Training statements for Flex

1. Statements

1.1. Declaration

In below table, following data types are supported: **Integer, int, real, float, double, complex, string, str, char, character, boolean, bool.**

Statement	Intent	Entity
real x	declare_var	• name(x) • type(real)
x real	declare_var	• name(x) • type(real)
x is real	declare_var	• name(x) • type(real)
int x	declare_var	• name(x) • type(int)
x integer	declare_var	• name(x) • type(int)
x is int	declare_var	• name(x) • type(int)
float x	declare_var	• name(x) • type(float)
x float	declare_var	• name(x) • type(float)
x is float	declare_var	• name(x) • type(float)
double x	declare_var	• name(x) • type(double)
x double	declare_var	• name(x) • type(double)
x is double	declare_var	• name(x) • type(double)

complex x	declare_var	• name(x) • type(complex)
x complex	declare_var	• name(x) • type(complex)
x is complex	declare_var	• name(x) • type(complex)
char x	declare_var	• name(x) • type(char)
x character	declare_var	• name(x) • type(char)
x is char	declare_var	• name(x) • type(char)
str x	declare_var	• name(x) • type(string)
x str	declare_var	• name(x) • type(string)
x is string	declare_var	• name(x) • type(string)
bool x	declare_var	• name(x) • type(boolean)
x bool	declare_var	• name(x) • type(boolean)
x is boolean	declare_var	• name(x) • type(boolean)
Multi variable declaration		
x, y, z real	declare_multi_var	• name(x) • name(y) • name(z) • type(real)
real x, y, z	declare_multi_var	• name(x) • name(y) • name(z) • type(real)
x, y, z are real	declare_multi_var	• name(x) • name(y) • name(z)

		• type(real)
x, y, z integer	declare_multi_var	• name(x) • name(y) • name(z) • type(integer)
int x, y, z	declare_multi_var	• name(x) • name(y) • name(z) • type(int)
x, y, z are integers	declare_multi_var	• name(x) • name(y) • name(z) • type(integer)
Containers		
P: array of floats	declare_array	• name(P) • type(float)
P is a list of integers	declare_array	• name(P) • type(integer)
P is an array of integers	declare_array	• name(P) • type(integer)
x: list of integers	declare_array	• name(x) • type(integer)
x: array of integers	declare_array	• name(x) • type(integer)

1.2. Assignment

Statements	**Intent**	**Entities**
A is 12	assign	• name(A) • value(12)
A = 12	initialize_assign	• name(A) • value(12)
A = [1,2,3]	initialize_assign	
A = {1,2,3}	initialize_assign	

A = 12.05	initialize_assign	• name(A) • value(12.05)
A = “hello”	initialize_assign	• name(A) • value(“hello”)
A = B	initialize_assign	• name(A) • value(B)
Arr is [1,2,3]	initialize_assign	
x = y	initialize_assign	• name() • value()
x = true	initialize_assign	• name() • value()
x = false	initialize_assign	• name() • value()
x = 0	initialize_assign	• name() • value()
x = 0.0001	initialize_assign	• name() • value()
x = 1,2,3	initialize_assign	• name() • value()
x = 1, 2, 3	initialize_assign	• name() • value()
x = {1, 2, 3}	initialize_assign	• name() • value()
x = [1, 2, 3]	initialize_assign	• name() • value()
x = add(1,2)	initialize_assign	• name() • value()
x = a - b / c * (d+ e)	initialize_assign	2. name() 3. value()

3.1. Control

3.1.1. Conditional

Statement	Intent	Entity
If a>1	begin_if	condition(a>1)
Otherwise	begin_else	none
else	begin_else	none
Else if a==1	begin_else_if	condition(a==1)
Elif a==1	begin_else_if	condition(a==1)
Switch a	begin_switch	switch_var(a)
Case ‘1’ =	begin_case	case_value(‘1’)
Case 1 :	begin_case	case_value(1)
switch(a)	begin_switch	switch_var(a)
Case 1 ->	begin_case	case_value(1)
Otherwise if (a==1)	begin_else_if	condition(a==1)
Unless a<1	begin_unless	condition(a<1)
If a is greater than 1	begin_if	condition(a is greater than 1)
If a is equal to 1	begin_if	condition(a is equal to 1)
If a is not 1	begin_if	condition(a is not 1)
If a is lesser than 1	begin_if	condition(a is lesser than 1)

3.1.2. Loop

Statement	Intent	Entity
for every element e in container	begin_for_each	loop_over(container) loop_as(e)
for every e in c	begin_for_each	loop_over(c) loop_as(e)
while a is not 0	begin_while	condition(a is not 0)
while a != 0	begin_while	condition(a != 0)
until a is not 0	begin_until	condition(a is not 0)

for each item in items	begin_for_each	loop_over(items) loop_as(item)
for every item in items	begin_for_each	loop_over(items) loop_as(item)
unless a is 0	begin_unless	condition(a is 0)
do	begin_do	none

3.2. I/O

3.2.1. Console

3.2.1.1. Input

Statements	**Intent**	**Entities**
Input a	input	var_name(a)
Get value of a from user	input	var_name(a)
Input a, b	input	var_name(a) var_name(b)
Prompt user to enter a	input	var_name(a)
Get arr	input	var_name(arr)
Take arr as input	input	var_name(arr)
Get value of a, b from user	input	var_name(a) var_name(b)
Get a,b	input	var_name(a) var_name(b)
Take a, b as input	input	var_name(a) var_name(b)

3.2.1.2. Output

Statements	**Intent**	**Entities**
Print “hello”	print	to_print(“hello”)
print “some string of text”	print	to_print(“some string of text”)
Print hello	print	to_print(hello)

Print value of hello	print	to_print(hello)
Display total_marks	print	to_print(total_marks)
Print a	print	to_print(a)
Output student.marks	print	to_print(student.marks)
Display arr	print_elements	to_print(a)
Print all values of arr	print_elements	to_print(a)
Output arr	print_elements	to_print(arr)
Print all elements of arr	print_elements	to_print(arr)

3.2.2. File*

4. Data types

4.1. Built-in

4.1.1. Integer

int, integer

4.1.2. Real

real, float, double

4.1.3. Complex

complex

4.1.4. String

string, str

4.1.5. Character

char, character

4.1.6. Boolean

bool, boolean

4.2. User-defined

Would work like classes in Python.

```
type Graph
      nodes is a list of Nodes
      edges is a list of Edges
```

```
Graph type
      nodes: list of Nodes
      edges: list of Edges
```

3. Containers

3.1. Array/list

x is a list of integers

x is an array of integers

x: list of integers

x: array of integers

3.2. Hash map* / dictionary*

4. Functions

Statement	Intent	Entity
Def add(int a, int b)		
Define add(int a, int b)		
add(a int, b int) int		
add(int a, int b)		
add(int a, int b) int		
add(a,b)		
add(a, add(a,b))		

5. Import

Statement	Intent	Entity
Import file		
Import dir.file		
Import dir1.dir2.file		
Import file.function		
From file import function, global		
From dir.file import function, global		
From dir1.dir2.file import function, global		

6. Comments

Will be implemented using a parser

Statements	Intent	Entities
# this is comment		
// this is comment		
; this is comment		
/* Multi Line */		

Operators Handling

Arithmetic

Comparison

Logical

Assignment

Bitwise

Statement	**Intent**	**Entity**
A + b		
10 - 12		
19 * A		
Student.marks / 10		
-a		
Add a and b		
Add a,b		
Subtract a from b		
A ** 2		
A raise to the power of 2		
Increment a		
++a		
Are a and b equal?		
A == b		
A = b ?		
A > b ?		
Is a greater than b?		
Is a less than b?		

a+=10		
Increment a by 10		
Decrement Student.marks by 10		
A & b		
A bitwise and b		
Perform bitwise xor between a and b		
a==b and b==c		
a==b or b<c		
Not a		