

**ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΕΧΝΟΛΟΓΙΑ ΛΟΓΙΣΜΙΚΟΥ – ΗΥ352**

**ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 2016
ΔΙΔΑΣΚΩΝ: ΑΝΤΩΝΙΟΣ ΣΑΒΒΙΔΗΣ**

ΕΡΓΑΣΙΑ (ομάδες δύο ατόμων)

Ανάθεση: Παρασκευή 18 Νοεμβρίου 2016

Παράδοση: Πριν την εξεταστική (ακριβής ημερομηνία θα ανακοινωθεί όταν θα πλησιάζουμε σε εκείνη την ημερομηνία)

Θέμα – Κατασκευή γλώσσας για JSON (JavaScript Object Notation)

Θέμα της εργασίας είναι η κατασκευή μιας γλώσσας ειδικού σκοπού για τον ορισμό και χρήση δεδομένων σε μορφή JSON. Η γλώσσα που θα κατασκευάσετε (περιγράφεται παρακάτω) θα γίνεται compiled ως C++ οπότε θα χρειαστείτε ένα ή περισσότερα header files με κατάλληλους ορισμούς ώστε το πρόγραμμά σας που είναι γραμμένο στη γλώσσα ειδικού σκοπού να αντιστοιχεί σε valid C++ κώδικα και να κάνει compile σωστά. Όπως και στη C++ τα whitespaces αγνοούνται. Ένα πρόγραμμα θα έχει πάντα την εξής μορφή:

```
#include <JSONlang.h>
json variable definition1
json variable definition2
...
PROGRAM_BEGIN
    json variable definition3
    ...
    json editing statement1
    json variable definitionk
    ...
    json editing statement2
    ...
    json editing statementn
PROGRAM_END
```

Στοιχεία της γλώσσας

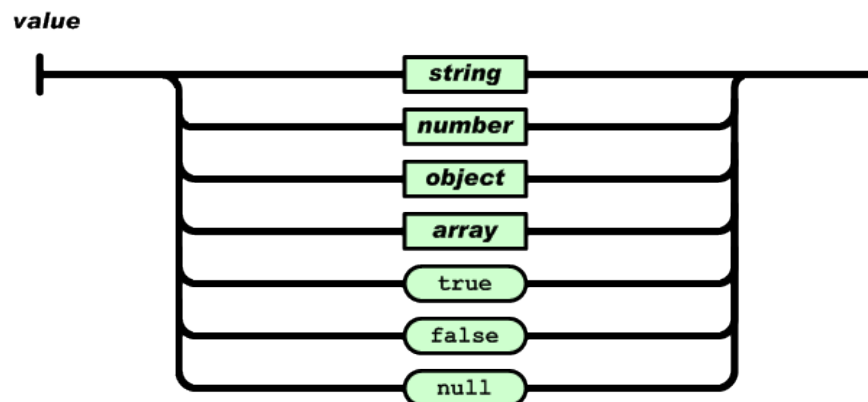
Ορισμός json variable

Η δήλωση ενός json variable γίνεται με το παρακάτω συντακτικό (προσέξτε ότι στο τέλος κάθε δήλωσης δεν υπάρχει semicolon):

➤ **JSON**(*name*) = *value*

Το *name* είναι το αναγνωριστικό που αντιστοιχεί στο συγκεκριμένο json variable ώστε να μπορούμε να αναφερθούμε σε αυτό στη συνέχεια.

Το *value* είναι η τιμή που θα πάρει το json variable. Ως value θεωρούμε οτιδήποτε από αυτά που φαίνονται στην παρακάτω εικόνα:



Στον παρακάτω πίνακα ακολουθεί η ανάλυση για κάθε κατηγορία value της γλώσσας:

| <i>Value Type</i> | <i>Επεξήγηση</i> | <i>Παράδειγμα</i> | <i>Συντακτικό στη γλώσσα μας</i> |
|-------------------|---|---|--|
| <i>string</i> | Αλφαριθμητικό (ακολουθία από χαρακτήρες) | "example" | STRING("example") |
| <i>number</i> | Αριθμός (ακέραιος ή δεκαδικός) | 23 | NUMBER(23) |
| <i>true</i> | Αληθής | true | TRUE |
| <i>false</i> | Ψευδής | false | FALSE |
| <i>null</i> | No value | null | NULL |
| <i>object</i> | Μη διατεταγμένο σύνολο από ζευγάρια κλειδιών με τιμές. Μεταξύ τους τα ζευγάρια χωρίζονται με κόμμα. Ένα object μπορεί να έχει κανένα, ένα ή περισσότερα ζευγάρια. | { "title : "C++11", "year": 2011 } | OBJECT{ KEY(title) : STRING("title"), KEY(year) : NUMBER(2011) } |
| <i>array</i> | Διατεταγμένο σύνολο από τιμές. Οι τιμές μεταξύ τους χωρίζονται με κόμμα. Ένας πίνακας μπορεί να περιέχει καμία, μία ή περισσότερες τιμές. | [] // empty array [2016, "HY352", { "id": 352 }] | ARRAY // empty array ARRAY[NUMBER(2016), STRING("HY352"), OBJECT{ KEY(id): NUMBER(352) }] |

Παραδείγματα:

```
//define emptyObj json with empty object
JSON(emptyObj) = OBJECT{}
//define emptyArray json with empty array
JSON(emptyArray) = ARRAY
//define book json with an object containing data for a book
JSON(book) = OBJECT {
  KEY(title) : STRING("Gone Girl"),
  KEY(published) : NUMBER(2012),
  KEY(type) : STRING("Thriller"),
  KEY(author) : OBJECT{
    KEY(firstname) : STRING("GILLIAN"),
    KEY(sirname) : STRING("FLYNN"),
    KEY(age) : NUMBER(45)
  }
}

//define week_temperatures json with an array of numbers
JSON(week_temperatures) = ARRAY [
  NUMBER(20), NUMBER(19.5), NUMBER(19), NUMBER(20),
  NUMBER(19), NUMBER(18.5), NUMBER(19)
]
```

```
//define students json with an array of objects representing students
JSON(students) = ARRAY [
    OBJECT{
        KEY(name)    : STRING("Nikos Nikolaou"),
        KEY(id)      : NUMBER(4444),
        KEY(grades)  : ARRAY[
            OBJECT { KEY(hy100) : NUMBER(7.5) },
            OBJECT { KEY(hy150) : NUMBER(8) },
            ...
        ]
    },
    ...
]
```

Επεξεργασία json variables και values

Η γλώσσα θα πρέπει να υποστηρίζει την επεξεργασία τιμών των json variables που έχουν οριστεί. Για αυτό το λόγο υπάρχουν οι παρακάτω ενέργειες:

➤ **SET** *json_lvalue* **ASSIGN** *value*

Αναθέτει τιμή σε εσωτερικούς κόμβους ενός json variable ή και σε ολόκληρο το json variable. Σε περίπτωση που δεν υπάρχει ο κόμβος προστίθεται ένα νέο στοιχείο. Παραδείγματα με βάση τα παραπάνω ορισμένα json variables:

```
//change 3rd day temperature from 19 to 22
SET week_temperatures[2] ASSIGN NUMBER(22)
//add email address for 1st student
SET students[0]["email"] ASSIGN STRING("csd444@csd.uoc.gr")
//assign new object in emptyObj json
SET emptyObj ASSIGN OBJECT {
    KEY(a) : STRING("alpha")
}
```

➤ **SET** *json_array* **APPEND** *value1, value2, ...*

Προσθέτει τα values που δίνονται ως όρισμα (value1, value2, ...) στο json value που θα πρέπει να είναι πίνακας. Για παράδειγμα, αν θέλουμε να προσθέσουμε 3 επιπλέον θερμοκρασίες στο json week_temperatures:

```
//appends values 23, 22, 20 to the end of the temperature array
SET week_temperatures APPEND NUMBER(23), NUMBER(22), NUMBER(20)
//appends a grade for course hy255
SET students[0]["grades"] APPEND OBJECT { KEY(hy255) : NUMBER(9) }
```

➤ **ERASE** *json_value_or_variable*

Διαγράφει εσωτερικά json values από τα json objects ή arrays στα οποία περιέχονται. Σε περίπτωση json variable (που θα πρέπει να είναι object ή array) σβήνει όλα τα στοιχεία του. Παραδείγματα:

```
ERASE book["author"]["age"] //removes age from author object of book
ERASE book["type"] //removes type of book
ERASE book //removes all book data, now book is an empty object
```

➤ Χρήση των `json values` ως εκφράσεις με υποστήριξη αριθμητικών, λογικών, συσχετιστικών τελεστών και τελεστών ισότητας

- Οι αριθμητικοί τελεστές (+, -, *, /, %) και συσχετιστικοί τελεστές (>, >=, <, <=) υποστηρίζονται μόνο μεταξύ αριθμών.
- Ειδική περίπτωση είναι ο τελεστής + που μπορεί να χρησιμοποιηθεί επίσης και μεταξύ strings, arrays ή object.
 - Για τα αλφαριθμητικά το + θα ενώνει τα αλφαριθμητικά. Π.χ. Η πράξη `STRING("hello") + STRING(" world")` θα έχει ως αποτέλεσμα το value `STRING("hello world")`.
 - Για τα ARRAY και OBJECT θα δημιουργεί νέο value με την ένωση των τιμών. Π.χ. Η πράξη `ARRAY[NUMBER(1), NUMBER(2)] + ARRAY[NUMBER(3), NUMBER(4)]` θα έχει ως αποτέλεσμα το value `ARRAY[NUMBER(1), NUMBER(2), NUMBER(3), NUMBER(4)]`
- Οι λογικοί τελεστές (&&, ||, !) υποστηρίζονται μόνο μεταξύ boolean values.
- Οι τελεστές ισότητας (==, !=) υποστηρίζονται μεταξύ values του ίδιου τύπου
 - Για arrays και objects συγκεκριμένα η ισότητα θα πρέπει να τσεκάρει αναδρομικά να είναι ίσα όλα τα περιεχόμενα τους
- Οποιοσδήποτε άλλος συνδυασμός τελεστών θα θεωρείται λάθος και θα πρέπει η γλώσσα να πετάει αντίστοιχο μήνυμα λάθους.

Οι αριθμητικές πράξεις μπορούν να χρησιμοποιηθούν και κατά το definition των json, π.χ.:

```
JSON(hy352_nik) = OBJECT{ KEY(exam):NUMBER(7), KEY(project):NUMBER(8) }
JSON(students) = ARRAY [
  OBJECT{
    KEY(name) : STRING("Nikos ") + STRING("Nikolaou"),
    KEY(id) : NUMBER(4444),
    KEY(grades) : ARRAY[
      OBJECT {
        KEY(hy352):
          hy352_nik["exam"] * NUMBER(0.75) + hy352_nik["project"] * NUMBER(0.25)
      }
    ]
  }
]
```

Επιπλέον η γλώσσα θα πρέπει να υποστηρίζει τις παρακάτω τελεστές

➤ `SIZE_OF(json_value_or_variable)`

- Επιστρέφει το πλήθος των στοιχείων σε ένα array ή object (για τις υπόλοιπες τιμές επιστρέφει 1)

➤ `IS_EMPTY(json_value_or_variable)`

- Ελέγχει αν ένα array ή object είναι κενό (για τις υπόλοιπες τιμές επιστρέφει false)

➤ `HAS_KEY(json_value_or_variable, key)`

- Ελέγχει αν υπάρχει το κλειδί στο object (για τις υπόλοιπες τιμές επιστρέφει false)

➤ `TYPE_OF(json_value_or_variable)`

- Επιστρέφει τον τύπο του ορίσματος του ως string (δηλαδή ένα εκ των "string", "number", "boolean", "object", "array", "null").

Εκτύπωση *json variables και values*

Η γλώσσα πρέπει να υποστηρίζει την εκτύπωση json expressions με το εξής συντακτικό:

➤ **PRINT** *json_expression₁, json_expression₂, ...*

Παραδείγματα:

```
PRINT book["title"] //prints:Gone Girl
PRINT book["author"] //prints:{firstname:"Gillian",sirname:"Flynn",age: 45}
PRINT book //prints: the whole json for book
PRINT HAS_KEY(book, "author") //prints: true
//prints: Book has key author? True
PRINT STRING("Book has key author? "), HAS_KEY(book, "author")
```

Hints

Για να μετατρέψετε το συντακτικό της γλώσσας σε *valid C++* χρησιμοποιήστε:

- ✧ Τη δυνατότητα για operator overloading που σας προσφέρει η C++, δίνοντας μεγάλη προσοχή στην προτεραιότητα των operators.
 - `operator[]`
 - Για τον ορισμό arrays
 - `operator,`
 - Για να μαζεύετε εκφράσεις που έχουν κόμμα ανάμεσα τους, πχ για τα arrays, για print πολλαπλών values κλπ.
- ✧ Δημιουργία προσωρινών στιγμιότυπων ως επιστρεφόμενα αποτελέσματα, αλλά και ως βοηθητικά στιγμιότυπα σε εκφράσεις.
- ✧ Αρκετά τον preprocessor αφού λέξεις κλειδιά της γλώσσας όπως *JSON*, *OBJECT*, *ARRAY*, *STRING*, κτλ. θα είναι macros που θα κρύβουν μετατροπές σε strings, κλήσεις συναρτήσεων, κάποιους operators ή και βοηθητικά προσωρινά στιγμιότυπα.
- ✧ Τους initializers της C++11 `-{}-` για κλήση constructors και αρχικοποίηση μεταβλητών