Вариант 68 (***)

Разработать систему для управления клеточным роботом, осуществляющим передвижение по клеточному лабиринту. Клетка лабиринта имеет форму правильного шестиугольника.
Робот может передвинуться в соседнюю клетку в случае отсутствия в ней препятствия; препятствиями могут являться как стены, так и коробки, которые робот может передвигать. Робот может одновременно передвигать несколько коробок, но не более определенной массы (определяется средой выполнения).

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

- Знаковых целочисленных литералов в десятичном формате и шестнадцатеричном формате (INT). Размер числа неограничен. Дополнительно определены литералы для бесконечности **INF**, минус бесконечности **–INF** и неопределенного значения **NAN**.
- Логических литералов **T[RUE]**, **F[ALSE]**, **U[NDEF[** (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 EXIT или EMPTY => TRUE, BOX или WALL => FALSE, UNDEF ⇔ UNDEF; преобразование в INT и обратно: EMPTY ⇔ 0, WALL ⇔ INF, BOX ⇔ вес коробки, EXIT ⇔ -INF, UNDEF ⇔ NAN;
- VAR – переменная произвольного типа;
- Блок объявления переменных и констант в соответствующих форматах:
  - Переменные предварительно могут объявляться, но это не обязательно; тип переменной определяется при объявлении и первой инициализации и изменяется при присваивании значения другого типа; переменные являются одномерными массивами произвольной размерности; элемент массива может содержать ссылку на другую переменную; размерность определяется при инициализации, в дальнейшем может быть расширена.
  - Формат опционального объявления **VAR|INT|BOOL|CELL <имя переменной 1>[, имя переменной 2,…]**

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

- Обращение к переменной **<имя переменной> [(индекс)]**
  - индекс – любое неотрицательное целое число (может быть отрицательным)
  - если индекс не указан, то считается, что обращаемся к переменной в целом;
  - в разных элементах массива могут храниться значения разных типов;
  - обращение к неопределенной переменной, либо по неопределенному индексу не является ошибкой (возвращается неопределенное значение).
- Операторов присваивания;
  - Присваивание на уровне элементов массива
    <имя переменной>(индекс в размерности) := <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>**
  - **#<имя переменной>** (считает сумму во всех элементах массива)
- Бинарных и унарных логических операторов:
  - **<логическое выражение 1> ^ <логическое выражение 2>** (оператор xor)
- Операторов вхождения сравнения (результат логическое значение)
  - **<арифметическое выражение> < <арифметическое выражение>**
  - **<арифметическое выражение> > <арифметическое выражение>**

- **<выражение> = <выражение>**
- Оператора цикла
  - **while <логическое выражение> do <предложения языка 1> [finish <предложения языка 2>] done**
  - оператор выполняет тело цикла (предложения языка 1), пока значение в логическом выражении истинно, при ложном значении управление передается опциональному блоку finish, которое выполняется 1 раз; при неопределенном значении выход из цикла без захода в блок finish.
- Условных операторов
  - **if <логическое выражение> do <предложения языка 1> done [eldef do <предложения языка 2> done] [elund do <предложения языка 3> done]**;
  - если логическое выражение истинно, то выполняется блок предложений 1, если ложно, то выполняется опциональный блок 2, если не определено, то опциональный блок 3.
- Операторов управления роботом
  - Оператор перемещения роботом на заданное число клеток вперед / назад **forward <арифметическое выражение>** / **backward <арифметическое выражение>**, робот предварительно проверяет возможность выполнить команду, если это возможно – то перемещается и возвращает true, если нет – то не перемещается и возвращает false; существует вероятность (вероятность успеха определяется средой выполнения, пропорциональна расстоянию перемещения и обратно пропорциональна весу груза), что робот не сможет выполнить команду до конца – в этом случае он останавливается в соответствующей клетке и возвращает UNDEF;
  - Операторы поворота на 60 градусов **left**, **right**; робот не может повернуться, если вес груза превышает некоторое граничное значение.
  - Операторы загрузить/выгрузить коробку **load <арифметическое выражение>** / **drop <арифметическое выражение>**; робот может загружать условно бесконечное количество коробок, но может поворачиваться, только если вес коробок не превышает некоторого граничного значения (определятся средой выполнения); робот загружает / выгружает коробку в / на клетку находящуюся перед ним; коробка помещается в слот определяемый арифметическим выражением; оператор возвращает true, если операция выполнена успешно, false – если нет (слот / клетка заняты); undef – в клетке / слоте нет коробки для манипуляции.
  - Операторы осмотра окрестностей:
    - **look** – оператор возвращает расстояние до ближайшего препятствия в направлении движения робота;
    - **test** – возвращает тип препятствия ближайшего препятствия в направлении движения робота.
- Описатель функции
  - **function [<имя функции>] (параметр функции) do <предложения языка> done**. Возврат значений из функции и передача значений в функции происходит через параметр функции (переменную). Функция является изолированной областью видимости. Функции могут быть объявлены внутри другой функции.
  - Точкой входа в программу является функция с именем main.
  - Функция может быть прервана оператором **return**
- Оператор вызова процедуры
  - **<имя функции> (параметр функции)**
  -

  Язык является регистронезависимым. Предложения языка завершаются символом перевода строки.

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.