# Введение

# Неформальная постановка задачи

Разработать проект транслятора из подмножества языка «Java» в эквивалентное подмножество языка «Go».

Подмножество языка «Java» включает:

1. Типы данных:***byte ,short, int,long,float,double,char,boolean.***
2. Операторы:

***Арифметические операторы(***Таблица 1***).***

|  |  |  |
| --- | --- | --- |
| Оператор | Описание | Пример |
| + | Складывает значения по обе стороны от оператора | A + B даст 30 |
| - | Вычитает правый операнд из левого операнда | A - B даст -10 |
| \* | Умножает значения по обе стороны от оператора | A \* B даст 200 |
| / | Оператор деления делит левый операнд на правый операнд | B / A даст 2 |
| % | Делит левый операнд на правый операнд и возвращает остаток | B % A даст 0 |
| ++ | Инкремент - увеличивает значение операнда на 1 | B++ даст 21 |
| -- | Декремент - уменьшает значение операнда на 1 | B-- даст 19 |

Таблица 1-Арифметические операторы

***Операторы сравнения(Таблица 2).***

|  |  |  |
| --- | --- | --- |
| Оператор | Описание | Пример |
| == | Проверяет, равны или нет значения двух операндов, если да, то условие становится истинным | (A == B) — не верны |
| != | Проверяет, равны или нет значения двух операндов, если значения не равны, то условие становится истинным | (A != B) — значение истинна |
| > | Проверяет, является ли значение левого операнда больше, чем значение правого операнда, если да, то условие становится истинным | (A > B) — не верны |
| < | Проверяет, является ли значение левого операнда меньше, чем значение правого операнда, если да, то условие становится истинным | (A < B) — значение истинна |
| >= | Проверяет, является ли значение левого операнда больше или равно значению правого операнда, если да, то условие становится истинным | (A >= B) — значение не верны |
| <= | Проверяет, если значение левого операнда меньше или равно значению правого операнда, если да, то условие становится истинным | (A <= B) — значение истинна |

Таблица 2-Операторы сравнения

***Побитовые операторы(Таблица 3).***

|  |  |  |
| --- | --- | --- |
| Оператор | Описание | Пример |
| & (побитовое и) | Бинарный оператор AND копирует бит в результат, если он существует в обоих операндах. | (A & B) даст 12, который является 0000 1100 |
| | (побитовое или) | Бинарный оператор OR копирует бит, если он существует в любом из операндов. | (A | B) даст 61 который равен 0011 1101 |
| ^ (побитовое логическое или) | Бинарный оператор XOR копирует бит, если он установлен в одном операнде, но не в обоих. | (A ^ B) даст 49, которая является 0011 0001 |
| ~ (побитовое дополнение) | Бинарный оператор дополнения и имеет эффект «отражения» бит. | (~ A) даст -61, которая является формой дополнением 1100 0011 в двоичной записи |
| << (сдвиг влево) | Бинарный оператор сдвига влево. Значение левых операндов перемещается влево на количество бит, заданных правым операндом. | A << 2 даст 240, который 1111 0000 |
| >> (сдвиг вправо) | Бинарный оператор сдвига вправо. Значение правых операндов перемещается вправо на количество бит, заданных левых операндом. | A >> 2 даст 15, который является 1111 |
| >>> (нулевой сдвиг вправо) | Нулевой оператор сдвига вправо. Значение левых операндов перемещается вправо на количество бит, заданных правым операндом, а сдвинутые значения заполняются нулями. | A >>> 2 даст 15, который является 0000 1111 |

Таблица 3-Побитовые операторы

***Логическое операторы(Таблица 4).***

|  |  |  |
| --- | --- | --- |
| Оператор | Описание | Пример |
| && | Называется логический оператор «И». Если оба операнда являются не равны нулю, то условие становится истинным | (A && B) — значение false |
| || | Называется логический оператор «ИЛИ». Если любой из двух операндов не равен нулю, то условие становится истинным | (A || B) — значение true |
| ! | Называется логический оператор «НЕ». Использование меняет логическое состояние своего операнда. Если условие имеет значение true, то оператор логического «НЕ» будет делать false | !(A && B) — значение true |

Таблица 4-Логические операторы

***Операторы присваивания(Таблица 5).***

|  |  |  |
| --- | --- | --- |
| Оператор | Описание | Пример |
| = | Простой оператор присваивания, присваивает значения из правой стороны операндов к левому операнду | C = A + B, присвоит значение A + B в C |
| += | Оператор присваивания «Добавления», он присваивает левому операнду значения правого | C += A, эквивалентно C = C + A |
| -= | Оператор присваивания «Вычитания», он вычитает из правого операнда левый операнд | C -= A, эквивалентно C = C - A |
| \*= | Оператор присваивания «Умножение», он умножает правый операнд на левый операнд | C \* = A эквивалентно C = C \* A |
| /= | Оператор присваивания «Деление», он делит левый операнд на правый операнд | C /= A эквивалентно C = C / A |
| %= | Оператор присваивания «Модуль», он принимает модуль, с помощью двух операндов и присваивает его результат левому операнду | C %= A, эквивалентно C = C % A |

Таблица 5-Операторы присваивания

1. Операторы циклов: цикл for, цикл do, цикл while.
2. Операторы ветвления: if else, тернарный условный оператор (условное выражение ? выражени1: выражение2).
3. Стандартные функции:

double min( a, b ) – возвращает минимум из a и b

double log( a ) – возвращает натуральный логарифм аргумента a

double pow( a, b ) – возвращает значение a, возведенное в степень b

double sqrt( a ) – возвращает квадратный корень из a

double time() – возвращает текущее значение модельного времени (в единицах модельного времени)

Date date() – возвращает текущую модельную дату (Date является стандартным Java классом)

int getMinute() – возвращает текущую минуту в часе текущей модельной даты

double minute()– возвращает интервал времени, соответствующий одной минуте, в выбранных в данной модели единицах модельного времени.

# Синтаксис входного языка

# Контекстные условия

# Таблица соответствия

# Проект лексического анализатор

# Таблица ключевых слов

# Типы лексем

# Ошибки лексического анализатора

# Конечный автомат лексического анализатора

# Проект синтаксического анализатора

# Заключение

# Список литературы