Вариант 70 (***)

Разработать систему для управления клеточным роботом, осуществляющим передвижение по клеточному лабиринту. Клетка лабиринта имеет форму квадрата, клетки лабиринта могут быть разделены стенами (толщиной стен можно пренебречь), из различных материалов (пластик, дерево, бетон, и т.п.)
Робот может передвинуться в соседнюю клетку в случае отсутствия между ними препятствия, у робота есть ограниченные возможности разрушать препятствия (см. соответствующий раздел).

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

- Знаковых целочисленных литералов в десятичном формате;
- Логические литералы **false**, **true, undefined**;
- Строковые литералы задаются в двойных (“строка”) или одинарных (‘строка’) кавычках
- Объявление переменных и констант в форматах:
  - Логическая переменная **boolean <имя переменной1>[:= логическое выражение][, <имя переменной2>[:= логическое выражение], …];** значения true и false**;**
  - Целочисленная переменная **integer <имя переменной1>[:= арифметическое выражение][, <имя переменной2>[:= арифметическое выражение], …];**
  - Строковая переменная **string <имя переменной1>[:= строковое выражение][, <имя переменной2>[:= строковое выражение], …];**
  - Объявление одномерных массивов **vector of** <**тип элемента**> **<имя переменной1**>**, [<имя переменной2> …]**; массив динамический, при необходимости может увеличиваться (или уменьшаться) в размерах;
- Доступ к элементу массива <**имя переменной**> [**индекс**]; индексация элементов с 0; возвращает ссылку на объект.
- Занесения и извлечение элементов из начала / конца массива
  - **<имя массива> push | pop front | back <выражение>**
- Операторы преобразования типа
  - **<выражение 1> to <имя типа | выражение 2>**
  - выражение 1 преобразуется к заданному типу или типу выражения 2;
  - преобразования в строку определены для всех типов (включая вектор); при преобразовании в вектор атомарного типа получается вектор с элементами заданного типа (если указан) или данного типа, если тип элементов вектора не задан.

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

- Оператор присваивания:
  - **<переменная> := <выражение >** присвоение левому операнду значения правого; оператор право ассоциативен;

- Арифметических / логических / строковых / поэлементных бинарных операторов сложения (для строк конкатенации; для логических “или”), вычитания (для строк – разность; для логических типов “или-не”); операторы возвращают временный объект с результатом вычислений:
  - **<выражение> +|- <выражение>**
  - размерность и типы массивов должны совпадать
- Операторов сравнения (**<. >, =, <>**), возвращают true при выполнении условия, false при не выполнении; сравнения для массивов выполняются, начиная с первого элемента:
  - **< выражение> оператор < выражение>**;
- Объединение предложений в группы с помощью оператора ‘**begin**‘ и ’**end**’;
- Операторов цикла **do <предложение языка / группа предложений> until <логическое выражение>**, тело цикла выполняется до тех пор, пока выражение в условии является истинным.
- Условных операторов **if <логическое выражение> then <предложение языка / группа**

предложений 1>[else <предложение языка / группа предложений 2>], выполняется первое тело оператора, если логическое выражение в условии является истинным, в противном случае выполняется второе (является опциональным);

- Операторов управления роботом
  - перемещения робота на одну клетку вправо (**right**), влево (**left**), вперед (**forward**), назад (**back**); робот может перемещаться не поворачиваясь; если оператор невозможно выполнить из-за наличия препятствия, оператор вернет true, иначе false.
  - поворот робота направо (**rotate_right**), налево (**rotate_left**); всегда возвращает true;
  - измерения расстояния до ближайшего препятствия с помощью точеного лазерного дальномера (**lms**);
  - определение типа препятствия при помощи точеного рефлектометра (**reflect**); возвращается стока описания материала (Пр.: “WOOD”, “STEEL”, “CONCRETE”, “PLASTIC”); выход из лабиринта обозначается строкой “EXIT”; материалы обладают определенной прочностью (задается средой выполнения); стены могут быть из комбинации материалов (Пр: “STELL-GLASS-PLASIC-GLASS”) с прочностью пропорциональной прочности компонент.
  - уничтожить препятствие по направлению движения при помощи бура (**drill**); прочность бура конечна (задается средой выполнения) и уменьшается на прочность разрушенного препятствия; возвращает уровень оставшейся прочности; если прочности не хватает, то препятствие остается, а прочность бура уменьшается до нуля.

- Описатель функции
  - **function of <тип возвращаемого значения> <имя функции> ([<имя параметра1> = <значение по умолчанию 1>],.., […]) <предложение языка / группа предложений языка>**
  - Функция является отдельной областью видимости, параметры передаются в функцию по значению; из функции параметр возвращается по значению.
  - Возвращаемое значение является либо последним предложением языка, либо значением после оператора **return <выражение>.**
  - Точкой входа в программу является функция с именем **application**; возврат из программы осуществляется по достижении конца функции;
  - вызов функции application в программе запрещен;
  - Функция не может быть определена внутри другой функции; но может быть определена до использования;
  - Если в списке параметров присутствует символ …, то функция принимает произвольное число параметров;
  - Функция может принимать в качестве параметра имя другой функции
  - Если параметр не передан явно и не задано значение по умолчанию, то функция берет параметр из текущей области видимости из переменной с данным именем (должно выводится предупреждение на этапе анализа кода); в противном случае семантическая ошибка;
- Оператор вызова процедуры
  - **<имя функции> ([<имя параметра 1>] = <выражение 1>,…),** вызов функции может быть в любом месте программы.
  - Функция может принимать операнды в порядке перечисления либо по заданному имени (Пример: function of boolean test(first, second) return first < second: приведенные вызовы тождественны test (5, 7), test(second=7, first = 5)).
  - Часть параметров (в начале вызова) могут передаваться по позиции, часть (в конце списка параметров) по имени.

Пример корректного кода:

```
function of boolean test(first, second)
return first < second

function of integer num_test(tester,…)
```

```
if tester() return 42 else return 100500

function of integer application (int_param_count=0, vector_of_strings)
num_test(test, first = 5, second=25)
```

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, выход из лабиринта замурован в стене лабиринта, робот может обнаружить выход столкнувшись со стеной либо при помощи дальномера.