

Вариант 71 (***)

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:

- Знаковых целочисленных литералов в двоичном (начинается с нуля, содержит только 0 и 1 в записи числа) и десятичном формате;
 - Логических литералов **true** и **false**; логические константы и выражения преобразуются к знаковым целочисленным как 1 и 0 соответственно, 0 преобразуется в false, любое другое число в true;
 - Объявление переменных/констант в форматах:
 - Целочисленная переменная-вектор со знаком **[c][v]int** <имя идентификатора> = <арифметическое выражение> | {арифметическое выражение 1, арифметическое выражение 2, ...};
 - Логическая переменная-вектор **[c][v]bool** <имя идентификатора> = <логическое выражение> | {логическое выражение 1, логическое выражение 2, ...};
 - Целочисленная переменная-матрица со знаком **[c][m]int** <имя идентификатора> = <арифметическое выражение> | {арифметическое выражение 1, арифметическое выражение 2, ...} | {{арифметическое выражение 1, арифметическое выражение 2, ...}, ...};
 - Логическая переменная-матрица со знаком **[c][m]bool** <имя идентификатора> = <логическое выражение> | {логическое выражение 1, арифметическое выражение 2, ...} | {{арифметическое выражение 1, арифметическое выражение 2, ...}, ...};
 - Матрицы должны быть прямоугольными;
 - Если не указаны модификаторы “v” и “m”, то это переменная-скаляр;
 - Если присутствует префикс “с”, то объявляемый идентификатор является константой;
 - Определены преобразования переменных с меньшей размерностью в большую; обратное преобразование не определено.
 - Обращение к идентификаторам по индексу (получение переменной такой же или меньшей размерности) (на примере матриц)
 - <имя идентификатора> (выражение арифметический скаляр 1, выражение арифметический скаляр 2) – получение скаляра с заданными координатами;
 - <имя идентификатора> (выражение арифметический скаляр 1, [:]) – получение вектора столбца матрицы с заданным номером; символ ‘:’ является опциональным;
 - <имя идентификатора> ([:], выражение арифметический скаляр 1) – получение вектора строки матрицы с заданным номером; символ ‘:’ является опциональным;
 - <имя идентификатора> (выражение арифметический вектор 1, [:]) – получение матрицы, состоящей из столбцов с заданными номерами; символ ‘:’ является опциональным;
 - <имя идентификатора> ([:], выражение арифметический вектор 1) – получение матрицы, состоящей из строк с заданными номерами; символ ‘:’ является опциональным;
 - <имя идентификатора> (выражение логическая матрица 1) – получение матрицы, состоящей из ячеек с элементами соответствующими истинным значениям в логической матрице; индексы должны формировать прямоугольную матрицу;
- Пример корректных логических матриц:
- | | |
|-----------|----------|
| (1,1,0,1 | (1,0,0,1 |
| 1,1,0,1 | 1,1,0,0 |
| 1,1,0,1 | 0,1,1,0 |
| 0,0,0,0) | 0,0,1,1) |
- <имя идентификатора> (выражение логический вектор 1, [:]) – получение матрицы, состоящей из столбцов с номерами, соответствующими истинным значениям

- в логическом векторе; символ ‘.’ является опциональным;
- **<имя идентификатора> ([:], выражение логический вектор 1)** – получение матрицы, состоящей из строк с номерами, соответствующими истинным значениям в логическом векторе; символ ‘.’ является опциональным;
- при использовании логических векторов/матриц для индексации размерности исходной и логической матрицы/вектора должны совпадать;
- аналогично для векторов
- применение индексации возвращает ссылки на ячейки в оригинальной матрице/векторе

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

- Операторов присваивания ‘<-’;
- Арифметических операторов:
 - **<выражение> + <выражение>**
 - **<выражение> - <выражение>**
 - **<выражение> * <выражение>** (матричное умножение)
 - **<выражение> .* <выражение>** (поэлементное умножение)
 - **<выражение>’** (транспонирование матрицы)
 - **sum (<выражение>)** (поэлементное суммирование)
 - **<выражение> <<** (циклический сдвиг влево)
 - **<выражение> >>** (циклический сдвиг вправо)
- Логических операторов (результат логическое выражение):
 - **! <логическое выражение>** (отрицание);
 - **<логическое выражение> and|&& <логическое выражение>** (конъюнкция);
- Операторов сравнения:
 - **<арифметическое выражение> < <арифметическое выражение>;**
 - **<арифметическое выражение> > <арифметическое выражение>;**
- Операторов цикла
 - **for <идентификатор скаляр> = <арифметическое выражение скаляр 1>: <арифметическое выражение скаляр 2> begin[for] <предложения языка> end[for]** (выполнение тела цикла с изменением идентификатора скаляра от значения выражения 1 до значения выражения 2; выражения вычисляются при инициализации цикла)
- Условных операторов **if <логическое выражение> begin[if]<предложение языка / группа предложений>end[if];**
- Операторов управления роботом
 - перемещения робота на одну клетку в заданном направлении относительно текущего **move(арифметическое выражение)**. Если оператор невозможно выполнить из-за наличия препятствия, он возвращает логическое значение **false**, робот остается стоять на месте, иначе – **true**.
 - Поворот робота **right, left**. После выполнения оператора робот разворачивается в указанном направлении относительно текущего.
 - Измерение расстояния до первого препятствия **wall**, возвращает количество клеток до ближайшего препятствия в текущем направлении.
 - Измерение расстояния до первого препятствия **exit**, возвращает количество **true**, если робот видит выход в текущем направлении, иначе **false**.
- Описатель функции
 - **[<тип возвращаемого значения 1> <имя возвращаемого значения>[,...]] = function <имя функции> ([<тип переменной 1> <имя переменной 1> [= <значение по умолчанию переменной 1>],...]) begin предложения языка end.** Функция является отдельной областью видимости, параметры передаются в функцию по значению; возвращаемые значения передаются из функции по значению. Функция может быть объявлена в любом месте программы, при объявлении она не выполняется.

- Оператор вызова функции
 - **<имя функции> [<выражение 1>,...]**, вызов процедуры может быть в любом месте программы после ее объявления.

Предложение языка завершается символом перевода строки; предложение может быть продолжено на другой строке, если в конце стоит символ ‘...’ 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 и начальное положение робота задается в текстовом файле.