ДОМАШНЕЕ ЗАДАНИЕ

Первая военная задача

ХАН РОМАН БПИ207

ВАРИАНТ 28

Первая военная задача

ОПИСАНИЕ ЗАДАНИЯ (вариант 28):

Для полного описания задачи приведу условие, которое написано в файле с задачей:

Темной-темной ночью прапорщики Иванов, Петров и Нечепорчук занимаются хищением военного имущества со склада родной военной части. Будучи умными людьми и отличниками боевой и строевой подготовки, прапорщики ввели разделение труда. Иванов выносит имущество со склада, Петров грузит его в грузовик, а Нечепорчук 9 подсчитывает рыночную стоимость добычи. Требуется составить многопоточное приложение, моделирующее деятельность прапорщиков. При решении использовать парадигму «производитель-потребитель».

В сущности, требуется разработать многопоточное консольное приложение на языке программирования C, организация потоков в котором соответствует модели “производитель – потребитель” *(стиль написания – произвольный)*. Исходя из условия, можно выделить 4 потока, которые должны быть реализованы:

Название потока	**Описание**
Главный поток	В этом потоке выполняется функция main, из которой и стартует приложение. Инициализируется буфер склада исходя от введенных параметров.
Поток Иванова	Происходит удаление объектов из буфера склада и перемещение их в буфер рук Иванова.
Поток Петрова	Происходит удаление объектов из буфера рук Иванова и перемещение их в буфер грузовика.
Поток Нечепорчука	Добавляет цену вновь пребывших объектов к сумме, считая таким образом суммарную стоимость вывезенного имущества.

Также нужно: разработать тестовые входные данные и провести тестирование и отладку программы на этих данных (при необходимости, программа должна правильно обрабатывать переполнение по данным); описать подробно описать используемую модель вычислений; зафиксировать количество заголовочных файлов, программных файлов, общий размер исходных текстов, полученный размер исполняемого кода (если он формируется), время выполнения программы для различных тестовых наборов данных.

ОПИСАНИЕ ВЗОИМОДЕЙСТВИЯ С ПРОГРАММОЙ

Ввод данные происходит через строку аргументов при запуске программы. В сущности, у программы должно быть 2 режима работы: генерация склада с рандомным количеством объектов и рандомной ценой и заполнение склада данными из аргументов командной строки. Формат команд следующий:

- Для ввода информации об имуществе из аргументов:
  -f цена1, цена2, цена3, …, ценаN
  Где -f – обозначающий флаг, а ценаI – цена i-го объекта на складе.
  Пример:
  -f 1 2 3 4 5 – в программе будет создан склад с 5-ю объектами, цены которых следующие – 1, 2, 3, 4, 5
- Для рандомной генерации склада:
  -n N
  Где -n – обозначающий флаг, а N – целое число, обозначающее количество элементов на складе.
  Пример:
  -n 100 – будет сгенерирован склад, содержащий 100 объектов, цены на которые варьируются в диапазоне от 0 до 1000

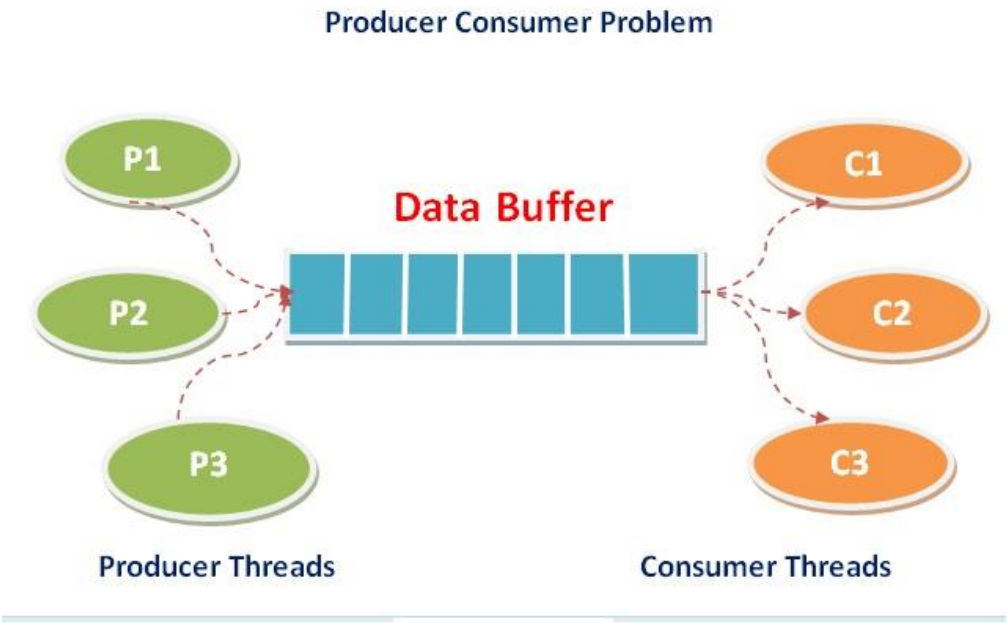
Стоит отметить, что в программе существует ограничение на количество объектов на складе – их не может быть больше 10000.

ОПИСАНИЕ ИСПОЛЬЗОВАННОЙ МОДЕЛИ ВЫЧИСЛЕНИЙ:

В презентации в лекции (и в учебном пособии “СРЕДСТВА РАЗРАБОТКИ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ” от Сибирского Федерального Университета соответственно) дано следующее определение модели “ производитель-потребитель”:

Производители и потребители – это парадигма взаимодействующих неравноправных процессов. Одни из процессов «производят» данные, другие их «потребляют». Часто такие процессы организуются в *конвейер*, через который проходит информация. Каждый процесс конвейера потребляет выход своего предшественника и производит входные данные для своего последователя. Другой распространенный способ организации потоков – древовидная структура, на этом основан, в частности, принцип *дихотомии*.

На деле, в данной модели все потоки разделяются на 2 категории – производители и потребители. Производители добавляют данные в буфер, а потребители – как-либо их обрабатывают (либо удаляют из буфера). Буфером же является некоторое хранилище данных (например, массив). Схематично модель можно представить следующим образом:



Взято с сайта (https://teachtojava.blogspot.com/2016/08/producer-consumer-problem-in-java.html)

Существуют 2 различные разновидность данной модели – с ограниченным буфер и без. Модели с неограниченным буфером менее подвержены проблемам с многопоточность, но при этом трудно реализуемы с практической точки зрения (нельзя говорить о бесконечной памяти, работая с физическим устройством, пространство памяти которого в принципе конечна). Модель же с ограниченным буфером сталкивается с рядом проблем (взято из видео https://www.youtube.com/watch?v=l6zkaJFjUbM&t=1151s):

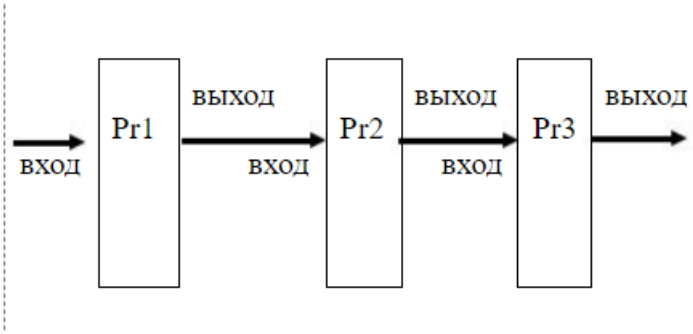
- Производитель пытается положить данные в полностью заполненный буфер. Если такое происходит, то это приводит к утечке данных либо тех, которые уже есть в буфере, либо тех, которые сгенерировал производитель. Следовательно, надо всегда проверять возможность добивать данные в буфер, прежде чем сделать это.
- Потребитель пытается достать данные из пустого буфера. Противоположная предыдущему пункту проблема. Здесь не происходит потеря данных, однако такая ситуация может привести к непредсказуемому поведению программы, когда мы обращаемся к неинициализированным ячейкам памяти, которые в целом могут быть зарезервированы под какой-то служебный процесс (следовательно, доступ к ним закрыт и программа упадет), либо содержать мусор. Для этих целей в потоке потребителя всегда нужно проверять на возможность достать элемент из буфера, прежде чем делать это.

- Наличие общей памяти. Буфер является общей памятью для производителей и потребителей, что приводит к тому, что одна область данных доступна сразу нескольким потокам, очередность выполнения которых предсказать в принципе невозможно. Это приводит к некорректной обработке неатомарных операций (таких, как инкремент, например), а так как неатомарных операций в программировании на самом деле много, то это становится большой проблемой в разработке приложения. Для совершения операций с общей памятью возникает необходимость синхронизации потоков. Обычно в этих целях применяют mutex объекты (блокирование доступа только за одним потоком) или семафор (похож на mutex с отличием, что доступ можно выдавать сразу нескольким потокам).

Если обратиться к вышеупомянутому пособию Сибирского Федерального Университета, то в нем же поясняется понятие организации потоков в конвейер:

> “ *конвейер* – последовательность процессов, в которой каждый потребляет данные предшественника и поставляет данные для последующего процесса”

К этому определению приложена иллюстрация, которая дает более наглядное понимание вышесказанного:



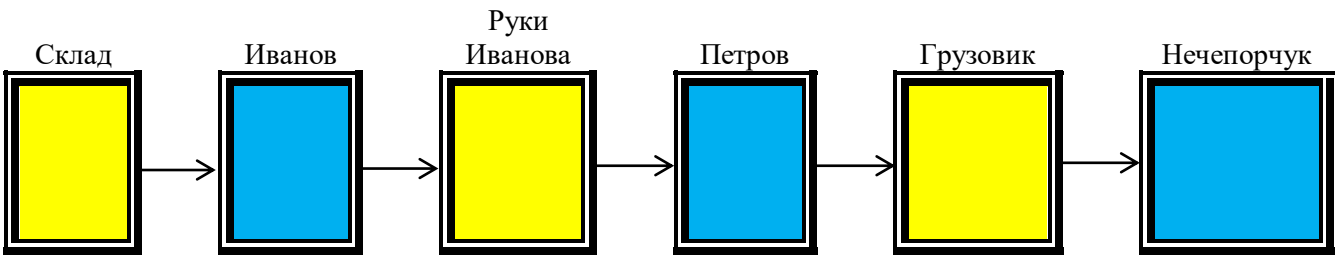
Для упрощения здесь опущено изображение промежуточных буферов, которые существуют для связей между потоками. Стоит также отметить, что связи в конвейере однонаправленные, то есть производитель может выдавать данные потребителю, но потребитель как-либо влиять на работу производитель – нет.

Из вышеприведенной схемы и данного определения можно сделать вывод, что модели “производитель-потребитель” соответствует наличие в программе несколько буферов, для которых есть как производители, так и потребители, притом производитель для одного буфера может быть потребителем другого. Именно эти рассуждения я и использовал для реализации программы в рамках текущего задания.

В качестве примера использования данной модели приведу пример текущего задания. По условию задания каждый из прапорщиков выполняет строго свою функцию, разбивая общую задачу (своровать имущество со склада) на компоненты. Логично в данном случае применить конвейерную реализацию, в которой каждый поток-человек готовит данные для последующего. В то же время появилось разделение и для 3-х буферов, которые выступают как источниками данных, так и связками между потоками.

Так, Иванов берет имущество из буфера-склада, Петров забирает имущество с рук Иванова и перекладывает его в буфер грузовика, Нечепорчук же отслеживает состояние грузовика и обновляет суммарную рыночную цену, если в грузовике прибавилось имущество. Иными словами: Иванов - потребитель для буфера склада, но производитель для буфера своих рук, Петров – потребитель для буфера рук Иванова, но производитель для буфера грузовика, Нечепорчук – только потребитель для буфера грузовика.

Схематично схему изобразить можно следующим образом:



Здесь желтым цветом изображены буферы, а синим – потоки, которые берут из них данные (или добавляют).

Все потоки использую mutex объект для синхронизации в действиях с счетчиками буферов, что снижает скорость выполнения, однако в данной модели иначе обойтись нельзя.

Ссылка на учебное пособие - https://studfile.net/preview/4419687/page:3/

ОСНОВНЫЕ ХАРАКТЕРИСТИКИ ПРОГРАММЫ:

- Число заголовочных файлов – 2 (buffer.h, print.h)
- Число модулей реализации – 2
- Общий размер исходных текстов – 248 строк кода вместе с комментариями (6.91 кб).
- Размер исполняемого файла – 24.4 кб.
- Время выполнения программы для различных тестовых прогонов:

Номер теста	Время выполнения в секундах
Тест1 (ручной ввод)	0.011903
Тест2 (рандомная генерация)	0.031368
Тест3 (пустой ввод)	0.000055
Тест4 (неверный флаг)	0.000085
Тест5 (некорректные аргументы)	0.000081
Тест6 (больше 10000)	0.000079
Тест7 (10000 элементов)	13.578694