle.

Основные дополнительные требования к новой программе:

• Проект программы должен быть многофайловам.

• Каждая сущность приложения должна быть представлено отдельным

исходным файлом.

• Все файлы могут располагаться в одной директории.

• Объявленные функции должны храниться в заголовочных файлах.

• Расчет сложной функции произвести с помощью отдельных несинхро-

низированных потоков.

Соберите и запустите приложение, сравните результаты с результатами

предыдущей программы. Дайте объяснения.

Задание №3 лабораторной работы №21 «Концепция «Файл» и операции ввода-вывода, созда-

ние программы»

Написать на языке СИ (С++) программу графика значений расчета формулы

со сложными математическими функциями. Коэффициенты сложных

функций могут быть определены в приложении. Диапазон графика

(количество итераций) должны вводиться пользователем. В программе должен

быть предусмотрен расчет времени выполнения программы.

Расчитанные значения должны записываться в файл.

График должен выводиться на экран. В языке C нет встроенных средств для

построения графиков, однако вы можете использовать библиотеки, такие как

GNUplot для создания графиков из данных, или генерировать данные в файл и

затем визуализировать их с помощью другой программы. Установите GNUplot,

если он у вас еще не установлен.

Скомпилируйте и запустите программу.

Программа создаст файл funtion\_data.txt, который будет содержать значения x и y.

Используйте следующие команды в GNUplot для визуализации данных:

set title "График квадратного уравнения"

set xlabel "x"

set ylabel "y"

plot " funtion\_data.txt" using 1:2 with lines title "y = f1(x)/f2(x)"

Эти команды устанавливают заголовок и подписи осей, а также строят график

из данных, записанных в файл. Вы можете запустить GNUplot, затем ввести

указанные команды для построения графика.

**Листинг кода**

Functions.h и functions.cpp - объявление и реализация функций для вычислений.

Functions.h

**#ifndef FUNCTIONS\_H**

**#define FUNCTIONS\_H**

**double complexFunction(double x);**

**#endif // FUNCTIONS\_H**

**functions.cpp**

**#include "functions.h"**

**#include <cmath>**

**double complexFunction(double x) {**

**return sin(x) + cos(x) + x \* x; // Пример сложной функции**

**}**

**file\_io.h и file\_io.cpp - функции для работы с файлами.**

**file\_io.h:**

**#ifndef FILE\_IO\_H**

**#define FILE\_IO\_H**

**void writeDataToFile(const char\* filename, double\* xValues, double\* yValues, int count);**

**#endif // FILE\_IO\_H**

**file\_io.cpp**

****#include "file\_io.h"****

****#include <fstream>****

****void writeDataToFile(const char\* filename, double\* xValues, double\* yValues, int count) {****

****std::ofstream outFile(filename);****

****for (int i = 0; i < count; ++i) {****

****outFile << xValues[i] << " " << yValues[i] << "\n";****

****}****

****outFile.close();****

****}****

**graph.h **и** graph.cpp **- функции для построения графиков (или для генерации данных для графиков).****

**graph.h**

****#ifndef GRAPH\_H****

****#define GRAPH\_H****

****void plotGraph(const char\* filename);****

****#endif // GRAPH\_H****

**graph.cpp**

**#include "graph.h"**

**#include <cstdlib>**

**void plotGraph(const char\* filename) {**

**std::string command = "gnuplot -e \"set title 'График квадратного уравнения'; "**

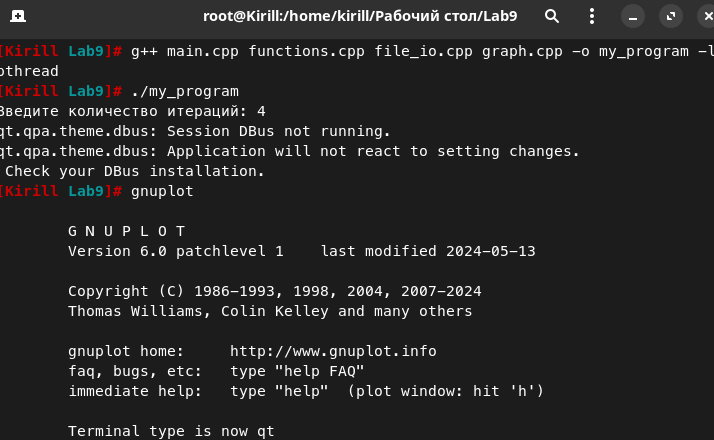
**"set xlabel 'x'; set ylabel 'y'; "**

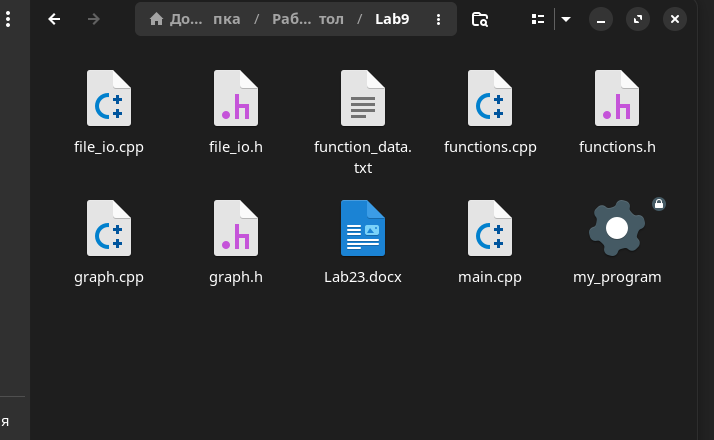
**"plot '" + std::string(filename) + "' using 1:2 with lines title 'y = f1(x)/f2(x)'\"";**

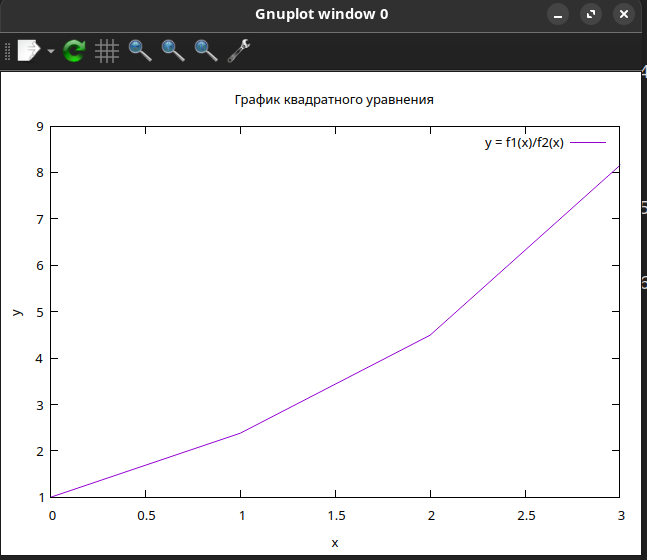
**system(command.c\_str());**

**}**

Скриншоты







**Структура файлов**:

* **main.cpp**: Основной файл, который управляет процессом. Он получает входные данные, запускает вычисления в потоке и вызывает функции для записи данных и построения графика.
* **functions.h** и **functions.cpp**: Содержат объявление и реализацию сложной функции, которую вы будете вычислять.
* **file\_io.h** и **file\_io.cpp**: Реализуют функции для записи данных в файл.
* **graph.h** и **graph.cpp**: Содержат функции для вызова GNUplot для построения графика.

**Потоки**:

* Использование потоков позволяет выполнять вычисления параллельно с выполнением других задач, таких как запись данных в файл.

**Ввод и вывод**:

* Программа запрашивает у пользователя количество итераций, вычисляет значения по заданной функции и сохраняет их в файл function\_data.txt.



Задание 2

Проведите рефакторинг программы из 1-го задания так чтобы синхронизиро-

вать соответствующие потоки.

Соберите и запустите приложение, сравните результаты с результатами

предыдущей программы. Дайте объяснения

Листинг

Измененный main

#include <iostream>

#include <thread>

#include <mutex>

#include <condition\_variable>

#include "functions.h"

#include "file\_io.h"

#include "graph.h"

std::mutex mtx; // Мьютекс для синхронизации

std::condition\_variable cv; // Условная переменная

bool ready = false; // Флаг для проверки готовности

void calculateValues(double\* xValues, double\* yValues, int count) {

for (int i = 0; i < count; ++i) {

yValues[i] = complexFunction(xValues[i]);

}

// Защита доступа к флагу готовности

std::lock\_guard<std::mutex> lock(mtx);

ready = true; // Устанавливаем флаг готовности

cv.notify\_one(); // Уведомляем основной поток

}

int main() {

int iterations;

std::cout << "Введите количество итераций: ";

std::cin >> iterations;

double\* xValues = new double[iterations];

double\* yValues = new double[iterations];

for (int i = 0; i < iterations; ++i) {

xValues[i] = i; // Или другой способ задания x

}

std::thread calcThread(calculateValues, xValues, yValues, iterations);

// Ожидание, пока поток не завершит вычисления

{

std::unique\_lock<std::mutex> lock(mtx);

cv.wait(lock, [] { return ready; }); // Ожидание уведомления

}

writeDataToFile("function\_data.txt", xValues, yValues, iterations);

plotGraph("function\_data.txt");

calcThread.join(); // Ждем завершения потока

delete[] xValues;

delete[] yValues;

return 0;

}

**Мьютекс и условная переменная**:

* + Мы добавили std::mutex для защиты доступа к общему ресурсу (флагу ready).
  + Условная переменная cv используется для блокировки основного потока до тех пор, пока поток вычислений не завершит свою работу и не уведомит основной поток.

1. **Флаг готовности**:
   * ready — это логическая переменная, которая сигнализирует, что вычисления завершены.
2. **Синхронизация**:
   * В основном потоке происходит ожидание с помощью cv.wait(lock, [] { return ready; });, что делает его неактивным до тех пор, пока поток вычислений не установит флаг ready и не вызовет cv.notify\_one().

Рефакторинг добавил синхронизацию потоков, что делает программу более надежной и предотвращает возможные ошибки, связанные с одновременным доступом к общим ресурсам.

* Программа продолжает выполнять те же вычисления, но теперь с гарантией, что данные не будут записываться, пока вычисления не завершены.

Скриншоты



Задание 3

Проведите компиляцию многофайлового проекта с прохождением всех ста-

дий компиляции. Для ускорения работы примените утилиту make.

Запустите и сравните полученный результат с результатом программы из ла-

бораторной работы №21.

Исходные файлы программ обязательно должны содержать коментарии.

Сборка проекта должна содержать файлы с результатами препроцессинга. Ис-

следуйте фалы препроцессора, найдите в них код своей программы.

Определите размеры исходных, препроцессорных, ассемблерных, объектных

и исполняемых файлов. С помощью соответствующий консольных команд

определите форматы этих файлов. Результаты подтвердите скриншотами.

Листинг

// main.cpp

#include <iostream>

#include <thread>

#include <mutex>

#include <condition\_variable>

#include "functions.h"

#include "file\_io.h"

#include "graph.h"

std::mutex mtx; // Мьютекс для синхронизации

std::condition\_variable cv; // Условная переменная

bool ready = false; // Флаг для проверки готовности

// Функция для вычисления значений

void calculateValues(double\* xValues, double\* yValues, int count) {

for (int i = 0; i < count; ++i) {

yValues[i] = complexFunction(xValues[i]);

}

std::lock\_guard<std::mutex> lock(mtx);

ready = true;

cv.notify\_one();

}

int main() {

// Основная логика программы

}

# Makefile для сборки проекта Lab9

CXX = g++

CXXFLAGS = -Wall -g

TARGET = my\_program

# Исходные файлы

SOURCES = main.cpp functions.cpp file\_io.cpp graph.cpp

OBJECTS = $(SOURCES:.cpp=.o)

# Правила сборки

all: $(TARGET)

$(TARGET): $(OBJECTS)

$(CXX) $(OBJECTS) -o $@

# Правило для компиляции исходных файлов в объектные

%.o: %.cpp

$(CXX) $(CXXFLAGS) -c $< -o $@

# Правило для препроцессинга

preprocess:

$(CXX) $(CXXFLAGS) -E main.cpp > main.i

# Очистка

clean:

rm -f $(OBJECTS) $(TARGET) main.i

Скриншоты

