Точилина Полина БПИ199.

Вариант 25

Первая задача об Острове Сокровищ (с Openmp)

Шайка пиратов под предводительством Джона Сильвера высадилась на берег Острова Сокровищ. Не смотря на добытую карту старого Флинта, местоположение сокровищ попрежнему 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.

Используемая модель

Источник информации

В модели делегирования есть два типа потоков: 1 - управляющий, 2 - рабочий. Управляющий поток создаёт рабочие и раздаёт им задания.

Для взаимодействия между потоками использованы mutex (чтобы ограничивать одновременный доступ) и conditional variable (чтобы уведомлять потоки о новых задачах и прогбуждать их от ожидания).

Код программы

includes

```
#include <iostream>
#include <string>
#include <vector>
#include <time.h>
#include <queue>
#include <algorithm>
#include <omp.h>
#include <thread>
```

Глобальные переменные

```
// высота острова
int fieldHeight;
// ширина острова
int fieldWidth;
// номер ячейки с сокровищем
int treasureNumber;
// Найден ли клад
bool found = false;
// группа нашедшая клад
int foundGroup;
// группы пиратов
std::vector<int> groups;
// задания для потоков
std::vector<int> tasks;
std::queue<int> freeThreads;
// остров
std::vector<std::string> island;
```

Считывание целочисленных значений

Переписан stoi, чтобы не было исключений

Написан метод, который считывает числа в интервале и сравнивает stoi с начальной строкой, чтобы не пропускать строки, которые просто начинаются с числа.

```
bool stoi(const std::string& str, int* p_value, std::size_t* pos = 0, int base = 10) {
    // обертка чтобы не было исключений
    try {
        *p_value = std::stoi(str, pos, base);
        return true;
    }
    catch (const std::exception& e)
    {
        return false;
    }
}

int readInt(int min = INT_MIN, int max = INT_MAX) {
    int result;
    bool correctInput = false;
    std::string tmp;
    std::cin >> tmp;
    correctInput = stoi(tmp, &result);

    while (!correctInput || std::to_string(result) != tmp || result < min || result > max) {
        std::cout << "Некорректный ввод. Попробуйте ещё раз\n";
        std::cin >> tmp;
        correctInput = stoi(tmp, &result);
    }

    return result;
}
```

Вывод информации об острове

Данный метод используется, чтобы вывести все ячейки острова с текущей информацией о том, исследованы ли они

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

Приветствие пользователя и установка начальных значений

Данный метод объясняет пользователю суть программы и просит ввести его начальные данные для работы программы

Генерируется рандомное место сокровища на острове

```
void setup() {
    // добавляем еще в векторы главный поток, чтобы сравнять индексацию потоков и ячеек векторов
    groups.push_back(0);
    tasks.push_back(0);

    std::cout << "Джон, мы видим остров!\n"
        << "Однако он слишком большой, а карта не даёт четких указаний о местоположении сокровищ.\n"
        << "Я предлагаю разделить остров на секции и пиратов на группы.\n"
        << "(Один пират исследует один участок за 5 секунд. Группа из n пиратов работает в n раз быстрее)\n\n";

    std::cout << "Введите высоту острова (> 0):\n";
    fieldHeight = readInt(1);

    std::cout << "Введите ширину острова (> 0):\n";
    fieldWidth = readInt(1);

    std::cout << std::endl;

    for (int i = 0; i < fieldHeight * fieldWidth; i++) island.push_back("_");
    outIsland();
    std::cout << std::endl;

    std::cout << "Введите количество пиратов (> 0):\n";
    int people = readInt(1);
    int freePeople = people;
    std::cout << std::endl;

    for (int i = 0; ; i++) {
        if (freePeople == 0)
            break;
        std::cout << "Количество свободных пиратов:  " << freePeople << std::endl;
        std::cout << "Введите количество пиратов, из которых будет сформирована группа " << i + 1 << " (0 < n <= free pirates)\n";
        int currentGroup = readInt(1, freePeople);
        groups.push_back(currentGroup);
        freePeople -= currentGroup;
    }

    std::cout << "\nГруппы пиратов сформированы:\n";
    for(int i = 1; i < groups.size(); i++)
        std::cout << groups[i] << " ";
    std::cout << " \n\nВы прибыли на остров. Поиски начинаются.\n\n"
        << "-----------------------------------------------------------------\n\n";

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

Функция потоков-рабочих

Поток ожидает новой задаче в своей ячейке вектора задач

```
void groupFunction(int members) {
    int tNumber = omp_get_thread_num();

    while (true) {
        // ожидаем поступления задания
        while (tasks[tNumber] == -1) {}

        int task = tasks[tNumber];
```

Выполнение задачи

Поток завершается, если задача = -2 (код завершения)

В ином случае поток уведомляет о начале выполнения задания

Поток засыпает на время равное 5 / members (один пират исследует ячейку за 5 секунд), симулируя выполнение задания

```
        // завершение миссии
        if (task == -2)
            return;

#pragma omp critical (cout)
        {
            std::cout << tNumber << " начала обследовать ячейку " << task << std::endl;
        }

        // выполнение задания
        std::this_thread::sleep_for(std::chrono::seconds(5 / members));
```

Обработка информации

После пробуждения поток определяет, был ли найден клад и уведомляет главный поток о том, что работа завершена

Поток выводит информацию о текущем статусе острова после заверщения миссии

```
            // нашли ли клад
            if (task == treasureNumber) {
                island[task] = "X";

#pragma omp critical (cout)
                {
                    std::cout << std::endl << tNumber << " закончила обследовать ячейку " << task << std::endl;
                    std::cout << "Капитан, мы нашли клад! Дождёмся ушедшие группы и можно отплывать.\n";
                    outIsland();
                    std::cout << std::endl;
                }
#pragma omp critical (tasks)
                {
                    foundGroup = tNumber;
                    found = true;
                    freeThreads.push(tNumber);
                }
                return;
            }

            island[task] = "O";

#pragma omp critical (cout)
            {
                std::cout << std::endl << tNumber << " закончила обследовать ячейку " << task << std::endl;
                outIsland();
                std::cout << std::endl;
            }

#pragma omp critical (tasks)
            {
                if (tasks[tNumber] != -2)
                    tasks[tNumber] = -1;
                freeThreads.push(tNumber);
            }
        }
    }
```

Функция main

Активирует setup и настраивает локаль, рандом и количество потоков

```
int main()
{
    setlocale(LC_ALL, "rus");
    srand(time(0));
    setup();
    // устанавливаем количество потоков
    omp_set_num_threads(std::min((int)groups.size(), fieldHeight * fieldWidth + 1));
```

Далее функция регулирует основной поток

Добавляем в очередь номера будущих потоков, симулирующих команды пиратов

Если введено команд больше, чем ячеек в острове, то лишние команды проигнорируются (останутся у корабля)

Распараллеливаем на количество потоков вызов функции для отработки метода группы (мастер туда не заходит)

```
    // добавляем все потоки в очередь
    for (int i = 1; i < std::min((int)groups.size(), fieldHeight * fieldWidth + 1); i++) {
        freeThreads.push(i);
        tasks.push_back(-1);
    }

#pragma omp parallel
    {
        if (omp_get_thread_num() != 0)
            groupFunction(groups[omp_get_thread_num()]);
```

В цикле для каждой ячейки острова мастер ждет свободную команду и отдает ей на выполнение данную ячейку, если клад еще не был найден

```
#pragma omp master
        {
            // для каждой ячейки острова
            for (int i = 0; i < fieldHeight * fieldWidth; i++) {
                // ждем свободного потока
                while (freeThreads.empty()) {}

                // проверяем найден ли клад
                if (found) break;

                // блокируем вывод и выводим команду.
#pragma omp critical (cout)
                {
                    std::cout << "\nКоманде " << freeThreads.front() << " назначено новое задание в ячейке " << i << "\n\n";
                }

#pragma omp critical (tasks)
                {
                    // добавляем задачу треду в очереди
                    tasks[freeThreads.front()] = i;
                    // удаляем тред из очереди
                    freeThreads.pop();
                }
            }
```

После того как был найден клад или остров закончился мастер выходит из цикла

Поток ожидает, пока не будет найден клад

(Например если была 1 команда и клад находился в последней ячейке, нам необходимо дождаться его нахождения после завершения цикла)

Далее мастер уведомляет все рабочие потоки о необходимости завершения и ожидает их

```
            while (!found) {}

#pragma omp critical (tasks)
            {
                for (int i = 0; i < tasks.size(); i++) tasks[i] = -2;
            }
        }
    }
    // так как мы вышли из блока, все потоки завершились

    std::cout << "Клад найден группой " << foundGroup << "\n";

    std::cout << "Все пираты вернулись к кораблю, можно отплывать!\n";
}
```

Тестирование программы

Протестируем проверку ввода острова

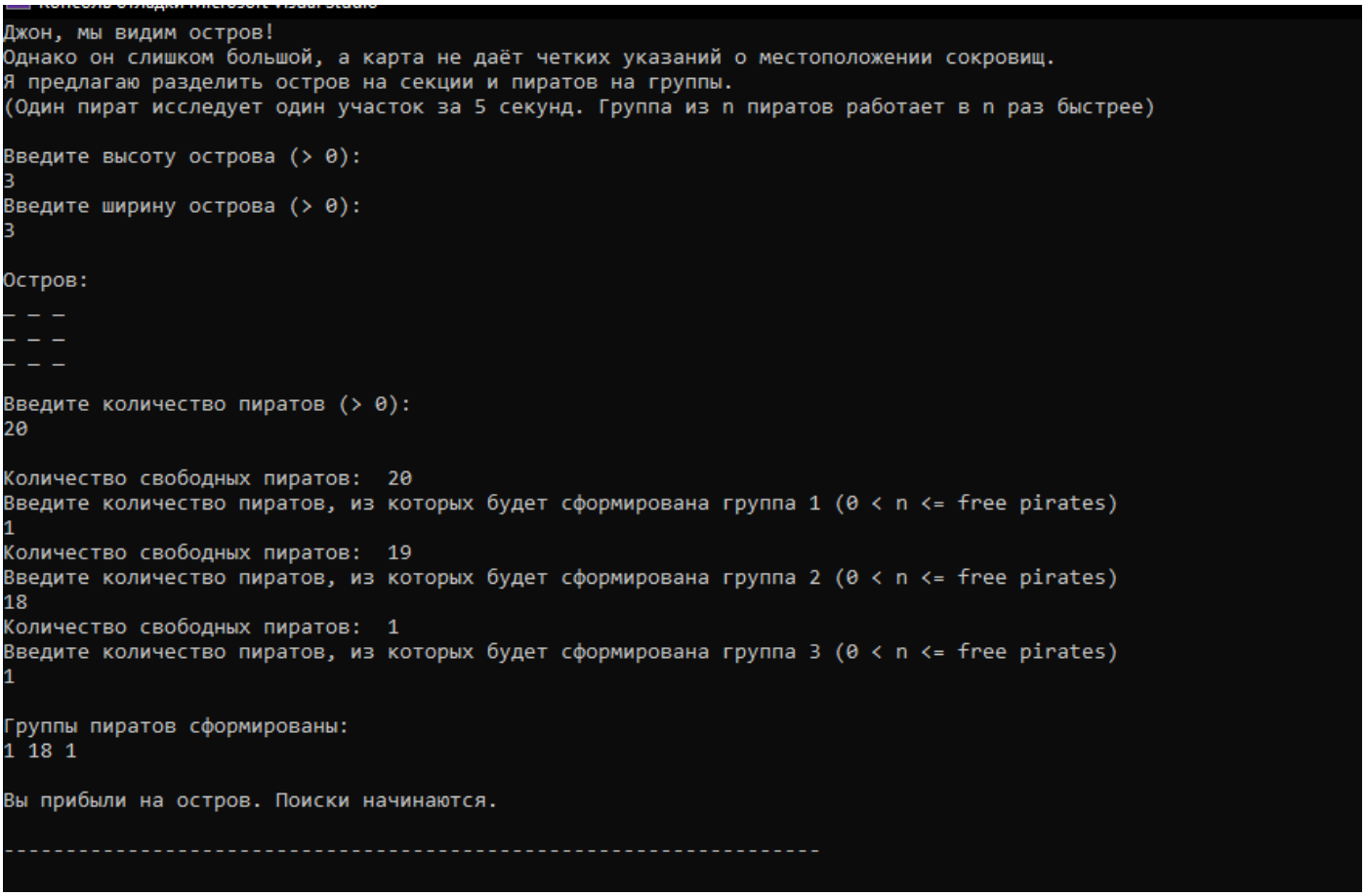
```
Джон, мы видим остров!
Однако он слишком большой, а карта не даёт четких указаний о местоположении сокровищ.
Я предлагаю разделить остров на секции и пиратов на группы.
(Один пират исследует один участок за 5 секунд. Группа из n пиратов работает в n раз быстрее)

Введите высоту острова (> 0):
-1
Некорректный ввод. Попробуйте ещё раз
sf
Некорректный ввод. Попробуйте ещё раз
0
Некорректный ввод. Попробуйте ещё раз
3.6
Некорректный ввод. Попробуйте ещё раз
10
Введите ширину острова (> 0):
10

Остров:
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
```

Протестируем проверку ввода пиартов

```
Введите количество пиратов (> 0):
-2
Некорректный ввод. Попробуйте ещё раз
0
Некорректный ввод. Попробуйте ещё раз
sg
Некорректный ввод. Попробуйте ещё раз
2.5
Некорректный ввод. Попробуйте ещё раз
20

Количество свободных пиратов:  20
Введите количество пиратов, из которых будет сформирована группа 1 (0 < n <= free pirates)
19
Количество свободных пиратов:  1
Введите количество пиратов, из которых будет сформирована группа 2 (0 < n <= free pirates)
9
Некорректный ввод. Попробуйте ещё раз
0
Некорректный ввод. Попробуйте ещё раз
1

Группы пиратов сформированы:
19 1

Вы прибыли на остров. Поиски начинаются.

----------------------------------------------------------------
```

Проверим на корретном вводе

Вводим все необходимые значения



Так как команда №1 в 18 раз больше двух других, она завершает свои задачи быстрее.

```
Команде 1 назначено новое задание в ячейке 0

1 начала обследовать ячейку 0

Команде 2 назначено новое задание в ячейке 1

2 начала обследовать ячейку 1

Команде 3 назначено новое задание в ячейке 2

3 начала обследовать ячейку 2

2 закончила обследовать ячейку 1
Остров:
_ O _
_ _ _
_ _ _


Команде 2 назначено новое задание в ячейке 3

2 начала обследовать ячейку 3

2 закончила обследовать ячейку 3
Остров:
_ O _
O _ _
_ _ _


Команде 2 назначено новое задание в ячейке 4

2 начала обследовать ячейку 4
```

После нахождения клада командой, основной поток подождал возвращения двух оставшихся и программа завершилась

```
2 закончила обследовать ячейку 4
Капитан, мы нашли клад! Дождёмся ушедшие группы и можно отплывать.
Остров:
_ O _
O X _
_ _ _


1 закончила обследовать ячейку 0
Остров:
O O O
O X _
_ _ _


3 закончила обследовать ячейку 2
Остров:
O O O
O X _
_ _ _

Клад найден группой 2
Все пираты вернулись к кораблю, можно отплывать!
```

Проверим на работе только с одной командой

Вводим все необходимые значения

```
Джон, мы видим остров!
Однако он слишком большой, а карта не даёт четких указаний о местоположении сокровищ.
Я предлагаю разделить остров на секции и пиратов на группы.
(Один пират исследует один участок за 5 секунд. Группа из n пиратов работает в n раз быстрее)

Введите высоту острова (> 0):
2
Введите ширину острова (> 0):
2

Остров:
_ _
_ _

Введите количество пиратов (> 0):
20

Количество свободных пиратов:  20
Введите количество пиратов, из которых будет сформирована группа 1 (0 < n <= free pirates)
20

Группы пиратов сформированы:
20

Вы прибыли на остров. Поиски начинаются.

-----------------------------------------------------------------
```

Поток последовательно отдает ячейки команде, пока не будет найден клад.

```
Команде 1 назначено новое задание в ячейке 0

1 начала обследовать ячейку 0

1 закончила обследовать ячейку 0
Остров:
O _
_ _


Команде 1 назначено новое задание в ячейке 1

1 начала обследовать ячейку 1

1 закончила обследовать ячейку 1
Капитан, мы нашли клад! Дождёмся ушедшие группы и можно отплывать.
Остров:
O X
_ _

Клад найден группой 1
Все пираты вернулись к кораблю, можно отплывать!
```

Проверим на ввод большего количества команд, чем есть ячеек на острове

Вводим все необходимые значения

```
Джон, мы видим остров!
Однако он слишком большой, а карта не даёт четких указаний о местоположении сокровищ.
Я предлагаю разделить остров на секции и пиратов на группы.
(Один пират исследует один участок за 5 секунд. Группа из n пиратов работает в n раз быстрее)

Введите высоту острова (> 0):
2
Введите ширину острова (> 0):
2

Остров:
_ _
_ _

Введите количество пиратов (> 0):
5

Количество свободных пиратов:  5
Введите количество пиратов, из которых будет сформирована группа 1 (0 < n <= free pirates)
1
Количество свободных пиратов:  4
Введите количество пиратов, из которых будет сформирована группа 2 (0 < n <= free pirates)
1
Количество свободных пиратов:  3
Введите количество пиратов, из которых будет сформирована группа 3 (0 < n <= free pirates)
1
Количество свободных пиратов:  2
Введите количество пиратов, из которых будет сформирована группа 4 (0 < n <= free pirates)
1
Количество свободных пиратов:  1
Введите количество пиратов, из которых будет сформирована группа 5 (0 < n <= free pirates)
1

Группы пиратов сформированы:
1 1 1 1 1

Вы прибыли на остров. Поиски начинаются.

------------------------------------------------------------------
```

Все команды кроме одного ушли выполнять задания (поток для последней команды не был создан)

```
Команде 1 назначено новое задание в ячейке 0

1 начала обследовать ячейку 0

Команде 2 назначено новое задание в ячейке 1

2 начала обследовать ячейку 1

Команде 3 назначено новое задание в ячейке 2

3 начала обследовать ячейку 2

Команде 4 назначено новое задание в ячейке 3

4 начала обследовать ячейку 3

1 закончила обследовать ячейку 0
Капитан, мы нашли клад! Дождёмся ушедшие группы и можно отплывать.
Остров:
X O
O _


2 закончила обследовать ячейку 1
Остров:
X O
O O


3 закончила обследовать ячейку 2
Остров:
X O
O O


4 закончила обследовать ячейку 3
Остров:
X O
O O

Клад найден группой 1
Все пираты вернулись к кораблю, можно отплывать!
```

/