ΚΕΦΑΛΑΙΟ 5

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΤΗΝ ΠΛΕΥΡΑ ΤΟΥ ΕΞΥΠΗΡΕΤΗΤΗ

5.1 Εισαγωγή

Οι δυναμικές ιστοσελίδες, που χρησιμοποιούν προγραμματισμό στην πλευρά του εξυπηρετητή, λαμβάνουν τα δεδομένα που εισάγει ο χρήστης, στη συνέχεια τα επεξεργάζονται ή ανατρέχουν σε κάποια βάση δεδομένων, που είναι εγκατεστημένη στον εξυπηρετητή, και δημιουργούν μία προσαρμοσμένη ιστοσελίδα την οποία και βλέπει ο χρήστης. Το περιεχόμενο της παραγόμενης ιστοσελίδας εξαρτάται από την ερμηνεία του προγραμματιστικού σεναρίου. Σε αντίθεση με την τεχνολογία, που χρησιμοποιείται στον προγραμματισμό από την πλευρά του πελάτη π.χ. JavaScript, ο προγραμματισμός από την πλευρά του εξυπηρετητή παράγει κανονικές ιστοσελίδες HTML. Οι ιστοσελίδες αυτές δεν περιέχουν άλλο κώδικα εκτός από τις ετικέτες HTML και μπορούν να εμφανίζονται ορθές σε όλους τους φυλλομετρητές, χωρίς να απαιτείται κάποια επιπλέον ρύθμιση σε αυτούς, π.χ. ενεργοποίηση της εκτέλεσης σεναρίων Java.

Για την κατασκευή αυτών των σεναρίων μπορούν να χρησιμοποιηθούν διάφορες τεχνολογίες και γλώσσες σεναρίων. Οι τρεις πιο διαδεδομένες τεχνολογίες, τις οποίες θα μελετήσουμε παρακάτω, είναι οι: CGI, PHP και ASP.

Η τεχνολογία **CGI** (Common Gateway Interface) αποτελεί μια διαδεδομένη μέθοδος προγραμματισμού από την πλευρά του εξυπηρετητή. Για την κατασκευή σεναρίων CGI μπορούν να χρησιμοποιηθούν πολλές γλώσσες γενικού προγραμματισμού, όπως C, C++ και η PERL, την οποία και θα μελετήσουμε. Η Perl (Practical Extraction και Report Language) είναι μία από τις πρώτες γλώσσες που χρησιμοποιήθηκαν για τον προγραμματισμό παγκόσμιο Ιστό. Ο αριθμός των έτοιμων βιβλιοθηκών προγραμμάτων που είναι διαθέσιμα αυτή την στιγμή την καθιστούν ένα ευπροσάρμοστο εργαλείο. Πολλοί άνθρωποι βρίσκουν δύσκολο τον προγραμματισμό σε Perl και αυτός είναι πιθανώς ο λόγος για τον οποίο έχουν γίνει δημοφιλέστερες, ευκολότερες και πιο εξειδικευμένες λύσεις. Αντίθετα από τον κώδικα PHP και ASP, η Perl δεν μπορεί να ενσωματωθεί στα έγγραφα HTML. Ολόκληρη η σελίδα HTML πρέπει να γραφτεί ως πρόγραμμα Perl που εξάγει τα αποτελέσματα της ιστοσελίδας στο φυλλομετρητή του χρήστη. Τα αρχεία που περιέχουν προγράμματα σε Perl έχουν συνήθως επέκταση ".pl".

Η **PHP** (Personal HomePage Tools) είναι μία γλώσσα και διερμηνέας σεναρίων που είναι ελεύθερα διαθέσιμη. Αρχικά χρησιμοποιήθηκε σε κεντρικούς υπολογιστές σε περιβάλλον UNIX, αλλά μπορεί να χρησιμοποιηθεί και με άλλες πλατφόρμες λειτουργικών συστημάτων. Μία σελίδα HTML που περιλαμβάνει σενάρια με PHP έχει τη χαρακτηριστική επέκταση ".php", ".php3", ".php 4" ή ".phtml".

Η τεχνολογία **ASP** (Active Server Pages) εμπεριέχεται σε σελίδες HTML που περιέχουν μικρά σενάρια τα οποία επεξεργάζονται στον εξυπηρετητή πριν σταλεί η σελίδα στον χρήστη. Η ASP αποτελεί χαρακτηριστικό γνώρισμα του Internet Information Server (IIS) της Microsoft. Για την κατασκευή ASP σεναρίων μπορούν να χρησιμοποιηθούν διάφορες γλώσσες όπως η VBScript, η Jscript, ακόμα και η Java.

Παρακάτω θα μελετήσουμε την VBScript. Τα αρχεία HTML που περιέχουν ASP σενάρια έχουν επέκταση ".asp".

Ο πίνακας 5.1.1 παρουσιάζει μία απλή σύγκριση των τριών αυτών τεχνολογιών.

Η σύγκριση των τεσσάρων τεχνολογιών είναι δύσκολη. Πολλοί προγραμματιστές προτιμούν τη μέθοδο που τυχαίνει να ξέρουν καλύτερα ή τη μέθοδο που είναι ήδη καθιερωμένη στο ίδρυμα στο οποίο εργάζονται ή σπουδάζουν. Η Perl μπορεί να απαιτεί περισσότερο χρόνο για την εκμάθησή της, σε σχέση με τις άλλες εναλλακτικές λύσεις αλλά είναι η πληρέστερη γλώσσα προγραμματισμού για τον Ιστό και παρέχει μεγαλύτερη ευχέρεια υλοποίησης. Προγραμματιστές που είναι εξοικειωμένοι με γλώσσες όπως Visual Basic, VBscript, ή JavaScript ενδεχομένως να βρουν την παραγωγή σελίδων σε ASP πολύ πιο εύκολη, αφού μπορούν να επαναχρησιμοποιηθούν άμεσα. Η ομαλή συνεργασία με τεχνολογίες όπως η XML ή με αντικείμενα Active X μπορεί να επιβάλει σε πολλές περιπτώσεις την χρήση της ASP. Η PHP συνδυάζει μία ισχυρή γλώσσα προγραμματισμού με σχετική ευκολία χρήσης και μπορεί να είναι η προφανής επιλογή για κάποιον λιγότερο έμπειρο προγραμματιστή, οποίος χρειάζεται γρήγορα αποτελέσματα χωρίς δαπάνες λογισμικού.

Πίνακας 5.1.1 – Σύγκριση τεχνολογιών Perl, PHP και ASP

	Perl	**PHP**	**ASP**
Κόστος	Δωρεάν	Δωρεάν	Η ASP διανέμεται δωρεάν με τον IIS της Microsoft
Ενσωμάτωση σε σελίδα HTML	Όχι	Ναι	Ναι
Ευκολία χρήσης	+	+++	++
Διεπαφή με βάση δεδομένων	+++	++	++
Χειρισμός κειμένου	+++	+++	+
Χειρισμός Εικόνων	++	++	+
Βασιζόμενοι στον χρόνο εκμάθησης και στην πολυπλοκότητα: +++: εύκολο, +: δύσκολο			

5.2 CGI

Το **CGI scripting** (Common Gateway Interface) είναι η μέθοδος με την οποία ένας υπολογιστής δικτύου μπορεί να λάβει τα στοιχεία από (ή να στείλει στοιχεία σε) βάσεις δεδομένων, έγγραφα, και άλλα προγράμματα και να παρουσιάσει τα στοιχεία αυτά σε ιστοσελίδες του Διαδικτύου. Ποιο απλά το CGI αποτελεί μέθοδο προγραμματισμού για το Διαδίκτυο.

Κανονικά όταν ένας φυλλομετρητής ανατρέχει σε ένα URL συμβαίνουν τα ακόλουθα. Πρώτα ο υπολογιστής έρχεται σε επαφή με τον κεντρικό υπολογιστή http, ο οποίος φιλοξενεί την ιστοσελίδα με το αντίστοιχο URL. Ο κεντρικός υπολογιστής HTTP εξετάζει το όνομα του αρχείου που ζητείται από τον υπολογιστή και το στέλνει. Ο φυλλομετρητής του τοπικού υπολογιστή παρουσιάζει έπειτα το αρχείο στην κατάλληλη μορφή.

Είναι δυνατό όμως, να μην επιστραφεί το αρχείο στο οποίο «δείχνει» το URL, αλλά να εκτελεστεί σαν ένα οποιοδήποτε πρόγραμμα και να επιστραφούν (στον υπολογιστή) τα αποτελέσματα της εκτέλεσης αυτής. Αυτή η λειτουργία καλείται Common Gateway Interface ή CGI. Τα προγράμματα αυτά καλούνται σενάρια CGI. Τα σενάρια **CGI** μπορούν να γραφτούν σε μία πληθώρα γλωσσών προγραμματισμού, με πιο δημοφιλή την **Perl.**

5.2.1Βασικοί κανόνες κατασκευής CGI σεναρίων σε Perl

Τοποθετούμε υποχρεωτικά την παρακάτω γραμμή στην κορυφή του σεναρίου Perl που θα γράψουμε:

```
#!/usr/local/bin/perl
```

Με αυτή τη γραμμή λέμε στον εξυπηρετητή, που θα εκτελέσει το σενάριο, ότι αυτά που ακολουθούν είναι σε γλώσσα Perl. Ο εξυπηρετητής πρέπει να γνωρίζει τι είναι αυτό που του ζητείται να εκτελέσει, έτσι δεν πρέπει να ξεχνάμε να ξεκινάμε τον κώδικα μας με αυτή τη γραμμή.

Στο τέλος κάθε γραμμής πρέπει να βάζουμε ένα ερωτηματικό (;). Αν δεν το βάλουμε, τότε το πρόγραμμα δεν θα εκτελεστεί ποτέ.

Βάζουμε σαφή σχόλια σε κάθε γραμμή κώδικα που γράφουμε, γεγονός απαραίτητο σε κάθε πρόγραμμα που δημιουργούμε. Κάθε γραμμή σχολίου ξεκινάει με το χαρακτήρα "#", που δηλώνει ότι πρόκειται για σχόλιο. Εξαίρεση αποτελεί η γραμμή που δηλώνει την γλώσσα συγγραφής του σεναρίου, δηλαδή η `#!/usr/local/bin/perl`. Τα παρακάτω αποτελούν μερικά παραδείγματα σχολίων:

```
# Με τον παρακάτω κώδικας θα κατασκευάσουμε μια μεταβλητή, η
τιμή # της οποίας θα αποθηκεύεται σε κάποιο αρχείο log.

my $in{'logdata'} = <<END;
$in{'name'}
$in{'formelement1'}   |   $in{'radiobutton2'}
$in{'checkbox1'}      |   $in{'checkbox2'}


END
```

Με τα σχόλια ο κώδικας γίνεται καλογραμμένος και μπορεί να επεξεργαστεί πιο εύκολα από κάποιον που δε τον έχει γράψει και προσπαθεί να τον καταλάβει και, ίσως, να τον τροποποιήσει.

Ο κώδικας πρέπει να είναι καθαρός, για να είναι συντηρήσιμος. Ο παρακάτω κώδικας δεν είναι λάθος, αλλά είναι όμως κακογραμμένος και δύσκολος στην ανάγνωση και κατανόηση.

```
for ($i=0;$i<=$#max;$i++){if ($max[$i]~=/hello there/g;)
{print  "bad code";}else { print "bye";}}
```

Ακόμα και με σχόλια θα έπαιρνε μέρες να καταλάβει ένα άτομο πως πρόκειται απλά για έναν βρόχο "for", που τρέχει διαμέσου ενός πίνακα "max", ψάχνοντας για την φράση "hello there". Αν η φράση υπάρχει, τότε εκτυπώνει "bad code"; αν δεν υπάρχει θα εκτυπώσει "bye". Ο παραπάνω κώδικας καλογραμμένος είναι:

```
for ($i = 0; $i <= $#max; $i++) {
       if ($max[$i] ~= /hello there/g;) {
              print "bad code";
       }else {
              print "bye";
       }
}
```

Είναι χρήσιμο να δίνουμε στις μεταβλητές ονόματα που περιγράφουν αυτό που δηλώνουν. Στα ονόματα μεταβλητών απαγορεύονται τα κενά αλλά επιτρέπεται ο χαρακτήρας υπογράμμισης (_), για να ξεχωρίσουμε τις λέξεις. Π.χ. το $book_page_i_am_on είναι πιο κατανοητό από το $bookpageiamon.

Στα ονόματα των μεταβλητών μπορούμε να χρησιμοποιήσουμε όλα τα γράμματα (κεφαλαία και πεζά) της αλφαβήτου, αλλά όχι άλλους ειδικούς χαρακτήρες, όπως π.χ. !, @, #, $, %, ^, &, *, (,), ~ κτλ..

Έχοντας δει τους κανόνες, μπορούμε να προχωρήσουμε στην εκμάθηση των βασικών εντολών της **Perl.**

5.2.2 Οι μεταβλητές στην Perl

Οι μεταβλητές στην Perl μπορούν να είναι ένας από τους παρακάτω τρεις τύπους:

- βαθμωτή (scalar),
- πίνακας (array), ή
- συσχετιζόμενος πίνακας (associative array).

<u>Βαθμωτές μεταβλητές</u>

Μία βαθμωτή μεταβλητή μπορεί να περιέχει αριθμούς, γράμματα, φράσεις, κτλ. Το σύμβολο του δολαρίου ($) προηγείται πάντα αυτής της μεταβλητής. Έτσι αν θελήσουμε να δώσουμε μία τιμή στην μεταβλητή $comp, γράφουμε:

Μία λέξη: `$comp = "word";`
Μία φράση: `$comp = "This is a phrase!";`
Αριθμούς: `$comp = 2;`

Μπορούμε επίσης να προσθέτουμε και να αφαιρούμε ως εξής:

```
$comp= 2 + 2;
$comp= 4 - 2;
```

Η `$comp` θα ήταν ίση με 4 + 2 ή 4-2, αντίστοιχα.

Οι εκχωρήσεις σε μεταβλητές γίνονται πάντα από αριστερά στα δεξιά. Π.χ. `2 + 2 = $comp;` είναι λάθος.

<u>Μεταβλητές τύπου πίνακα</u>

Μία μεταβλητή τύπου πίνακα κρατάει πολλές βαθμωτές μεταβλητές τύπου σε αριθμημένες θέσεις. Η πρόσθεση επιπλέον θέσεων μπορεί να γίνεται δυναμικά με αποτέλεσμα να αυξάνεται η μεταβλητή. Υπάρχει, επίσης, η δυνατότητα μείωσης, αλλά αυτό απλά είναι χάσιμο χρόνου. Στις μεταβλητές τύπου πίνακα προηγείται το σύμβολο @. Όταν αναφερόμαστε σε μία μόνο θέση, μπορούμε να χρησιμοποιήσουμε το $. Μπορούμε να δηλώσουμε όσες θέσεις θέλουμε με τη μία εντολή γράφοντας:

```
@comp = ("1","2","one","hello world","learning Perl");
```

Για να αναφερθούμε σε κάθε θέση ξεχωριστά, χρησιμοποιούμε την κλήση με αριθμό (call by number): π.χ. στο παραπάνω παράδειγμα η `$comp[3]` είναι ίση με `"hello world"`, και η `$comp[1]` είναι ίση με `"2"`. Όπως παρατηρούμε η Perl, όπως και η JavaScript ξεκινάει την αρίθμηση από το μηδέν. Γι' αυτό `$comp[0]` ισούται με `"1"`, `$comp[2]` με `"one"`, κτλ..

Μπορούμε να βρούμε πόσες θέσεις έχει ένας πίνακας ελέγχοντας την ενσωματωμένη μεταβλητή `$#comp`. Για παράδειγμα, ο αριθμός των θέσεων του πίνακα `@comp`, που ορίσαμε παραπάνω θα είναι 4. (Ξεκινάμε τη μέτρηση από 0, επομένως οι θέσεις είναι 5.)

Για να δηλώσουμε μία θέση τη φορά γράφουμε, π.χ.:

```
$comp[0] = "2000000";
```

Σημείωση: Τα `$comp[0], $comp[30],` ή `$comp[οτιδήποτε]` δεν έχουν καμία σχέση με το $comp από το προηγούμενο παράδειγμα. Το μόνο που μοιράζονται είναι το ίδιο όνομα, που όμως δεν δημιουργεί κάποιο πρόβλημα εφόσον οι δύο μεταβλητές είναι διαφορετικού τύπου. Καλό είναι να μη δίνουμε σε διαφορετικές μεταβλητές ίδια ονόματα..

Μεταβλητές τύπου σχετιζόμενου πίνακα

Ο τελευταίος τύπος μεταβλητής είναι ο συσχετιζόμενος πίνακας. Τέτοιοι πίνακες χρησιμοποιούνται για την αποθήκευση συσχετιζόμενης πληροφορίας. Είναι πιο εύκολοι από τους απλούς πίνακες που είδαμε προηγουμένως, γιατί δεν χρειάζεται να θυμόμαστε τη θέση στην οποία έχει αποθηκευτεί κάποια πληροφορία. Το σύμβολο % προηγείται του ονόματος ενός συσχετιζόμενου πίνακα, αλλά όπως και στους άλλους πίνακες μπορούμε και εδώ να χρησιμοποιήσουμε το σύμβολο $, για να αναφερθούμε σε μία θέση ξεχωριστά. Για τον ορισμό ενός συσχετιζόμενου πίνακα γράφουμε:

```
%comp
=("song","track","lyrics","music","artist","band","year","drink");
```

Κάθε τιμή σε αυτό το παράδειγμα είναι το κλειδί για την επόμενη τιμή. Έτσι, `$comp{'song'}` είναι `"track"` και `$comp{'music'}` είναι `"artist"`, κοκ. Για να ορίσουμε μία θέση τη φορά, γράφουμε:

```
$comp{'song'} = "Tuesday is gonne";
```

Για να επεξεργαστούν οι μεταβλητές χρησιμοποιούμε τους τελεστές της Perl.

5.2.3 Τελεστές

Οι τελεστές χρησιμοποιούνται για πράξεις ανάμεσα στις μεταβλητές. Υπάρχουν τρεις διαφορετικοί τύποι τελεστών: εκχώρησης, σύγκρισης, και μαθηματικοί.

Τελεστές εκχώρησης

Οι τελεστές εκχώρησης (Assignment operators) δίνουν μία τιμή σε μία μεταβλητή. Οι τελεστές εκχώρισης της PERL παρουσιάζονται στον πίνακα 5.2.1

Πίνακας 5.2.1 – Τελεστές εκχώρησης στην Perl

Τελεστής	Περιγραφή
=	Κάνει την τιμή της μεταβλητής στα αριστερά ίση με οτιδήποτε βρίσκεται στα δεξιά.
+=	Προσθέτει την τιμή που είναι δεξιά στην τιμή αριστερά και κάνει την μεταβλητή στα αριστερά ίση με το αποτέλεσμα.
-=	Ίδιο με το παραπάνω, μόνο που γίνεται αφαίρεση.

Τελεστές σύγκρισης

Οι τελεστές σύγκρισης (comparison operators) συγκρίνουν δύο τιμές και δίνουν μία τρίτη τιμή που εξαρτάται από το αποτέλεσμα της σύγκρισης. Οι τελεστές σύγκρισης παρουσιάζονται στον πίνακα 5.2.2.

Πίνακας 5.2.2 – Τελεστές σύγκρισης στην Perl

Τελεστής	Περιγραφή
<	Επιστρέφει την τιμή 'αληθής' (true) αν η τιμή της μεταβλητής στα αριστερά είναι μικρότερη από αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή 'ψευδής' (false).
>	Επιστρέφει την τιμή 'αληθής' (true) αν η τιμή της μεταβλητής στα αριστερά είναι μεγαλύτερη από αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή 'ψευδής' (false).
>=	Επιστρέφει την τιμή 'αληθής' (true) αν η τιμή της μεταβλητής στα αριστερά είναι μεγαλύτερη ή ίση από αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή 'ψευδής' (false).
<=	Επιστρέφει την τιμή 'αληθής' (true) αν η τιμή της μεταβλητής στα αριστερά είναι μικρότερη ή ίση από αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή 'ψευδής' (false).
==	Επιστρέφει την τιμή 'αληθής' (true) αν οι τιμές και στις δύο πλευρές είναι ίσες; Αλλιώς επιστρέφει την τιμή 'ψευδής' (false).
eq	Το ίδιο με το παραπάνω, μόνο που συγκρίνει αλφαριθμητικά/κείμενο.
!=	Επιστρέφει την τιμή 'αληθής' (true) αν η τιμή της μεταβλητής στα δεξιά δεν είναι ίση με την τιμή στα αριστερά. Αλλιώς επιστρέφει την τιμή 'ψευδής' (false) αν είναι ίσες.
ne	Το ίδιο με την παραπάνω.

Οι τελεστές σύγκρισης είναι ιδανικοί για τις δηλώσεις if/else που είδαμε πιο πριν. Παρακάτω παρουσιάζεται ένα απλό παράδειγμα που συνδυάζει την χρήση των δηλώσεων if/else και των τελεστών.

```
#Τοποθέτηση της τιμής 5 στην μεταβλητή $comp.
$comp = 5;
# Σύγκριση της μεταβλητής $comp και αντίδραση σύμφωνα με το
```

```
#  αποτέλεσμα.
if ($comp < 5) {
       print "Good!";
} elsif ($comp >= 6) {
       print "Just ok";
} else {
       print "perfect!";
}
```

Μαθηματικοί τελεστές

Οι μαθηματικοί τελεστές (mathematical operators) εκτελούν μαθηματικές πράξεις, όπως φαίνεται στον πίνακα 5.2.3:

Πίνακας 5.2.3 – Μαθηματικοί τελεστές στην Perl

Τελεστής	Περιγραφή
*	Πολλαπλασιάζει δύο τιμές.
/	Διαιρεί δύο τιμές.
+	Προσθέτει δύο τιμές.
-	Αφαιρεί δύο τιμές.
++	Προσθέτει 1 στην τιμή στα αριστερά (έτσι αν $i είναι ίσο με 1, $i++ θα γίνει ίσο με 2).
--	Αφαιρεί 1 από την τιμή στα αριστερά.

Με εξαίρεση τους δύο τελευταίους μαθηματικούς τελεστές, οι υπόλοιποι δεν μπορεί να χρησιμοποιηθούν αν δεν χρησιμοποιήσουμε έναν τελεστή εκχώρησης για την αποθήκευση της τιμής. Έτσι θα πρέπει να γράψουμε για παράδειγμα:

```
$comp = 5 * 3;
```

5.2.4 Εντολές διακλάδωσης στην Perl

Όταν θέλουμε ο κώδικάς που γράφουμε να κάνει διάφορες ενέργειες, οι οποίες εξαρτώνται από συγκεκριμένες συνθήκες, χρησιμοποιούμε τις εντολές διακλάδωσης. Στην Perl μια απλή εντολή διακλάδωσης είναι η `if/elseif/else`. Παρακάτω παρουσιάζεται μια εντολή τέτοιου είδους.

```
if (statement) {
       task
} elsif (statement) {
       a different task
} else {
       a task if all else fails
}
```

5.2.5 Βρόχοι επανάληψης

Με την χρήση των βρόχων επανάληψης μπορούμε να επαναλάβουμε κάποιες γραμμές κώδικα, χωρίς να χρειάζεται να τους ξαναγράφουμε. Οι βρόχοι επαναλαμβάνονται και σταματούν μόνο όταν ικανοποιούνται κάποιοι παράμετροι.

Όλοι οι βρόχοι πρέπει να περιέχουν κάποια εντολή τερματισμού. Για σωστό και εύκολα συντηρήσιμο πρόγραμμα τύπων πρέπει να μπορούμε να παρακολουθούμε τις μεταβλητές μας σε κάθε βήμα του βρόχου.

Η γενική μορφή ενός βρόχου είναι η παρακάτω:

```
εντολή (ελέγχου) {

        δηλώσεις;
}
```

Οι βασικοί τύποι βρόχων στην Perl είναι ο βρόχος `for`, ο βρόχος `while`, και ο βρόχος `foreach`.

Ο βρόχος For

Ο βρόχος `for` είναι στη μορφή του ο πιο πολύπλοκος όλων των τύπων των βρόχων. Η πολυπλοκότητα στον προγραμματισμό σημαίνει και περισσότερες δυνατότητες. Η μορφή ενός βρόχου `for` είναι για παράδειγμα:

```
for ($i = 0; $i <= $#comp; $i++) {
       print "$comp[$i]";
}
```

Στο βρόχο `for` ορίζουμε μία μεταβλητή που υπάρχει μόνο εντός του βρόχου. Δεν χρειάζεται να αναφέρεται το `$i`, αλλά είναι μία έτοιμη μεταβλητή, που χρησιμοποιείται όταν οι τιμές αυξάνονται διαδοχικά. Στο παραπάνω παράδειγμα, ορίσαμε τη μεταβλητή `$i` ίση με `0`. Στη συνέχεια είπαμε στο βρόχο να επαναληφθεί, όσο το `$i` είναι μικρότερο ή ίσο με τον αριθμό των θέσεων στον πίνακα `@comp`. Μετά τίθεται το `1` στο `$i` στο τέλος κάθε επανάληψης του βρόχου. Το πρόγραμμα εκτυπώνει τη θέση του πίνακα `@comp` που αντιπροσωπεύει το `$i`. Έτσι, στην πρώτη επανάληψη του βρόχου θα εκτυπωθεί `$comp[0]`. Στη δεύτερη, θα εκτυπωθεί `$comp[1]`. Ο βρόχος θα σταματήσει όταν η συνθήκη που ορίστηκε δε θα ισχύει πια.

Ο βρόχος While

Ο βρόχος `while` είναι παρόμοιος με τον `for`, μόνο που δεν είναι τόσο ανεξάρτητος. Ένα παράδειγμα ενός τέτοιου βρόχου είναι:

```
While ($comp ne "intel") {
       print "$comp";
       $comp = <NAMES>;
}
```

Αυτός ο κώδικας εκτελεί τις εντολές εντός του βρόχου όσο η μεταβλητή `$comp` δεν ισούται με "intel". Ο συγκεκριμένος αυτός τύπος βρόχου είναι λίγο επικίνδυνος από την άποψη ότι μπορεί να «πέσει» σε ατέρμονο loop, αν η μεταβλητή `$comp` δεν αλλάζει την τιμή της κάπου αλλού μέσα στο πρόγραμμα.

Ο βρόχος Foreach

Οι βρόχοι `Foreach` είναι μία πιο χαλαρή έκδοση των βρόχων `for`. Π.χ.:

```
foreach $slotnum (@comp) {
       print "$slotnum";
}
```

Ο βρόχος θα εκτελεστεί τόσες φορές όσες είναι οι θέσεις του πίνακα `comp`. Σε κάθε επανάληψη ο βρόχος θα πάρει τη τιμή και θα την εκχωρήσει στο `$slotnum` για να χρησιμοποιηθεί αργότερα. Έτσι ο πίνακας `@comp` θα ξεκινήσει με τη θέση 0 και θα φτάσει στην 100 (αν υπάρχει) ή θα σταματήσει αν δεν υπάρχουν διαθέσιμες θέσεις. Ο βρόχος `foreach` είναι χρήσιμος στην προσπέλαση συσχετιζόμενων πινάκων μιας και οι θέσεις σε τέτοιους πίνακες δεν είναι αριθμημένες.

```
foreach $slotname (keys (%comp)) {
       print "$comp{$slotname}";
}
```

Ο κώδικας παίρνει κάθε τιμή από τον πίνακα `%comp`, αυτό γίνεται με τη χρήση της ενσωματωμένης συνάρτησης `keys` της Perl και στη συνέχεια εκτυπώνει την τιμή αυτή.

5.2.6 Εισαγωγή δεδομένων

Για να είναι χρήσιμο ένα πρόγραμμα θα πρέπει με κάποιο τρόπο να έχει πρόσβαση σε κάποια δεδομένα εισόδου και ταυτόχρονα να μπορεί να εμφανίζει κάποια αποτελέσματα. Για να μπορέσουμε να διαχειριστούμε επιδέξια όσα μάθαμε ως τώρα, θα δούμε την είσοδο-έξοδο (input-output) δεδομένων. Υπάρχουν δύο μεγάλες κατηγορίες εισόδου και εξόδου: είσοδος-έξοδος από τον χρήστη και είσοδος-έξοδος από κάποιο αρχείο.

Στο Unix τα πάντα, ακόμα και οι συσκευές εισόδου / εξόδου (π.χ. οθόνη, εκτυπωτής κτλ.) θεωρούνται αρχεία και η Perl είναι μία γλώσσα προγραμματισμού προσαρμοσμένη σε αυτό το λειτουργικό σύστημα. Για να διαβάσει ένα πρόγραμμα κάποια δεδομένα που βρίσκονται σε ένα αρχείο θα πρέπει πρώτα να το ανοίξει, να διαβάσει τα δεδομένα και μετά να ξανακλείσει το αρχείο. Το αρχείο αναγνωρίζεται εσωτερικά από το πρόγραμμα ως κάποιο αντικείμενο που ονομάζεται «file handler». Κάθε αρχείο έχει το δικό του file handler. Έτσι για να εκτυπωθεί κάτι στην οθόνη, αφού αποτελεί αρχείο για το UNIX, θα έπρεπε τα προγράμματα να ανοίγουν το αρχείο που αντιπροσωπεύει την οθόνη, να γράφουν και να το ξανακλείνουν. Επειδή όμως η οθόνη αποτελεί την προκαθορισμένη έξοδο και είσοδο πολλών προγραμμάτων, το UNIX, θεωρεί ότι το αρχείο της οθόνης είναι πάντα ανοικτό και δεν θεωρεί απαραίτητες τις διαδικασίες (εντολές) ανοίγματος και κλεισίματος της οθόνης.

Επομένως όταν λέμε ότι έχουμε είσοδο-έξοδο δεδομένων από τον χρήστη εννοούμε ότι το πρόγραμμα διαβάζει από το αρχείο που αντιπροσωπεύει την οθόνη. Όταν έχουμε και είσοδο-έξοδο δεδομένων από/σε κάποιο αρχείο (file input-output) τότε θα πρέπει να γράψουμε ειδικές εντολές που θα ανοίγουν, διαβάζουν/ γράφουν και μετά θα κλείνουν το αρχείο.

Το Standard input είναι: `$guess = <STDIN>;`

Η παραπάνω γραμμή θα εμφανιζόταν στο χρήστη σαν μια προτροπή ώστε να εισάγει τα δεδομένα του και ο,τιδήποτε εισάγει ο χρήστης θα αποθηκευτεί στη μεταβλητή `$usersaid`. Όπως έχουμε αναφέρει παραπάνω, ένα αρχείο αναγνωρίζεται εσωτερικά από το πρόγραμμα ως κάποιο αντικείμενο που ονομάζεται «file handler». Τα ονόματα που δίνουμε σε αυτά τα αντικείμενα είναι γραμμένα με κεφαλαία γράμματα. Τα «file handler» αποτελούν μια αναφορά/ παραπομπή σε κάποιο σενάριο ενός ανοιχτού αρχείου. Τα αρχεία που παριστούν την προκαθορισμένη είσοδο και έξοδο είναι πάντα ανοιχτά, έτσι ώστε να μη χρειάζεται να

τα ανοίγουμε και να τα κλείνουμε σαν κανονικά αρχεία. Για παράδειγμα η εντολή `$stuff = <FILE_HADLER>;` διαβάζει μία γραμμή εισόδου και την αποθηκεύει στη μεταβλητή `$stuff` από το «file handler» με όνομα FILE_HADLER.

Η εντολή print είναι η καθιερωμένη μέθοδος εξόδου δεδομένων. Για να εκτυπώσουμε την είσοδο/ τα δεδομένα στην οθόνη θα χρησιμοποιήσουμε:

```
print "$usersaid";
```

Με όλα τα παραπάνω μπορούμε να φτιάξουμε ένα μικρό παιχνίδι:

```
#!/usr/local/bin/perl
print "\nPlease enter a number: ";
$guess = <STDIN>;
while ($guess != 20) {
print "\nSorry! You guessed wrong! Please, guess again!";
$guess = <STDIN>;
}

print "Congratulations! You got it!";
```

Μπορούμε να έχουμε είσοδο ή έξοδο από αρχείο με μερικά βήματα ακόμα. Πρώτα ανοίγουμε το αρχείο:

```
open (NAMES, "/usr/people/ferm/names.txt");
```

Το "File handler" στην συγκεκριμένη περίπτωση είναι το NAMES.
Όταν έχουμε ανοίξει το αρχείο, μπορούμε να διαβάσουμε μία γραμμή του γράφοντας:

```
$fileinput = <NAMES>;
```

και αποθηκεύοντας την σε μία βαθμωτή μεταβλητή ή να χρησιμοποιήσουμε πίνακα όπου κάθε θέση είναι μία ξεχωριστή γραμμή:

```
@fileinput = <NAMES>;
```

Εφόσον έχουμε τα δεδομένα, πρέπει να βεβαιωθούμε ότι κλείσαμε το file handler στην περίπτωση που θέλουμε να γράψουμε πάλι κάτι σε αυτό.

***Σημείωση*:** Ορίζεται ρητά ότι δεν μπορούμε να διαβάζουμε και να γράφουμε ταυτόχρονα στο ίδιο αρχείο.

Κλείνουμε ένα αρχείο γράφοντας:

```
close (NAMES);
```

Υπάρχουν δύο διαφορετικοί μέθοδοι για να γράψουμε σε αρχείο: εγγραφή (writing) και προσάρτηση (appending). Κατά την εγγραφή διαγράφουμε το περιεχόμενο του αρχείου και ξαναγράφουμε το αρχείο από την αρχή. Αν το αρχείο δεν υπάρχει, δημιουργείται το νέο αρχείο. Κατά την προσάρτηση προσθέτουμε νέα πληροφορία στο τέλος του υπάρχοντος αρχείου.

Άνοιγμα αρχείου για εγγραφή σε αρχείο:

```
Open (OUTFILE, ">/usr/people/perllog.txt");
```

Άνοιγμα αρχείου για προσάρτηση:

```
Open (OUTFILE, ">>/usr/people/perllog.txt");
```

Αφού αποφασίσουμε τον τρόπο εγγραφής, ας δούμε πώς μπορούμε να εκτυπώσουμε/ γράψουμε σε ένα αρχείο. Ακολουθείται η ίδια διαδικασία όπως κατά την εκτύπωση στην καθιερωμένη έξοδο, με την εξαίρεση ότι καθορίζουμε ακριβώς που θέλουμε να εκτυπωθούν τα δεδομένα μας, θέτοντας το «file handler» μεταξύ της εκτύπωσης και της πρότασης που θέλουμε να γραφτεί στο αρχείο:

```
print OUTFILE "Here's a line of output";
```

5.2.7 Η συνάρτηση splitting

Η συνάρτηση Splitting ψάχνει για ένα δείγμα σε ένα αλφαριθμητικό και σπάει το αλφαριθμητικό σε έναν πίνακα βασισμένη στο δείγμα. Η χρήση της συνάρτησης γίνεται σύμφωνα με τον τύπο: split(/έκφραση_διαχωρισμού/, έκφραση_προς_διαχωρισμό).

Στο παρακάτω παράδειγμα η εντολή `@stuff = split (/ /, $stuff);` θα διασπάσει τα περιεχόμενα της μεταβλητής `$stuff`, χρησιμοποιώντας ως διαχωριστικό το κενό « ». Το αποτέλεσμα της split στο παρακάτω παράδειγμα είναι ότι ο πίνακας @stuff θα έχει τα εξής περιεχόμενα: @stuff=(“Learning”,”Perl”,”fast”).

```
#!/usr/local/bin/perl
# Αλφαριθμητική μεταβλητή (string).

$stuff = "Learning Perl fast";

# Διάσπαση του αλφαριθμητικού σε πίνακα με διαχωριστικό το κενό.
@stuff = split (/ /, $stuff);

# Εκτύπωση κάθε θέσης του πίνακα σε διαφορετική γραμμή.
for ($i = 0; $i <= $#stuff; $i++) {
       print "Slot $i: $stuff[$i]\n";
}

# Εκτύπωση του αριθμού των λέξεων του αλφαριθμητικού.
print "There were $#stuff words.\n";
```

5.2.8 Μετατρέποντας την Perl σε CGI

Η συγγραφή προγραμμάτων CGI προϋποθέτει την γνώση κατασκευής φορμών σε HTML.

Στο κεφάλαιο 3 μιλήσαμε για τις φόρμες εισαγωγής δεδομένων και την ετικέτα <FORM> της HTML. Υπάρχουν δύο μέθοδοι για να στείλουμε δεδομένα από τον φυλλομετρητή μας στον αντίστοιχο εξυπηρετητή. Στην παράμετρο <METHOD> της ετικέτας <FORM> μπορούμε να ορίσουμε την μέθοδο που θα χρησιμοποιήσουμε, η οποία μπορεί να είναι είτε η GET είτε η POST, όπως για παράδειγμα φαίνεται παρακάτω.

```
<FORM METHOD="POST" ACTION="/cgi-bin/myscript.pl">
```

Στην παράμετρο ACTION τοποθετούμε τη διαδρομή για το πρόγραμμα CGI που θα επεξεργαστεί τα δεδομένα που θα στείλουμε. Στην παράμετρο METHOD επιλέγουμε την μέθοδο (POST ή GET). Θεωρείται ως προεπιλογή η GET. Αν επιλέξουμε να χρησιμοποιήσουμε την GET τότε όταν στείλουμε την φόρμα θα εμφανιστούν/ ενσωματωθούν τα δεδομένα μας στην διεύθυνση URL που χρησιμοποιούμε. Συγκεκριμένα, έστω ότι κάνουμε μία αναζήτηση στην ιστοσελίδα με διεύθυνση: http://leykada.physics.auth.gr/ όταν στείλουμε τα δεδομένα στην γραμμή που εμφανίζεται η διεύθυνση (URL) θα υπάρχει κάτι σαν το παρακάτω. Αν όμως επιλέξουμε την POST τα δεδομένα θα παραμείνουν κρυμμένα για τον απλό χρήστη του φυλλομετρητή.

```
http://leykada.physics.auth.gr/cgi-bin/heal-linksearch/heal-linksearch.cgi?
SearchTerm1=network&Elsevier=NO&Springer=NO&Wilson=NO&MCB=NO&
Kluwer=NO&IOP=NO&ACS=NO&Oxford=NO&Wiley=YES&ACM=NO&
Blackwell=NO&Taylor=NO&Lippincott=NO&Boolean=AND&
SearchTerm2=&Search=Search
```

Άλλη διαφορά ανάμεσα στις δύο αυτές μεθόδους αποτελεί το γεγονός ότι με την GET μπορούμε να στείλουμε δεδομένα μήκους 1024 χαρακτήρων και μόνο. Επίσης αν σε μία ιστοσελίδα έχουμε χρησιμοποιήσει την POST τότε η επιλογή του φυλλομετρητή μας «επιστροφή στην προηγούμενη σελίδα» δεν θα λειτουργήσει όπως θα περιμέναμε και αυτό διότι τα δεδομένα που είχαμε εισάγει θα έχουν χαθεί. Δεν ισχύει το ίδιο με την μέθοδο GET.

Τα CGI επεξεργάζονται δεδομένα εισόδου με διαφορετικό τρόπο από την Perl και αυτή είναι η μόνη διαφορά που μπορεί κανείς να εντοπίσει. Μόλις το πρόγραμμα CGI επεξεργαστεί τα δεδομένα εισόδου, τα μετατρέπει σε μεταβλητές που τις μεταχειρίζονται με τον ίδιο τρόπο από τα CGI και τα σενάρια Perl. Η είσοδος στα CGI μπορεί να ανακτηθεί με δύο τρόπους: "get" και "post". Αν είμαστε απόλυτα σίγουροι για τον τύπο της εισόδου που θα πάρουμε, τότε χρειάζεται να χρησιμοποιήσουμε μόνο ένα τύπο ανάκτησης εισόδου. Για να καλύψουμε όμως όλες τις πιθανές περιπτώσεις, γράφουμε το εξής:

```
if ($ENV{'REQUEST_METHOD'} eq "GET") {

                $in = $ENV{'QUERY_STRING'};
       } else {
                $in = <STDIN>;
      }
```

Αυτή η εντολή απευθύνει μία ερώτηση στον εξυπηρετητή, αν η μέθοδος (ο τρόπος με τον οποίο ο εξυπηρετητής δίνει την πληροφορία) είναι η "get". Τότε το πρόγραμμα θα διαβάσει την πληροφορία από το Query String και θα την καταχωρίσει σε μία μεταβλητή που ονομάζεται $in. Αλλιώς, θα την διαβάσει μέσω της καθορισμένης εισόδου όπως η Perl.

5.2.9 Ένα πρόγραμμα CGI

Θα γράψουμε τώρα ένα μικρό πρόγραμμα για την αναζήτηση τραγουδιών, που μπορούμε να προσθέσουμε στην προσωπική μας σελίδα.

Πριν ξεκινήσουμε να χτίζουμε το CGI, θα πρέπει να δώσουμε ένα όνομα στην είσοδό μας. Έχουμε μία φόρμα που δέχεται μόνο ένα είδος εισόδου: τον τίτλο του τραγουδιού. Ονομάζουμε έτσι το πεδίο της φόρμας "title".

Στο πεδίο της φόρμας μπορούμε να γράψουμε έναν τίτλο τραγουδιού και να το αναζητήσουμε. Τον τίτλο μπορούμε να γράψουμε τον κώδικα και με κεφαλαία και με πεζά.

Φτιάχνουμε την φόρμα με το παρακάτω HTML:

```
<form action="/cgi-bin/το_cgi_programma_mas" method="get">
<input type="text" length="20" name="title"><br>
<input type="submit" value="Τα τραγούδια που προτιμώ">
</form>
```

Για βάση δεδομένων ένα απλό αρχείο κειμένου, οργανωμένο έτσι ώστε να είναι εύκολα προσπελάσιμο από ένα μικρό πρόγραμμα.

Για την ανάκτηση ενός αρχείου δεδομένων όπως τα παραπάνω, πρέπει να γνωρίζουμε πώς θα είναι δομημένα τα δεδομένα.

Έστω ότι το αρχείο δεδομένων ονομάζεται music.dat. Έχουμε ορίσει και μία σειρά από κανόνες μορφοποίησης για το αρχείο δεδομένων. Για να είναι λειτουργικό το σενάριο θα πρέπει να υπάρχει αυστηρή δομή στο αρχείο αυτό. Το scripting είναι μία δραστηριότητα βασισμένη στη δομή.

Το αρχείο δεδομένων για το παράδειγμα μας, music.dat, θα πρέπει να έχει την μορφή :

```
---- τίτλος τραγουδιού με μικρά γράμματα ----

τίτλος

περιγραφή
```

Έτσι έχουμε, για παράδειγμα τα δεδομένα :

```
---- tuesday ----

Tuesday is gone

Το Tuesday is gone είναι ένα τραγούδι των Lynyrd Skynyrd.
Περιλαμβάνεται στον ομώνυμο δίσκο του συγκροτήματος. κτλ…….
```

5.2.10 Επεξεργασία των δεδομένων εισόδου

Για να ανακτήσουμε την πληροφορία που αντιστοιχεί στα δεδομένα εισόδου του χρήστη (που αντιστοιχεί στον τίτλο του τραγουδιού που θα εισάγει ο χρήστης στο πεδίο με το όνομα "title" της φόρμας), ανοίγουμε το αρχείο δεδομένων μας με τον τρόπο που θα ανοίγαμε κάθε άλλο αρχείο στην Perl. Στη συνέχεια το διαβάζουμε σε μία μεταβλητή τύπου πίνακα και κλείνουμε το αρχείο:

```
open (DATA, "/usr/people/music.dat");
my @data = <DATA>;
Close (DATA);
```

Το πρόγραμμα τώρα έχει όλη την πληροφορία από το αρχείο δεδομένων και είναι πλέον έτοιμο να ψάξει για τη συγκεκριμένη πληροφορία που ταιριάζει στην είσοδο που έχει δώσει ο χρήστης. Πρέπει να έχουν δοθεί στο χρήστη σαφείς οδηγίες για το πώς πρέπει να γράψει τον τίτλο του τραγουδιού. Όλο το πρόγραμμα μέχρι εδώ είναι:

```
# Αυτή θα είναι πάντα η πρώτη γραμμή ενός Perl προγράμματος!
#!/usr/local/bin/perl

# Άνοιγμα της βάσης/αρχείου δε δεδομένων music.dat και
αποθήκευσή # του/της στη μεταβλητή @data.
open (DATA, "/kapou_brisketai_kai_auto/music.dat");
my @data = <DATA>;
close (DATA);

# Με το παρακάτω γίνεται τοποθέτηση των δεδομένων εισόδου στη
# μεταβλητή $in, ανεξάρτητα από την μέθοδο εισαγωγής που έχει
# χρησιμοποιηθεί.

my $in;
if ($ENV{'REQUEST_METHOD'} eq "GET") {
 $in = $ENV{'QUERY_STRING'};
} else {
  $in = <STDIN>;
 }
```

Ας δούμε λίγο τα δεδομένα εισόδου του χρήστη και τις τροποποιήσεις που πρέπει να κάνουμε ώστε τα δεδομένα να περάσουν στο πρόγραμμα σωστά. Για παράδειγμα, θα πρέπει να βεβαιωθούμε ότι δεν θα έχουμε διάφορους παράξενους χαρακτήρες, οι οποίοι δημιουργούνται όταν τα δεδομένα εισόδου μεταφραστούν σε κώδικα URI. (URI είναι ο κώδικας στον οποίο μεταφράζονται οι μη αλφαριθμητικοί χαρακτήρες κατά την αποστολή μιας φόρμας και είναι η μέθοδος με την οποία ο φυλλομετρητής περνάει την πληροφορία στο CGI.). Έτσι, μπορούμε να γράψουμε:

```
$in =~ s/%(..)/pack("c",hex($1))/ge;
```

Το $s αποτελεί μία μεταβλητή της Perl που χρησιμοποιείται για την εύρεση και αντικατάσταση χαρακτήρων μέσα σε ένα σύνολο χαρακτήρων (string). Ο τελεστής ισοδυναμίας =~ (equivalent) συνδέει το κείμενο όπου έχουν αντικατασταθεί κάποιοι χαρακτήρες με το αρχικό.

Η συνάρτηση "s/πρωτότυπο_κείμενο/ κείμενο_για_αντικατάσταση/ge" ψάχνει για κανονικές εκφράσεις ανάμεσα στην πρώτη και δεύτερη κάθετο (/) και μετά τις αντικαθιστά με ο,τιδήποτε υπάρχει ανάμεσα στην δεύτερη και τρίτη κάθετο. Η εντολή pack πακετάρει το κείμενο και το μετατρέπει σε χαρακτήρες αναγνώσιμους για τον άνθρωπο. Το g, στο τέλος της συνάρτησης, δηλώνει ότι η αντικατάσταση θα γίνει σε όλες τις περίπτωσεις. Το e δηλώνει ότι το νέο κείμενο θα πρέπει να ελεγχθεί αν αποτελεί έκφραση της Perl και τότε μόνο να γίνει η αντικατάσταση. Παρακάτω θα

συναντήσουμε και το /i το οποίο δηλώνει ότι η αντικατάσταση/αναζήτηση θα γίνει λαμβάνοντας υπόψη τα κεφαλαία και πεζά γράμματα.

Επομένως, η παραπάνω εντολή αναζήτησης και αντικατάστασης ψάχνει να δει αν κάποιοι χαρακτήρες ξεκινάνε με το σύμβολο % και τους μετατρέπει σε χαρακτήρες αναγνώσιμους, χρησιμοποιώντας την δεκαεξαδική τους αναπαράσταση.

Αν ο τίτλος του τραγουδιού που θα εισάγει ο χρήστης είναι περισσότερες από μία λέξεις, η μεταβλητή $in θα περιέχει σύμβολα "+" στη θέση των κενών. Γι' αυτό το λόγο θα γράψουμε:

```
$in =~ s/\+/ /g;
```

Αυτή η εντολή αναζήτησης και αντικατάστασης της Perl ψάχνει και απαλείφει, όλα (g), τα σύμβολα "+" και τα αντικαθιστά με κενά.

Τέλος, θα χρειαστεί να μετακινήσουμε τα δεδομένα εκτός της $in και να τους δώσουμε ένα πιο περιγραφικό όνομα:

```
my %music = split (/=/, $in);
```

Αυτή η γραμμή κώδικα διασπά τα περιεχόμενα της $in χρησιμοποιώντας σαν οριοθέτη το =, σε έναν συσχετιζόμενο πίνακα με όνομα `%music`. Το my υποδηλώνει ότι η μεταβλητή έχει περιορισμένη ισχύ, είναι, δηλαδή, τύπου private. Αφού το όνομα του πεδίου της φόρμας είναι "title", τα δεδομένα θα αποθηκευτούν στη θέση `$music{'title'}`.

5.2.11 Ταίριασμα

Η εκτύπωση/εμφάνιση αποτελεσμάτων σε μία ιστοσελίδα είναι ισοδύναμη με την εκτύπωση στην προκαθορισμένη έξοδο, με την εξαίρεση ότι μπορούμε να χρησιμοποιήσουμε ετικέτες HTML. Πρέπει να πούμε στον φυλλομετρητή τι θα πάρει σαν αποτέλεσμα. Σε αυτή την περίπτωση, θα πάρει ένα αρχείο HTML. Έτσι, προσθέτουμε στην πρώτη δήλωση εκτύπωσης που θα σταλεί στο χρήστη το:

```
Content-Type: text/html\n\n
```

Ο χρήστης δε θα το δει έτσι, παρά μόνο ο φυλλομετρητής. Εκτυπώνουμε, λοιπόν, την επικεφαλίδα της σελίδας των αποτελεσμάτων μας ως εξής:

```
print <<END;

Content-Type: text/html\n\n

<html>
<body bgcolor="#000000" text="#ffffff">
<center>
<h2>Results</h2>
</center>
<p>

END
```

Προσέξτε ότι χρησιμοποιήσαμε HTML μέσα στον κώδικα CGI. Είναι σαν να γράφουμε μία κανονική σελίδα HTML σε τμήματα, τα οποία όμως περιέχονται σε εντολές εκτύπωσης.

Μέχρι το σημείο αυτό, έχουμε εκτυπώσει την επικεφαλίδα “Results”, την πληροφορία του χρήστη σε μία μεταβλητή και το αρχείο δεδομένων αποθηκευμένο σε μία μεταβλητή πίνακα. Τώρα θα πρέπει να προσπελάσουμε τον πίνακα και να προσπαθήσουμε να ταιριάξουμε τους τίτλους. Αν κάποιος τίτλος ταιριάζει στο ζητούμενο, θα εκτυπώσουμε τη σχετική πληροφορία. Αν δεν ταιριάζει θα εκτυπώσουμε ένα μήνυμα λάθους.

Για την προσπέλαση ενός πίνακα μπορούμε να χρησιμοποιήσουμε έναν βρόχο `for` ή ένα βρόχο `foreach`. Σε αυτό το παράδειγμα, θα χρησιμοποιήσουμε τον βρόχο `for`.

```
for ($i = 0; $i <= $#data; $i++) {
}
```

Αυτό διατρέχει τις θέσεις στον `@data` μέχρι να μην υπάρχουν άλλες θέσεις και σταματάει. Πρέπει να ελέγξουμε αν ο τίτλος είναι αποθηκευμένος σε μία θέση.

```
if ($data[$i] =~ /-- $music{'title'} --/i) {
}
```

Αν υπάρχει ταίριασμα, θα χρησιμοποιήσουμε τον ακόλουθο κώδικα μέσα στο `if`:

```
print <<END;
<b><font size="+1">$data[$i + 1]</font></b><br>
$data[$i + 2]<br>
END
last;
```

Με τις παραπάνω εντολές προσπαθούμε να βρούμε το σωστό τίτλο χρησιμοποιώντας τη γραμμή με παύλες που θα έχουμε βάλει στη μορφοποίηση της βάσης δεδομένων και αυτή είναι η γραμμή `$i`. Το πραγματικό περιεχόμενο που θέλουμε να εμφανίσουμε, δηλαδή ο τίτλος και οι στίχοι του τραγουδιού, βρίσκεται στις δύο γραμμές μετά τη γραμμή με τις παύλες. Θα εκτυπώσουμε αυτό το περιεχόμενο προσθέτοντας 1 ή 2 στη μεταβλητή `$i`. Θα χρησιμοποιήσουμε τη γλώσσα HTML για να έχουμε άριστη εμφάνιση. Έτσι, αν η γραμμή με τις παύλες ήταν στη γραμμή 32, το `$i` θα έπαιρνε την τιμή 32. Για να πάρουμε τον τίτλο του τραγουδιού χωρίς τη γραμμή με τις παύλες, θα προσθέσουμε 1 στο `$i` γιατί είναι στην επόμενη γραμμή: 33. Κατά συνέπεια, για να πάρουμε τους στίχους του τραγουδιού, που υπάρχουν στην τρίτη γραμμή της βάσης δεδομένων μου, στη γραμμή 34 δηλαδή, θα πρέπει να προσθέσουμε 2 στο `$i`. Σημειώστε, ότι χρησιμοποιήσαμε έναν μαθηματικό τελεστή/τελεστή μαθηματικών πράξεων και όχι έναν τελεστή εκχώρησης. Με αυτόν τον τρόπο, η τιμή του `$i` θα παραμείνει 32. Μόλις εκτυπώσει την πληροφορία, χρησιμοποιεί την εντολή "last" για να πει στο πρόγραμμα ότι πήρε ότι χρειάζεται. Στην ουσία το "last" λέει στον βρόχο ότι η επανάληψη αυτή είναι η τελευταία και θα πρέπει να σταματήσει.

Για να αντιμετωπίσουμε την περίπτωση που ο χρήστης δώσει έναν τίτλο που δεν υπάρχει στη βάση δεδομένων μας, θα πρέπει να προσθέσουμε μία ακόμα εντολή. Μία δήλωση "`elseif`" στο παραπάνω τμήμα εντολών `if` δίνει μία λύση στο

πρόβλημα, διότι θα λειτουργήσει μόνο όταν το πρόγραμμα είναι στην τελευταία του επανάληψη και δεν έχει βρει κάτι που να ταιριάζει.

```
if ($data[$i] =~ /-- $music{'title'} --/i) {
      print <<END;
      <b><font size="+1">$data[$i + 1]</font></b><br>
      $data[$i + 2]<br>

      END
      last;
} elsif ($i == $#data) {

}
```

Αν το `$i` ισούται με τον αριθμό των θέσεων στον `@data` και ο τίτλος δεν υπάρχει σε καμία θέση, τότε μπορούμε να χρησιμοποιήσουμε την παρακάτω:

```
Print "<b><I>Sorry! The lyrics are missing!</i></b>";
```

Όλος ο κώδικας φαίνεται παρακάτω:

```
for ($i = 0; $i <= $#data; $i++) {
        if ($data[$i] =~ /-- $music{'title'} --/i) {

              print <<END;
              <b><font size="+1">$data[$i + 1]</font></b><br>
                     $data[$i + 2]<br>

                END
                last;
       } elsif ($i == $#data) {
           print    "<b><i>    Sorry!    The    lyrics    are
missing!</i></b>";
        }
  }
```

Ολοκληρώνουμε το πρόγραμμα γράφοντας:

```
Print <<END;
<center><hr size=1 width=50%></center>
</body>
</html>

END
```

5.2.12 Ανακεφαλαίωση

Είδαμε ένα απλό πρόγραμμα CGI, ας το ονομάσουμε my_first_cgi. Το πρόγραμμα αυτό το έχουμε «ανεβάσει» χρησιμοποιώντας το πρωτόκολλο ftp, στο φάκελο `/cgi-bin` του εξυπηρετητή HTTPD, στον οποίο διαθέτουμε τα αντίστοιχα δικαιώματα.

Παρακάτω παρουσιάζεται ο κώδικας του παραδείγματος μας, ολοκληρωμένα.

```
#!/usr/local/bin/perl

#-----------------------------------------------------------------

# Ο παρακάτω τρόπος θα διαβάσει τα δεδομένα ανεξάρτητα από το αν
# η μέθοδος είναι η "get" ή η "post," και θα τα αποθηκεύσει στην
# $in.
#-----------------------------------------------------------------

my $in;
if ($ENV{'REQUEST_METHOD'} eq "GET") {
       $in = $ENV{'QUERY_STRING'};
} else {
       $in = <STDIN>;
}

#-----------------------------------------------------------------
# Η πρώτη εντολή αντικαθιστά τον χαρακτήρα «+» με το κενό.
# η δεύτερη εντολή μετατρέπει τους χαρακτήρες που περιέχονται
# στην URI σε μη-αλφαριθμητικούς
# Η επόμενη εντολή αποθηκεύει την είσοδο στον %music
#-----------------------------------------------------------------

$in =~ s/\+/ /g;
$in =~ s/%(..)/pack("c",hex($1))/ge;
my %music = split (/=/, $in);

#-----------------------------------------------------------------
# Αυτή η εντολή διαβάζει/τοποθετεί την βάση δεδομένων στον πίνακα
# @data και κλείνει το αρχείο.
#-----------------------------------------------------------------
open (DATA, "../htdocs/music.dat");
my @data = <DATA>;
close (DATA);

#-----------------------------------------------------------------
# Εκτυπώνονται τα γενικά του εγγράφου HTML
#-----------------------------------------------------------------
print <<END;
Content-Type: text/html\n\n

<html>
<body bgcolor="#000000" text="#ffffff">
<center><h2>Results</h2></center><p>
<center><hr size=1 width=50%></center>
END
#-----------------------------------------------------------------
# Ο παρακάτω βρόχος διατρέχει τον πίνακα ψάχνοντας και
# εκτυπώνοντας τον τίτλο, περιγραφή κτλ.
#-----------------------------------------------------------------
for ($i = 0; $i <= $#data; $i++) {
       if ($data[$i] =~ /-- $music{'title'} --/i) {
              print <<END;
<center><b><font size="+1">$data[$i + 1]</font></b></center><br>
$data[$i + 2]<br>
```

```
END
                last;
        } elsif ($i == $#data) {
                print  "<b><i>Sorry!  I  haven't  reviewed  that  one
yet!</i></b>";
        }
}
#-----------------------------------------------------------------
# Εκτυπώνει τα τελικά για την ολοκλήρωση του εγγράφου HTML.
#-----------------------------------------------------------------
print <<END;
<center><hr size=1 width=50%></center>
</body>
</html>
END
```

Στον φάκελο /htdocs/ τοποθετούμε το αρχείο music.dat με τα δεδομένα μας, στην προκειμένη περίπτωση την συλλογή με τα τραγούδια, που έχει την δομή:

```
---- τίτλος τραγουδιού με μικρά γράμματα ----
τίτλος

περιγραφή
```

Έχουμε κατασκευάσει την παρακάτω φόρμα, με την οποία θα ζητήσουμε από το πρόγραμμα CGI να αναζητήσει τα δεδομένα από ένα αρχείο και να επιστρέψει τα αποτελέσματα.

```
<form action="/cgi-bin/το_cgi_programma_mas" method="get">
<input type="text" length="20" name="title"><br>
<input type="submit" value="Τραγούδια που προτιμώ">

</form>
```

Περισσότερες πληροφορίες για τη γλώσσα Perl μπορείτε να βρείτε στην διεύθυνση www.perl.com

5.3 PHP

Η PHP είναι μία ισχυρή γλώσσα για τη συγγραφή σεναρίων που προσαρμόζεται πολύ καλά στην HTML. Αποτελεί, δε ένα πολύ καλό εργαλείο για το σχεδιασμό δυναμικών ιστοσελίδων.

Ένα από τα σημαντικότερα χαρακτηριστικά της PHP είναι ότι είναι μία γλώσσα ανοιχτού κώδικα και διανέμεται ελεύθερα, που σημαίνει ότι κάθε υλικό αναφοράς της μπορεί να εντοπιστεί εύκολα και να χρησιμοποιηθεί από τον καθένα. Επίσης μπορεί να χρησιμοποιηθεί με ευκολία σε διάφορα λειτουργικά συστήματα.

Για τον έλεγχο των προγραμμάτων PHP απαιτείται ένας εξυπηρετητής που υποστηρίζει τη γλώσσα.

5.3.1 Τι ακριβώς είναι η PHP

Η PHP είναι ένα πρόγραμμα που μπορεί να εγκατασταθεί σαν ενσωματωμένο επιπλέον λογισμικό (module) σε έναν εξυπηρετητή ιστού (Web server), είτε σαν αυτόνομος εκτελέσιμος κώδικας. Προτιμάται η πρώτη περίπτωση, αφού έτσι είναι γρηγορότερη η εκτέλεση και δεν επιβαρύνεται ο εξυπηρετητής με επιπλέον απαιτήσεις σε μνήμη και χώρο στον δίσκο. Χρησιμοποιείται με εκδόσεις του Apache, Microsoft IIS, Netscape Enterprise Server και άλλων πακέτων.

Η σύνταξη της γλώσσας PHP είναι παρόμοια με αυτήν της PERL και της C. Στην πραγματικότητα η PHP ξεκίνησε σαν μία εύκολη Perl και γράφτηκε τα τέλη του 1994 από τον Rasmus Lerdorf. Τα επόμενα χρόνια εξελίχθηκε στην PHP/FI 2.0. και PHP/FI. Το καλοκαίρι του 1997 οι Zeev Suraski και Andi Gutmans έφτιαξαν ένα καινούριο επεξεργαστή, οποίος οδήγησε στην PHP 3.0. Η έκδοση PHP 3.0. όρισε την σύνταξη για τις εκδόσεις 3 και 4.

Όταν κάποιος πελάτης (χρήστης) επισκεφτεί την ιστοσελίδα που περιέχει κώδικα PHP, ο εξυπηρετητής εκτελεί τον κώδικα και αυτό που επιστρέφεται στον χρήστη είναι μία ολοκληρωμένα σχεδιασμένη ιστοσελίδα HTML. Όλη τη δουλειά την κάνει ο εξυπηρετητής και όχι κάποιος φυλλομετρητής. Ο πελάτης (χρήστης) από την πλευρά του δε χρειάζεται κάποιο πρόσθετο εργαλείο ή πρόγραμμα για να δει τη λειτουργία της PHP — φτάνει στον χρήστη κανονικά σαν κώδικας HTML.

Η PHP είναι μία γλώσσα για τη συγγραφή σεναρίων. Αυτό σημαίνει πως ο κώδικας, όπως και της HTML, δεν χρειάζεται να μεταγλωττιστεί πριν χρησιμοποιηθεί.

Σημείωση: Επισκεφτείτε τη σελίδα www.PHP.net Είναι ένα κέντρο ελέγχου, με πολύ υλικό αναφοράς για τη γλώσσα και συμβουλές από χρήστες που χρησιμοποιούν και ασχολούνται με την PHP παγκόσμια.

Η PHP μπορεί να :

- Πάρει πληροφορία από φόρμες και να τις χρησιμοποιήσει με διάφορους τρόπους : να τις αποθηκεύσει σε μία βάση δεδομένων, να δημιουργήσει εξαρτημένες/υποθετικές σελίδες που εξαρτώνται από αυτά που λέει η φόρμα, να τοποθετήσει cookies στο φυλλομετρητή του χρήστη, να στείλει e-mail.
- Πιστοποιήσει την αυθεντικότητα και να εντοπίσει χρήστες.
- Εκτελέσει παράλληλες συζητήσεις στην ιστοσελίδα που κατασκευάζουμε.
- Εξυπηρετήσει διαφορετικές ιστοσελίδες για άτομα που χρησιμοποιούν διαφορετικούς φυλλομετρητές ή συσκευές.

- Δημοσιεύσει μία ολόκληρη ιστοσελίδα χρησιμοποιώντας μόνο μία φόρμα σχεδίου (server-side includes-style).
- Εξυπηρετήσει σελίδες XML.

Πριν δούμε τις χρήσεις της PHP, θα χρειαστεί να ρίξουμε μία γρήγορη ματιά στα δομικά τμήματά της, ξεκινώντας από ένα παράδειγμα. Το παρακάτω πρόγραμμα έχει το όνομα "test.php". Όταν καλείται από έναν φυλλομετρητή, ο χρήστης απλά θα διάβαζε "Αυτό είναι ένα παράδειγμα!"

```
<?php
print ("Αυτό είναι ένα παράδειγμα!");
?>
```

Όλα τα σενάρια PHP περικλείονται στις εντολές: <?php και ?>, που είναι οι ετικέτες αρχής και τέλους του προγράμματος PHP. Το παραπάνω πρόγραμμα χρησιμοποιεί την εντολή print για να εκτυπώσει την φράση "This is a test" στην οθόνη του χρήστη.

5.3.2 Βασικοί κανόνες κατασκευής PHP εφαρμογών

Ο παραπάνω κώδικας γράφεται εντός της σελίδας HTML με το όνομα αρχείο1.php, ως εξής:

```
<html>
<head><title> Παράδειγμα </title></head>
<body>
<font color="red">Ο PHP κώδικας δημιουργεί μια σελίδα που
λέει:</font>
<p>
  <?php
  print ("Αυτό είναι ένα παράδειγμα!");
  ?>
</body>
</html>
```

Η HTML διαχειρίζεται το παραπάνω κώδικα ως απλή HTML, αλλά ο τμήματα που βρίσκεται μέσα στα <?php και ?> θα τα επεξεργαστεί ως PHP.

Οι βασικοί κανόνες συγγραφής της PHP είναι οι ακόλουθοι:

Ονομασία αρχείων

Για να λειτουργήσει ένα πρόγραμμα σε PHP, θα πρέπει το αρχείο στο οποίο είναι ο κώδικας ή οποιοδήποτε αρχείο καλεί μέσα στον κώδικα να έχει την κατάληξη .php (παλιότερες εκδόσεις χρησιμοποιούσαν τις επεκτάσεις .php3 και .phtml). Όπως στην HTML, τα αρχεία αποθηκεύονται ως απλό κείμενο.

Σχόλια

Στην PHP κάθε γραμμή σχολίου ξεκινάει με "//". Αν θέλουμε να γράψουμε ένα μπλοκ σχολίων ή να βγάλουμε προσωρινά εκτός επεξεργασίας κάποιο κομμάτι κώδικα μπορούμε να το βάλουμε εντός των συμβόλων "/*" και "*/":

```
<?php
// Αυτές οι γραμμές θα αγνοηθούν.
// Αποτελούν σχόλια
```

```
print ("Αυτό είναι ένα παράδειγμα!");
/*
και αυτές οι γραμμές θα αγνοηθούν. Αποτελούν ένα τμήμα
σχολίων.
*/
?>
```

Αρχή και τέλος κώδικα

Κάθε τμήμα κώδικα PHP ξεκινάει με "<?php" (ή σύντομα "<?" αν το δέχεται ο εξυπηρετητής).

Τελειώνουμε τον κώδικα PHP γράφοντας "?>" στο τέλος του.

Διαχωρισμός εντολών

Με λίγες εξαιρέσεις, κάθε ξεχωριστή εντολή που γράφουμε τελειώνει με το σύμβολο ";".

Παρενθέσεις

Η τυπική συνάρτηση φαίνεται ως εξής:

```
Print ( );
```

Η "print" είναι η συνάρτηση και αυτά που η συνάρτηση επεξεργάζεται βρίσκονται μέσα τις παρενθέσεις. Η "print" αποτελεί εξαίρεση καθώς λειτουργεί και χωρίς τις παρενθέσεις. Στη θέση του print () μπορούμε να χρησιμοποιήσουμε και το echo ().

Όπως και την HTML, η πραγματική μορφοποίηση του κώδικα σε PHP (όπου βάζουμε κενά, αλλαγή γραμμής, κτλ..) δεν επηρεάζει το αποτέλεσμα, εκτός από εκείνα τα κομμάτια του κώδικα που λένε στο φυλλομετρητή πώς θα εμφανίσει την ιστοσελίδα.

Έτσι ο κώδικας:

```
<?php
      print ("Αυτό είναι ένα παράδειγμα!");
?>
```

είναι ο ίδιος με τον παρακάτω:

```
<?php print ("Αυτό είναι ένα παράδειγμα!"); ?>
```

5.3.3 Ένα πρόγραμμα PHP

Γράφουμε το παρακάτω κομμάτι κώδικα σε ένα αρχείο:

```
<html><body>
  <?php
       print ("Αυτό είναι ένα παράδειγμα!");
  ?>
</body></html>
```

Αποθηκεύουμε το αρχείο με κάποιο όνομα (το όνομα του αρχείου δεν πρέπει να έχει κενά) και την κατάληξη .php στον κατάλογο root του εξυπηρετητή μας (στα Windows είναι συνήθως ο κατάλογος "wwwroot" μέσα στον κατάλογο "inetpub" στο δίσκο C:).

Το επόμενο βήμα είναι να ανοίξουμε το αρχείο στον φυλλομετρητή μας. επειδή απαιτείται ο εξυπηρετητής για την εκτέλεση του κώδικα PHP, θα πρέπει να

ανοίξουμε το αρχείο μέσω ενός URL που θα εντοπίσει το σωστό αρχείο στον εξυπηρετητή μας. Στα Windows, θα το βρούμε κοιτώντας τις ρυθμίσεις δικτύου (Network settings) στον πίνακα ελέγχου (Control Panel). Στην καρτέλα "Identification" θα δούμε το "Computer name". Το όνομα του τοπικού υπολογιστή είναι το αρχικό URL. Αν, π.χ. το όνομα του υπολογιστή είναι "mycomp" για να δούμε τα περιεχόμενα του αρχικού καταλόγου γράφουμε "http://mycomp" στον φυλλομετρητή. Για να ανοίξουμε το "test.php" σε έναν κατάλογο που ονομάζεται "examples" μέσα στον αρχικό κατάλογο, θα γράψουμε "`http://mycomp/examples/test.php`" για να δούμε το παράδειγμα.

Αν έχουμε κωδικούς πρόσβασης σε ένα εξυπηρετητή δικτύου που υποστηρίζει PHP, μπορούμε να φορτώσουμε τα αρχεία μας με FTP οπουδήποτε στον εξυπηρετητή.

Μηνύματα λάθους

Αν δεν λειτουργήσει το παραπάνω παράδειγμα ή κάποιο πρόγραμμα που θα επιχειρήσουμε να τρέξουμε, θα δούμε ένα μήνυμα λάθους όπως αυτό:

```
Parse error: parse error in
C:\Inetpub\wwwroot\documents\test.php on line 12
```

Για κάθε γραμμή κώδικα στην οποία έχει εντοπιστεί κάποιο λάθος θα πάρουμε ένα τέτοιο μήνυμα λάθους. Αν πάρουμε κάποιο τέτοιο μήνυμα, πηγαίνουμε και ελέγχουμε το αρχείο μας test.php και κοιτάμε μήπως ξεχάσαμε κάποια ετικέτα τέλους, ένα σύμβολο (;) ή τα εισαγωγικά.

5.3.4 Οι μεταβλητές στην PHP

Οι μεταβλητές στην PHP, όπως και στην Perl, μπορούν να είναι ένας από τους παρακάτω τρεις τύπους:

- βαθμωτή (scalar),
- πίνακας (array), ή
- συσχετιζόμενος πίνακας (associative array).

Οι μεταβλητές είναι ο κύριος μηχανισμός για τη μεταφορά δεδομένων μεταξύ σελίδων ή τμημάτων σελίδων.

Υπάρχουν τρία βασικά πράγματα τα οποία μπορούμε να κάνουμε με μεταβλητές:

- Να τις θέσουμε, δηλαδή να τους δώσουμε μία ή και περισσότερες τιμές.
- Να τις επαναθέσουμε, δηλαδή να τους εκχωρήσουμε άλλη τιμή από αυτή που είχαν ως τώρα.
- Να τις προσπελάσουμε, δηλαδή να διαβάσουμε την τιμή μιας μεταβλητής και μετά να τις επεξεργαστούμε.

Βαθμωτές μεταβλητές

Μία βαθμωτή μεταβλητή μπορεί να περιέχει αριθμούς, γράμματα, φράσεις, κτλ. Οι μεταβλητές αυτές στην PHP ξεκινούν με το σύμβολο ("$"). Στο παρακάτω κομμάτι κώδικα ορίζουμε μία μεταβλητή, την χρησιμοποιούμε, μετά την ορίζουμε εκ νέου και την ξαναχρησιμοποιούμε. Η τιμή που έχει κάποια μεταβλητή μπορεί να αλλαχθεί κάθε στιγμή. Στο παρακάτω πρόγραμμα παρουσιάζεται ένα παράδειγμα για την αρχικοποίηση και την επανάθεση μεταβλητών.

```
<?php
  $stuff = "Τι κάνεις;";
   print ("Γεια σου, $stuff");
   print ("<p>");
   $stuff = "Πώς σε λένε;";
   print ("Είμαι μια χαρά, ευχαριστώ! Λοιπόν, $stuff");
?>
```

Όλες οι μεταβλητές ξεκινάνε με "$". Έτσι στο παραπάνω κώδικα αρχικά δημιουργήσαμε μία βαθμωτή μεταβλητή με το όνομα "stuff" και της δώσαμε την τιμή "Τι κάνεις;", χρησιμοποιώντας τη δήλωση: `$stuff = "Τι κάνεις;";`. Στην συνέχεια εκτυπώνουμε μια φράση σε συνδυασμό με την τιμή που περιέχει η μεταβλητή `$stuff`. Αφού έχουμε δημιουργήσει μια νέα παράγραφο στην ιστοσελίδα μας , με την εκτύπωση της ετικέτας <p>, επαναθέτουμε τη μεταβλητή `$stuff` δίνοντας της την τιμή "Πώς σε λένε;". Εκτυπώνουμε την φράση "Είμαι μια χαρά, ευχαριστώ! Λοιπόν, " σε συνδυασμό με την τιμή της μεταβλητής `$stuff` και τερματίζουμε το πρόγραμμά μας .

Με τα παραπάνω ορίσαμε και καλέσαμε μεταβλητές μέσα στον ίδιο κώδικα, αλλά η δύναμη της PHP βρίσκεται στο ότι μπορούμε να ορίσουμε μία μεταβλητή τοπικά — για παράδειγμα από μία φόρμα που συμπληρώνει ένας χρήστης — και μετά να χρησιμοποιήσουμε την μεταβλητή αργότερα.

Η σύνταξη για τον καθορισμό μεταβλητών είναι:

- να τις θέσουμε τιμή με το σύμβολο = , π.χ. `$stuff = "Τι κάνεις;";`
- να χρησιμοποιούμε εισαγωγικά αν η τιμή είναι χαρακτήρες (`"Τι κάνεις;"`. Οι αριθμοί δεν απαιτούν εισαγωγικά)
- να τελειώνουμε κάθε εντολή με το σύμβολο (;).

Καλούμε τη μεταβλητή απλά αναφέροντας το όνομά της.

Ονομασία μεταβλητών

Μπορούμε να ονομάσουμε μία μεταβλητή όπως θέλουμε, αρκεί να λάβουμε υπόψη μας τα ακόλουθα:

- Να αρχίζει με γράμμα
- Να αποτελείται από γράμματα, αριθμούς και τον χαρακτήρα υπογράμμισης (_);
- Δεν χρησιμοποιείται και αλλού (όπως "print").

Σημείωση: Στα ονόματα μεταβλητών έχουν σημασία τα κεφαλαία και τα πεζά, έτσι `$baby_names` και `$Baby_names` δεν είναι τα ίδια.

Μέχρι τώρα δίναμε στις μεταβλητές κείμενο ως τιμές, αλλά μπορούμε να τους δώσουμε και αριθμούς καθώς και άλλα αντικείμενα (objects, arrays, booleans).

Χρησιμοποιούμε απλά ή διπλά εισαγωγικά για να προβάλουμε κείμενο όπως στο:

```
Print ("Αυτό είναι ένα παράδειγμα!");
```

Η παραπάνω εντολή εκτυπώνει την φράση "Αυτό είναι ένα παράδειγμα!". Αν θελήσουμε να εκτυπώσουμε το κείμενο με τα εισαγωγικά πρέπει να τα διαφύγουμε με τον χαρακτήρα "\", που ορίζει στην PHP να μη χρησιμοποιήσει τον επόμενο χαρακτήρα σαν μέρος του κώδικα. Έτσι, για να εκτυπωθεί το κείμενο " Αυτό είναι ένα παράδειγμα!" (με τα εισαγωγικά να φαίνονται στο αποτέλεσμα) θα γράψουμε:

```
Print (" \" Αυτό είναι ένα παράδειγμα!\"" );
```

Μεταβλητές τύπου πίνακα

Οι πίνακες μας δίνουν την δυνατότητα να αποθηκεύσουμε όχι μόνο μία τιμή μέσα σε μία μεταβλητή, αλλά μία σειρά από τιμές σε μία μεταβλητή.
Αν θελήσουμε να καταγράψουμε όλους τους μαθητές μιας τάξης δημοτικού, θα μπορούσαμε να θέσουμε καθέναν ως μία κανονική μεταβλητή ως εξής:

```
$stu1 = "Μαρία";
$stu2 = "Γιάννης";
$stu3 = "Λευτέρης";
...
```

Ένας πίνακας μας επιτρέπει να αποθηκεύσουμε όλα αυτά σε μία μόνο μεταβλητή, που θα ονομάσουμε $students. Κάθε στοιχείο της μεταβλητής έχει το δικό του κλειδί που χρησιμοποιείται για την πρόσβαση εκείνου του στοιχείου του πίνακα, που μπορεί να είναι είτε σειρά γραμμάτων ή αριθμών.

Η συνάρτηση array() εκχωρεί μία σειρά τιμών στον πίνακα μας με τη μία με τον ακόλουθο τρόπο:

```
$students = array ( "Μαρία", "Γιάννης", "Λευτέρης" );
```

Αυτή η εντολή αποθηκεύει τα ονόματα όλων των μαθητών σε μία μεταβλητή ($students) η οποία είναι πίνακας, και αυτόματα εκχωρεί και ένα αριθμημένο κλειδί σε κάθε στοιχείο ξεκινώντας με τη σειρά και δίνοντας στο πρώτο στοιχείο τον αριθμό 0. Έτσι, η Μαρία είναι το στοιχείο [0], ο Γιάννης το [1], ο Λευτέρης το [2], κτλ..
Μπορούμε τώρα να αναφερόμαστε σε οποιοδήποτε στοιχείο του πίνακα ως εξής: $students[2].

Ο παρακάτω κώδικας θα εκτυπώσει το τρίτο στοιχείο του πίνακα, που είναι ο Λευτέρης.

```
<?php
print "$students[2]";
?>
```

Υπάρχει και ένας άλλος τρόπος για να ορίσουμε έναν πίνακα ή ακόμα και να προσθέσουμε σε έναν ήδη υπάρχοντα πίνακα, ορίζοντας κάθε στοιχείο ξεχωριστά:

```
$students[ ] = "Μαρία";
$students[ ] = "Γιάννης";
$students[ ] = "Λευτέρης";
```

Για να προσθέσουμε ένα νέο μαθητή γράφουμε (ανεξάρτητα από τον τρόπο που χρησιμοποιήσαμε για τη δημιουργία του πίνακα):

```
$students[ ] = "Βασίλης";
```

Η PHP δίνει αυτόματα στον Βασίλη έναν αριθμό, τον αμέσως επόμενο κενό δηλαδή, που σε αυτή την περίπτωση είναι το [3].

Μεταβλητές τύπου συσχετιζόμενου πίνακα

Οι συσχετιζόμενοι πίνακες διαχωρίζουν τα περιεχόμενα στοιχεία όχι με αριθμούς, αλλά με ονόματα που εμείς καθορίζουμε. Μέσα στη συνάρτηση array() καθορίζουμε ζεύγη όπου δίνουμε το όνομα του κλειδιού και την τιμή του χρησιμοποιώντας το συνδυασμό των συμβόλων "=" και ">", π.χ. : key=>"value". Για παράδειγμα ο κώδικας:

```
$StudJohn = array (
           name=>"John",
           haircolor=>"black",
           eyecolor=>"green",
           age=>17);
```

λέει στον πίνακα να δημιουργήσει τα κλειδιά "name", "haircolor" "eyecolor" και "age" και να δώσει σε κάθε κλειδί μία τιμή.

Μπορούμε να πάμε σε οποιοδήποτε τμήμα του πίνακα θελήσουμε μέσω των ονομάτων των κλειδιών που ορίσαμε. Για παράδειγμα:

```
print $StudJohn[eyecolor];
```

θα δώσει green. Μπορούμε επίσης να θέσουμε κάθε κλειδί ξεχωριστά:

```
$stud[name] = "John";
$stud[haircolor] = "black";
$stud[eyecolor] = "green";
$stud[age] = 17;
```

Ένας πολυδιάστατος πίνακας είναι ένας πίνακας (οι μαθητές της τάξης του δημοτικού) φτιαγμένος από άλλους πίνακες (για κάθε μαθητή, ένας πίνακας που περιέχει το όνομα, το χρώμα των μαλλιών, το χρώμα των ματιών και την ηλικία του μαθητή). Δημιουργούμε πολυδιάστατους πίνακες κατασκευάζοντας έναν πίνακα, όπως παρακάτω:

```
$stud = array (              );
```

Γεμίζουμε τον πίνακα αυτό με έναν πίνακα από μαθητές, στον οποίο έχουμε προσδιορίσει τα κλειδιά, ως εξής:

```
$stud = array (
           array ( name=>"John",
                     haircolor=>"black",
                     eyecolor=>"green",
                     age=>17 ),
           array ( name=>"Mary",
                     haircolor=>"blond",
                     eyecolor=>"blue",
                     age=>16 ),
           array ( name=>"Kenny",
                     haircolor=>"brown",
                     eyecolor=>"brown",
                     age=>17 ),
           array ( name=>"Bill",
                     haircolor=>"blond",
```

```
                    eyecolor=>"green",
                    age=>16 )
                           );
```

Για να τον χρησιμοποιήσουμε, μπορούμε να πάρουμε οποιοδήποτε μέρος της πληροφορίας που περιλαμβάνεται σε αυτόν δίνοντας το όνομα του πρώτου πίνακα $stud (του πίνακα που περιέχει τους άλλους), το όνομα του υπο-πίνακα και μετά το όνομα του κλειδιού για το χαρακτηριστικό που θέλουμε να πάμε.

Για να βρούμε την ηλικία του Kenny θα γράψουμε:

```
Print $stud[2][age];
```

5.3.5 Τελεστές

Οι τελεστές χρησιμοποιούνται για πράξεις ανάμεσα στις μεταβλητές. Υπάρχουν τρεις διαφορετικοί τύποι τελεστών: σύγκρισης, λογικοί και αριθμητικοί.

Τελεστές σύγκρισης

Είδαμε σε προηγούμενη παράγραφο πώς χρησιμοποιούμε το σύμβολο "=", όταν ορίσαμε μεταβλητές στα σενάρια/προγράμματα που γράψαμε. Το σύμβολο αυτό ονομάζεται τελεστής εκχώρησης και είναι ο πιο απλός τελεστής. Π.χ. `$a = b` σημαίνει ότι «στη μεταβλητή a εκχωρείται η τιμή b».

Εκτός από τον τελεστή εκχώρησης, έχουμε και τελεστές σύγκρισης, λογικούς τελεστές και αριθμητικούς τελεστές.

Ο τελεστής σύγκρισης μας δίνει τη δυνατότητα να συγκρίνουμε αν τα στοιχεία είναι ίσα, πανομοιότυπα, μικρότερο ή μεγαλύτερο το ένα από το άλλο. Στον πίνακα 5.3.1 παρουσιάζονται οι τελεστές σύγκρισης όπως αυτοί χρησιμοποιούνται από την PHP.

Πίνακας 5.3.1 – Οι τελεστές σύγκρισης της PHP

Τελεστής	Περιγραφή
==	Επιστρέφει την τιμή ‘αληθής’ (true) αν η τιμή της μεταβλητής στα αριστερά είναι ίση με αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false).
===	Επιστρέφει την τιμή ‘αληθής’ (true) αν η τιμή της μεταβλητής αριστερά είναι ίση με αυτή στα δεξιά, και είναι και του ίδιου τύπου. (PHP 4 μόνο). Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false).
!=	Επιστρέφει την τιμή ‘αληθής’ (true) αν η τιμή της μεταβλητής αριστερά είναι δεν ίση με αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false).
<>	Επιστρέφει την τιμή ‘αληθής’ (true) αν η τιμή της μεταβλητής αριστερά είναι δεν ίση με αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false).
!==	Επιστρέφει την τιμή ‘αληθής’ (true) αν η τιμή της μεταβλητής αριστερά είναι δεν ίση με αυτή στα δεξιά και δεν είναι και του ίδιου τύπου. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false).
<	Επιστρέφει την τιμή ‘αληθής’ (true) αν η τιμή της μεταβλητής αριστερά είναι μικρότερη από αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false).

>	Επιστρέφει την τιμή ‘αληθής’ (true) αν η τιμή της μεταβλητής αριστερά είναι μεγαλύτερη από αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false).
<=	Επιστρέφει την τιμή ‘αληθής’ (true) αν η τιμή της μεταβλητής αριστερά είναι μικρότερη ή ίση με αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false).
>=	Επιστρέφει την τιμή ‘αληθής’ (true) αν η τιμή της μεταβλητής αριστερά είναι μεγαλύτερη ή ίση με αυτή στα δεξιά. Αλλιώς επιστρέφει την τιμή ‘ψευδής’ (false).

Λογικοί Τελεστές

Στον πίνακα 5.3.2 παρουσιάζονται οι λογικοί τελεστές όπως αυτοί χρησιμοποιούνται από την PHP.

Πίνακας 5.3.2 – Οι λογικοί τελεστές της PHP

Τελεστής	Περιγραφή
and	Επιστρέφει την τιμή ‘αληθής’ (true) αν η τιμή της μεταβλητής αριστερά και στα δεξιά είναι ‘αληθής’. Αλλιώς αν κάποια από τις δύο είναι ‘ψευδής’ επιστρέφει την τιμή ‘ψευδής’ (false).
or	Επιστρέφει την τιμή ‘αληθής’ (true) αν η τιμή της μεταβλητής αριστερά ή στα δεξιά είναι ‘αληθής’, είτε είναι ‘αληθής’ και οι δύο. Αλλιώς αν και οι δύο είναι ‘ψευδής’ επιστρέφει την τιμή ‘ψευδής’ (false).
xor	Επιστρέφει την τιμή ‘αληθής’ (true) αν η τιμή της μεταβλητής αριστερά ή στα δεξιά είναι ‘αληθής’. Αλλιώς αν και οι δύο είναι ‘ψευδής’, είτε είναι ‘αληθής’ μία από τις δύο, επιστρέφει την τιμή ‘ψευδής’ (false).
!	Επιστρέφει την τιμή ‘αληθής’ (true) αν η τιμή της μεταβλητής δεν είναι ‘αληθής’. Αλλιώς η τιμή της μεταβλητής είναι ‘αληθής’ επιστρέφει την τιμή ‘ψευδής’ (false).
&&	Επιστρέφει την τιμή ‘αληθής’ (true) αν η τιμή της μεταβλητής αριστερά και στα δεξιά είναι ‘αληθής’. Αλλιώς αν κάποια από τις δύο είναι ‘ψευδής’ επιστρέφει την τιμή ‘ψευδής’ (false). Ίδιος με τον τελεστή ‘and’.
||	Επιστρέφει την τιμή ‘αληθής’ (true) αν η τιμή της μεταβλητής αριστερά ή στα δεξιά είναι ‘αληθής’, είτε είναι ‘αληθής’ και οι δύο. Αλλιώς αν και οι δύο είναι ‘ψευδής’ επιστρέφει την τιμή ‘ψευδής’ (false). Ίδιος με τον τελεστή ‘or’.

Αριθμητικοί Τελεστές

Στον πίνακα 5.3.3 παρουσιάζονται οι αριθμητικοί τελεστές όπως αυτοί χρησιμοποιούνται από την PHP.

Πίνακας 5.3.3 – Οι αριθμητικοί τελεστές της PHP

Τελεστής	Περιγραφή
+	Αθροίζει δύο τιμές.

-	Αφαιρεί δύο τιμές.
*	Πολλαπλασιάζει δύο τιμές.
/	Διαιρεί δύο τιμές.
%	Παρουσιάζει το υπόλοιπο της διαίρεσης δύο αριθμών.

5.3.6 Εντολές διακλάδωσης στην PHP

Όλη η ιδέα των δυναμικών ιστοσελίδων βρίσκεται στο να κάνουμε την ιστοσελίδα μας όσο πιο έξυπνη γίνεται — να έχουμε κώδικα σε σημείο όπου θα μπορεί να πάρει όλων των ειδών τις αποφάσεις βασισμένος σε διαφορετικά δεδομένα εισόδου από τον χρήστη, σε συνθήκες των χρηστών (τι είδους φυλλομετρητή χρησιμοποιεί ο επισκέπτης Χ) ή πληροφορία που μπορούμε να καθορίσουμε οι ίδιοι.

Παραδείγματα αυτών μπορεί να είναι:

- Αφού ένας χρήστης εισάγει μία διεύθυνση ηλεκτρονικού ταχυδρομείου, να ελέγχει αν αυτή έχει τη σωστή μορφή (whoever@wherever.com) και αν δεν έχει τη μορφή αυτή, να δείχνει μία σελίδα που να λέει π.χ. "Please, enter a valid email address!"
- Να εξυπηρετεί δύο ή και περισσότερες διευθύνσεις π.χ. μία με κατάληξη .com και μία άλλη σε .gr.
- Να ξέρουμε πότε ένας πελάτης στο ηλεκτρονικό μας κατάστημα συναντά ένα μικρό ποσό χρημάτων για μία ηλεκτρονική πληρωμή με πιστωτική κάρτα.

Εντολή διακλάδωσης If

Ο τρόπος για να δημιουργήσουμε έξυπνες ιστοσελίδες είναι να χρησιμοποιήσουμε δηλώσεις / εντολές διακλάδωσης, όπως οι: If, Else, και Elseif, μαζί με τελεστές σύγκρισης και λογικούς τελεστές. Το πιο σημαντικό από αυτά είναι η δήλωση if, που δίνει τη δυνατότητα να αποδώσουμε με κώδικα τις παρακάτω συνθήκες :

```
ΑΝ κάποια συνθήκη είναι αληθής, τότε κάνε κάτι.
ΑΝ η συνθήκη είναι ψευδής, τότε αγνόησέ το.
```

Η σύνταξη είναι η ακόλουθη:

```
if (συνθήκη) {
// Ο κώδικας αυτός θα εκτελεστεί εάν η συνθήκη είναι αληθής
}
```

Στο παρακάτω παράδειγμα πρώτα θέτουμε τη μεταβλητή `$FavoriteDay` ίση με την τιμή «Παρασκευή». Στη συνέχεια εξετάζουμε την συνθήκη εάν η τιμή της μεταβλητής `$FavoriteDay` είναι ίση με «Παρασκευή» και εφόσον είναι ίση εκτυπώνεται η φράση «Μου αρέσει η Παρασκευή πολύ!!».

```
<?php
$FavoriteDay = "Παρασκευή";
if ($FavoriteDay == "Friday") {
print ("Μου αρέσει η Παρασκευή πολύ!!");
}
```

```
?>
```

Εντολή διακλάδωσης Else

Το Else ενσωματώνεται στη δήλωση του if και μπορούμε να εκφράσουμε το εξής:

```
ΑΝ κάποια συνθήκη είναι αληθής, κάνε κάτι;
ΑΛΛΙΩΣ, στην περίπτωση που η πρώτη συνθήκη δεν είναι αληθής,
κάνε κάτι άλλο.
```

Η σύνταξη είναι η ακόλουθη:

```
if (συνθήκη) {
// Ο κώδικας αυτός θα εκτελεστεί εάν η συνθήκη είναι αληθής
} else {
// Ο κώδικας αυτός θα εκτελεστεί εάν η συνθήκη είναι ψευδής
}
```

Ένα παράδειγμα που χρησιμοποιεί το else είναι το παρακάτω:

```
<?php
$FavoriteDay = " Παρασκευή”;
 if ($FavoriteDay == "Σαββάτο") {
print ("Μου αρέσει η Παρασκευή πολύ!!);
} else {
print ("Δεν σου αρέσει το Σαββάτο!!");
}
?>
```

Εντολή διακλάδωσης Elseif

Ενώ το else είναι κατά κάποιο τρόπο ένας έλεγχος που εκτελεί κάτι όσο η δήλωση if δεν είναι αληθής, το elseif εκτελεί κάτι αν συναντηθεί μία συγκεκριμένη συνθήκη:

```
ΑΝ κάποια συνθήκη είναι αληθής, κάνε κάτι.
ΑΛΛΙΩΣ ΑΝ κάποια άλλη συγκεκριμένη συνθήκη είναι αληθής, κάνε
κάτι άλλο.
```

Στο παρακάτω παράδειγμα φαίνεται η χρήση της elseif:

```
<?php
$FavoriteDay = "Κυριακή";

if ($FavoriteDay== "Σαββάτο") {
print ("Μου αρέσει το Σαββάτο πολύ!!");
} elseif ($FavoriteDay = Κυριακή) {
print ("Επιτέλους, Κυριακή!");
}
?>
```

Θα μπορούσαμε να προσθέσουμε και ένα else στο τέλος για να καλύψουμε την περίπτωση που το FavoriteDay δεν έχει την τιμή «Σαββάτο», ούτε τη τιμή «Κυριακή».

5.3.7 Βρόχοι επανάληψης

Οι βρόχοι επανάληψης είναι πολύ χρήσιμοι, γιατί επιτρέπουν να προγραμματίσουμε τον κώδικα να επαναλαμβάνεται μέχρι να ικανοποιούνται κάποιες συνθήκες. Υπάρχουν διάφορα είδη βρόχων επανάληψης. Ας δούμε τον πιο βασικό εδώ, που είναι ο βρόχος "while":

```
while (συνθήκη που είναι αληθής)
{
// κάνε κάποια συγκεκριμένη ενέργεια
}
```

Οι βρόχοι while χρησιμοποιούνται συχνά με μία μεταβλητή που είναι ακέραιος, ο οποίος αυξάνεται ή μειώνεται κατά μία μονάδα, κάθε φορά που εκτελείται ο βρόχος. Αυτό σημαίνει ότι μπορούμε να κάνουμε το πρόγραμμά μας να προσθέτει (ή να αφαιρεί) σε μια μεταβλητή το ένα (ή το δύο, ή το τρία, κτλ..) κάθε φορά που διατρέχεται ο βρόχος μέχρι η μεταβλητή να φτάσει σε μία ανώτατη ή κατώτατη τιμή που εμείς ορίσαμε.

Έτσι, αν θέλουμε ένα σενάριο που θα μας εκτυπώνει αριθμούς από 1 μέχρι 10, πρέπει να ακολουθήσουμε τα παρακάτω βήματα:

α. η μεταβλητή $Digit = 1.
β. εκτύπωσε $Digit.
γ. πρόσθεσε 1 στο $Digit.
δ. πήγαινε στο βήμα β. και εκτέλεσε το σενάριο πάλι με τη νέα τιμή του $Digit.
ε. σταμάτα όταν το $Digit φτάσει το 11.

Η σύνταξη για την αύξηση ή μείωση της τιμής μιας μεταβλητής είναι:

$a++;	προσθέτει κάθε φορά 1 στη τιμή της μεταβλητής $a
$a--;	αφαιρεί κάθε φορά 1 από την τιμή της μεταβλητής $a

Ο κώδικας σε PHP θα είναι:

```
<?php
$Digit = 1;
while ($Digit <= 10)
{
print ("$Digit");
$Digit++;
}
?>
```

Στο παραπάνω παράδειγμα αρχικά ορίζουμε μια μεταβλητή και την αρχικοποιούμε στη τιμή 1. Στην συνέχεια όσο η μεταβλητή είναι μικρότερη από το 10 (α) εκτυπώνεται και (β) αυξάνεται κατά μία μονάδα. Όταν η μεταβλητή γίνει ίση ή μεγαλύτερη από 10 το πρόγραμμα τερματίζεται.

5.3.8 Εισαγωγή δεδομένων

Παρακάτω θα δούμε πώς μπορούμε να δημιουργήσουμε φόρμες HTML για να συλλέξουμε πληροφορία εισόδου από τον χρήστη. Τα δεδομένα τοποθετούνται σε μεταβλητές και επεξεργάζονται την πληροφορία που συλλέξαμε. Θα φτιάξουμε μία ιστοσελίδα που περιέχει μια φόρμα με πεδία για την συλλογή ονόματος και στοιχείων διεύθυνσής , τα οποία τα εμφανίζει εκτυπωμένα σε μία άλλη ιστοσελίδα.

Ας δημιουργήσουμε την πρώτη ιστοσελίδα που περιέχει μία φόρμα, χρησιμοποιώντας απλή HTML και ας την ονομάσουμε “myform.html”.

```
<html>
 <head>
    <title>Πρώτη ιστοσελίδα με φόρμα</title>
 </head>
 <body>

  <form action="address.php" method="post">

    Το όνομά μου είναι:
    <br> <input type="text" name="YourName">

   <p> Η διεύθυνση μου είναι: <br>
   <input type="text" name="YourAddr">
   <p>
   <input type="submit" name="submit" value="Send">
  </form>

 </body>
</html>
```

Αυτή είναι μία απλή φόρμα σε HTML. Αναλυτικότερα ο παραπάνω κώδικας με τη δήλωση `<form action="address.php" method="post">` λέει στον φυλλομετρητή ποιο έγγραφο PHP θα επεξεργάζεται τα αποτελέσματα της φόρμας, καθώς και ποια μέθοδος (Post ή Get) θα χρησιμοποιηθεί για την συλλογή των δεδομένων. Στη συνέχεια με τη δήλωση `<input type="text" name="YourName">` καθορίζουμε ότι το στοιχείο της φόρμας που θέλουμε εδώ είναι κείμενο "text" ή ένα πεδίο κειμένου (θα μπορούσαμε να είχαμε και ένα radio button, ή check box, κτλ..). Η δήλωση `name="YourName"` καθορίζει πως ο,τιδήποτε γράψει ο χρήστης μέσα στο πεδίο κειμένου θα γίνει μία μεταβλητή που καλείται "YourName". Δημιουργούμε άλλο ένα στοιχείο στη φόρμα με την δήλωση `<input type="text" name="YourAddr">` , που χρησιμοποιείται για την συλλογή της διεύθυνσης του χρήστη. Η δήλωση `<input type="submit" name="submit" value="Send">` δημιουργεί ένα κουμπί αποστολής με την επιγραφή "Send".

Παρακάτω θα δούμε την δομή του και το περιεχόμενο του αρχείου `address.php` το οποίο επεξεργάζεται τα δεδομένα εισόδου της προηγούμενης φόρμας. Το αρχείο αυτό έχει τον ακόλουθο κώδικα:

```
<html>
<head>
<title>Δεύτερη ιστοσελίδα</title>

</head>
```

```
<body bgcolor="#FFFFFF" text="#000000">

<p>
Γειά σου <?php print $YourName; ?>

<p>
Η διεύθυνση σου είναι: <b> <?php print $YourAddr; ?> !?! </b>

<p>Bye!

</body>
</html>
```

Βλέπουμε ότι για να "περάσουμε" μεταβλητές από μια ιστοσελίδα σε μια άλλη αυτές θα πρέπει να περιλαμβάνονται σε PHP κώδικα. Έτσι στο παραπάνω παράδειγμα η δήλωση `<?php print $YourName;?>` εκτυπώνει την μεταβλητή `YourName` που δημιουργήθηκε και έλαβε τιμή σε προηγούμενη ιστοσελίδα.

5.3.9 Προχωρημένες συναρτήσεις

Η PHP είναι μια εξαιρετικά εύκαμπτη γλώσσα σεναρίων, η οποία παρέχει πολλές δυνατότητες πέρα από μία βιβλιοθήκη με έτοιμες συναρτήσεις. Από ταξινόμηση σε αλφαβητική σειρά μέχρι αποστολή ηλεκτρονικού ταχυδρομείου και σύνδεση με βάση δεδομένων, μας δίνει και τη δυνατότητα να δημιουργήσουμε τις δικές μας συναρτήσεις για να κάνουν ότι χρειαζόμαστε σχετικά με την ιστοσελίδα μας. Οι συναρτήσεις που δημιουργούμε εκτελούνται με τον ίδιο τρόπο με τις συναρτήσεις βιβλιοθήκης της PHP.

Συναρτήσεις που δημιουργούμε εμείς έχουν την ίδια μορφή:

```
<?php
function MyFunction ()
{
//Κώδικας συνάρτησης
}
?>
```

Ξεκινάμε μία συνάρτηση με τις λέξεις function NameOfFunction(), με τις λέξεις NameOfFunction() να είναι λέξεις τις επιλογής μας (χωρίς κενά).
Στη συνέχεια καθορίζουμε τους κανόνες της συνάρτησης εντός των αγκύλων που ακολουθούν « {} ».

Υπάρχουν συναρτήσεις που απαιτούν ορίσματα και συναρτήσεις που δεν απαιτούν ορίσματα. Ένα όρισμα είναι μία μεταβλητή που έρχεται εξωτερικά της συνάρτησης, αλλά που η συνάρτηση χρειάζεται για να εκτελεστεί.

Ας δούμε μία συνάρτηση που δεν χρειάζεται ορίσματα:

```
<?php
function test()
{
   print "<b>Αυτό είναι ένα απλό παράδειγμα συνάρτησης </b>";
}
test();
?>
```

Στο παραπάνω παράδειγμα με την δήλωση `function test()`ξεκινάμε τον ορισμό της συνάρτησης με όνομα `test`. Το περιεχόμενο της συνάρτησης περικλείεται σε άγκιστρα ({) και εκτυπώνει απλά τη φράση "`Αυτό είναι ένα απλό παράδειγμα συνάρτησης`". Στη συνέχεια γίνεται η κλήση της συνάρτησης με την εντολή `test();`.

Για να δούμε μία συνάρτηση που απαιτεί ορίσματα, ας πάρουμε το εξής παράδειγμα: είστε εργαζόμενος και πληρώνεστε με την ώρα. Η απλή συνάρτηση που θα γράψουμε θα υπολογίζει το μηνιαίο μισθό σας.
Πρώτα θα δημιουργήσουμε μία σελίδα με μία φόρμα και ένα πεδίο όπου θα δίνετε τις ώρες που δουλέψατε το μήνα. Ο αριθμός που θα εισάγετε θα περάσει σε μία μεταβλητή με το όνομα `$hours`.

Ο κώδικας της συνάρτησης θα είναι:

```
<html>
<head>
<title>Μισθός</title>
</head>
<body>
<?php
$earnings_per_hour = 2000;
function MyPayment($hours, $earnings_per_hour) {
$monthly_payment = ($hours*$earnings_per_hour);
print "Αυτό το μήνα θα εισπράξεις: \$$monthly_payment!";
}
MyPayment($hours, $earnings_per_hour);
?>
</body>
</html>
```

Στο παραπάνω παράδειγμα αρχικά θέτουμε την μεταβλητή `$earnings_per_hour` σε 2000. Στη συνέχεια δημιουργούμε την συνάρτηση `MyPayment` που σχετίζεται με τις μεταβλητές `$hours` και `$earnings_per_hour`. Έπειτα `$monthly_payment`, η τιμή της οποίας είναι οι ώρες επί το μισθό ανά ώρα. Εκτυπώνουμε στην οθόνη μία πρόταση που χρησιμοποιεί την τιμή της μεταβλητής `$monthly_payment`. Τέλος με τη δήλωση `MyPayment($hours, $earnings_per_hour);`, καλούμε και εκτελούμε την συνάρτηση με ορίσματα τις μεταβλητές `$hours` και `$earnings_per_hour` .

***Σημείωση*:** Με το σύμβολο του δολαρίου ($) ξεκινάνε οι μεταβλητές και όταν τον χρησιμοποιούμε σε κάποια εκτύπωση το σύστημα καταλαβαίνει ότι θα ακολουθήσει το όνομα κάποιας μεταβλητής. Για να δώσουμε στο σύστημα να καταλάβει ότι πρόκειται για κάποιον απλό χαρακτήρα και όχι αρχή ονόματος μεταβλητής, χρησιμοποιούμε το σύμβολο “/”. Όπως για παράδειγμα στη δήλωση print `" Αυτό το μήνα θα εισπράξεις: \$$monthly_payment!";`.

5.4 ASP

ASP είναι τα αρχικά του Active Server Pages, η οποία είναι μία τεχνολογία της Microsoft. Η ASP παρέχει ένα περιβάλλον προγραμματισμού στην πλευρά του εξυπηρετητή. Είναι ένα ακόμα εξαιρετικό εργαλείο για τη δημιουργία δυναμικών ιστοσελίδων. Η ASP χρησιμοποιεί την HTML για να εμφανίσει πληροφορία στην πλευρά του πελάτη (σε κάποιον φυλλομετρητή δηλαδή). Όταν ένας φυλλομετρητής ζητήσει μία σελίδα ASP, ο εξυπηρετητής επεξεργάζεται το αίτημα και επιστρέφει αποτελέσματα στον φυλλομετρητή σε μορφή απλής HTML. Ο πυρήνας της ASP είναι το σενάριο. Η ASP λειτουργεί αυτόματα με δημοφιλείς γλώσσες σεναρίων όπως VBScript και Jscript. Είναι δυνατόν, όμως, να περιλαμβάνει και άλλες γλώσσες, όπως JavaScript ακόμα και Perl αρκεί να υπάρχουν εγκατεστημένα τα αντίστοιχα προγράμματα διερμηνείας των γλωσσών αυτών. Στα παραδείγματα που θα ακολουθήσουν θα δημιουργήσουμε σενάρια ASP, χρησιμοποιώντας την VBScript. Η ASP μπορεί να συνδεθεί με όλες τις γνωστές βάσεις δεδομένων συμπεριλαμβανομένων των Access, SQL Server και Oracle. Ένα αρχείο ASP έχει την κατάληξη ".asp".

Η συνηθισμένη γλώσσα σεναρίων της ASP είναι η VBScript (Visual Basic Script) της Microsoft, όπως είδαμε στο κεφάλαιο 4. Είναι ένα υποσύνολο της Visual Basic που χρησιμοποιείται στις ιστοσελίδες για τον εμπλουτισμό τους.

Ένα αρχείο ASP περιέχει ετικέτες της HTML και κώδικα που τρέχει στον εξυπηρετητή και δεν εμφανίζεται στον φυλλομετρητή του χρήστη εντός των συμβόλων <% και %>. Τα σενάρια/προγράμματα του εξυπηρετητή εκτελούνται στον εξυπηρετητή και τα αποτελέσματα αποστέλλονται στον πελάτη. Γι' αυτό το λόγο ο πηγαίος κώδικας ASP δεν φαίνεται ποτέ στον φυλλομετρητή.

```
<%@ Language=VBScript %>
<% Response.Buffer = True %>
<html>
<head>
<title>My first ASP page</title>
</head>
<body>
<h2>Welcome!</h2>
<br><br>
<h2><% Response.Write ("This is my first ASP page!" %></h2>
<br><br>
<h3>The current time is <%= Now %></h3>
</body>
</html>
```

Η γραμμή `<%@ Language=VBScript %>` καθορίζει την γλώσσα σεναρίων που χρησιμοποιείται. Τα σύμβολα `<% ... %>` λένε στον εξυπηρετητή ότι το πρόγραμμα/σενάριο που υπάρχει μέσα τους πρέπει να εκτελεστεί πριν η σελίδα μεταδοθεί στον φυλλομετρητή. Αν δεν υπάρχει καθορισμός γλώσσας, η VBScript είναι η προκαθορισμένη γλώσσα.

Η γραμμή `<% Response.Buffer = True %>` ορίζει τον εξυπηρετητή να δημιουργήσει ολόκληρη την ιστοσελίδα στη μνήμη πριν τη στείλει στον πελάτη. Προκαθορισμένα, η ιστοσελίδα στέλνεται στον πελάτη γραμμή προς γραμμή, όπως ακριβώς δημιουργείται.

5.4.1 Μεταβλητές της VBScript

Οι μεταβλητές χρησιμοποιούνται για την προσωρινή αποθήκευση δεδομένων που χρειάζεται το πρόγραμμα. Όλες οι μεταβλητές που χρησιμοποιούνται μέσα στο πρόγραμμα πρέπει να έχουν οριστεί. Στην VBScript οι μεταβλητές ορίζονται με την δεσμευμένη λέξη Dim. Επίσης στην VBScript δεν διαφέρουν τα ονόματα αν είναι γραμμένα με κεφαλαία ή πεζά (έτσι, vbscript είναι το ίδιο με VBScript). Η τιμή μιας μεταβλητής μπορεί να αντικατασταθεί οποιαδήποτε στιγμή με μία άλλη τιμή.

Μερικά παραδείγματα καθορισμού μεταβλητών και εκχώρησης τιμών:

```
<%
Dim MyName
MyName = “My Name”
Response.Write(“My name is “ & MyName)
%>
```

5.4.2 Εντολές διακλάδωσης

Πολλές φορές όταν γράφουμε κώδικα, θέλουμε να κάνει διάφορες ενέργειες που εξαρτώνται από συγκεκριμένες συνθήκες.

Η VBScript προσφέρει τις εξής δύο τεχνικές διακλάδωσης:

Εντολή διακλάδωσης IF…THEN

Χρησιμοποιούμε την εντολή αυτή όταν έχουμε λίγες επιλογές έκβασης. Πρώτα αποτιμάται η συνθήκη και μετά καθορίζονται οι εντολές που θα εκτελεστούν.

Η εντολή έχει τη γενική μορφή:

```
IF condition THEN
      [statements to execute]
ELSE
      [other statements]
END IF
```

Ένα παράδειγμα:

```
<%
      IF NewMember=TRUE then
            Response.Write "Welcome, new member!"
      ELSE
            Response.Write "Welcome back, existing member!"
      END IF
%>
```

Μπορούμε να χρησιμοποιήσουμε περισσότερες από μία εντολές σε κάθε όρο/πρόταση. Ο όρος ELSE είναι προαιρετικός.

SELECT…CASE

Χρησιμοποιείται όταν έχουμε πολλές διακλαδώσεις. Η εντολή SELECT συγκρίνει μία μεταβλητή με μία λίστα τιμών. Όταν βρει τιμές να ταιριάζουν, εκτελεί τα βήματα που ακολουθούν.

Η γενική σύνταξη της εντολής είναι:

```
SELECT CASE variable
CASE value1
```

```
      [statements1]
CASE value2
      [statements2]
…
CASE ELSE
      [other statements]
END SELECT
```

Γενικά εφαρμόζουμε την SELECT…CASE για περισσότερες από δύο πιθανότητες (cases). Μπορούμε να έχουμε και πιο πολλές εντολές σε κάθε πρόταση CASE. Οι εντολές στην CASE ELSE θα εκτελεστούν αν δεν ταιριάζει καμία CASE. Ο όρος αυτός εφαρμόζεται για να καλύψει μη αναμενόμενες καταστάσεις.
Παράδειγμα:

```
select case FavoriteColor
case "Black"
Response.Write "Black black heart…"
case "Blue"
Response.Write "Blue eyes, baby's got blue eyes ..."
case "Red"
Response.Write "Red Red Wine ..."
case else
Response.Write "I've never heard of that color before!"
end select
```

5.4.3 Βρόχοι επανάληψης

Ένας βρόχος επαναλαμβάνει μία ακολουθία βημάτων μέχρι να γίνει αληθής μία συγκεκριμένη συνθήκη.

Βρόχος FOR

Χρησιμοποιούμε ένα βρόχο for για να επαναλάβουμε ένα τμήμα κώδικα για ένα καθορισμένο αριθμό επαναλήψεων.

```
FOR counter=startvalue TO endvalue [STEP stepvalue]
      [statements to execute]
NEXT
```

Ο μετρητής (counter) είναι μία αριθμητική μεταβλητή, που αυξάνει τον εαυτό της από μία τιμή βημάτων (stepvalue) κάθε φορά που ο βρόχος επαναλαμβάνεται. Το startvalue είναι η αρχική τιμή του μετρητή (counter). Η τελική τιμή (endvalue) είναι η τιμή τέλος του μετρητή. Όταν ο μετρητής είναι μεγαλύτερος από την τελική τιμή, ο βρόχος σταματάει αν η τιμή βημάτων (stepvalue) είναι θετικός αριθμός. Αν η τιμή βημάτων (stepvalue) είναι αρνητικός αριθμός, ισχύει το αντίθετο. Η τιμή βήματος αυξάνει σταθερά τον μετρητή, ο οποίος αυξάνει κατά 1 αν δεν χρησιμοποιείται τιμή για το βήμα.

```
<%
      FOR loop_c = 1 TO 5
      Response.Write (loop_c, <br>)
      NEXT
%>
```

Βρόχος WHILE

Ένας βρόχος WHILE επαναλαμβάνεται όσο μία συγκεκριμένη συνθήκη είναι αληθής. Σταματάει να εκτελείται μόλις η συνθήκη γίνει ψευδής.

```
DO WHILE condition
      [statements to execute]
LOOP
```

Πρώτα γίνεται αποτίμηση της συνθήκης. Αν είναι αληθής, θα εκτελεστούν οι εντολές που βρίσκονται μέσα στον βρόχο. Οι εντολές αυτές θα εκτελούνται επαναλαμβανόμενα μέχρι η συνθήκη να γίνει ψευδής.

```
DO
      [statements to execute]
LOOP WHILE condition
```

Πρώτα εκτελούνται μία φορά οι εντολές μέσα στον βρόχο. Γίνεται αποτίμηση της συνθήκης στη συνέχεια και αν η συνθήκη είναι αληθής, ο βρόχος επαναλαμβάνεται. Οι εντολές μέσα στο βρόχο εκτελούνται επαναλαμβανόμενα μέχρι η συνθήκη να γίνει ψευδής. Ο βρόχος σε αυτή την περίπτωση εκτελείται τουλάχιστον μία φορά.

```
<%
      Dim loop_c
      loop_c = 1
      DO
            Response.Write (loop_c, <br>)
            loop_c = loop_c +1
      WHILE loop_c < 5
%>
```

5.4.4 Διαδικασίες και υπορουτίνες της VBScript

Μία διαδικασία ή υπορουτίνα είναι ένα σύνολο εντολών που επιστρέφει αποτελέσματα σε συγκεκριμένα ερωτήματα. Χρησιμοποιώντας μία διαδικασία μας επιτρέπεται να εκτελέσουμε την ίδια αποστολή περισσότερες από μία φορές χωρίς να ξαναγράψουμε τον κώδικα. Υπάρχουν δύο βήματα για να δημιουργήσουμε μία διαδικασία:

- χρησιμοποιώντας τις δεσμευμένες λέξεις SUB και End SUB για τον ορισμό μιας υπορουτίνας και
- χρησιμοποιώντας CALL για να καλέσουμε την υπορουτίνα που μόλις ορίσαμε.

Ορίζουμε, για παράδειγμα, μία διαδικασία με όνομα convert, όπως παρακάτω:

```
<%
      SUB convert ()
      'Convert 100 Euros to DM and round to 2 decimal
      cur = round((100*2), 2)
      'Write the result out
```

```
      Response.Write(Euros 100 is equivalent to " & cur & "
DM.")
      END SUB
%>
```

και στη συνέχεια την καλούμε ως εξής:

```
<% Call convert %>
```

***Σημείωση*:** Παρατηρήστε στον παραπάνω κώδικα ότι πριν τα σχόλια προηγείται το σύμβολο (').

5.4.5 Συναρτήσεις της VBScript

Όπως ακριβώς σε μία υπορουτίνα, μία συνάρτηση ορίζει μία σειρά από δηλώσεις/εντολές που εκτελούνται κάθε φορά που την καλούμε. Η διαφορά μεταξύ μιας συνάρτησης και μιας υπορουτίνας είναι ότι η τελευταία εκτελεί μία σειρά βημάτων και μετά τελειώνει, ενώ η πρώτη εκτελεί κάποιον κώδικα και επιστρέφει μία τιμή. Για να ορίσουμε μία συνάρτηση χρησιμοποιούμε τις δεσμευμένες λέξεις FUNCTION,…END FUNCTION.

Ορισμός μιας συνάρτησης με το όνομα convert_func

```
<%
      FUNCTION convert_func(c_eur)
      'Convert Euros to DM and round to 2 decimal
      cur = round((c_eur*2), 2)
      END FUNCTION
%>
```

Στη συνέχεια καλούμε τη συνάρτηση με όρισμα το 100 και παίρνουμε τα αποτελέσματα:

```
<%
      Dim cur_c
      cur_c = convert_func( 100 )
      Response.Write("Euro 100 is equivalent to "& cur_c &".")
%>
```

Στη δήλωση: `convert_func(c_eur)`, το `c_eur` καλείται όρισμα. Ένα όρισμα είναι μία μεταβλητή που δίνεται στη συνάρτηση και μπορεί να χρησιμοποιηθεί από τη συνάρτηση για να κάνει τους υπολογισμούς.

5.4.6 Αντικείμενα στην ASP

Στην ASP ένα αντικείμενο είναι μία λογισμική αναπαράσταση ενός αντικειμένου του πραγματικού κόσμου. Ένα αντικείμενο έχει ιδιότητες (properties), μεθόδους (methods) και γεγονότα (events). Για την ανάπτυξη εφαρμογών Ιστού χρησιμοποιούνται συνήθως πέντε ενσωματωμένα στην αντικείμενα ASP.

Αντικείμενο Response

Όταν κάποιος χρήστης δώσει μία διεύθυνση ιστοσελίδας ή πατήσει κάποιον σύνδεσμο, τότε στέλνεται ένα αίτημα στον εξυπηρετητή. Ο εξυπηρετητής με τη σειρά του χειρίζεται το αίτημα και είναι καθήκον του αντικειμένου του εξυπηρετητή Response να μεταδώσει μία απάντηση στο χρήστη.

Η μέθοδος Write

Η μέθοδος Write είναι υπεύθυνη για την εκτύπωση της πληροφορίας στη μορφή που καταλαβαίνει ο φυλλομετρητής του χρήστη. Αυτή η πληροφορία μπορεί να είναι το περιεχόμενο μιας μεταβλητής από την πλευρά του εξυπηρετητή, η επιστρεφόμενη τιμή μιας συνάρτησης ή κάποιο σταθερό κείμενο εντός εισαγωγικών.

Μπορούμε να προσθέσουμε ετικέτες HTML, για τη μορφοποίηση του κειμένου, όπως φαίνεται παρακάτω.

```
<% Response.Write "<h3>Hello there!</h3>" %>
```

Η ιδιότητα Buffer

Η ιδιότητα αυτή καθορίζει αν τα αποτελέσματα από τον εξυπηρετητή θα αποθηκευτούν προσωρινά στην ενδιάμεση μνήμη (buffer), μέχρι να ολοκληρωθεί η επεξεργασία. Όταν τα αποτελέσματα αποθηκευτούν, ο εξυπηρετητής θα κρατήσει την απόκριση προς τον φυλλομετρητή μέχρι να επεξεργαστούν όλα τα σενάρια στην πλευρά του εξυπηρετητή.

Για την μη αποθήκευση των αποτελεσμάτων, χρησιμοποιούμε τη δήλωση:

```
<% Response.Buffer = False %>.
```

Αντικείμενο Request

Το αντικείμενο Request περιέχει όλα τα δεδομένα που στέλνονται στον εξυπηρετητή όταν ένας φυλλομετρητής κάνει την αίτηση.

Συλλογή Form

Όταν κάποιος χρήστης στέλνει κάποιο αίτημα από μία φόρμα και θέσει το χαρακτηριστικό method ίσο με POST, τα δεδομένα που έχουν σταλεί μαζί με τη φόρμα αποθηκεύονται στη συλλογή φόρμας. Για παράδειγμα, αν έχουμε κάποια φόρμα με ένα πεδίο κειμένου με το όνομα Surname, θα μπορούσαμε να πάρουμε την πληροφορία με αυτό τον τρόπο:

```
<% Request.Form ("Surname") %>
```

Συλλογή QueryString

Ένας άλλος τρόπος για να στείλουμε πληροφορία στον εξυπηρετητή είναι με τη χρήση του QueryString. Το QueryString είναι πληροφορία που έχει προσαρτηθεί σε ένα URL με ένα ερωτηματικό «?».Το QueryString είναι στη μορφή ενός ζεύγους όνομα/τιμή. Π.χ.

```
<%Request.QueryString ("email") %>
```

Αν μία φόρμα έχει σταλεί με τη μέθοδο GET, η πληροφορία έχει μεταδοθεί με την προσάρτηση της στο URL, όπως φαίνεται στο παρακάτω παράδειγμα URL.
Π.χ. http://somepath/test.asp?username=myid&email=myemail@server.com

Αντικείμενο Server

Το αντικείμενο Server παρέχει πρόσβαση σε μεθόδους και ιδιότητες στον εξυπηρετητή.

Η μέθοδος CreateObject

Αυτή η μέθοδος δημιουργεί ένα στιγμιότυπο ενός αντικειμένου. Η επιστρεφόμενη αναφορά μπορεί να χρησιμοποιηθεί στον κώδικα.

Η μέθοδος Transfer

Η μέθοδος Transfer σταματάει την εκτέλεση της τρέχουσας σελίδας και μεταφέρει τον έλεγχο στη σελίδα που καθορίζει. Όλες οι πληροφορίες κατάστασης στην κανονική σελίδα θα σταλούν στη νέα. Η μέθοδος αυτή αποτελεί έναν αποδοτικό τρόπο για την ανακατεύθυνση σελίδων.

```
<%
      If user = true then
            Server.Transfer (“loginok.asp”)
      Else
            Server.Transfer (“error.asp”)
      End If
%>
```

Αντικείμενο Session

Το αντικείμενο Session χρησιμοποιείται για να κρατάει πληροφορία για την σύνοδο ενός χρήστη. Με τον όρο σύνοδος (session) εννοούμε τη χρονική διάρκεια επίσκεψης ένος χρήστη στην ιστοσελίδα. Ξεκινάει μόλις ο χρήστης ανοίξει κάποια ιστοσελίδα .asp και συνεχίζει όσο ο χρήστης πηγαίνει από ιστοσελίδα σε σελίδα μέσα στον δικτυακό τόπο. Οι μεταβλητές Session μπορούν να αποθηκεύσουν πληροφορία που θα είναι διαθέσιμη σε κάθε ιστοσελίδα της συνεδρίασης και είναι προσωπική για τον χρήστη. Χρησιμοποιούνται συχνά για την παρακολούθηση των κινήσεων του χρήστη.

Η γενική σύνταξη είναι:

```
<% Session (“variableName”) = aValue %>
```

Το αντικείμενο Session έχει δύο γεγονότα: OnStart και OnEnd. Το Session_OnStart γεγονός καλείται κάθε φορά που κάποιος νέος χρήστης ξεκινάει μία συνεδρίαση και το Session_onEnd καλείται όταν τερματίσει μία συνεδρίαση. Μπορούμε να δώσουμε αρχική τιμή σε μία μεταβλητή συνεδρίασης κατά το Session_OnStart.

Αντικείμενο Application

Σκοπός του αντικειμένου Application είναι η αποθήκευση πληροφοριών που μπορούν να μοιραστούν από όλους τους χρήστες μιας εφαρμογής ταυτόχρονα. Το αντικείμενο αυτό αποθηκεύει σφαιρική πληροφορία και υπάρχει μόνο ένα τέτοιο αντικείμενο για κάθε εφαρμογή στον εξυπηρετητή ιστού. Και εδώ μπορούμε να δώσουμε αρχική τιμή σε μεταβλητές εφαρμογής μέσα στο γεγονός Application_OnStart και ότι βρίσκεται μέσα στο γεγονός Application_OnEnd θα εκτελεστεί όταν τερματίσει η εφαρμογή.
Το παρακάτω παράδειγμα αποθηκεύει την τιμή 25 σε ένα αντικείμενο τύπου apllication.

```
Application("num") = 25
```

Μετά μπορούμε να αναφερθούμε στο αντικείμενο του προηγούμενου παραδείγματος ως εξής:

```
Application("NEW_num") = Application("num") + 1
```

5.4.7 SSI

Είναι δυνατό να εισάγουμε το περιεχόμενο ενός άλλου αρχείου σε μία σελίδα ASP πριν ο εξυπηρετητής την εκτελέσει, με την ψευδοεντολή #INCLUDE. Όταν η ASP δει το #INCLUDE, τα περιεχόμενα του αρχείου θα τοποθετηθούν μέσα στο αρχείο ASP στη θέση του #INCLUDE. Έτσι τα περιεχόμενα των SSI (Server Side Includes) συμπεριφέρονται σαν να ήταν μέσα στο ίδιο το αρχείο ASP. Η συνηθισμένη χρήση του SSI είναι η αποθήκευση συναρτήσεων, επικεφαλίδων και στοιχείων που θα χρησιμοποιηθούν σε πολλαπλές σελίδες.
Για να εισάγουμε τα περιεχόμενα ενός αρχείου που βρίσκεται στον ίδιο κατάλογο με αυτόν που έχουμε αποθηκεύσει το αρχείο asp, γράφουμε :

```
<!--#include file="file.txt" -->
```

Εάν το αρχείο που θέλουμε να εισάγουμε βρίσκεται σε διαφορετικό κατάλογο, τότε γράφουμε:

```
<!--#include virtual="/somedir/file.txt" -->
```

Εάν θέλουμε να εισάγουμε τα περιεχόμενα ενός αρχείου που βρίσκεται στον αρχικό κατάλογο /root, τότε γράφουμε:

```
<!--#include virtual="/file.txt" -->
```

5.5 Ερωτήσεις – Ασκήσεις – Θέματα για ανάπτυξη

PERL	
Ποιο (ή ποια) αποτελεί σωστό όνομα για μία αριθμητική μεταβλητή;	1. $hello 2. $_test 3. $trala_la_la_la_la_la_ 4. $fries&gravy 5. $96tears 6. $tea_for_2
Ποιες λειτουργίες έχουν αυτοί οι τελεστές;	1. && 2. & 3. ^ 4. ne 5. .
Ποια η τιμή των πράξεων;	1. 17 * 2 ** 3 / 9 % 2 << 2 2. 0 && (171567 * 98275 / 1174.5 ** 4) 3. 1171 ^ 904 4. "abc" . "de" x 2
Υποθέστε ότι έχουν γίνει οι παρακάτω αναθέσεις τιμών : @list = (1, 2, 3); $scalar1 = "hello"; $scalar2 = "there"; Τι είναι αποθηκευμένο στην μεταβλητή @newlist σε καθεμία περίπτωση:	1. @newlist = @list; 2. @newlist = reverse(@list[1,2]); 3. @newlist = ($scalar1, @list[1,1]); 4. ($dummy, @newlist) = @list; 5. @newlist[2,1,3] = @list[1,2,1]; 6. @newlist = <STDIN>;
Μπορώ να γράψω σε ένα αρχείο και να το διαβάσω αργότερα;	Α. Ναι αλλά όχι ταυτόχρονα. Για να διαβάσεις ένα αρχείο στο οποίο έχει γράψει κάτι θα πρέπει να το κλείσεις πρώτα και μετά να το ξανανοίξεις για διάβασμα.

Πόσες φορές επαναλαμβάνεται ο κάθε βρόχος;	1. for ($count = 0; $count < 7; $count++) { print ("$count\n"); }
	2. $count = 1;
	do {
	print ("$count\n");
	} until ($count++ > 10);
	3. for ($count = 1; $count <= 10; $count++) {
	last if ($count == 5);
	}

PHP	
Ποιο/ ποια από τα παρακάτω εισάγει κώδικα PHP ;	1. <? 2. <% 3. και τα δύο 4. κανένα
Με ποιο/ποια γλώσσα προγραμματισμού από τις παρακάτω σχετίζεται η PHP;	1. ASP 2. perl

	3. HTML 4. XML
Τι σημαίνει PHP ;	1. personal homepage tool 2. hypertext pre-processor 3. programming hypertext pages 4. personal 5. hypertext programming
Πώς μπορείτε να εμφανίσετε ένα κείμενο στο κάποιο έγγραφο;	1. print '….'; 2. echo '…'; 3. και με τα δύο 4. με κανένα από τα δύο
Πως δημιουργείται μία array μεταβλητή ;	1. var = array {'value1','value2'}; 2. var = array ['value1','value2']; 3. var = array ('value1','value2'); 4. var = array 'value1','value2';
Με ποια βάση δεδομένων η PHP συνεργάζεται βέλτιστα;	1. MS SQL 2. MySQL 3. MS Access 4. Oracle
Πώς συγκρίνουμε τις τιμές δύο μεταβλητών;	1. $x == $y; 2. $x = $y; 3. $x eguals $y 4. $x is $y
Ποιο από τα παρακάτω δεν πραγματοποιείται με κώδικα PHP;	1. Η διαγραφή αρχείων από τον εξυπηρετητή 2. Η ανακατεύθυνση σε άλλη ιστοσελίδα 3. Η αλλαγή τις τιμής στην status bar του φυλλομετρητή 4. Η ανάγνωση της IP διεύθυνσης του επισκέπτη σε μία ιστοσελίδα..
Με ποιο σύμβολο ορίζουμε μία μεταβλητή;	1. %var 2. $var 3. &var 4. ?var
ASP	
Τι σημαίνει ASP;	1. Active Server Pages 2. Active Standard Pages 3. A Server Page 4. All Standard Page
Πώς εισάγουμε ASP scripts;	1. <script>...</script> 2. <%>...</%> 3. <%...%> 4. <&>...</&>
Πώς θα γράφατε "Το πρώτο μο υ απλό ASP script" σε ASP;	1. Response.Write("Το πρώτο μου απλό ASP script ") 2. Document.Write("Το πρώτο μου απλό ASP script ") 3. "Το πρώτο μου απλό ASP script"
Το "<%=" είναι το ίδιο με:	1. <%Equal

	2. <%Response.Write 3. <%Write 4. <%Document.Write
Η ASP βασίζεται σε:	1. PERL 2. JavaScript 3. EcmaScript 4. VBScript
Πώς θα δηλώσουμε ότι ο κώδικας ASP θα είναι γραμμένος σε JavaScript;	1. Θα ξεκινήσουμε γράφοντας: <% language="javascript" %> 2. Η JavaScript είναι η προεπιλεγμένη γλώσσα προγραμματισμού 3. Θα ξεκινήσουμε γράφοντας: <%@ language="javascript" %> 5. Θα τελειώσουμε γράφοντας: <% language="javascript" %>
Πώς μπορούμε να «πάρουμε» τα δεδομένα από μία φόρμα που έχει σταλεί με την μέθοδο "get";	1. Request.QueryString 2. Request.Form
Πώς μπορούμε να «πάρουμε» τα δεδομένα από μία φόρμα που έχει σταλεί με την μέθοδο "post";	1. Request.QueryString 2. Request.Form
Η σελίδα 1 έχει τον σύνδεσμο Go. Πώς μπορώ από μία άλλη σελίδα να πάρω την τιμή της παραμέτρου “color”;	1. Request.QueryString("color") 2. Get("color") 3. Response.QueryString("color") 4. Response.Parameter("color")
Ποια ASP ιδιότητα χρησιμοποιείται για να χαρακτηρίσει ένα χρήστη;	1. An ASP Cookie 2. The Server object 3. The Application object
Όλοι οι χρήστες της ίδια εφαρμογής μοιράζονται ΕΝΑ αντικείμενο session;	1. Αληθής 2. Ψευδής
Όλοι οι χρήστες της ίδια εφαρμογής μοιράζονται ΕΝΑ αντικείμενο Application;	1. Αληθής 2. Ψευδής

Ερωτήσεις Ανάπτυξης

1. Πότε τερματίζεται ένας βρόχος while;
2. Πότε μία δήλωση until τερματίζει έναν βρόχο;
3. Τι κάνει ο τελεστής == ;
4. Ποιο είναι το αποτέλεσμα της παρακάτω δήλωσης ;14 + 6 * 3 - 10 / 2

Ασκήσεις

1. Φτιάξτε ένα πρόγραμμα σε PERL που διαβάζει 2 αριθμούς από μία φόρμα HTML, τους προσθέτει και εκτυπώνει το αποτέλεσμα τους.
2. Γράψτε ένα πρόγραμμα σε PERL που διαβάζει δύο αριθμούς από μία φόρμα HTML, διαιρεί τον πρώτο με τον δεύτερο και εκτυπώνει το αποτέλεσμα, αλλά:
   Εκτυπώνει μήνυμα λάθους εάν ο δεύτερος αριθμός είναι 0.

Εάν ο πρώτος αριθμός είναι 0 και ο δεύτερος 1 τότε απλά εκτυπώνει τον πρώτο, αφού δεν χρειάζεται να κάνει κάποια διαίρεση.

3. Γράψτε ένα πρόγραμμα που χρησιμοποιεί τον βρόχο while και εκτυπώνει τους αριθμούς από το 1 έως το 10 σε φθίνουσα ταξινόμηση.
4. Γράψτε ένα πρόγραμμα που χρησιμοποιεί τον βρόχο until για να εκτυπώσει τους αριθμούς από το 1 έως το 10 σε φθίνουσα ταξινόμηση.
5. Γράψτε ένα πρόγραμμα που διαβάζει δύο αριθμούς από μία φόρμα HTML, διαιρεί τον πρώτο με τον δεύτερο και εκτυπώνει το αποτέλεσμα, και το υπόλοιπο.
6. Γράψτε ένα πρόγραμμα που μετράει όλες τις εμφανίσεις της λέξης, που δίνει ο χρήστης από μία φόρμα HTML, σε ένα αρχείο δεδομένων.
7. Γράψτε τα προγράμματα των ασκήσεων 1-6 χρησιμοποιώντας PHP
8. Γράψτε τα προγράμματα των ασκήσεων 1-6 χρησιμοποιώντας ASP