Вариант 73(\*\*\*)

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. Роботу известны координаты выходов, но не известен маршрут и конфигурация лабиринта.

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

-        Знаковых целочисленных литералов в десятичном формате, литералы не ограничены в размерах, выделение памяти под литералы осуществляется квантами по 4 байта;

-        Объявление переменных и констант в форматах:

-        Целочисленная переменная со знаком[**const**]**value <имя переменной>**[**= < арифметическое выражение>**]; const – указание на то, что идентификатор является константой, поле инициализации при этом обязательно;

-        Указатель на область памяти **[const] pointer [const] [<выражение типа>]  [= <выражение – адрес>]**; модификаторы const являются опциональными, первый говорит о том, что указатель не изменяется – в этом случае выражение адрес является обязательным, второе говорит о том, что объект по указателю доступен только для чтения (изначально объект может не быть константным); <имя типа> является опциональным, в этом случае предполагается, что указатель указывает на value; выражение-адрес является арифметическим выражением, но на этапе выполнения проверяется, что результат является адресом некоторого объекта с заданным типом – иначе ошибка времени выполнения.

-        Массив элементов [**const**]**array of <выражение типа>** [**= <арифметическое выражение>**]; модификатор const является опциональным, его присутствие указывает на то, что массив является статическим, в этом случае поле арифметического выражения, определяющее размер, является обязательным; в случае динамических массивов, присутствие поля c размером – указывает нижнюю границу массива и квант увеличения/уменьшения;  верхняя граница является плавающей в зависимости от наполненности массива  (по правилу увеличения квантами указанного размера при необходимости); по умолчанию размер массива и квант = 4 элементам; элементом массива может быть любой тип, в том числе и массив.

-        Доступ к элементу массива <**имя массива**>**[арифметическое выражение индекс]**; индексация начинается с 0;

-        Определен доступ на модификацию к элементу следующим за последним заполненным.

-        Обращение к несуществующему элементу массива – ошибка времени выполнения;

-        Доступ к значению по указателю **\* <имя указателя>**

-        Получение адреса переменной / константы & <имя идентификатора>

-        Оператор получения размера идентификатора в элементах ( для value и pointer всегда 1, для массива – количество заполненных элементов)

Применяется строгая типизация, преобразования между различными типами не определены;

-        Операторов присваивания ‘=’;

-        Арифметических бинарных операторов сложения, вычитания, умножения, целочисленного деления, получения остатка от деления (+, -, \*, /, %);

-        **< арифметическое выражение> оператор < арифметическое выражение>**

-        Указательных бинарных операторов сложения **+**; результат новый указатель в рамках данного элемента (выход за границы – ошибка времени выполнения)

-        **< выражение указатель> оператор < арифметическое выражение>**

-        Указательных бинарных операторов разности **-**; результат новый указатель в рамках данного элемента

-        **< выражение указатель> оператор < арифметическое выражение>**

-        Операторов сравнения для арифметических выражений и указателей на элементы массива (на один массив – иначе ошибка времени выполнения) (!=, <=, >=), возвращают 1 при выполнении условия и 0 при не выполнении:

-        **<арифметическое выражение> оператор <арифметическое выражение>**

(приоритет операторов в порядке убывания (\*,/,%),(+,-),(<=,>=,!=)могут применяться операторные скобки ‘(‘ и ’)’, для переопределения порядка вычисления операторов в выражениях).

-        Объединение предложений в группы с помощью скобок **{ }**;

-        Операторов цикла **while (<арифметическое выражение>)  <предложение языка / группа предложений>[finish <предложение языка / группа предложений>]**

-        выполняется тело цикла до тех пор, пока выражение в условии отлично от 0.

-        выполнение цикла может быть прервано оператором **break**, в этом случае управление блоку finish не передается

-        при штатном завершении цикла управление передается опциональному блоку finish, которое выполняется 1 раз

-        Условных операторов **[not]zero? (арифметическое выражение) <предложение языка / группа предложений>,**выполняется тело оператора, если арифметическое выражение в условии равно(zero) 0 или неравно (notzero) 0;

-        Оператор итерации по элементам массива **foreach <имя идентификатора> <имя функции> ([<дополнительный параметр 1>,…])**

-        функция выполняется для каждого элемента массива (если идентификатор не массив, то выполняется для данного элемента); функция должна принимать в качестве первого значение элемента итерируемого массива; если функция принимает еще параметры, то они передаются через опциональные дополнительные параметры.

-        Операторов управления роботом

-        перемещения робота на одну клетку в заданном направлении **top, bottom, left, right**. Если оператор невозможно выполнить из-за наличия препятствия, то оператор возвращает 0, иначе 1.

-        Робот может оставить на текущей клетке портал при помощи оператора **portal;** робот может телепортироваться в оставленный ранее портал при помощи оператора teleport, при этом портал закрывается; ранее оставленные порталы используются по принципу стека.

-        При попадании в клетку с выходом, выполнение программы завершается.

-        Описатель функции

-        **<тип возвращаемого значения>** <**имя функции> ([<тип параметра> <имя параметра>,..]) группа предложений языка**. Функция является отдельной областью видимости, параметры передаются в функцию по значению. Возвращаемым значением является значение после оператора **return**. Возможно объявление функций внутри других функций, в этом случае им доступны переменные «родительской» функции. Точкой входа в программу является функция с именем **main**.

-        Оператор вызова функции

-        **имя функции** **(**имена переменных, разделенных запятой**),**вызов функции может быть в любом месте программы.

Предложение языка завершается символом ‘**;**’. Язык является регистронезависимым.

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.