Σημειώσεις γλώσσας Pascal

Κεφάλαιο 1
Τι είναι ένα πρόγραμμα ΗΥ;

Αυτό το κεφάλαιο απευθύνεται σε αρχάριους χρήστες Αν είστε τελείως πρωτάρηδες στους υπολογιστές, θα βρείτε χρήσιμες τις πληροφορίες στο κεφάλαιο αυτό. Αν όμως, έχετε κάποια εμπειρία στον προγραμματισμό, μπορείτε να το παραβλέψετε μια και αναφέρεται σε βασικά στοιχεία της Pascal.

Τι είναι ένα πρόγραμμα ΗΥ; Ο υπολογιστής (computer) δεν είναι τίποτα άλλο από μια μηχανή, που έχει τη δυνατότητα να εκτελεί μαθηματικές πράξεις πολύ γρήγορα και με μεγάλη ακρίβεια, αλλά δεν μπορεί να κάνει τίποτα αν δεν εκτελεστεί κάποιο πρόγραμμα γραμμένο από κάποιον άνθρωπο. Επιπλέον, αν κάποιος άνθρωπος γράψει ένα πρόγραμμα το οποίο μετατρέπει τα δεδομένα σε σκουπίδια και ο ΗΥ υπάκουα και πολύ γρήγορα θα μετατρέψει τα δεδομένα σε σκουπίδια. Είναι πιθανόν να γράψει κανείς ένα πρόγραμμα με ένα πολύ μικρό λάθος, και να έχει την εντύπωση ότι το πρόγραμμα του δημιουργεί τα σωστά αποτελέσματα. Είναι στην κρίση του προγραμματιστή να σχεδιάσει ένα πρόγραμμα ώστε αυτό να επιτυγχάνει τα επιθυμητά αποτελέσματα. Ένα πρόγραμμα είναι απλά μια συνταγή που εφαρμόζει ο υπολογιστής πάνω στα δεδομένα ώστε να αποκομίσει τα επιθυμητά αποτελέσματα. Είναι όπως περίπου η συνταγή ψησίματος ενός κέικ. Τα δεδομένα αντιστοιχούν στα συστατικά, συμπεριλαμβάνοντας και την θερμότητα που παρέχεται από την ηλεκτρική κουζίνα. Το πρόγραμμα αντιστοιχεί στις οδηγίες της συνταγής για ανακάτεμα, αναμονή, θερμότητα, ψύξη και ότι άλλο μπορεί να επιτελεστεί επί των συστατικών. Η έξοδος του προγράμματος μπορεί να αντιστοιχισθεί με το τελικό κέικ, έτοιμο για σερβίρισμα. Έτσι ένα πρόγραμμα μπορούμε να πούμε ότι συνίσταται από δυο μέρη, τα δεδομένα σύμφωνα με τα οποία αυτό ενεργεί και τον κώδικα (πρόγραμμα) που διαχειρίζεται τα δεδομένα. Τα δεδομένα και το πρόγραμμα είναι αχώριστα όπως παρουσιάστηκε από τα παραπάνω.

Τι είναι οι σταθερές ; Σχεδόν κάθε πρόγραμμα απαιτεί μερικούς αριθμούς που δεν αλλάζουν ποτέ κατά την εκτέλεσή του. Αυτοί οι αριθμοί μπορούν να καθοριστούν μια φορά και να χρησιμοποιούνται όσο συχνά επιθυμούμε κατά την διάρκεια λειτουργίας του προγράμματος. Για να επιστρέψουμε στην αντιστοίχηση ενός προγράμματος με μια συνταγή, αν έχετε καθορίσει το μέγεθος του κουταλιού της σούπας, μπορείτε να το χρησιμοποιήσετε χωρίς να ελέγχετε την χωρητικότητά του. Κάθε φορά

όταν γράφετε ένα πρόγραμμα μπορείτε να καθορίσετε την τιμή του π=3.141592 και να συνεχίσετε να την χρησιμοποιείτε όταν χρειάζεται, ξέροντας ότι είναι διαθέσιμη και σωστή.

Τι είναι μεταβλητές ; Εκτός από τις σταθερές, σχεδόν κάθε πρόγραμμα χρησιμοποιεί κάποιους αριθμούς, των οποίων η τιμή αλλάζει κατά την διάρκεια του προγράμματος. Μπορούν να καθοριστούν ως μεταβλητές και μετά να αποκτήσουν όποια τιμή επιθυμούμε, που έχει βέβαια σχέση με το πρόγραμμα. Ένα παράδειγμα θα μπορούσε να είναι ο αριθμός των αυγών στην συνταγή μας. Αν ένα στρώμα κέικ απαιτεί 2 αυγά, 3 στρώματα απαιτούν 6 αυγά. Ο αριθμός των αυγών λοιπόν θα μπορούσε να είναι μια μεταβλητή.

Πως καθορίζουμε σταθερές και μεταβλητές ; Όλες οι σταθερές και οι μεταβλητές έχουν ένα όνομα και μια τιμή. Στο τελευταίο παράδειγμα, το όνομα της μεταβλητής είναι "eggs", και έχει τιμή 2 ή 6 ανάλογα με το πότε ελέγξαμε στα αποθηκευμένα δεδομένα. Σε ένα πρόγραμμα στις σταθερές και στις μεταβλητές δίνονται ονόματα σχετικά με το θέμα, και μπορούν να αποθηκεύσουν οποιαδήποτε τιμή μέσα στα επιτρεπόμενα όρια τα οποία εμείς καθορίζουμε.

Τι είναι τόσο καλό με την Pascal ; Μερικές γλώσσες προγραμματισμού επιτρέπουν στον προγραμματιστή να καθορίσει τις σταθερές και τις μεταβλητές με έναν τυχαίο τρόπο έκφρασης και μετά να συνδυάζουν δεδομένα με έναν ακόμα πιο τυχαίο τρόπο. Για παράδειγμα, αν αναμειγνύατε τον αριθμό των αυγών, στην παραπάνω συνταγή, με τον αριθμό των φλιτζανιών από αλεύρι, θα το κάνατε με μια έγκυρη μαθηματική πρόσθεση, αλλά θα είχατε ένα τελείως άχρηστο αποτέλεσμα. Μερικές γλώσσες προγραμματισμού σας επιτρέπουν να επιτελέσετε μια τέτοια πράξη και να εμφανίσετε το άχρηστο παραγόμενο αποτέλεσμα. Επειδή όμως η Pascal απαιτεί από σας να καθορίσετε τις σταθερές και τις μεταβλητές σας με έναν ακριβή τρόπο, η πιθανότητα μια τέτοιας ανώφελης πράξης ελαχιστοποιείται. Ένα καλογραμμένο πρόγραμμα σε Pascal έχει πολλά σημεία ελέγχου ώστε να αποφεύγεται το ενδεχόμενο μιας απολύτως μπερδεμένης και χωρίς ουσία εκτύπωσης. Σημειώστε πάντως, όσον αφορά τα προηγούμενα, πως "καλογραμμένο πρόγραμμα" σε Pascal είναι ένα θέμα προς συζήτηση. Είναι στο χέρι του προγραμματιστή να καθορίσει τη δομή των δεδομένων με ένα τέτοιο τρόπο ώστε το πρόγραμμα να μπορεί να αποτρέψει την "παραγωγή" σκουπιδιών. Σε τελευταία ανάλυση, το πρόγραμμα δεν μπορεί να είναι καλύτερο από την ανάλυση που έχει προηγηθεί για αυτό το θέμα. Αν είστε ένας αρχάριος προγραμματιστής,

μην τρομοκρατηθείτε με τίποτα από τα παραπάνω. Η Pascal είναι ορθά σχεδιασμένη, χρήσιμο εργαλείο που χρησιμοποιήθηκε με επιτυχία από πολλούς αρχάριους χρήστες και επαγγελματίες. Με αυτές τις λίγες προειδοποιήσεις, είστε έτοιμοι να ξεκινήσετε.

Κεφάλαιο 2
Ξεκινώντας με την PASCAL

Το πρώτο σας πρόγραμμα σε Pascal Ας ξεκινήσουμε με ένα πρόγραμμα το οποίο δεν κάνει τίποτα, αλλά είναι ένα παράδειγμα του πιο συνηθισμένου προγράμματος Pascal. Φορτώστε την Turbo Pascal, στη συνέχεια φορτώστε το αρχείο **EX01_2.PAS** στο περιβάλλον εργασίας, ως αρχείο προς επεξεργασία. Αυτό είναι και μια ένδειξη ότι ξέρετε να χειριστείτε το περιβάλλον εργασίας.  Πρόγραμμα ..... **EX01_2.PAS** Θα πρέπει να έχετε τώρα ανοικτό στην οθόνη σας το πιο συνηθισμένο πρόγραμμα της Pascal, και μπορούμε να ρίξουμε μια ματιά για να ξεκαθαρίσουμε τι ακριβώς κάνει. Η πρώτη γραμμή απαιτείται στις δηλώσεις της Standard Pascal και είναι το όνομα του προγράμματος, το οποίο μπορεί να είναι όποιο όνομα σας αρέσει, αρκεί βέβαια να πληρεί τους όρους τους όποιους πρέπει να πληρεί ένα αναγνωριστικό (**indentifier**) όπως θα δούμε στην επόμενη παράγραφο. Δεν πρέπει να περιέχει κενούς χαρακτήρες, διαφορετικά θα ελαμβάνετο ως δυο λέξεις και θα μπέρδευε τον compiler. Η πρώτη λέξη **Program** είναι η πρώτη από τις δεσμευμένες λέξεις που αναφέρθηκαν νωρίτερα και είναι η ένδειξη προς τον compiler της Pascal ότι το επόμενο όνομα είναι το όνομα του προγράμματος. Επισημαίνουμε ότι το όνομα τελειώνει με ένα ερωτηματικό. Η Pascal χρησιμοποιεί το σύμβολο αυτό ως διαχωριστή εκφράσεων (**statements**) και παρότι όλες οι εκφράσεις δεν καταλήγουν με ερωτηματικό, οι περισσότερες το κάνουν. Η σωστή χρήση του ερωτηματικού θα ξεκαθαριστεί αργότερα στο μυαλό σας. Η Turbo Pascal δεν απαιτεί την έκφραση **program**, αλλά για να διατηρηθεί συμβατή με την Standard Pascal, απλά θα αγνοήσει την όλη έκφραση. Προτείνεται να συμπεριλάβετε την έκφραση αυτή, για να διατηρείτε τη συμβατότητα με τη Standard Pascal, και για να δείξετε μέσω του ονόματος του προγράμματος το σκοπό του προγράμματος.

Τι είναι ένα αναγνωριστικό ; Όλα τα αναγνωριστικά, συμπεριλαμβανομένου και του ονόματος του προγράμματος, των ονομάτων των συναρτήσεων και των διαδικασιών καθώς και τα ονόματα των σταθερών και των μεταβλητών θα ξεκινούν με ένα γράμμα και θα είναι μια σύνθεση από οποιοδήποτε συνδυασμό γράμματος και ψηφίου, χωρίς ενδιάμεσα κενά. Τα πεζά ή τα κεφαλαία γράμματα δεν είναι διαφορετικά και μπορούν να χρησιμοποιούνται κατά βούληση (αν αυτός ο ορισμός σας φαίνεται δύσκολος μην ανησυχείτε, θα διαλευκανθεί

αργότερα, απλά έπρεπε να αναφερθεί νωρίς). Ο βασικός ορισμός της Pascal απαιτεί ότι οποιαδήποτε υλοποίηση (π.χ. ένας compiler που σχεδιάστηκε από μια εταιρεία) πρέπει να χρησιμοποιεί τουλάχιστον 8 χαρακτήρες από το αναγνωριστικό ως σημαντικούς και να αγνοεί τους υπόλοιπους χαρακτήρες, αν χρησιμοποιούνται περισσότεροι από 8. Οι περισσότερες υλοποιήσεις χρησιμοποιούν πολύ περισσότερους από 8 χαρακτήρες. Όλες οι εκδόσεις της Turbo Pascal επιτρέπουν να χρησιμοποιηθούν μέχρι 63 χαρακτήρες για ένα αναγνωριστικό. Η Standard Pascal δεν επιτρέπει τη χρήση της κάτω παύλας σε ένα αναγνωριστικό αλλά οι περισσότερες υλοποιήσεις της Pascal επιτρέπουν τη χρήση της κάτω παύλας μετά τον πρώτο χαρακτήρα. Επειδή όλες οι εκδόσεις των Turbo Pascal Compilers επιτρέπουν την χρήση της κάτω παύλας σε ένα αναγνωριστικό, άρα θα χρησιμοποιείται ελεύθερα σε αυτό το εγχειρίδιο. Η κάτω παύλα χρησιμοποιείται στο πρόγραμμα με όνομα "**Puppy_Dog**" που θα έπρεπε να βρίσκεται στην οθόνη σας τώρα. Επιστρέφοντας στο πρόγραμμα που έχουμε ως παράδειγμα, η γραμμή 2 είναι κενή και αγνοείται από όλους τους compilers της Pascal. Περισσότερα για την κενή γραμμή θα πούμε στο τέλος αυτού του κεφαλαίου.

Τώρα για το πρόγραμμα Οι γραμμές 3 και 4 αποτελούν το βασικό πρόγραμμα Pascal, το οποίο σε αυτήν την περίπτωση δεν κάνει απολύτως τίποτα. Αυτό είναι μια εικόνα του πιο μικρού προγράμματος Pascal. Οι δυο λέξεις **begin** και **end** είναι οι δυο επόμενες λέξεις που θα μας απασχολήσουν. Οποιαδήποτε λογική ομαδοποίηση κώδικα Pascal μπορεί να απομονωθεί συμβολίζοντάς το με τις λέξεις **begin** και **end**. Θα χρησιμοποιείτε αυτή την κατασκευή συνεχώς καθώς θα γράφετε κώδικα, επομένως καλό είναι να τη μάθετε πολύ καλά. Κώδικας που θα εκτελείται υπό συνθήκες θα εντάσσεται μέσα σε **begin** και **end**, όπως επίσης και κώδικας σε loop, και κώδικας που εμπεριέχεται σε διαδικασίες (**subroutines**-αν και λέγονται procedures στην Pascal), καθώς και σε άλλες περιπτώσεις. Στο συγκεκριμένο πρόγραμμα, τα **begin** και **end** χρησιμοποιούνται για να καθορίσουν το κυρίως πρόγραμμα και κάθε πρόγραμμα Pascal θα έχει το κυρίως πρόγραμμα σ' αυτή τη θέση. Επειδή δεν θα επιτελεστεί καμία ενέργεια σε αυτό το πρόγραμμα δεν υπάρχει και καμία σχέση ανάμεσα στις δεσμευμένες λέξεις **begin** και στο **end**. Τέλος, υπάρχει και κάποιο άλλο πολύ σημαντικό σημείο αυτού του προγράμματος: η τελεία που ακολουθεί το **end**. Η τελεία αυτή είναι σημάδι προς τον compiler ότι τελείωσε το πρόγραμμα του εκτελέσιμου μέρους και επίσης ότι τελείωσε η μεταγλώττιση. Κάθε πρόγραμμα Pascal έχει μια και μόνη τελεία

η οποία θα βρίσκεται στο τέλος του προγράμματος. Σκεφτείτε ένα πρόγραμμα Pascal ως μια πρόταση με μια τελεία στο τέλος. Αγνοείστε τις γραμμές 6 και 10 για λίγα λεπτά και θα τις περιγράψουμε εκτενέστερα αργότερα. Αυτή ήταν η περιγραφή του πρώτου μας προγράμματος. Τώρα είναι καιρός να το μεταγλωττίσετε και να το τρέξετε. Πατήστε <Alt>+R για να μεταγλωττίσετε και να εκτελέσετε το πρόγραμμα και μετά <Alt>+F5 για να δείτε τα αποτελέσματα της εκτέλεσης. Βέβαια αυτό το πρόγραμμα δεν κάνει τίποτα. Ας δούμε λοιπόν ένα που να κάνει κάτι. ***Ένα πρόγραμμα που να κάνει κάτι*** Φορτώστε το πρόγραμμα **EX02_2.PAS** και κοιτάξτε το στην οθόνη σας.

Πρόγραμμα ..... **EX02_2.PAS** Το όνομα του προγράμματος είναι **Kitty_cat**. Πάλι υπάρχουν τα **begin** και **end** για να καθορίσουμε την περιοχή του κυρίως προγράμματος και στο τέλος η τελεία. Όμως, έχουμε και δυο εκφράσεις ανάμεσα στο **begin** και στο **end**. Η **Writeln** είναι μια ειδική λέξη και μάλλον δεν είναι έκπληξη ότι σημαίνει να γράψει μια γραμμή από δεδομένα κάπου. Χωρίς τον τροποποιητή (**modifier**), ο οποίος θα εξηγηθεί επακριβώς αργότερα, θα γράψει στην αρχικά ορισμένη συσκευή, που στην περίπτωση του PC είναι η οθόνη. Τα δεδομένα μέσα στην παρένθεση είναι τα δεδομένα που θα εκτυπωθούν, και παρότι υπάρχουν πολλά ήδη δεδομένων που θα επιθυμούσαμε να εκτυπώσουμε, θα περιοριστούμε στο πιο απλό παράδειγμα για την ώρα. Οποιαδήποτε πληροφορία ανάμεσα στις αποστρόφους θα εκτυπωθεί ως απλό κείμενο. Η ειδική λέξη **Writeln** δεν είναι δεσμευμένη λέξη αλλά είναι καθορισμένη από το σύστημα να κάνει μια ειδική δουλειά για σας, δηλαδή να εκτυπώνει μια γραμμή από δεδομένα στην οθόνη. Είναι στην πραγματικότητα, μια διαδικασία που παρέχεται για σας από τους συγγραφείς της Turbo Pascal σαν προγραμματιστική βοήθεια προς εσάς. Μπορείτε, αν το επιθυμείτε, να χρησιμοποιήσετε αυτό το όνομα ως αναγνωριστικό για άλλο σκοπό στο πρόγραμμά σας, αλλά αν το κάνετε αυτό δεν θα μπορείτε να χρησιμοποιήσετε την βασική διαδικασία για εμφάνιση δεδομένων. Σε αυτήν την περίπτωση, θα πρέπει να φροντίσετε μόνοι σας να εκτυπώσετε τα δεδομένα σας. Σημειώστε προσεκτικά ότι ορισμένες λέξεις είναι δεσμευμένες και δεν μπορούν να επανακαθοριστούν και να χρησιμοποιηθούν για κάποιον άλλο σκοπό ενώ κάποιες άλλες είναι ειδικές μια και μπορούν να επανακαθοριστούν. Προφανώς δεν θα θέλετε να επανακαθορίσετε κάποιες από τις ειδικές λέξεις. Μέχρι να αποκτήσετε

σημαντική προγραμματιστική εμπειρία απλά χρησιμοποιείστε τις ως εργαλεία. Σημειώστε το ερωτηματικό στο τέλος της γραμμής 4. Είναι ο διαχωριστής εκφράσεων που αναφέρθηκε προηγούμενα και λέει στον compiler της Pascal ότι η γραμμή είναι πλήρης, δηλαδή τίποτα από τα επόμενα δεν μπορεί να θεωρηθεί ως συνέχεια αυτής της έκφρασης. Η επόμενη έκφραση, στην γραμμή 5, είναι μια ξεχωριστή έκφραση που μπορεί να εκτελεστεί σειριακά ακολουθώντας την έκφραση της γραμμής 4. Αυτό το πρόγραμμα θα εκτυπώσει τις δυο γραμμές κειμένου και θα τερματιστεί. Τώρα είναι καιρός να εκτελέσουμε το πρόγραμμα. Μεταγλωττίστε το και τρέξτε το όπως κάνατε και με το πρώτο παράδειγμα. Πρέπει να βλέπετε τις δυο γραμμές κειμένου στην οθόνη κάθε φορά που τρέχετε το πρόγραμμα. Όταν βαρεθείτε να τρέχετε το πρόγραμμα **EX02_2.PAS** μπορούμε να πάμε στο άλλο παράδειγμα.

Ένα άλλο πρόγραμμα με περισσότερα στοιχεία προς εκτύπωση Παρατηρήστε το παράδειγμα με όνομα **EX03_2.PAS**.

Πρόγραμμα ..... **EX03_2.PAS** Αυτό το πρόγραμμα έχει τρεις γραμμές που θα εμφανίσουν κάτι στην οθόνη αλλά οι δυο πρώτες είναι άγνωστες επειδή χρησιμοποιείται μια άλλη ειδική λέξη, η **write**. Η **write** είναι μια άλλη διαδικασία η οποία υποχρεώνει το κείμενο να εκτυπωθεί όπως και με την **writeln**, αλλά ο κέρσορας δεν αλλάζει γραμμή στην έξοδο. Η **writeln** γράφει το προς εκτύπωση κείμενο και αλλάζει γραμμή, η **write** όχι. Το τελικό αποτέλεσμα είναι και οι τρεις γραμμές να εκτυπώνονται στην ίδια γραμμή όταν το πρόγραμμα εκτελεστεί. Προσέξτε ότι στο τέλος κάθε λέξης υπάρχει ο κενός χαρακτήρας, ώστε το τελικό αποτέλεσμα να είναι καλαίσθητο. Μεταγλωττίστε και εκτελέστε το πρόγραμμα. Τώρα ίσως πρέπει να γυρίσετε στον editor και να έχετε στην οθόνη σας το **EX03_2.PAS** και να προσθέσετε μερικές ακόμα εντολές εκτύπωσης για να πειραματιστείτε με τις εντολές **write** και **writeln**. Αν βαρεθήκατε να προσθέτετε εντολές εμφάνισης ας πάμε στο επόμενο παράδειγμα και ας δούμε πως μπορούμε να εισάγουμε σχόλια στον κώδικά μας.

Προσθέτοντας σχόλια στον πρόγραμμα Το αρχείο με όνομα **EX04_2.PAS** είναι παρόμοιο με τα προηγούμενα αλλά έχει σχόλια τα οποία έχουν προστεθεί για μια επίδειξη της χρήσης τους.

Πρόγραμμα ..... **EX04_2.PAS**

Η Pascal καθορίζει τα σχόλια ως οτιδήποτε ανάμεσα στα σύμβολα (* και *) ή στα σύμβολα { και }. Αρχικά μόνο το δεύτερο ζευγάρι είχε καθοριστεί αλλά επειδή πολλά πληκτρολόγια δεν είχαν τα άγκιστρα χρησιμοποιήθηκαν και οι παρενθέσεις σε συνδυασμό με το * και έτσι έχουμε δυο τύπους συμβόλων για σχόλια. Τα περισσότερα από τα σχόλια έχουν σκοπό την επεξήγηση μέσα στον κώδικα. Επίσης τα σχόλια μπορούν να απλώνονται σε δυο γραμμές συνεχόμενες, οι γραμμές 11 και 12 που φυσιολογικά θα εκτύπωναν το "**send money**", δεν είναι κώδικας Pascal αλλά σχόλια. Προσπαθήστε να μεταγλωττίσετε και να τρέξετε το πρόγραμμα, μετά σβήστε τα σύμβολα για τα σχόλια και παρατηρήστε την εκτύπωση "**send money**". Θα πρέπει να παρατηρήσουμε το εξής. Αν και κάποιοι compilers επιτρέπουν να υπάρχει κάποιο σχόλιο με τη μορφή (* } ή κάποιοι άλλοι με τη μορφή { *), είναι πολύ κακό προγραμματιστικά και πρέπει να αποφεύγεται. Το πρωτόκολλο **ANSI Pascal** επιτρέπει τέτοια χρήση αλλά η Turbo Pascal δεν επιτρέπει αυτή την αστεία χρήση των σχολίων. Η Turbo Pascal δεν σας επιτρέπει να ενσωματώσετε σχόλια χρησιμοποιώντας τους ίδιους διαχωριστές, αλλά επιτρέπει να ενσωματώνετε τον έναν τύπο μέσα στον άλλο. Αυτό μπορεί να χρησιμοποιηθεί ως βοήθεια αποσφαλμάτωσης. Αν γενικά χρησιμοποιείτε τα σύμβολα (* και *) για σχόλια, μάλλον θα πρέπει να χρησιμοποιείτε τα { και } στην Turbo Pascal για να μετατρέψετε σε σχόλιο ένα ολόκληρο κομμάτι κώδικα κατά τη διάρκεια της αποσφαλμάτωσης, ακόμα και αν περιέχει και άλλα σχόλια μέσα του. Αυτό είναι ένα τέχνασμα που πρέπει να θυμάστε όταν φθάσετε στο σημείο να γράψετε προγράμματα σημαντικού μεγέθους. Όταν έχετε επιτυχώς τροποποιήσει και εκτελέσει το πρόγραμμα με τα σχόλια, θα συνεχίσουμε για να εξηγήσουμε την πρακτική του format και πως η Pascal πραγματικά ψάχνει διαμέσου του κώδικά σας για τις εκτελέσιμες εκφράσεις. Πρέπει να αναφερθεί ότι το πρόγραμμα με όνομα **EX04_2.PAS** δεν περιλαμβάνει καλό στυλ σχολίων. Το πρόγραμμα πρέπει να εξηγεί που τα σχόλια μπορούν να χρησιμοποιηθούν και μοιάζει πολύ κομματιαστό και ανοργάνωτο. Περισσότερα παραδείγματα θα σας δείξουν καλύτερα παραδείγματα χρήσης σχολίων, όπως και η εμπειρία σας μέσα από αυτό το σύγγραμμα.

Τα αποτελέσματα από το στάδιο της εκτέλεσης Θα πρέπει τώρα να είστε ικανοί να διακρίνετε το σκοπό των γραμμών 17 έως και 23 του προγράμματος αυτού. Κάθε ένα από τα προγράμματα -παραδείγματα σε αυτό το εγχειρίδιο περιέχουν τα αποτελέσματα από την εκτέλεση του αντίστοιχου προγράμματος στο τέλος του κώδικα. Αυτό επιτρέπει τη

μελέτη του εγχειριδίου με διαφορετικούς τρόπους, όπως με τη βοήθεια ενός browser, όπως κάνετε τώρα ή με την εκτύπωση του κειμένου και των αντίστοιχων παραδειγμάτων. Με αυτό το κείμενο και ένα αντίγραφο από τα προγράμματα περιλαμβανομένων και των αποτελεσμάτων της εκτέλεσης, δεν χρειάζεστε πρόσβαση σε υπολογιστή για να μελετήσετε. Βεβαίως χρειάζεστε πρόσβαση στον υπολογιστή για να δείτε, να μεταγλωττίσετε και να εκτελέσετε τα προγράμματα, πράγμα το οποίο με όλη μου την καρδία σας προτείνω να κάνετε.

Εξάσκηση καλής μορφοποίησης Μελετήστε το **EX05_2.PAS** για να δείτε ένα παράδειγμα καλού στυλ προγραμματισμού.

Πρόγραμμα ..... **EX05_2.PAS** Είναι σημαντικό να σημειώσουμε ότι η Pascal δεν σας καθοδηγεί που πρέπει να τοποθετείτε το **carriage return** (**Enter**) ή πόσα κενά να εισάγετε μεταξύ δεσμευμένων λέξεων, ειδικών λέξεων κτλ.. Η Pascal χρησιμοποιεί το συνδυασμό δεσμευμένων λέξεων και παρουσιάσει τα ερωτηματικά για να καθορίσει τη λογική δομή ενός προγράμματος. Μέχρι τώρα έχουμε μόνο δυο εκτελέσιμες εκφράσεις, τις οποίες χρησιμοποίησα για να φτιάξω ένα πρόγραμμα αρκετά κατανοητό. Μεταγλωττίστε και εκτελέστε το πρόγραμμα για να δείτε ότι πραγματικά κάνει αυτό που είχατε αρχικά σκεφτεί.

Εξάσκηση μη καλής μορφοποίησης Μελετήστε το **EX06_2.PAS** για να δείτε ένα παράδειγμα φρικτού στυλ προγραμματισμού.

Πρόγραμμα ..... **EX06_2.PAS** Δεν είναι ορατό αυτό με μια ματιά αλλά το πρόγραμμα που κοιτάτε είναι ακριβώς ίδιο με το προηγούμενο. Η Pascal δεν ενδιαφέρεται ποιο ακριβώς ζητάτε να τρέξετε γιατί γι' αυτήν είναι ακριβώς τα ίδια. Για σας είναι διαφορετικά, και το δεύτερο θα ήταν ένα χάλι αν επιχειρούσατε να δείτε τι κάνει κάποια στιγμή στο μέλλον. Το **EX06_2.PAS** είναι ένα καλό παράδειγμα για να σας δείξει ότι η Pascal δεν νοιάζεται για το στυλ πάνω στην οθόνη. Νοιάζεται μόνο για τη δομή, συμπεριλαμβανομένων και των δεσμευμένων λέξεων, των διαχωριστών όπως τα κενά και τα ερωτηματικά. Τα **Ente**r λαμβάνονται μόνο ως παραπάνω κενά. Μπορείτε να εισάγεται απεριόριστα κενά σχεδόν παντού εκτός από τα σημεία ανάμεσα στις δεσμευμένες λέξεις ή στα ονόματα των μεταβλητών. Πρέπει να επιδείξετε κάποια προσοχή στο προγραμματιστικό στυλ αλλά μην ανησυχείτε ακόμα. Θα ήταν καλό για σας να χρησιμοποιείτε το στυλ που έχει υιοθετηθεί μέχρι τώρα στο εγχειριδιο

αυτό μέχρι να αποκτήσετε εμπειρία στον προγραμματισμό με Pascal. Όσο ο καιρός περνά θα αναπτύξετε το σωστό προγραμματιστικό στυλ. Δεν είναι μόνο η εικόνα του προγράμματος σημαντική, τα ονόματα που χρησιμοποιούνται για τις μεταβλητές μπορεί να βοηθήσουν πάρα πολύ ή καθόλου, όπως θα δούμε στο επόμενο κεφάλαιο. Μεταγλωττίστε και εκτελέστε το **EX06_2.PAS.**

Ασκήσεις προγραμματισμού

1. Γράψτε ένα πρόγραμμα που να εμφανίζει το όνομά σας στην οθόνη. **2.** Διαμορφώστε το πρόγραμμα ώστε, να εμφανίζει το όνομα και τη διεύθυνσή σας σε μια γραμμή, μετά αλλάξτε το, αντικαθιστώντας το **write** με **writeln** ώστε να εμφανίζονται τα στοιχεία αυτά σε διαφορετικές γραμμές.

Προγράμματα κεφαλαίου 2

```
  (* Κεφάλαιο 1 - Πρόγραμμα 1 EX01_2*)
1  program Puppy_Dog;
2
3  begin
4  end.
5
6  { Result of execution
7
8  (There is no output from this program )
9
10  }
```

```
  (* Κεφάλαιο 2 - Πρόγραμμα 2 EX02_2.pas *)
1  program Kitty_Cat;
2
3  begin
4      Writeln('This program');
5      Writeln('actually does something.');
6  end.
7
8  { Result of execution
```

```
9
10  This program
11   actually does something.
12
13
14   }
```

(* Κεφάλαιο 2 - Πρόγραμμα 3 EX03_2.pas *)

```
1   program Guppy_The_Fish;
2
3   begin
4      Write('This will ');
5      Write('all be ');
6      Writeln('on one line.');
7   end.
8
9
10   { Result of execution
11
12   This will all be on one line.
13
14  }
```

(* Κεφάλαιο 2 - Πρόγραμμα 4 EX04_2.pas *)

```
1   program Lots_Of_Comments;
2
3   begin { This is the start of the main program }
4
5      (* This is a comment that is ignored by the Pascal compiler *)
6      { This is also ignored }
7
8      Writeln('I am in Pascal school, Dad'); (* Comment *)
9      Writeln('All students are always broke'); {Comment}
10      (*
11        Writeln('Send money');
12        Writeln('Send money');
13       *)
14       Writeln('I am really getting hungry');
```

```
15  end. (* This is the end of the main program *)
16
17  { Result of execution
18
19  I am in Pascal school, Dad
20  All students are always broke
21  I am really getting hungry
22
23  }
```

(* Κεφάλαιο 2 - Πρόγραμμα 5 EX05_2.pas *)

```
1   program Good_Programming_Style;
2
3   begin
4
5       Write('Programming style ');
6       Write ('is a matter of ');
7       Writeln ('personal choice');
8
9        Write('Each person ');
10       Write ('can choose ');
11       Writeln ('his own style');
12
13       Write('He can be ');
14       Write ('very clear, or ');
15       Writeln ('extremely messy');
16  end.
17
18
19  { Result of execution
20
21  Programming style is a matter of personal choice
22  Each person can choose his own style
23  He can be very clear, or extremely messy
24
25  }
```

{ Result of execution

```
I am in Pascal school, Dad
All students are always broke
I am really getting hungry

}

  (* Κεφάλαιο 2 - Πρόγραμμα 6 EX06_2.pas *)
program Ugly_Programming_Style;begin Write('Programming style ');
 Write ('is a matter of ');
 Writeln('personal choice');Write('Each person ');
 Write('can choose ');Writeln
 ('his own style');Write('He can be ');Write
 ('very clear, or ');
 Writeln('extremely messy');end.


 { Result of execution

 Programming style is a matter of personal choice
 Each person can choose his own style
 He can be very clear, or extremely messy

 }
```

Σημειώσεις γλώσσας Pascal

Κεφάλαιο 3
Απλοί Τύποι Δεδομένων

Τι είναι ένας τύπος δεδομένων ; Ένας τύπος στην Pascal, και σε πολλές άλλες δημοφιλείς γλώσσες προγραμματισμού, καθορίζει μια μεταβλητή με τέτοιο τρόπο που κατ' αρχήν προσδιορίζει το εύρος των τιμών που είναι ικανή να αποθηκεύσει, και επίσης καθορίζει ένα σύνολο από λειτουργίες που είναι επιτρεπτό να εφαρμοσθούν πάνω στις μεταβλητές αυτού του τύπου. Η Turbo Pascal έχει 8 βασικούς τύπους δεδομένων οι οποίοι έχουν προκαθοριστεί και μπορούν να χρησιμοποιηθούν οπουδήποτε σε ένα πρόγραμμα, υπό τον όρο να χρησιμοποιούνται σωστά. Αυτό το κεφάλαιο είναι αφιερωμένο στο να σας παρουσιάσει τη χρήση των 8 αυτών τύπων ξεκαθαρίζοντας το επιτρεπτό διάστημα από τιμές που μπορούν να λάβουν, και παρουσιάζοντας τις λειτουργίες που μπορούν να γίνουν σε μεταβλητές αυτών των τύπων. Ακολουθεί η παρουσίαση αυτών των 8 τύπων καθώς και μια σύντομη περιγραφή.

integer	Όλοι οι αριθμοί στο διάστημα από -32768 έως 32767
byte	Όλοι οι ακέραιοι από 0 έως 255
real	Όλοι οι πραγματικοί αριθμοί από 1E-38 έως 1E+38
boolean	Μπορεί να έχει μόνο τιμές TRUE και FALSE
char	Όλοι οι χαρακτήρες του κώδικα ASCII
shortint	Όλοι οι ακέραιοι από -128 έως 127
word	Όλοι οι ακέραιοι από 0 έως 65535
longint	Όλοι οι ακέραιοι από -2147483648 έως 2147483647

Σημειώστε ότι 4 απ' αυτούς τους τύπους (**char**, **shortint**, **word** και **longint**) δεν αποτελούν τμήμα της Standard Pascal αλλά προστέθηκαν επιπλέον στον compiler της Turbo Pascal. Επιπρόσθετα στους παραπάνω τύπους στην Turbo Pascal στην έκδοση 5.0 υπάρχουν οι εξής τύποι:

single	Όλοι οι πραγματικοί με 7 σημαντικά ψηφία
double	Όλοι οι πραγματικοί με 15 σημαντικά ψηφία
extended	Όλοι οι πραγματικοί με 19 σημαντικά ψηφία
comp	Όλοι οι ακέραιοι από -10E18 έως 10E18

Η Turbo Pascal 5.0 και οι νεώτερες εκδόσεις έχουν αυτούς τους 4 τύπους διαθέσιμους που χρησιμοποιούν τον 80x87 μαθηματικό συνεπεξεργαστή. Επειδή η Turbo Pascal εξομοιώνει με λογισμικό τον συνεπεξεργαστή, αυτός δεν είναι απόλυτα απαραίτητος για τη χρήση αυτών των μεταβλητών. Βέβαια, η ύπαρξη μαθηματικού συνεπεξεργαστή

επιτρέπει, όλες οι πράξεις να εκτελεστούν πιο γρήγορα. Ένας πλήρης καθορισμός των διαθέσιμων τύπων κάθε compiler μπορεί να βρεθεί στο εγχειρίδιο αναφοράς της γλώσσας (**Reference Manual**). Θα ήταν καλό να διαβάσετε αυτές τις σελίδες για να μάθετε πως τους δηλώνουμε και τους χρησιμοποιούμε μέσα στο πρόγραμμα. Στα επόμενα παραδείγματα θα τους χρησιμοποιήσουμε ένα προς ένα.

Οι πρώτες μας μεταβλητές Οι **integer** είναι οι πιο εύκολοι στην κατανόηση, έτσι θα ξεκινήσουμε από αυτούς με ένα απλό πρόγραμμα που χρησιμοποιεί μερικούς **integer** με έναν πολύ απλό τρόπο. Φορτώστε το αρχείο **EX01_3.PAS** στην Turbo Pascal και ας το δούμε.

Πρόγραμμα ..... **EX01_3.PAS** Αμέσως μετά την επικεφαλίδα του προγράμματος μία άλλη δεσμευμένη λέξη φαίνεται, η λέξη **var**. Αυτή η δεσμευμένη λέξη χρησιμοποιείται για να δηλώσει μια μεταβλητή πριν αυτή χρησιμοποιηθεί οπουδήποτε στο πρόγραμμα. Υπάρχει ένας κανόνας που δεν μπορεί να παραβιαστεί στην Pascal και λέει πως "τίποτα δεν μπορεί να χρησιμοποιηθεί αν δεν έχει προηγουμένως δηλωθεί". Ο compiler θα παραπονεθεί, βγάζοντας λάθος στην μεταγλώττιση, αν προσπαθήσετε να χρησιμοποιήσετε μια μεταβλητή χωρίς να την έχετε δηλώσει. Μοιάζει λίγο ενοχλητικό να πρέπει να δηλώνεις κάθε μεταβλητή πριν τη χρήση της, αλλά αυτός ο κανόνας σας απαλλάσσει από λάθη στην ορθογραφία των μεταβλητών πριν αυτά δημιουργήσουν άλλα προβλήματα. Κάποιες άλλες γλώσσες απλά ορίζουν μια νέα μεταβλητή (π.χ. αν υπάρχει ένας διαφορετικός χαρακτήρας) και συνεχίζουν την εκτέλεση του προγράμματος με αποτέλεσμα τα παραγόμενα από τις πράξεις αποτελέσματα να είναι σκουπίδια. Σημειώστε ότι υπάρχει μόνο μια χρήση της δεσμευμένης λέξης **var**, αλλά χρησιμοποιείται για να καθορίσει τρεις διαφορετικές μεταβλητές τις **Count**, **X** και **Y**. Όταν η λέξη **Var** εντοπίζεται, ο compiler θα συνεχίσει να αναγνωρίζει ορισμούς μεταβλητών μέχρι να βρει την επόμενη δεσμευμένη λέξη. Θα ήταν επιτρεπτό να βάλουμε το **var** στην δεύτερη γραμμή αλλά δεν είναι και απαραίτητο. Επίσης επιτρεπτό θα ήταν να τοποθετήσουμε όλες τις μεταβλητές σε μια γραμμή αλλά αυτό εξαρτάται από το προγραμματιστικό στυλ του καθένα. Μετά από το σύμβολο της άνω-κάτω τελείας(:) σε κάθε γραμμή, υπάρχει η λέξη **integer** η οποία είναι βασικό αναγνωριστικό (**standard identifier**), και είναι κατά κάποιο τρόπο διαφορετικό από μια δεσμευμένη λέξη. Ένα βασικό αναγνωριστικό είναι προκαθορισμένο όπως μια δεσμευμένη λέξη, αλλά μπορείτε να το επανακαθορίσετε, χάνοντας με

αυτόν τον τρόπο τον αρχικό του σκοπό και νόημα. Προς το παρόν και για πολύ καιρό ακόμα, μην το κάνετε.

Το πρώτο μας αριθμητικό Τώρα που έχουμε 3 μεταβλητές δηλωμένες ως **integer**, είμαστε ελεύθεροι να τις χρησιμοποιήσουμε σε ένα πρόγραμμα με οποιοδήποτε τρόπο επιθυμούμε, αρκεί να τις χρησιμοποιήσουμε σωστά. Αν επιχειρήσουμε να αντιστοιχήσουμε έναν πραγματικό αριθμό στην μεταβλητή **X**, ο compiler θα δημιουργήσει ένα λάθος, και θα εμφανίσει κάτι ασυνάρτητο. Παρατηρήστε την αρχή του κυρίως σώματος του προγράμματος. Υπάρχουν 3 εκφράσεις που εκχωρούν τιμές στις **X**,**Y** και **Count**. Ένα καλό σημείο στα μαθηματικά θα έλεγε ότι η **Count** είναι μόνο ίσο με **X+Y** αρκεί αυτά να έχουν καθοριστεί, όμως το σύμβολο **=** που χρησιμοποιείται σε τόσες άλλες γλώσσες δεν χρησιμοποιείται εδώ. Χρησιμοποιείται το σύμβολο :=, και μπορεί να διαβαστεί όπως "αντικαθίσταται με την τιμή...", όταν το διαβάζεις. Ένας άλλος, πιο γρήγορος, τρόπος είναι η λέξη "παίρνει". Έτσι, **X:=X+1** διαβάζεται, "το **X** παίρνει την τιμή του **X + 1**". Θα δούμε αργότερα ότι το απλό σύμβολο της ισότητας είναι δεσμευμένο για άλλη χρήση στην Pascal. Οι τρεις πρώτες εκφράσεις δίνουν στο **X** την τιμή 12, στο **Y** την τιμή 13 και στο **Count** την τιμή 12+13 δηλαδή 25. Αν έχουμε απαίτηση να δούμε αυτές τις τιμές που είναι αποθηκευμένες στον ΗΥ θέλουμε μια άλλη επέκτασή της. Αυτό το πρώτο κομμάτι δεδομένων μέσα στην παρένθεση πρέπει να σας είναι γνωστό, αλλά το δεύτερο κομμάτι είναι κάτι νέο για σας. Πολύπλοκες εμφανίσεις μπορούμε να τις διαχειριστούν με ένα **writeln** αρκεί τα πεδία να είναι χωρισμένα με κόμμα (,). Για να εμφανίσεις την τιμή μιας μεταβλητής απλά γράψε το όνομα της στο πεδίο εμφάνισης. Ο αριθμός που ακολουθεί την μεταβλητή σε κάθε περίπτωση είναι ο αριθμός στηλών που έχει στη διάθεσή της η μεταβλητή προς εμφάνιση. Ο αριθμός είναι προαιρετικός και μπορεί να παραληφθεί, επιτρέποντας στο σύστημα να χρησιμοποιήσει τόσες στήλες, όσες χρειάζεται. Για λόγους παρουσίασης, σε κάθε μια μεταβλητή χρησιμοποιείται και διαφορετικός αριθμός στηλών. Σε αυτό το σημείο μπορείτε να μεταγλωττίσετε και να τρέξετε το πρόγραμμα **EX01_3.PAS**. Για να σας παρουσιάσουμε τους διάφορους τρόπους εμφάνισης δεδομένων, φορτώστε το αρχείο **EX02_3.PAS** και παρατηρήστε ότι εμφανίζονται τα ίδια νούμερα αλλά με διαφορετικό τρόπο. Οι εκφράσεις **writeln** έχουν κοπεί σε κομμάτια και τα ξεχωριστά κομμάτια εμφανίζονται με εκφράσεις **write** και **writeln**. Παρατηρήστε ειδικά ότι το **writeln** μόνο του μετακινεί τον κέρσορα στην επόμενη γραμμή αφού εμφανίσει την τιμή των μεταβλητών

ενώ, το **write** όχι. Μεταγλωττίστε και τρέξτε το πρόγραμμα και παρατηρήστε την εκτύπωση στην οθόνη έχοντας την εντύπωση ότι τα 2 προγράμματα είναι ίδια.

Πρόγραμμα ..... **EX02_3.PAS**

Τώρα ας χρησιμοποιήσουμε πολλές μεταβλητές Φορτώστε το αρχείο **EX03_3.PAS** για να παρατηρήσετε ένα μικρό πρόγραμμα που χρησιμοποιεί 5 από τους βασικούς τύπους δεδομένων.

Πρόγραμμα ..... **EX03_3.PAS** Οι μεταβλητές απλά δέχονται τιμές και στη συνέχεια εμφανίζονται. Μια λεπτομερής και ολοκληρωμένη περιγραφή των επιλογών της **write** παρέχονται στο εγχειρίδιο χρήσης της Turbo Pascal. Θα ήταν προς δικό σας όφελος να διαβάσετε αυτό το απόσπασμα, γιατί από δω και μπρος πολύ λίγα θα ειπωθούν για την **write**. Θα συζητήσουμε τη μέθοδο εγγραφής σε αρχεία στο δίσκο σε επόμενο κεφάλαιο του εγχειριδίου αυτού. Επιστροφή στους βασικούς τύπους. Η Pascal κάνει συνεχώς ελέγχους για εμφανή λάθη. Είναι παράνομο να αντιστοιχείς την τιμή κάποιας μεταβλητής με μια τιμή που είναι άλλου τύπου ή έξω από τα επιτρεπτά όρια της μεταβλητής αυτής. Υπάρχουν διαδικασίες που μετατρέπουν την τιμή μιας μεταβλητής από τον ένα τύπο στον άλλο όταν αυτό είναι απαραίτητο. Ας υποθέσουμε για παράδειγμα, ότι θα θέλατε να χρησιμοποιήσετε την τιμή ενός **integer** σε μια πράξη με πραγματικούς αριθμούς. Αυτό είναι πιθανό αφού πρώτα μετατρέψουμε τον **integer** σε **real** της ίδιας τιμής και χρησιμοποιώντας την νέα μεταβλητή στις επιθυμητές πράξεις. Η νέα **real** μεταβλητή πρέπει βέβαια να είναι δηλωμένη στο τμήμα **var** ως **real** προτού χρησιμοποιηθεί. Λεπτομέρειες για το πως θα κάνετε τέτοιες μετατροπές θα δείτε στο πρόγραμμα **EX08_3.PAS** αργότερα σ' αυτό το κεφάλαιο.

Πρόγραμμα ..... **EX04_3.PAS** Από τη στιγμή που έχουμε κάποιες μεταβλητές δηλωμένες, καλό θα ήταν να χρησιμοποιήσουμε τις ιδιότητες των υπολογιστών, για τις οποίες έγιναν και διάσημοι, δηλαδή τις αριθμητικές πράξεις. Δυο προγράμματα είναι διαθέσιμα για να αναφερθούμε σε κάποιες από τις δυνατότητες της Pascal σε αριθμητικές πράξεις. Το πρόγραμμα **EX04_3.PAS** χρησιμοποιεί **real** και το **EX05_3.PAS** που χρησιμοποιεί **integer** μεταβλητές. Μπορείτε να τα δείτε, να τα μεταγλωττίσετε και να τα τρέξετε χωρίς σχόλια από μένα παρά

μόνο αυτά που είναι μέσα στα προγράμματα. Ίσως πρέπει να εμφανίσετε μερικά από τα αποτελέσματα. Μπορείτε να συμβουλευτείτε και τον οδηγό της Turbo Pascal.  Πρόγραμμα ..... **EX05_3.PAS** Το παράδειγμα **EX05_3.PAS** παρουσιάζει τις μαθηματικές δυνατότητες της Pascal με αριθμούς **Integer**. Μια μεταβλητή με τύπο **byte** χρησιμοποιείται όπως και ο **integer** αλλά με μικρότερο διάστημα τιμών. Ένας αριθμός **byte** καταλαμβάνει και λιγότερο χώρο στη μνήμη του ΗΥ αφού απαιτείται ένα **byte**, ενώ στην περίπτωση του **integer** απαιτούνται 2 byte μνήμης για κάθε αριθμό.

Μεταβλητές Boolean Ας ρίξουμε μια ματιά σε μια μεταβλητή με τύπο **boolean**, η οποία μπορεί να λάβει μόνο τιμές **TRUE** ή **FALSE.** Αυτή η μεταβλητή χρησιμοποιείται σε loop, **EndOFFile** αναγνωριστικά ή σε οποιεσδήποτε άλλες συνθήκες **TRUE** ή **FALSE** μέσα στο πρόγραμμα. Οι μεταβλητές μπορούν να συγκριθούν μεταξύ τους και να καθορίσουν μια **boolean** μεταβλητή. Μια πλήρης λίστα με τους σχεσιακούς τελεστές είναι η επόμενη:

=	ίσον
<>	διάφορο
>	μεγαλύτερο
<	μικρότερο
>=	μεγαλύτερο ή ίσο από
<=	μικρότερο ή ίσο από

Αυτοί οι τελεστές μπορούν να χρησιμοποιηθούν για να συγκρίνουμε οποιουσδήποτε από τους απλούς τύπους δεδομένων όπως **integer**, **char**, **byte** και **real** ή σταθερές και μπορούν να χρησιμοποιηθούν για να συγκρίνουμε μεταβλητές με τύπο **boolean**. Μια επίδειξη είναι ότι καλύτερο για να μάθετε γύρω από τις μεταβλητές αυτές, φορτώστε λοιπόν το αρχείο **EX06_3.PAS** και παρατηρήστε το.  Πρόγραμμα ..... **EX06_3PAS** Σε αυτό το πρόγραμμα καθορίζουμε μερικές **boolean** μεταβλητές και δυο **integer** για να τους χρησιμοποιήσουμε, και ξεκινάμε αποδίδοντας τιμές στις ακέραιες μεταβλητές. Η έκφραση **Junk=Who** στην γραμμή 14 είναι βασικά μία λειτουργία **boolean** και δεν είναι αληθής αν η τιμή της **Junk** δεν είναι ίση με την τιμή της **Who**. Το αποτέλεσμα είναι λοιπόν **FALSE** και αυτή η τιμή αποθηκεύεται στην μεταβλητή **A**. Στη μεταβλητή **B** εκχωρείται η τιμή **TRUE** γιατί η έκφραση **Junk=(Who-1)** είναι αληθής. Οι μεταβλητές **Boolean C** και **D** παίρνουν τιμές ομοίως και

δεν χρειάζεται κανένα σχόλιο. Μετά από την εκχώρηση τιμής στη μεταβλητή με το μεγάλο όνομα, τα περιεχόμενα όλων των μεταβλητών εκτυπώνονται. Σημειώστε ότι αν η **A** ή η **B** είναι **TRUE** το αποτέλεσμα είναι **TRUE** στη γραμμή 18.

Που χρησιμοποιούμε τις μεταβλητές boolean ; Θα βρούμε πολλές χρήσεις για τις μεταβλητές με τύπο **boolean** όταν θα μελετήσουμε τα loops και τις συνθήκες **(if**) σύντομα, αλλά μέχρι τότε μπορούμε να μάθουμε μόνο τι είναι. Συχνά, σε μια έκφραση με συνθήκη, θα θέλετε να προβείτε σε κάποιες ενέργειες αν και οι δυο συνθήκες είναι αληθείς, σε αυτή την περίπτωση θα χρησιμοποιήσετε την δεσμευμένη λέξη **and** και δυο **boolean** εκφράσεις. Αν και οι δυο συνθήκες είναι αληθείς το αποτέλεσμα είναι **TRUE**. Η γραμμή 29 είναι ένα τέτοιο παράδειγμα. Αν οι μεταβλητές **B**, **C**, **D** είναι όλες αληθείς, τότε το αποτέλεσμα θα είναι **TRUE**. Αν μια από αυτές είναι **FALSE**, το αποτέλεσμα θα είναι **FALSE** και η **A** θα πάρει την τιμή **FALSE**. Στην γραμμή 31, όπου παρουσιάζεται ο τελεστής **OR**, αν μια από τις τρεις μεταβλητές είναι **TRUE** το αποτέλεσμα είναι **TRUE**, και αν και οι τρεις είναι **FALSE** τότε το αποτέλεσμα είναι **FALSE**. Ένας άλλος τελεστής **boolean** είναι το **not** το οποίο παρουσιάζεται στη γραμμή 30. Παρατηρείστε τη γραμμή 33, η οποία λέει ότι το αποτέλεσμα είναι **TRUE** μόνο αν η μεταβλητή **Junk** είναι κατά μια μονάδα μικρότερη από τη **Who** ή αν είναι ίσες. Αυτό καθορίζει και το επίπεδο ευελιξίας που παρέχεται με μια μεταβλητή **boolean**. Μεταγλωττίστε και τρέξτε αυτό το πρόγραμμα, μετά προσθέστε κάποιες εκτυπώσεις για να δείτε αν οι μεταβλητές παίρνουν τιμές που περιμένετε να πάρουν.

Μικρός κύκλος ή πλήρης αποτίμηση ; Ας υποθέσουμε ότι έχετε πολλές εκφράσεις **boolean "and"** μαζί και όταν η αποτίμηση ξεκινά, το πρώτο παραγόμενο αποτέλεσμα είναι **FALSE**. Αν η πρώτη έκφραση είναι **FALSE**, είναι αδύνατο για τις επόμενες εκφράσεις να πάρουν τελική τιμή **TRUE**, γιατί το πρώτο **FALSE** θα οδηγήσει το τελικό αποτέλεσμα να γίνει **FALSE**. Μοιάζει σαν άσκοπη κατανάλωση χρόνου εκτέλεσης αν κάνω πράξεις για τις οποίες γνωρίζω το αποτέλεσμα, αλλά αυτό ακριβώς κάνει η Standard Pascal εξ ορισμού. Αυτό είναι γνωστό ως πλήρης αποτίμηση (**evaluation**) μιας **boolean** έκφρασης. Αν το σύστημα είναι αρκετά έξυπνο να καταλάβει ότι το τελικό αποτέλεσμα είναι γνωστό, θα μπορούσε να σταματήσει την εκτέλεσή του. Αυτό είναι γνωστό ως πλήρης αποτίμηση (**complete evaluation**) μικρού κύκλου (**short arcuit evaluation**) μιας **boolean** έκφρασης, και μπορεί να εφαρμοσθεί αν ένας όρος από μια συνθήκη "**OR**" δώσει ως αποτέλεσμα **TRUE**, μια και το αποτέλεσμα θα

ήταν πάντα **TRUE**. Η Turbo Pascal 5.0 και οι επόμενες εκδόσεις, σας επιτρέπουν να διαλέξετε μεταξύ πλήρους και μικρού κύκλου αποτίμησης. Το εξ ορισμού για αυτούς τους compilers είναι ο μικρός κύκλος αποτίμησης και μπορεί να αλλάξει από τις επιλογές όταν χρησιμοποιείτε το ολοκληρωμένο περιβάλλον, ή χρησιμοποιώντας μια ντιρεκτίβα του compiler.

Ας δούμε τις μεταβλητές Char Ένας τύπος μεταβλητής **char** είναι μια πολύ χρήσιμη μεταβλητή, αλλά σχεδόν ποτέ δεν χρησιμοποιείται μόνη. Είναι πολύ ισχυρή όταν χρησιμοποιείται σε έναν πίνακα ή σε μια άλλη δηλωμένη από το χρήστη δομή, πράγμα το οποίο είναι πέρα από το σκοπό αυτού του κεφαλαίου. Ένα πολύ απλό πρόγραμμα, το **EX07_3.PAS** θα σας δώσει μια ιδέα πως μπορεί να χρησιμοποιηθεί μια μεταβλητή τύπου **Char**. Μελετήστε, μετά μεταγλωττίστε και τρέξτε το **EX07_3.PAS** για μια μικρή ιδέα για το τι μπορεί να κάνει ο τύπος αυτός.

 Πρόγραμμα ..... **EX07_3.PAS** Μελετήστε το πρόγραμμα **EX08_3.PAS** για αν δείτε μέσω παραδειγμάτων μετατροπής δεδομένων από έναν τύπο σε άλλον. Τα σχόλια κάνουν το πρόγραμμα αρκετά ευκολονόητο.  Πρόγραμμα ..... **EX08_3.PAS** ***Εκτεταμένοι τύποι Integer*** Στο πρόγραμμα **EX09_3.PAS** θα δούμε τη χρήση εκτεταμένων τύπων **Integer**, που είναι διαθέσιμοι με τον compiler της Pascal.  Πρόγραμμα ..... **EX09_3.PAS** Στο πρόγραμμα αυτό κατ' αρχήν δηλώνουμε 4 μεταβλητές, μετά καταχωρούμε τιμές σε κάθε μια από αυτές και στη συνέχεια εμφανίζουμε τις τιμές των μεταβλητών στην οθόνη. Όταν μεταγλωττίσετε και τρέξετε το πρόγραμμα, θα δείτε ότι η μεταβλητή **Big_int** μπορεί να αποθηκεύσει έναν μεγαλύτερο αριθμό. Πρέπει να τονιστεί ότι οι υπολογισμοί στις γραμμές 13 και 21 δίνουν διαφορετικά αποτελέσματα, αν και δείχνουν να υπολογίζουν την ίδια αριθμητική έκφραση. Γιατί συμβαίνει αυτό θα το εξηγήσουμε στη συνέχεια. Η ποσότητα με όνομα **MaxInt** που χρησιμοποιείται στις γραμμές 10 και 13 είναι μια σταθερά, κατασκευασμένη από το σύστημα και συμβολίζει την μεγαλύτερη τιμή που μια μεταβλητή **integer** μπορεί να αποθηκεύσει. Στην πρώτη παράγραφο αυτού του κεφαλαίου καθορίσαμε ότι η μεγαλύτερη τιμή μιας **integer** μεταβλητής είναι 32767 και τρέχοντας το πρόγραμμα θα δείτε πως η μεταβλητή **Index** θα αποκτήσει αυτήν την τιμή. Η σταθερά

MaxInt έχει τύπο που είναι από ένα διεθνή τύπο ακεραίου όπως όλες οι αριθμητικές σταθερές στην γραμμή 13. Το αποτέλεσμα υπολογίζεται από τον αριθμό των σημαντικών ψηφίων υπαγορευόμενο από το αριστερό κομμάτι της εν λόγω έκφρασης η οποία είναι τύπου **longint**. Όταν πάμε στην γραμμή 21, όμως, η μεταβλητή **Index** είναι τύπου **integer** και έτσι οι υπολογισμοί γίνονται διαμέσου των σταθερών που είναι τύπου **integer**, το οποίο έχει ως αποτέλεσμα ορισμένα σημαντικά ψηφία να "κουτσουρεύονται". Το αποτέλεσμα αυτό μετατρέπεται σε τύπο **longint** και καταχωρείται στη μεταβλητή **Big_int** και η τιμή εμφανίζεται στη γραμμή 22. Μετά από αυτή τη συζήτηση θα πρέπει να έχετε εμπεδώσει ότι είναι πολύ σημαντικό να χρησιμοποιείς για τις μεταβλητές σου τους κατάλληλους τύπους δεδομένων γιατί διαφορετικά καταναλώνεται μεγάλος χώρος στη μνήμη και καθυστερεί η εκτέλεση των πράξεων. Η εμπειρία θα καθορίζει τους κατάλληλους τύπους δεδομένων σε κάθε εφαρμογή. ***Εκτεταμένοι τύποι Real*** Εμφανίστε το πρόγραμμα **EX10_3.PAS** ως ένα παράδειγμα που χρησιμοποιεί τους νέους τύπους "**real**" που είναι διαθέσιμοι με τις νεώτερες εκδόσεις της Pascal.

Πρόγραμμα ..... **EX10_3.PAS** Το τι κάνει αυτό το πρόγραμμα είναι εύκολο να καταλάβετε άρα δεν χρειάζεται να πούμε τίποτα. Οταν το τρέξετε, μπορείτε να παρατηρήσετε την συγκριτική ακρίβεια κάθε ενός από τους τύπους μεταβλητών. Για μια ακόμα φορά, πρέπει να έχετε στο μυαλό ότι χρησιμοποιώντας τον μεγαλύτερο δυνατό τύπο μεταβλητής, καταναλώνουμε παραπάνω αποθηκευτικό χώρο και ωφέλιμο χρόνο εκτέλεσης, αλλά κερδίζουμε μεγαλύτερη ακρίβεια. ***Άσκηση προγραμματισμού*** **1.** Γράψτε ένα πρόγραμμα που να περιέχει πολλές μεταβλητές, κάντε μαθηματικές πράξεις με αυτές και εκτυπώστε τα αποτελέσματα.

Προγράμματα κεφαλαίου 3

```
1  (* Κεφάλαιο 3 - Πρόγραμμα 1 EX01_3.pas*)
2  program Integer_Variable_Demo;
3
4  var  Count : integer;
5         X,Y : integer;
6
```

```
7  begin
8     X := 12;
9     Y := 13;
10     Count := X + Y;
11     Writeln('The value of X is',X:4);
12     Writeln('The value of Y is',Y:5);
13     Writeln('And Count is now ',Count:6);
14 end.
15
16 { Result of execution
17
18 The value of X is 12
19 The value of Y is 13
20 And Count is now 25
21
22 }
```

```
1  (* Κεφάλαιο 3 - Πρόγραμμα 2 EX02_3.pas*)
2  program More_Integer_Demos;
3
4  var  X,Y : integer;
5         Count : integer;
6
7  begin
8     X := 12;
9     Y := 13;
10     Count := X + Y;
11     Write('The value of X is');
12     Writeln(X:4);
13     Write('The value of Y is');
14     Writeln(Y:5);
15     Write('And Count is now ');
16     Write(Count:6);
17     Writeln;
18 end.
19
20
21 { Result of execution
```

```
22
23  The value of X is 12
24  The value of Y is 13
25  And Count is now 25
26
27  }
```

```
1   (* Κεφάλαιο 3 - Πρόγραμμα 3 EX03_3.pas*)
2   program All_Simple_Variable_Types;
3
4   var  A,B : integer;
5          C,D : byte;
6          Dog_Tail : real;
7          Puppy : boolean;
8          Animal_Cookies : char;
9
10  begin
11     A := 4;
12     B := 5;
13     C := 212;
14     D := C + 3;
15     Dog_Tail := 345.124556;
16     Puppy := B > A; (* since B is greater than A, Puppy
17     will be assigned the value TRUE *)
18     Animal_Cookies := 'R'; (* this is a single character *)
19
20     Writeln('The integers are',A:5,B:5);
21     Writeln('The bytes are', C:5,D:5); (* notice that the spaces
22     prior to the C will be
23     ignored on output *)
24     Writeln('The real value is',Dog_Tail:12:2,Dog_Tail:12:4);
25     Writeln;
26     Writeln('The boolean value is ',Puppy,Puppy:13);
27     Writeln('The char variable is an ',Animal_Cookies);
28  end.
29
30  { Result of execution
31
```

```
32  The integers are 4 5
33  The bytes are 212 215
34  The real value is 345.12 345.1246
35
36  The boolean value is TRUE TRUE
37  The char variable is an R
38
39  }
```

```
1   (* Κεφάλαιο 3 - Πρόγραμμα 4 EX04_3.pas*)
2   program Real_Math_Demo;
3
4   var A,B,C,D : real;
5
6   begin
7      A := 3.234; {simply assigning a value}
8      B := A + 1.101; {adding a constant}
9      C := A + B; {summing two variables}
10     D : = 4.234*A*B; {multiplication}
11     D := 2.3/B; {division}
12     D := A - B; {subtraction}
13     D := (A + B)/(12.56-B); {example of a multiple expression}
14
15     {It will be left up to you to print out some of the above values}
16  end.
17
18  { Result of execution
19
20  (There is no output from this program )
21
22  }
```

```
1   (* Κεφάλαιο 3 - Πρόγραμμα 5 EX05_3.pas*)
2   program Integer_Math_Demo;
3
4   var   A,B,C,D : integer;
5           E : real;
6
```

```
7  begin
8     A := 9; (* Simple assignment *)
9     B : = A + 4; (* simple addition *)
10    C := A + B; (* simple addition *)
11    D := 4*A*B; (* multiplication *)
12    E := A/B; (* integer division with the result
13    expressed as a real number *)
14    D := B div A; (* integer division with the result
15    expressed as a truncated integer
16     number *)
17    D := B mod A; (* d is the remainder of the division,
18     in this case d = 4 *)
19    D := (A + B) div (B + 7); (* composite math statement *)
20
21     (* It will be up to you to print out some of these values *)
22 end.
23
24
25 { Result of execution
26
27 (There is no output from this program)
28
29 }
```

```
1  (* Κεφάλαιο 3 - Πρόγραμμα 6 EX06_3.pas*)
2  program Illustrate_What_Boolean_Math_Looks_Like;
3
4  (* notice the program name, it can be up to 63 characters long.
5  Variables can also be very long as we will see below *)
6
7  var    A,B,C,D : boolean;
8         A_Very_Big_Boolean_Name_Can_Be_Used : boolean;
9         Junk,Who : integer;
10
11 begin
12    Junk := 4;
13    Who := 5;
14    A := Junk = Who; {since Junk is not equal to Who, A is false}
```

```
15      B := Junk = (Who - 1); {This is true}
16      C := Junk < Who; {This is true}
17      D := Junk > 10; {This is false}
18      A_Very_Big_Boolean_Name_Can_Be_Used := A or B; {Since B is
true,
19       the result is true}
20      Writeln('result A is ',A);
21      Writeln('result B is ',B);
22      Writeln('result C is ',C);
23      Writeln('result D is ',D:12); {This answer will be right justified
24        in a 12 character field}
25      Writeln('result A_Very_Big_Boolean_Name_Can_Be_Used is ',
26      A_Very_Big_Boolean_Name_Can_Be_Used);
27
28       (* Following are a few boolean expressions. *)
29      A := B and C and D;
30      A := B and C and not D;
31      A := B or C or D;
32      A := (B and C) or not (C and D);
33      A := (Junk = Who - 1) or (Junk = Who);
34  end.
35
36  { Result of execution
37
38  result A is FALSE
39  result B is TRUE
40  result C is TRUE
41  result D is FALSE
42  result A_Very_Big_Boolean_Name_Can_Be_Used is TRUE
43
44  }
```

```
1   (* Κεφάλαιο 3 - Πρόγραμμα7 EX07_3.pas*)
2   program Char_Demonstration;
3
4   var   Letter : char;
5         Number : char;
6         Dogfood : char;
```

```
7
8  begin
9     Letter := 'P';
10     Number := 'A';
11     Do gfood := 'S';
12     Write(Letter,Number,Dogfood);
13      Letter := Number; (* This is now the 'A' *)
14      Number := 'L';
15      Dogfood := 'C';
16      Write(Dogfood,Letter,Number);
17      Writeln;
18 end.
19
20 { Result of execution
21
22 PASCAL
23
24 }

1  (* Κεφάλαιο 3 - Πρόγραμμα 8 EX08_3.pas*)
2  program Convert_From_Type_To_Type;
3
4  var   Index,Count : integer;
5          Error_Ind : integer;
6          Size,Cost : real;
7          Letter : char;
8          Name,Amount : string[12];
9
10 begin
11     Index := 65;
12     Count := 66;
13     Cost := 124.678;
14     Amount := '12.4612';
15
16     Letter := Chr(Index); (* convert integer to char *)
17     Size := Count; (* convert integer to real *)
18
19     Index := Round(Cost); (* real to integer, rounded *)
```

```
20      Count := Trunc(Cost); (* real to integer, truncated *)
21
22       Index := Ord(Letter); (* convert char to integer *)
23      Str(Count,Name); (* integer to string of char *)
24      Val(Amount,Size,Error_Ind); (* string to real note that
25        "Error_Ind" is used for
26      returning an error code *)
27
28      Writeln('Name is ',Name,' and Size is ',Size:10:4);
29  end.
30
31   32  { Result of execution
33
34  Name is 124 and Size is 12.4612
35
36  }
```

```
1   (* Κεφάλαιο 3 - Πρόγραμμα 9 EX09_3.pas*)
2   program Extended_Integer_Types;
3
4   var   Index : integer;
5           Big_int : longint;
6           Small_int : shortint;
7           Pos_int : word;
8
9   begin
10     Index := MaxInt;
11     Small_int := 127;
12     Pos_int := Index + 256 * Small_int;
13     Big_int := 1000 * MaxInt + 1234;
14
15     Writeln('Index = ',Index:12);
16     Writeln('Small_int = ',Small_int:12);
17     Writeln('Pos_int = ',Pos_int:12);
18     Writeln('Big_int = ',Big_int:12);
19     Writeln;
20
21     Big_int := 1000 * Index + 1234;
```

```
22      Writeln('Big_int = ',Big_int:12);
23  end.
24
25
26  { Result of execution
27
28  Index = 32767
29  Small_Int = 127
30  Pos_Int = 65279
31  Big_Int = 32768234
32
33  Big_Int = 234
34
35  }
```

```
1   (* Κεφάλαιο 3 - Πρόγραμμα 10 EX10_3.pas*)
2   program Extended_Real_Types;
3
4   (* Note: If you are using TURBO Pascal Version 5.0 or newer *)
5   (* and you do not have a Math Co_Processor, you can *)
6   (* still compile and run this program by using the *)
7   (* compiler directive as explained in the User's Guide. *)
8
9   var  Number : real;
10         Small_Number : single;
11         Big_Number : double;
12         Huge_Number : extended;
13         Whole_Number : comp;
14
15  begin
16      Number := 10000000000000000000000000.0;
17      Small_Number := 10000000000000000000000000.0;
18      Big _Number := 10000000000000000000000000.0;
19      Huge_Number := 10000000000000000000000000.0;
20      Whole_Number := 10000000000000000.0;
21
22      Writeln('Number = ',Number :40:3);
23      Writeln('Small_Number = ',Small_Number:40:3);
```

```
24      Writeln('Big_Number = ',Big_Number :40:3);
25      Writeln('Huge_Number = ',Huge_Number :40:3);
26      Writeln('Whole_Number = ',Whole_Number:40:3);
27  end.
28
29
30  { Result of execution
31
32  Number = 999999999998590010000000.000
33  Small_Number = 1000000025377642900000000000.000
34  Big_Number = 100000000000000005000000000.000
35  Huge_Number = 10000000000000000000000000000.000
36  Whole_Number = 100000000000000000.000
37
38  }
```

Κεφάλαιο 4
Επαναληπτικές Δομές και Δομές Ελέγχου

Κάθε πρόγραμμα που έχουμε εξετάσει μέχρις αυτό το σημείο ήταν απλό και καμία εντολή του δεν επαναλαμβάνονταν. Όπως και σε όλες τις άλλες γλώσσες, η Pascal έχει επιπλέον χαρακτηριστικά για να εκτελεί επαναληπτικές διαδικασίες και διακλαδώσεις υπό συνθήκες. Θα απασχοληθούμε με αυτά τα θέματα στη συνέχεια.
Η επαναληπτική δομή For **θα ξεκινήσουμε με τη δομή που ίσως είναι και η πιο εύκολη στην κατανόηση, τη For.**

Πρόγραμμα ..... EX01_4.PAS Αυτή χρησιμοποιείται για να επαναλάβουμε μια εντολή της Pascal όσες φορές επιθυμούμε. Φορτώστε το αρχείο EX01_4.PAS και θα συζητήσουμε τα loops που παρουσιάζονται εδώ. Το παράδειγμα που παρουσιάζεται στις γραμμές 13 και 14, είναι η πιο απλή επανάληψη (loop) και δεν κάνει τίποτα άλλο από το να εκτελεί ένα writeln 7 φορές. Έχουμε 3 νέες δεσμευμένες λέξεις το for, το to και το do που χρησιμοποιούνται ως εξής. Οποιαδήποτε απλή μεταβλητή με τύπο integer, byte ή char μπορεί να χρησιμοποιηθεί ως δείκτης επανάληψης με την προϋπόθεση ότι πρέπει να υπάρχει μεταβλητή που να είναι δηλωμένη ως var. Μετά τη δεσμευμένη λέξη do μπορεί να ακολουθεί οποιαδήποτε απλή εντολή της Pascal που μπορεί να επαναληφθεί τόσες φορές όσες ο καθορισμένος αριθμός επαναλήψεων. Σημειώστε ότι στο loop αυτό ο δείκτης της επανάληψης συνεχώς αυξάνεται ενώ με την προσθήκη του downto (στη θέση του to) μπορεί να μειώνεται όπως φαίνεται στο τελευταίο παράδειγμα αυτού του προγράμματος. Ας σημειώσουμε ότι η αύξηση ή η μείωση μπορεί να γίνει μόνο κατά 1 στην Pascal.
Μια ομάδα εντολών της Pascal **Το παράδειγμα που δίνεται στις γραμμές 18 και 22 περιέχει την πρώτη μιας ομάδας εντολών (compound statement) σε Pascal. Είχε αναφερθεί στο κεφάλαιο 1 ότι το ζευγάρι δεσμευμένων λέξεων begin-end μπορεί να χρησιμοποιηθεί για να δηλώσει μια ομάδα εντολών (bloc). Σε αυτήν την περίπτωση, η συνδυασμένη έκφραση που ξεκινά με begin στο τέλος της γραμμής 18 μέχρι και το τέλος της γραμμής 22, είναι η απλή εντολή που θα εκτελεστεί 10 φορές. Μια δεύτερη μεταβλητή η Total εμφανίζεται να μεταβάλλεται μέσα στο σώμα του βρόχου**

(loop). Οποιαδήποτε έγκυρη εντολή της Pascal μπορεί να εκτελεστεί μεταξύ των begin-end, συμπεριλαμβανομένου και άλλου βρόχου επανάληψης, δίνοντας στις φωλιασμένες (nested) βρόχους όποιο βάθος επιθυμούμε. Το τρίτο παράδειγμα δείχνει πως ο τύπος μεταβλητής char μπορεί να χρησιμοποιηθεί σε ένα βρόχο for. Η Pascal απαιτεί ο δείκτης επανάληψης, το σημείο έναρξης και το σημείο τερματισμού, να είναι ίδιου τύπου, διαφορετικά θα υπάρξει μήνυμα λάθους κατά τη διάρκεια της μεταγλώττισης. Επιπρόσθετα, πρέπει να είναι μεταβλητές του ίδιου τύπου integer, byte ή char, Το σημείο έναρξης και το σημείο τερματισμού μπορεί να είναι σταθερές ή εκφράσεις αυθαίρετης πολυπλοκότητας. Το τέταρτο παράδειγμα είναι ένας βρόχος που μειώνεται, όπως αναφέρθηκε και νωρίτερα. Χρησιμοποιεί τη δεσμευμένη λέξη downto και πρέπει να είναι κατανοητό.

Η εντολή IF **Η Pascal έχει δυο εντολές ελέγχου την if και την case. Θα ρίξουμε μια ματιά στην πιο απλή τώρα, την if. Φορτώστε το EX02_4.PAS για να δείτε και μόνοι σας το ζευγάρι των δεσμευμένων λέξεων if -then στις γραμμές 11 και 12.**

Πρόγραμμα ..... EX02_4.PAS Οποιαδήποτε συνθήκη που μπορεί να οδηγήσει σε μια λογική (boolean) απάντηση τοποθετείται ανάμεσα στο ζεύγος λέξεων if -then. Αν το αποτέλεσμα είναι TRUE τότε η εντολή ή το σύνολο των εντολών που ακολουθούν το then εκτελούνται, και αν είναι FALSE, τότε οι εντολές αυτές παρακάμπτονται. Βέβαια είναι εύκολο να μαντέψετε ότι αν υπάρχει μπλοκ εντολών αυτές θα βρίσκονται ανάμεσα σε ένα begin και σε ένα end. Μελετήστε το παράδειγμα 1 και θα δείτε ότι η γραμμή εκτυπώνεται πάντα γιατί το 3 είναι ίσο με 1+2. Το δεύτερο παράδειγμα στις γραμμές 14 έως 19, είναι παρόμοιο με το πρώτο αλλά έχει αντικατασταθεί η μοναδική εντολή με μια ομάδα από εντολές και θα πρέπει να είναι εύκολο για σας να κατανοήσετε τι κάνει το παράδειγμα αυτό. Το τρίτο παράδειγμα στις γραμμές 21 έως 24, περιέχει μια νέα δεσμευμένη λέξη, την else. Όταν το if παίρνει τιμή FALSE, η εντολή (ή οι εντολές) που ακολουθούν το then παρακάμπτονται και αν υπάρχει ερωτηματικό, τότε το if αγνοείται τελείως. Αν αντί για ερωτηματικό υπάρχει else τότε εκτελείται η εντολή ή η ομάδα εντολών που το ακολουθούν. Μόνο μια από τις δυο ομάδες εντολών θα εκτελεστεί κάθε φορά.

Περαιτέρω παρατήρηση του τρίτου παραδείγματος θα το ξεκαθαρίσει αυτό στο μυαλό σας. Σημειώστε ότι ο compiler της Pascal ψάχνει είτε για ένα ερωτηματικό για να τερματίσει το if είτε για τη δεσμευμένη λέξη else. Θα λάβετε ένα μήνυμα λάθους, αν συμπεριλάβετε και το ερωτηματικό πριν από το else. Είναι ένα συνηθισμένο λάθος και διορθώνεται εύκολα.

Το μπλοκ if-then-else **Συγκεντρώστε όλη σας την προσοχή γιατί η επόμενη αρχή είναι δύσκολη στην κατανόηση με την πρώτη ματιά, αλλά με την χρήση θα διαλευκανθεί, οπότε και θα γίνει η πιο χρήσιμη γνώση στον προγραμματισμό με Pascal. Από τη στιγμή που όλο το μπλοκ κώδικα if then else είναι εξ ορισμού μία απλή εντολή της Pascal μπορεί να χρησιμοποιηθεί οπουδήποτε μια εντολή είναι νόμιμη χωρίς διαχωριστές begin end. Αυτό φαίνεται στο τέταρτο παράδειγμα του προγράμματος EX02_4.PAS. Οι γραμμές 27 έως 30 περιλαμβάνουν μια απλή εντολή Pascal, και και ο γραμμές 32 μέχρι 35 περιλαμβάνουν μια απλή εντολή. Η εντολή if ξεκινά από τη γραμμή 26, έχει μια εντολή σε κάθε κλάδο της (then-else) και θεωρείται ως μια εντολή, μια εντολή που ξεκινά από τη γραμμή 26 και τελειώνει στη γραμμή 35. Ξαναδιαβάστε αυτήν την παράγραφο μέχρι να την καταλάβετε καλά, επειδή είναι μια πολύ βασική αρχή. Η εντολή If then else είναι μια από τις πιο χρησιμοποιημένες, πιο χρήσιμες και από τις πιο σημαντικές αρχές της Pascal. Γι' αυτό το λόγο πρέπει να την μάθετε πολύ καλά. Προσπαθήστε να αλλάξετε μερικές από τις συνθήκες στο πρόγραμμα και να τις εμφανίσετε για να δείτε τις αλλαγές που συμβαίνουν. Όταν είστε έτοιμοι, θα πάμε σε ένα παράδειγμα που χρησιμοποιεί και βρόχους και συνθήκες μαζί.** ***Βρόχοι και συνθήκες μαζί*** **Φορτώστε το EX03_4.PAS και μελετήστε το για μερικά λεπτά. Περιέχει τα περισσότερα από όσα έχετε διδαχθεί μέχρι τώρα και θα πρέπει να σας είναι κατανοητό. Περιλαμβάνει ένα βρόχο (γραμμές 7 μέχρι 17) με δυο εντολές μέσα του (γραμμές 8 και 9 και γραμμές 10 μέχρι 16), και έναν άλλο βρόχο (γραμμές 11 μέχρι 15) που περιλαμβάνει μια εντολή if.**

Πρόγραμμα ..... EX03_4.PAS

Πρέπει να προσέξετε το στυλ γραφής που χρησιμοποιούμε εδώ. Το begin είναι στο τέλος της γραμμής που ξεκινά ο έλεγχος και το end ευθυγραμμίζεται με τον έλεγχο, ώστε να είναι σαφές με ποιο if

συνδέεται. Όλες οι εντολές οι οποίες ακολουθούν μια δομή ελέγχου αρχίζουν την παράγραφο πιο μέσα και αυτό βοηθά στην αναγνωσιμότητα του προγράμματος. Θα αναπτύξετε την δική σας μέθοδο διαμόρφωσης του κώδικά σας εν καιρώ αλλά μέχρι τότε σας συμβουλεύω να ακολουθείτε τις οδηγίες αυτού του παραδείγματος, το οποίο είναι γραμμένο με έναν τρόπο που είναι αποδεκτός από όλους τους προγραμματιστές σε Pascal. Ένα εύκολο λάθος που μπορείτε να κάνετε πρέπει να σημειωθεί τώρα. Αν ένα πλεονάζον ερωτηματικό έμπαινε το τέλος του if, στην γραμμή 8, ο κώδικας που ακολουθεί θα εκτελείτο πάντα, αφού η κενή εντολή (δεν υπάρχει τίποτα ανάμεσα στο then και στο ερωτηματικό) θα ήταν η εντολή που αντιστοιχεί στο then. Ο compiler δεν θα έδειχνε λάθος και δεν θα σας προειδοποιούσε κανείς. Προσθέστε ένα ερωτηματικό στο τέλος της γραμμής 8 για να δείτε το λάθος. Βέβαια, πρέπει να τρέξετε το πρόγραμμα για να δείτε να εκτυπώνονται και οι 10 τιμές της count.

Επιτέλους, ένα πρόγραμμα που να σημαίνει κάτι **Φορτώστε το αρχείο EX04_4.PAS και μελετήστε τη δομή του.**

Πρόγραμμα ..... EX04_4.PAS Σημειώστε την επικεφαλίδα που καθορίζει το πρόγραμμα και δίνει μια καλή εξήγηση για το τι κάνει το πρόγραμμα. Αυτό το πρόγραμμα δεν πρέπει να σας δημιουργεί κανένα πρόβλημα στο να καταλάβετε τι κάνει, αφού είναι τόσο καλά γραμμένο. Αν αλλάξετε το στυλ των κενών διαστημάτων εδώ, θα μάθετε να εκτιμάτε την καθαρότητα που παρέχεται από την χρήση κενών. Μεταγλωττίστε και τρέξτε το πρόγραμμα αυτό και θα έχετε μια λίστα με θερμοκρασίες σε βαθμούς Φαρενάιτ και δίπλα κάποιες σημειώσεις. Φορτώστε το αρχείο EX05_4.PAS που είναι ένα καλό παράδειγμα φτωχής ονοματολογίας των μεταβλητών. Αυτή η δομή του προγράμματος είναι παρόμοια με το τελευταίο πρόγραμμα, και όταν το τρέξετε, θα δείτε ότι παράγουν τα ίδια αποτελέσματα, αλλά σε σύγκριση με το τελευταίο πρόγραμμα είναι δύσκολο να καταλάβεις τι κάνει διαβάζοντας το πρόγραμμα.

Πρόγραμμα ..... EX05_4.PAS Η φτωχή επιλογή των ονομάτων των μεταβλητών και η έλλειψη σχολίων στο παρόν πρόγραμμα το καθιστά μη κατανοητό, αλλά πολλοί προγραμματιστές έχουν συνηθίσει να

γράφουν προγράμματα σαν κι αυτό. Πρέπει να είστε υπομονετικοί σε αυτόν τον τομέα, να δίνετε κατανοητά ονόματα στις μεταβλητές σας και να γράφετε πολλά σχόλια, ώστε τα προγράμματά σας να είναι κατανοητά. Αυτό το πρόγραμμα πρέπει να είναι εύκολα κατανοητό από σας, επομένως ας πάμε στο επόμενο παράδειγμα. ***Ο βρόχος Repeat Until*** **Οι δυο επόμενες δομές της Pascal είναι παρόμοιες επειδή είναι και οι δυο ασαφείς βρόχοι (ασαφείς αφού δεν είναι σταθερός ο αριθμός των επαναλήψεων). Ένας από τους βρόχους είναι αποτιμημένος στην αρχή και ο άλλος στο τέλος. Είναι πιο απλό να ξεκινήσουμε με την κατασκευή Repeat-Until η οποία είναι αποτιμημένη στο τέλος. Εξετάστε το αρχείο EX06_4.PAS για να δείτε ένα παράδειγμα χρήσης της Repeat-Until.** **Πρόγραμμα ….. EX06_4.PAS** **Δυο ακόμα δεσμευμένες λέξεις παρουσιάζονται εδώ, με ονόματα Repeat και Until. Αυτή η κατασκευή απλά επαναλαμβάνει τις εντολές ανάμεσα στις δυο αυτές λέξεις μέχρι η συνθήκη που ακολουθεί το Until να γίνει TRUE. Αυτή είναι η μόνη έκφραση στην Pascal που περιέχει πολλές εκφράσεις στο εσωτερικό της χωρίς να χρειάζεται να περιέχει begin και end. Κάτι που πρέπει να προσέξουμε πολύ σε αυτό το σημείο. Αφού ο βρόχος εκτελείται μέχρι η συνθήκη να γίνει αληθής και είναι πιθανό να μη γίνει ποτέ, και έτσι ο βρόχος δεν θα τερματιστεί ποτέ. Είναι στο χέρι σας, του προγραμματιστή, να διασφαλίσει ότι ο βρόχος θα τερματιστεί σίγουρα. Μεταγλωττίστε και εκτελέστε το πρόγραμμα για δείτε τα αποτελέσματα στην οθόνη.**

Ο βρόχος While **Το αρχείο EX07_4.PAS περιέχει ένα παράδειγμα μιας νέας δομής, του βρόχου While. Αυτό χρησιμοποιεί τις δεσμευμένες λέξεις while και do και θα εκτελέσει μια εντολή Pascal (ή μια ομάδα από εντολές - bloc εντολών - μέσα σε begin και end) συνεχώς μέχρι η συνθήκη ανάμεσα στις δυο λέξεις να γίνει FALSE.** **Πρόγραμμα ….. EX07_4.PAS** **Αυτός ο βρόχος είναι επίσης ασαφής και θα μπορούσε, όπως ο βρόχος Repeat-Until, να μην τερματιστεί ποτέ. Υπάρχουν δυο βασικές διαφορές στους δυο τελευταίους βρόχους. Ο βρόχος Repeat-Until είναι αποτιμημένος στο τέλος του βρόχου και εκτελείται οπωσδήποτε μια φορά. Ο βρόχος While είναι αποτιμημένος στην**

αρχή και υπάρχει πιθανότητα να μην εκτελεστεί ούτε μια φορά. Αυτό σας δίνει την ευελιξία να επιλέγετε το βρόχο που θα χρησιμοποιήσετε ανάλογα με αυτό που θέλετε να κάνετε. Μεταγλωττίστε και τρέξτε και εξετάστε τα αποτελέσματα του προγράμματος EX07_4.PAS. ***Η εντολή Case*** **Η τελευταία δομή έλεγχου παρουσιάζει άλλη μια δεσμευμένη λέξη, την λέξη case. Η case λογικά θα έπρεπε να συμπεριληφθεί με την If αφού είναι μια εντολή ελέγχου, αλλά την κρατήσαμε για το τέλος γιατί είναι πολύ ασυνήθιστη και θα χρησιμοποιηθεί λιγότερο από τις άλλες που συζητήσαμε σε αυτό το κεφάλαιο. Η εντολή Case χρησιμοποιείται για να επιλέξουμε μια από πολλές πιθανές προς εκτέλεση εντολές Pascal, βασισμένη στην τιμή μια απλής μεταβλητής.**  **Πρόγραμμα ..... EX08_4.PAS Φορτώστε το αρχείο EX08_4.PAS και παρατηρήστε το πρόγραμμα ως παράδειγμα μιας case εντολής. Η μεταβλητή μεταξύ μιας case και της δεσμευμένης λέξης of στη γραμμή 9 είναι η μεταβλητή που χρησιμοποιείται για την επιλογή και ονομάζεται επιλογέας case ή μεταβλητή επιλογής. Στη συνέχεια οι διάφορες επιλογές παραθέτονται γράφοντας μια τιμή ή ένα διάστημα, ακολουθούμενη από άνω κάτω τελεία (:) μια εντολή Pascal , και ένα ερωτηματικό για κάθε επιλογέα. Ακολουθώντας τη λίστα των επιλογών, ένα else μπορεί να προστεθεί για να καλύψει την πιθανότητα καμία από τις επιλογές να μην έχει εκτελεστεί. Τελικά, η εντολή end χρησιμοποιείται για να τον τερματισμό της δομής case. Σημειώστε ότι αυτή είναι μια από τις σπάνιες περιπτώσεις στην Pascal όπου στο end δεν αντιστοιχεί ένα begin. Το παράδειγμα χρησιμοποιεί τη μεταβλητή count ως επιλογέα, εκτυπώνει τους αριθμούς από 1 έως 5 στην οθόνη και εμφανίζει μήνυμα για τους αριθμούς που ανήκουν εκτός αυτού του εύρους τιμών. Το πρόγραμμα πρέπει να είναι αρκετά κατανοητό μετά από αυτό το σημείο. Βεβαιωθείτε ότι θα μεταγλωττίσετε και θα τρέξετε το πρόγραμμα αυτό. Φορτώστε και εμφανίστε το πρόγραμμα EX09_4.PAS που είναι ένα άλλο παράδειγμα χρήσης της case με κάποια άλλα επιπρόσθετα στοιχεία. Αυτό το πρόγραμμα χρησιμοποιεί την ίδια δομή με το προηγούμενο πρόγραμμα αλλά στη γραμμή 11 ένα σύνολο χρησιμοποιείται ως επιλογέας ώστε αν η τιμή της μεταβλητής Count είναι 7,ή 8 ή 9 αυτή η επιλογή θα**

εκτελεσθεί.  **Πρόγραμμα ..... EX09_4.PAS Στην γραμμή 12, 3 διαφορετικές παρατιθέμενες τιμές θα προκαλέσουν την επιλογή του αντίστοιχου σημείου του κώδικα. Μεγαλύτερης σημασίας είναι οι σύνθετες εντολές που χρησιμοποιούνται σε μερικές από τις επιλογές. Αν η Count έχει τιμή 2 ή 4 ή 6 μια σύνθετη εντολή θα εκτελεστεί και αν η τιμή είναι 3 εκτελείται ένας βρόχος for. Αν η τιμή είναι 1, εκτελείται μια εντολή if η οποία υποχρεώνει μια σύνθετη εντολή να εκτελεστεί. Σε αυτήν την περίπτωση η εντολή If θα εκτελείται πάντα αφού το TRUE είναι πάντα ίσο με TRUE, βέβαια οποιαδήποτε άλλη έκφραση boolean θα μπορούσε να χρησιμοποιηθεί σε αυτήν την έκφραση. Μεταγλωττίστε και εκτελέστε το πρόγραμμα, μελετήστε τα αποτελέσματα μέχρι να τα καταλάβετε καλά. Εδώ φτάσαμε στο τέλος του κεφαλαίου 4 και τώρα πια έχετε αρκετές πληροφορίες για να γράψετε οποιοδήποτε πρόγραμμα επιθυμείτε στην Pascal. Θα δείτε πάντως, ότι θα έχετε λίγες δυσκολίες αν επιχειρήσετε να γράψετε ένα μεγάλο πρόγραμμα, χωρίς να διαβάσετε τα κεφάλαια που ακολουθούν. Αυτά θα σας διδάξουν να γράφετε ευέλικτα προγράμματα.** ***Ασκήσεις Προγραμματισμού*** **1. Γράψτε ένα πρόγραμμα που να εμφανίζει σε λίστα τους αριθμούς από το 1 έως το 12 και να γράφει ένα μήνυμα μετά τον αριθμό που εκφράζει το μήνα γέννησής σας. 2. Γράψτε ένα πρόγραμμα που εμφανίζει τους αριθμούς από το 1 έως το 12 εκτός από τους αριθμούς 2 και 9.**

Προγράμματα κεφαλαίου 4

```
1   (* Κεφάλαιο 4 - Πρόγραμμα 1 EX01_4.pas*)
2   program Demonstrate_Loops;
3
4    var Count : integer;
5          Start : integer;
6          Ending : integer;
7          Total : integer;
8         Alphabet : char;
```

```
9
10 begin
11    Start := 1;
12   Ending := 7;
13   for Count := Start to Ending do(* Example 1 *)
14        Writeln('This is a count loop and we are in pass',Count:4);
15
16   Writeln;
17    Total := 0;
18    for Count := 1 to 10 do begin (* Example 2 *)
19          Total := Total + 12;
20          Write('Count =',Count:3,' Total =',Total:5);
21          Writeln;
22    end;
23
24    Writeln;
25    Write('The alphabet is ');
26    for Alphabet := 'A' to 'Z' do(* Example 3 *)
27        Write(Alphabet);
28    Writeln;
29
30    Writeln;
31    for Count := 7 downto 2 do(* Example 4 *)
32        Writeln('Decrementing loop ',Count:3);
33
34 end.
35
36 { Result of execution
37
38 This is a count loop and we are in pass 1
39 This is a count loop and we are in pass 2
40 This is a count loop and we are in pass 3
41 This is a count loop and we are in pass 4
42 This is a count loop and we are in pass 5
43 This is a count loop and we are in pass 6
44 This is a count loop and we are in pass 7
45
46 Count = 1 Total = 12
```

```
47  Count = 2 Total = 24
48  Count = 3 Total = 36
49  Count = 4 Total = 48
50  Count = 5 Total = 60
51  Count = 6 Total = 72
52  Count = 7 Total = 84
53  Count = 8 Total = 96
54  Count = 9 Total = 108
55  Count = 10 Total = 120
56
57  The alphabet is ABCDEFGHIJKLMNOPQRSTUVWXYZ
58
59  Decrementing loop 7
60  Decrementing loop 6
61  Decrementing loop 5
62  Decrementing loop 4
63  Decrementing loop 3
64  Decrementing loop 2
65
66  }
```

```
1   (* Κεφάλαιο 4 - Πρόγραμμα 2 EX02_4.pas*)
2   program Demonstrate_Conditional_Branching;
3
4   var One,Two,Three : integer;
5
6   begin (* main program *)
7     One := 1; (* these are to have some numbers *)
8     Two := 2; (* to use for illustrations *)
9     Three := 3;
10
11     if Three = (One + Two) then (* Example 1 *)
12       Writeln('three is equal to one plus two');
13
14     if Three = 3 then begin (* Example 2 *)
15        Write('three is ');
16        Write('equal to ');
17        Write('one plus two');
```

```
18      Writeln;
19    end;
20
21    if Two = 2 then (* Example 3 *)
22      Writeln('two is equal to 2 as expected')
23    else
24     Writeln('two is not equal to 2... rather strange');
25
26    if Two = 2 then(* Example 4 *)
27      if One = 1 then
28        Writeln('one is equal to one')
29      else
30       Writeln('one is not equal to one')
31    else
32      if Three = 3 then
33        Writeln('three is equal to three')
34      else
35        Writeln('three is not equal to three');
36
37  end. (* main program *)
38
39  { Result of execution
40
41  three is equal to one plus two
42  three is equal to one plus two
43  two is equal to 2 as expected
44  one is equal to one
45
46  }
```

```
1   (* Κεφάλαιο 4 - Πρόγραμμα 3 EX03_4.pas*)
2   program Loops_And_Ifs;
3
4   var Count,Index : integer;
5
6   begin (* Main program *)
7     for Count := 1 to 10 do begin (* Main loop *)
8       if Count < 6 then
```

```
9           Writeln('The loop counter is up to ',Count:4);
10        if Count = 8 then begin
11          for Index := 8 to 12 do begin (* Internal loop *)
12              Write('The internal loop index is ',Index:4);
13              Write(' and the main count is ',Count:4);
14              Writeln;
15          end; (* Internal loop *)
16        end; (* if Count = 8 condition *)
17      end; (* Main loop *)
18  end. (* Main program *)
19
20  { Result of execution
21
22  The loop counter is up to 1
23  The loop counter is up to 2
24  The loop counter is up to 3
25  The loop counter is up to 4
26  The loop counter is up to 5
27  The internal loop index is 8 and the main count is 8
28  The internal loop index is 9 and the main count is 8
29  The internal loop index is 10 and the main count is 8
30  The internal loop index is 11 and the main count is 8
31  The internal loop index is 12 and the main count is 8
32
33  }

1   (* Κεφάλαιο 4 - Πρόγραμμα 4 EX04_4.pas*)
2
(*************************************************************
****)
3   (* Centigrade to Farenheight temperature conversion *)
4   (* *)
5   (* This program generates a list of temperature conversions *)
6   (* with a note at the freezing point of water, and another *)
7   (* note at the boiling point of water. *)
8
(*************************************************************
****)
```

```
9
10  program Temperature_Conversion;
11
12  var Count,Centigrade,Farenheight : integer;
13
14   begin
15      Writeln('Centigrade to farenheight temperature table');
16      Writeln;
17      for Count := -2 to 12 do begin
18          Centigrade := 10*Count;
19          Farenheight := 32 + Centigrade*9 div 5;
20          Write(' C =',Centigrade:5);
21          Write(' F =',Farenheight:5);
22         if Centigrade = 0 then
23            Write(' Freezing point of water');
24         if Centigrade = 100 then
25           Write(' Boiling point of water');
26          Writeln;
27       end;
28  end.
29
30   { Result of execution
31
32   Centigrade to farenheight temperature table
33
34   C = -20 F = -4
35   C = -10 F = 14
36   C = 0 F = 32 Freezing point of water
37   C = 10 F = 50
38   C = 20 F = 68
39   C = 30 F = 86
40   C = 40 F = 104
41   C = 50 F = 122
42   C = 60 F = 140
43   C = 70 F = 158
44   C = 80 F = 176
45   C = 90 F = 194
46   C = 100 F = 212 Boiling point of water
```

```
47  C = 110 F = 230
48  C = 120 F = 248
49
50  }
```

```
1   (* Κεφάλαιο 4 - Πρόγραμμα 5 EX05_4.pas*)
2   program Temperature_Conversion;
3
4   var A1,A2,A3 : integer;
5
6   begin
7       Writeln('Centigrade to farenheight temperature table');
8       Writeln;
9       for A1 := -2 to 12 do begin
10          A2 := 10*A1;
11          A3 := 32 + A2*9 div 5;
12          Write(' C =',A2:5);
13          Write(' F =',A3:5);
14          if A2 = 0 then
15              Write(' Freezing point of water');
16          if A2 = 100 then
17              Write(' Boiling point of water');
18          Writeln;
19        end;
20  end.
21
22  { Result of execution
23
24  Centigrade to farenheight temperature table
25
26  C = -20 F = -4
27  C = -10 F = 14
28  C = 0 F = 32 Freezing point of water
29  C = 10 F = 50
30  C = 20 F = 68
31  C = 30 F = 86
32  C = 40 F = 104
33  C = 50 F = 122
```

```
34  C = 60 F = 140
35  C = 70 F = 158
36  C = 80 F = 176
37  C = 90 F = 194
38  C = 100 F = 212 Boiling point of water
39  C = 110 F = 230
40  C = 120 F = 248
41
42  }
```

```
1   (* Κεφάλαιο 4 - Πρόγραμμα 6 EX06_4.pas*)
2   program Repeat_Loop_Example;
3
4   var Count : integer;
5
6   begin
7      Count := 4;
8      repeat
9         Write('This is in the ');
10        Write('repeat loop, and ');
11        Write('the index is',Count:4);
12        Writeln;
13        Count := Count + 2;
14     until Count = 20;
15     Writeln(' We have completed the loop ');
16  end.
17
18  { Result of execution
19
20  This is in the repeat loop, and the index is 4
21  This is in the repeat loop, and the index is 6
22  This is in the repeat loop, and the index is 8
23  This is in the repeat loop, and the index is 10
24  This is in the repeat loop, and the index is 12
25  This is in the repeat loop, and the index is 14
26  This is in the repeat loop, and the index is 16
27  This is in the repeat loop, and the index is 18
28  We have completed the loop
```

```
29
30  }
```

```
1   (* Κεφάλαιο 4 - Πρόγραμμα 7 EX07_4.pas*)
2   program While_Loop_Example;
3
4    var Counter : integer;
5
6   begin
7        Counter := 4;
8       while Counter < 20 do begin
9           Write('In the while loop, waiting ');
10          Write('for the counter to reach 20. It is',Counter:4);
11          Writeln;
12          Counter := Counter + 2;
13       end;
14  end.
15
16   { Result of execution
17
18   In the while loop, waiting for the counter to reach 20. It is 4
19   In the while loop, waiting for the counter to reach 20. It is 6
20   In the while loop, waiting for the counter to reach 20. It is 8
21   In the while loop, waiting for the counter to reach 20. It is 10
22   In the while loop, waiting for the counter to reach 20. It is 12
23   In the while loop, waiting for the counter to reach 20. It is 14
24   In the while loop, waiting for the counter to reach 20. It is 16
25   In the while loop, waiting for the counter to reach 20. It is 18
26
27   }
```

```
1   (* Κεφάλαιο 4 - Πρόγραμμα 8 EX08_4.pas*)
2   program Demonstrate_Case;
3
4    var Count : integer;
5
6   begin (* main program *)
7       for Count := 0 to 8 do begin
```

```
8          Write(Count:5);
9         case Count of
10              1 : Write('One'); (* Note that these do not have *)
11              2 : Write('Two'); (* to be in consecutive order *)
12              3 : Write('Three');
13              4 : Write('Four');
14              5 : Write('Five');
15        else Write(' This number is not in the allowable list');
16        end;(* of case structure *)
17          Writeln;
18     end; (* of Count loop *)
19  end. (* of main program *)
20
21  { Result of execution
22
23   0 This number is not in the allowable list
24   1 One
25   2 Two
26   3 Three
27   4 Four
28   5 Five
29   6 This number is not in the allowable list
30   7 This number is not in the allowable list
31   8 This number is not in the allowable list
32
33  }
```

```
1   (* Κεφάλαιο 4 - Πρόγραμμα 9 EX09_4.pas*)
2   program A_Bigger_Case_Example;
3
4    var Count : integer;
5    Index : byte;
6
7   begin (* main program *)
8      for Count := 1 to 10 do begin
9            Write(Count:5);
10          case Count of
11                7..9 : Write(' Big Number');
```

```
12              2,4,6 : beginWrite(' Small');
13                          Write(' even');
14                          Write(' number.');
15                end;
16              3 :for Index := 1 to 3 do Write(' Boo');
17              1 :if TRUE then begin Write(' TRUE is True,');
18                          Write('and this is dumb.');
19                      end;
20              elseWrite(' This number is not in the allowable list');
21          end; (* of case structure *)
22          Writeln;
23     end; (* of Count loop *)
24  end. (* of main program *)
25
26  { Result of execution
27
28  1 TRUE is true, and this is dumb.
29  2 Small even number.
30  3 Boo Boo Boo
31  4 Small even number.
32  5 This number is not in the allowable list
33  6 Small even number.
34  7 Big number
35  8 Big number
36  9 Big number
37  10 This number is not in the allowable list
38
39  }
```

Κεφάλαιο 5
Διαδικασίες και Συναρτήσεις

Το περίγραμμα ενός προγράμματος Pascal Με πρόθεση να ορίσουμε σωστά τις διαδικασίες και τις συναρτήσεις, πρέπει να αναφέρουμε κάποιους ορισμούς. Αυτά που ακολουθούν είναι σημαντικές αρχές, γι' αυτό δείξτε μεγάλη προσοχή. Κάθε πρόγραμμα Pascal αποτελείται από 3 βασικά μέρη που μπορούν να οριστούν ως ακολούθως: Επικεφαλίδα (**Program heading**) - Αυτό είναι το πιο εύκολο κομμάτι αφού είναι μόνο μια γραμμή, και υπήρχε σε όλα τα προγράμματά μας μέχρι τώρα. Είναι απλά η γραμμή με τη λέξη **program**, και δεν μπορεί να είναι πιο πολύπλοκη από ότι ήταν μέχρι τώρα στην Turbo Pascal. Τμήμα δηλώσεων (**Declaration Part**) - Αυτό είναι το τμήμα του κώδικα, στο οποίο όλες οι σταθερές και οι μεταβλητές και οι βοηθητικές, καθορισμένες από το χρήστη, διαδικασίες δηλώνονται. Μερικά από τα προγράμματα που έχουμε εξετάσει, είχαν μια ή περισσότερες δηλώσεις μεταβλητών. Αυτά είναι τα μόνα στοιχεία του τμήματος δηλώσεων που έχουμε χρησιμοποιήσει μέχρι τώρα. Υπάρχουν τουλάχιστον 5 υποτμήματα επιμέρους δηλώσεων στο τμήμα δηλώσεων το πέμπτο υποτμήμα επιμέρους δηλώσεων είναι οι διαδικασίες και οι συναρτήσεις, που είναι τα θέματα που θα μας απασχολήσουν σε αυτό το κεφάλαιο. Τα υπόλοιπα υποτμήματα επιμέρους δηλώσεων θα τα εξετάσουμε στο επόμενο κεφάλαιο. Τμήμα εντολών (**Statement Part**) - Αυτό είναι το τελευταίο τμήμα σε οποιοδήποτε πρόγραμμα Pascal, και είναι αυτό που λέμε κυρίως πρόγραμμα. Είναι μια σύνθετη εντολή πλαισιωμένη από τις δεσμευμένες λέξεις **begin** και **end**. Σημειώστε πως καμία δεν είναι προαιρετική αλλά είναι και οι δυο απαραίτητες για την ύπαρξη του προγράμματος. Είναι πολύ σημαντικό να καταλάβετε τους παραπάνω ορισμούς γιατί θα αναφερόμαστε σε αυτούς συνεχώς κατά τη διάρκεια αυτού του κεφαλαίου και στην συνέχεια αυτού του εγχειριδίου. Μετά από αυτήν την εισαγωγή, ας πάμε στο πρώτο μας πρόγραμμα Pascal που θα περιλαμβάνει διαδικασίες. ***Οι πρώτες διαδικασίες*** Εξετάστε το πρόγραμμα **EX01_5.PAS** ως το πρώτο σας πρόγραμμα Pascal με διαδικασία και εμφανίστε το στην οθόνη σας. Θα παρατηρήσετε ότι δεν μοιάζει με τα προγράμματα που έχετε δει μέχρι τώρα γιατί έχει διαδικασίες μέσα στο κώδικα.

Πρόγραμμα ..... **EX01_5.PAS** Ας πάμε πάλι στους ορισμούς. Η πρώτη γραμμή είναι η επικεφαλίδα και δεν εμφανίζει καμία δυσκολία. Το τμήμα δηλώσεων ξεκινά με το τμήμα δηλώσεων μεταβλητών

στην γραμμή 4 και συνεχίζει συμπεριλαμβάνοντας και τις 3 διαδικασίες που τελειώνουν στην γραμμή 19. Οι γραμμές από 21 έως 26 αποτελούν το τμήμα εντολών. Μπορεί να μοιάζει παράξενο ότι εκτελέσιμες εντολές Pascal περιλαμβάνονται στο τμήμα δηλώσεων και όχι στο τμήμα εντολών. Αυτό οφείλεται στον καθορισμό της Pascal και θα αντιληφθήτε γιατί γίνεται αυτό όταν θα έχουμε ολοκληρώσει την μελέτη μας σχετικά με τις διαδικασίες και τις συναρτήσεις. Συνεχίζοντας να εξετάζουμε το **EX01_5.PAS** ας δούμε το τμήμα εντολών. Το πρόγραμμα, εξαιτίας της φύσης της Pascal και των προσεκτικά επιλεγμένων ονομάτων των διαδικασιών, μας λέει καθαρά τι θα κάνει. Θα γράψει μια επικεφαλίδα, 8 μηνύματα και μια πρόταση τελική. Το μόνο πρόβλημα που αντιμετωπίζουμε είναι πως θα γράψει αυτά τα μηνύματα. Εδώ έρχεται το τμήμα δηλώσεων να καθορίσει αυτές τις διεργασίες με λεπτομέρεια. Το τμήμα δηλώσεων έχει τις 3 διαδικασίες που θα καθορίσουν πλήρως τι ακριβώς θα γίνει όταν κληθούν από το κυρίως πρόγραμμα.

Οι δηλώσεις βρίσκονται στο τμήμα δηλώσεων Πρέπει να είναι σαφές σε σας ότι οι δηλώσεις των διαδικασιών πρέπει να βρίσκονται στο τμήμα δηλώσεων του προγράμματος μιας και αυτό ακριβώς κάνουν. Στην περίπτωση μιας μεταβλητής, μια μεταβλητή καθορίζεται, για να χρησιμοποιηθεί αργότερα από το κυρίως πρόγραμμα, και στην περίπτωση της διαδικασίας, η ίδια η διαδικασία καθορίζεται για να χρησιμοποιηθεί αργότερα από το κυρίως πρόγραμμα. Ας πάρουμε αυθαίρετα μια από τις διαδικασίες, την πρώτη, και ας την εξετάσουμε με λεπτομέρεια. Η πρώτη προς εκτέλεση εντολή που συναντάμε στο κυρίως πρόγραμμα είναι στη γραμμή 22 που λέει απλά **Write_A_Header**, ακολουθούμενη από το γνωστό μας πια ερωτηματικό. Αυτό είναι ένα απλό κάλεσμα διαδικασίας. Όταν ο compiler βρίσκει μια τέτοια εντολή, ψάχνει για μια προδηλωμένη διαδικασία με αυτό το όνομα που να μπορεί να εκτελεστεί. Αν βρει μια τέτοια στο τμήμα δηλώσεων, την εκτελεί. Αν όμως δεν βρει θα ψάξει στις βιβλιοθήκες της Pascal για αυτήν. Η εντολές **Write** και η **Writeln** είναι διαδικασίες του συστήματος, και τις έχετε χρησιμοποιήσει αρκετά μέχρι τώρα, άρα οι διαδικασίες δεν σας είναι απόλυτα άγνωστες. Αν ο compiler δεν βρει τίποτα ούτε στις βιβλιοθήκες τότε θα εμφανίσει ένα μήνυμα λάθους. Ανάλογα με την έκδοση που χρησιμοποιείται ο compiler θα ψάξει σε περισσότερες από μια βιβλιοθήκες. Περισσότερα γι' αυτό το θέμα θα πούμε αργότερα σε αυτό το εγχειρίδιο.

Πως να δηλώσεις και να καλέσεις μια διαδικασία Για να καλέσουμε μια διαδικασία, απλά χρειάζεται να επικαλεστούμε το όνομά

της. Για να δηλώσουμε μια απλή διαδικασία, χρησιμοποιούμε την δεσμευμένη λέξη **procedure** ακολουθούμενη από το όνομά της, με ένα ερωτηματικό στο τέλος. Μετά την επικεφαλίδα της διαδικασίας, ακολουθεί το τμήμα δηλώσεων της διαδικασίας ακολουθούμενο από ένα σώμα εντολών το οποίο δεν είναι τίποτα άλλο από μια σύνθετη εντολή πλαισιωμένη από τις δεσμευμένες λέξεις **begin** και **end**. Αυτό είναι ίδιο με το τμήμα εντολών του κυρίως προγράμματος με εξαίρεση το γεγονός ότι η διαδικασία τελειώνει με ερωτηματικό και όχι με τελεία. Οποιεσδήποτε έγκυρες εντολές σε Pascal μπορούν να μπουν ανάμεσα στο **begin** και στο **end**, και στην πραγματικότητα, δεν υπάρχει διαφορά σε αυτό που μπορεί να μπει σε μια διαδικασία από αυτό που μπορεί να μπει στο κυρίως πρόγραμμα. Το πρόγραμμα το οποίο εξετάζουμε δεν θα έκανε τίποτα το διαφορετικό αν αφαιρούσαμε την πρώτη διαδικασία και μεταφέραμε το Writeln που περιέχει, στο τμήμα εντολών, στη θέση του **Write_A_Header**. Αν αυτή η εντολή δεν είναι ξεκάθαρη, πηγαίνετε πίσω και ξαναδιαβάστε τις δυο τελευταίες παραγράφους, έως ότου να τις κατανοήσετε. Οι γραμμές 23 και 24 προκαλούν την κλήση της διαδικασίας **Write_A_Message** 8 φορές, κάθε φορά γράφεται και μια γραμμή στην οθόνη. Είναι αρκετό να ειπωθεί αυτή τη στιγμή, ότι η τιμή της μεταβλητής **Count**, όπως παρουσιάστηκε εδώ, είναι διαθέσιμη παντού, εννοώντας παντού μέσα στο κυρίως πρόγραμμα. Θα ξεκαθαρίσουμε την εμβέλεια των μεταβλητών σύντομα. Τέλος, καλείται η τελευταία διαδικασία, προκαλώντας την εμφάνιση ενός μηνύματος τερματισμού, και η εκτέλεση του προγράμματος τερματίζεται.

Ένας απαραβίαστος κανόνας της Pascal Αναφέραμε έναν απαραβίαστο κανόνα της Pascal και δεν πρέπει φυσικά να παραβιαστεί τώρα! Είχαμε πει σε προηγούμενο κεφάλαιο πως "τίποτα δεν μπορεί να χρησιμοποιηθεί αν δεν έχει δηλωθεί". Όλες λοιπόν οι διαδικασίες πρέπει να δηλώνονται στην αρχή, πριν από κάθε κλήση τους, πράγμα το οποίο διαφαίνεται και από το γεγονός ότι αποτελούν μέρος του τμήματος δηλώσεων και όχι του τμήματος εντολών. Μεταγλωττίστε και τρέξτε το **EX01_5.PAS** για να εξακριβώσετε ότι κάνει αυτό που περιμένατε από αυτό να κάνει. ***Περισσότερες κλήσεις διαδικασιών*** Υποθέτοντας ότι τρέξατε το **EX01_5.PAS** επιτυχώς και κατανοήσατε την έξοδό του, ας πάμε στο παράδειγμα **EX02_5.PAS** και ας το εξετάσουμε.  Πρόγραμμα ..... **EX02_5.PAS** Σε αυτό το πρόγραμμα θα δούμε πως καλούμε μια διαδικασία και παίρνουμε κάποια

δεδομένα για να τα χρησιμοποιήσουμε σ' άλλη διαδικασία. Για να ξεκινήσουμε, σημειώστε ότι υπάρχουν 3 κλήσεις διαδικασιών στο τμήμα εντολών του κυρίως προγράμματος και κάθε μια έχει έναν επιπρόσθετο όρο που δεν περιείχετο στις κλήσεις στο προηγούμενο πρόγραμμα, τη μεταβλητή με όνομα **Index** μέσα σε παρενθέσεις. Αυτός είναι ο τρόπος της Pascal για να περνά την τιμή μιας μεταβλητής ως όρισμα μιας διαδικασίας. Θα σημειώστε ότι η μεταβλητή **Index** καθορίστηκε ως ένας **integer** στο μέρος του τμήματος δηλώσεων. Αφού χρησιμοποιούμε μία μεταβλητή ως όρισμα, όταν επισκεπτόμαστε την διαδικασία **Print_Data_Out**, θα περιμέναμε να υπάρχει μια μεταβλητή για είσοδο, διαφορετικά θα υπήρχε πρόβλημα (**type mismach**). Στην πραγματικότητα, ελέγχοντας την επικεφαλίδα στην γραμμή 7, καθορίζεται ότι πραγματικά περιμένει μια μεταβλητή **integer**, αλλά προτιμά να αποκαλεί την μεταβλητή αυτή ως **Puppy** μέσα στη διαδικασία. Αποκαλώντας την μεταβλητή με διαφορετικό όνομα φαίνεται ότι δεν υπάρχει πρόβλημα όσο το κυρίως πρόγραμμα δεν προσπαθεί να αποκαλεί την μεταβλητή του **Puppy**, και η διαδικασία δεν προσπαθεί να χρησιμοποιήσει το όνομα **Index**. Και τα δυο στην ουσία αναφέρονται στο ίδιο κομμάτι δεδομένων απλά αναφέρονται σ' αυτό με διαφορετικό όνομα. ***Τυπικές και πραγματικές παράμετροι*** Οι παράμετροι που γράφονται εντός των παρενθέσεων στην επικεφαλίδα της συνάρτησης, ονομάζονται τυπικές (**formal**) παράμετροι, ενώ αυτές που γράφονται στο εσωτερικό των παρενθέσεων στο κυρίως πρόγραμμα αναφέρονται ως πραγματικές (**actual**). Παρατηρείστε ότι η επόμενη διαδικασία καλείται με πραγματική παράμετρο την μεταβλητή **index**. Και στις δυο περιπτώσεις, οι διαδικασίες απλά εμφανίζουν την παράμετρο ως είσοδο, και κάθε μια προσπαθεί να αλλάξει την τιμή της παραμέτρου πριν την επιστρέψει στο κυρίως πρόγραμμα. Παρατηρήστε ότι στη μια αυτό πετυχαίνει και στην άλλη όχι. Στο κυρίως πρόγραμμα έχουμε ένα βρόχο, στον οποίο η μεταβλητή **Count** αυξάνεται από το 1 έως το 3 και η Pascal δεν μας επιτρέπει να αλλάξουμε τη μεταβλητή του βρόχου και έτσι κάνουμε ένα αντίγραφο της τιμής της στη γραμμή 21 και το ονομάζουμε **Index**. Μπορούμε τότε να αλλάξουμε την τιμή της **Index** στο κυρίως πρόγραμμα αν το επιθυμούμε. ***Κλήση με την τιμή*** Στην γραμμή 7, η επικεφαλίδα της διαδικασίας δεν περιέχει την δεσμευμένη λέξη **var** μπροστά από την παράμετρο και το πέρασμα της παραμέτρου είναι μόνο προς μια κατεύθυνση γιατί έτσι έχει οριστεί να είναι. Χωρίς την δεσμευμένη λέξη **var** μπροστά από την μεταβλητή **Puppy**, το σύστημα φτιάχνει ένα αντίγραφο της **Index**, και περνά το αντίγραφο στη

διαδικασία, και μπορεί να κάνει οτιδήποτε με την τιμή αυτή, χρησιμοποιώντας το νέο όνομα **Puppy**, αλλά όταν ο έλεγχος επιστρέφει στο κυρίως πρόγραμμα, η αρχική τιμή της μεταβλητής **Index** υπάρχει ακόμα. Το αντίγραφο του **Index** με όνομα **Puppy** είναι καθορισμένο στη διαδικασία, αλλά η πραγματική μεταβλητή με όνομα **Index** παραμένει αναλλοίωτη. Μπορείτε να σκεφτείτε την παράμετρο που περνάμε χωρίς το **var** ως παράμετρο ενός δρόμου. Αυτό ονομάζεται "κλήση με τιμή" επειδή μόνο η τιμή της πραγματικής μεταβλητής περνά στην διαδικασία.

Κλήση με αναφορά Στη γραμμή 13, η δεύτερη διαδικασία έχει τη δεσμευμένη λέξη **var** μπροστά από το επιθυμητό όνομα της μεταβλητής, με όνομα **Cat**, έτσι δεν παίρνει μόνο την τιμή της μεταβλητής, μπορεί και να την τροποποιήσει, και στο τέλος επιστρέφει την τροποποιημένη τιμή στο κυρίως πρόγραμμα. Αντίγραφο δεν φτιάχνεται, αλλά η αυθεντική μεταβλητή με όνομα **Index** περνά πραγματικά στην διαδικασία και η διαδικασία μπορεί να την τροποποιήσει, κατά κάποιο τρόπο επικοινωνώντας με το κυρίως πρόγραμμα. Το επίσημο όνομα της παραμέτρου είναι **cat** στην διαδικασία, που στην πραγματικότητα είναι ένα άλλο όνομα της μεταβλητής **Index** του κυρίου προγράμματος. Μια παράμετρος με το **var** μπροστά της είναι κατά κάποιο τρόπο κατάσταση δυο κατευθύνσεων. Αυτό είναι "κλήση με αναφορά" αφού μια αναφορά στην αυθεντική μεταβλητή περνά στην διαδικασία. ***Κάποια νέα ορολογία*** Για να αναλύσουμε την νέα ορολογία των δυο τελευταίων παραγράφων, το όνομα της παραμέτρου στο προς κλήση πρόγραμμα αναφέρεται ως πραγματική παράμετρος, και το όνομα της παραμέτρου στην διαδικασία αναφέρεται ως τυπική παράμετρος. Στο τελευταίο παράδειγμα, η πραγματική παράμετρος έχει το όνομα **Index** και η τυπική παράμετρος στην διαδικασία το όνομα **cat**. Πρέπει να σημειωθεί ότι αποκαλείται τυπική παράμετρος είτε έχουμε "κλήση της διαδικασίας με αναφορά" είτε "κλήση με την τιμή". Αυτή η ορολογία χρησιμοποιείται και σε πολλές άλλες γλώσσες, όχι μόνο στην Pascal. Όταν τρέχετε αυτό το πρόγραμμα, θα βρείτε ότι η πρώτη διαδικασία δεν είναι σε θέση να επιστρέψει την τιμή 12 πίσω στο κυρίως πρόγραμμα, αλλά η δεύτερη διαδικασία επιτυγχάνει να επιστρέψει την τιμή 35 στο κυρίως πρόγραμμα. Ξοδέψτε όσο χρόνο θέλετε μελετώντας αυτό το πρόγραμμα μέχρι να το καταλάβετε. Πρέπει να σημειωθεί ότι μπορείτε να περάσετε όσες παραμέτρους επιθυμείτε σε μια διαδικασία απλά δημιουργώντας μια λίστα, χωρισμένη με κόμμα στην κλήση και με ερωτηματικό στην επικεφαλίδα της διαδικασίας. Αυτό θα παρουσιαστεί στο επόμενο

πρόγραμμα. Μεταγλωττίστε και εκτελέστε το **EX02_5.PAS** και μελετήστε την έξοδό του. Πρέπει να είστε σε θέση να κατανοήσετε όλα τα στοιχεία που εμφανίζονται. Αν δεν γίνει κατανοητό, ξαναδιαβάστε τις τελευταίες παραγράφους. Για να επεκτείνετε τη γνώση σας στην Pascal, μελετήστε το πρόγραμμα **EX03_5.PAS** ως ένα παράδειγμα κλήσης διαδικασίας με περισσότερες από μια μεταβλητές στην κλήση.

Πρόγραμμα ..... **EX03_5.PAS** Κανονικά, θα έπρεπε να ομαδοποιήσετε τις 3 τιμές εισόδου για να κάνετε το πρόγραμμα πιο ευανάγνωστο, αλλά για λόγους παρουσίασης, είναι χωριστές. Παρατηρήστε ότι η μεταβλητή **Fruit** είναι μια μεταβλητή διπλής κατεύθυνσης επειδή είναι η τρίτη μεταβλητή στην λίστα με τις πραγματικές παραμέτρους και αντιστοιχεί στην τρίτη τυπική παράμετρο στην επικεφαλίδα της διαδικασίας. Αφού η τρίτη μεταβλητή είναι μια "κλήση με αναφορά", έχει τη δυνατότητα να επιστρέφει το άθροισμα των άλλων τριών παραμέτρων στο κυρίως πρόγραμμα. Αυτός είναι ο λόγος που καθορίστηκε με αυτόν τον τρόπο. Μεταγλωττίστε και τρέξτε το **EX03_5.PAS** για να δείτε ότι κάνει αυτό που περιμένατε από αυτό να κάνει σύμφωνα με τις παραπάνω εξηγήσεις. ***"Κλήση με αναφορά" ή "Κλήση με την τιμή"*** Θα μπορούσε κανείς να σκεφτεί ότι θα ήταν μια καλή ιδέα να τοποθετήσει απλά την λέξη **var** μπροστά από κάθε τυπική παράμετρο σε κάθε επικεφαλίδα διαδικασίας, για να κερδίσει μέγιστη ευελιξία, αλλά χρησιμοποιώντας όλες τις "κλήσεις με αναφορά" τελικά μπορεί να περιορίσει την ευελιξία του. Υπάρχουν δυο λόγοι για να χρησιμοποιήσεις "κλήση με την τιμή" όταν μπορείς. Το πρώτο είναι απλά για να προστατεύσει κάποια δεδομένα από το να αλλοιωθεί η τιμή τους μέσα στην διαδικασία. Αυτό είναι ένα πολύ σημαντικό θέμα στην κατασκευή λογισμικού γνωστή ως "απόκρυψη πληροφοριών" και είναι η βασική αρχή τουαντικειμενοστραφούς προγραμματισμού, ο οποίος θα συζητηθεί στα κεφάλαια 14 και 15 αυτού του συγγράμματος.
Κατά δεύτερο λόγο είναι η ικανότητα να χρησιμοποιήσουμε μια σταθερά στην κλήση διαδικασίας. Τροποποιήστε την γραμμή 17 του **EX03_5.PAS** ως εξής: **Add_The_Fruit(12,Orange,Fruit,Pear);** και μεταγλωττίστε και τρέξτε το πρόγραμμα. Αφού η μεταβλητή **Value1** είναι μια "κλήση με την τιμή", η σταθερά 12 μπορεί να χρησιμοποιηθεί και το πρόγραμμα μπορεί να μεταγλωττιστεί και να εκτελεστεί. Παρόλα αυτά, αν αλλάξεις την γραμμή 17 σε
: **Add_The_Fruit(Apple,Orange,32,Pear);** θα δείτε ότι δεν θα

εκτελεστεί επειδή η μεταβλητή **Total** συμμετέχει ως "κλήση με αναφορά" και το σύστημα πρέπει να είναι ικανό να επιστρέψει την τιμή στην τυπική παράμετρο **Total**. Δεν μπορεί να το κάνει αυτό επειδή το 32 είναι μια σταθερά και όχι μια μεταβλητή. Μια κλήση με αναφορά πρέπει να έχει μια μεταβλητή ως πραγματική παράμετρο. Ως άσκηση προγραμματισμού, κάντε την **Value1** κλήση αναφοράς προσθέτοντας τη λέξη **var** μπροστά από αυτή στην γραμμή 6, και θα δείτε ότι ο compiler δεν θα σας επιτρέψει να αντικαταστήσετε την μεταβλητή **Apple** με την σταθερά 12 όπως ειπώθηκε νωρίτερα σε αυτήν την παράγραφο. Η προηγούμενη συζήτηση πρέπει να σας έχει πείσει ότι και οι δυο " κλήση με την τιμή" και "κλήση με αναφορά" έχουν μια χρήσιμη θέση στον προγραμματισμό με Pascal και είναι στη θέλησή σας ποια θα χρησιμοποιήσετε. Όταν θα νοιώσετε ότι έχετε καταλάβει τις παραπάνω έννοιες και έχετε μεταγλωττίσει και εκτελέσει το **EX03_5.PAS** αρκετές φορές για να μελετήσετε τα αποτελέσματα των αλλαγών που πραγματοποιούνται, θα μπορέσετε να συνεχίσετε με τη μελέτη της εμβέλειας των μεταβλητών χρησιμοποιώντας το πρόγραμμα **EX04_5.PAS**. ***Μια πολλαπλά καθορισμένη μεταβλητή*** Αν εξετάσετε το πρόγραμμα **EX04_5.PAS**, θα σημειώσετε ότι η μεταβλητή **Count** είναι δηλωμένη διπλά, μια φορά στο τμήμα δηλώσεων του κυρίως προγράμματος και μια στο τμήμα δηλώσεων της διαδικασίας **Print_Some_Data**. Αυτό είναι απόλυτα νόμιμο και είναι μέρος των συμβάσεων της Pascal.

Πρόγραμμα ..... **EX04_5.PAS** Η μεταβλητή **Index** δηλώνεται μόνο στο κυρίως πρόγραμμα και η χρήση της είναι νόμιμη σε ολόκληρο το πρόγραμμα, συμπεριλαμβανομένων και των διαδικασιών και ονομάζεται γενική μεταβλητή (**global variable**). Η μεταβλητή **Count** είναι επίσης δηλωμένη στο κυρίως πρόγραμμα και νόμιμα μπορεί να χρησιμοποιείται παντού, ακόμα και στη συνάρτηση που υπάρχει και άλλη μεταβλητή με το όνομα **Count**. Οι δυο μεταβλητές στην ουσία είναι κάτι το τελείως διαφορετικό, η μία είναι διαθέσιμη μόνο εκτός της διαδικασίας και η άλλη είναι διαθέσιμη μόνο μέσα στη διαδικασία, έτσι είναι αόρατη στο κυρίως πρόγραμμα, αφού είναι καθορισμένη ένα επίπεδο πιο χαμηλά από αυτό του κυρίως προγράμματος. Είναι μόνο διαθέσιμη στο εσωτερικό της συνάρτησης στην οποία είναι ορισμένη. Οποιαδήποτε μεταβλητή είναι διαθέσιμη σε οποιοδήποτε σημείο του προγράμματος σύμφωνα με την δήλωσή της, αλλά μόνο στο ίδιο επίπεδο ή σε χαμηλότερο. Αυτό σημαίνει ότι οποιαδήποτε διαδικασία στο τμήμα δηλώσεων ενός προγράμματος

μπορεί να χρησιμοποιήσει οποιαδήποτε μεταβλητή είναι καθορισμένη στο τμήμα δηλώσεων του προγράμματος. Οποιαδήποτε μεταβλητή έχει δηλωθεί μέσα σε μια διαδικασία δεν μπορεί να χρησιμοποιηθεί από το κυρίως πρόγραμμα από τη στιγμή που η δήλωση είναι σε ένα χαμηλότερο επίπεδο από το κυρίως πρόγραμμα. Τρέξετε το **EX04_5.PAS** πριν συνεχίσετε στο επόμενο παράδειγμα.

Διαδικασίες που καλούν άλλες διαδικασίες Φορτώστε και παρατηρείστε το **EX05_5.PAS** για να δείτε ένα παράδειγμα από διαδικασίες που καλούν άλλες διαδικασίες.

Πρόγραμμα ..... **EX05_5.PAS** Κρατήστε το αυτό στο μυαλό, "τίποτα δεν μπορεί να χρησιμοποιηθεί αν δεν έχει προηγουμένως δηλωθεί", και η ιεραρχία των διαδικασιών θα είναι καθαρή σε αυτό το παράδειγμα. Σημειώστε ότι η διαδικασία **Three** καλεί την διαδικασία **Two** η οποία καλεί την διαδικασία **One**. Τρέξτε το **EX05_5.PAS** και μελετήστε την έξοδο μέχρι να καταλάβετε γιατί εκτυπώνεται κάθε γραμμή και με τον τρόπο που γίνεται. Τώρα που έχετε μια καλή γνώση πάνω στις διαδικασίες, θέλουμε να σημειώσουμε κάτι εξίσου σημαντικό. Θυμηθείτε ότι κάθε πρόγραμμα Pascal αποτελείται από τρία μέρη, την επικεφαλίδα, το τμήμα δηλώσεων και το τμήμα εντολών. Το τμήμα δηλώσεων αποτελείται από 5 μοναδικά μέρη, τα 4 από τα οποία θα τα μελετήσουμε στο επόμενο κεφάλαιο, ενώ το τελευταίο μέρος αποτελείται από τις διαδικασίες και τις συναρτήσεις. Θα καλύψουμε τις συναρτήσεις στο επόμενο παράδειγμα, άρα προς το παρόν δεχτείτε το γεγονός ότι μια συνάρτηση είναι όπως μια διαδικασία. Η διαδικασία λοιπόν, αποτελείται από 3 μέρη, την επικεφαλίδα, το τμήμα δηλώσεων και το τμήμα εντολών. Μια διαδικασία, εξ ορισμού, δεν είναι κάτι διαφορετικό από ένα πρόγραμμα Pascal μαζί με το κυρίως πρόγραμμα, και ένας μεγάλος αριθμός διαδικασιών μπορεί να εντοπιστεί στο τμήμα δηλώσεων του κυρίου προγράμματος. Αυτές οι διαδικασίες είναι όλες στη σειρά, η μία μετά την άλλη. Από τη στιγμή που η διαδικασία έχει οριστεί όπως το κυρίως πρόγραμμα, μοιάζει πιθανό να μπορεί να υπάρχει μια διαδικασία στο τμήμα δηλώσεων μιας διαδικασίας. Αυτό είναι απόλυτα νόμιμο και γίνεται συχνά, αλλά θυμηθείτε ότι η εσωτερική διαδικασία μπορεί να κληθεί μόνο από την διαδικασία που την έχει στο εσωτερικό της, και όχι από το κυρίως πρόγραμμα. Αυτός είναι ένας τρόπος απόκρυψης πληροφοριών που γίνεται δημοφιλείς στην σύγχρονη κατασκευή λογισμικού. Αυτή η τελευταία παράγραφος είναι μάλλον λίγο δύσκολη στην κατανόηση. Μην ανησυχείτε

γι' αυτό τόσο τώρα, μα καθώς θα αποκτάτε εμπειρία προγραμματιστή, θα δείτε πολύ καθαρά πως χρησιμοποιούνται και οι φωλιασμένες (**embedded**) διαδικασίες. ***Τώρα ας κοιτάξουμε μια συνάρτηση*** Και τώρα ας εξετάσουμε το πρόγραμμα με όνομα **EX06_5.PAS** για να δούμε τι είναι μια συνάρτηση και πως την χρησιμοποιούμε.

Πρόγραμμα ..... **EX06_5.PAS**

Σε αυτό το πολύ απλό πρόγραμμα, έχουμε μια συνάρτηση που απλά πολλαπλασιάζει το άθροισμα δυο μεταβλητών με το 4 και επιστρέφει το αποτέλεσμα. Η βασική διαφορά μεταξύ μιας συνάρτησης και μιας διαδικασίας είναι ότι η συνάρτηση επιστρέφει μία μόνη τιμή και καλείται με έναν μαθηματικό τρόπο, σε μια εντολή **writeln**, ή οπουδήποτε αλλού είναι νόμιμο να χρησιμοποιούμε μια μεταβλητή, αφού και η συνάρτηση είναι και αυτή μια μεταβλητή. Παρατηρώντας την επικεφαλίδα της συνάρτησης, στην γραμμή 6, θα σημειώσετε ότι μια συνάρτηση ξεκινά με την δεσμευμένη λέξη **function**. Στη συνέχεια παρατηρούμε ότι οι δυο μεταβλητές μέσα στην παρένθεση καθορίζονται ως τύπου **integer**, και μετά την παρένθεση υπάρχει μια άνω κάτω τελεία και η λέξη **integer**. Η τελευταία αυτή λέξη χρησιμοποιείται για να καθορίσει τι τύπου θα είναι η μεταβλητή που θα επιστραφεί στο κυρίως πρόγραμμα. Οποιαδήποτε κλήση αυτής της συνάρτησης στην πραγματικότητα σημαίνει την επιστροφή μιας τιμής τύπου **Integer** μετά την ολοκλήρωση της κλήσης. Έτσι στη γραμμή 14, η συνάρτηση υπολογίζεται και η επιστρεφόμενη τιμή χρησιμοποιείται στη θέση της συνάρτησης που κλήθηκε. Η τιμή επιστρέφεται αντιστοιχίζοντας την επιστρεφόμενη τιμή στο όνομα της συνάρτησης όπως φαίνεται στη γραμμή 8. Συνεπώς το αποτέλεσμα της συνάρτησης αντιστοιχεί στη μεταβλητή με όνομα **Feet**. Σημειώστε ότι μια συνάρτηση επιστρέφει μια τιμή και ίσως να επιστρέφει και συμπληρωματικές τιμές αν μερικές από τις τυπικές παραμέτρους δηλωθούν ως "κλήσεις με αναφορά". Μεταγλωττίστε και τρέξτε το πρόγραμμα αυτό. ***Τώρα λίγα λόγια όσον αφορά το μυστήριο της αναδρομής*** Ένα από τα μεγαλύτερα μυστήρια της Pascal και πολλών άλλων γλωσσών προγραμματισμού, είναι η αναδρομική κλήση των διαδικασιών. Ένας απλός ορισμός της αναδρομής είναι: *αναδρομή είναι η δυνατότητα μιας διαδικασίας να καλεί τον εαυτό της*. Μελετήστε το παράδειγμα **EX07_5.PAS** ως ένα παράδειγμα αναδρομής.

Πρόγραμμα .....**EX07_5.PAS** Το κυρίως πρόγραμμα είναι

πολύ απλό, καθορίζει την μεταβλητή με όνομα **Count** να έχει τιμή 7 και καλεί την διαδικασία **Print_And_Increment**. Η διαδικασία προτιμά να αναφέρεται στη μεταβλητή με το όνομα **Index** αλλά αυτό δεν μας προκαλεί κανένα πρόβλημα γιατί καταλαβαίνουμε ότι το όνομα της τυπικής παραμέτρου μπορεί να είναι ένα οποιοδήποτε έγκυρο όνομα. Η διαδικασία γράφει μια γραμμή στην οθόνη ακολουθούμενη από την τιμή τής **Index**, και στη συνέχεια μειώνεται η τιμή της μεταβλητής **Index**. Η εντολή **If** παρουσιάζει το ενδιαφέρον κομμάτι αυτού του προγράμματος. Αν η τιμή της **Index** είναι μεγαλύτερη του μηδενός, και τώρα είναι 6, τότε η διαδικασία **Print_And_Increment** καλείται ξανά. Αυτό μοιάζει να δημιουργεί ένα πρόβλημα, εκτός από το γεγονός ότι αυτό είναι απόλυτα έγκυρο στην Pascal. Όταν καλούμε την διαδικασία για δεύτερη φορά, η τιμή της μεταβλητής **Index** εμφανίστηκε ότι είναι 6, και ξανά μειώνεται. Αφού είναι τώρα 5, η ίδια θα κληθεί ξανά, και το ίδιο θα συμβαίνει έως ότου να γίνει μηδέν, οπότε και κάθε συνάρτηση θα τερματιστεί και θα επιστρέψει ο έλεγχος στο κυρίως πρόγραμμα. ***Όσον αφορά τις αναδρομικές συναρτήσεις*** Αυτός είναι ένας πραγματικά κουτός τρόπος να υλοποιείς το συγκεκριμένο πρόγραμμα, αλλά είναι το απλούστερο αναδρομικό παράδειγμα που μπορεί να γραφτεί και το πιο εύκολο στην κατανόηση. Αν θα έχετε περιστασιακή χρήση της αναδρομής, τότε μην την φοβάστε. Θυμηθείτε ότι οι αναδρομικές διαδικασίες πρέπει να έχουν κάποιες μεταβλητές συγκλίνοντες σε κάποια τιμή, αλλιώς θα έχουμε έναν ατελείωτο βρόχο. Μεταγλωττίστε και τρέξτε αυτό το πρόγραμμα και παρατηρείστε την μειούμενη τιμή όσο εφαρμόζεται η αναδρομή. ***Η εκ των προτέρων αναφορά*** Περιστασιακά θα έχετε την ανάγκη να αναφέρεστε σε μια διαδικασία πριν αυτή δηλωθεί. Σε αυτήν την περίπτωση θα χρειαστείτε μια εκ των προτέρων αναφορά. Το πρόγραμμα **EX08_5.PAS** είναι ένα παράδειγμα μιας τέτοιας αναφοράς.

Πρόγραμμα .....**EX08_5.PAS** Σε αυτό το πρόγραμμα κάθε μια από τις διαδικασίες καλεί την άλλη, μια μορφή αναδρομής. Αυτό το πρόγραμμα, όπως και το προηγούμενο, μετρά από το 7 έως το 0 με κουτό τρόπο, αλλά είναι το πιο απλό πρόγραμμα με εκ των προτέρων αναφορά. Η πρώτη διαδικασία, **Write_A_Line**, έχει την επικεφαλίδα της ορισμένη με τον ίδιο ακριβώς τρόπο όπως κάθε άλλη διαδικασία, αλλά σε αντίθεση με το τυπικό σώμα μιας διαδικασίας, μόνο η δεσμευμένη λέξη **forward** γράφεται στη γραμμή 6. Αυτό λέει στον compiler ότι η διαδικασία θα δηλωθεί αργότερα. Η επόμενη διαδικασία καθορίζεται όπως συνήθως,

δηλαδή, το σώμα της **Write_A_Line** καθορίζεται από τη δεσμευμένη λέξη **procedure** και το όνομα της διαδικασίας. Η αναφορά στη μεταβλητή έχει καθοριστεί νωρίτερα. Κατά κάποιο τρόπο, κάθε μια από τις διαδικασίες δηλώνεται πριν κληθούν. Θα ήταν ενδεχόμενο, χρησιμοποιώντας την εκ των προτέρων αναφορά σε μεγάλη έκταση, να τοποθετήσεις το κυρίως πρόγραμμα πριν από όλες τις δηλώσεις διαδικασιών και να έχεις ένα πρόγραμμα δομημένο όπως και σε μερικές άλλες γλώσσες. Το στυλ προγραμματισμού θα ήταν απόλυτα έγκυρο όσον αφορά τον compiler, αλλά το πρόγραμμα θα ήταν αυθαίρετο και μη κατανοήσιμο. Καλά θα κάνετε να μείνετε προσκολλημένοι με τις συμβατικές τεχνικές της Pascal και να μην χρησιμοποιείτε την εκ των προτέρων αναφορά. Μεταγλωττίστε και τρέξτε αυτό το πρόγραμμα. ***Ο τύπος της διαδικασίας*** Μελετήστε το πρόγραμμα με όνομα **EX09_5.PAS**. Σ' αυτό χρησιμοποιείται ένα επιπλέον χαρακτηριστικό που προσέθεσε η Borland ξεκινώντας με την έκδοση 5.0. Σε αυτό το πρόγραμμα, η διαδικασία **Do_Math** καθορίστηκε ως τύπος διαδικασίας στην γραμμή 12, και 3 κανονικές διαδικασίες που καθορίζονται κάθε μια με τις ίδιες τυπικές παραμέτρους.

Πρόγραμμα ..... **EX09_5.PAS**

Στο πρόγραμμα αυτό, αφού η **Do_Math** είναι τύπου διαδικασίας είναι συμβατή με τις άλλες διαδικασίες, μπορεί να αντιστοιχιστεί με κάθε μια από τις άλλες διαδικασίες, και η κλήση της να είναι ισοδύναμη με την κλήση της διαδικασίας που είναι προς το παρόν συνδεδεμένη. Το πρόγραμμα πρέπει να είναι εύκολο στην κατανόηση με τα λίγα σχόλια, άρα πρέπει να αφιερώσετε χρόνο για να το καταλάβετε. Πρέπει να καταλάβετε ότι ο τύπος διαδικασίας χρησιμοποιείται για να καλέσετε πολλές διαφορετικές διαδικασίες. Σημειώστε τα σχόλια στις γραμμές 4 και 5 του προγράμματος. Όταν χρησιμοποιούμε ένα τύπο διαδικασίας ή έναν τύπο συνάρτησης, που είναι το θέμα του επόμενου παραδείγματος, η Turbo Pascal απαιτεί να χρησιμοποιήσετε την ντιρεκτίβα **F+**, που υποχρεώνει το σύστημα να χρησιμοποιήσει πολλές κλήσεις για όλες τις κλήσεις διαδικασιών. Μελετήστε το εγχειρίδιο της Turbo Pascal για να μάθετε περισσότερες λεπτομέρειες για τις ντιρεκτίβες του compiler. Εξετάστε το πρόγραμμα με όνομα **EX10_5.PAS** ως παράδειγμα ενός προγράμματος που χρησιμοποιεί τις ίδιες τεχνικές με το προηγούμενο αλλά χρησιμοποιεί μεταβλητές τύπου συναρτήσεων για το πρόγραμμα. Αυτό το πρόγραμμα πρέπει να είναι εύκολο στο να κατανοήσετε τις λεπτομέρειες της

λειτουργίας του. Ο μόνος κανόνας που απασχολεί τους τύπους διαδικασιών και συναρτήσεων που είναι παρόμοιες, είναι ότι μια τέτοια μεταβλητή (**subprogram type variable**) μπορεί να αντιστοιχιστεί με κάποια διαδικασία ή συνάρτηση αντίστοιχα αν η λίστα των παραμέτρων των τελευταίων είναι ίδια με αυτή των μεταβλητών. Αυτό περιλαμβάνει και τον τύπο της επιστρεφόμενης τιμής για μια συνάρτηση. Μια και η δυνατότητα αυτή είναι μια καινούργια προσθήκη της Turbo Pascal, και δεν έχει χρησιμοποιηθεί πολύ, γι' αυτό να μην ανησυχείτε πολύ αν δεν έχετε αντιληφθεί πλήρως την λειτουργία της.

Πρόγραμμα .....**EX10_5.PAS** ***Ασκήσεις προγραμματισμού*** **1.** Γράψτε ένα πρόγραμμα που να εμφανίζει το όνομά σας, τη διεύθυνσή σας, και τον αριθμό τηλεφώνου σας με ένα διαφορετικό **Writeln** σε διαφορετική διαδικασία. **2.** Προσθέστε μια εντολή στην διαδικασία του προγράμματος **EX07_5.PAS** για να εμφανίσετε την τιμή της **Index** μετά την κλήση στον εαυτό της για να μπορέσετε να δείτε την τιμή να αυξάνεται και τις αναδρομικές κλήσεις να επιστρέφουν στο επόμενο ανώτερο επίπεδο. **3.** Ξαναγράψτε το πρόγραμμα **EX04_4.PAS** τοποθετώντας τους υπολογισμούς μετατροπής των βαθμών Celsius σε βαθμούς Farenheight σε μια συνάρτηση

Προγράμματα κεφαλαίου 5

```
1   (* Κεφάλαιο 5 - Πρόγραμμα 1 EX01_5.pas*)
2   program First_Procedure_Call;
3
4   var Count : integer;
5
6   procedure Write_A_Header;
7     begin
8       Writeln('This is the header');
9     end;
10
11  procedure Write_A_Message;
12  begin
```

```
13     Writeln('This is the message and the count is',Count:4);
14  end;
15
16  procedure Write_An_Ending;
17  begin
18     Writeln('This is the ending message');
19  end;
20
21  begin (* main program *)
22     Write_A_Header;
23     for Count := 1 to 8 do
24        Write_A_Message;
25     Write_An_Ending;
26  end. (* of main program *)
27
28  { Result of execution
29
30  This is the header
31  This is the message and the count is 1
32  This is the message and the count is 2
33  This is the message and the count is 3
34  This is the message and the count is 4
35  This is the message and the count is 5
36  This is the message and the count is 6
37  This is the message and the count is 7
38  This is the message and the count is 8
39  This is the ending message
40
41  }
```

```
1   (* Κεφάλαιο 5 - Πρόγραμμα 2 EX02_5.pas*)
2   program Another_Procedure_Example;
3
4   var  Count : integer;
5          Index : integer;
6
7   procedure Print_Data_Out(Puppy : integer);
8   begin
```

```
9       Writeln('This is the print routine',Puppy:5);
10      Puppy := 12;
11  end;
12
13  procedure Print_And_Modify(var Cat : integer);
14  begin
15      Writeln('This is the print and modify routine',Cat:5);
16      Cat := 35;
17  end;
18
19  begin (* main program *)
20    for Count := 1 to 3 do begin
21        Index := Count;
22        Print_Data_Out(Index);
23        Writeln('Back from the print routine, Index =',Index:5);
24        Print_And_Modify(Index);
25        Writeln('Back from the modify routine, Index =',Index:5);
26        Print_Data_Out(Index);
27        Writeln('Back from print again and the Index =',Index:5);
28        Writeln; (* This is just for formatting *)
29    end;
30  end. (* of main program *)
31
32  { Result of execution
33
34  This is the print routine 1
35  Back from the print routine, Index = 1
36  This is the print and modify routine 1
37  Back from the modify routine, Index = 35
38  This is the print routine 35
39  Back from print again and the Index = 35
40
41  This is the print routine 2
42  Back from the print routine, Index = 2
43  This is the print and modify routine 2
44  Back from the modify routine, Index = 35
45  This is the print routine 35
46  Back from print again and the Index = 35
```

```
47
48  This is the print routine 3
49  Back from the print routine, Index = 3
50  This is the print and modify routine 3
51  Back from the modify routine, Index = 35
52  This is the print routine 35
53  Back from print again and the Index = 35
54
55  }
```

```
1   (* Κεφάλαιο 5 - Πρόγραμμα 3 EX03_5.pas*)
2   program Make_A_Fruit_Salad;
3
4   var Apple,Orange,Pear,Fruit : integer;
5
6   procedure Add_The_Fruit(Value1,Value2 : integer; (* one-way *)
7   var  Total : integer; (* two-way *)
8          Value3 : integer); (* one-way *)
9   begin
10       Total := Value1 + Value2 + Value3;
11  end;
12
13  begin (* main program *)
14      Apple := 4;
15      Orange := 5;
16      Pear := 7;
17      Add_The_Fruit(Apple,Orange,Fruit,Pear);
18      Writeln('The fruit basket contains ',Fruit:3,' fruits');
19  end. (* of main program *)
20
21
22  { Result of execution
23
24  The fruit basket contains 16 fruits
25
26  }
```

```
1   (* Κεφάλαιο 5 - Πρόγραμμα 4 EX04_5.pas*)
```

```
2  program Scope_Of_Variables;
3
4  var  Count : integer;
5         Index : integer;
6
7  procedure Print_Some_Data;
8  var    Count, More_Stuff : integer;
9  begin
10       Count := 7;
11       Writeln('In Print_Some_Data Count =',Count:5,
12       ' Index =',Index:5);
13 end; (* of Print_Some_Data procedure *)
14
15 begin (* Main program *)
16     for Index := 1 to 3 do begin
17         Count := Index;
18         Writeln('In main program Count =',Count:5,
19         ' Index =',Index:5);
20         Print_Some_Data;
21         Writeln('In main program Count =',Count:5,
22         ' Index =',Index:5);
23         Writeln;
24     end; (* Count loop *)
25 end. (* main program *)
26
27
28 { Result of execution
29
30 In main program Count = 1 Index = 1
31 In Print_Some_Data Count = 7 Index = 1
32 In main program Count = 1 Index = 1
33
34 In main program Count = 2 Index = 2
35 In Print_Some_Data Count = 7 Index = 2
36 In main program Count = 2 Index = 2
37
38 In main program Count = 3 Index = 3
39 In Print_Some_Data Count = 7 Index = 3
```

```
40  In main program Count = 3 Index = 3
41
42  }

1   (* Κεφάλαιο 5 - Πρόγραμμα 5 EX05_5.pas*)
2   program Procedure_Calling_A_Procedure;
3
4   procedure One;
5   begin
6       Writeln('This is procedure one');
7   end;
8
9   procedure Two;
10  begin
11      One;
12      Writeln('This is procedure two');
13  end;
14
15  procedure Three;
16  begin
17      Two;
18      Writeln('This is procedure three');
19  end;
20
21  begin (* main program *)
22      One;
23      Writeln;
24      Two;
25      Writeln;
26      Three;
27  end. (* of main program *)
28
29
30  { Result of execution
31
32  This is procedure one
33
34  This is procedure one
```

```
35  This is procedure two
36
37  This is procedure one
38  This is procedure two
39  This is procedure three
40
41  }
```

```
1   (* Κεφάλαιο 5 - Πρόγραμμα 6 EX06_5.pas *)
2   program Example_Function;
3
4   var  Dogs,Cats,Feet : integer;
5
6   function Quad_Of_Sum(Number1,Number2 : integer) : integer;
7   begin
8       Quad_Of_Sum := 4*(Number1 + Number2);
9   end;
10
11  begin (* main program *)
12      Dogs := 4;
13      Cats := 3;
14      Feet := Quad_Of_Sum(Dogs,Cats);
15      Writeln(' There are a total of',Feet:3,' paws.');
16  end. (* of main program *)
17
18
19  { Result of execution
20
21  There are a total of 28 paws.
22
23  }
```

```
1   (* Κεφάλαιο 5 - Πρόγραμμα 7 EX07_5.pas*)
2   program Try_Recursion;
3
4   var Count : integer;
5
6   procedure Print_And_Decrement(Index : integer);
```

```
7  begin
8       Writeln('The value of the index is ',Index:3);
9        Index := Index - 1;
10       if Index > 0 then
11         Print_And_Decrement(Index);
12 end;
13
14 begin (* main program *)
15     Count := 7;
16     Print_And_Decrement(Count);
17 end. (* main program *)
18
19
20 { Result of execution
21
22 The value of the index is 7
23 The value of the index is 6
24 The value of the index is 5
25 The value of the index is 4
26 The value of the index is 3
27 The value of the index is 2
28 The value of the index is 1
29
30 }
```

```
1  (* Κεφάλαιο 5 - Πρόγραμμα 8 EX08_5.pas*)
2  program Forward_Reference_Example;
3
4  var Number_Of_Times : integer;
5
6  procedure Write_A_Line(var Count : integer); forward;
7
8  procedure Decrement(var Index : integer);
9  begin
10      Index := Index - 1;
11      if Index > 0 then
12        Write_A_Line(Index);
13 end;
```

```
14
15  procedure Write_A_Line;
16  begin
17       Writeln('The value of the count is now ',Count:4);
18       Decrement(Count);
19  end;
20
21  begin (* main program *)
22     Number_Of_Times := 7;
23     Decrement(Number_Of_Times);
24     Writeln;
25     Number_Of_Times := 7;
26     Write_A_Line(Number_Of_Times);
27  end. (* of main program *)
28
29  { Result of execution
30
31  The value of the count is now 6
32  The value of the count is now 5
33  The value of the count is now 4
34  The value of the count is now 3
35  The value of the count is now 2
36  The value of the count is now 1
37
38  The value of the count is now 7
39  The value of the count is now 6
40  The value of the count is now 5
41  The value of the count is now 4
42  The value of the count is now 3
43  The value of the count is now 2
44  The value of the count is now 1
45
46  }
```

```
1   (* Κεφάλαιο 5 - Πρόγραμμα 9 EX09_5.pas*)
2   program Procedure_Type_Example;
3
```

```
4  {$F+} (* This forces far calls and is required by TURBO *)
5  (* Pascal to use a procedure type. *)
6
7  type  Procedure_Type = procedure(In1, In2, In3 : integer;
8                                   var Result : integer);
9
10 var  Number1, Number2, Number3 : integer;
11       Final_Result : integer;
12       Do_Math : Procedure_Type;
13
14
15 procedure Add(In1, In2, In3 : integer;
16 var Result : integer);
17 begin
18     Result := In1 + In2 + In3;
19     Writeln('The sum of the numbers is ',Result:6);
20 end;
21
22 procedure Mult(In1, In2, In3 : integer;
23 var Result : integer);
24 begin
25     Result := In1 * In2 * In3;
26     Writeln('The product of the numbers is',Result:6);
27 end;
28
29 procedure Average(In1, In2, In3 : integer;
30 var Result : integer);
31 begin
32     Result := (In1 * In2 * In3) div 3;
33     Writeln('The Average of the numbers is',Result:6);
34 end;
35
36 begin
37    Number1 := 10;
38    Number2 := 15;
39    Number3 := 20;
40
41    Do_Math := Add;
```

```
42     Do_Math(Number1, Number2, Number3, Final_Result);
43
44     Do_Math := Mult;
45     Do_Math(Number1, Number2, Number3, Final_Result);
46
47     Do_Math := Average;
48     Do_Math(Number1, Number2, Number3, Final_Result);
49  end.
50
51
52  { Result of execution
53
54  The sum of the numbers is 45
55  The product of the numbers is 3000
56  The average of the numbers is 1000
57
58  }
```

```
1   (* Κεφάλαιο 5 - Πρόγραμμα 10 EX10_5.pas*)
2   program Function_Type_Example;
3
4   {$F+} (* This forces far calls and is required by TURBO *)
5   (* Pascal to use a function type. *)
6
7   type Function_Type = function(In1, In2, In3 : integer) : integer;
8
9   var  Number1, Number2, Number3 : integer;
10         Final_Result : integer;
11         Do_Math : Function_Type;
12
13
14  function Add(In1, In2, In3 : integer) : integer;
15  var Temp : integer;
16  begin
17      Temp := In1 + In2 + In3;
18      Writeln('The sum of the numbers is ',Temp:6);
19      Add := Temp;
```

```
20  end;
21
22  function Mult(In1, In2, In3 : integer) : integer;
23  var Temp : integer;
24  begin
25     Temp := In1 * In2 * In3;
26     Writeln('The product of the numbers is',Temp:6);
27     Mult := Temp;
28  end;
29
30  function Average(In1, In2, In3 : integer) : integer;
31  var Temp : integer;
32  begin
33      Temp := (In1 * In2 * In3) div 3;
34      Writeln('The Average of the numbers is',Temp:6);
35      Average := Temp;
36  end;
37
38  begin
39     Number1 := 10;
40     Number2 := 15;
41     Number3 := 20;
42
43     Do_Math := Add;
44     Final_Result := Do_Math(Number1, Number2, Number3);
45
46     Do_Math := Mult;
47     Final_Result := Do_Math(Number1, Number2, Number3);
48
49     Do_Math := Average;
50     Final_Result := Do_Math(Number1, Number2, Number3);
51  end.
52
53  { Result of execution
54
55  The sum of the numbers is 45
56  The product of the numbers is 3000
57  The average of the numbers is 1000
```

```
58
59 }
```

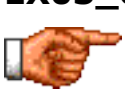
Κεφάλαιο 6
Πίνακες, Τύποι, Σταθερές και Ετικέττες

Πίνακες Στην αρχή αυτού του εγχειριδίου αναφέραμε ότι ένα πρόγραμμα αποτελείται από τα δεδομένα και τις εντολές προς εκτέλεση, που επιτελούν αλλαγές επί των δεδομένων. Μέχρι αυτό το σημείο καλύψαμε σχεδόν όλες τις εκτελέσιμες εντολές, και έτσι μπορούμε να πάμε πίσω και να συμπληρώσουμε κάποια κενά στον ορισμό των δεδομένων και συγκεκριμένα να ασχοληθούμε με τους πίνακες. Μία από τις πιο χρήσιμες δομές δεδομένων στην Pascal είναι ο πίνακας, που με απλά λόγια μπορούμε να πούμε πως είναι μια ομάδα από 2 ή περισσότερα ισοδύναμα στοιχεία, που όλα είναι του ιδίου τύπου. Ας πάμε κατευθείαν σε ένα παράδειγμα για να δούμε πως μοιάζει ένας πίνακας. Ανοίξτε το παράδειγμα **EX01_6.PAS** και σημειώστε την γραμμή 5 που ξεκινά με την λέξη **Automobiles**. Η μεταβλητή αυτή καθορίζεται να είναι τύπου **integer**, αλλά επιπρόσθετα καθορίζεται να έχει 12 διαφορετικές τιμές τις **Automobiles[1**], **Automobiles[2]**, **Automobiles[3]**,..., **Automobiles[12]**.  Πρόγραμμα .....**EX01_6.PAS** Οι αγκύλες χρησιμοποιούνται στην Pascal για να δηλώσουν έναν δείκτη μιας μεταβλητής πίνακα. Η δήλωση μεταβλητής πίνακα που δίνεται στη γραμμή 5 είναι η κλασική δήλωση για έναν πίνακα, δηλαδή δηλώνουμε το όνομα μιας μεταβλητής ακολουθούμενη από το σύμβολο : και την δεσμευμένη λέξη **array**, με το πλήθος των τιμών του πίνακα να δίνεται μέσα σε αγκύλες μετά ακολουθεί η δεσμευμένη λέξη **of** και τέλος δηλώνεται ο τύπος της μεταβλητής που θα έχει κάθε στοιχείο του πίνακα. ***Χρησιμοποιώντας τον πίνακα*** Κατά την χρησιμοποίηση των στοιχείων ενός πίνακα, κάθε στοιχείο του πίνακα απαιτείται να χρησιμοποιείται για τον ίδιο σκοπό και να είναι του ίδιου τύπου. Κάθε φορά που κάποια από τις μεταβλητές του πίνακα χρησιμοποιείται, πρέπει να έχει το μέρος με τις αγκύλες μιας και αυτό είναι τώρα μέρος της ονομασίας. Ο δείκτης, δηλαδή η μεταβλητή που προσδιορίζει κάθε στοιχείο του πίνακα, πρέπει να είναι του τύπου που έχει χρησιμοποιηθεί στην δήλωση και πρέπει η τιμή του να είναι μέσα στα επιτρεπόμενα όρια (αυτά που δηλώθηκαν) αλλιώς θα προκύψει κάποιο λάθος. Τώρα ας ξαναγυρίσουμε στο πρόγραμμα. Καθόσον η μεταβλητή **Index** παίρνει τιμές από 1 έως 12, όση είναι η διάσταση της μεταβλητής **Automobiles**, οι 12 μεταβλητές λαμβάνουν τιμές από 11 έως 22. Αφού αποθηκευτούν οι τιμές στις μεταβλητές, εμφανίζεται ένα μήνυμα και στη συνέχεια εμφανίζονται οι

τιμές του πίνακα. Θα πρέπει να επισημανθεί ότι αν και οι θέσεις του πίνακα ανήκουν στο διάστημα 1 έως 12, οι τιμές που μπορούν να αποθηκευθούν σε κάθε μια από τις 12 μεταβλητές κυμαίνονται στο διάστημα τιμών -32768 έως 32767. Ξανακοιτάξτε αυτό το κομμάτι του προγράμματος, μέχρι να το καταλάβετε πολύ καλά, γιατί αυτό είναι πολύ σημαντικό. Συγκρατήστε ότι ο πίνακας **Automobiles** αποτελείται στην πραγματικότητα από 12 διαφορετικές μεταβλητές τύπου **integer** και μπορούν να χρησιμοποιηθούν με οποιοδήποτε έγκυρο τρόπο μπορεί να χρησιμοποιηθεί μια οποιαδήποτε μεταβλητή ακεραίου τύπου. Μεταγλωττίστε και τρέξτε αυτό το πρόγραμμα. ***Πίνακες δυο διαστάσεων*** Αφού καταλάβατε το προηγούμενο πρόγραμμα, φορτώστε το πρόγραμμα **EX02_6.PAS** για να δείτε το επόμενο επίπεδο πολυπλοκότητας των πινάκων.

Πρόγραμμα .....**EX02_6.PAS** Θα δείτε ότι η μεταβλητή **Checkerboard** είναι ορισμένη ως ένας πίνακας 8 στοιχείων, όπου κάθε στοιχείο της δεν είναι απλού τύπου δεδομένων, αντίθετα κάθε στοιχείο είναι ένας άλλος πίνακας 8 στοιχείων, τύπου **integer**. Έχει συνολικά λοιπόν, η μεταβλητή **Checkerboard** 64 στοιχεία, κάθε ένα από τα οποία είναι τύπου **integer**. Η περίπτωση αυτή ονομάζεται πίνακας δυο διαστάσεων και μπορεί να τον φανταστεί κανείς σαν μια σκακιέρα 8 x 8 θέσεων. Ένας άλλος τρόπος να επιτύχουμε το ίδιο αποτέλεσμα είναι να καθορίσουμε ένα πίνακα δυο διαστάσεων όπως στην επόμενη γραμμή, όπου η **Value** έχει συνολικά 64 στοιχεία. Για να χρησιμοποιήσουμε οποιαδήποτε από τις δυο μεταβλητές στο πρόγραμμα, πρέπει να δηλώσουμε δυο ακόμη μεταβλητές για να πούμε στο πρόγραμμα σε ποίο από τα 64 στοιχεία θέλουμε να αναφερθούμε. Παρατηρώντας το πρόγραμμα θα αποκαλύψουμε δυο βρόχους, ο ένας φωλιασμένος στον άλλο, και οι δυο παίρνουν τιμές από το 1 έως το 8. Οι δείκτες των δυο βρόχων μπορούν να χρησιμοποιηθούν ως δείκτες αναφοράς στα στοιχεία των πινάκων. Η μεταβλητή **Checkerboard** δεικτοδοτείται και από τους δυο δείκτες. Η αποθηκευμένη τιμή δεν έχει καμιά σημασία, απλά πρέπει να δείτε τη διαδικασία. Αφού οι τιμές του **Checkerboard** είναι διαθέσιμες χρησιμοποιούνται για να υπολογιστούν κάποιες τιμές του πίνακα **Value** στη γραμμή 12 του προγράμματος. Αφού καθορίστηκαν όλες οι μεταβλητές, και πρέπει να καταλαβαίνετε ότι συνολικά έχουν οριστεί 128 μεταβλητές στο διπλό βρόχο, 64 για τον πίνακα **Checkerboard** και 64 για τον **Value**, μπορούν να εμφανιστούν. Το επόμενο κομμάτι του

προγράμματος κάνει το εξής, χρησιμοποιεί άλλο ένα διπλό βρόχο, και ένα **Write**. Κάθε φορά που περνάμε από το σώμα του βρόχου του λέμε να μας εμφανίσει μια από τις 64 τιμές του πίνακα **Checkerboard** με τους δείκτες **Index** και **Count** που καθορίζουν ποιο στοιχείο του πίνακα θα εμφανιστεί κάθε φορά. Διαβάστε προσεκτικά το πρόγραμμα ώστε να καταλάβετε επακριβώς τις ενέργειές του. Αφού εμφανίσαμε τον πίνακα που καθορίζεται από την μεταβλητή **Checkerboard** έχουμε ακόμα τον πίνακα που καθορίζεται από τη μεταβλητή **Value** ανέπαφο (στην πραγματικότητα ακόμα έχουμε διαθέσιμο τον πίνακα **Checkerboard** αφού δεν μεταβάλλαμε τα στοιχεία του). Πριν εκτυπώσουμε τον πίνακα **Value**, ας αλλάξουμε μερικά από τα στοιχεία του απλά για να δούμε πως αυτό γίνεται. Ο κώδικας στις γραμμές 24 έως και 26 απλά αλλάζει 3 από τις μεταβλητές για να παρουσιαστεί πως μπορείτε να επιτελέσετε πράξεις σε ολόκληρο τον πίνακα με τη βοήθεια των βρόχων, ή απλά ένα μέρος του πίνακα. Προσέξτε τη γραμμή 26, στην οποία **Value[3,6]** (που έχει την τιμή 3), χρησιμοποιείται ως δείκτης. Αυτό είναι απόλυτα έγκυρο αφού έχει δηλωθεί ως απλή μεταβλητή τύπου **Integer** και έχει τιμή μέσα στα όρια από 1 έως 8, συνθήκες που πρέπει να ισχύουν στην περίπτωσή μας. Το τελευταίο κομμάτι του προγράμματος απλά εκτυπώνει 64 τιμές από τη μεταβλητή **Value** όπως και προηγούμενα. Σημειώστε ότι όταν εκτελείτε το πρόγραμμα, οι 3 τιμές είναι στην πραγματικότητα αλλαγμένες όπως περιμέναμε. ***Οι πίνακες είναι ευέλικτοι*** Μερικές ακόμα λέξεις για τους πίνακες πριν προχωρήσουμε. Οι πίνακες στο τελευταίο πρόγραμμα καθορίστηκαν να είναι τετραγωνικοί, δηλαδή διαστάσεων 8 x 8, αλλά αυτή η επιλογή ήταν τελείως αυθαίρετη. Οι δείκτες των πινάκων επιλέχτηκαν να παίρνουν τιμές από 1 έως 8 αλλά θα μπορούσαν να παίρνουν τιμές από 101 έως 108 ή οποιαδήποτε άλλη τιμή. Και, όπως μπορείτε να μαντέψετε δεν είστε περιορισμένοι σε έναν πίνακα δυο διαστάσεων αλλά μπορείτε να ορίσετε μια μεταβλητή με πολλούς δείκτες για να πετύχετε το στόχο σας. Υπάρχει ένας περιορισμός στον αριθμό των διαστάσεων γιατί πολύ απλά μπορεί να καταναλώσετε όλη τη διαθέσιμη μνήμη ! ***Η δήλωση τύπου*** Τώρα που καταλάβατε τους πίνακες, ας ρίξουμε μια ματιά σε έναν καταλληλότερο τρόπο να τους δηλώσουμε εξετάζοντας το πρόγραμμα **EX03_6.PAS**.



Πρόγραμμα .....**EX03_6.PAS**.

Θα παρατηρήσετε ένα νέο κομμάτι στον κώδικα, που ξεκινά με τη λέξη **type**. Η λέξη αυτή είναι άλλη μια δεσμευμένη λέξη που χρησιμοποιείται για

να ξεκινήσει ένα κομμάτι κώδικα όπου δηλώνονται "από το χρήστη οριζόμενοι τύποι" (**user-defined types**). Ξεκινώντας με τους απλούς προκαθορισμένους τύπους (**predifined**) που μελετήσαμε νωρίτερα, μπορούμε να χτίσουμε τόσους καινούργιους τύπους, όσους χρειαζόμαστε και μπορεί να είναι τόσο πολύπλοκοι όσο επιθυμούμε. Τα 6 ονόματα (από το **Array_Def** μέχρι το **Boat**) στο τμήμα **type** δεν είναι μεταβλητές, αλλά έχουν καθοριστεί να είναι τύποι μεταβλητών και μπορούν να χρησιμοποιηθούν όπως χρησιμοποιούμε τους **integer**, **byte**, **real** κ.λ.π. ***Η Pascal ελέγχει τους τύπους πολύ σχολαστικά*** Αυτό είναι μια πολύ δύσκολη έννοια, αλλά και πολύ σημαντική. Ο compiler της Pascal είναι πολύ ιδιότροπος όσον αφορά τους τύπους των μεταβλητών που χρησιμοποιείτε σε κάποιο πρόγραμμα, κάνοντας πολλούς ελέγχους για να βεβαιωθεί ότι δεν χρησιμοποιήσατε λάθος τύπο σε κάποιο σημείο του προγράμματος. Ας δούμε τους τύπους που χρησιμοποιούνται στο τμήμα δηλώσεων μεταβλητών αυτού του προγράμματος. Σημειώστε ότι ο **Airplane** είναι ένας πίνακας **Dog_Food** και ο **Dog_Food** είναι ένας πίνακας **boolean**, άρα ο **Airplane** είναι πίνακας δυο διαστάσεων, και κάθε στοιχείο του είναι μια μεταβλητή τύπου **boolean**. Αυτές οι δηλώσεις δεν δηλώνουν μεταβλητές αλλά έναν τύπο δεδομένων καθορισμένο από τον χρήστη ο όποιος στη συνέχεια μπορεί να χρησιμοποιηθεί για να ορισθεί στο τμήμα **var** ένας πίνακας με **boolean** μεταβλητές. Αυτό στην πραγματικότητα έγινε από τον ορισμό του **Puppies**, που είναι ένας πίνακας αποτελούμενος από 72 (6 x 12) μεταβλητές τύπου **boolean**. Στο ίδιο σημείο η **Stuff** αποτελείται από 14 μεταβλητές, καθεμία από τις οποίες είναι μια **integer** μεταβλητή. Τα στοιχεία του πίνακα είναι, **Stuff[12]**, **Stuff[13]**, ..., **Stuff[25]**. Σημειώστε επίσης ότι **Stuff2** είναι επίσης καθορισμένος ακριβώς στο ίδιο σημείο και αποτελείται επίσης από 14 μεταβλητές. Μια προσεκτική επιθεώρηση θα δείξει ότι το αναγνωριστικό **Kitties** είναι μια μεταβλητή η οποία έχει τον ίδιο ορισμό με την μεταβλητή **Puppies**. Η δήλωση μεταβλητών του ιδίου τύπου με διαφορετικούς τρόπους δεν θεωρείται καλή προγραμματιστική τεχνική και καλό θα είναι να αποφεύγεται. Αυτό το παράδειγμα, παρέχεται για να παρουσιάσει μερικές από τις πλευρές των τύπων που καθορίζονται απο το χρήστη. Μεταγλωττίστε και τρέξτε το πρόγραμμα.

Η έννοια των τύπων είναι σημαντική; Αν και θα πρέπει να καταναλώνετε χρόνο για την προσεκτική επιλογή των τύπων για τις μεταβλητές που θα χρησιμοποιηθούν από το πρόγραμμα, παρ' όλα αυτά και ο compiler της Pascal θα κάνει αποσφαλμάτωση για σας αφού

είναι ιδιότροπος στη χρήση των μεταβλητών διαφορετικών τύπων. Ο έλεγχος αυτός ίσως να είναι λίγο φτωχός σε σχέση με άλλες γλώσσες όπως η **Modula2** ή η **Ada**, αλλά παραμένει πολύ χρήσιμος. Σε ένα μικρό πρόγραμμα όπως αυτό του παραδείγματος, η αξία του τμήματος δηλώσεων τύπων δεν μπορεί να εκτιμηθεί, αλλά σε ένα πρόγραμμα με πολλές μεταβλητές θα ήταν μεγάλο πλεονέκτημα. Αυτό θα παρουσιαστεί αργότερα. ***Η δήλωση σταθερών*** Εξετάζοντας το παράδειγμα **EX04_6.PAS** βλέπουμε ένα παράδειγμα δήλωσης σταθερών.

Πρόγραμμα .....**EX04_6.PAS** Η δεσμευμένη λέξη **const** δηλώνει την αρχή του τμήματος του προγράμματος που χρησιμοποιείται για να δηλωθούν οι σταθερές που θα μπορούν να χρησιμοποιηθούν σε οποιοδήποτε σημείο του προγράμματος αρκεί να υπόκεινται στους περιορισμούς του αντίστοιχου τύπου. Σε αυτό το παράδειγμα, η **Max_size** δηλώνεται ως σταθερά με την τιμή 12. Αυτή δεν είναι μεταβλητή και δεν μπορεί να μεταβληθεί στο πρόγραμμα. Προς το παρόν αγνοείστε τις δυο επόμενες δηλώσεις σταθερών. Όπως βλέπουμε στις δηλώσεις τύπων, υπάρχουν δυο τύποι δεδομένων καθορισμένοι από το χρήστη, που και οι δυο είναι τύποι πινάκων μεγέθους από 1 έως 12 αφού η **Max_Size** έχει την τιμή 12. Στη συνέχεια στο τμήμα δηλώσεων των μεταβλητών, βρίσκουμε 5 διαφορετικές μεταβλητές, όλες δηλωμένες ως πίνακες με μέγεθος από 1 ως 12 (κάποιοι είναι τύπου **integer** και κάποιοι άλλοι τύπου **char**). Όταν ερχόμαστε στο τμήμα εντολών βλέπουμε ότι απαρτίζεται από έναν μεγάλο βρόχο τον οποίο διατρέχουμε 12 φορές, αφού ο βρόχος εκτελείται **Max_Size** φορές. Στον παραπάνω ορισμό, μοιάζει να μην είναι πλεονέκτημα η χρησιμοποίηση σταθεράς, και δεν είναι, εκτός κι αν βρείτε έναν λόγο να αυξήσετε τις διαστάσεις των πινάκων από 12 σε 18. Αν το κάνετε αυτό, απλά πρέπει να επανακαθορίσετε την τιμή της σταθεράς, και να ξαναμεταγλωττίσετε. Αν δεν υπήρχε η σταθερά θα έπρεπε να είχατε αλλάξει όλες τις τιμές του 12 στο πρόγραμμα με το 18. Και βέβαια αυτό δεν θα ήταν πρόβλημα σε ένα μικρό πρόγραμμα, αλλά σε ένα πρόγραμμα 2000 γραμμών, φανταστείτε να ξεχνάγατε ένα 12 να το κάνετε 18. Θα ήταν ένα καλό πρόγραμμα παραγωγής σκουπιδιών. Αυτό το πρόγραμμα πρέπει να σας δώσει μια καλή ιδέα για το πως μπορούν να χρησιμοποιηθούν οι σταθερές, και καθώς αναπτύσσετε καλές προγραμματιστικές τεχνικές, θα χρησιμοποιείτε τις δηλώσεις σταθερών προς όφελός σας. ***Η σταθερά με τύπο*** Παραβλέψαμε την δεύτερη και την τρίτη δήλωση σταθεράς για ένα πολύ απλό λόγο. Δεν είναι

δηλώσεις σταθεράς. Η Turbo Pascal έχει καθορίσει, ως προσθήκη, την "σταθερά με τύπο" (**typed constant**). Χρησιμοποιώντας τη σύνταξη που βλέπετε, η **Index_Start** δηλώνεται ως μια μεταβλητή τύπου **integer** και αρχικοποιείται στην τιμή 49. Αυτή είναι μια πραγματική μεταβλητή και μπορεί να χρησιμοποιηθεί σαν τέτοια μέσα στο πρόγραμμα. Το ίδιο αποτέλεσμα θα επιτευχθεί και αν απλά δηλώσουμε την **Index_Start** στο τμήμα δηλώσεων και στο τμήμα εντολών της δώσουμε την τιμή 49. Όμοια και η **Check_It_Out** δεν είναι σταθερά αλλά μεταβλητή με αρχική τιμή **TRUE**, . Οι σταθερές με τύπο που καθορίστηκαν στην προηγούμενη παράγραφο έχουν και ένα επιπλέον χαρακτηριστικό, αρχικοποιούνται μόνο μια φορά, όταν το πρόγραμμα φορτώνεται. Ακόμα και όταν χρησιμοποιούνται σε μια διαδικασία ή συνάρτηση έχουν αρχικοποιηθεί και δεν αρχικοποιούνται σε κάθε κλήση. Μην ανησυχείτε πολύ γι' αυτό σε αυτό το σημείο, όταν θα αποκτήσετε εμπειρία, θα μπορείτε να χρησιμοποιήσετε αυτά τα στοιχεία πολύ αποτελεσματικά. ***Η δήλωση ετικέτας*** Το τελευταίο πρόγραμμα αυτού του κεφαλαίου **EX05_6.PAS** θα σας παρουσιάσει τη χρήση των ετικετών.

Πρόγραμμα .....**EX05_6.PAS** Στον ορισμό της Pascal, μια ετικέτα είναι ένας αριθμός από το 0 έως το 9999 και χρησιμοποιείται για να καθορίσει ένα σημείο στο πρόγραμμα στο οποίο θέλετε να μεταβείτε. Όλες οι ετικέτες πρέπει να καθορίζονται στην δήλωση ετικετών του προγράμματος προτού χρησιμοποιηθούν. Τότε μια νέα δεσμευμένη λέξη η **goto** χρησιμοποιείται. Ο καλύτερος τρόπος για να το καταλάβετε είναι να το κοιτάξετε μόνοι σας. Η Turbo Pascal έχει μια προσθήκη για τις ετικέτες. Κάθε έγκυρο αναγνωριστικό, όπως αυτά που χρησιμοποιούνται στις μεταβλητές, μπορεί να χρησιμοποιηθεί ως ετικέτα παράλληλα με τις ετικέτες με τιμές από 0 μέχρι 9999. Αυτό παρουσιάζεται στο πρόγραμμα **EX05_6.PAS**. Μεταγλωττίστε και τρέξτε το πρόγραμμα. ***Ο συμπιεσμένος πίνακας*** Όταν η Pascal αρχικά καθορίστηκε το 1971, πολλοί από τους υπολογιστές της εποχής χρησιμοποιούσαν πολύ μεγάλες λέξεις, π.χ.60 bits ήταν το μέγεθος μιας τυπικής λέξης. Η μνήμη ήταν πολύ ακριβή και οι μεγάλες μνήμες δεν ήταν συνηθισμένες. Ένα πρόγραμμα Pascal που χρησιμοποιούσε πίνακες ήταν αναποτελεσματικό διότι μόνο μια μεταβλητή μπορούσε να αποθηκευτεί σε μια λέξη. Αυτό ελάττωνε το χώρο αποθήκευσης αλλά χρειαζόταν πολύ χρόνο να "αποσυμπιέσεις" τις λέξεις ώστε να πάρεις τα δεδομένα. Ο προγραμματιστής είχε να επιλέξει μεταξύ γρήγορου προγράμματος και

κατανάλωσης μνήμης και οικονομίας στη μνήμη αλλά αργού χρόνου εκτέλεσης. Οι σύγχρονοι μικροϋπολογιστές έχουν και τα δυο βελτιστοποιημένα, μικρή λέξη, συνήθως 16 **bits**, και μεγάλη μνήμη. Ο συμπιεσμένος πίνακας ωστόσο δεν είναι υλοποιήσιμος σε πολλούς compilers και θα παραβλεφθεί κατά τη διάρκεια της μεταγλώττισης. Επίσης και όλες οι εκδόσεις της Turbo Pascal παραβλέπουν έναν συμπιεσμένο πίνακα. ***Μια ακόμα προσθήκη της Turbo Pascal*** Η Standard Pascal, όπως καθορίστηκε από τον **Niclaus Wirth**, απαιτεί τα διάφορα τμήματα στο τμήμα δηλώσεων του προγράμματος να βρίσκονται σε μια συγκεκριμένη σειρά και κάθε ένα να εμφανίζεται μόνο μια φορά. Η σειρά αυτή είναι **label**, **const**, **type**, **var** και τέλος οι διαδικασίες και οι συναρτήσεις. Φυσικά, αν κάποιο δεν χρειαζόταν μπορούσαμε να το παραλείψουμε. Αυτή είναι μια άκαμπτη απαίτηση αλλά χρειαζόταν από τον τυπικό ορισμό της Pascal ώστε να διδάσκονται καλές προγραμματιστικές τεχνικές στους αρχάριους σπουδαστές. Όλες οι εκδόσεις της Turbo Pascal δεν είναι τόσο άκαμπτες όσο η Standard Pascal. Σας επιτρέπετε να τοποθετήσετε τα τμήματα δηλώσεων σε οποιαδήποτε σειρά και όσο συχνά επιθυμείτε ώστε να ορίσετε ότι θέλετε πριν το χρησιμοποιήσετε, κάτι το οποίο είναι και ο απαραβίαστος όρος της Pascal. Μερικές φορές έχει νόημα να δηλώσεις μερικές μεταβλητές αμέσως μετά το **type** για να βρίσκονται αυτά τα δυο κοντά, και μετά κάποιους άλλους τύπους με τις αντίστοιχες μεταβλητές τους. Η Turbo Pascal σας δίνει αυτήν την επιπλέον ευελιξία που μπορείτε να τη χρησιμοποιήσετε προς όφελός σας. ***Ασκήσεις προγραμματισμού*** **1.** Γράψτε ένα πρόγραμμα που να αποθηκεύει τους αριθμούς 201 μέχρι 212 σε έναν πίνακα, μετά εμφανίστε τους στην οθόνη. **2.** Γράψτε ένα πρόγραμμα που να αποθηκεύει σ' έναν πίνακα 10 x 10, τα γινόμενα των δεικτών του, δηλαδή τον πίνακα της προπαίδειας. Εμφανίστε τον πίνακα. **3.** Τροποποιήστε το πρόγραμμα της άσκησης 2 συμπεριλαμβάνοντας μια σταθερά, έτσι ώστε απλά αλλάζοντας την σταθερά, να αλλάζει το μέγεθος πίνακα.

Παραδείγματα κεφαλαίου 6

```
1  (* Κεφάλαιο 6 - Πρόγραμμα 1 EX01_6.pas*)
2  program Simple_Arrays;
3
```

```
4  var  Count,Index : integer;
5         Automobiles : array[1..12] of integer;
6
7  begin
8     for Index := 1 to 12 do
9        Automobiles[Index] := Index + 10;
10    Writeln('This is the first program with an array');
11    Writeln;
12   for Index := 1 to 12 do
13       Writeln('automobile number',Index:3,' has the value',
14    Automobiles[Index]:4);
15    Writeln;
16    Writeln('End of program');
17 end.
18
19
20 { Result of execution
21
22 This is the first program with an array
23
24 automobile number 1 has the value 11
25 automobile number 2 has the value 12
26 automobile number 3 has the value 13
27 automobile number 4 has the value 14
28 automobile number 5 has the value 15
29 automobile number 6 has the value 16
30 automobile number 7 has the value 17
31 automobile number 8 has the value 18
32 automobile number 9 has the value 19
33 automobile number 10 has the value 20
34 automobile number 11 has the value 21
35 automobile number 12 has the value 22
36
37 End of program
38
39 }

1  (* Κεφάλαιο 6 - Πρόγραμμα 2 EX02_6.pas*)
```

```
2   program Multiple_Arrays;
3
4   var Index,Count : integer;
5         Checkerboard : array[1..8] of array[1..8] of integer;
6         Value : array[1..8,1..8] of integer;
7
8   begin (* Main program *)
9      for Index := 1 to 8 do begin (* index loop *)
10        for Count := 1 to 8 do begin
11           Checkerboard[Index,Count] := Index + 3*Count;
12           Value[Index,Count] := Index +
2*Checkerboard[Index,Count];
13        end;
14     end; (* of index loop *)
15
16     Writeln(' Output of checkerboard');
17     Writeln;
18    for Index := 1 to 8 do begin
19        for Count := 1 to 8 do
20           Write(Checkerboard[Index,Count]:7);
21       Writeln;
22     end;
23
24     Value[3,5] := -1; (* change some of the value matrix *)
25     Value[3,6] := 3;
26     Value[Value[3,6],7] := 2; (* This is the same as writing
27     Value[3,7] := 2; *)
28     for Count := 1 to 3 do
29        Writeln; (* Three blank lines *)
30     Writeln('Output of value');
31     Writeln;
32     for Count := 1 to 8 do begin
33        for Index := 1 to 8 do
34          Write(Value[Count,Index]:7);
35        Writeln;
36     end;
37  end. (* of main program *)
38
```

```
39
40  { Result of execution
41
42  Output of checkerboard
43
44  4 7 10 13 16 19 22 25
45  5 8 11 14 17 20 23 26
46  6 9 12 15 18 21 24 27
47  7 10 13 16 19 22 25 28
48  8 11 14 17 20 23 26 29
49  9 12 15 18 21 24 27 30
50  10 13 16 19 22 25 28 31
51  11 14 17 20 23 26 29 32
52
53
54  Output of value
55
56  9 15 21 27 33 39 45 51
57  12 18 24 30 36 42 48 54
58  15 21 27 33 -1 3 2 57
59  18 24 30 36 42 48 54 60
60  21 27 33 39 45 51 57 63
61  24 30 36 42 48 54 60 66
62  27 33 39 45 51 57 63 69
63  30 36 42 48 54 60 66 72
64
65  }
```

```
1   (* Κεφάλαιο 6 - Πρόγραμμα 3 EX03_6.pas*)
2   program Example_Of_Types;
3
4   type Array_Def = array[12..25] of integer;
5          Char_Def = array[0..27] of char;
6          Real_Array = array[-17..42] of real;
7          Dog_Food = array[1..6] of boolean;
8          Airplane = array[1..12] of Dog_Food;
9          Boat = array[1..12,1..6] of boolean;
10
```

```
11  var  Index,Counter : integer;
12       Stuff : Array_Def;
13       Stuff2 : Array_Def;
14       Puppies : Airplane;
15       Kitties : Boat;
16
17  begin (* main program *)
18     for Index := 1 to 12 do
19        for Counter := 1 to 6 do begin
20           Puppies[Index,Counter] := TRUE;
21           Kitties[Index,Counter] := Puppies[Index,Counter];
22        end;
23       Writeln(Puppies[2,3]:7,Kitties[12,5]:7,Puppies[1,1]:7);
24  end. (* of main program *)
25
26
27  { Result of execution
28
29  TRUE TRUE TRUE
30
31  }
```

```
1   (* Κεφάλαιο 6 - Πρόγραμμα 4 EX04_6.pas*)
2   program Example_Of_Constants;
3
4   const   Max_Size = 12; (* Pascal assumes this is a byte type, but it
5   can be used as an integer also *)
6            Index_Start : integer = 49; (* This is a typed constant *)
7            Check_It_Out : boolean = TRUE; (* Another typed constant
*)
8
9   type     Bigarray = array[1..Max_Size] of integer;
10           Chararray = array[1..Max_Size] of char;
11
12  var      Airplane : Bigarray;
13           Seaplane : Bigarray;
14           Helicopter : Bigarray;
15           Cows : Chararray;
```

```
16          Horses : Chararray;
17          Index : integer;
18
19  begin (* main program *)
20     for Index := 1 to Max_Size do begin
21        Airplane[Index] := Index*2;
22        Seaplane[Index] := Index*3 + 7;
23        Helicopter[Max_Size - Index + 1] := Index + Airplane[Index];
24        Horses[Index] := 'X';
25        Cows[Index] := 'R';
26     end;
27  end. (* of main program *)
28
29
30  { Result of execution
31
32  (There is no output from this program)
33
34  }
```

```
1   (* Κεφάλαιο 6 - Πρόγραμμα 5 EX05_6.pas*)
2   program Label_Illustration;
3
4   label    274,Repeat_Loop,Help,Dog;
5
6   var    Counter : byte; (* This limits us to a maximum of 255 *)
7
8   begin
9      Writeln('Start here and go to "help"');
10     goto Help;
11
12     Dog:
13       Writeln('Now go and end this silly program');
14       goto 274;
15
16     Repeat_Loop:
17        for Counter := 1 to 4 do Writeln('In the repeat loop');
18        goto Dog;
```

```
19
20      Help:
21        Writeln('This is the help section that does nothing');
22         goto Repeat_Loop;
23
24       274:
25         Writeln('This is the end of this spaghetti code');
26  end.
27
28
29  { Result of execution
30
31  Start here and go to "help"
32  This is the help section that does nothing
33  In the repeat loop
34  In the repeat loop
35  In the repeat loop
36  In the repeat loop
37  Now go and end this silly program
38  This is the end of this spaghetti code
39
40  }
```

Κεφάλαιο 7
Αλφαριθμητικά και Διαδικασίες για τη Χρήση τους

Τα αλφαριθμητικά της Pascal Σύμφωνα με τον ορισμό της Pascal, ένα αλφαριθμητικό **(string)** είναι απλά ένας πίνακας από 2 ή και περισσότερα στοιχεία τύπου **char**, και περιλαμβάνονται σε έναν πίνακα ο οποίος δηλώνεται στο τμήμα δηλώσεων μεταβλητών ως πίνακας σταθερού μήκους. Εξετάστε το πρόγραμμα με όνομα **EX01_7.PAS** .

Πρόγραμμα ..... **EX01_7.PAS** Σημειώστε ότι τα αλφαριθμητικά δηλώνονται στο τμήμα **type** αν και μπορούν να δηλωθούν και στο τμήμα δηλώσεων των μεταβλητών **var**. Η δήλωση τύπου έγινε για να αρχίσετε να βλέπετε τη χρήση του **type**. Τα αλφαριθμητικά, όπως ορίστηκαν εδώ, δεν είναι τίποτα άλλο από πίνακες με μεταβλητές τύπου **char**. ***Ένα αλφαριθμητικό είναι ένας πίνακας τύπου char*** Το ενδιαφέρον σημείο αυτού του προγράμματος είναι το εκτελέσιμο πρόγραμμα. Σημειώστε ότι όταν στην μεταβλητή **First_Name** καταχωρείται μια τιμή, η τιμή πρέπει να περιλαμβάνει 10 χαρακτήρες διαφορετικά θα προκύψει λάθος. Αν παραλείψετε ένα κενό, θα έχετε λάθος (**invalid error**). Η Pascal είναι συστηματική στο να σας επιτρέπει να καταχωρείται τιμές σε έναν πίνακα αλφαριθμητικών με τη βοήθεια ενός βρόχου. Για τη συνένωση δεδομένων, λειτουργία που ονομάζεται συνένωση (**concatention**), απαιτείται η χρήση ενός εκτεταμένου βρόχου όπως φαίνεται στο τελευταίο μέρος του προγράμματος. Δυο πράγματα πρέπει να επισημανθούν σε αυτό το πρόγραμμα. Πρώτα, σημειώστε το γεγονός ότι οι διαδικασίες των αλφαριθμητικών είναι πραγματικά διαδικασίες πινάκων και θα ακολουθήσουν όλα τα χαρακτηριστικά που συζητήθηκαν στο προηγούμενο κεφάλαιο. Δεύτερον, είναι πολύ φανερό ότι η Pascal είναι πολύ αδύναμη να διαχειριστεί δεδομένα κειμένου. Θα δούμε στη συνέχεια ότι η Turbo Pascal διαχειρίζεται με μεγάλη ευελιξία κείμενο. Τρέξτε το **EX01_7.PAS** και παρατηρήστε τα αποτελέσματα. ***Ο τύπος string της Turbo Pascal*** Παρατηρήστε το παράδειγμα **EX02_7PAS**.  Πρόγραμμα ..... **EX02_7.PAS** Θα δείτε ένα περισσότερο περιεκτικό πρόγραμμα που στην πραγματικότητα κάνει περισσότερα. Η Turbo Pascal έχει, ως προσθήκη στην Standard Pascal, την μεταβλητή τύπου **string**. Χρησιμοποιείται όπως φαίνεται, στο πρόγραμμα και ο αριθμός μέσα στις

αγκύλες είναι το μέγιστο μέγεθος του **string**. Στην πραγματική χρήση εντός του προγράμματος, η μεταβλητή μπορεί να χρησιμοποιηθεί ως μηδενικού μήκους με κανέναν χαρακτήρα έως και τον μέγιστο αριθμό χαρακτήρων με τον οποίο δηλώθηκε. Η μεταβλητή **First_Name** για παράδειγμα, στην πραγματικότητα έχει 11 θέσεις για αποθήκευση των δεδομένων του. Το παρόν μήκος έχει αποθηκευθεί ως **First_Name[0]** και τα δεδομένα είναι αποθηκευμένα ως **First_Name[1]** μέχρι **First_Name[10]**. Όλα τα δεδομένα είναι αποθηκευμένα ως **byte** μεταβλητές, συμπεριλαμβανομένου και του μήκους, άρα το μήκος περιορίζεται σε ένα μέγιστο 255 χαρακτήρων.

Αλφαριθμητικά με μεταβλητό μήκος Τώρα κοιτάξτε στο πρόγραμμα αυτό καθ' εαυτώ. Ακόμα και όταν η μεταβλητή **First_Name** έχει καθοριστεί να έχει μήκος 10, είναι απόλυτα έγκυρο να καταχωρήσουμε σε αυτήν μια σταθερά 4 χαρακτήρων, με το **First_Name[0]** αυτόματα να παίρνει την τιμή 4 από το σύστημα και οι τελευταίες 6 θέσεις να μην είναι καθορισμένες και άχρηστες. Όταν το πρόγραμμα εκτελείται οι 3 μεταβλητές εκτυπώνονται και φαίνεται ότι οι μεταβλητές είναι μικρότερες από το πλήρες μέγεθός τους όπως αυτό καθορίστηκε στο τμήμα **var**. Χρησιμοποιώντας τον τύπο **string** είναι εύκολο να ενώσετε διαφορετικά αλφαριθμητικά σε ένα όπως μπορεί να φανεί και από την εκχώρηση του **Full_Name**. Ο τελεστής της συνένωσης είναι το σύμβολο της πρόσθεσης και χρησιμοποιείται για να συνενώνει αλφαριθμητικά και ξεχωριστούς χαρακτήρες όπως φαίνεται στη γραμμή 14. Σημειώστε ότι υπάρχουν και δυο κενά, στην σταθερά, που έχουν εισαχθεί ανάμεσα στα τμήματα του πλήρους ονόματος, έτσι όταν εμφανίζεται να είναι διαμορφωμένο καθαρά και αρκετά ευανάγνωστο. Τρέξτε το **EX02_7.PAS** και παρατηρήστε την έξοδο. ***Παρατηρήστε μια μεταβλητή τύπου string*** Το επόμενο παράδειγμα - πρόγραμμα με όνομα **EX03_7.PAS** σκοπό έχει να σας δείξει τι περιλαμβάνεται μέσα σε μια μεταβλητή **string**.  Πρόγραμμα ..... **EX03_7.PAS** Το πρόγραμμα είναι ισοδύναμο με το προηγούμενο εκτός από κάποιες εντολές που προστέθηκαν στο τέλος. Προσέξτε την καταχώρηση στην μεταβλητή **Total**. Η συνάρτηση **Length** είναι διαθέσιμη στην Turbo Pascal και επιστρέφει το προσωρινό μήκος μιας μεταβλητής τύπου **string**. Επιστρέφει μια μεταβλητή τύπου **byte** με την τιμή που περιέχεται στη θέση [0] της μεταβλητής. Εμφανίζουμε τον αριθμό των χαρακτήρων του **string** σε αυτό το σημείο, και μετά εμφανίζουμε κάθε

χαρακτήρα σε μια γραμμή για να δείξουμε ότι το **string** είναι ουσιαστικά ένας πίνακας. Το εγχειρίδιο της Turbo Pascal έχει μια πλήρη αναφορά για όλες τις διαδικασίες και τις συναρτήσεις που είναι διαθέσιμες για την διαχείριση των **strings** μόνο σε αυτήν. Ανατρέξτε λοιπόν, εκεί για περισσότερες λεπτομέρειες. Η χρήση αυτών θα είναι πεντακάθαρη αφού καταλάβετε τα θέματα που καλύφθηκαν εδώ. ***Ασκήσεις προγραμματισμού*** **1.** Γράψτε ένα πρόγραμμα στο οποίο να αποθηκεύετε σε 3 μεταβλητές το όνομα, το όνομα του πατέρα και το επίθετό σας. Μετά να τις εμφανίσετε σε μία γραμμή. Συνενώστε τα ονόματα με κενά ανάμεσα τους και μετά εμφανίστε το πλήρες σας όνομα ως μια και μόνη μεταβλητή.

Παραδείγματα κεφαλαίου 7

```
1   (* Κεφάλαιο 7- Πρόγραμμα 1 EX01_7.pas*)
2   program Pure_Pascal_Strings;
3
4   type   Long_String = array[1..25] of char;
5          String10 = array[1..10] of char;
6          String12 = array[1..12] of char;
7
8   var    First_Name : String10;
9          Initial : char;
10         Last_Name : String12;
11         Full_Name : Long_String;
12         Index : integer;
13
14  begin (* main program *)
15     First_Name := 'John ';
16     Initial := 'Q';
17     Last_Name := 'Doe ';
18     Writeln(First_Name,Initial,Last_Name);
19
20    for Index := 1 to 10 do
21        Full_Name[Index] := First_Name[Index];
22     Full_Name[11] := Initial;
```

```
23    for Index := 1 to 12 do
24        Full_Name[Index + 11] := Last_Name[Index];
25    for Index := 24 to 25 doFull_Name[Index] := ' ';
26        Writeln(Full_Name);
27  end. (* main program *)
28
29  { Result of execution
30
31  John QDoe
32  John QDoe
33
34  }
```

```
1   (* Κεφάλαιο 7- Πρόγραμμα 2 EX02_7.pas*)
2   program TURBO_Pascal_Strings;
3
4   var    First_Name : string[10];
5          Initial : char;
6          Last_Name : string[12];
7          Full_Name : string[25];
8
9   begin (* main program *)
10     First_Name := 'John';
11     Initial := 'Q';
12     Last_Name := 'Doe';
13     Writeln(First_Name,Initial,Last_Name);
14     Full_Name := First_Name + ' ' + Initial + ' ' + Last_Name;
15     Writeln(Full_Name);
16  end. (* main program *)
17
18
19  { Result of execution
20
21  JohnQDoe
22  John Q Doe
23
24  }
```

```
1  (* Κεφάλαιο 7- Πρόγραμμα 3 EX03_7.pas*)
2  program What_Is_In_A_String;
3
4  var    First_Name : string[10];
5         Initial : char;
6         Last_Name : string[12];
7         Full_Name : string[25];
8         Index,Total : integer;
9
10 begin (* main program *)
11    First_Name := 'John';
12    Initial := 'Q';
13    Last_Name := 'Doe';
14    Writeln(First_Name,Initial,Last_Name);
15    Full_Name := First_Name + ' ' + Initial + ' ' + Last_Name;
16    Writeln(Full_Name);
17
18    Total := Length(Full_Name);
19    Writeln('The string contains ',Total:4,' characters');
20    for Index := 1 to Length(Full_Name)do
21       Writeln(Full_Name[Index]);
22    Writeln('End of program');
23 end. (* main program *)
24
25
26 { Result of execution
27
28 JohnQDoe
29 John Q Doe
30 The string contains 10 characters
31 J
32 o
33 h
34 n
35
36 Q
37
38 D
```

```
39 o
40 e
41 End of program
42
43 }
```

Κεφάλαιο 8
Βαθμωτοί Τύποι, Υποπεριοχές και Σύνολα

Οι βαθμωτοί τύποι της Pascal Ένας βαθμωτός τύπος δεδομένων (**scalar**), που ονομάζεται και απαριθμητός (**enumerated**) τύπος δεδομένων, είναι μια λίστα από τιμές τις οποίες μια μεταβλητή αυτού του τύπου μπορεί να πάρει. Μελετήστε το πρόγραμμα **EX01_8.PAS** ως ένα παράδειγμα που κάνει χρήση βαθμωτών τύπων.

Πρόγραμμα ..... **EX01_8.PAS** Ο πρώτος τύπος δηλώσεων καθορίζει ο τύπος **Days** να είναι ένας τύπος που να μπορεί να έχει μια τιμή από 7 διαφορετικές. Επιπλέον στην **Day** μπορεί να αντιστοιχιστεί η τιμή Mon, Tue κ.λ.π. πράγμα το οποίο είναι πιο κατανοητό από τη χρήση του 0 για την Δευτέρα, του 1 για την Τρίτη κ.λ.π. Αυτή η τεχνική κάνει το πρόγραμμα ευκολότερο στην κατανόηση. Εσωτερικά, η Pascal δεν αντιστοιχίζει πραγματικά την τιμή Mon στην μεταβλητή Day, αλλά χρησιμοποιεί μια αναπαράσταση **integer** για όλες τις τιμές. Αυτό είναι σημαντικό να το κατανοήσετε γιατί πρέπει να καταλάβετε ότι δεν μπορείτε να εκτυπώσετε τα Mon, Tue,... αλλά μπορείτε μόνο να τα χρησιμοποιήσετε για να δεικτοδοτήσετε εντολές ελέγχου. Η δεύτερη γραμμή του τμήματος ορισμού **type** καθορίζει την **Time_Of_Day** ως έναν άλλο βαθμωτό τύπο που μπορεί να έχει 4 τιμές, από αυτές που είναι στη λίστα. Η μεταβλητή **Time** μπορεί να λάβει μια από τις τέσσερις τιμές που έχουν καθοριστεί στον τύπο **Time_Of_Day**. Πρέπει να είναι σαφές ακόμα και αν μπορεί να λάβει την τιμή Morning, δεν μπορεί να πάρει την τιμή Morning_Time ή οποιαδήποτε άλλη έκφραση σχετική με το Morning, αφού πρέπει να έχει απόλυτα την ίδια ορθογραφία για να γίνει κατανοητή από τον compiler. Στη συνέχεια, στο τμήμα δηλώσεων των μεταβλητών δηλώθηκαν αρκετές μεταβλητές τύπου **real** για να μας επιτρέψουν να παρουσιάσουμε τη χρήση των μεταβλητών βαθμωτού τύπου. Αφού εμφανίσουμε μια επικεφαλίδα στις γραμμές 16 μέχρι 20 στη συνέχεια οι πραγματικές μεταβλητές αρχικοποιούνται.

Ένας μεγάλος μεταβλητός βαθμωτός βρόχος Το υπόλοιπο του προγράμματος είναι ένα μεγάλο **loop** που ελέγχεται από τη μεταβλητή **Day** και περνάει από όλες τις τιμές της, μία σε κάθε επανάληψη. Σημειώστε ότι ο βρόχος θα μπορούσε να είναι από Tue μέχρι Sat ή ότι άλλο επιθυμείτε και δεν είναι απαραίτητο να χρησιμοποιεί όλες τις τιμές της **Day**. Χρησιμοποιώντας την **Day** ως την μεταβλητή της **Case** , το

όνομα κάθε μιας από τις μέρες τις εβδομάδας εμφανίζεται καθώς διατρέχουμε το βρόχο. Ένας άλλος βρόχος ελεγχόμενος από τη μεταβλητή **Time**, εκτελείται 4 φορές, μια για κάθε τιμή της. Οι δυο συνθήκες του **case** μαζί με το εσωτερικό **loop** χρησιμοποιούνται για να υπολογιστεί η συνολική τιμή για κάθε περίοδο και για κάθε τιμή. Τα δεδομένα είναι προσεκτικά διαμορφωμένα για να σχηματίζουν έναν εμφανίσιμο πίνακα με πληρωμές ως συνάρτηση της μεταβλητής **Time** και της **Day**. Προσέξτε ότι οι μεταβλητές βαθμωτού τύπου δεν συμμετέχουν ποτέ στους υπολογισμούς, και δεν εμφανίζονται ποτέ. Χρησιμοποιούνται μόνο για να ελέγχουν τη ροή της λογικής. Είναι μια καλή προγραμματιστική τεχνική παρά να προσπαθείς να θυμηθείς ότι η Mon αναπαρίσταται με 0, Tue με 1 κ.λ.π. Στην πραγματικότητα, αυτοί οι αριθμοί χρησιμοποιούνται για την εσωτερική παρουσίαση των μεταβλητών αυτών αλλά μπορούμε να ηρεμήσουμε και να αφήσουμε την Pascal να ανησυχεί για αυτά τα θέματα. Μεταγλωττίστε και εκτελέστε το πρόγραμμα παρατηρώντας την έξοδο στην οθόνη.

Ας κοιτάξουμε τι είναι η υποπεριοχή Εξετάστε το πρόγραμμα **EX02_8.PAS** ως ένα παράδειγμα υποπεριοχής (**subrange**) και μερικά συμπληρωματικά για τις μεταβλητές βαθμωτού τύπου .

Πρόγραμμα ..... **EX02_8.PAS** Ίσως είναι σκόπιμο να καθορίσουμε μερικές μεταβλητές που παίρνουν τιμές από μια περιοχή τιμών βαθμωτού τύπου. Σημειώστε ότι η **Days** καθορίζεται να είναι βαθμωτού τύπου όπως και στο προηγούμενο πρόγραμμα, και η **Work** καθορίζεται να έχει ακόμα πιο μικρό πεδίο τιμών. Στη δήλωση **var**, η **Day** καθορίζεται να είναι τύπου **Days** και μπορεί να πάρει ως τιμή οποιαδήποτε από τις μέρες της εβδομάδας. Αλλά η μεταβλητή **Workday**, που είναι τύπου **Work**, μπορεί να λάβει ως τιμές μόνο τις μέρες από Mon ως Fri. Αν υπάρξει μια απόπειρα να δώσετε την τιμή Sat τότε θα εμφανιστεί μήνυμα λάθους. Ένα προσεκτικά γραμμένο πρόγραμμα δεν θα έρθει ποτέ σε τέτοια θέση, και θα ήταν ενδεικτικό αν συνέβαινε κάποιο λάθος, με το πρόγραμμα ή με τα δεδομένα. Αυτό είναι από τα πλεονεκτήματα της Pascal σε σχέση με τις άλλες γλώσσες προγραμματισμού και είναι ένας λόγος για σχολαστικό έλεγχο δήλωσης μεταβλητών σε ένα πρόγραμμα. Μεγαλύτερη επεξήγηση θα δείξει ότι η μεταβλητή **Index** παίρνει τιμές από 1 ως 12. Αν κατά τη διάρκεια της εκτέλεσης του προγράμματος δοθεί άλλη τιμή εκτός αυτής της περιοχής, θα εμφανιστεί μήνυμα λάθους. Ας υποθέσουμε ότι η μεταβλητή αναφερόταν στον αριθμό

των υπαλλήλων σας και αυτό ήταν ακριβώς 12. Αν εσείς λοιπόν επιχειρούσατε να δώσετε την τιμή 27 ή -8 στη μεταβλητή **Index** θα υπήρχε σίγουρα ένα λάθος στα δεδομένα και θα θέλατε να μπορούσατε να σταματήσετε την εκτέλεση του προγράμματος για να διορθώσετε το λάθος. Η Pascal θα σας είχε σώσει από μεγάλο μπελά. ***Μερικές λανθασμένες εντολές*** Με την πρόθεση να έχετε ένα πρόγραμμα που δεν θα έχει λάθη μεταγλώττισης, και ταυτόχρονα να δείχνει κάποια λάθη, ο κώδικας που περιέχεται στις γραμμές από 16 έως 27 είναι σχόλια και δεν πρόκειται να εκτελεστούν. Αυτό είναι ένα τέχνασμα και να το χρησιμοποιήσετε όταν αποσφαλματώνετε ένα πρόγραμμα δηλαδή ένα κομμάτι με προβλήματα μπορεί να γίνει σχόλιο μέχρι να είστε έτοιμοι να το συμπεριλάβετε στο υπόλοιπο. Τα λάθη είναι επεξηγήσιμα από μόνα τους. Υπάρχουν 7 εντολές ανάθεσης ως παραδείγματα χρησιμοποίησης μεταβλητής υποπεριοχής στις γραμμές 29 ως 35. Σημειώστε ότι η μεταβλητή **Day** μπορεί πάντα να αντιστοιχισθεί με την τιμή της **WorkDay** ή της **Weekend**, αλλά το αντίστροφο δεν είναι αληθές επειδή η **Day** μπορεί να πάρει τιμές που δεν επιτρέπονται για τις άλλες. ***Τρεις πολύ χρήσιμες συναρτήσεις*** Οι γραμμές 37 έως 42 του προγράμματος παρουσιάζουν την χρήση τριών πολύ σημαντικών συναρτήσεων για τη χρήση των μεταβλητών βαθμωτού τύπου. Η πρώτη είναι η **Succ** η οποία επιστρέφει την επόμενη τιμή του ορίσματος. Αν είμαστε στην τελευταία τιμή, ένα μήνυμα λάθους εμφανίζεται στην οθόνη. Η επόμενη είναι η **Pred** η οποία επιστρέφει την προηγούμενη τιμή του ορίσματος. Τελευταία η συνάρτηση **Ord** που επιστρέφει την σειρά της τιμής στην μεταβλητή βαθμωτού τύπου. Όλες οι μεταβλητές βαθμωτού τύπου έχουν μια εσωτερική αναπαράσταση που ξεκινά από την τιμή 0 και αυξάνεται κατά 1 μέχρι να φτάσει στην τελική τιμή. Στο παράδειγμά μας, το **Ord(Day)** είναι 5 αν η **Day** έχει αντιστοιχιστεί με το **Sat**, αλλά το **Ord(Weekend)** είναι 0 αν η **Weekend** έχει αντιστοιχιστεί με το **Sat**. Καθώς κερδίζετε εμπειρία, θα καταλάβετε καλύτερα αυτές τις συναρτήσεις. Μερικές ακόμα σκέψεις για τις υποπεριοχές πριν πάμε σε άλλο θέμα. Κάθε υποπεριοχή καθορίζεται πάντα από δυο προκαθορισμένες σταθερές, και πάντα καθορίζεται με αύξουσα διάταξη. Μια μεταβλητή καθορισμένη ως υποπεριοχή είναι ουσιαστικά μια μεταβλητή με περιορισμένο πεδίο τιμών. Η εμπειρία λέει πως πρέπει να χρησιμοποιείται όσο πιο συχνά γίνεται για να αποφεύγεται η παραγωγή σκουπιδιών. Σπάνια υπάρχουν μεταβλητές που δεν έχουν περιορισμένο πεδίο τιμών. Τα όρια μπορούν να δώσουν μια εξήγηση για το τι κάνει το πρόγραμμα και να σας

βοηθήσουν να καταλάβετε τη λειτουργία του. Οι τύποι υποπεριοχής μπορούν να κατασκευαστούν μόνο χρησιμοποιώντας απλούς τύπους όπως **integer**, **char**, **byte** ή βαθμωτούς τύπους. Μεταγλωττίστε και εκτελέστε αυτό το πρόγραμμα ακόμα και αν δεν έχει τίποτα προς εμφάνιση. Προσθέστε μερικές εντολές εμφάνισης για να δείτε μερικές τιμές που καταχωρούνται σε μερικές από τις μεταβλητές. ***Σύνολα*** Τώρα ένα νέο θέμα, τα σύνολα (**sets**). Εξετάζοντας το πρόγραμμα **EX03_8.PAS** θα δούμε κάποια σύνολα.  Πρόγραμμα ..... **EX03_8.PAS** Κατ' αρχήν δηλώνεται το αναγνωριστικό **Goodies** που ορίζει ένα βαθμωτό τύπο. Στη συνέχεια δηλώνεται ένα σύνολο με τις δεσμευμένες λέξεις **set of** ακολουθούμενες από τον προκαθορισμένο βαθμωτό τύπο. Ακολουθούν οι δηλώσεις πολλών μεταβλητών που είναι τύπου **Treat**, αφού μπορούν ατομικά να πάρουν τιμές από όλο το σύνολο. Ας δούμε την μεταβλητή **Ice_Cream_Cone** που έχει οριστεί ως σύνολο από στοιχεία τύπου **Treat**. Αυτή η μεταβλητή αποτελείται από πολλά στοιχεία του **Goodies**. Στο πρόγραμμα, καθορίζουμε ότι αποτελείται από **Ice_Cream** και **Cone**. Το σύνολο **Ice_Cream_Cone** αποτελείται από δυο στοιχεία. Στις γραμμές 21 μέχρι 26, θα δείτε 4 πιο απολαυστικά επιδόρπια ορισμένα ως σύνολα από τα συστατικά. Σημειώστε ότι η **banana_split** αρχικά καθορίζεται μέσω ενός διαστήματος τιμών, και στη συνέχεια ένας άλλος όρος προστίθεται στο σύνολο παρουσιάζοντας πως μπορείτε να προσθέσετε ένα στοιχείο σε ένα σύνολο. Και οι 5 είναι συνενωμένες σε ένα σύνολο με όνομα **Mixed**, μετά αυτό αφαιρείται (εξάγεται) από το συνολικό σύνολο με τα συστατικά για να σχηματίσει το σύνολο με τα συστατικά που δεν χρησιμοποιούνται σε κανένα από τα επιδόρπια. Στη συνέχεια κάθε συστατικό ελέγχεται αν είναι στο σύνολο με τα αχρησιμοποίητα συστατικά ή όχι, και αν είναι εμφανίζεται. Σημειώστε ότι η λέξη **in** είναι και αυτή δεσμευμένη. Εκτελέστε το πρόγραμμα που αναπαριστά τη λίστα με τα αχρησιμοποίητα συστατικά. Σε αυτό το παράδειγμα, καλύτερη προγραμματιστική πρακτική θα είχε εφαρμοστεί αν είχε οριστεί μια νέα μεταβλητή, με πιθανό όνομα **Remaining** για τα αχρησιμοποίητα συστατικά στη γραμμή 32. Θέλαμε να σας δείξουμε ότι η **Mixed** μπορούσε να αποκτήσει τιμή αφαιρούμενη από ολόκληρο το σύνολο, έτσι χρησιμοποιήθηκε αυτό το φτωχό για μεταβλητή όνομα. Όταν μεταγλωττίσετε και εκτελέσετε αυτό το πρόγραμμα θα δείτε ότι τα αποτελέσματα αυτού του παραδείγματος είναι χωρίς σημασία αλλά σας οδηγεί στο να σκεφτείτε το γεγονός ότι τα σύνολα μπορούν να

χρησιμοποιηθούν για απογραφή συμβάντων, πιθανώς ενός πλάνου ενεργειών, ή σε κάποιο άλλο χρήσιμο σύστημα. ***Αναζήτηση με τα σύνολα*** Το πρόγραμμα Pascal **EX04_8.PAS** είναι πιο χρήσιμο από το αμέσως προηγούμενο.  Πρόγραμμα ..... **EX04_8.PAS** Κατ' αρχήν ξεκινάμε με μια μικρή πρόταση και αναζητούμε σε αυτήν όλα τα πεζά γράμματα και δημιουργούμε μια λίστα από αυτά που χρησιμοποιούνται. Αφού χρησιμοποιούμε τμήμα από το πλήρες εύρος των **chars**, δεν χρειάζεται να ορίσουμε βαθμωτό τύπο πριν ορίσουμε το σύνολο. Μπορούμε να ορίσουμε το σύνολο χρησιμοποιώντας την έκφραση **'a'...'z**'. Το σύνολο **Data_Set** είναι αντιστοιχισμένο με την τιμή του κενού συνόλου στην πρώτη γραμμή του προγράμματος, και η μεταβλητή **Print_Group**, είναι το κενό αλφαριθμητικό. Η μεταβλητή **Storage** αντιστοιχίζεται με την πρόταση προς αναζήτηση. Μέσα στο βρόχο ελέγχετε ένας-ένας χαρακτήρας και είναι ή ένας **"non-lower case character"** (μη πεζός χαρακτήρας) ή ως ένας χαρακτήρας που ήδη έχει βρεθεί σε προηγούμενη επανάληψη ή ένας νέος χαρακτήρας που προστίθεται στο σύνολο. Σας αφήνουμε να μελετήσετε τις λεπτομέρειες του προγράμματος, μια και πιστεύουμε ότι δεν θα συναντίσετε κανένα πρόβλημα αφού δεν υπάρχει τίποτα νέο εδώ. Τρέξτε το πρόγραμμα και παρατηρήστε πως μεγαλώνει η λίστα με νέα γράμματα όσο η φράση σαρώνεται. ***Ασκήσεις προγραμματισμού*** **1.** Τροποποιήστε το **EX04_8.PAS** ψάχνοντας για κεφαλαία γράμματα.

Παραδείγματα κεφαλαίου 8

```
1  (* Κεφάλαιο 8 - Πρόγραμμα 1 EX01_8.pas*)
2  program Enumerated_Types;
3
4  type    Days = (Mon,Tue,Wed,Thu,Fri,Sat,Sun);
5            Time_Of_Day = (Morning,Afternoon,Evening,Night);
6
7  var    Day : Days;
8           Time : Time_Of_Day;
9           Regular_Rate : real;
```

```
10          Evening_Premium : real;
11          Night_Premium : real;
12          Weekend_Premium : real;
13          Total_Pay : real;
14
15  begin (* main program *)
16     Writeln('Pay rate table':33);
17     Writeln;
18     Write(' DAY Morning Afternoon');
19     Writeln(' Evening Night');
20     Writeln;
21
22     Regular_Rate := 12.00; (* This is the normal pay rate *)
23     Evening_Premium := 1.10; (* 10 percent extra for working late *)
24     Night_Premium := 1.33; (* 33 percent extra for graveyard *)
25     Weekend_Premium := 1.25; (* 25 percent extra for weekends *)
26
27     for Day := Mon to Sun do begin
28        case Day of
29           Mon : Write('Monday ');
30           Tue : Write('Tuesday ');
31           Wed : Write('Wednesday ');
32           Thu : Write('Thursday ');
33           Fri : Write('Friday ');
34           Sat : Write('Saturday ');
35           Sun : Write('Sunday ');
36        end; (* of case statement *)
37
38        for Time := Morning toNight do begin
39           case Timeof
40              Morning : Total_Pay := Regular_Rate;
41              Afternoon : Total_Pay := Regular_Rate;
42              Evening : Total_Pay := Regular_Rate * Evening_Premium;
43              Night : Total_Pay := Regular_Rate * Night_Premium;
44           end; (* of case statement *)
45
46           caseDay of
47              Sat : Total_Pay := Total_Pay * Weekend_Premium;
```

```
48              Sun : Total_Pay := Total_Pay * Weekend_Premium;
49            end; (* of case statement *)
50
51            Write(Total_Pay:10:2);
52          end; (*of "for Time" loop *)
53          Writeln;
54      end; (* of "for Day"loop *)
55  end. (* of main program *)
56
57  { Result of execution
58
59  Pay rate table
60
61  DAY Morning Afternoon Evening Night
62
63  Monday 12.00 12.00 13.20 15.96
64  Tuesday 12.00 12.00 13.20 15.96
65  Wednesday 12.00 12.00 13.20 15.96
66  Thursday 12.00 12.00 13.20 15.96
67  Friday 12.00 12.00 13.20 15.96
68  Saturday 15.00 15.00 16.50 19.95
69  Sunday 15.00 15.00 16.50 19.95
70
71  }
```

```
1   (* Κεφάλαιο 8 - Πρόγραμμα 2 EX02_8.pas*)
2   program Scaler_Operations;
3
4   type    Days = (Mon,Tue,Wed,Thu,Fri,Sat,Sun);
5           Work = Mon..Fri;
6           Rest = Sat..Sun;
7
8   var     Day : Days; (* This is any day of the week *)
9           Workday : Work; (* These are the the working days *)
10          Weekend : Rest; (* The two weekend days only *)
11          Index : 1..12;
12          Alphabet : 'a'..'z';
```

```
13          Start : 'a'..'e';
14
15  begin (* main program *)
16  (* The following statements are commented out because they contain
17  various errors that will halt compilation.
18
19  Workday := Sat; Sat is not part of Workday's subrange.
20  Rest := Fri; Fri is not part of Weekend's subrange.
21  Index := 13; Index is only allowed to go up to 12,
22  Index := -1; and down to 1.
23  Alphabet := 'A' Alphabet, as defined, includes only the
24  lower case alphabet.
25  Start := 'h' h is not in the first five letters.
26
27  End of commented out section. *)
28
29     Workday := Tue;
30     Weekend := Sat;
31     Day := Workday;
32     Day := Weekend;
33     Index := 3+2*2;
34     Start := 'd';
35     Alphabet := Start;
36     (* since Alphabet is "d" *)
37     Start := Succ(Alphabet); (* Start will be 'e' *)
38     Start := Pred(Alphabet); (* Start will be 'c' *)
39     Day := Wed;
40     Day := Succ(Day); (* Day will now be 'Thu' *)
41     Day := Succ(Day); (* Day will now be 'Fri' *)
42     Index := Ord(Day); (* Index will be 4 (Fri = 4) *)
43  end. (* of main program *)
44
45  { Result of execution
46
47  (There is no output from this porgram.)
48
49  }
```

```
1  (* Κεφάλαιο 8 - Πρόγραμμα 3 EX03_8.pas*)
2  program Define_Some_Sets;
3
4  type     Goodies = (Ice_Cream,Whipped_Cream,Banana,Nuts,Cherry,
5           Choc_Syrup,Strawberries,Caramel,Soda_Water,
6           Salt,Pepper,Cone,Straw,Spoon,Stick);
7
8            Treat = set ofGoodies;
9
10 var     Sundae : Treat;
11         Banana_Split : Treat;
12         Soda : Treat;
13         Ice_Cream_Cone : Treat;
14         Nutty_Buddy : Treat;
15         Mixed : Treat;
16         Index : byte;
17
18 begin
19 (* define all ingredients used in each treat *)
20    Ice_Cream_Cone := [Ice_Cream,Cone];
21    Soda := [Straw,Soda_Water,Ice_Cream,Cherry];
22    Banana_Split := [Ice_Cream..Caramel];
23    Banana_Split := Banana_Split + [Spoon];
24    Nutty_Buddy := [Cone,Ice_Cream,Choc_Syrup,Nuts];
25    Sundae := [Ice_Cream,Whipped_Cream,Nuts,Cherry,Choc_Syrup,
26    Spoon];
27
28 (* combine for a list of all ingredients used *)
29
30    Mixed := Ice_Cream_Cone + Soda + Banana_Split + Nutty_Buddy
+
31    Sundae;
32    Mixed := [Ice_Cream..Stick] - Mixed; (* all ingredients not used *)
33
34    if Ice_Cream in Mixed then Writeln('Ice cream not used');
35    if Whipped_Cream in Mixed then Writeln('Whipped cream not
used');
36    if Banana in Mixed then Writeln('Bananas not used');
```

```
37      if Nuts in Mixed then Writeln('Nuts are not used');
38      if Cherry in Mixed then Writeln('Cherrys not used');
39      if Choc_Syrup in Mixed then Writeln('Chocolate syrup not used');
40      if Strawberries in Mixed then Writeln('Strawberries not used');
41      if Caramel in Mixed then Writeln('Caramel is not used');
42      if Soda_Water in Mixed then Writeln('Soda water is not used');
43      if Salt in Mixed then Writeln('Salt not used');
44      if Pepper in Mixed then Writeln('Pepper not used');
45      if Cone in Mixed then Writeln('Cone not used');
46      if Straw in Mixed then Writeln('Straw not used');
47      if Spoon in Mixed then Writeln('Spoon not used');
48      if Stick in Mixed then Writeln('Stick not used');
49  end.
50
51  { Result of execution
52
53  Salt not used
54  Pepper not used
55  Stick not used
56
57  }
```

```
1   (* Κεφάλαιο 8 - Πρόγραμμα 4 EX04_8.pas*)
2   program Find_All_Lower_Case_Characters;
3
4   const    String_Size = 30;
5
6   type      Low_Set = set of 'a'..'z';
7
8   var     Data_Set : Low_Set;
9           Storage : string[String_Size];
10          Index : 1..String_Size;
11          Print_Group : string[26];
12
13  begin (* main program *)
14      Data_Set := [];
15      Print_Group := '';
16      Storage := 'This is a set test.';
```

```
17
18      for Index := 1 to Length(Storage) do begin
19          if Storage[Index] in['a'..'z'] then begin
20              if Storage[Index] in Data_Set then
21                  Writeln(Index:4,' ',Storage[Index],
22                      ' is already in the set')
23              else begin
24                  Data_Set := Data_Set + [Storage[Index]];
25                  Print_Group := Print_Group + Storage[Index];
26                  Writeln(Index:4,' ',Storage[Index],
27                      ' added to group, complete group = ',
28                              Print_Group);
29              end;
30          end
31          else
32              Writeln(Index:4,' ',Storage[Index],
33                              ' is not a lower case letter');
34      end;
35  end. (* of main program *)
36
37  { Result of execution
38
39   1 T is not a lower case letter
40   2 h added to group, complete group = h
41   3 i added to group, complete group = hi
42   4 s added to group, complete group = his
43   5 is not a lower case letter
44   6 i is already in the set
45   7 s is already in the set
46   8 is not a lower case letter
47   9 a added to group, complete group = hisa
48   10 is not a lower case letter
49   11 s is already in the set
50   12 e added to group, complete group = hisae
51   13 t added to group, complete group = hiseat
52   14 is not a lower case letter
53   15 t is already in the set
54   16 e is already in the set
```

```
55  17 s is already in the set
56  18 t is already in the set
57  19 . is not a lower case letter
58
59  }
```

Κεφάλαιο 9
Εγγραφές

Μια απλή εγγραφή Ερχόμαστε στην πιο σημαντική από όλες τις δομές δεδομένων της Pascal, την εγγραφή (**record**). Μια εγγραφή αποτελείται από έναν αριθμό μεταβλητών κάθε μια από τις οποίες μπορεί να είναι οποιουδήποτε προκαθορισμένου τύπου, και μπορεί να περιλαμβάνει ακόμη και εγγραφές. Είναι προτιμότερο από το να προσπαθούμε να καθορίσουμε μια εγγραφή με λεπτομέρειες, να πάμε στο πρώτο παράδειγμα πρόγραμμα, το **EX01_9.PAS**.  Πρόγραμμα .....**EX01_9.PAS** Με τη βοήθεια αυτού του προγράμματος θα παρουσιάσμε την χρήση της εγγραφής. Στο τμήμα δηλώσεων τύπων (**Type section**) δηλώνεται το αναγνωριστικό **Description** που είναι τύπου εγγραφή. Η εγγραφή αυτή αποτελείται από 3 πεδία, το **Year**, το **Model** και το **Engine**. Σημειώστε ότι τα τρία πεδία είναι διαφορετικού τύπου, αφού ο τύπος εγγραφής μπορεί να περιέχει διαφορετικούς τύπους. Έχετε ένα πλήρες παράδειγμα για το πως δηλώνεται μια εγγραφή. Αποτελείται από το αναγνωριστικό **Description**, το σύμβολο =, τη δεσμευμένη λέξη **record**, τη λίστα των στοιχείων, ακολουθούμενα από τη δεσμευμένη λέξη **end**. Αυτό είναι ένα από τα μέρη της Pascal που το **end** χρησιμοποιείται χωρίς να προηγείται **begin**. Σημειώστε ότι αυτή η δήλωση καθορίζει μόνο έναν τύπο και όχι μια μεταβλητή. Αυτό γίνεται στο τμήμα δηλώσεων όπου η μεταβλητή **Truck** καθορίζεται ως μια εγγραφή με τύπο **Description** και η μεταβλητή **Cars** καθορίζεται να έχει 10 εγγραφές αυτού του τύπου. Η μεταβλητή **Truck** έχει 3 στοιχεία, το **Year**, το **Model** και το **Engine** και κάποιο ή και όλα αυτά τα στοιχεία μπορούν να χρησιμοποιηθούν για αποθήκευση μεταβλητών. Όταν αντιστοιχίζουμε δεδομένα στην μεταβλητή **Truck**, υπάρχουν στην πραγματικότητα 3 μέρη στη μεταβλητή, άρα χρησιμοποιούμε 3 εντολές καταχώρησης, κάθε μια για κάθε υπό-πεδίο. Με σκοπό να αντιστοιχίσουμε τιμές στα διάφορα υπό-πεδία, το όνομα της μεταβλητής ακολουθείται από το όνομα του υπό-πεδίου με μια διαχωριστική τελεία. Ο συνδυασμός "μεταβλητή.υπό-πεδίο" είναι το όνομα μιας μεταβλητής. Κρατήστε στο μυαλό σας ότι η **Truck** είναι μια πλήρης εγγραφή που περιέχει 3 μεταβλητές και για να αναφερθούμε σε κάποια από αυτές θα πρέπει να χρησιμοποιήσουμε το όνομα του υπό-πεδίου. Παρατηρήστε τη γραμμή 16 μέχρι 18 του προγράμματος, όπου στα 3 πεδία καταχωρούνται ασήμαντα δεδομένα με μόνο σκοπό την παρουσίαση. Στο

πεδίο **Year** αντιστοιχίζεται μια **integer** τιμή, στο πεδίο **Model** καταχωρείται η τιμή Pickup και στο **Engine** η τιμή Diesel. Στη συνέχεια χρησιμοποιείται ένας βρόχος για να καταχωρήσουμε τιμές και στις 10 μεταβλητές της **Cars**. Για λόγους παρουσίασης, χρησιμοποιούνται ουσιαστικά 30 μεταβλητές. Μερικές λαμβάνουν τιμές με τυχαίο τρόπο στις γραμμές από 26 μέχρι 30, προσέχοντας να αποθηκεύονται οι σωστές τιμές στους σωστούς τύπους. Τελικά, και οι 10 σύνθετες μεταβλητές, που ουσιαστικά πρόκειται για 30 μεταβλητές οι οποίες είναι ομαδοποιημένες με λογικό τρόπο, εκτυπώνονται χρησιμοποιώντας το όνομα "μεταβλητή.υπό-πεδίο" που περιγράφτηκε νωρίτερα. Αν κάτι στην παραπάνω περιγραφή μιας εγγραφής δεν ξεκαθαρίστηκε στο μυαλό σας, μελετήστε το πολύ προσεκτικά. Η εγγραφή είναι βασική δομή της Pascal, και δεν έχετε καμιά ελπίδα να καταλάβετε την επόμενη παράγραφο αν δεν καταλάβετε αυτή καλά. Βεβαιωθείτε ότι θα μεταγλωττίσετε και θα τρέξετε το **EX01_9.PAS** ώστε να μελετήσετε την έξοδο. ***Η υπέρ-εγγραφή*** Εξετάστε το πρόγραμμα Pascal **EX02_9.PAS** για να δείτε τη δομή μιας πολύ ενδιαφέρουσας εγγραφής.

Πρόγραμμα .....**EX02_9.PAS** Πρώτα έχουμε τη δήλωση μιας σταθεράς. Αγνοείστε την προς το παρόν μιας και θα επανέλθουμε σε αυτό αργότερα. Μέσα στο τμήμα δηλώσεων έχουμε τη δήλωση 3 εγγραφών, και με μεγαλύτερη προσοχή, θα σημειώσετε ότι οι δυο πρώτες εγγραφές συμμετέχουν στον καθορισμό της τρίτης εγγραφής. Η εγγραφή με το όνομα **Person**, ουσιαστικά δηλώνει 9 μεταβλητές, 3 μέσω της εγγραφής με όνομα **Full_Name**, 3 μέσω του δικού της ονόματος, και 3 μέσω της εγγραφής **Date**. Για μια ακόμα φορά, εδώ γίνεται η δήλωση τύπου και δεν δηλώνεται καμία μεταβλητή, αυτό γίνεται στο τμήμα δηλώσεων των μεταβλητών. Το κομμάτι με τις δηλώσεις **var** καθορίζει μερικές μεταβλητές ξεκινώντας με τον πίνακα **Friend** που περιέχει 50 (εξαιτίας της δήλωσης της σταθεράς στο τμήμα **const**) εγγραφές τύπου καθορισμένου από το χρήστη, με όνομα **Person**. Αφού ο τύπος αυτός καθορίζει 9 πεδία, έχουμε καθορίσει μέχρι τώρα 9x50=450 διαφορετικές μεταβλητές, καθεμία με τον δικό της τύπο. Θυμηθείτε ότι η Pascal είναι ιδιότροπη για την αντιστοίχηση δεδομένων με το σωστό τύπο. Κάθε μια από τις 450 ξεχωριστές μεταβλητές έχει τον δικό της τύπο που είναι δεμένος με αυτόν, και ο compiler θα εμφανίσει μήνυμα λάθους αν αντιστοιχίσετε κάποια τιμή με λάθος τύπο μεταβλητής. Αφού το αναγνωριστικό **Person** χρησιμοποιείται για τον καθορισμό τύπου

δεδομένων, μπορεί να χρησιμοποιηθεί για να καθορίσει περισσότερες από μια μεταβλητές, και στην πραγματικότητα χρησιμοποιείται ξανά για να καθορίσει 3 άλλες εγγραφές, τις **Self**, **Mother** και **Father**. Αυτές οι 3 εγγραφές αποτελούνται από 9 μεταβλητές η καθεμιά, έτσι έχουμε 27 επιπλέον μεταβλητές που μπορεί να τις διαχειριστεί το πρόγραμμα. Τέλος έχουμε την μεταβλητή **Index** καθορισμένη ως μια απλή μεταβλητή τύπου **byte**. ***Πως να διαχειριστείτε όλα αυτά τα δεδομένα*** Στο πρόγραμμα ξεκινάμε αντιστοιχίζοντας δεδομένα σε όλα τα πεδία της **Self** στις γραμμές από 31 ως 43. Εξετάζοντας τις 3 πρώτες εντολές του κυρίως προγράμματος, βλέπουμε την κατασκευή που μάθαμε στο τελευταίο παράδειγμα να χρησιμοποιείται εδώ, δηλαδή χρησιμοποιώντας την τελεία ανάμεσα στα ονόματα των πεδίων. Η κύρια εγγραφή ονομάζεται **Self**, και μας ενδιαφέρει το πρώτο της τμήμα, πιο συγκεκριμένα το τμήμα **Name** της εγγραφής τύπου **Person**. Αυτό αποτελείται από 3 μέρη και πρέπει να προσδιορίσουμε για ποιο πεδίο ενδιαφερόμαστε. Άρα η **Self.Name.First_Name** είναι η πλήρης περιγραφή του πρώτου υπο-πεδίου της **Self** και χρησιμοποιείται στη γραμμή 31 όπου καταχωρείται η τιμή "Charley". Τα επόμενα πεδία τα διαχειριζόμαστε με τον ίδιο τρόπο και αυτό είναι αρκετά κατανοητό. ***Τι είναι η εντολή with ;*** Συνεχίζοντας, στο τελευταίο πεδίο, το πεδίο **city**, υπάρχουν μόνο 2 πεδία που απαιτούνται γιατί δεν δηλώθηκε ως τμήμα άλλης εγγραφής. Έτσι, το τέταρτο πεδίο καθορίζεται πλήρως ως **Self.City**. Σημειώστε την εντολή **"with Self do"**. Αυτό είναι μια συντομογραφία που χρησιμοποιείται με τις εγγραφές για να απλοποιούμε των κώδικα. Από το **begin** στην γραμμή 34 στο αντίστοιχο **end** στην γραμμή 43, όλες οι μεταβλητές που ανήκουν στην εγγραφή αυτή χρησιμοποιούνται χωρίς να γραφτεί το **"Self."** μπροστά από αυτές. Απλοποιεί πολύ τον κώδικα η χρήση της εντολής **with**. Θα δείτε ότι οι **City**, **State** και **Zipcode** δέχονται τιμές χωρίς να γράφεται το αναγνωριστικό **Self** πριν από αυτές. Στη συνέχεια για το πεδίο **Day** παρατηρούμε ότι αν και είναι πεδίο τρίτου επιπέδου γράφουμε **"Self.Birthday.Day"** αλλά παραβλέπουμε την **"Self."** αφού βρισκόμαστε ακόμα στην περιοχή του **with** που αυτόματα προσθέτει το **Self** πριν από τα αντίστοιχα πεδία. Για να παρουσιάσουμε καλύτερα την εντολή αυτή, χρησιμοποιούμε και μια ακόμη **with** στη γραμμή 39 μέχρι 42. Σε αυτήν την περιοχή και τα δυο αναγνωριστικά διαχειρίζονται αυτόματα για να απλουστεύσουν τον κώδικα, και η **Month** θα γραφόταν ισοδύναμα **Self.Birthday.Month** αν αφαιρεθούν και τα δυο **with**. ***Σε τι έκταση μπορείτε να χρησιμοποιήσετε φωλιασμένο***

το with ; Θα αναρωτιέστε πόσα επίπεδα φωλιασμένων εγγραφών (**nested records**) επιτρέπονται στις δηλώσεις μιας εγγραφής. Δεν καθορίζεται πουθενά ότι υπάρχει περιορισμός σύμφωνα με τον ορισμό της Pascal, αλλά θα σας δώσουμε μια οδηγία σχετικά μ' αυτό το θέμα. Στην Turbo Pascal, σας επιτρέπετε να έχετε μέχρι και 9 επίπεδα, και δεν θα αξίζει να δοκιμάσετε να φωλιάσετε εντολές **with** βαθύτερα. Κάθε πρόγραμμα που απαιτεί περισσότερα από 9 επίπεδα είναι μακριά από το σκοπό της προγραμματιστικής σας δυνατότητας, και της δικής μου, για πολύ καιρό ακόμη. Μετά την αντιστοίχηση μιας τιμής στη **Year**, ολόκληρη η εγγραφή **Self** έχει λάβει τιμή, δηλαδή συνολικά 9 μεταβλητές. Πρέπει να σημειωθεί ότι αν και η **Self** αποτελείται από 9 μεταβλητές, είναι πιο σωστό να καλούμε την **Self** με το όνομά της γιατί είναι μεταβλητή τύπου εγγραφής. ***Υπέρ-εντολές ανάθεσης τιμών*** Η εντολή στη γραμμή 45, **"Mother:=Self"**, είναι πολύ ενδιαφέρουσα. Επειδή οι δυο είναι εγγραφές του ιδίου τύπου, και οι δυο περιέχουν 9 μεταβλητές και η Pascal είναι αρκετά έξυπνη να το αναγνωρίσει αυτό, και έτσι να αντιστοιχίσει τις τιμές αυτές που περιέχονται στη **Self** στα αντίστοιχα πεδία της **Mother**. Έτσι μέσω μιας εντολής, η εγγραφή **Mother** καθορίστηκε πλήρως. Η εντολή στη γραμμή 46 αντιστοιχίζει τις ίδιες τιμές της **Mother** στις αντίστοιχες μεταβλητές της **Father**, και οι επόμενες δυο γραμμές αντιστοιχίζουν και τις 50 μεταβλητές της **Friend**, τα ίδια δεδομένα. Μέχρι αυτό το σημείο του προγράμματος δημιουργήσαμε 450+27=477 ξεχωριστά κομμάτια δεδομένων. Θα μπορούσαμε να τα εκτυπώσουμε όλα, αλλά αφού είναι δεδομένα χωρίς νόημα, θα ήταν μόνο κατανάλωση χαρτιού. Οι γραμμές 49 μέχρι 52 εμφανίζουν 3 απλά κομμάτια από δεδομένα για δική σας βοήθεια. ***Τι καλό βγαίνει από αυτά*** Πρέπει να σας είναι εμφανές τι κάνει αυτό το πρόγραμμα. Τώρα, ας επιστρέψουμε, στην σταθερά της γραμμής 4 όπως το είχαμε υποσχεθεί. Ο αριθμός των φίλων καθορίστηκε να είναι 50 και χρησιμοποιείται για τον καθορισμό του μεγέθους του πίνακα και στο βρόχο στη γραμμή 47. Μπορείτε τώρα να αλλάξετε αυτόν τον αριθμό και να δείτε πόσο μεγάλος γίνεται ο πίνακας, αλλά θα περιοριστείτε στον αριθμό 1000 επειδή 64K είναι η διαθέσιμη μνήμη για το εκτελέσιμο πρόγραμμα, και όλα αυτά τα δεδομένα έχουν διαθέσιμο χώρο για να αποθηκευτούν 64K. Πρέπει να σημειωθεί ότι η Turbo Pascal επιτρέπει σε ένα πρόγραμμα να είναι μεγαλύτερο από 64K χρησιμοποιώντας τις μονάδες αλλά και πάλι θέτει έναν περιορισμό (των 64K) σε κάθε μονάδα προς μεταγλώττιση. Δείτε πόσο μεγάλος μπορεί να είναι ο αριθμός των φίλων

πριν να διακοπεί το πρόγραμμα από υπερχείλιση στη μνήμη. Σημειώστε αυτόν τον αριθμό, γιατί όταν πάμε στο κεφάλαιο περί "Δεικτών και Δυναμικής κατανομής", θα δείτε μια αξιοσημείωτη αύξηση στο επιτρεπόμενο μέγεθος, ειδικά αν έχετε ένα μεγάλο μέγεθος μνήμης RAM εγκατεστημένο στον υπολογιστή σας. ***Μια διαφορετική μέθοδος*** Αν οποιοδήποτε σημείο από αυτό το κεφάλαιο εξακολουθεί να μην είναι ξεκάθαρο, θα ήταν καλό για σας να γυρίσετε πίσω και να το ξαναδείτε αυτή τη στιγμή. Το επόμενο παράδειγμα θα βάλει σε δοκιμασία το μυαλό σας. Εξετάστε το πρόγραμμα **EX03_9.PAS** ως ένα παράδειγμα με μεταβλητού μήκους εγγραφές.  Πρόγραμμα .... **EX03_9.PAS** Σε αυτό το παράδειγμα, αρχικά καθορίζουμε μια μεταβλητή βαθμωτού τύπου, με όνομα **Kind_Of_Vehicle**, για χρήση στο record. Μετά καθορίζεται ο τύπος **Vehicle**, με σκοπό να ορίσουμε πολλά διαφορετικά οχήματα, κάθε ένα με διαφορετικού είδους δεδομένα. Είναι δυνατό να καθορίσουμε όλες τις μεταβλητές για όλους τους τύπους μηχανών, αλλά θα ήταν χάσιμο αποθηκευτικού χώρου να καθορίσουμε τον αριθμό των ελαστικών για ένα σκάφος, ή τον αριθμό των προπελών που χρησιμοποιούνται σε ένα αυτοκίνητο ή σε ένα φορτηγό. Η εγγραφή μεταβλητού μήκους μας επιτρέπει να καθορίσουμε επακριβώς τα δεδομένα για κάθε όχημα χωρίς να ξοδεύουμε αποθηκευτικό χώρο. ***Τι είναι ένα πεδίο ετικέτας ;*** Στη δήλωση εγγραφής έχουμε την συνηθισμένη επικεφαλίδα ακολουθούμενη από 3 μεταβλητές καθορισμένες με τον ίδιο τρόπο όπως οι εγγραφές στα δυο τελευταία προγράμματα. Τότε ερχόμαστε στις εντολές **case**. Ακολουθώντας αυτήν την εντολή, η εγγραφή είναι διαφορετική για κάθε έναν από τους 4 τύπους που ορίστηκαν κατά τη δήλωση του βαθμωτού τύπου. Η μεταβλητή **What_Kind** καλείται πεδίο ετικέτας (**tag-field**) και πρέπει να δηλωθεί βαθμωτού τύπου πριν από τη δήλωση της εγγραφής. Το πεδίο ετικέτας χρησιμοποιείται για να επιλεγεί το μεταβαλλόμενο μέρος της εγγραφής, όταν το πρόγραμμα χρησιμοποιεί μια από τις μεταβλητές της εγγραφής αυτού του τύπου. Το πεδίο ετικέτα ακολουθείται από το σύμβολο της άνω κάτω τελείας (:) και τον τύπο και μετά ακολουθεί η δεσμευμένη λέξη **of**. Στη συνέχεια παρατίθεται μια λίστα από τα μεταβαλλόμενα μέρη της εγγραφής και για καθένα απ' αυτά δηλώνονται οι αντίστοιχες μεταβλητές. Η λίστα των μεταβλητών για κάθε μεταβαλλόμενο μέρος της εγγραφής καλείται λίστα πεδίων. Μερικοί κανόνες πρέπει να ειπωθούν σε αυτό το σημείο. Τα μεταβαλλόμενα μέρη μιας εγγραφής δεν είναι απαραίτητο να έχουν τον ίδιο αριθμό μεταβλητών

σε κάθε λίστα πεδίων, και στην πραγματικότητα, ένα ή περισσότερα από τα μεταβαλλόμενα μέρη μπορεί να μην έχουν καθόλου μεταβλητές. Στην περίπτωση αυτή, πρέπει να προσθέσουμε ένα ζευγάρι άδειων παρενθέσεων ακολουθούμενες από ένα ερωτηματικό. Όλες οι μεταβλητές σε όλο το κομμάτι αυτό πρέπει να έχουν μοναδικά ονόματα. Οι 3 μεταβλητές **Wheels**, **Tires** και **Tyres**, όλες σημαίνουν το ίδιο πράγμα προς τον χρήστη, αλλά είναι διαφορετικές για τον compiler. Μπορείτε να χρησιμοποιείσετε τα ίδια αναγνωριστικά και σε άλλες εγγραφές και για να δηλώσετε απλές μεταβλητές κάπου αλλού στο πρόγραμμα. Ο compiler μπορεί να διαπιστώσει ποια μεταβλητή εννοείτε κάθε φορά. Ομως η χρησιμοποίηση του ιδίου ονόματος μεταβλητής είναι κακή προγραμματιστική πρακτική, γιατί μπορείτε να μπερδευτείτε εσείς ή κάποιος άλλος που θα θελήσει να διαβάσει το πρόγραμμά σας και να κατανοήσει τι κάνει. Ο τελικός κανόνας είναι ότι το μεταβλητό κομμάτι μιας εγγραφής πρέπει να είναι το τελευταίο μέρος της εγγραφής, και στην πραγματικότητα, το τελευταίο κομμάτι όλων ή κάποιας μεταβλητής εγγραφής μπορεί να έχει μεταβλητό κομμάτι. Αυτό είναι αρκετά πάνω από το δικό μας επίπεδο χρήσης της Pascal για την ώρα τουλάχιστον. ***Χρησιμοποιώντας μεταβαλλόμενη εγγραφή*** Στην γραμμή 22 δηλώνουμε 4 μεταβλητές τύπου **Vehicle**. Μια από τις τέσσερις μεταβλητές τύπου **Vehicle** έχει το όνομα **Ford**. Οι 7 γραμμές που καταχωρούν τιμές στη μεταβλητή **Ford** είναι παρόμοιες με τα προηγούμενα παραδείγματα με μόνη εξαίρεση τη γραμμή 28. Σε αυτή τη γραμμή το πεδίο ετικέτα που επιλέγει το συγκεκριμένο μεταβαλλόμενο μέρος που θα χρησιμοποιηθεί τίθεται ίσο με την τιμή **Truck**, η οποία είναι καθορισμένη ως βαθμωτού τύπου. Αυτό σημαίνει ότι οι μεταβλητές με ονόματα **Motor**, **Tires** και **Payload** είναι διαθέσιμες για χρήση με την εγγραφή **Ford**, αλλά οι μεταβλητές με ονόματα **Wheels**, **Engine**, **tyres**, κ.λ.π. δεν είναι διαθέσιμες στην εγγραφή με όνομα **Ford**. Στη συνέχεια, θα καθορίσουμε την εγγραφή **Sunfish** ως μια βάρκα, και αποδίδουμε τιμές σ' όλες τις μεταβλητές της στις γραμμές 33 μέχρι 41. Όλες οι μεταβλητές της **Sunfish** λαμβάνουν τιμές αλλά με μια τυχαία σειρά για να παρουσιάσουμε ότι δεν χρειάζεται να αποδώσουμε σ' αυτές τιμές με μια συγκεκριμένη σειρά. Πρέπει να θυμάστε την εντολή **with** από το προηγούμενο παράδειγμα. Στη συνέχεια αποδίδουμε εκ νέου τιμή στην **Ford**. Τέλος η μεταβλητή **Mac** γίνεται ίση με την μεταβλητή **Sunfish** στην γραμμή 48. Όλες οι μεταβλητές της εγγραφής **Sunfish** αντιγράφηκαν στην **Mac** συμπεριλαμβανομένων και των πεδίων

ετικετών, κάνοντας τη μεταβλητή **Mac** βάρκα. ***Πως θα δείτε τα περιεχόμενα εγγραφών*** Καθορίσαμε τη μεταβλητή **Ford** να είναι αυτοκίνητο, και οι **Sunfish** και **Mac** να είναι βάρκες. Η **Schwinn** δεν έχει λάβει τιμές και άρα δεν έχει καθόλου δεδομένα, και είναι κατά κάποιο τρόπο άχρηστη. Το πεδίο ετικέτα της **Ford** καθορίστηκε ως **car**, και άρα επαληθεύεται το **if**, και το μήνυμα στην γραμμή 51 θα εμφανισθεί. Η **Sunfish** δεν είναι **bicycle**, έτσι δεν θα εμφανισθεί το μήνυμα. Η **Mac** έχει καθοριστεί ως **boat**, άρα θα εκτυπωθεί ένα μήνυμα. Ακόμα και αν μπορούμε να εκτελέσουμε εντολές ανάθεσης σε εγγραφές, δεν μπορούμε να τις χρησιμοποιήσουμε σε μαθηματικές πράξεις όπως είναι η πρόσθεση, ή ο πολλαπλασιασμός. Χρησιμοποιούνται απλά για αποθήκευση δεδομένων. Βέβαια, οι μεταβλητές μιας εγγραφής μπορούν να χρησιμοποιηθούν σε οποιαδήποτε αριθμητική έκφραση η οποία είναι έγκυρη όσον αφορά τον τύπο της κάθε μεταβλητής. Βεβαιωθείτε ότι θα μεταγλωττίσετε και θα τρέξετε το **EX03_9.PAS** και μελετήστε την έξοδό του μέχρι να την κατανοήσετε πλήρως. ***Ασκήσεις προγραμματισμού*** **1.** Γράψτε ένα απλό πρόγραμμα με μια εγγραφή για να αποθηκεύσετε τα ονόματα 5 φίλων σας και εμφανίστε τα ονόματα τους.

Παραδείγματα κεφαλαίου 9

```
1   (* Κεφάλαιο 9 - Πρόγραμμα 1 EX01_9.pas*)
2   program A_Small_Record;
3
4   type     Description = record
5                Year : integer;
6                Model : string[20];
7                Engine : string[8];
8              end;
9
10  var       Truck : Description;
11            Cars : array[1..10] of Description;
12            Index : integer;
13
14  begin (* main program *)
```

```
15
16     Truck.Year := 1988;
17     Truck.Model := 'Pickup';
18     Truck.Engine := 'Diesel';
19
20     for Index := 1 to 10 do begin
21        Cars[Index].Year := 1930 + Index;
22        Cars[Index].Model := 'Duesenburg';
23        Cars[Index].Engine := 'V8';
24     end;
25
26     Cars[2].Model := 'Stanley Steamer';
27     Cars[2].Engine := 'Coal';
28     Cars[7].Engine := 'V12';
29     Cars[9].Model := 'Ford';
30     Cars[9].Engine := 'rusted';
31
32     Write('My ',Truck.Year:4,' ');
33     Write(Truck.Model,' has a ');
34     Writeln(Truck.Engine,' engine.');
35
36     for Index := 1 to 10 do begin
37        Write('My ',Cars[Index].Year:4,' ');
38        Write(Cars[Index].Model,' has a ');
39        Writeln(Cars[Index].Engine,' engine.');
40     end;
41  end. (* of main program *)
42
43
44  { Result of execution
45
46  My 1988 Pickup has a Diesel engine.
47  My 1931 Duesenburg has a V8 engine.
48  My 1932 Stanley Steamer has a Coal engine.
49  My 1933 Duesenburg has a V8 engine.
50  My 1934 Duesenburg has a V8 engine.
51  My 1935 Duesenburg has a V8 engine.
52  My 1936 Duesenburg has a V8 engine.
```

```
My 1937 Duesenburg has a V12 engine.
My 1938 Duesenburg has a V8 engine.
My 1939 Ford has a rusted engine.
My 1940 Duesenburg has a V8 engine.

}
```

```
(* Κεφάλαιο 9 - Πρόγραμμα 2 EX02_9.pas*)
program A_Larger_Record;

const Number_Of_Friends = 50;

type    Full_Name = record
                First_Name : string[12];
                Initial : char;
                Last_Name : string[15];
          end;

          Date = record
                Day : byte;
                Month : byte;
                Year : integer;
          end;

          Person = record
                Name : Full_Name;
                City : string[15];
                State : string[2];
                Zipcode : string[5];
                Birthday : Date;
          end;

var     Friend : array[1..Number_Of_Friends] of Person;
          Self, Mother, Father : Person;
          Index : byte;

begin (* main program *)
    Self.Name.First_Name := 'Charley';
```

```
32      Self.Name.Initial := 'Z';
33      Self.Name.Last_Name := 'Brown';
34      with Self do begin
35          City := 'Anywhere';
36          State := 'CA';
37          Zipcode := '97342';
38          Birthday.Day := 17;
39         with Birthday do begin
40              Month := 7;
41              Year := 1938;
42          end;
43      end; (* all data for self now defined *)
44
45      Mother := Self;
46      Father := Mother;
47      for Index := 1 to Number_Of_Friends do
48          Friend[Index] := Mother;
49      Write(Friend[27].Name.First_Name,' ');
50      Write(Friend[33].Name.Initial,' ');
51      Write(Father.Name.Last_Name);
52      Writeln;
53  end. (* of main program *)
54
55  { Result of execution
56
57  Charley Z Brown
58
59  }
```

```
1   (* Κεφάλαιο 9 - Πρόγραμμα 3 EX03_9.pas *)
2   program Variant_Record_Example;
3
4   type    Kind_Of_Vehicle = (Car,Truck,Bicycle,Boat);
5
6            Vehicle = record
7                   Owner_Name : string[25];
8                   Gross_Weight : integer;
9                   Value : real;
```

```
10                case What_Kind : Kind_Of_Vehicle of
11                   Car : (Wheels : integer;
12                            Engine : string[8]);
13                   Truck : (Motor : string[8];
14                             Tires : integer;
15                   Payload : integer);
16                              Bicycle : (Tyres : integer);
17                   Boat : (Prop_Blades : byte;
18                             Sail : boolean;
19                   Power : string[8]);
20             end; (* of record *)
21
22 var    Sunfish,Ford,Schwinn,Mac : Vehicle;
23
24 begin (* main program *)
25    Ford.Owner_Name := 'Walter'; (* fields defined in order *)
26    Ford.Gross_Weight := 5750;
27    Ford.Value := 2595.00;
28    Ford.What_Kind := Truck;
29    Ford.Motor := 'V8';
30    Ford.Tires := 18;
31    Ford.Payload := 12000;
32
33    with Sunfish do begin
34       What_Kind := Boat; (* fields defined in random order *)
35       Sail := TRUE;
36       Prop_Blades := 3;
37       Power := 'wind';
38       Gross_Weight := 375;
39       Value := 1300.00;
40       Owner_Name := 'Herman and George';
41    end;
42
43    Ford.Engine := 'flathead'; (* tag-field not defined yet but it *)
44    Ford.What_Kind := Car; (* must be before it can be used *)
45    Ford.Wheels := 4;
46    (* notice that the non variant part is not redefined here *)
47
```

```
48     Mac := Sunfish; (* entire record copied, including the tag-field *)
49
50     if Ford.What_Kind = Car then (* this should print *)
51         Writeln(Ford.Owner_Name,' owns the car with a ',Ford.Engine,
52         ' engine.');
53
54     if Sunfish.What_Kind = Bicycle then (* this should not print *)
55         Writeln('The sunfish is a bicycle which it shouldn''t be');
56
57     if Mac.What_Kind = Boat then (* this should print *)
58         Writeln('The mac is now a boat with',Mac.Prop_Blades:2,
59         ' propeller blades.');
60 end. (* of main program *)
61
62 { Result of execution
63
64 Walter owns the car with the flathead engine.
65 The mac is now a boat with three propeller blades.
66
67 }
```

16/06/2011 11:08:00

16/06/2011 11:08:00

16/06/2011 11:08:00

16/06/2011 11:08:00

16/06/2011 11:08:00

16/06/2011 11:08:00