Бескоровайный Артем, ИДЗ-3, Вариант 8

Задание

Военная задача. Анчуария и Тарантерия — два крохотных латиноамериканских государства, затерянных в южных Андах. Диктатор Анчуарии, дон Федерико, объявил войну диктатору Тарантерии, дону Эрнандо. У обоих диктаторов очень мало солдат, но очень много снарядов для минометов, привезенных с последней американской гуманитарной помощью. Поэтому армии обеих сторон просто обстреливают наугад территорию противника, надеясь поразить минометы противника. Стрельба ведется хаотично до полного уничтожения всех минометов противника, размещенных на некоторой прямоугольной площадке размером N*N. То есть координаты целей, хоть и случайные, согласуются между разными минометчиками одной армии (это предохраняет от стрельбы в одну и ту же точку). Интервалы между выстрелами (задержки) зависят от расстояния до точки попадания, формируемой случайно. Повторная стрельба по одним и тем же координатам не производится. Количество минометов у каждой стороны одинаково и задается аргументом командной строки. Размер поля также задается в командной строке. Создать приложение, моделирующее военные действия. Каждая страна — отдельный клиент. Сервер отвечает за прием координат от клиентов и передает эти координаты другим клиентам. Он также получает информацию от клиентов об их уничтожении. Расположение минометов порождается каждым клиентом случайно.

Отчет :

Преведено решение задачи на языке C на 10 баллов

Тестирование :

Полученная программа была протестирована на всевозможных случаях, ошибок не обнаружено.

Сборка :

Для компиляции использовать:

```
gcc client.c -o client
gcc server.c -o test
gcc observer.c -o observer
```

Для запуска сервера использовать:

```
./server
```

Для запуска клиента использовать:

```
./client
```

Для запуска наблюдателя использовать:

```
./observer
```

Пояснение

4-5 баллов:

Описание работы:

```
Разработано клиент-серверное приложение, соответсвующее ТЗ.
1) Для начала работы необходимо запустить сервер.
2) Теперь можно запустить, как наблюдателя, так и клиента.

Описание работы алгоритма приложения:
Всего есть 100 игровых комнат, в каждой игровой комнате может находиться два игрока.
Как только клиент подключается к серверу, он автоматически заходит в свободную
комнату(если таковой нет, то мы сразу же его отключаем, т.к. на сервере нет мест).
Далее клиент ждет, пока комната не будет заполнена(пока не подключится второй игрок).
Как только подключается второй игрок - начинается игра.
Во время игры игроки в лобби по очереди обстреливают друг друга, фиксируя попадания.
Одновременно может идти столько игр, сколько всего лобби на сервере.
Также отмечу, что под каждое лобби создается отдельный поток, в котором и происходит
процесс игры. Это несовсем рационально, т.к. в одном потоке должно быть больше игр, но
поскольку ограничения не велики, то ничего страшного.
При запуске программы она через define задает порт и ip адрес, я реализовал так,
поскольку считаю, что нет особой разницы как делать данный пункт, при необходимости
легко переделать на консолоь, но это менее удобно.
Программа корректно обрабатывает всевозмодные случаи закрытия клиента, сервера,
наблюдателя.
```

6-8 баллов

```
В программу можно доабвлять наблюдателя, который выводит текущее количество активных
лобби на сервере.
Выводится сообщение каждый раз при изменении лобби.
Наблюдатели могут подключаться и отключаться вовремя работы приложения,при этом будет
выведено соответсвующее сообщение о подключении.
Каждый раз перед отправкой сигнала наблюдателям, программа проверяет можно ли
связаться с ними. Если связаться нельзя, то удаляем наблюдателя из списка
наблюдателей.(т.к. соединение нарушено)
```

9 баллов

```
В приложении можно подключать и отключать клиентов в любой момент времени без
возникновения ошибок.
```

10 баллов

```
При завершении работы программы сервер обрабатывает спецаильный синал SIGINT, в
котором закрывает все соединения и освобождает ресурсы.
Внутри клиентов в своб очередь предусмотрено получение сигнала о закрытии соединения с
сервером и соответсвенное закрытие клиентов.
```

Тестирование

Программа была протестирована на множестве разных случаев, примеры тестирования можно увидеть в папке Tests.