MILLA

Documentatie

Limbaj de programare specializat pentru jocuri 2D

Micu Sebastian

Solca Dragos-Catalin

1220bF

Cap 1. Care sunt tipurile de date?

Simple:

- NUMAR INTREG (pozitiv, negativ -> **5, 13, 20**);
- NUMAR CU VIRGULA (pozitiv, negativ -> **3.2, -7.5, 14.6**);
- LITERA (mica,mare -> **a, A**);

Complexe:

- CUVANT (format din mai multe litere sau cifre si sa nu inceapa cu cifra -> **sebi0**, **Dragos**, **Atmega324**);
- PROPOZITIE (orice scris intre **” ”** -> **” I love ALF”);**
- OBIECT (animal, masina, om, instrument, mancare etc.) -> cu ajutorul unui „click” putem sa il programam sa faca o actiune (mers, salt, rostogolit, vorbit etc.);

Cap 2. Cum se definesc variabilele?

Pentru variabile, declaram astfel:

variabila_mea = TIP DE DATA/EXPRESIE (exemplu: **a=5** sau **a=2*3**)

Pentru obiecte, declaram astfel:

OBIECT nume_de_obiect (exemplu: **OBIECT masina**);

Cap 3. Cum se reprezintă în memorie tipurile de date?

Milla este un **limbaj interpretat**, toate variabilele sunt alocate pe **heap** (dinamic).

Cap 4. Cum arată expresiile (adunare, scădere și alți operatori specifici)?

Adunare:

variabila1 + variabila2

Scadere:

variabila1 - variabila2

Inmultire:

variabila1 * variabila2

Impartire:

variabila1 / variabila2

Comparatie:

variabila1 > variabila2

variabila1 == variabila2

variabila1 < variabila2

Rotunjire:

ROT NUMAR (exemplu: **ROT 2.9** (rezultatul este **3**); **ROT 1.4** (rezultatul este **1**))

Putere:

NUMAR LA NUMAR (exemplu: **2 LA 3** (rezultatul est **8**))

Radical (radicalul este mereu de ordinul 2):

RAD NUMAR (exemplu: **RAD 2** (rezultatul este **1.41**))

Functii trigonometrice (*pentru avansati, rezultatul este cu doua zecimale dupa virgula*):

SIN NUMAR (exemplu: **sin 0** (rezultatul este **0**))

COS NUMAR (exemplu: **cos 45** (rezultatul este **0.7**))

TAN NUMAR (exemplu: **tan 60** (rezultatul este **1.73**))

COT NUMAR (exemplu: **cot 45** (rezultatul este **1**))

Cap 5. Cum se scriu functiile(actiunile) unui obiect?

Milla are urmatoarele functii (actiuni care pot fi facute):

Obiectul (personajul) poate sa:

- **mearga:** numeObiect MERGE 10 (numarul reprezinta cate casute merge – 1 casuta = 50 pixeli)
- **sara:** numeObiect SARE 5 (numarul reprezinta cate casute sare)
- **se rostogoleasca:** numeObiect ROSTOGOL 3 (numarul reprezinta valoarea unghiului la care se intoarce personajul – 1 unitate = 45 grade)
- **vorbeasca:** numeObiect SPUNE propozitie (exemplu: Caine SPUNE ”ham ham”)

Cap 6. Cum se scriu instructiunile?

Instructiuni de decizie

DACA expresie ATUNCI instructiune

DACA expresie ATUNCI instructiune ALTFEL instructiune

(exemplu: **DACA** variabila_mea > 5 **ATUNCI** SPUNE “mai mare “ **ALTFEL** SPUNE “mai mic “)

Instructiuni cu repetitie

REPETA expresie instructiuni (exemplu: **REPETA** 5 SPUNE “ham-ham “ -> **spune** “ham-ham “ **de 5 ori**)

Instructiuni cu enumerari

PENTRU variabila DIN numar LA numar instructiune (exemplu: **PENTRU** i **DIN** 0 **LA** 10 SARE 5)

Cap 7. Cum se scriu comentariile?

Pentru a face mai usoara scrierea de cod putem adauga cate un comentariu pe linie astfel:

```
!obiectul merge 5 casute!
```

```
!obiectul spune ceva!
```

ANEXA - WebAssembly

Declarare variable:

i32.const EXPRESIE

i32.const VARIABILA

i32.store

Tipuri de date:

Numar intreg: i32.const NUMAR

Numar cu virgula: f32.const NUMAR

Litera: (module

(import "io" "readchar" (func $readchar (result i32)))

(import "io" "writechar" (func $writechar (param $char i32)))

(func $start (local $a i32)

call $readchar

set_local $a

get_local $a

call $writechar)

(start $start))

Cuvant: (module

(import "io" "mem" (memory 1))

(import "io" "writestr" (func $writestr (param $strAddr i32)))

(data $s (i32.const 0) "masina\00")

(func $start

i32.const 0

call $writestr)

(start $start))

MERGE: (module

(import “milla” “merge” (func $merge (param i32)))

(func $start (local $i i32)

get_local $i

call $merge

set_local $i)

(start $start))

SARE: (module

(import “milla” “sare” (func $sare (param i32)))

(func $start (local $i i32)

```
                    get_local $i
                    call $sare
                    set_local $i)
        (start $start))
```

ROSTOGOL:

```
(module
    (import “milla” “rostogol” (func $rostogol (param i32)))
        (func $start (local $i i32)
                    get_local $i
                    call $rostogol
                    set_local $i)
        (start $start))
```

SPUNE:

```
(module
        (import "milla" "mem" (memory 1))
        (import "milla" "spune" (func $spune (param $strAddr i32)))
        (data $s (i32.const 0) "ham-ham\00")
            (func $start
                    i32.const 0
                    call $spune)
        (start $start))
```

Bibliografie:

Ca inspiratie pentru functiile predefinite (MERGE, SARE, ROSTOGOL, SPUNE), ne-am folosit de limbajul de programare **Scratch**.

Pentru sintaxa instructiunilor de decizie/ cu repetitie, am avut ca sursa de inspiratie limbajul natural, cunoscut si sub numele de **pseudocod**.

Am ales aceste doua limbaje deoarece sunt cunoscute pentru simplitatea lor si, oricine, chiar si fara cunostinte in programare, le poate invata cu usurinta.

- ✓ https://scratch.mit.edu/
- ✓ https://pseudocod1.weebly.com/sintaxa.html