Пояснительная записка к домашнему заданию 5
Тимохина Софья Константиновна БПИ207
Вариант 27

Описание задачи

Задача о Винни-Пухе - 3 или мстительные пчелы. Неправильные пчелы, подсчитав в конце месяца убытки от наличия в лесу Винни-Пуха, решили разыскать его и наказать в назидание всем другим любителям сладкого. Для поисков медведя они поделили лес на участки, каждый из которых прочесывает одна стая неправильных пчел. В случае нахождения медведя на своем участке стая проводит показательное наказание и возвращается в улей. Если участок прочесан, а Винни-Пух на нем не обнаружен, стая также возвращается в улей. Требуется создать многопоточное приложение, моделирующее действия пчел. При решении использовать парадигму портфеля задач.

Необходимо разработать алгоритм решения задания, с учетом разделения вычислений между несколькими потоками. Избегать ситуаций неуправляемого изменения одних и тех же общих данных несколькими потоками. Если же избежать этого невозможно, необходимо использовать мьютексы и критические секции. А также провести отладку и тестирование разработанной программы. Программа должна правильно обрабатывать входные данные в соответствии с условием задания и реагировать на некорректно вводимые исходные данные. Ввод основных данных должен осуществляться в допустимом для условия задачи диапазоне без введения искусственных ограничений.

Решение приведено на языке C++. Для решения проблемы с доступом к переменной, общей для всех потоков, а также некорректного вывода из нескольких потоков сразу были использованы мьютексы и критические секции.

Задача была реализована таким образом, что программа принимает в качестве входных параметров (или рандомно генерирует их) площадь леса, количество участков и местоположение медведя. Корректными параметрами считаются положительные числа. При этом площадь леса должна быть не более 1000 (такое число было выбрано с целью оптимизации памяти), количество участков и местоположение медведя - не больше площади леса. Все участки оптимально разбиваются на регионы, каждый регион имеет площадь, а также местоположение медведя. В том случае, если медведь находится в другом регионе, в текущем регионе его местоположение равняется нулю.

Ввод данных производится через командную строку, при этом существует 2 варианта формата:

- Рандомная генерация

  Для использования рандомной генерации следует посылать флаг -r.

- Ввод информации

  Для использования заданной пользователем информации следует посылать флаг -f, а также 3 неотрицательных целых числа: площадь леса, количество регионов, местоположение медведя. Требования к параметрам указаны выше. Все данные проверяются, в случае некорректного ввода выводится ошибка и программа приостанавливается.

  Пример:

  -n 1000 10 25

  Площадь в данном случае равна 1000, количество регионов - 10, положение медведя - 25.

Описание используемой парадигмы

Для решения задачи была использована парадигма многопоточного программирования взаимодействующие равные, конкретно один из распространенных способов динамического распределения задач под названием портфель задач.

Портфель задач реализуется с помощью разделяемой переменной, доступ к которой в один момент времени имеет лишь один процесс (поток). Задача делится на конечное число подзадач, при этом подзадачи, как правило, однотипные. Поток сначала обращается к портфелю задач для выяснения номера задачи, а затем берет нужные данные для выяснения задачи. В том случае, если задачи закончились, процесс завершается. В программе также реализован функционал завершения работы всех потоков в случае достижения цели одним из них. [https://pro-prof.com/forums/topic/parallel-programming-paradigms].

Общий алгоритм работы выглядит следующим образом:

```
int NextProblem = 0; // портфель задач

process worker [i = 0 to n-1] {

   получить задачу из портфеля;

   while ( задача существует)   {

      выполнить задачу;

      получить задачу из портфеля;

   }

}
```

Из преимуществ данной парадигмы можно отметить:

- Она проста в реализации

  Для ее работы нужно даль представление задач, определить набор задач, дать программу выполнения задач, определить критерий окончания

- Программы с использованием данной парадигмы легко масштабируются изменением числа процессов
- Упрощается балансировочная нагрузка (освободившиеся процессы берут на себя решение новых задач*) [http://window.edu.ru/catalog/pdf2txt/971/67971/41350?p_page=20]

* Не касается данной программы, по условию, процесс не берет новую задачу, а приостанавливает выполнение.

Как парадигма использована в программе

Класс Beehive, владеющий информацией о регионах и в зависимости от их (регионов) количества создающий новые потоки (стаи пчел), которые проверяют данный регион, запускает потоки. Чтобы все потоки работали одновременно корректно, класс имеет поле threads_, вектор из потоков. Создание потоков:

```
    for (int i = 0; i < regions_.size(); ++i) {
        std::thread tr(&Beehive::searchRegion, this,regions_[i], i + 1);
        threads_.emplace_back(std::move(tr));
    }

    for(auto& thr : threads_) {
        thr.join();
    }
```

Функция поиска:

```
void Beehive::searchRegion(Region *region, const int &flock_index) {
    for (int i = 0; i < region->getArea(); ++i) {
        {
            std::lock_guard<std::mutex> lock(mtx_);
            // Check if another one flock didn't find a bear
            if (bear_found_) {
                std::cout << "[Flock " << flock_index << "] got a signal that the bear
was found.\n";
                break;
            }
            std::cout << "[Flock " << flock_index << "] searching in the point " << i +
1 << " " << '\n';
            // Check if we found a bear
            if (region->hasBear(i)) {
                std::cout << "[Flock " << flock_index << "] found a bear!\n";
                std::cout << "**I'm against violence so they dispersed peacefully**\n";
                bear_found_ = true;
                break;
            }
        }
        std::this_thread::sleep_for(std::chrono::milliseconds(getSleepTime()));
    }
```

Как можно видеть, в коде использованы мьютексы и критические секции для реализации корректного вывода и доступа к переменной bear_found_ - булевого флага, благодаря которому стаи (потоки) понимают, нужно ли обращаться к следующей задаче (по факту, следующей клетке региона) или же стоит приостановиться.
При этом можно заметить, что потоки засыпают:

```
std::this_thread::sleep_for(std::chrono::milliseconds(getSleepTime()));
```

Это сделано, можно сказать, в декоративных целях для имитации поиска. Спят они от 10 до 410 мс (рандомное время).

Каждый поток имеет свой регион, при этом задачи для каждого потока однотипны, а условия завершения одни и те же.

Основные характеристики программы

- Число заголовочных файлов: 2
- Число модулей реализации: 4
- Число строк: 209 (6 кб)
- Время выполнения:

Команда	Время (секунд)
-n 100 12 86	3.6569
-n 300 13 89	6.1896
-n 500 11 111	4.5511
-n 500 3 111	5.481
-n 1000 15 118	9.0333
-r	Чаще около 5