ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ “СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИИ И ИНФОРМАТИКИ”

Кафедра ВС

Лабораторная работа № 3
«РАСПРЕДЕЛЁННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ»

Выполнил:
ст. гр. МГ-165 Марков В.А.
Проверил:
Фульман В.О.

Новосибирск 2016

Задание на лабораторную работу

Базовое задание

Разработать программу, реализующую модель работы склада, отвечающего за хранение и продажу некоторого товара (одного). Склад содержит N помещений, каждый из которых может хранить определённое количество единиц товара. Поступающий товар помещается в одно из помещений специальным погрузчиком. За товаром прибило K покупателей, каждому из которых требуется по L_k единиц товара. Площадка перед складом мала и на ней может в один момент времени находиться либо погрузчик, либо один из покупателей. Если покупателям требуется больше товара, чем имеется на складе, то они ждут новых поступлений, периодически проверяя склад. Время работы склада ограниченно.

Основная нить (функция main) выполняет следующие действия:

- Формирует начальное заполнение склада (для каждого помещения случайным образом выбирается число из диапазона от 1 до 40);
- Обрабатывает опции командной строки, в которой должно быть указано сколько клиентов будет обслуживаться складом и в течении какого времени должен склад работать;
- Порождает заданное количество нитей, каждая из которых реализует алгоритм работы покупателя. Каждому покупателю случайным образом назначается количество требуемых единиц продукции (число из диапазона от 1 до 1000).
- Настраивает таймер (alarm) таким образом, чтобы он сработал по окончании времени работы склада;
- Запускает алгоритм работы погрузчика;
- После срабатывания таймера принудительно завершает все выполняющиеся нити (если таковые имеются).
- Завершает работу программы.

Алгоритм работы погрузчика:

- Пытается попасть на площадку перед складом;
- Как только попадет на площадку, ищет хотя бы один склад, в котором нет продукции, и заполняет его максимально возможным образом;
- Покидает площадку;
- «Засыпает» на 5 секунд;
- Цикл повторяется до срабатывания таймера;

Алгоритм работы покупателя:

- Пытается попасть на площадку перед складом;

- Как только попадет на площадку, ищет хотя бы один склад, в котором есть продукция, и забирает либо столько, сколько надо, либо всю продукцию;
- Покидает площадку;
- «Засыпает» на 5 секунд;
- Цикл повторяется до тех пор, пока покупателю нужна продукция;
- Программа должна на экран выводить информацию о помещениях склада.

Листинг функция main

```
int main(int argc, char const *argv[])
{
    signal(SIGALRM, signalhandler);
    try {
        if (argc != 4) {
            throw string("usage: " + string(argv[0]) + " [storage N] [custome K]
[time to live T]");
        }
    } catch (string &err) {
        cerr << err << endl;
        return -1;
    }
    int N = atoi(argv[1]);
    int K = atoi(argv[2]);
    int T = atoi(argv[3]);
    cout << "storage     : " << N << endl;
    cout << "customer    : " << K << endl;
    cout << "time to live: " << T << endl;
    Timer timer(T);
    Storage mainStorage(N);
    Loader loader(&mainStorage);
    mainStorage.showStorage();
    vector<thread> customerThreads;
    createAndRunCustomers(&customerThreads, K, &mainStorage);
    loader.createAndRunLoader();

    while(1) {}
    return 0;
}
```

Листинг таймер

```
Timer::Timer(int timeToLive)
{
```

```
    struct itimerval nval, oval;

    nval.it_interval.tv_sec  = timeToLive; // interval
    nval.it_interval.tv_usec = 0;
    nval.it_value.tv_sec     = timeToLive; // time until next expiration
    nval.it_value.tv_usec    = 0;

    setitimer(ITIMER_REAL, &nval, &oval);
}

void signalhandler(int sig)
{
    cout << __func__ << endl;
    exit(0);
}
```

Листинг погрузчик

```
void Loader::createAndRunLoader()
{
    while (1) {
        this->mainStorage_->lockLoadingArea();

        this->mainStorage_->fillRooms(100);

        this->mainStorage_->unlockLoadingArea();

        this_thread::sleep_for(chrono::seconds(5));
    }
}
```

Листинг клиент

```
int Customer::getProductionFromStorage()
{
    this->mainStorage->lockLoadingArea();

    this->takenProduction += mainStorage->getProduction((this->needProduction -
this->takenProduction));

    this->mainStorage->unlockLoadingArea();
}

void Customer::runCustomer()
{
    cout << "Customer " << this->idCustomer << " need " << this->needProduction
<< endl;

    while (this->needProduction != this->takenProduction) {
        this->getProductionFromStorage();
        this->showInfo();
        cout << "Customer " << this->idCustomer << " wait for production" << endl;
        this_thread::sleep_for(chrono::seconds(5));
    }

    cout << "Customer " << this->idCustomer << " got all production" << endl;
}

void Customer::showInfo()
{
    cout << "**************************************" << endl;
    cout << "*Customer " << this->idCustomer << endl;
    cout << "*Need     " << this->needProduction << endl;
```

```
    cout << "*Take     " << this->takenProduction << endl;
    cout << "****************************************" << endl;
}
```

Листинг описание класса склад

```
class Room
{
    private:
        int idRoom_;
        int roomCapacity_;

    public:
        Room(int id);
        ~Room();

        void showInfo();

        int checkProduction(int needProduction);
        int getProduction(int needProduction);
        void fillRoom(int production);
};

class Storage
{
    private:
        vector<Room*> rooms_;
        mutex loadingAreaMutex_;

    public:
        Storage(int rooms);
        ~Storage();

        void lockLoadingArea();
        void unlockLoadingArea();

        int getProduction(int needProduction);
        void fillRooms(int production);

        void showStorage();

};
```

Результат работы

```
usage: ./main [storage N] [custome K] [time to live T]
ubuntu@ubuntu-VB:~/DCS/lab3/base$ ./main 10 3 30
storage     : 10
customer    : 3
time to live: 30
+--------------------+
| Storage:           |
+--------------------+
| +----------------
| |Room     : 0
| |Capacity : 28
| +----------------
```

```
| +----------------
| |Room     : 1
| |Capacity : 29
| +----------------
| +----------------
| |Room     : 2
| |Capacity : 32
| +----------------
| +----------------
| |Room     : 3
| |Capacity : 32
| +----------------
| +----------------
| |Room     : 4
| |Capacity : 33
| +----------------
| +----------------
| |Room     : 5
| |Capacity : 33
| +----------------
| +----------------
| |Room     : 6
| |Capacity : 33
| +----------------
| +----------------
| |Room     : 7
| |Capacity : 33
| +----------------
| +----------------
| |Room     : 8
| |Capacity : 33
| +----------------
| +----------------
| |Room     : 9
| |Capacity : 34
| +----------------
+---------------------+
Customer 0 need 512
Customer 2 need 829
Customer 1 need 991

...
```

```
Customer 2 wait for production
Customer 1 wait for production
Customer 0 wait for production

...

**************************************
*Customer 0
*Need     512
*Take     329
**************************************

Customer 0 wait for production

...

*Customer 1
*Need     991
*Take     991
**************************************

*Customer 2
*Need     829
*Take     0
**************************************

Customer 2 wait for production
**************************************

...

Customer 0 wait for production
Customer 1 got all production

*** Loader ***
+--------------------+
| Storage:           |
+--------------------+
| +----------------
| |Room     : 0
| |Capacity : 100
| +----------------
| +----------------
```

```
| |Room     : 1
| |Capacity : 100
| +----------------
| +----------------
| |Room     : 2
| |Capacity : 100
| +----------------
| +----------------
| |Room     : 3
| |Capacity : 100
| +----------------
| +----------------
| |Room     : 4
| |Capacity : 100
| +----------------
| +----------------
| |Room     : 5
| |Capacity : 100
| +----------------
| +----------------
| |Room     : 6
| |Capacity : 100
| +----------------
| +----------------
| |Room     : 7
| |Capacity : 100
| +----------------
| +----------------
| |Room     : 8
| |Capacity : 100
| +----------------
| +----------------
| |Room     : 9
| |Capacity : 100
| +----------------
+---------------------+

...

Customer 0 got all production
Customer 2 got all production
```

```
                        *** Loader ***

                        ...

                        signalhandler
```

Основные задания

Доработайте программу умножения матриц из лабораторной работы № 2 с наилучшим способом обхода оперативной памяти так, чтобы использовалось автоматическое распараллеливание циклов for. Продемонстрируйте, что результат умножения матриц получился правильным. Оцените получившееся ускорение выполнения программы.

Листинг умножение матриц

```
void multiplyMatrix(vector<vector<int>> &a, vector<vector<int>> &b, int m, int n,
int q)
{
    //cout << "The new matrix is:" << endl;

    for (auto i = 0; i < m; ++i) {
        for (auto k = 0; k < q; ++k) {
            for (auto j = 0; j < n; ++j) {
                c1[i][j] = c1[i][j] + (a[i][k]*b[k][j]);
            }
        }

    }
}

void multiplyMatrixOMP(vector<vector<int>> &a, vector<vector<int>> &b, int m, int
n, int q)
{
    int i, j, k;

    #pragma omp parallel shared(a,b,c2) private(i,j,k)
    {
        #pragma omp for schedule(static)
        for (i = 0; i < m; ++i) {
            for (k = 0; k < q; ++k) {
                for (j = 0; j < n; ++j) {
                    c2[i][j] = c2[i][j] + (a[i][k]*b[k][j]);
                }
            }
        }
    }
}

int verfication(int m, int q)
{
    for (auto i = 0; i < m; ++i) {
        for (auto k = 0; k < q; ++k) {
            if (c1[i][k] != c2[i][k]) {
                std::cout << __FILE__<< " c1[" << i << "][" << k << "] != c2[" <<
i << "][" << k << "] " << c1[i][k] << " != " << c2[i][k] << std::endl;
                return 0;
            }
        }
```

```
    }
    return 1;
}
```

Результат работы

```
row_row.cpp multiplyMatrix() elapsed time   : 31.2746 seconds.
row_row.cpp multiplyMatrixOMP() elapsed time: 3.93512 seconds.
row_row.cpp Speedup matrix size 1024        : 7.94755
row_row.cpp omp_get_num_threads()           : 8
row_row.cpp verfication pass
```

Задания повышенной сложности

Спроектируйте и разработайте параллельное приложение, реализующее игру «крестики-нолики».

- Одна нить отвечает за интерактивное взаимодействие с пользователем. Ожидает ввод с клавиатуры определённых клавиш (остальные клавиши игнорируются). Если пользователь нажимает клавишу S, то нить сообщает второй нити, что можно начать или прекратить «играть» (см. ниже).
- Если нажата клавиша T, то пользователю предлагается ввести целое число в диапазоне от 1 до 10. После ввода первая нить меняет значение общей переменной timeThink.
- Если пользователь нажимает клавишу A, то ему предлагается ввести целое число в диапазоне от 5 до 30, которое записывается в общую переменную timeRestart.
- Вторая нить реализует имитацию игры «Крестики-нолики» между двумя игроками. Каждый игрок «думает» в течение времени timeThink и делает свой ход случайным образом в любую свободную ячейку.
- Игра продолжается до тех пор, пока какой-то из игроков не выиграет или не будет заполнено все поле.
- После окончания игры происходит перезапуск после ожидания timeRestart секунд.

```
ubuntu@ubuntu-VB:~/DCS/lab3/extended$ ./main
*** default running ***
Game started think time = 5, restart time 1
s
Changing game state: GAME_START
Player 1 - make move
. . .
. x .
. . .
Player 2 - make move
. . .
o x .
. . .
Player 1 - make move
. . .
o x .
. . x
```

```
                    Player 2 - make move
                    o . .
                    o x .
                    . . x
                    Player 1 - make move
                    o . .
                    o x .
                    x . x
                    Player 2 - make move
                    o o .
                    o x .
                    x . x
                    Player 1 - make move
                    o o x
                    o x .
                    x . x
                    Player 2 - make move
                    o o x
                    o x o
                    x . x
                    Player 1 - make move
                    o o x
                    o x o
                    x x x
                    3 X won!
                    Game started think time = 5, restart time 1
                    s
                    Changing game state: GAME_STOP
                    q
                    Game close...
                    ubuntu@ubuntu-VB:~/DCS/lab3/extended$
```

Пример работы игры крестики нолики

Листинг main функции

```
int main(int argc, char const *argv[])
{
    GlobalVariables parametrs;

    thread key  (createAndRunKeyReader, &parametrs);
    thread game (createAndRunKeyGame, &parametrs);

    key.join();
    game.join();

    return 0;
}
```

Листинг метода класса, считывающего нажатия клавиш

```
void createAndRunKeyReader(GlobalVariables *parametrs)
{
    KeyBoardReader keyBoardReader(parametrs);

    while (parametrs->getGameState() != GAME_END) {
        char c = getchar();

        switch (c)
        {
            case 's':
                cout << "Changing game state: ";
                keyBoardReader.readKey('s');
                break;
```

```
            case 'q':
                keyBoardReader.readKey('q');
                break;

            case 'a':
                cout << "Changing time to restart" << endl;
                keyBoardReader.readKey('a');
                break;

            case 't':
                cout << "Changing time to think" << endl;
                keyBoardReader.readKey('t');
                break;

            default:
                break;
        }
    }

    cout << "Game close..." << endl;
}
```

Листинг описания класса глобальных переменных

```
enum {
    GAME_START = 0,
    GAME_STOP  = 1,
    GAME_END   = 2
};

class GlobalVariables
{
    private:
        int gameState_;
        int timeToRestart_;
        int timeToThink_;

        std::mutex mutex_;
    public:
        GlobalVariables();
        ~GlobalVariables();

        void setGameState(int newState);
        void setTimeToRestart(int newTime);
        void setTimeToThink(int newTime);

        int getGameState();
        int getTimeToRestart();
        int getTimeToThink();
};
```

Листинг описание класса симулирующего игру в крестики нолики

```
class TicTacToySimulate
{
    private:
        char gameFiled_[3][3];
        GlobalVariables *parametrs_;
    public:
        TicTacToySimulate(GlobalVariables *parametrs);
        ~TicTacToySimulate();

        void runGame();
        void runGame_(int think, int restart);
```

```
        void printGameFiled();
        void makeMove(char player);
        int checkWin();
};
```