Β. Εισαγωγή στον Προγραμματισμό Η/Υ με την JavaScript

Β.1 Τύποι Δεδομένων

Όλες οι γλώσσες προγραμματισμού (πρέπει να) υποστηρίζουν πέντε (5) πρωταρχικούς τύπους δεδομένων:

```
char (character)

int (integer)

float (floating point)

double (double floating point)

boolean (true/false)
```

Επειδή η γλώσσα JavaScript δεν είναι αυστηρά δομημένη γλώσσα (strongly typed) --δηλ. δεν κάνει κατά την ώρα της μεταγλώττισης αυστηρό έλεγχο των τύπων που χρησιμοποιούνται-- μπορούμε να εκχωρούμε τιμές μεταξύ διαφορετικών τύπων (δίχως ο μεταγλωττιστής να το ελέγχει) με απρόσμενα πολλές φορές αποτελέσματα.

Η γλώσσα JavaScript υποστηρίζει τους παρακάτω πρωταρχικούς τύπους δεδομένων:

```
Number (που περιλαμβάνει int, float & double)
Boolean (true/false)
String (που περιλαμβάνει και τον τύπο char)
```

Β.2 Αναγνωριστικά και Δηλώσεις μεταβλητών

Αναγνωριστικό (identifier) είναι μια ακολουθία από γράμματα και ψηφία, όπου ο πρώτος χαρακτήρας είναι γράμμα. Στην JavaScript τα κεφαλαία γράμματα είναι διαφορετικά από τα πεζά (case sensitive). Τα αναγνωριστικά μπορούν να έχουν οποιοδήποτε μήκος και δεν πρέπει γενικώς να περιλαμβάνουν ειδικούς χαρακτήρες, εκτός από ελάχιστες εξαιρέσεις, όπως η κάτω παύλα. Τα αναγνωριστικά μπορούν να χρησιμοποιούνται στις δηλώσεις ονομάτων μεταβλητών, στην ονομασία συναρτήσεων και γενικά σε ονόματα συμβόλων που επιλέγουν οι προγραμματιστές.

Δηλώσεις μεταβλητών μπορούν να γίνουν στο σώμα των συναρτήσεων ως τοπικές μεταβλητές (local variables) και έξω από όλες τις συναρτήσεις ως καθολικές μεταβλητές (global variables).

Οι δηλώσεις μεταβλητών γίνονται όπως παρακάτω:

```
var a=5, b;
var ch='*';
var m;
```

Όπως φαίνεται παραπάνω στη JavaScript μπορούμε να κάνουμε είτε μόνο δηλώσεις ή ταυτόχρονα δηλώσεις και αρχικοποιήσεις μεταβλητών.

Β.3 Κυριολεκτικά (Literals)

Σταθερές τιμές μπορούμε να εκχωρούμε σε μεταβλητές, όπως φαίνεται παρακάτω:

```
var m = 12;  //ακέραιος
var ch = "*";  //χαρακτήρας
var myF = 12.5;  //δεκαδικός
```

Οι δηλώσεις σταθερών για αλφαριθμητικά (κυριολεκτικά αλφαριθμητικά – literal strings) γίνονται μέσα διπλά (ή μονά) εισαγωγικά: "-----". Αριθμοί εκχωρούνται χωρίς εισαγωγικά.

Στην JavaScript έχουμε κάποιες προκαθορισμένες σταθερές (ακολουθίες διαφυγής) που χρησιμοποιούνται κυρίως μέσα σε εντολές εκτύπωσης για την εκτύπωση ειδικών χαρακτήρων:

\n	αλλαγή γραμμής
\0	Μηδενικός χαρακτήρας (null)
\t	Οριζόντιος στηλογνώμονας (tab)
\"	διπλά εισαγωγικά
\´	απλά εισαγωγικά
\\	Ανάποδη κάθετος (backslash)

Β.4 Πίνακες (Arrays) και Δομές (Structs)

Πίνακας (array) είναι μία διατεταγμένη ακολουθία τιμών του ίδιου πρωταρχικού τύπου δεδομένων (αν και στην JavaScript μπορούμε να χρησιμοποιούμε και στοιχεία διαφορετικού τύπου μέσα στον ίδιο πίνακα).

Δηλώσεις πινάκων κάνουμε ως εξής:

```
var dim = new Array(5);    //Θέσεις από dim[0] έως dim[4]
var dim[];           // δήλωση δίχως διάσταση
var b[12,5,7,2]      // αυτόματα παράγεται η διάσταση
```

Με την πρώτη δήλωση δηλώνουμε τον πίνακα ως αντικείμενο πίνακα ενώ με τις δύο τελευταίες δηλώσεις δηλώνουμε πίνακες ως κυριολεκτικά.

Οι δομές (structs) περικλείουν τιμές διαφορετικών τύπων δεδομένων. Για παράδειγμα, ο ορισμός μιας δομής πελάτη (customer) θα μπορούσε να γίνει όπως παρακάτω:

```
var CUSTOMER = {
      name:"C. Papadopoulos",    //   ονοματεπώνυμο
      age:32,                    //   ηλικία
      height:1.90;               //   ύψος
};
```

B.5 Τελεστές

Η JavaScript παρέχει τέσσερις τύπους τελεστών: αριθμητικούς, σύγκρισης (ή συσχεσιακοί), λογικοί και τελεστές χειρισμού bits**.** Επίσης παρέχεται ο τελεστής εκχώρησης ‘ **=**’, που εκχωρεί μία τιμή σε μία μεταβλητή:

B.5.1 Αριθμητικοί τελεστές

-, +, *, /, % (mod), --, ++

Οι ++,--, αυξάνουν και μειώνουν αντίστοιχα την τιμή του τελεσταίου δηλαδή της μεταβλητής στην οποία εφαρμόζονται κατά ένα και μπορούν να τοποθετηθούν πριν ή μετά τον τελεσταίο. Στην περίπτωση που χρησιμοποιηθούν μπροστά τότε πρώτα εκτελείται η πράξη και μετά η εκχώρηση, ενώ αν τοποθετηθούν μετά, τότε πρώτα εκτελείται η εκχώρηση και μετά η πράξη.

Π.χ.:

```
a=1; b=--a; (Αποτέλεσμα b=0, a=0)
a=1; b=a--  (Αποτέλεσμα b=1, a=0)
```

Ο τελεστής + χρησιμοποιείται και ως τελεστής συνένωσης αλφαριθμητικών όταν οι τελεσταίοι είναι strings.

Π.χ.:

```
a="Hello"; b=" World!"; c=a+b;
(Αποτέλεσμα c="Hello World!")
```

Β.5.2 Συσχεσιακοί τελεστές

```
>          ΜΕΓΑΛΥΤΕΡΟ
>=         ΜΕΓΑΛΥΤΕΡΟ ή ΙΣΟ
<          ΜΙΚΡΟΤΕΡΟ
<=         ΜΙΚΡΟΤΕΡΟ ή ΙΣΟ
==         ΙΣΟ
!=         ΔΙΑΦΟΡΟ
```

Β.5.3 Λογικοί τελεστές

```
&&         AND   (ΛΟΓΙΚΟ ΚΑΙ)
||         OR    (ΛΟΓΙΚΟ Η)
!          NOT   (ΛΟΓΙΚΟ ΟΧΙ)
```

Β.5.4 Τελεστές χειρισμού bits

Οι τελεστές χειρισμού bits εφαρμόζονται στα ψηφία (bits) μιας ψηφιοσυλλαβής (byte):

```
&     AND
|     OR
^     XOR
~     συμπλήρωμα ως προς 1
>>    shift left
<<    shift right
```

Για παράδειγμα:

```
x=5;           x:     0 0 0 0 0 1 0 1
x=x & 255;  255:     1 1 1 1 1 1 1 1
                     ---------------
                     0 0 0 0 0 1 0 1
```

Β.6 Παραστάσεις

Με την χρήση των παραπάνω τελεστών μπορούμε να ορίζουμε παραστάσεις πράξεων, όπως:

```
i = i + 5;
j = j - 12;
k = k * 70;
```

Επίσης μπορούν να χρησιμοποιούνται συντμήσεις στον κώδικα των παραστάσεων:

```
i += 5;
j -= 12;
k *= 70;
```

Β.7 Είσοδος/Εξοδος

Για την εκτύπωση σταθερών και μεταβλητών η JavaScript παρέχει τις παρακάτω συναρτήσεις, που συντάσσονται ως εξής:

`document.write("string") ή document.write(s)`

`document.getElementById("myp").innerHTML="string"`

όπου s μεταβλητή και myp είναι το id ενός <p> tag.

Για το διάβασμα και την εκχώρηση σε μεταβλητές τιμές που δίνει ο χρήστης μπορούμε να χρησιμοποιούμε HTML Forms και να εκχωρούμε την ιδιότητα .value των στοιχείων της φόρμας σε JavaScript μεταβλητές.

Β.8 Το πρώτο πρόγραμμα στη Javascript

Στη HTML υπάρχει το tag <script> απ' όπου το πρόγραμμα ξεκινάει να εκτελείται.

```
<!DOCTYPE html>
<html lang="en-US">
<head>
      <title> First Programm in JS </title>
      <meta charset="UTF-8">
</head>
<body>
<script>
      document.write("<h1> First JS EXAMPLE IN PED-15 </h1>");
</script>
</body>
</html>
```

Β.9 Δομές Ελέγχου

Οι δομές ελέγχου χρησιμοποιούνται για τον έλεγχο της ροής του προγράμματος. Οι βασικές δομές ελέγχου είναι οι διακλαδώσεις (`if – then – else, switch`) και οι επαναλήψεις (`while, do-while, for`).

Β.9.1 Δομή if – then – else

Σύνταξη: `If (expr) {stmt; stmt; ---} else {stmt; stmt;---}`

Παράδειγμα:

```
if (i>0) pos=i; else neg=-i;
if (sex==”Man”) expectedTimeToLive = 80;
else expectedTimeToLive = 90;

FYLO = (sex==”Man”) ? ”Άνδρας” : ”Γυναίκα” ;
// Τριαδικός τελεστής συντόμευσης if-then-else
```

Στη μεταβλητή `FYLO` εκχωρείται η τιμή ”Άνδρας” αν η συνθήκη `(sex==”Man”)` είναι αληθής, αλλιώς εκχωρείται η τιμή ”Γυναίκα”.

Β.9.2 Δομή switch

Σύνταξη:

```
Switch (expr)
{
      case value1: stmt; break;
      case value2: stmt; break;
      ..............
      ..............
      default: stmt;
}
```

Παράδειγμα:

```
var foo1=3, foo2=5, tmp;
var operator="+";
switch (operator)
{
      case ‘+‘: tmp = foo1+foo2; break;
      case ‘-‘: tmp = foo1-foo2; break;
      case ‘*‘: tmp = foo1*foo2; break;
      case ‘/‘: tmp = foo1/foo2; break;
      default: tmp=0;
}
Document.write (tmp); // θα επιστρέψει 8
```

Β.9.3 Δομή while-do

Σύνταξη: `While (expr) <block>`

Παράδειγμα:

```
var j=1;
while (j<12)
{
      document.write("*");
      j++;
}
```

Β.9.4 Δομή do-while

Σύνταξη: Do <block> While (expr)

Το <block> της Do-While εκτελείται τουλάχιστον μια φορά αντίθετα με την while-do που μπορεί να μην εκτελεστεί καθόλου.

```
var i=1;
do
{
      i++;
      document.write("*");
 } while (i<12);
```

Β.9.5 Δομή for

Σύνταξη: for (αρχική συνθήκη; Συνθήκη εξόδου; βήμα) <block>

Παραδείγματα:

```
      for (i=0; i<90; i++) document.write("*");
// Τυπώνει ένα * σε κάθε επανάληψη (loop) της for

for (i=1; i<=10; i++)
{
      document.write("*");

}
// Τυπώνει 10 αστεράκια
```

Β.10 Εντολές μεταφοράς ελέγχου

Β.10.1 Εντολή return

Έχει ως αποτέλεσμα την μεταφορά της εκτέλεσης του κώδικα από τη συνάρτηση στο κύριο πρόγραμμα από όπου κλήθηκε η συνάρτηση. Αν το `return` περιέχει κάποια τιμή ή επιστρεφόμενη μεταβλητή --π.χ. `return (i)`, η τιμή ή η τιμή της μεταβλητής επιστρέφεται από τη συνάρτηση.

Β.10.2 Εντολή break

Η εντολή `break` τερματίζει την εκτέλεση μιας επαναληπτικής συνήθως δομής ελέγχου, όταν κάποια συνθήκη γίνει αληθής, ώστε να μην εκτελείται και ο υπόλοιπος κώδικα της δομής ελέγχου. Μία συνηθισμένη χρήση της `break` είναι στο σώμα της εντολής `switch`, όπου κάθε περίπτωση ελέγχου (case) έχει μία break, ώστε αν η συνθήκη είναι αληθής να μην εκτελεστούν και τα υπόλοιπα case.