

Course > Лабораторная работа №3 > Интерпретатор для клеточного робота > Интерпретатор для клеточного робота

< Previous

Next >

Интерпретатор для клеточного робота

Bookmark this page

Вариант 62 (***)

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

Робот может передвинуться в соседнюю клетку в случае отсутствия препятствия между клетками.

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

- Знаковых и беззнаковых целочисленных литералов в тридцатидвуричном формате, по умолчанию литералы беззнаковые, для объявления знаковых используется префикс со знаком '+' или '-';
- Объявление переменных и констант в форматах:
 - Целочисленная беззнаковая переменная **tiny** <имя переменной1>[<имя переменной2> ...] << **целочисленный беззнаковый литерал**>; значения 0 и 1;
 - Целочисленная переменная **small** <имя переменной1>[<имя переменной2> ...] << **целочисленный литерал**>; знаковость переменной определяется знаковостью литерала, диапазон значений для знаковых от -16 до 15, для беззнаковых от 0 до 31.
 - Целочисленная переменная **normal** <имя переменной1>[<имя переменной2> ...] << **целочисленный литерал**>; знаковость переменной определяется знаковостью литерала, диапазон значений для знаковых от -512 до 511, для беззнаковых от 0 до 1023.
 - Целочисленная переменная **big** <имя переменной1>[<имя переменной2> ...] << **целочисленный литерал**>; знаковость переменной определяется знаковостью литерала, диапазон значений для знаковых от -16384 до 16383, для беззнаковых от 0 до 32767.
 - Объявление двумерных массивов **field** <тип_элемента> <тип размерности> <имя переменной1>[<имя переменной2> ...] << **целочисленный литерал**>; знаковость переменной определяется знаковостью литерала, диапазон значений соответствует ограничениям поля <тип элемента>; диапазон индексов ограничен размером тип размерности.
- Доступ к элементу массива <имя массива> [индекс по горизонтали индекс по вертикали]

Применяется правило продвижения типов к размерности большего операнда tiny->small->normal->big; при присваивании меньшему по размерности операнда большего или отрицательного знакового беззнаковому (выход за разрядную сетку), то значение округляется до ближайшего входящего в заданную разрядную сетку.

- Операторов присваивания '>>' и '<<':
- <арифметическое выражение> >> <переменная> присвоение правому операнду значения левого, левое присвоение менее приоритетно; оператор левоассоциативен;
- <переменная> << <арифметическое выражение> присвоение левому операнду значения правого, правое присвоение более приоритетно; оператор правоассоциативен;

(Пр: допустимо присвоение A<>C -> A=B, C=B и A>>B<<C -> B = A, A>>B>>C -> B = A, C=A, и.т.п.)

- Арифметических бинарных операторов сложения, вычитания, умножения, целочисленного деления (+, -, *, /); деление на 0 приводит к результату максимальному по модулю со знаком делимого; арифметические операторы над массивами не определены:
- < арифметическое выражение> оператор < арифметическое выражение>
- Операторов сравнения (<, <=, >=), возвращают 1 при выполнении условия и 0 при не выполнении, сравнение для массивов не определены:
- <арифметическое выражение>оператор <арифметическое выражение>; (приоритет операторов в порядке убывания (*,/),(+,-)(<=,>=,<,>); могут применяться операторные скобки '(' и ')', для переопределения порядка вычисления операторов в выражениях).
- Определены унарные операторы явного преобразования в знаковый формат '-' и беззнаковый формат '+'
- оператор <имя переменной>
- Объединение предложений в группы с помощью оператора ':'
- Операторов цикла **until** <арифметическое выражение> **do** <предложение языка / группа предложений>, условие цикла преобразуется к tiny, тело цикла выполняется до тех пор, пока выражение в условии отлично от 0 (=1).
- Условных операторов **check** арифметическое выражение **do** <предложение языка / группа предложений>, выполняется тело оператора, если арифметическое выражение в условии отлично от 0;
- Операторов управления роботом
 - перемещения робота на одну клетку в текущем направлении **go**, поворот налево на 60 градусов **rl**, поворот направо на 60 градусов **rr**. Если оператор невозможно выполнить из-за наличия препятствия, оператор вернет 0, иначе 1 (поворот можно выполнить всегда).
 - Осмотр ближайших окрестностей с помощью сонаров **sonar**, возвращает беззнаковый small в битах которого закодировано наличие препятствия в ближайшей клетке, от -120 до 120 градусов относительно текущего положения, (т.е. у робота нет заднего сонара), последующий вызов данного оператора при условии, что робот не двигался и не поворачивался, вернет информацию о наличии препятствий в радиусе двух клеток и т.д. (сигнал сквозь препятствие не проходит)
 - Проверка направления к выходу **compass**; возвращает знаковый big, в котором указано относительное направление к выходу из лабиринта в минутах.
- Описатель функции
 - <тип возвращаемого значения> <имя функции> [<тип параметра> <имя параметра>...] **begin** группа предложений языка **end**). Функция является отдельной областью видимости, параметры передаются в функцию по значению. Точкой входа в программу является функция с именем **main**.
 - Возврат из функции осуществляется по оператору **return** <возвращаемое значение>
- Оператор вызова процедуры
 - <имя функции> (имена переменных разделенных пробелом), вызов функции может быть в любом месте программы.

Предложение языка завершается символом '.' (простое, последнее в составном), или ';' (не последнее простое в составном). Язык является регистрозависимым (все операторы имена идентификаторов строчные, а литералы заглавные), ключевые слова могут быть сокращены до минимально распознаваемых лексем, например small = smal=sma=sm, т.к. нет ключевых слов начинающихся на sm (до s сокращать нельзя, т.к. есть sonar).

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.

< Previous

Next >