НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии
Дисциплина: «Архитектура вычислительных систем»

ДОМАШНЕЕ ЗАДАНИЕ №4. ВАРИАНТ 17

Отчёт

Выполнил:
студент группы БПИ198

Лямзин Дмитрий

Москва
2020

1. Описание задачи

Номер варианта: 17

Условие задачи: Задача об инвентаризации по книгам. После нового года в библиотеке университета обнаружилась пропажа каталога. После поиска и наказания, виноватых ректор дал указание восстановить каталог силами студентов. Фонд библиотека представляет собой прямоугольное помещение, в котором находится M рядов по N шкафов по K книг в каждом шкафу. Требуется создать многопоточное приложение, составляющее каталог. При решении задачи использовать метод «портфель задач», причем в качестве отдельной задачи задается внесение в каталог записи об отдельной книге.

2. Используемая модель вычислений

2.1. Метод “портфель задач”

Для использования метода “портфель задач” в качестве модели решения была выбрана парадигма параллельного программирования “взаимодействующие равные” [1], в которой исключен не занимающийся непосредственными вычислениями управляющий поток. Портфель задач был реализован с помощью разделяемой переменной “task_number”, в которой хранился целочисленный номер задачи, которую должен был выполнить поток, получивший доступ к переменной. Сразу после того, как поток получал номер задачи, значение переменной увеличивалось на 1, после чего выполнялась задача, и доступ к портфелю мог быть передан следующему процессу. Выполнение задач различными процессами производилось до тех пор, пока значение “task_number”, изначально равное 0, не становилось равно значению переменной “max_task_number”, хранящей общее число задач, которое необходимо решить (т.е. количество книг, информацию о которых необходимо занести в каталог).

2.2. Алгоритм решения задачи

В качестве входных данных программа принимает целые числа M, N, K, обозначающие количество рядов, шкафов в каждом ряду и книг в каждом шкафу соответственно. Значения переменных должны быть целыми числами от 1 до 100 включительно, и, если это условие не выполняется, программа просит пользователя ввести новые значения до тех пор, пока они не станут корректными. Затем для каждой из N*M*K, хранящихся в библиотеке с помощью генератора псевдослучайных чисел (метод “rand()”) генерируется идентификатор из диапазона от 1 до 100 включительно. Номер ряда, номер шкафа в ряду и номер книги в шкафу назначаются каждой книге в момент генерации идентификаторов в порядке возрастания. Таким образом, присвоение идентификаторов начинается с первой книги первого шкафа в первом ряду и заканчивается последней книгой последнего шкафа в последнем ряду. После генерации нового идентификатора информация о книге выводится в консоль в определённом формате (Рис. 1).

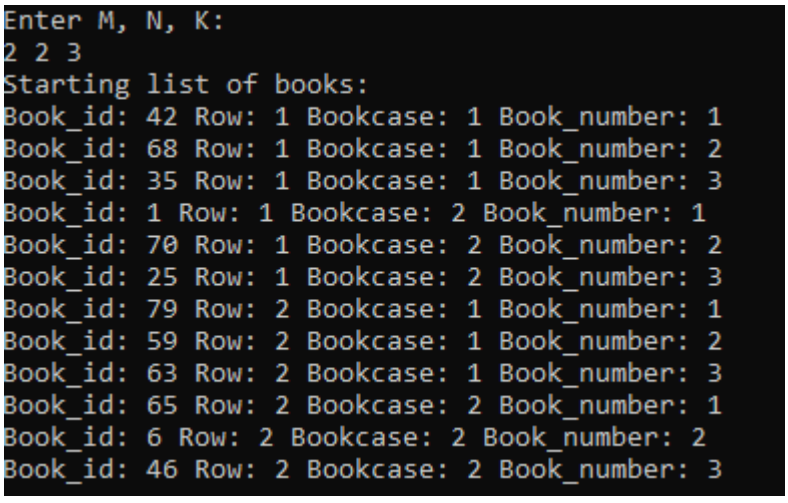

Рис. 1 Пример ввода входных данных и вывода информации о сгенерированных книгах.

Идентификаторы книг записываются в вектор “book_ids”, данные об идентификаторах и местоположениях книг отдельно сохраняются в двумерный вектор “books”. Затем вектор “book_ids” сортируется и создаётся двумерный вектор “library_list”, содержащий в себе векторы вида {x,y}, где x - целочисленный идентификатор книги, а y - порядковый номер данного идентификатора в векторе “book_ids”. Сохранять данную пару необходимо для того, чтобы книги могли записываться в каталог в порядке увеличения их идентификаторов. Запись информации о книге в каталог (что равносильно ещё выводу на экран) осуществляется с помощью N*M*K потоков, создаваемых с помощью библиотеки Open MP. Обращаясь к методу “write”, каждый поток считывает номер задачи, которую ему необходимо выполнить, берёт из вектора “library_list” пару значений, чей индекс соответствует считанному значению “task_number” и ищет книгу с соответствующим идентификатором в векторе “books”. Поскольку книг с искомым идентификатором может быть несколько, поток с помощью цикла проходит вектор “books” от начала до конца и ищет среди всех одинаковых идентификаторов идентификатор с нужным порядковым номером из library_list. При нахождении нужного элемента информация о нём записывается в каталог, после чего проход цикла и выполнение задачи завершается, и поток готовится к получению следующей задачи.

Для контроля изменения значения “task_number” в методе “write” используется критическая секция (“#pragma omp critical”), которая не позволяет двум потокам менять номер текущей задачи и записывать данные в каталог одновременно.

3. Тестирование программы

Выходные данные, полученные при тестировании программы на корректных значениях параметров M, N, K можно увидеть на скриншотах, приложенных к отчёту на GitHub. При вводе некорректных значений программа выводит в консоль сообщение с просьбой повторить ввод (Рис. 2) и повторяет это действие до тех пор, пока введённые данные не станут корректными.

```
Enter M, N, K:
-1 1 1
Wrong input! M, N, K should be integer numbers from 1 to 100. Try it again, please
Enter M, N, K:
2 2 abc
Wrong input! M, N, K should be integer numbers from 1 to 100. Try it again, please
Enter M, N, K:
10000 1 1
Wrong input! M, N, K should be integer numbers from 1 to 100. Try it again, please
Enter M, N, K:
2 2 2
Starting list of books:
Book_id: 42 Row: 1 Bookcase: 1 Book_number: 1
Book_id: 68 Row: 1 Bookcase: 1 Book_number: 2
Book_id: 35 Row: 1 Bookcase: 2 Book_number: 1
Book_id: 1 Row: 1 Bookcase: 2 Book_number: 2
Book_id: 70 Row: 2 Bookcase: 1 Book_number: 1
Book_id: 25 Row: 2 Bookcase: 1 Book_number: 2
Book_id: 79 Row: 2 Bookcase: 2 Book_number: 1
Book_id: 59 Row: 2 Bookcase: 2 Book_number: 2
```

Рис. 2. Реакция программы на некорректный ввод данных.

ПРИЛОЖЕНИЕ 1

Список литературы

1. Парадигмы параллельного программирования - Блог программиста. [Электронный ресурс] // URL: https://pro-prof.com/forums/topic/parallel-programming-paradigms, дата обращения: 15.11.2020