Вариант 67 (***)

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 в ней препятствия; препятствиями могут являться как стены, так и коробки, которые робот может передвигать (одновременно только одну)

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

- Знаковых целочисленных литералов в десятичном формате (INT); программа принимает также вещественные значения в классическом и научном форматах, но преобразует их отбрасыванием дробной части в целочисленный формат (5.75 == 5; 5.75E2 = 575).
- Логических литералов **true** и **false** (BOOL); логические константы и выражения преобразуются к знаковым целочисленным как 1 и 0 соответственно, целочисленные к логическим 0 – false, все остальное – true;
- Тип описатель клетки CELL = {EMPTY, WALL, BOX, EXIT, UNDEF}; может преобразовываться в логический тип: если состояние клетки определено - true, иначе false (возможно дальнейшее продвижение в целочисленный тип по правилам логического типа); преобразование в CELL возможно только значений false и 0;
- Блок объявления переменных и констант в соответствующих форматах:
  - Переменные предварительно не объявляются, тип переменной определяется при первой инициализации и в дальнейшем не меняется; переменные являются массивами произвольной размерности; размерность определяется при инициализации, в дальнейшем может быть расширена.

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

- Обращение к переменной **<имя переменной>[(индекс в размерности 1, индекс в размерности 2,..)]**
  - индекс – любое целое число (может быть отрицательным)
  - если индекс не указан, то считается, что обращаемся к элементу с индексом 0;
  - в разных элементах массива могут храниться значения разных типов;
  - обращение к неопределенной переменной, либо по неопределенному индексу является ошибкой времени выполнения.
- Операторов присваивания;
  - Присваивание на уровне элементов массива
    <имя переменной>[(индекс в размерности 1, индекс в размерности 2,..)] <= <выражение>
  - Присваивание на уровне массива
    <имя переменной> <= <имя переменной>
- Бинарных и унарных арифметических операторов:
  - **<арифметическое выражение 1> + < арифметическое выражение 2>**
  - **< арифметическое выражение 1> - < арифметическое выражение 2>**
  - **- <арифметическое выражение 1>**
- Бинарных и унарных логических операторов:
  - **<логическое выражение 1> & <логическое выражение 2>**
  - **<логическое выражение 1> | <логическое выражение 2>**
  - **- <логическое выражение 1>**
- Операторов вхождения во множество и сравнения (результат логическое значение)
  - **<выражение> in <переменная>**
  - **<переменная> all in <переменная>**
  - **<переменная> some in <переменная>**
  - **<выражение> less <переменная>**

  - **<переменная> all less <переменная>**
  - **<переменная> some less<переменная>**

- Оператора цикла
  - **from <выражение1> to <выражение2> [with step <выражение3>] do <функция>**
  - оператор выполняет тело цикла, пока значение в выражении 1 не совпадает с выражением 2 с шагом определяемым выражением 3
  - переменные, используемые в выражениях, могут модифицироваться в теле цикла (см. про области видимости)
- Условных операторов **if <логическое выражение> do <функция>**;
- Операторов управления роботом
  - Оператор перемещения роботом **go left**, **go right**, **go up**, **go down**, если робот сталкивается со стеной или коробкой, то оператор возвращает false, иначе true.
  - Операторы поднять/опустить коробку **pick** / **drop left**,…; если операцию невозможно выполнить, то оператор возвращает false, иначе true.
  - Оператор осмотра окрестностей **look left**,…; оператор возвращает переменную с заполненными клетками до ближайшего препятствия (или выхода) с указанием его типа; или неопределенно, если нет препятствия в радиусе видимости, который определятся средой (может динамически изменяться). Пример: коробка выше робота на 2 клетки робота res = look up; res(0)=EMPTY, res(1)=EMPTY, res(2) = BOX.
- Описатель функции
  - **function [<имя функции>] <предложения языка> end**. Возврат значений из функции и передача значений в функции происходит через переменные в родительской области видимости. Функция является отдельной областью видимости. Функции могут быть объявлены внутри другой функции. Переменные из дочерней области видимости не видны в родительской, если не объявлены там. Функция возвращает результат последнего предложения в теле функции (для использования в выражениях). Точкой входа в программу является функция с именем main. Функции могут быть анонимные, т.е. не иметь имени.
- Оператор вызова процедуры
  - **do <имя функции>**

  Язык является регистрозависимым. Предложения языка завершаются символом перевода строки. Если необходимо на одной строке разместить несколько предложений, то они разделяются символом ‘;’.

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.