**4 ЛАБОРАТОРНАЯ РАБОТА №4**

**«Синхронизация процессов и потоков в ОС Windows.**

**Семафоры.»**

**4.1 Цель работы**

Изучение программного интерфейса приложений (API) операционных систем Windows 9x, NT, ME, XP, 7. Приобретение практических навыков синхронизации потоков, с использованием семафоров в средах программирования Borland Delphi, C++ Builder или Visual Studio.

**4.2 Вариант задания – 17**

4.2.1 Написать программу, содержащую два потока. Первый поток генерирует последовательность чисел и помещает их в кольцевой\* буфер из Nbuf элементов (с проверкой на свободное место в буфере с использованием механизма семафоров). Второй считывает данные из буфера и выводит их на экран. \*) при заполнении кольцевого буфера добавление элементов продолжается сначала, т.е. для вычисления индекса очередного элемента используется операция деления по модулю Nbuf(Pascal: i mod Nbuf; Cи: I % Nbuf).

4.2.2 Определить максимальную длину буфера при различных приоритетах оттоков.

**4.3 Ход работы**

4.3.1 Написана программа на языке си и её текст представлен ниже.

3.3.1.1 Программа Sort3.c

#

4.3.2 Тестовые примеры продемонстрированы на рисунках 3.1 – 3.4.

Были написаны основная программа Master3 и дополнительная программа Sort3, запускаемая основной программой как дополнительный процесс. После запуска было обнаружено, что выводы 3 процессов «перемешиваются» друг с другом (Рисунок 3.1).

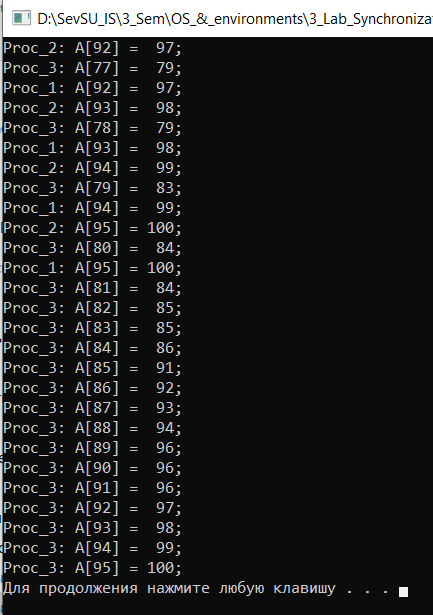


Рисунок 3.1 ­– Вывод программы Master3 (без мьютексов)

После чего программы были переписаны с использованием мьютексов. После запуска Master3\_mtx на экран были выведены данные, не «перемешанные» между собой, что продемонстрировано на рисунке 3.2.

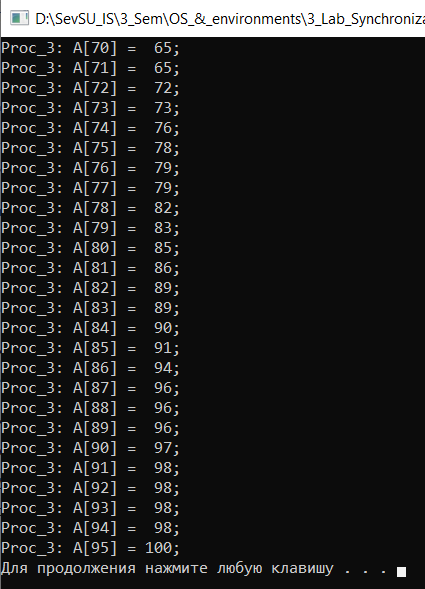


Рисунок 3.2 ­– Вывод программы Master3\_mtx (с использованием мьютексов)

Была написана программа, представленная в файле Master3\_crit.c. После запуска было обнаружено, что данные «перемешиваются» при выводе на экран (Рисунок 3.3). После чего программа была переписана с использованием критических секций, скомпилирована и запущена. Данные были выведена на экран, не «перемешавшись», что продемонстрировано на Рисунке 3.4.

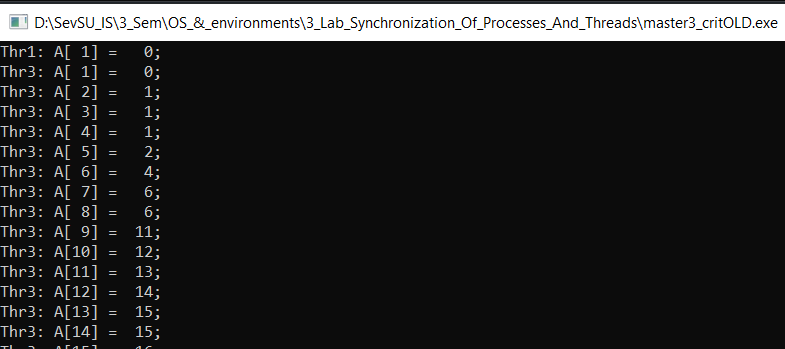


Рисунок 3.3 – Вывод программы без использования критических секций

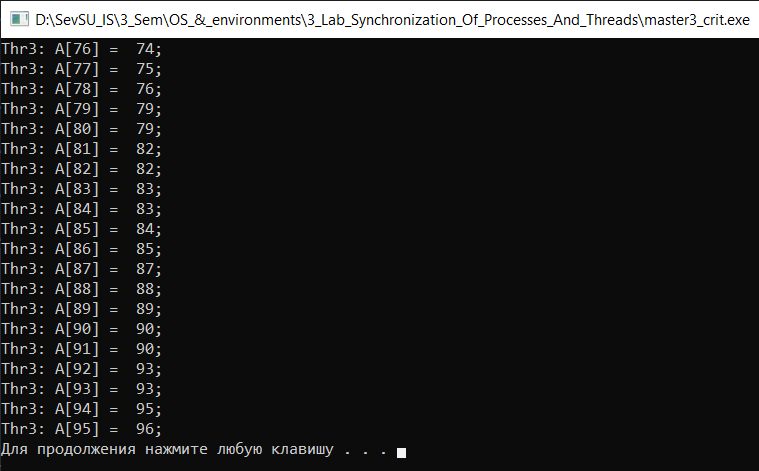


Рисунок 3.3 – Вывод программы с использованием критических секций

Результаты тестирования полностью соответствуют ожиданиям.

**Выводы**

При выполнении данной лабораторной работы были получены навыки разработки программ с мьютексами и критическими секциями с помощью средств WinAPI. Также были повторно закреплены навыки синхронизации процессов и потоков с помощью функции WaitForSingleObject. Полученные знания позволят создавать многопоточные приложения, которые на многоядерных системах будут работать эффективнее.