**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук  
Департамент программной инженерии

**Микропроект № 2**

по дисциплине

"Архитектура вычислительных систем"

**Пояснительная записка**

Исполнитель:

студент группы БПИ192

Гураевский М. Д.

**Москва 2020**

Содержание

[**Введение** 3](#_Toc58878200)

[**Наименование программы** 3](#_Toc58878201)

[**Текст задания** 3](#_Toc58878202)

[**Документы, на основании которых ведётся разработка** 3](#_Toc58878203)

[**Описание программы** 3](#_Toc58878204)

[**Тестирование** 4](#_Toc58878205)

[**Текст программы** 6](#_Toc58878206)

[**Формат вывода** 7](#_Toc58878207)

[**Список литературы** 8](#_Toc58878208)

# **Введение**

## **Наименование программы**

Данная программа представляет собой упрощённую версию задачи о писателях и читателях – регулируются только действия читателей.

## **Текст задания**

Задача о читателях и писателях-2 («грязное чтение»). Базу данных разделяют два типа потоков – читатели и писатели. Читатели выполняют транзакции, которые просматривают записи базы данных, транзакции писателей и просматривают, и изменяют записи. Предполагается, что в начале БД находится в непротиворечивом состоянии (т. е. отношения между данными имеют смысл). Транзакции выполняются в режиме «грязного чтения», то есть процесс-писатель не может получить доступ к БД только в том случае, если ее занял другой процесс-писатель, а процессы-читатели ему не мешают. Создать многопоточное приложение с потоками-писателями и потоками-читателями. Реализовать решение, используя семафоры, и не используя блокировки чтения-записи.

## **Документы, на основании которых ведётся разработка**

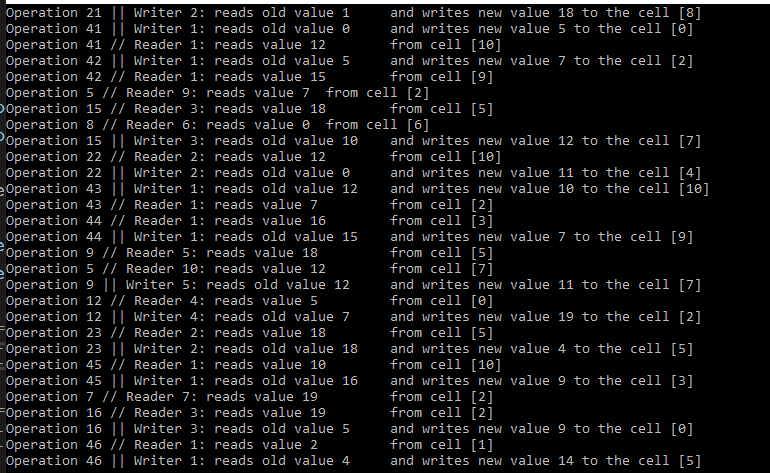
Программа выполнена в рамках задания по дисциплине "Архитектура вычислительных систем", в соответствии с учебным планом подготовки бакалавров по направлению 09.03.04 «Программная инженерия», 2 курс 2 модуль.

Основанием для разработки является письмо профессора факультета компьютерных наук Легалова Александра Ивановича от 28.11.20

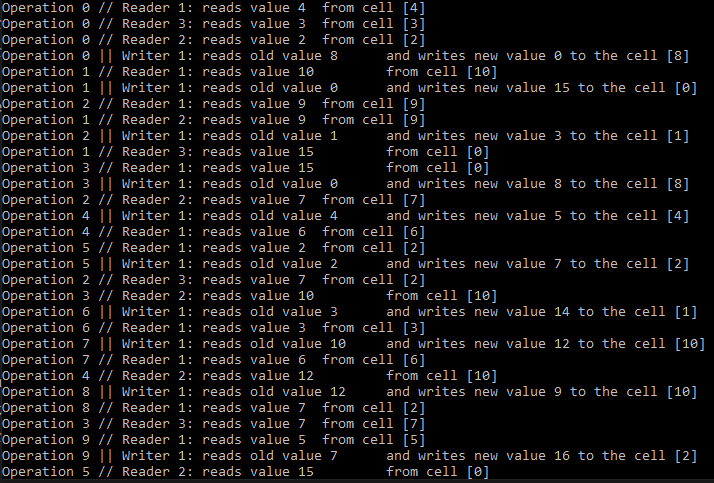
## **Описание программы**

Базу данных разделяют два типа потоков – читатели и писатели. Читатели выполняют транзакции, которые просматривают записи базы данных, транзакции писателей и просматривают, и изменяют записи. Предполагается, что в начале БД находится в непротиворечивом состоянии (т. е. отношения между данными имеют смысл). Транзакции выполняются в режиме «грязного чтения», то есть процесс-писатель не может получить доступ к БД только в том случае, если ее занял другой процесс-писатель, а процессы-читатели ему не мешают. Создать многопоточное приложение с потоками-писателями и потоками-читателями. Реализовать решение, используя семафоры, и не используя блокировки чтения-записи

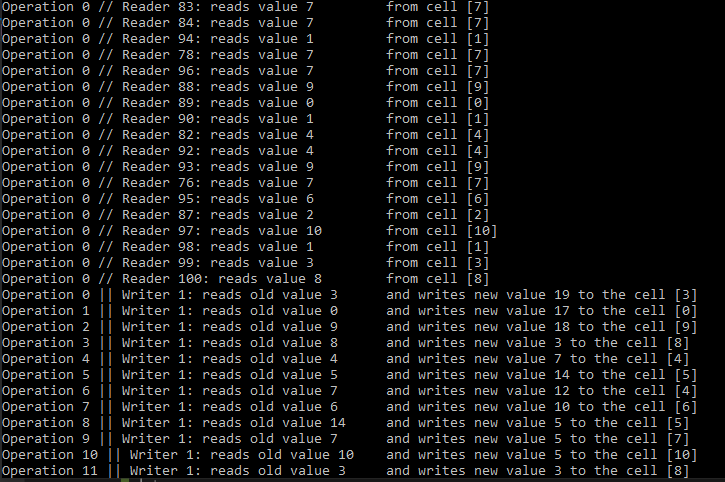
# **Тестирование**



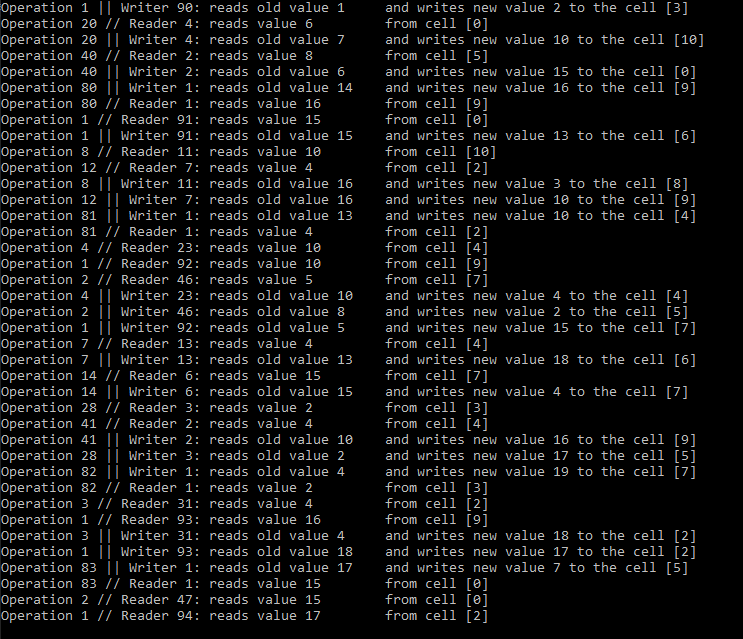
*(10 читателей, 5 писателей, по 100 операций у каждого)*



*(3 читателя, 1 писатель, по 10 операций у каждого)*



*(100 читателей, 1 операция, 1 читатель, 100 операций)*



100 писателей, 100 читателей, 100 операций у каждого

# **Текст программы**

#define HAVE\_STRUCT\_TIMESPEC

#define \_CRT\_SECURE\_NO\_WARNINGS

#define \_CRT\_RAND\_S

#define RAND\_MAX

#include <stdio.h>

#include <semaphore.h>

#include <thread>

#include <iostream>

#include <random>

using namespace std;

mt19937 mersenne(static\_cast<unsigned int>(time(0)));

int database[11] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

sem\_t semaphore;

void dataWriter(int threadNum) {

int data;

int id;

int result;

for (int i = 0; i < 100; i++) {

sem\_wait(&semaphore);

data = mersenne() % 20;

id = mersenne() % 11;

result = database[id];

database[id] = data;

printf("Operation %d || Writer %d: reads old value %d \tand writes new value %d to the cell [%d]\n"

, i, threadNum, result, data, id, result);

sem\_post(&semaphore);

this\_thread::sleep\_for(chrono::seconds(threadNum));

}

}

void dataReader(int threadNum) {

int result;

int id;

for (int i = 0; i < 100; i++) {

id = mersenne() % 11;

result = database[id];

printf("Operation %d // Reader %d: reads value %d \tfrom cell [%d]\n", i, threadNum, result, id);

this\_thread::sleep\_for(chrono::seconds(threadNum));

}

}

int main() {

const int countReader = 100;

const int countWriter = 100;

sem\_init(&semaphore, 0, 1);

thread writers[countReader];

thread readers[countReader];

for (int i = 0; i < countReader; i++)

readers[i] = thread(dataReader, i + 1);

for (int i = 0; i < countWriter; i++)

writers[i] = thread(dataWriter, i + 1);

for (int i = 0; i < countWriter; ++i)

writers[i].join();

for (int i = 0; i < countReader; ++i)

readers[i].join();

sem\_destroy(&semaphore);

return 0;

}

# **Формат вывода**

В начале строки выводится номер операции у данного типа и номера потока, далее определяется тип потока (“Writer” или “Reader”), а также его порядковый номер. После, в зависимости от типа, строка продолжается либо что и откуда он прочитал (это в случае, если тип потока “Reader”), либо что и откуда прочитал, а после на что заменил (это в случае, если тип потока “Writer”). Порядковый номер писателя или читателя – это номер потока, под которым он лежит.

Опишу по отдельности:

Строка потока *читателя*

1. Operation {Номер операции}
2. // (Выводится для более удобного форматирования строк зрительно)
3. Reader {Номер потока в массиве потоков типа Reader}
4. reads value {Значение читаемое из ячейки}
5. from cell [{индекс в БД данной ячейки, из который читатель читал}]

Строка потока *писателя*

1. Operation {Номер операции}
2. || (Выводится для более удобного форматирования строк зрительно)
3. Writer {Номер потока в массиве потоков типа Writer}
4. reads old value {Значение читаемое из ячейки}
5. and writes new value {Новое задаваемое значение в ячейку БД}
6. to the cell [{индекс в БД данной ячейки, из который читатель читал}]

# **Список литературы**

1. https://www.youtube.com/watch?v=4GdTcqE0iqg&feature=emb\_logo&ab\_channel=davidko
2. https://ravesli.com/urok-71-generatsiya-sluchajnyh-chisel-funktsii-srand-i-rand/#toc-5
3. https://learnc.info/c/pthreads\_semaphores.html
4. ftp://sourceware.org/pub/pthreads-win32