Кафедра прикладной математики и кибернетики

Выполнили:

Студенты 3 курса группы ИП-111  
Корнилов А.А.,  
Попов М.И.,

Толкач А.А.

Проверил:

Профессор кафедры ПМиК  
Малков Е.А.

ЛАБОРАТОРНАЯ РАБОТА №11

По дисциплине: «Операционные системы»

Сибирский государственный университет  
телекоммуникаций и информатики

Новосибирск, 2023

Министерство цифрового развития, связи  
и массовых коммуникаций Российской Федерации

**Задание:** реализуйте алгоритм “производитель-потребитель” для конечного буфера и циклического буфера.

**Цель:** получение навыков синхронизации с использованием мьютексов и семафоров.

**Выполнение работы:**

Была написана программа генерации и вывода чисел для конечного буфера (Листинг 1) и бесконечного буфера (Листинг 2)

|  |
| --- |
| #include <iostream>  #include <queue>  #include <thread>  #include <mutex>  #include <condition\_variable>  using namespace std;  const int buffer\_size = 5; // Размер буфера  queue<int> buffer; // Очередь для буфера  mutex mtx; // Мьютекс для синхронизации доступа к буферу  condition\_variable buffer\_empty; // Условная переменная для оповещения о пустом буфере  condition\_variable buffer\_full; // Условная переменная для оповещения о полном буфере  // Функция производителя  void producer(int id) {  for (int i = 0; i < 10; ++i) {  unique\_lock<mutex> lock(mtx);  // Проверка, если буфер полон, ждем, пока не появится место  buffer\_full.wait(lock, [] { return buffer.size() < buffer\_size; });  // Генерация элемента  int item = rand() % 100;  cout << "Изготовитель №" << id << " придумал число: " << item << endl;  // Добавление элемента в буфер  buffer.push(item);  // Оповещаем потребителя о наличии нового элемента  buffer\_empty.notify\_all();  }  }  // Функция потребителя  void consumer(int id) {  for (int i = 0; i < 10; ++i) {  unique\_lock<mutex> lock(mtx);  // Проверка, если буфер пуст, ждем, пока не появится элемент  buffer\_empty.wait(lock, [] { return !buffer.empty(); });  // Извлечение элемента из буфера  int item = buffer.front();  buffer.pop();  cout << "Потребитель №" << id << " получил число: " << item << endl;  // Оповещаем производителя о наличии места в буфере  buffer\_full.notify\_all();  }  }  int main() {  // Создание потоков производителей и потребителей  thread producer1(producer, 1);  thread producer2(producer, 2);  thread consumer1(consumer, 1);  thread consumer2(consumer, 2);  // Ожидание завершения потоков  producer1.join();  producer2.join();  consumer1.join();  consumer2.join();  return 0;  } |

Листинг 1 – программа lab11\_1.c

|  |
| --- |
| #include <iostream>  #include <queue>  #include <thread>  #include <mutex>  #include <condition\_variable>  const int bufferSize = 5; // Размер буфера  std::queue<int> buffer; // Очередь буфера  std::mutex mtx; // Мьютекс для обеспечения безопасности доступа к буферу  std::condition\_variable producerCV, consumerCV; // Условные переменные для синхронизации производителя и потребителя  void producer() {  for (int i = 0; ; ++i) {  std::unique\_lock<std::mutex> lock(mtx);  producerCV.wait(lock, []() { return buffer.size() < bufferSize; }); // Ждем, пока буфер не станет меньше максимального размера  buffer.push(i);  std::cout << "Производитель произвел: " << i << std::endl;  lock.unlock();  consumerCV.notify\_one(); // Уведомляем потребителя  }  }  void consumer() {  for (;;) {  std::unique\_lock<std::mutex> lock(mtx);  consumerCV.wait(lock, []() { return !buffer.empty(); }); // Ждем, пока буфер не станет непустым  int data = buffer.front();  buffer.pop();  std::cout << "Потребитель потребил: " << data << std::endl;  lock.unlock();  producerCV.notify\_one(); // Уведомляем производителя  }  }  int main() {  std::thread producerThread(producer);  std::thread consumerThread(consumer);  producerThread.join();  consumerThread.join();  return 0;  } |

Листинг 2 – программа lab11\_2.cpp