Вариант 80 (***)

Разработать систему для управления роботом, осуществляющим передвижение по лабиринту. Ячейка лабиринта имеет форму квадрата.
Робот может передвинуться в соседнюю ячейку в случае отсутствия в ней препятствия. Роботу известны координаты выходов из лабиринта, но известен маршрут до них.

1. Разработать формальный язык для описания действий клеточного робота с поддержкой следующих литералов, операторов и предложений:

- Знаковых целочисленных литералов в десятичном формате;
- Логических литералов **istinno** и **lozhno**; логические константы и выражения преобразуются к знаковым целочисленным как 1 и 0 соответственно, 0 преобразуется в **lozhno**, любое другое число в **istinno**;
- Объявление переменных/констант в форматах:
  - Целочисленная переменная со знаком **tseloye <имя идентификатора>[= <арифметическое выражение>]**;
  - Логическая переменная **logicheskoe <имя идентификатора > [= <логическое выражение>]**;
  - Квадратная ячейка пространства **yacheyka <имя идентификатора > = {<арифметическое выражение координата X>, <арифметическое выражение координата Y>, <логическое выражение признак занятости препятствием>}**;
  - к свойствам ячейки можно обратиться при помощи оператора <**имя переменной ячейки**>=><**поле ячейки**>;
  - Переменная массив **massiv <имя идентификатора> = {<арифметическое выражение размерность в пространстве 1>[, <арифметическое выражение размерность в пространстве 2>,…]}**; в одном массиве могут храниться элементы разных типов (в том числе массивы);

- Обращение к элементу массива
  - **<имя идентификатора> [<арифметическое выражение индекс 1, [арифметическое выражение индекс 2,…)]]** – получение элемента массива с заданными координатами;
- Оператор получения размерности массива
  - **razmer <имя идентификатора массива>**

Применяется строгая типизация, если преобразование не определено и типы не совпадают, то это семантическая ошибка.

- Оператор сравнения типов **proverka {<имя идентификатора / элементы массива / имя типа> <имя идентификатора / элементы массива / имя типа>}** ; если типы совпадаю возвращает **istinno**, иначе **lozhno**.
- Операторов присваивания ‘=’;
- Арифметических операторов:
  - **<арифметическое выражение> + <арифметическое выражение>**
  - **< арифметическое выражение > - < арифметическое выражение >**
- Логических операторов (результат логическое выражение):
  - ~ **<логическое выражение >** (отрицание) ;
  - **<логическое выражение > && <логическое выражение>** (конъюнкция);
  - **<логическое выражение > || <логическое выражение>** (дизъюнкция);
  - Операторов сравнения:
  - **<арифметическое выражение> < <арифметическое выражение>**;
  - **<арифметическое выражение> > <арифметическое выражение>**;
- Операторов цикла
  - **tsikl <идентификатор скаляр> = <арифметическое выражение скаляр 1>: <арифметическое выражение скаляр 2> nachalo <предложения языка > konets**

(выполнение тела цикла с изменением идентификатора скаляра от значения выражения 1 до значения выражения 2; выражения вычисляются при инициализации цикла)

- Условных операторов **esli <логическое выражение> nachalo <предложение языка / группа предложений> konets**;
- Операторов управления роботом
  - перемещения робота на одну клетку в заданном направлении относительно текущего **idi->pered** – вперед, **idi->zad** – назад, **idi->levo** – влево, **idi->pravo** – вправо; если робот сталкивается с препятствием, то он ломается.
  - поворота робота относительно текущего направления, **poverni->levo** – влево, **poverni->pravo** – вправо; разворот назад и поворот вперед не определены и являются ошибками.
  - Измерение расстояния до первого препятствия в заданном направлении **smotri->pered**– вперед, **smotri->zad** – назад, **smotri->levo** – влево, **smotri->pravo** – вправо;
  - Действия робота могут задаваться последовательностью операторов разделенных символом **->**, если в последовательности встречаются операторы получения информации о расстоянии до препятствий, то группа операторов возвращается матрицу ячеек с относительно последнего местоположения робота.
  - Робот может взлететь и перелететь через одно препятствие оператор **vzleti-><оператор управления роботом>**, когда робот взлетел оператор измерения расстояния до первого препятствия игнорирует их в ближайших клетках; пример использования **vlzeti->smotri->pered**, **vzleti->idi->pravo**, после выполнения действия в воздухе робот автоматически преземляется**.**
  - Вместо слов **pered**, **zad**, **levo**, **pravo** могут использоваться их аналоги с приставкой **v** и **na** (**napered**, **vzad**, **nalevo**, **vpravo** и т.п.)
    - **<оператор управления роботом 1>->** [**…**]
    - Для последовательности определён оператор прерывания выполнения последовательности, если в соседней ячейке (для оператора обзора) обнаружено препятствие - **ostanov**
      - Пример: **idi->pered->smorti->pered->ostanov->idi-levo -** переместиться вперед; посмотреть вперед; остановиться, если в соседней клетке препятствие, иначе продолжить выполнение программы; переместиться влево.
  - Оператор получения текущего местоположения робота – **gdeya?**; возвращает переменную ячейку с координатами.
- Описатель функции
  - **[<тип возвращаемого значения 1> funktsiya <имя функции> ([<тип переменной 1> <имя переменной 1>** [**,…**]**) nachalo предложения языка konets**. Функция является отдельной областью видимости, параметры передаются в функцию по значению; возвращаемые значения передаются из функции по значению. Функция может быть объявлена в любом месте программы, в т.ч. внутри другой функции (в этом случае она является локальной функцией, для той в которой она объявлена), при объявлении она не выполняется.
- Оператор вызова функции
  - **<имя функции> (<выражение 1>,…),** вызов процедуры может быть в любом месте программы (в том числе до ее объявление).

Предложение языка завершается символом ‘;’. Язык является регистрозависимым,

2. Разработать с помощью flex и bison интерпретатор разработанного языка. При работе интерпретатора следует обеспечить контроль корректности применения языковых конструкций (например, инкремент/декремент константы); грамматика языка должна быть по возможности однозначной.

3. На разработанном формальном языке написать программу для поиска роботом выхода из лабиринта. Описание лабиринта и начальное положение робота задается в текстовом файле.