Definición del lenguaje: Tokke

Tipos de variables

- int -> int i = 4
- char -> char car = ‘s’
- float -> float f = 23.124f
- double -> double d = 24.43
- boolean -> boolean b = false
- byte -> byte b = 12b

Literales

- null: valor null
- 1 : número entero
- 1.232 : número double
- 1.324f : número float
- false/true : booleanos
- 255b : byte

Estructuras de datos

(Arrays no implementados en la generación de código)

- Arrays estáticos:
  - Arrays de dimensión fija y de elementos de un único tipo.
  - El valor de la variable que identifica al array será la referencia a la posición de memoria donde está alojado (se almacena de forma continua, terminando en un valor nulo).
  - Declaración:
    - *type*[] arr = [1,2,3,4,5,6] // array de valores prefijados
    - *type[]*
    - *type* arr = *type*[n] // se crea un array de n valores de tipo *type*
  - Acceso:
    - *arr[v]*
  - Asignación:
    - *arr[v] = x*
  - Liberación de memoria:
    - free arr
  - Multidimensionales:
    - *type*[][] v = [[2,1], [3,2]]
    - *type*[][] v = *type*[m][n] // se crea una matriz de m x n de valores de tipo *type*

Operaciones

Asignación

int e = 10

Se permiten casteos implícitos entre tipos de datos numéricos, manteniéndose siempre el tipo de la izquierda. (No implementado en la generación de código).

int i = 10

double e = i resulta en 10.0

Matemáticas

El orden de cálculo es de izquierda a derecha, teniendo en cuenta la precedencia de los operandos en matemáticas: paréntesis, potencias, multiplicación, división, suma y resta.

Aritméticas:

- Suma, resta, multiplicación, división : + - * /
- Potencia: ^
- Módulo: %
- Suma unitaria: i++ / ++i (tomar el valor después o antes de la suma, respectivamente)
- Resta unitaria: i— / --i (tomar el valor después o antes de la resta, respectivamente)
- i+=1
- i-=1
- i*=2
- i/=2
- Comparaciones: a>b, a<b, a>=b, a<=b, a != b, a==b
    - Las comparaciones que se realizan entre valores numéricos, convirtiendo ambos valores al tipo de dato más grande.
    - Las comparaciones entre string se puede realizar solamente con ==

Bit a bit

- AND: &&
- OR: ||
- XOR: ^^
- Negación: !
- Desplazamiento:
    - << 4 (4 bits a la izquierda)
    - >> 4 (4 bits a la derecha)

Booleanas

- AND: &&
- OR: ||
- Negación: !

Cadenas de caracteres

- Concatenación: + (No implementado en la generación de código).
- Substring: (No implementado en la generación de código).
    - char[] str = new char[n]
    - str[0] = ‘c’
    - str[1] = ‘a’
    - str[2] = ‘n’
    - str[3] = ‘o’
    - str[4] = ‘n’
    - str = splice(str, 0, 2) // “can”

Funciones propias

- print(str) / println(str) : imprime por consola
- size(arr) : devuelve número de elementos (No implementado en la generación de código).
- splice(arr, ini, fin) : devuelve subarray de *arr* entre ini y fin, ambos inclusive. (No implementado en la generación de código).

Estructuras de control

For

for(int i = 0; i < 10; i++) {

}

for i in array {

}

While

while(exp){

}

If

if(exp) {

} else if (exp) { (Else if no implementado en la generación de código).

} else {

}

Funciones

def *type* nombre ([*type* nombre1, (*type* nombre2, …)]) {

}

Las funciones solo aceptan paso de parámetros por valor.

Estructura del programa

- Declaraciones globales
- Declaración de funciones
- Función *main*

```
int a

int b = 3

def int suma(int a, float b) {

        return a + b

}

def int main() {

        a = 2

        print(suma(a + b))

        return 0

}
```