Вариант 78 (***)

Разработать систему для управления клеточным роботом, осуществляющим передвижение по клеточному лабиринту. Клетка лабиринта имеет форму шестиугольника.
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**;
   - Знаковых целочисленных литералов в восьмеричном (начинаются с 0), десятичном (начинаются с цифры отличной от 0, если не содержат цифры 8 и 9) или шестнадцатеричном форматах (начинаются с '0x', если не содержат A, B, C, D, E, F или начинаются не с цифры);
   - Объявление переменных в форматах:
     - Переменная **digit|logic <имя переменной 1>,** [**имя переменной 2,…**] [**[размерность 1, размерность 2,…]**] **= <выражение со значением элемента по умолчанию>**; размерность по умолчанию 1,0,..; индексы элементов переменных начинаются с 0; тип элементов определяется типом литерала со значением по умолчанию; размерность задается арифметическим или логическим выражением, т.е. вычисляется во время выполнения; размерность (кроме всех последних), не может быть равна 0, данная ситуация является ошибкой времени выполнения, в результате робот обижается и самоуничтожается.
   - Доступ к элементу массива <**имя переменной**> [**[индекс1, индекс2,…]**]; индексация элементов с 1; индекс по умолчанию 1,… (первый элемент в любой размерности); если размерность переменной не соответствует индексу, то это ошибка времени выполнения, в результате робот обижается и самоуничтожается;
   - Оператор определения размера элемента применяется к типу и переменной:
     - **size (имя переменной)**; возвращает размерность переменной в виде временной переменной с размерностью [N].

   Применяется строгая типизация, если типы и размерности операндов не совпадают, то это ошибка времени выполнения, в результате робот обижается и самоуничтожается.
   Определены явные операторы преобразования типов **to logic <имя переменной>** и **to digit <имя переменной>**, преобразующие любую переменную соответственно к логическому или целочисленному типу (true = 1, false = 0, (X<>0) = true, (X=0) = false);
   Определены операторы изменения размерности **resize <имя переменной> [размерность]**, в результате возвращается временная переменная с изменой размерностью; при расширении ранее неизвестные значения элементов задаются равными 0 или TRUE в соответствии с типом переменной; при уменьшении, не попавшие в размерность элементы отбрасываются;

   - Оператор присваивания:
     - **<переменная> = <арифметическое выражение | логическое выражение>** присвоение левому операнду значения правого;

   Все логические и арифметические операторы выполняются на поэлементной основе.

   - Арифметических бинарных операторов сложения, вычитания, умножения, целочисленного деления (**+, -, *,** /); операторы возвращают временную переменную с результатом вычислений:
     - **< арифметическое выражение> оператор < арифметическое выражение>**
   - Операторов сравнения с нулем (**eq, lt, gt, lte, gte**), возвращают true при выполнении условия для большинства элементов переменной, false при не выполнении для большинства;
   - Поэлементный оператор сравнения с нулем (**eq [], lt [], gt [], lte [], gte []**), возвращают временную переменную с результатом сравнения (размерность равна исходной):
     - **оператор <арифметическое выражение>**;

- Логические операторы
  - **! <логическое выражение> - не**;
  - **<логическое выражение> && <логическое выражение> - и**;
  - **most <логическое выражение>, возвращает true, если большинство элементов true, иначе false**

  Унарные операторы обладают высшим приоритетом, затем идут бинарные арифметические и логические; могут применяться операторные скобки ‘(‘ и ’)’, для переопределения порядка вычисления операторов в выражениях.
- Объединение предложений в группы с помощью оператора ‘{‘ и ’}’;
- Операторов цикла **for <имя переменной счетчика> stop <имя граничной переменной> step <имя переменной шага> <предложение языка / группа предложений>**, тело цикла выполняется до тех пор, пока не достигнута или не пройдена с заданным шагом граница для каждого из элементов; шаги выполняются последовательно, начиная с первой размерности и первого в ней элемента (пример, пусть переменная счетчик ((1,2,3),(4,5,6)), граница ((0,0,0), (10, 10, 10)), шаг ((-1,-1,-1), (2,2,2) тогда цикл выполнится последовательно 1, 2, 3, 4, 4 и 3 раза);
- Условных операторов **check < логическое выражение> then <предложение языка / группа предложений> [otherwise <предложение языка / группа предложений**;
- Операторов управления роботом
  - перемещения робота на некоторое количество клеток в текущем направлении **move <арифметическое выражение>.** Если оператор невозможно выполнить из-за наличия препятствия, робот посчитает, что его хотят убить, обидится и самоуничтожится.
  - Поворот на определенное количество секторов (60 градусов), по часовой (+) или против часовой стрелки (-)) **rotate <арифметическое выражение>** , если робота заставить долго вертеться на месте, он посчитает, что его хотят сломать, обидится и самоуничтожится.
  - Робот видит во всех направлениях, на расстояние от 1 до 5 клеток, в зависимости от настроения робота и наличия препятствий. Для того, что бы получить информацию, о том, что видит робот, используется оператор **surroundings <имя переменной>**, который возвращает 3х мерную логическую переменную, в которой в пространстве (N,N,1) описаны видимые роботом стены, а в пространстве (N,N,2) видимые роботом выходы из лабиринта. Если робот видит выход, то его настроение повышается. Если робота долго спрашивать об окружении, не двигаясь в другие клетки, то робот начинает считать, что ему не доверяют, обижается и самоуничтожается.
- Описатель функции
  - **<тип> routing <имя функции> <имя параметра 1>, [<имя параметра 2>, …] группа предложений языка <выражение>**. Функция является отдельной областью видимости, параметры передаются в функцию по значению; из функции параметр возвращается по значению, Результат прошлого вызова функции всегда доступен. Функция может быть объявлена только в глобальной области видимости. Точкой входа в программу является функция **pathfinder.** В теле функции обязательно должен быть оператор возврата значения **return <имя переменной>.**
- Оператор вызова функции
  - **perform <имя функции> <имя параметра 1>, [<имя параметра 2>, …]** вызов функции может быть в любом месте программы.

Предложение языка завершается символом ‘;’. Предложение языка может начинаться со слова **please**, и заканчиваться словом **thank you**, если с роботом быть недостаточно вежливым, то он обижается и самоуничтожается, если быть чрезмерно вежливым, то у него начинается паранойя и он предпочитает самоуничтожиться, что бы с ним не сделали ничего плохого. 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. Если робот слишком долго не может найти выход из лабиринта, то он начинает паниковать и самоуничтожается. Параметры истеричности робота являются случайными величинами, и задаются для каждого экземпляра робота при помещении его в лабиринт средой выполнения.