Домашнее задание 1

Задача 1. Обедающие философы (Dining philosophers problem)

Постановка задачи

Пять безмолвных философов сидят за круглым столом, перед каждым философом стоит тарелка спагетти. Вилки лежат на столе между каждой парой ближайших философов (всего 5 вилок).

Каждый философ может либо есть, либо размышлять. Приём пищи не ограничен количеством оставшихся спагетти (бесконечный запас). Тем не менее, философ может есть только тогда, когда одновременно держит две вилки.

Каждый философ может взять ближайшую вилку (слева или справа, если она доступна), или положить - если он уже держит её. Взятие каждой вилки и возвращение её на стол являются раздельными действиями, которые должны выполняться одно за другим.

Если требуемая вилка занята соседом, голодный философ вынужден ждать - он не может вернуться к размышлениям, не поев. После окончания еды философ кладет обе вилки на стол для того, чтобы ими могли воспользоваться другие философы.

Задача состоит в разработке модели поведения (параллельного алгоритма), при котором ни один из философов не будет голодать, то есть будет вечно чередовать приём пищи и размышления.

Задание

Требуется реализовать решение задачи об обедающих философах в виде многопоточной программы, в которой каждый философ представлен отдельным потоком. Фазы размышления и приёма пищи реализуются при помощи «засыпания» на псевдослучайное количество миллисекунд. Реализация должна удовлетворять следующим требованиям:

- Ни один из философов не голодает (будет вечно чередовать приёмы пищи и размышления при бесконечном выполнении программы)
- Ресурсы равномерно распределяются между философами (при одинаковом поведении философы едят примерно одинаковое количество раз, никто из них не получает преимущества — справедливость)
- Одновременно могут есть несколько философов
- Реализация масштабируется (количество приемов пищи не падает быстро) при увеличении количества философов

Требования к программе

Программа должна быть реализована на C++11.

Программа должна поддерживать следующие аргументы командной строки:

- Количество философов (nphils)
- Время работы программы в секундах (duration)

- Максимальная длительность состояния «думает» в миллисекундах (think_delay_max)
  Задержка в этом состоянии равномерно распределена от 0 до указанного значения
- Максимальная длительность состояния «ест» в миллисекундах (eat_delay_max)
  Задержка в этом состоянии равномерно распределена от 0 до указанного значения
- Флаг включения отладочного вывода (debugflag, 0 — выключен, 1 — включен)

Все перечисленные аргументы являются обязательными и передаются в указанном порядке.

Пример запуска программы:

```
$ ./phil 5 60 100 100 1
```

В этом случае будет запущено 5 потоков-философов на протяжении 1 минуты, с максимальными временами состояний в 100 миллисекунд и отладочным выводом.

При включенном отладочном выводе программа должна печатать в стандартный поток вывода сообщения о наступлении следующих событий (приведен формат сообщений, в квадратных скобках указывается номер философа от 1 до nphils):

Философ перешел в новое состояние

- `[1] thinking`
- `[1] hungry`
- `[1] eating`

Философ взял или отпустил вилку

- `[1] took left fork`
- `[1] put right fork`

После окончания заданного времени работы программа должна напечатать в стандартный поток вывода статистику по всем философам и завершить работу.

Статистика для каждого философа представляет строку вида

`[phil_num] eat_count wait_time`

где

- `phil_num` — номер философа от 1 до *N*
- `eat_count` — суммарное количество приемов пищи
- `wait_time` — суммарное время ожидания (пребывания в голодном состоянии) в миллисекундах

Пример реализации программы на C++ с учетом описанных требований приведен в файле phil.cpp. Однако данная реализация подвержена взаимной блокировке (deadlock).

Задача 2. Поисковый робот (Web crawler)

Постановка задачи

Напишите многопоточный поисковый робот (Web crawler), реализующий обход Web-графа в ширину и сохраняющий на диск все посещенные страницы.

При запуске роботу передаются адрес начальной страницы, максимальная глубина обхода, максимальное количество загружаемых страниц и путь к директории, в который сохраняются посещенные страницы.

Робот не должен посещать одну и ту же страницу более одного раза.

Попытайтесь добиться максимальной скорости работы робота и обоснуйте используемый подход.

Требования к реализации

Программа должна быть реализована на C++.

Программа должна поддерживать следующие аргументы командной строки:

- Адрес начальной страницы
- Максимальная глубина обхода
- Максимальное количество загружаемых страниц
- Путь к директории, в который сохраняются посещенные страницы

Все перечисленные аргументы являются обязательными и передаются в указанном порядке.

Программа может поддерживать дополнительные аргументы (например, количество потоков), следующие после описанных выше. При этом программа должна работать при отсутствии этих дополнительных аргументов, используя значения по-умолчанию или вычисляя их динамически.

Пример запуска программы:

```
$ ./crawler http://www.yandex.ru/ 2 100 /home/pages/
```

Программа должна завершать работу при достижении максимальной глубины обхода или максимального количества загруженных страниц (что наступит раньше).

Рекомендации

Для загрузки страниц на C++ рекомендуется использовать библиотеку libcurl.

В Ubuntu библиотеку можно установить с помощью команды:

```
$ sudo apt-get install libcurl4-openssl-dev
```

В Fedora библиотеку можно установить с помощью команды:

```
$ sudo yum install libcurl*
```

Пример использования libcurl для загрузки содержимого страницы в строку std::string приведен в файле curl.cpp.

Компиляция примера с помощью GCC:

```
$ g++ -Wall ./curl.cpp -o curl -lcurl
```