Урок 1

Крестики-нолики. Разработка алгоритма

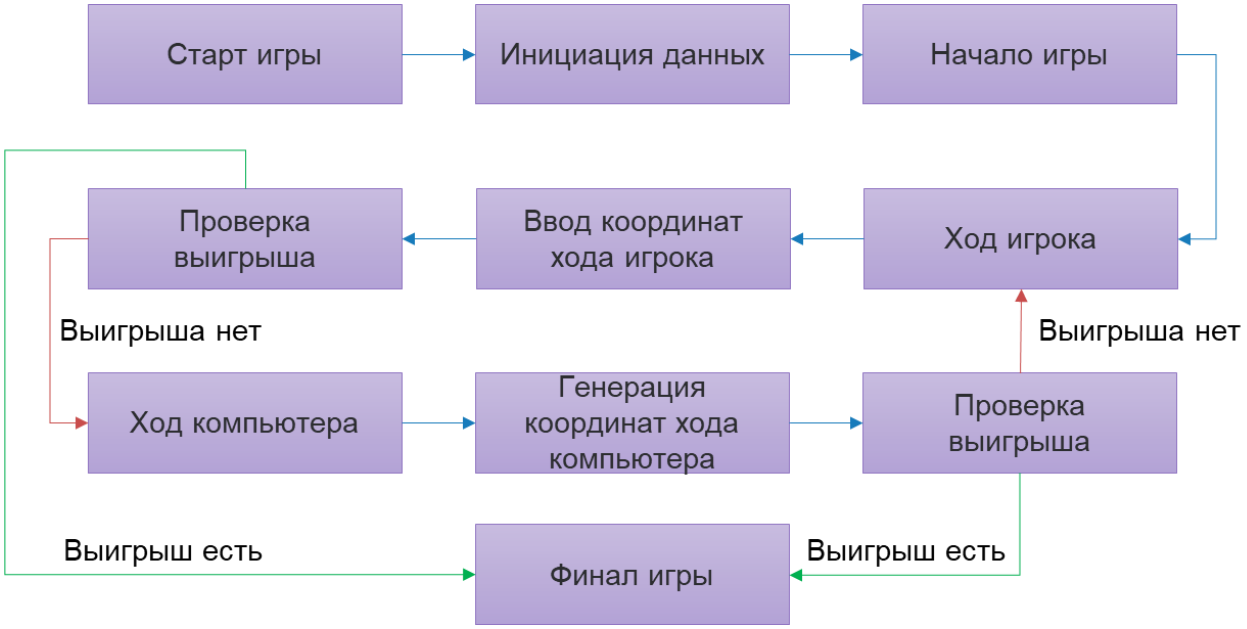
Схема алгоритма. Набор методов.

Начало разработки

Начало разработки

Большинство игр подчиняются строгому алгоритму. Он ложится в основу кода при программировании игры для ее реализации на компьютере. Чтобы разработка была прозрачной, надо четко сформулировать принцип работы и последовательность действий.

Изобразим наш алгоритм в виде пошаговой схемы:



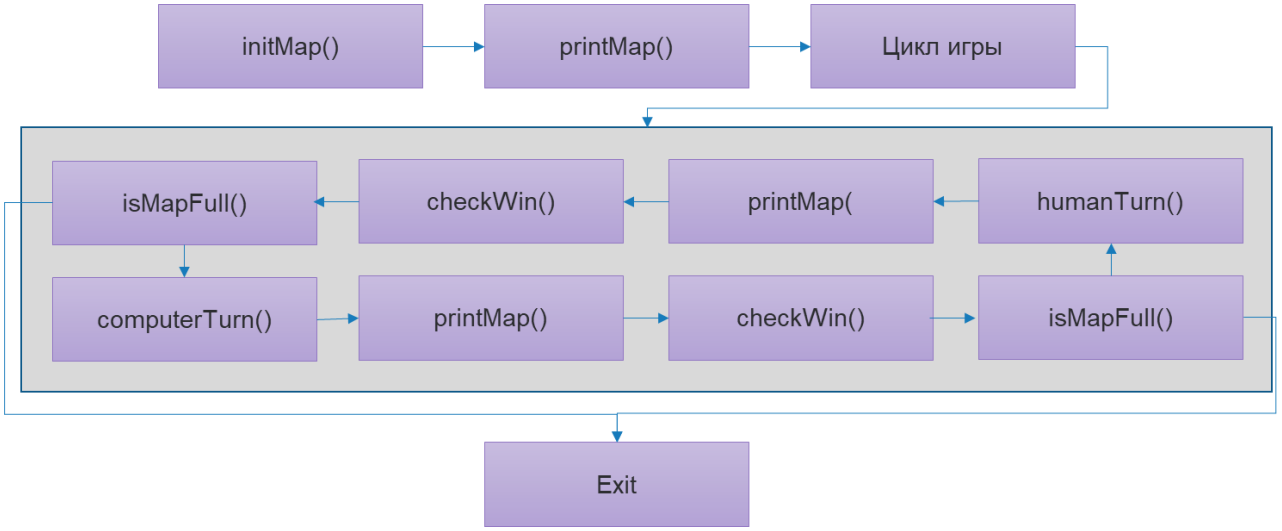
Начинаем с запуска игры — по сути, это метод main в главном классе программы. Затем создаем поле игры, ее состояние, участников. Только после этого можно стартовать — предлагать первому игроку сделать ход в выбранную им ячейку.

Сам ход тоже будет состоять из нескольких шагов: считывания координат, проверки доступности ячейки и обновления поля. Затем нужно проверить поле на наличие выигрышной комбинации или ничьей. После этого передать ход другому игроку (компьютеру) или завершить игру. Видим, что она состоит из циклов, а каждый из них составляют фазы — ходы игроков.

Схема также показывает, что каждый ход будет менять у поля состояние, которое надо где-то хранить.

Проверку выигрыша для компьютера и реального игрока нет смысла разделять, так как логика аналогична — разница только в проверяемом символе игрока. Поле не бесконечно — игра кончится, когда будет заполнена последняя ячейка.

Ввиду этих фактов преобразуем схему в набор методов:



Игровое поле будет хранить состояние — **map**. Надо выводить его на экран в начале и конце игры, а также на каждой фазе ходов.

Если посмотреть на классическое игровое поле, сразу становится понятно, что состояние нужно хранить в виде двумерного массива размерностью 3 на 3 с типом **char**. Тогда ход игрока будет вводом двух координат выбранной ячейки — по горизонтали и вертикали. Для поля 3 на 3 будет 8 победных комбинаций.

В реализации игры помогут вспомогательные сущности — классы **Scanner** и **Random**. Первый будет обеспечивать удобную работу с пользовательским вводом координат, а второй — генерировать случайные числа для игры с простой версией противника-компьютера, который ходит в случайную незанятую ячейку.