АВС. ИДЗ-4

Гареев Данир БПИ-236

Задание

Изучить работу с потоками. Научиться разбивать задачу на части, для последующего их выполнения различными потоками.

Вариант 12

Задача об Острове Сокровищ. Шайка пиратов под предводительством Джона Сильвера высадилась на берег Острова Сокровищ. Не смотря на добытую карту острова старого Флинта, местоположение сокровищ по-прежнему остается загадкой, поэтому искать клад приходится практически на ощупь. Так как Сильвер ходит на деревянной ноге, то самому бродить по джунглям ему не с руки. Джон Сильвер поделил остров на участки, а пиратов на небольшие группы. Каждой группе поручается искать клад на одном из участков, а сам Сильвер ждет на берегу. Пираты, обшарив свою часть острова, возвращаются к Сильверу и докладывают о результатах. Требуется создать многопоточное приложение с управляющим потоком, моделирующее действия Сильвера и пиратов. Примечание. Количество участков превышает число поисковых групп. Потоки должны задавать каждую группу и Сильвера (управляющий поток).

Решение на 8 баллов

Все файлы хранятся на GitHub по ссылке.
Есть 4 файла:

1. main.cpp
2. conf.txt
3. output.txt - файл с результатами тестов
4. output1.txt - файл с результатами тестов
5. output2.txt - файл с результатами тестов

Логика программы

Задаётся остров с N * M ячейками (где N - высота, M - ширина острова). Создается остров и в случайной ячейке располагается клад. Пираты делятся на группы, создаются потоки для групп. Джон Сильвер (управляющий поток) управляет очередью задач. Рабочие потоки (группы пиратов) извлекают задания, обрабатывают их. Группа исследует ячейку и обновляет поле. Если клад найден, устанавливается флаг завершения и все потоки останавливаются.

Модель делегирования: управляющий и рабочие

1. Общие потоки

Программа реализует модель делегирования, где главный управляющий (основной поток) распределяет задания между рабочими (группы пиратов, работающие в отдельных потоках). Основной поток отвечает за распределение задач, контроль выполнения и завершение работы программы.

2. Очереди

Очередь `tasksQueue` управляет заданиями для рабочих потоков. Основной поток назначает задания группам, помещая их идентификаторы в очередь, а рабочие потоки извлекают из нее задачи. Очередь используется для упорядоченного выполнения работы и предотвращения гонок данных.

3. Синхронизация

Синхронизация потоков осуществляется с использованием:

- `std::mutex` для защиты общих ресурсов (`mtx`, `coutMtx`)
- `std::condition_variable` (`cv`, `group_cv`) для организации ожидания задач рабочими потоками и их пробуждения после назначения задания.
- `std::atomic<bool>` для управления завершением работы (`found`).

4. Этапы выполнения

1. Чтение параметров острова (размер поля и количество пиратов) из командной строки или файла конфигурации.
2. Инициализация острова и случайное размещение клада.
3. Разделение пиратов на группы и создание рабочих потоков.
4. Основной поток поочередно назначает задания группам (поиск в ячейках острова).
5. Рабочие потоки выполняют задания и обновляют состояние острова.
6. Если клад найден, работа останавливается, и программа завершает выполнение.

5. Многопоточность и конкуренция

Программа использует несколько потоков для выполнения задач:

- Основной поток управляет процессом и назначает задания.
- Каждый рабочий поток (группа пиратов) выполняет назначенные задания независимо.
- Очередь задач и синхронизация предотвращают конфликты между потоками.
- Используются мьютексы для безопасного вывода информации в консоль и файл.

6. Закрытие программы

1. Устанавливается флаг `found = true`.
2. Все ожидающие потоки пробуждаются с помощью `cv.notify_all()`.
3. Рабочие потоки завершают выполнение и выходят из цикла работы.
4. Основной поток завершает оставшиеся задачи, закрывает файл вывода и корректно завершает выполнение программы.

Случайные числа

1. Определение расположения клада

Генерируется случайное число в диапазоне `[0, fieldHeight * fieldWidth - 1]`, определяющее ячейку, где спрятан клад.

2. Инициализация генератора случайных чисел

Используется текущее время в секундах (`std::time(0)`) как зерно (`seed`) для генерации случайных чисел, чтобы при каждом запуске программы клад располагался в новой случайной позиции.

Пример работы программы

Тест 1

Используем файл -f conf.txt в котором записаны строки 3 3 20. Выбираем группы 1 19. Выводим данные в output1.txt

```
❯ ./pirate_simulator output1.txt -f conf.txt
Остров:
_ _ _
_ _ _
_ _ _
Введите число пиратов в группе 1: 1
Введите число пиратов в группе 2: 19
Команде 1 назначено задание в ячейке 0
Команде 2 назначено задание в ячейке 1
Группа 1 начала обследовать ячейку 0
Группа 2 начала обследовать ячейку 1
Группа 2 закончила обследование ячейки 1
Остров:
_ O _
_ _ _
_ _ _
Команде 2 назначено задание в ячейке 2
Группа 2 начала обследовать ячейку 2
Группа 2 закончила обследование ячейки 2
Остров:
_ O O
_ _ _
_ _ _
Команде 2 назначено задание в ячейке 3
Группа 2 начала обследовать ячейку 3
Группа 2 закончила обследование ячейки 3
```

```
Остров:
_ O O
O _ _
_ _ _
Команде 2 назначено задание в ячейке 4
Группа 2 начала обследовать ячейку 4
Группа 2 закончила обследование ячейки 4
Остров:
_ O O
O O _
_ _ _
Команде 2 назначено задание в ячейке 5
Группа 2 начала обследовать ячейку 5
Группа 2 нашла клад в ячейке 5!
Остров:
_ O O
O O X
_ _ _
Группа 1 закончила обследование ячейки 0
Остров:
O O O
O O X
_ _ _
```

Тест 2

Используем входные данные с консоли. Выбираем группы 1 18 1. Выводим данные в output.txt

```
❯ ./pirate_simulator output.txt 3 3 20
Остров:
_ _ _
_ _ _
_ _ _
Введите число пиратов в группе 1: 1
Введите число пиратов в группе 2: 18
Введите число пиратов в группе 3: 1
Команде 1 назначено задание в ячейке 0
Команде 2 назначено задание в ячейке 1
Команде 3 назначено задание в ячейке 2
Группа 1 начала обследовать ячейку 0
Группа 2 начала обследовать ячейку 1
Группа 3 начала обследовать ячейку 2
Группа 2 закончила обследование ячейки 1
Остров:
_ O _
_ _ _
_ _ _
Команде 2 назначено задание в ячейке 3
Группа 2 начала обследовать ячейку 3
Группа 2 закончила обследование ячейки 3
```

```
Остров:
_ O _
O _ _
_ _ _
Команде 2 назначено задание в ячейке 4
Группа 2 начала обследовать ячейку 4
Группа 2 закончила обследование ячейки 4
Остров:
_ O _
O O _
_ _ _
Команде 2 назначено задание в ячейке 5
Группа 2 начала обследовать ячейку 5
Группа 2 закончила обследование ячейки 5
Остров:
_ O _
O O O
_ _ _
Команде 2 назначено задание в ячейке 6
Группа 2 начала обследовать ячейку 6
Группа 2 нашла клад в ячейке 6!
Остров:
_ O _
O O O
X _ _
Группа 1 закончила обследование ячейки 0
Остров:
O O O
O O O
X _ _
```

```
Группа 3 закончила обследование ячейки 2
Остров:
O O O
O O O
X _ _
```

Тест 3

Используем файл -f conf.txt в котором записаны строки 3 3 20. Выбираем одну группу. Выводим данные в output2.txt

```
❯ ./pirate_simulator output2.txt 3 3 20
Остров:
_ _ _
_ _ _
_ _ _
Введите число пиратов в группе 1: 20
Команде 1 назначено задание в ячейке 0
Группа 1 начала обследовать ячейку 0
Группа 1 закончила обследование ячейки 0
Остров:
O _ _
_ _ _
_ _ _
Команде 1 назначено задание в ячейке 1
Группа 1 начала обследовать ячейку 1
Группа 1 закончила обследование ячейки 1
Остров:
O O _
_ _ _
_ _ _
Команде 1 назначено задание в ячейке 2
Группа 1 начала обследовать ячейку 2
Группа 1 закончила обследование ячейки 2
Остров:
O O O
_ _ _
_ _ _
Команде 1 назначено задание в ячейке 3
Группа 1 начала обследовать ячейку 3
Группа 1 закончила обследование ячейки 3
```

```
Остров:
O O O
O _ _
_ _ _
Команде 1 назначено задание в ячейке 4
Группа 1 начала обследовать ячейку 4
Группа 1 закончила обследование ячейки 4
Остров:
O O O
O O _
_ _ _
Команде 1 назначено задание в ячейке 5
Группа 1 начала обследовать ячейку 5
Группа 1 закончила обследование ячейки 5
Остров:
O O O
O O O
_ _ _
Команде 1 назначено задание в ячейке 6
Группа 1 начала обследовать ячейку 6
Группа 1 закончила обследование ячейки 6
Остров:
O O O
O O O
O _ _
Команде 1 назначено задание в ячейке 7
Группа 1 начала обследовать ячейку 7
Группа 1 нашла клад в ячейке 7!
```

```
Остров:
O O O
O O O
O X _
```

Выполненные требования

На 4-5

1. Соблюдены общие требования к отчёту
2. В отчете должен быть приведен сценарий, описывающий одновременное поведение представленных в условии задания сущностей в терминах предметной области.
3. Описана модель параллельных вычислений, используемая при разработке многопоточной программы.
4. Описаны входные данные программы, включающие вариативные диапазоны, возможные при многократных запусках.
5. Реализовано консольное приложение, решающее поставленную задачу с использованием одного варианта изученных синхропримитивов.
6. Ввод данных в приложение реализован с консоли во время выполнения программы.
7. Для используемых генераторов случайных чисел описаны их диапазоны и то, как интерпретируются данные этих генераторов.
8. Вывод программы информативный, отражает все ключаевые протекающие в ней события.
9. В программе присутствуют комментарии, поясняющие выполняемые действия и описание используемых объектов и переменных.
10. Результаты работы программы представлены в отчете.

На 6-7

1. В отчете подробно описан обобщенный алгоритм, используемый при реализации программы исходного словесного сценария. В котором показано, как на программу отображается каждый из субъектов предметной области.
2. В программу добавлена генерация случайных данных в допустимых диапазонах.
3. Реализован ввод исходных данных из командной строки при запуске программы вместо ввода параметров с консоли во время выполнения программы.
4. Результаты изменений отражены в отчете.

На 8

1. В программу, наряду с выводом в консоль, добавлен вывод результатов в файл. Имя файла для вывода данных задается в командной строке как один из ее параметров.
2. Результаты работы программы выводятся на экран и записываться в файл.
3. Наряду с вводом исходных данных через командную строку добавлен альтернативный вариант их ввода из файла, который по сути играет роль конфигурационного файла. Имя этого файла задается в командной строке вместо параметров, которые в этом случае не вводятся. Управление вводом в командной строке осуществляется с использованием соответствующих ключей.
4. Приведено не менее трех вариантов входных и выходных файлов с различными исходным данными и результатами выполнения программы.
5. Ввод данных из командной строки расширен с учетом введенных изменений.
6. Результаты изменений отражены в отчете.

Исходный код

main.cpp

```cpp
#include <iostream>
#include <fstream>
#include <thread>
#include <vector>
#include <queue>
#include <mutex>
#include <condition_variable>
#include <string>
#include <cstdlib>
#include <ctime>
#include <climits>
```

```cpp
#include <atomic>
#include <sstream>

//
int fieldHeight, fieldWidth, people;
std::vector<std::string> island;
std::mutex mtx, coutMtx;
std::condition_variable cv, group_cv;
std::queue<int> tasksQueue;
std::vector<int> tasks;
bool found = false;
int treasureNumber;
std::vector<int> groups;
std::atomic<int> activeGroup{1}; //                    ,


void outIsland(std::ostream& os) {
    os << "             :\n";
    for (int i = 0; i < fieldHeight; i++) {
        for (int j = 0; j < fieldWidth; j++) {
            os << island[i * fieldWidth + j] << " ";
        }
        os << std::endl;
    }
}

void groupFunction(int id, int members, std::ofstream& out) {
    while (true) {
        std::unique_lock<std::mutex> lock(mtx);
        cv.wait(lock, [id] { return tasks[id] != -1 || found; });

        if (found) return;  //                                          ,


        int task = tasks[id];
        tasks[id] = -1;
        lock.unlock();

        {
            std::lock_guard<std::mutex> lock(coutMtx);
            std::cout << "             " << id << "                                                    "
                 << task << std::endl;
            out << "              " << id << "                                                  " <<
                task << std::endl;
        }

        std::this_thread::sleep_for(std::chrono::milliseconds(5000 / members)); //


        if (task == treasureNumber) {
            island[task] = "X";

            {
                std::lock_guard<std::mutex> lock(coutMtx);
                std::cout << "             " << id << "                                          " << task
                    << "!\n";
                out << "             " << id << "                                        " << task << "!\
                    n";
                outIsland(std::cout);
                outIsland(out);
            }

            std::lock_guard<std::mutex> lock(mtx);
            found = true;
            cv.notify_all();
            return;
        }

        island[task] = "0";

        {
            std::lock_guard<std::mutex> lock(coutMtx);
            std::cout << "             " << id << "
                        " << task << std::endl;
            out << "              " << id << "
                 " << task << std::endl;
            outIsland(std::cout);
            outIsland(out);
        }

        std::lock_guard<std::mutex> lock2(mtx);
        tasksQueue.push(id);
```

```
        cv.notify_all();
    }
}

bool loadConfig(const std::string& filename) {
    std::ifstream file(filename);
    if (!file) {
        std::cerr << "            :
                                " << filename << "\n";
        return false;
    }

    std::string line;
    if (std::getline(file, line)) {
        std::istringstream iss(line);
        if (!(iss >> fieldHeight >> fieldWidth >> people)) {
            std::cerr << "            :
                                                    .\n";
            return false;
        }
        if (fieldHeight <= 0 || fieldWidth <= 0 || people <= 0) {
            std::cerr << "            :
                                                    .\n";
            return false;
        }
    } else {
        std::cerr << "            :                                    .\n";
        return false;
    }

    return true;
}

int main(int argc, char* argv[]) {
    setlocale(LC_ALL, "rus");
    std::srand(std::time(0));

    if (argc < 3) {
        std::cerr << "                        : " << argv[0] << " <          _          > [-f <
                        _                        > | <            > <            > <
            >]\n";
        return 1;
    }

    std::ofstream out(argv[1]);
    if (!out) {
        std::cerr << "                                                            .\n";
        return 1;
    }

    if (std::string(argv[2]) == "-f") {
        if (argc < 4) {
            std::cerr << "            :
                                                                    `-f`.\n";
            return 1;
        }
        if (!loadConfig(argv[3])) {
            return 1;
        }
    } else {
        try {
            fieldHeight = std::stoi(argv[2]);
            fieldWidth = std::stoi(argv[3]);
            people = std::stoi(argv[4]);
            if (fieldHeight <= 0 || fieldWidth <= 0 || people <= 0) {
                std::cerr << "            :
                                                        .\n";
                return 1;
            }
        } catch (const std::exception& e) {
            std::cerr << "            :
                                    .\n";
            return 1;
        }
    }

    groups.push_back(0);
    tasks.push_back(0);
    island.resize(fieldHeight * fieldWidth, "_");
    outIsland(std::cout);
    outIsland(out);
```

```cpp
    treasureNumber = std::rand() % (fieldHeight * fieldWidth);

    int freePeople = people;
    for (int i = 1; freePeople > 0; i++) {
        std::cout << "                                                          " << i << ": ";
        int count;
        while (true) {
            std::cin >> count;
            if (std::cin.fail() || count <= 0 || count > freePeople) {
                std::cin.clear();
                std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
                std::cout << "                                  .
                                                    ,                              " << freePeople
                     << ".\n";
            } else {
                break;
            }
        }
        groups.push_back(count);
        freePeople -= count;
    }

    std::vector<std::thread> threads;
    for (size_t i = 1; i < groups.size(); i++) {
        threads.emplace_back(groupFunction, i, groups[i], std::ref(out));
        tasks.push_back(-1);
        tasksQueue.push(i);
    }

    for (int i = 0; i <= fieldHeight * fieldWidth; i++) {
        std::unique_lock<std::mutex> lock(mtx);
        cv.wait(lock, [] { return !tasksQueue.empty() || found; });

        if (found)
            break;

        int threadId = tasksQueue.front();
        tasksQueue.pop();
        tasks[threadId] = i;

        {
            std::lock_guard<std::mutex> coutLock(coutMtx);
            std::cout << "                " << threadId << "
                            " << i << std::endl;
            out << "                " << threadId << "
                            " << i << std::endl;
        }

        cv.notify_all();
        group_cv.notify_all(); //
    }

    {
        std::lock_guard<std::mutex> lock(mtx);
        found = true;
        cv.notify_all();
    }

    for (auto& t : threads) {
        if (t.joinable()) {
            t.join();
        }
    }

    out.close();
    return 0;
}
```