**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФГБОУ ВО «БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра «Информатика и программное обеспечение»

**Расчетно-графическая работа**

По дисциплине «Операционные системы»

**Тема:**Решение проблемы производителя и потребителя с помощью мониторов

Руководитель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ к.т.н., доц. Гулаков К.В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г.

Студент гр. О-18-ПРИ-РПС-Б

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Лебедев В.В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г.

**БРЯНСК 2020**

Содержание

[ВВЕДЕНИЕ 2](#_Toc41587482)

* 1. [Проблема производителя и потребителя 2](#_Toc41587483)
  2. [Решение с помощью монитора 2](#_Toc41587484)

[1. ОПИСАНИЕ АЛГОРИТМА 3](#_Toc41587485)

* 1. [Общее описание алгоритма 3](#_Toc41587486)
  2. [Детальное описание алгоритма 3](#_Toc41587487)
  3. [Листинг программы 3](#_Toc41587488)

[2. ДЕМОНСТРАЦИЯ РАБОТЫ 4](#_Toc41587489)

[ЗАКЛЮЧЕНИЕ 5](#_Toc41587490)

[СПИСОК ЛИТЕРАТУРЫ 6](#_Toc41587491)

# Введение

## Цель работы

Целью работы является изучение проблемы производителя и потребителя, а также реализация решения данной проблемы с помощью мониторов.

## Проблема производителя и потребителя

Проблема производителя и потребителя (producer-consumer) является одной из задач на организацию взаимодействия процессов.

Пусть два процесса обмениваются информацией через буфер ограниченного размера. Производитель закладывает информацию в буфер, а потребитель извлекает ее оттуда. Проблема заключается в том, что буфер ограничен и производитель не может поместить информацию в заполненный буфер, а потребитель не может взять информацию из пустого.

Решением для производителя является ожидание освобождения буфера, а для потребителя – ожидание заполнения

## Решение с помощью монитора

Существует несколько способов решения данной проблемы, один из них — это решение с помощью мониторов.

Монитор – это набор процедур, переменных и других структур данных, объединённых в особый модуль или пакет. Процессы могут вызывать процедуры монитора, но у процедур, объявленных вне монитора, нет прямого доступа к внутренним структурам данных монитора. При обращении к монитору в любой момент времени активным может быть только 1 процесс. Обычно при вызове процедуры монитора первые несколько команд процедуры проверяют, нет ли в мониторе активного процесса. Если таковой есть, вызывающему процессу придется подождать, в противном случае запрос удовлетворяется. Преимуществом данного решения является то, что взаимное исключение обеспечивает компилятор. Обеспечив попадание в критические области через процедуры монитора, можно не бояться нахождения в критических областях двух процессов одновременно.

# Описание Алгоритма

## Общее описание алгоритма

Если буфер заполнен, то производитель должен ждать, пока в нем появится место, чтобы положить туда новую порцию информации. Если буфер пуст, то потребитель должен дожидаться нового сообщения

Монитор содержит условные переменные full и empty для блокировки процессов, которые обращаются к монитору. Данные условные переменные обеспечивают активность только одного процесса, обращающихся к монитору. Методы put и get, описанные в мониторе позволяют безопасно извлекать и записывать информацию в буфер. Любые взаимодействия с буфером происходят с помощью этих переменных

Производитель генерирует данные и записывает и в буфер с помощью метода put

Потребитель считывает данные из буфера с помощью метода get

## Детальное описание алгоритма

Производитель:

* Генерирует данные и подготавливает их к записи
* Вызывает метод монитора put для записи в буфер

Потребитель

* Вызывает метод монитора get для получения данных
* Обрабатывает эти данные

Метод get

* Проверяет наличие элементов в буфере, если элементов нет, то вызывает empty.wait() (wait(empty))
* Извлекает данные
* Если извлек данные из полного буфера вызывает full.notify\_one() (signal(full))

Метод put

* Проверяет на сколько заполнен буфер, если буфер заполнен, то вызывает full.wait() (wait(full))
* Записывает данные в буфер
* Если записал данные в пустой буфер, то вызывает empty.notify\_one() (signal(empty))

Блок-схема

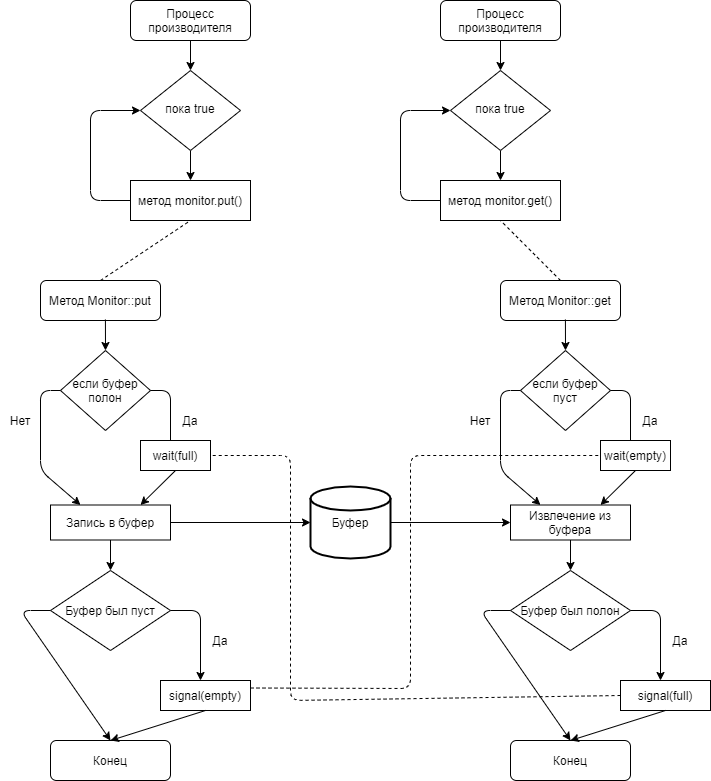


Рисунок 1 Блок-схема

## Листинг программы

Функция Producer

|  |
| --- |
| void Producer(void\* number)  {  srand(time(0));  while (flag)  {  int data\_new = rand() % 10 ;  monitor.put(data\_new);  int n = rand() % 3 + 1;  Sleep(n \* 100);  }  } |

Листинг 1 функция Producer

Функция Consumer

|  |
| --- |
| void Consumer(void\* number)  {  srand(time(0));  while (flag)  {  int data\_;  monitor.get(data\_);  int n = rand() % 3 + 1;  Sleep(n \* 100);  }  } |

Листинг 2 функция Consumer

Класс Monitor

|  |
| --- |
| class Monitor  {  public:  Monitor() {count = 0;}  void put(int \_data);  void get(int& \_data);  private:  int data[BufSize];  int count;  condition\_variable full, empty;  mutex m;  };  void Monitor::put(int \_data)  {  unique\_lock<mutex>lock(m);  if (count >= BufSize)  {  full.wait(lock);  }  data[count] = \_data;  cout << endl << "Producer put: " << \_data << endl;  count = count + 1;  if (count == BufSize) cout << "Buffer is full" << endl;    cout << "count: " << count << endl;  cout << "Buffer: ";  for (int i = 0; i < count; i++)  {  cout << data[i] << " ";  }  cout << endl << endl;  if(count==1) empty.notify\_one();  }  void Monitor::get(int& \_data)  {  unique\_lock<mutex>lock(m);    if (count == 0)  {  empty.wait(lock);  }  \_data = data[count-1];  cout << endl << "Consumer get: " << \_data << endl;  count = count - 1;  if (count == 0) cout << "Buffer is empty" << endl;  cout << "count: " << count << endl;  cout << "Buffer: ";  for (int i = 0; i < count; i++)  {  cout << data[i] << " ";  }  cout << endl << endl;  if(count == BufSize-1) full.notify\_one();  } |

Листинг 3 класс Monitor

# Демонстрация работы

## Описание программы

Программа представляет собой консольное приложение, в котором запускаются два потока и выполняют функции Producer и Consumer, которые помещают информацию в буфер или извлекают ее, после каждого помещения/извлечения выводится следующая информация:

* Значение помещенного/извлеченного элемента данных
* Кол-во заполненных ячеек буфера
* Все значения, записанные в буфер
* Буфер пуст или полон (если элементы есть, но буфер не полон, то ничего не выводится)

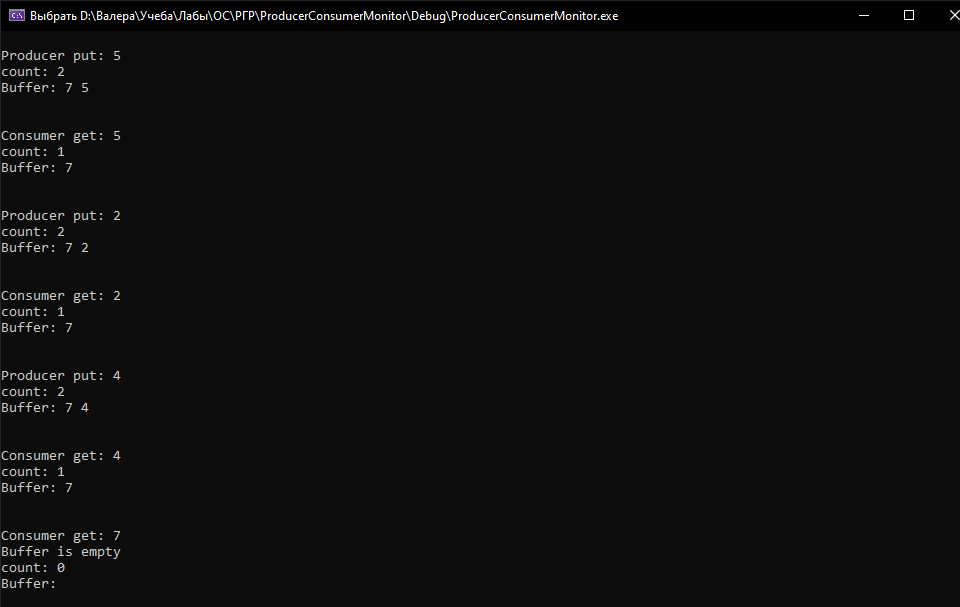


Рисунок 2 Результат работы программы

# Заключение

Использование мониторов очень удобный способ решения проблемы производителей и потребителей. Данные метод позволяет избежать множества ошибок, а также гарантирует автоматическое взаимное исключение, реализуемое процедурами монитора.

В результате работы была изучена проблема читателей и писателей, а также ее решение с помощью мониторов, была написана программа, демонстрирующая данное решение. Были получены знания о работе потоков и их взаимодействия, а также знания о том, как можно использовать мониторы для синхронизации процессов

Ссылка на GitHub с проектом:

<https://github.com/ValeriusPajiloy/RGR_OS_Lebedev.git>

# Список литературы

1. Руссинович М. Внутреннее устройство Microsoft Windows, 6-е издание. / Д.Соломон, М. Руссинович. — СПб.: Питер, 2016. – 800 С.
2. Олифер В.Г., Олифер Н.А. Сетевые операционные системы: Учебник для вузов. 2-е изд. / В.Г. Олифер, Н.А. Олифер. – СПб.: Питер, 2016. – 669 с.
3. Дейтел Х., Дейтел П., Чофнес Д.Р. Операционные системы. Основы и принципы [пер. с англ.] / Х. Дейтел, П. Дейтел, Д.Р. Чофнес– М.: БиномПресс, 2017. – 1204 с.
4. Таненбаум Э., Современные операционные системы [пер. с англ.] / Э.Таненбаум. – СПб.: Питер, 2016. – 1120 с.
5. А. Ю. Молчанов Системное программное обеспечение: /. - СПб.; М.; Нижний Новгород: Питер: Питер принт, 2017. - 395 с.
6. Албахари Дж., Албахари Б., «C# 7.0. Карманный справочник.» / Албахари Дж., Албахари Б. – СПб.: Питер, 2017. – 224 с.
7. Официальный сайт “Microsoft” [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-/dotnet/csharp/progra.. свободный (26.05.2020)