Вариант 61 (***)

Разработать систему для управления клеточным роботом, осуществляющим передвижение по клеточному лабиринту. Клетка лабиринта имеет форму квадрата.
Робот может передвинуться в соседнюю клетку в случае отсутствия стенки между клетками.

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

- Знаковых и беззнаковых целочисленных литералов в десятичном формате, по умолчанию литералы знаковые, для объявления беззнаковых используется суффикс '**u**';
- Объявление переменных и констант в форматах:
  - Целочисленная переменная со знаком [**const**] **signed <имя переменной>** [**<- <арифметическое выражение>**]; const – указание на то, что идентификатор является константой, поле инициализации при этом обязательно;
  - Целочисленная переменная без знака [**const**] **unsigned <имя переменной>** [**<- <арифметическое выражение>**];
  - Элемент карты мира [**const**] **cell** [**<- (top|ntop, right|nright, down|ndown, left|nleft)**]; top|ntop,.. указывают на наличие или отсутствие соответствующей стенки, указываются в любом порядке, могут отсутствовать, в этом случае считается, что наличие соответствующей стенки не изменяется, по умолчанию стенки отсутствуют; дублирование противоречивых указаний на наличие стенки в одном кортеже недопустимо и является семантической ошибкой.
  - Объявление двумерных матриц **matrix** <**тип_элемента**> <**имя переменной**> (**максимальный индекс по горизонтали**, **максимальный индекс по вертикали**) | **matrix** <**тип_элемента**> <**имя переменной**>; индексация начинается с 0u;
- Доступ к элементу матрицы <**имя массива**> (**индекс по горизонтали**, **индекс по вертикали**); индексация начинается с 0u;

Применяется преобразование типов между signed и unsigned, если нельзя выполнить подобное преобразование (выход за разрядную сетку), то это считается ошибкой времени выполнения и программа должна аварийно завершиться; при вычислениях при смешении операндов выполняется продвижение к первому операнду; корректность инициализации констант и переменных проверяется на этапе первичного сканирования (компиляции / при построении внутреннего представления для интерпретации). Для ячеек определено преобразование в целочисленные типы, следующим образом cell -> signed,unsigned: 0, если нет стенок; 1, если есть хотя бы одна стенка; signed, unsigned -> cell: ячейка без стенок, если 0; ячейка полностью ограниченная стенками, если значение отлично от 0. Преобразование для матрицы в другие типы не определено и является семантической ошибкой;

- Операторов присваивания '<-';
- Арифметических бинарных операторов сложения, вычитания, умножения, целочисленного деления, получения остатка от деления (+, -, *, /, %); деление и остаток от деления на 0 приводят к ошибке времени выполнения (корректность выражений с нулевыми константными литералами проверяется на этапе первичного сканирования); операторы для матриц выполняются на поэлементной основе, в рамках минимально допустимых значений индексов; для ячеек определены операторы -,/, % выполняются по семантике исключающего или, + -или, *- и над состояниями наличия/отсутствия стен:
  - **< арифметическое выражение> оператор < арифметическое выражение>**
- Операторов сравнения (=, <, >), возвращают 1 при выполнении условия и 0 при не выполнении, сравнение для матриц и ячеек не опредлено:
  - **<арифметическое выражение>оператор <арифметическое выражение>**;
  - **<арифметическое выражение> оператор <арифметическое выражение>**;

  (приоритет операторов в порядке убывания (*,/,%),(<,>,=),(+,-) могут применятся операторные скобки '(' и ')', для переопределения порядка вычисления операторов в выражениях).
- Унарный оператор # для матриц, для целочисленных элементов вычисляет среднее

значение, и записывает во все элементы матрицы, для матрицы ячеек ‘объединяет’ стенки соседних ячеек, т.е. если два соседних элемента матрицы имеют разное состояние смежной стенки, то это состояние исправляется в соответствующей ячейке в пользу его наличия;, возвращает модифицированное состояние матрицы.
  - **оператор <имя переменной>**

- Объединение предложений в группы с помощью скобок **( )**;
- Операторов цикла **testrep (<арифметическое выражение>) <предложение языка / группа предложений>**, выполняется тело цикла до тех пор, пока выражение в условии отлично от 0.
- Условных операторов **testonce (арифметическое выражение) <предложение языка / группа предложений>,** выполняется тело оператора, если арифметическое выражение в условии отлично от 0;
- Операторов управления роботом
  - перемещения робота на одну клетку в заданном направлении **top, bottom, left, right**. Если оператор невозможно выполнить из-за наличия препятствия, то это ошибка времени выполнения (робот разбивается об стенку).
  - Осмотр ближайших окрестностей с помощью рентгеновского аппарата **xray**, возвращает матрицу размером 5x5 с состоянием соседних с роботом ячеек, робот находится в ячейке (3,3).
- Описатель функции
  - **func <имя функции> ([<тип параметра> <имя параметра>,..]) группа предложений языка**. Функция является отдельной областью видимости, параметры передаются в функцию по значению. Возвращаемым значением является результат последнего предложения в теле функции. Объявление функций внутри других функций не допустимо. Точкой входа в программу является функция с именем **start**.
- Оператор вызова процедуры
  - **call имя функции (**имена переменных разделенных пробелом**),** вызов функции может быть в любом месте программы.

Предложение языка завершается символом ‘**;**’. Язык является регистрозависимым, т.е. INT <> int <> InT <> …

2. Разработать с помощью flex и yacc 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.