

Вариант 64 (***)

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**;
- 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 **VAR** <имя переменной> [[размерность 1, размерность 2,...]] = <литерал со значением элемента по умолчанию>; размерность по умолчанию 1,0,...; индексы элементов переменных начинаются с 1; 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 **LOGITIZE** <имя переменной> и **DIGITIZE** <имя переменной>, преобразующие любую переменную соответственно к логическому или целочисленному типу (**TRUE** = 1, **FALSE** = 0, $(X < 0) = \text{TRUE}$, $(X = 0) = \text{FALSE}$);

Определены операторы изменения размерности **REDUCE** <имя переменной> [размерность] и **EXTEND** <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, непопавший 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 (**MXEQ**, **MXLT**, **MXGT**, **MXLTE**, **MXGTE**), возвращают **TRUE** при выполнении условия для большинства элементов переменной, **FALSE** при не выполнении для большинства;
- Поэлементный оператор сравнения с нулем (**ELEQ**, **ELLT**, **ELGT**, **ELLTE**, **ELGTE**), возвращают временную переменную с результатом сравнения (размерность равна

исходной):

- оператор <арифметическое выражение>;
- Логические операторы
 - NOT <логическое выражение>;
 - <логическое выражение> AND <логическое выражение>;
 - MXTRUE <логическое выражение>, возвращает TRUE, если большинство элементов TRUE, иначе FALSE
 - MXFALSE <логическое выражение>, возвращает TRUE, если большинство элементов FALSE, 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> **BOUNDARY** <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 **SWITCH** <логическое выражение> <TRUE|FALSE> <предложение языка / группа предложений> [<FALSE|TRUE> <предложение языка / группа предложений>], выполняется первое тело оператора, если логическое выражение в условии совпадает с выбором, иначе выполняется второе;
- Операторов управления роботом
 - перемещения робота на одну клетку в текущем направлении **MOVE**. Если оператор невозможно выполнить из-за наличия препятствия, робот посчитает, что его хотят убить, обидится и самоуничтожится.
 - Поворот налево **ROTATE LEFT**, поворот направо **ROTATE RIGHT**, 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 настройки робота и наличия препятствий. Для того, что бы получить информацию, о том, что видит робот, используется оператор **GET ENVIRONMENT**, 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
 - **TASK** <имя функции> <имя параметра 1>, [<имя параметра 2>, ...] группа предложений языка <выражение>. Функция является отдельной областью видимости, параметры передаются в функцию по ссылке; из функции параметр возвращается по ссылке, Результат прошлого вызова функции всегда доступен. Функция может быть объявлена только в глобальной области видимости. Точкой входа в программу является функция **FINDEXIT**. В теле функции обязательно должен быть оператор возврата значения **RESULT** <имя переменной>.
- Оператор вызова функции
 - **DO** <имя функции> <имя параметра 1>, [<имя параметра 2>, ...] вызов функции может быть в любом месте программы.
- Получение результата выполнения функции
 - **GET** <имя функции>

Предложение языка завершается символом перевода строки. Предложение языка может начинаться со слова **PLEASE**, и заканчиваться словом **THANKS**, 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.