

# Интерпретатор для клеточного робота

## Вариант 63 (\*\*\*)

Разработать систему для управления клеточным роботом, осуществляющим передвижение по клеточному лабиринту. Клетка лабиринта имеет форму правильного треугольника.

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 десятичном формате; по умолчанию литералы соответствуют типу `int`, для задания литерала с типом `short` используется префикс `'S'`;
- Логические литералы **false**, **true**, **undefined**;
- Объявление переменных и констант в форматах:
  - Логическая переменная **bool** `<имя переменной1>[set логическое выражение][, <имя переменной2>[set логическое выражение], ...]`; значения `true` и `false`;
  - Целочисленная переменная **short** `[int] <имя переменной1>[set арифметическое выражение][, <имя переменной2>[set арифметическое выражение], ...]`;
  - Целочисленная переменная **int** `<имя переменной1>[set арифметическое выражение][, <имя переменной2>[set арифметическое выражение], ...]`;
  - Объявление одномерных массивов **vector of** `<тип элемента> <имя переменной1> [[арифметическое выражение (размер)]] set {значение элемента1, значение элемента 2,...}, [ <имя переменной2> ...]`; обязательно либо поле размера, либо поле задания значений, если в объявлении присутствуют оба поля, то количество переменных в списке инициализации должно совпадать с размерностью массива; возможно объявление массива массивов, в этом случае инициализация применяется только на уровне внешнего массива (пример: `vector of vector of vector of short Matrix2x3x2 set {{{1,2},{3,4},{4,5}}, {{6,7},{8,9},{10,11}}}`);
- Доступ к элементу массива `<имя переменной> [индекс]`; индексация элементов с 0; возвращает ссылку на объект, (пример: `Matrix2x3x2[0][2][1]` вернет ссылку на элемент со значением 5);
- Оператор определения размера элемента применяется к типу и переменной:
  - **sizeof** (`тип | имя переменной`); `vector` не может использоваться в качестве типа.

Для переменных определено следующее соотношение размеров `sizeof(bool) <= sizeof(short) < sizeof(int)`. Вычисления по умолчанию выполняются в разрядной сетке `short`, происходит автоматическое расширение разрядной сетки до `int` при переполнении; переполнение разрядной сетки `int` считается исключительной ситуацией и программа должна завершаться аварийно.

- Оператор присваивания:
    - `<переменная> set <арифметическое выражение | логическое выражение>` присвоение левому операнду значения правого; оператор право ассоциативен;
  - Арифметических бинарных операторов сложения, вычитания (**add**, **sub**); операторы возвращают временный объект со результатом вычислений:
    - `< арифметическое выражение> оператор < арифметическое выражение>`
  - Операторов сравнения (**first|second smaller**, **first|second larger**), возвращают `true` при выполнении условия, `false` при не выполнении, и `undefined` при равенстве; сравнение для массивов не определены (сравнивать элементы массивов, не являющиеся массивами можно):
    - `<арифметическое выражение>оператор <арифметическое выражение>;`
  - Логические операторы (**[not] or**, **[not] and**), возвращают `true`, `false` или `undefined`
    - `<логическое выражение>оператор <логическое выражение>;`
- Все операторы равноприоритетны; могут применяться операторные скобки `'('` и `')'`, для переопределения порядка вычисления операторов в выражениях.
- Определено преобразование: арифметических типов в логические `< 0 = false`, `0 = undefined`, `> 0 = true`; логических типов в арифметические `true = 1`, `false = -1`, `undefined = 0`;

- Объединение предложений в группы с помощью оператора **'begin'** и **'end'**;
- Операторов цикла **do** <предложение языка / группа предложений> **while** <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>, выполняется первое тело оператора, если логическое выражение в условии является истинным, если ложным, то второе тело, в противном случае не выполняется ничего;
- Операторов управления роботом
  - перемещения робота на одну клетку вправо (**[move] right**), влево (**[move] left**), перемещение по вертикали (**move**). Если оператор невозможно выполнить из-за наличия препятствия, оператор вернет undefined, иначе -1 для перемещения влево и вниз, и 1 для движения вправо и вверх.
  - Измерения расстояния до ближайшего препятствия с помощью сканирующего лазерного дальномера (**lms**), возвращает расстояние до ближайшего препятствия; (расстояние до левого со знаком '-', до правого со знаком '+'); дальномер имеет радиус действия, которое определяется средой выполнения, при наличии препятствия только вне радиуса, возвращается 0 (undefined); после выполнения текущего сканирования дальномер переключается на противоположное направление. В случае если дальномер обнаруживает в стене выход из лабиринта, то он возвращает значение со знаком обратным ожидаемому. Дальномер не работает по вертикали, наличия препятствия в соседней клетке проверяется только при помощи активного бампера (попыткой переместиться в клетку);
- Описатель функции
  - **function** <имя функции> (<тип параметра> <имя параметра>,...) группа предложений языка **return** <выражение>. Функция является отдельной областью видимости, параметры передаются в функцию по значению; из функции параметр возвращается по значению, тип возвращаемого значения выводится из типа выражения после слова return. Точкой входа в программу является функция с именем **work**. Возврат из функции осуществляется по достижении конца функции
- Оператор вызова процедуры
  - <имя функции> (имена переменных разделенных пробелом), вызов функции может быть в любом месте программы.

Предложение языка завершается символом **';**'. Язык является регистронезависимым имена переменных могут быть сокращены до минимально распознаваемых лексем в данной области видимости.

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.