Manual de usuario para el lenguaje *Statlang*

Índice

[Declaración de variables 1](#_Toc119512742)

[Acceso a variables 1](#_Toc119512743)

[Constantes 2](#_Toc119512744)

[Operaciones 2](#_Toc119512745)

[Estatutos 3](#_Toc119512746)

[Condiciones 4](#_Toc119512747)

[Ciclos 5](#_Toc119512748)

[Declaración de funciones 5](#_Toc119512749)

[Llamadas a funciones 6](#_Toc119512750)

[Funciones especiales 6](#_Toc119512751)

# Declaración de variables

Las variables se declaran al inicio del programa (para las globales) o de la función (para las locales). En *main* no se declaran variables; se usan las globales. La sintaxis para la declaración de variables es la siguiente:

var

    int num;

    float j;

    int arr[10];

    float floatarr[92];

    int mat[12][5];

    float fmat[4][1];

Los nombres de variables deben empezar con una letra y pueden consistir de letras y números.

Se pueden declarar dos tipos básicos de variables: *int*s y *float*s. También se pueden declarar arreglos y matrices de estos dos tipos básicos. Si se quiere declarar un arreglo, se pone [ (*tamaño)* ] después de la variable, y para una matriz, se añade otro [ *(tamaño)* ].

El tamaño máximo de cada dimensión de una variable no atómica es 100.

# Acceso a variables

Cuando se intente usar una variable, el lenguaje le da prioridad a las que existen en el contexto local. Si la variable no existe localmente, entonces usa la global. Así, si hay una variable local y una global que se llaman igual, usará la local.

El acceso a variables no atómicas se realiza de la siguiente manera:

arr[5]

fmat[0][0]

La indización de los arreglos va de 0 a (*tamaño de la dimensión – 1)*, como en C.

# Constantes

En este lenguaje pueden usarse 3 tipos de constantes: *ints, floats*, y *string*s. Los *int*s y *float*s pueden ser tanto positivos como negativos. Los *string*s deben estar rodeados por “.

5

-4

3.5

-6.8

"hola"

# Operaciones

Pueden utilizarse las siguientes operaciones:

exp + exp

exp - exp

exp \* exp

exp / exp

exp < exp

exp > exp

exp <> exp

exp == exp

exp | exp

exp & exp

Las expresiones usan asociatividad izquierda, y se sigue el siguiente orden de prioridades:

*Mayor prioridad* -------------> *Menor prioridad*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| \* / | + - | < > <> == | & | | |

Pueden usarse paréntesis para alterar el orden de las operaciones.

(exp1 + exp2) \* exp3

En caso que se sumen, resten, multipliquen o dividan un *int* y un *float*, el resultado se convertirá a *float*.

Nótese que este lenguaje utiliza lógica numérica: un valor de 0 es considerado “false”, y cualquier otro valor es considerado “true”. Por esto, las operaciones booleanas regresan un int que puede ser 0 o 1.

# Estatutos

Pueden utilizarse los siguientes estatutos:

id = exp

read("nombre.json", matriz)

write("resul", exp, ...)

return exp

El estatuto *read* se usa para poblar una matriz con un conjunto de datos. Estos datos deben estar en un archivo JSON, y deben seguir el siguiente formato:

{

    "0": {

        "0": 5,

        "1": -7,

        "2": 0,

    },

    "1": {

        "0": -45,

        "1": 23,

        "2": 1,

    }

}

Al usar el *read*, debe tenerse cuidado de que la matriz tenga las dimensiones exactas para el conjunto de datos que se quiere copiar. También debe recordarse que los datos deben ser todos del mismo tipo, ya sea *int* o *float*, y que no deben exceder los 100 elementos por dimensión.

Con *write*, pueden mostrarse valores en consola. Estos valores pueden ser variables, constantes, o expresiones.

El *return* sólo debe usarse dentro de funciones que regresen un valor; no puede usarse en una función *void* o en *main*.

# Condiciones

Pueden usarse estatutos condicionales de la siguiente manera:

if (exp) {

    ...

}

También puede añadirse un *else* después del if:

if (exp) {

    ...

}

else

{

    ...

}

Pueden anidarse condiciones:

if (n == 0) {

    return 0

}

else {

    if (n == 1) {

            return 1

    }

    else {

        m = fib(n-1)

        p = fib(n-2)

        return m+p

    }

}

# Ciclos

Para usar ciclos se usa la siguiente sintaxis:

from exp1 to exp2 do

{

    ...

}

El ciclo se repetirá *exp2 – exp1* veces.

Pueden anidarse ciclos:

from 0 to 3 do

{

    write("on i", i)

    j = i+1

        from j to 5 do

        {

            write("on j", j)

            j = j+1

        }

        i = i+1

}

# Declaración de funciones

Las funciones se declaran después de la declaración de variables globales, y antes del *main*. La sintaxis usada es la siguiente:

func int mifuncion (int n, float k);

var

   int m;

   float p;

{

    return n

}

Los parámetros de una función únicamente son de entrada, y la función puede ser de tipo *int*, *float* o *void*. Si es *int* o *float*, debe tener un estatuto *return*; si es *void*, no debe tenerlo.

# Llamadas a funciones

Si una función regresa *int* o *float*, debe ser llamada como parte de una expresión.

i = mifuncion(3,4.0) + 5

Por otro lado, si la función es *void*, debe llamarse como un estatuto independiente.

funcvoid(5.6)

El lenguaje no soporta el anidamiento de funciones.

# Funciones especiales

La sintaxis para las funciones especiales es la siguiente:

someNum = mean(matriz)

someNum = median(matriz)

someNum = mode(matriz)

someNum = variance(matriz)

someNum = stddev(matriz)

histogram(matriz)

boxplot(matriz)

Todas estas funciones toman un único parámetro, una matriz.

Las primeras 5 funciones (*mean, median, mode, variance* y *stddev*) regresan un *float*. A continuación se describe lo que hace cada una de estas primeras 5 funciones:

*mean***:** calcula la media de los datos.

*median*: calcula la mediana de los datos.

*mode*: calcula la moda de los datos.

*variance*: calcula la varianza de los datos

*stddev*: calcula la desviación estándar de los datos.

Las funciones *histogram* y *boxplot* generan diagramas estadísticos en una nueva ventana. Son funciones *void*. A continuación se describe lo que hacen:

*histogram*: grafica los datos en un histograma.

*boxplot:* grafica los datos en un diagrama de caja.