#DEVELOBEAR2023

Правила игры

Вы - высокоразвитая колония вирусов, которая способна рассылая свои вирусы захватывать живые клетки организма. Колония умеет увеличивать свою численность, когда её вирусы находятся внутри живой клетки. Разные клетки приносят вам разное количество новых вирусов за шаг.

Однако, существуют препятствия. Каждая ранее никем не зараженная клетка имеет уровень сопротивления, обусловленное количеством ее собственных “нейтральных” вирусов. Что бы ее захватить, нужно послать на нее свои вирусы больше чем у нейтральной клетки.

Есть еще одна высокоразвитая колония вирусов, которая также пытается захватить организм. За неё играет другая команда. Вы можете захватывать не только нейтральные клетки, но и завоевывать чужие. Вирусы соперника так же как и ваши увеличиваются, за счет клеток, которые они захватили.

Игровое поле это некоторый организм с живыми клетками. Клетки могут быть нейтральные, захвачены вами или соперниками.

Изначально у вас и у соперника по одной клетке. На каждой по 100 вирусов.
Вы можете отправить на другую клетку свои вирусы, чтобы заразить соседнюю клетку.

Чтобы заразить клетку следует отправить большее количество вирусов, чем имеющиеся на ней сейчас вирусы соперника или чем уровень сопротивления клетки, если она еще никем не заражена. Например, в клетке живет 30 вирусов, прилетает 35 ваших вирусов. Клетка переходит к вам, при этом вирусов становится 5: вы потеряли в бою 30 единиц, но захватили клетку.

Если соперники приплыли к клетке одновременно, то бой сначала проходит в пространстве около клетки, потом уцелевшие проникают в клетку (это правило важно, если клетка пока еще нейтральна, то после схватки с соперником вам может не хватить вирусов для захвата клетки).
Если захватчиков приплыло столько же, сколько и вирусов в уже захваченной соперником клетке, то клетка снова становиться нейтральной, но с нулевым сопротивлением.

При захвате клетки, вирусы начинают увеличивать свою численность. Для каждой клетки заложен свой уровень прироста вирусов. Прирост происходит при каждом ходе. Клетка может вмещать в себя неограниченное количество вирусов.

На нейтральных клетка тоже есть “хорошие” вирусы, которые создают уровень сопротивления клетки к захватчикам, но количество “хороших” вирусов при каждом ходе остается неизменным.

Плавание между клетками занимает время, обусловленное разных расстоянием между клетками.

Общее количество ходов игры ограничено.

Словарь применяемых в коде терминов

Cells - живая клетка
FreeCells - никем не занятая клетка
MovingGroup - группа вирусов плывущая к клетке
Team - команда
OwnCells - живые клетки принадлежащие вам
OwnMovingGroup - группы перемещающихся вирусов принадлежащие вам
currentMovingGroups - группы вирусов, которые находятся в плавании в начале вашего хода
nextStepMovingGroups - группы вирусов, которые ваш алгоритм подготовит на этом ходу, для отправки к другим клеткам
Nearest - ближайшее
Distance - дистанция
Enemy - соперник
DistanceMap - карта расстояний
Distance - расстояние
Step - шаг
Current - текущий
maxSteps - максимальное количество шагов

Цель игры и определение победителя

К концу игры иметь суммарное количество клеток больше, чем у соперника.

Победитель - игрок с наибольшим количеством захваченных клеток. Если количество клеток совпадает, тогда победитель - игрок с наибольшим количеством вирусов (на всех клетках + захватчики в пути). Если количество вирусов тоже совпадает, тогда побеждает тот, чей алгоритм затратил меньше времени на вычисления, если и тут паритет, тогда ничья.

Игра заканчивается, когда у одного из игроков не остается ни одного вируса или при достижении максимального количества ходов.

Входные данные

На каждом ходу программе игрока выдается полная информация о текущем состоянии игры. Не нужно и даже очень опасно пытаться сохранять в глобальных переменных какие-либо данные между двумя ходами. Они могут сохраниться, а могут и не сохраниться.

Входные данные для игрока:

1. Номер игрока, за который играет программа. Для одиночных игр - всегда 0, а в турнире будет и 0, и 1.
2. Номер текущего хода и ограничение на общее количество ходов.
3. Информация о клетках: количество, идентификатор, информация о текущем владельце, количество вирусов в клетке, о скорости роста вирусов в клетке. Для указания не зараженной клетки используется -1.
4. Карта с расстояниями между клетками в виде таблицы расстояний.
5. Информация о перемещающихся группах вирусов (чьи вирусы плывут, откуда плывут, куда плывут, сколько вирусов, сколько ходов осталось плыть).

Все входные данные доступны как переменные в примере кода, который предоставляется всем участникам.

Выходные данные

Выходными данными являются команды на перемещение вирусов из одной клетки на другую.
1. Первое число - общее количество команд на перемещение.
2. Далее каждая команда на перемещение записывается в виде 3-ех чисел:

- клетка старта
- клетка назначения
- сколько вирусов послать

Особенности

Существует ограничение по времени выполнения алгоритма. Точного значения не приводим, но времени дается более чем достаточно, и носит защитный характер от зацикливания. Если алгоритм, 3 раза подряд был выключен из-за таймаута, ему засчитывается поражение.

В случае если была дана некорректная команда, она игнорируется. Например, отправили больше вирусов, чем есть в клетке.

Если алгоритм падает по ошибке, ход засчитывается, как будто игрок не стал посылать вирусы.

Задача

Каждый участник должен написать алгоритм, который отдавая команды на распространение вирусов, захватывает клетки как нейтральные, так и клетки соперника.

Игровое поле не одно и может меняться от игры к игре. Поэтому алгоритм нужно писать задавая правила, на основании состояния клеток и количества ваших вирусов. Закладывать в алгоритм знание о положении клеток нельзя, т.к. игровых полей будет несколько.

Каждой команде будет выдан проект с функциями и переменными, которые помогут вам написат.

ь код.

Для начала изучения вам нужно перейти к файлу:

Java проект: файл Handler.java,

код вашего алгоритма нужно размещать в функции makeMove

Python проект: файл game.py,

код вашего алгоритма нужно размещать в функции make_move

C++ проект: файл handler.cpp,

код вашего алгоритма нужно размещать в функции Handler::makeMove

Важно!!!

В этой функции вам и нужно размещать ваш алгоритм. Для примера уже написан работающий в первые 10 шагов алгоритм, который нацелен на захват пустых клеток. Изучите его. Попробуйте написать ваш алгоритм используя ту же концепцию, а именно только используя фильтрацию и сортировку массивов.

Запуск и отладка алгоритма

Процесс запуска алгоритма приведен на скриншотах ниже.

C, C++, Pascal

Особенность этих языков, в том, что перед первым запуском алгоритма, нужно скомпилировать проект. Также нужно компилировать проект каждый раз после внесения в его код, каких либо изменений.

Самый просто способ скомпилировать проект, нажать сочетание клавиш:

Ctrl+shift+B

Или же зайти в меню:

Menu > Terminal > Run build task ( )

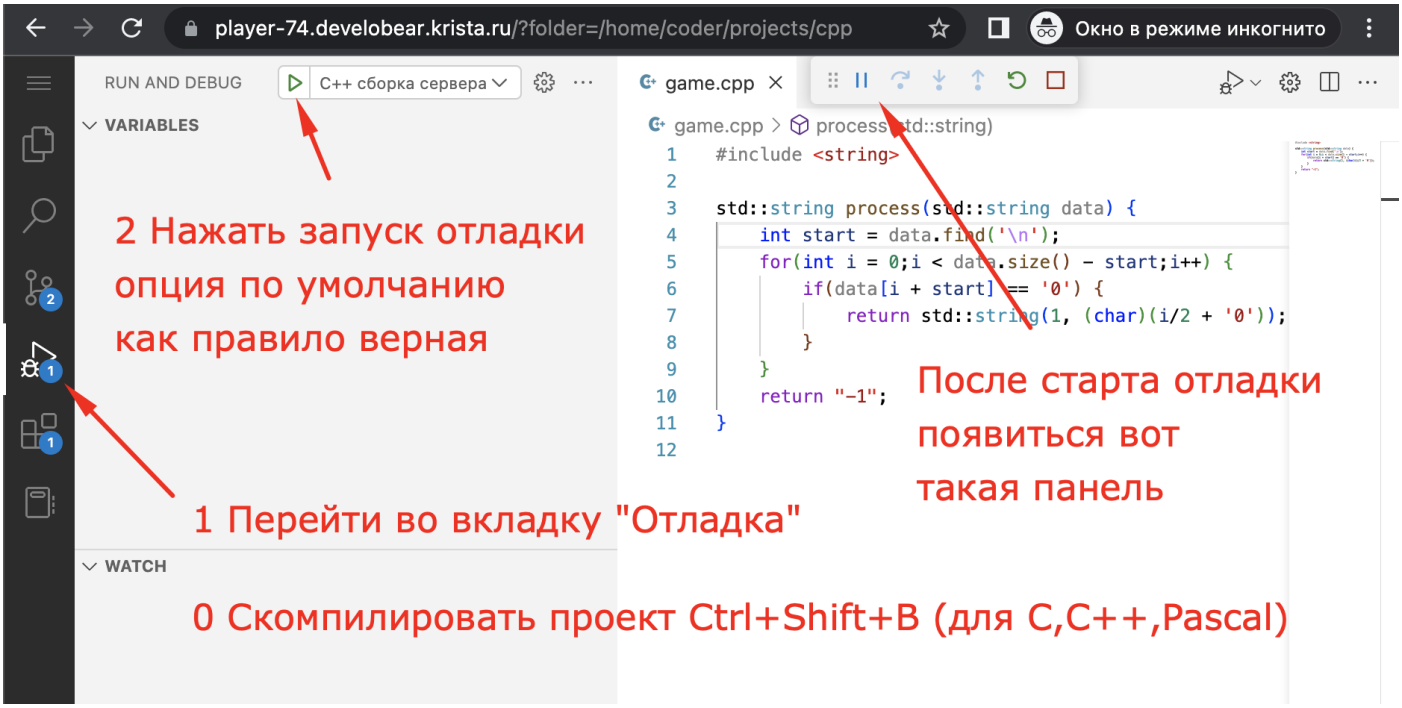
Java

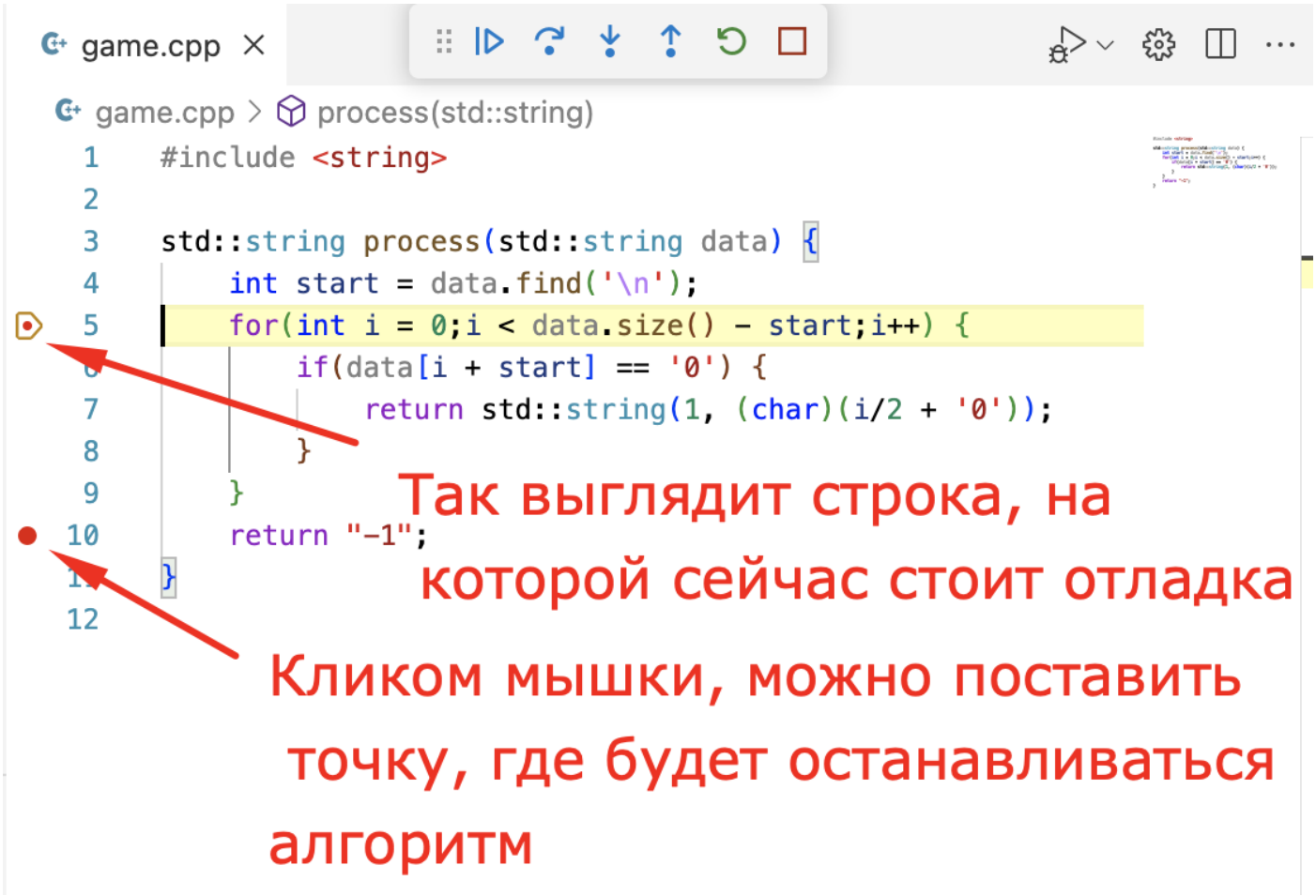
Компилировать **java** проект не надо. Но не забывайте останавливать и перезапускать проект каждый раз как были внесены изменения в код.

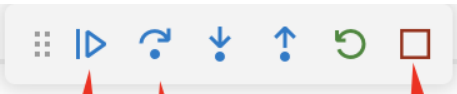
Python

Компилировать **python** проект не надо. Но не забывайте останавливать и перезапускать проект каждый раз как были внесены изменения в код.

Скриншоты


player-74.develobear.krista.ru/?folder=/home/coder/projects/cpp
Окно в режиме инкогнито
RUN AND DEBUG
C++ сборка сервера
VARIABLES
WATCH
game.cpp
game.cpp > process(std::string)
#include <string>
std::string process(std::string data) {
int start = data.find('\n');
for(int i = 0;i < data.size() - start;i++) {
if(data[i + start] == '0') {
return std::string(1, (char)(i/2 + '0'));
}
}
return "-1";
}
2 Нажать запуск отладки
опция по умолчанию
как правило верная
После старта отладки
появиться вот
такая панель
1 Перейти во вкладку "Отладка"
0 Скомпилировать проект Ctrl+Shift+B (для C,C++,Pascal)


game.cpp
game.cpp > process(std::string)
#include <string>
std::string process(std::string data) {
int start = data.find('\n');
for(int i = 0;i < data.size() - start;i++) {
if(data[i + start] == '0') {
return std::string(1, (char)(i/2 + '0'));
}
}
return "-1";
}
Так выглядит строка, на
которой сейчас стоит отладка
Кликом мышки, можно поставить
точку, где будет останавливаться
алгоритм



Остановить
отладку

Сделать шаг без захода
в функции

Продолжить актоматическое
выполнение кода

Ручное управление

Пример входных данных

```
//Номер игрока, за который играет программа.
//Для одиночных игр - всегда 0, а в турнире будет и 0, и 1.
0
//количество ходов всего
100
//номер текущего хода
2
//Количество планет
3
//Планеты: ID планеты, Номер игрока, Количество населения, Прибавление за ход
0 0 10 2
1 -1 5 3
2 1 10 2
//Количество строк и столбцов в матрице расстояний
3
//Построчная развёртка матрицы расстояний между планетами в ходах
0 4 8
4 0 4
8 4 0
//Количество перемещающихся групп
2
// Перемещающиеся поселенцы:
// Номер игрока, Планета с которой летят, Планета на которую, Сколько поселенцев,
Сколько ходов осталось лететь
0 0 1 5 3
1 2 1 10 3
```

Выходные данные

Выходными данными являются команды на перемещение поселенцев.
1. Первое число - общее количество команд на перемещение.
2. Далее каждая команда на перемещение записывается в виде 3-ех чисел:
   - планета старта
   - планета назначения
   - сколько колонистов

Примеры выходных данных

Нет команд на перемещение в текущем ходу:

```
0
```

Одна команда на перемещение 5-ти колонистов с планеты 0 на планету 1.

```
1
0 1 5
```

Три команды на перемещение
- 5-ти колонистов с планеты 0 на планету 1.
- 15-ти колонистов с планеты 2 на планету 3.
- 10-ти колонистов с планеты 0 на планету 2.

```
3
```

```
0 1 5
2 3 15
0 2 10
```