Пояснительная записка

Проломов Виталий Алексеевич БПИ208

Вариант 25

1. Постановка задачи

Задача о производстве булавок. В цехе по заточке булавок все необходимые операции осуществляются тремя рабочими. Первый из них берет булавку и проверяет ее на предмет кривизны. Если булавка не кривая, то рабочий передает ее своему напарнику. Напарник осуществляет собственно заточку и передает заточенную булавку третьему рабочему, который осуществляет контроль качества операции. Требуется создать многопоточное приложение, моделирующее работу цеха. При решении использовать парадигму «производитель-потребитель

1. Особенности моей реализации

В моей реализации Рабочие представлены потоками – каждый со своей задачей и со своими особенностями поведения.

Рабочий 1 «проверяет булавку на дефекты» - и с вер-тью 0.01 выбрасывает булавку (в следствие ее непригодности), иначе передает ее 2ому рабочему.

2ой рабочий всегда передает заточенную булавку 3ему рабочему, но если он не находит под рукой булавки (кол-во еще не заточенных булавок, проверенных 1ым рабочим), то он «отвлекается» и засыпает на 3 мс.

3ий рабочий проверяет качество заточки булавок и с вер-тью 4/100 забраковывает ее. 3ий рабочий, как и 2ой, засыпает на некоторое время, если не находит под рукой новую булавку (заточенную, но еще не оцененную) – но он спит 5 мс.

Сон рабочих также предотвращает работу потока, для которого в данный момент нет работы, что позволяет добиться лучшей производительности

Отдельного класса для булавки я не вводил, тк нет достаточного кол-ва инф-ии для того, чтобы хранить ее в классе. Хранится лишь кол-во булавок на каждом шаге производства в специальных переменных.

Как и просят в задании, используется модель «производитель-потребитель» 1ый рабочий является производителем для 2ого и 3его рабочего, они получают инф-ию от него в виде изменения кол-ва проверенных булавок.

В добавок, 2ой рабочий явл-ся рабочим для 3его рабочего.

Мною были использованы std::atomic переменные, поскольку у нескольких рабочих присутствует доступ к одним и тем же переменным, которые они изменяют (пример – доступ к кол-ву проверенных, но не заточенных булавок есть как у 1ого, так и у 2ого рабочего. И из-за того, что операция инкремента не является атомарной (происходит в 3 действия), могут возникнуть ошибки в изменении таких переменных. Std::atomic как раз решает эту проблему, обязывая проводить все действия с этой такой переменной за один раз (в один момент с такой переменной может работать лишь 1 поток).

1. Ввод

*Корректный ввод*

Ввод производится путем передачу единственного параметра X при запуске программы – кол-ва булавок, которые рабочие должны обработать. Если X > 0, то генерируется Х булавок.

Если X == 0, то кол-во булавок генерируется случайным образом – в диапазоне от 1 до 10000 (ограничение сделано для информативности вывода – если булавок будет больше, в консоль выводится слишком много инф-ии для восприятия, а также не всегда весь вывод помещается в консоль).

*Некорректный ввод*

Если X < 0, то программа прерывается и выводит соотв. сообщение об ошибки.

Если Х – не число, программа также приостанавливается.

Если параметров больше или меньше, программа прерывается.

1. Вывод

Весь вывод осуществляется в консоль.

Сперва пишется какой именно рабочий сообщает инф-ии, затем непосредственно его сообщение.

1. Уточнение

У меня в argv всегда нулевым параметром стоит расположение проекта, так что и в программе я проверяю условие (argc == 2). Если вдруг на других ОС это не так, то следует также добавить еще 1 параметр в начале – расположение проекта.

1. Анализ особенностей работы с многопоточной программой

Программа, использующая несколько потоков работает быстрее при больших кол-вах данных, что является ее несомненным плюсом. Однако, необходимо следить за тем, чтобы все работало корректно и не возникало типичных ошибок, связанных с работой многопоточной программы – например описанная мною выше (с неатомарными операциями).

Также (нельзя однозначно – плюс или минус) процесс выполнения многопоточной программы не является детерминированным – нельзя с уверенностью утверждать, в какой последовательности выполнятся все указанные инструкции потоков.

1. Ссылки на источники по изучению модели
2. Видео объяснение модели <https://www.youtube.com/watch?v=VpI4XkpnQok>
3. Статья, объясняющая основные концепты модели <https://coderlessons.com/articles/java/model-parallelizma-proizvoditel-i-potrebitel>
4. Пример кода

<https://russianblogs.com/article/6801113920/>

1. Скелет программы

<https://it.wikireading.ru/hDTnL35jrG>

1. Форум с некоторыми уточнениями и вопросами   
   <https://coderoad.ru/1371249/C-%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C-%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C-%D0%BF%D0%BE%D1%82%D1%80%D0%B5%D0%B1%D0%B8%D1%82%D0%B5%D0%BB%D1%8C>
2. Решения возможных проблем с запуском

Я крайне надеюсь, что все отработает как надо, но при скачивании проекта с гитхаба я столкнулся с некоторыми проблемами.

Сначала я разархивировал проект в загрузки, а ее путь содержит папку с названием на русском языке (и папку не могло найти, тк вместо русского текста были иероглифы)

Затем мне пришлось вручную добавить конфигурацию выполнения программы ( CMake Application).

Опять же, верю что все запустится, но на всякий случай добавлю это сюда.