НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Дисциплина: «Архитектура вычислительных систем»

Домашнее задание

«Задача об инвентаризации по рядам. OpenMP»

Вариант 16

Выполнил: Казанцев Никита Олегович,

студент БПИ191, ФКН ПИ, ВШЭ

Москва 2020

1. Текст задания

Задача об инвентаризации по рядам. 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 составление каталога одним студентом для одного.

2. Описание принципа выполнения

Имеется класс БИБЛИОТЕКА. Библиотека содержит в себе множество из объектов класса РЯД книг. Каждый ряд книг содержит в себе множество объектов класса ШКАФ. А шкаф содержит в себе множество объектов класса КНИГА, каждая книга характеризуется каким-то случайным номеров. Каждая книга - уникальное число. Они не отсортированы и произвольны (используется генератор случайных чисел). Пусть число обозначает фамилию автора книги и ее название.

Следовательно, каталог - это одномерный массив, в котором, например, все числа отсортированы, например, в порядке возрастания. Каждый элемент этого массива – объект книги, в которой автору (числу) ставится в соответствие номер ряда, номер полки, номер места на полке.

В коде запускается M потоков, которые берут задачу из портфеля задач и асинхронно формируют свои отсортированные по числам (авторам) элементы.

3. Описание входных данных

Программа запрашивает на ввод три беззнаковых целочисленных значения, описанных ниже, на ввод.

```
unsigned int ryadi = 10;
unsigned int shkafi = 5;
unsigned int knigi = 5;
```

Переменная *ryadi* обозначает количество рядов с книгами в библиотеке.

Переменная *shkafi* обозначает количество шкафов в рядах с книгами в библиотеке.

Переменная *knigi* обозначает количество книг в шкафах в библиотеке.

```
Введите кол-во рядов, кол-во шкафов в ряду, кол-во книг в шкафу
10 20 -1

Process finished with exit code 1
```

1 Пример некорректного ввода

```
Введите кол-во рядов, кол-во шкафов в ряду, кол-во книг в шкафу
3 4 5
Библиотека до восстановления
Ряд#: 0
Шкаф#: 0
книги:
1921919875 1405804598 742794292 821225633 461814562
```

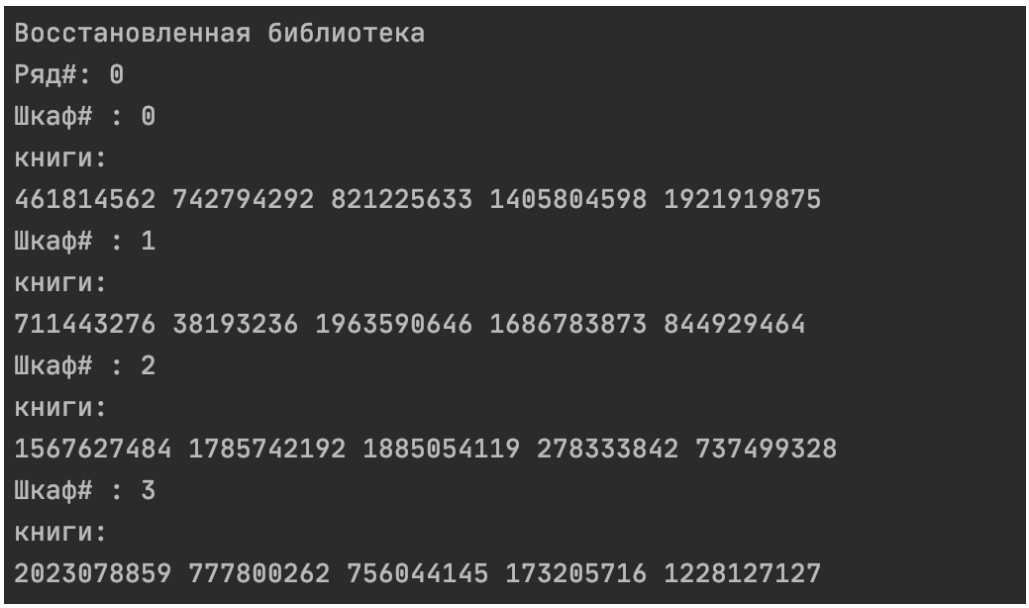
2 Пример корректного ввода

4. Описание выходных данных

Выходные данные представляют собой вывод в консоль состояния библиотеки (ее рядов) ДО и ПОСЛЕ. Выводится номер ряда, а дальше с новой строки выводятся шкафы с номерными обозначениями книг внутри А так же выводится сам каталог, где каждый элемент - какой-то номер, обозначающий книгу.

```
Ряд#: 0
Шкаф#: 0
книги:
1921919875 1405804598 742794292 821225633 461814562
Шкаф#: 1
книги:
711443276 38193236 1963590646 1686783873 844929464
Шкаф#: 2
книги:
1567627484 1785742192 1885054119 278333842 737499328
Шкаф#: 3
книги:
2023078859 777800262 756044145 173205716 1228127127
```

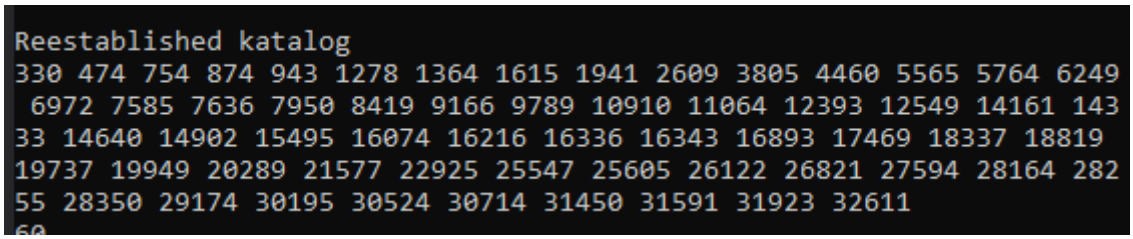
3 Пример вывода одного из многих рядов



4 Результат работы алгоритма

В 4 приложении выше показан результат работы алгоритма на примере одного из многих рядов, где произошло восстановление библиотеки (сортировка).

А так же выводится **сам каталог**, где каждый элемент - какой-то номер, обозначающий книгу.



5 Результат работы алгоритма. КАТАЛОГ

5. Описание ключевых переменных

Данные о книгах, рядах, шкафах хранятся в объекте класса Library *lib*. В объекте класса Library содержится массив объектов класса ryad *ryadi* – именно здесь хранится информация о конкретном ряде и шкафах, книгах внутри. Каждый объект класса ryad содержит в себе массив шкафов класса shkaf *shkafi*. В каждом объекте класса shkaf содержится массив из объектов класса book, где хранятся номерные значения книг.

6. Текст программы

```
#include <iostream>
#include <cmath>
#include <string>
#include <fstream>
#include <cstdlib>
#include <map>
#include "omp.h"
#include <iostream>
#include <string>
#include <algorithm>
#include <thread>
#include <vector>
#include <iostream>
#include <array>
using namespace std;

unsigned int Cryadi = 10;
unsigned int Cshkafi = 5;
unsigned int Cknigi = 5;

// класс конкретной книги
class book { // The class
public: // Access specifier
 int num; // Attribute (int variable)

 book() {
		num = rand();
 }

};

// класс шкафа где хранятся книги
```

```
class shkaf { // The class
private:
 int katalogi_k = Cknigi;
public: // Access specifier

 vector<book> katalogs; // Attribute (katalog variable)
 shkaf() {
		//katalogs = new book [this->katalogi_k];

		for (int i = 0; i < katalogi_k; ++i) {
			katalogs.push_back(book());
		}
 }
};

// класс ряда, где находятся шкафы где хранятся книги
class ryad { // The class
private:
 int shkafi_k = Cshkafi;
public: // Access specifier
 shkaf* shkafi; // Attribute (shkaf variable)

 ryad() {
		shkafi = new shkaf[this->shkafi_k];

		for (int i = 0; i < shkafi_k; ++i) {
			shkafi[i] = shkaf();
		}
 }
};

// класс библиотеки где множество рядов, где находятся шкафы где хранятся книги
class library { // The class
private:
```

```
 int ryadi_k = Cryadi;
public: // Access specifier
 ryad* ryadi; // Attribute (ryad variable)

 library() {
        ryadi = new ryad[this->ryadi_k];

        for (int i = 0; i < ryadi_k; ++i) {
                ryadi[i] = ryad();
        }
 }
};

//Реализация параллельных вычислений возможна с использо-
//ванием так называемого портфеля задач.
// в нашем случае в качестве отдельной задачи задается составление каталога одним студентом для одного ряда.
//
// а каждый элемент этой иерархии - класс (книжка),
// в которой автору (числу) ставится в соответствие номер ряда, номер полки, номер места на полке.

library lib = library(); // библиотека заполнилась случайными рядами со случайной расстановкой книг

//Фонд библиотека представляет собой прямоугольное помещение,
// в котором находится M рядов по N шкафов по K книг в каждом шкафу.
// То есть, имеется трехмерный массив размерностью M на N.

//Следовательно, каталог - это одномерный массив, в котором,
// все числа отсортированы в порядке возрастания - нам этого нужно добиться (восстановить каталог)

// компаратор для функции сортировки, сравнивающий два объекта класса книга
bool comparator(const book lhs, const book rhs) {
```

```
 return lhs.num < rhs.num;
}

// функция сортировки каталога
void sort_catalog(int ryadd, int shkaff) {
 sort(lib.ryadi[ryadd].shkafi[shkaff].katalogs.begin(),
lib.ryadi[ryadd].shkafi[shkaff].katalogs.end(), comparator);
}

int main(int args, const char* argv[]) {

 vector<std::pair<int, int> > backpack; // портфель с задачами
 vector<int> Allbooks;
 int Iryad;
 int Ishkaf;
 int Ikniga;
 cout<<"Input number of RYADS, number of SHKAFS in RYAD, number of BOOKS in
SHKAF"<<endl;

 cin>>Iryad>>Ishkaf>>Ikniga;

 if (Iryad <= 0 || Ishkaf <= 0 || Ikniga <= 0)
        return 1;

 Cryadi = Iryad;
 Cshkafi = Ishkaf;
 Cknigi = Ikniga;

 cout << "Library before reestablishing" << endl;
 lib = library();

 for (int ryad = 0; ryad < Cryadi; ++ryad) {
        cout << "Ryad#: " << ryad << endl;
        for (int shkaf = 0; shkaf < Cshkafi; ++shkaf) {
```

```
                cout << "Shkaf#: " << shkaf << endl;
                cout << "Books:" << endl;
                for (int kniga = 0; kniga < Cknigi; ++kniga) {
                        //Каждая книга - уникальное число.
                        // Они не отсортированы и произвольны (можно использовать генератор случайных чисел).
                        // Пусть число обозначает фамилию автора книги и ее название.
                        cout << lib.ryadi[ryad].shkafi[shkaf].katalogs[kniga].num << " ";
                        backpack.push_back(pair<int, int>(ryad, shkaf)); // загружаем задачу в портфель задач
                }
                cout << endl;
        }
 }

 cout << endl << endl << endl;

 sort(lib.ryadi[0].shkafi[0].katalogs.begin(), lib.ryadi[0].shkafi[0].katalogs.end(), comparator);
 //Нужно запустить M потоков, которые асинхронно формируют свои отсортированные по числам (авторам) элементы.

 // Нужно запустить M потоков, которые асинхронно формируют свои отсортированные по числам (авторам) элементы.
 // т.е. каждый шкаф сортируется, чтоб сформировать каталог в правильном порядке
        //Устанавливаем число потоков



#pragma omp parallel shared (backpack, Iryad, Ishkaf) num_threads(Iryad*Ishkaf)
 {
#pragma omp for nowait
 for (int i = 0; i < Iryad; i++) {
        for (int j = 0; j < Ishkaf; j++) {
                pair<int, int> task = backpack.back();
```

```
                cout << omp_get_thread_num() << " ";
                // поток взял задачу на выполнение.
                sort_catalog(task.first, task.second);

                backpack.pop_back(); // задачу удалили

        }
}
}


// наша библиотека после восстановления
cout << "\nReestablished library" << endl;
for (int ryad = 0; ryad < Cryadi; ++ryad) {
        cout << "Ryad#: " << ryad << endl;
        for (int shkaf = 0; shkaf < Cshkafi; ++shkaf) {
                cout << "Shkaf# : " << shkaf << endl;
                cout << "Books: " << endl;
                for (int kniga = 0; kniga < Cknigi; ++kniga) {
                        cout << lib.ryadi[ryad].shkafi[shkaf].katalogs[kniga].num << " ";
                }
                cout << endl;
        }
        cout << endl;
}


// запускаем в потоках заполнение вектора
// т.е. формируем пока еще не готовый каталог
#pragma omp parallel shared (Allbooks, Cryadi, Cshkafi, Cknigi) num_threads(5)
{
#pragma omp for nowait
        for (int ryad = 0; ryad < Cryadi; ++ryad) {
                for (int shkaf = 0; shkaf < Cshkafi; ++shkaf) {
```

```
                                for (int kniga = 0; kniga < Cknigi; ++kniga) {

Allbooks.push_back(lib.ryadi[ryad].shkafi[shkaf].katalogs[kniga].num);
                                }
                        }
                }
}

// сортируем каталог. т.е. восстаанавливаем его
sort(Allbooks.begin(), Allbooks.end());

cout << "\nReestablished katalog" << endl;
for (int book = 0; book < Allbooks.size(); ++book) {
        cout << Allbooks[book] << " ";
}

cout <<endl<< Allbooks.size() << " ";
return 0;
}
```

7. Использованная литература и интернет источники

a. Html version of downloadable textbook "OpenMP topic: Loop parallelism" https://pages.tacc.utexas.edu/~eijkhout/pcse/html/omp-loop.html [интернет ресурс]
b. Алгоритмы параллельных вычислений и программирование http://window.edu.ru/catalog/pdf2txt/971/67971/41350?p_page=20 [интернет ресурс]