Вариант 69 (***)

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.

1. Разработать формальный язык для описания действий клеточного робота с поддержкой следующих литералов, операторов и предложений:
   - Логические литералы **TRUE**, **FALSE** (LOGIC);
   - Знаковых целочисленных литералов в десятичном формате (NUMERIC);
   - Строковых литералов, строка ASCII символов заключенных в двойные кавычки “”; в строке могут применяться любые escape последовательности (STRING);
   - Литерал UNDEF, который описывает неинициализированное значение переменной
   - Объявление переменных в форматах:
     - Переменная **<тип> <имя переменной>** [**[размер массива]**]; переменные инициализируются значением по умолчанию UNDEF. Если поле размер массива отсутствует, то переменная трактуется как одиночная переменная, иначе как массив переменных определенного типа; определены все базовые преобразования типа между LOGIC, NUMERIC, STRING; если содержимое STRING не может быть преобразовано естественным образом, то возвращается значение 0 или FALSE, соответственно); для пользовательских типов могут быть определены соответствующие преобразования
   - Объявление типа запись
     - **RECORD <имя типа> DATA** [**<тип поля 1> <имя поля 1>, …**] [**CONVERSION TO <тип 1> <имя процедуры преобразователя 1,…>**] [**CONVERSION FROM <тип 1> <имя процедуры преобразователя 1,…>**]
     - Преобразования типов опциональны; первый параметр процедуры преобразователя данного типа, второй – преобразуемого / целевого; при преобразовании создается временный объект;
     - Объявления переменных типа запись такое же, как объявление переменных встроенных типов;

   - Доступ к элементу массиву:
     - <**имя переменной**> **[индекс численный литерал / одиночная переменная]**; индексация элементов с 0; результат – ссылка на элемент с заданным индексом;
     - <**имя переменной**> [**[массив типа]**]; индексация элементов с 0; результат – новый массив из элементов с индексами из массива индексов;
     - для индекса должно быть определено преобразование в NUMERIC;
     - индексы должны быть положительными и в диапазоне элементов массива; иначе возврат неопределенных значений в заданных позициях;

   Если преобразование не определено и типы не совпадают, то это результат неопределенное значение.

   - Оператор обращения к элементу структуры:
     - <**имя переменной**> **[имя поля]**

   - Оператор присваивания:
     - **<переменная>** [**[индекс1]**] **= <выражение>** присвоение левому операнду значения правого; оператор правоассоциативен (может применяться в цепочках, если <выражение> l-value).

   Все логические и арифметические операторы выполняются на поэлементной основе для массивов. В случае неопределенных операндов – результат не определен (UNDEF), за исключением логических типов (используются правила в базисе 0,1,X). В случае неоднозначности типизации, тип приводится к левому операнду, если оператор используется в

форме ‘.<оператор>’, к правому для – ‘<оператор>.’

- Бинарных операторов:
  - **<выражение>** [.]+[.] **<выражение>**
    - сложение для NUMERIC, дизъюнкция для LOGIC
  - **<выражение>** [.]-[.] **<выражение>**
    - вычитание для NUMERIC, исключающее или для LOGIC
  - **<выражение>** [.]*[.] **<выражение>**
    - умножение для NUMERIC, конъюнкция для LOGIC
  - **<выражение>** [.]/[.] **<выражение>**
    - деление для NUMERIC, штрих Шеффера для LOGIC
  - **<выражение>** [.]^[.] **<выражение>**
    - Возведение в степень для NUMERIC, стрелка Пирса для LOGIC
- Унарный оператор смены знака (NUMERIC) / логическое отрицание (LOGIC):
  - **- < выражение >**
- Операторы сравнения (результат логическое значение)
  - **<выражение>** [.]<[.] **<выражение>**
  - **<выражение>** [.]>[.] **<выражение>**
  - **<выражение>** [.]**?**[.] **<выражение>**
    - равенство
  - **<выражение>** [.]**!**[.] **<выражение>**
    - неравенство
- Операторные скобки (группировка предложений языка)
  - **BLOCK <предложение языка 1>, <предложение языка 2>…UNBLOCK**
- Операторов цикла / условный оператор
  - {<**логическое выражение**>} <**группа предложений языка**>
  - тело цикла выполняется до тех пор, пока логическое выражение истинно.
- Описатель процедуры
  - **PROC** <**имя функции**> **[<тип параметра 1> <параметр 1>** [**&**][**, <тип параметра 1> <параметр 2> [&],...]] <предложение / группа предложений языка>**.
  - Процедура является отдельной областью видимости, параметры передаются в процедуру по значению, если используется модификатор ‘&’, то параметры передаются по ссылке.
  - Процедура может быть объявлена только в глобальной области видимости.
  - Процедура может использоваться только после ее объявления.
- Оператор вызова процедуры
  - **<имя процедуры> [<выражение 1>[,выражение 2]].**
- Управление роботом осуществляется посредством вызова специальных «системных» функций
  - MOVEUP [NUMERIC steps &]; в процедуру передается количество шагов в верх по лабиринту, процедура возвращает количество шагов, которые робот не смог выполнить из-за наличия препятствий.
  - Аналогично MOVEDOWN, MOVERIGTH, MOVELEFT.
  - PINGUP [NUMERIC result &]; в процедуру передается значение 0 – поиск выхода в верх по лабиринту, 1 – поиск стены, UNDEF – любое препятствие. Процедура возвращает количество клеток от позиции робота до препятствия заданного типа; если препятствия заданного типа нет в области видимости, то возвращается UNDEF.
  - Аналогично PINGDOWN, PINGRIGTH, PINGLEFT.
  - Из лабиринта могут быть несколько выходов, каждый из которых защищен паролем – если робот не может назвать правильный пароль для данного выхода, то он не может пройти через данный выход. Пароли могут быть написаны на стенах лабиринта.
    - Для чтения паролей со стены используется система машинного зрения при помощи процедуры VISION [STRING passwords &], которая возвращает массив строк с паролями, которые робот смог прочитать с соседних стен (во всех направлениях)

– Для ввода пароля на выходе используется речевой процессор при помощи процедуры VOICE (STRING password); робот автоматически выходит из лабиринта, если выход открывается.

Предложение языка завершается символом перевода строки. 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, координаты выхода из лабиринта и начальное положение робота задается в текстовом файле.