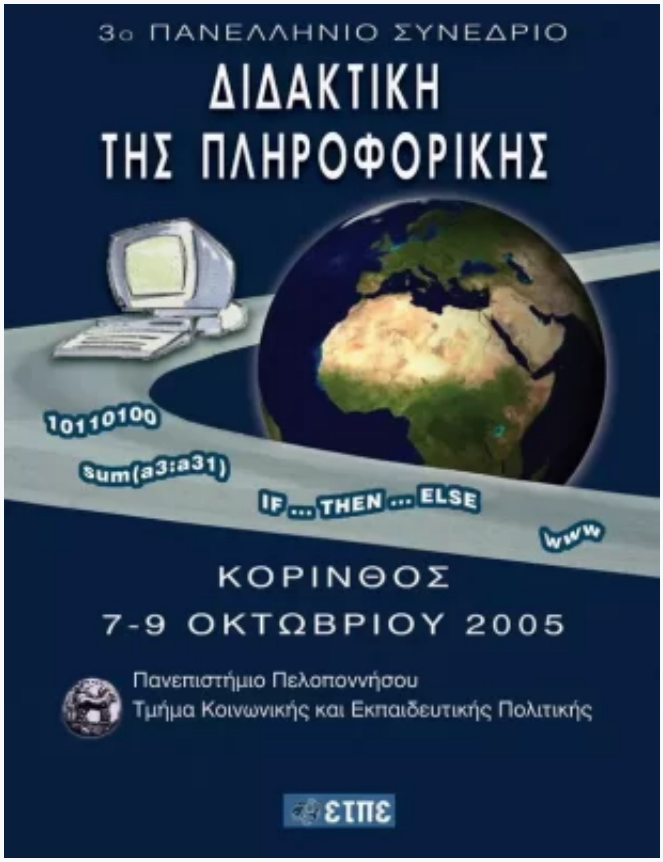
Συνέδρια της Ελληνικής Επιστημονικής Ένωσης Τεχνολογιών Πληροφορίας & Επικοινωνιών στην Εκπαίδευση

Τόμ. 1 (2005)

3ο Πανελλήνιο Συνέδριο «Διδακτική της Πληροφορικής»



Διδακτική του Προγραμματισμού

Αθανάσιος Μάργαρης, Ευθύμιος Κότσιαλος

Βιβλιογραφική αναφορά:

Μάργαρης Α., & Κότσιαλος Ε. (2026). Διδακτική του Προγραμματισμού. *Συνέδρια της Ελληνικής Επιστημονικής Ένωσης Τεχνολογιών Πληροφορίας & Επικοινωνιών στην Εκπαίδευση*, *1*, 519–528. ανακτήθηκε από https://eproceedings.epublishing.ekt.gr/index.php/cetpe/article/view/8747

Διδακτική του Προγραμματισμού

Αθανάσιος Μάργαρης, Ευθύμιος Κότσιαλος
Τμήμα Εφαρμοσμένης Πληροφορικής, Πανεπιστήμιο Μακεδονίας
amarg@uom.gr, ekots@uom.gr

ΠΕΡΙΛΗΨΗ
Στόχος αυτής της εργασίας αποτελεί η παρουσίαση των βασικών αρχών και τεχνικών που θα πρέπει να ακολουθούνται για την επιτυχή διδασκαλία μαθημάτων προγραμματισμού σε Τμήματα Πληροφορικής της Τριτοβάθμιας Εκπαίδευσης. Η παρουσίαση που ακολουθεί, θίγει αρκετά σημαντικά ζητήματα που συσχετίζονται με το θέμα αυτό, όπως είναι τα κριτήρια με τα οποία θα επιλέξουμε τη γλώσσα προγραμματισμού που θα διδάξουμε, ο τρόπος διδασκαλίας των δύο βασικών μεθόδων προγραμματισμού (που είναι ο διαδικαστικός και ο αντικειμενοστραφής προγραμματισμός), καθώς και η παρουσίαση εφαρμογών που βοηθούν τον φοιτητή να αναπτύξει τις προγραμματιστικές του ικανότητες.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: *Προγραμματισμός, Μεθοδολογία διδασκαλίας*

ΕΙΣΑΓΩΓΗ
Ένας από τους πιο σημαντικούς κλάδους της επιστήμης της Πληροφορικής, είναι αυτός του Προγραμματισμού, ο οποίος συνίσταται στη μεθοδολογία σχεδίασης και ανάπτυξης εφαρμογών σε κάποια από τις γλώσσες προγραμματισμού που σήμερα είναι διαθέσιμες. Αυτή η ικανότητα ανάπτυξης εφαρμογών θεωρείται πάρα πολύ σημαντική για κάποιον φοιτητή που ειδικεύεται στην επιστήμη της Πληροφορικής και για το λόγο αυτό τα τελευταία χρόνια έχουν αναπτυχθεί πολλές εκπαιδευτικές μέθοδοι όσον αφορά τη διδασκαλία του αντίστοιχου γνωστικού αντικειμένου. Ας σημειωθεί πως αυτές οι μέθοδοι δεν εφαρμόζονται πάντα με τον ίδιο τρόπο. Εάν για παράδειγμα οι φοιτητές προέρχονται από σχολείο στο οποίο διδάσκεται η τεχνική του προγραμματισμού είτε με τη βοήθεια κάποιας πραγματικής γλώσσας είτε με τη βοήθεια ειδικά σχεδιασμένων γλωσσών και τη χρήση εργαλείων όπως η Γλωσσομάθεια (Νικολαΐδης 2004), η κατάσταση είναι αρκετά πιο απλή, καθώς οι φοιτητές υποτίθεται πως γνωρίζουν ήδη τις βασικές αρχές ανάπτυξης εφαρμογών. Στην αντίθετη περίπτωση όμως, θα πρέπει να ληφθεί ιδιαίτερη μέριμνα για τη σωστή διδασκαλία του προγραμματισμού, καθώς εάν αυτός διδαχθεί και εφαρμοσθεί με εσφαλμένο τρόπο θα οδηγήσει στην ανάπτυξη εφαρμογών με ανορθόδοξες τεχνικές, οι οποίες γενικά είναι πολύ δύσκολο να διορθωθούν, ενώ προφανώς θα αποτελέσει ένα γνωστικό πεδίο ανεπιθύμητο και απωθητικό για τους φοιτητές. Στην περιγραφή που ακολουθεί υποθέτουμε πως οι φοιτητές δεν διαθέτουν προγενέστερη γνώση πάνω σε αρχές και μεθοδολογίες

Πρακτικά Εργασιών 3ου Πανελλήνιου Συνεδρίου «Διδακτική της Πληροφορικής»
Α. Τζιμογιάννης (επιμ.)
Πανεπιστήμιο Πελοποννήσου
Κόρινθος, 7-9 Οκτωβρίου 2005

προγραμματισμού και επομένως η διδασκαλία του ξεκινά τυπικά και ουσιαστικά από μηδενική βάση.

Λόγω της εξαιρετικής σημασίας του σωστού τρόπου διδασκαλίας του προγραμματισμού, η ανάπτυξη διδακτικών μεθοδολογιών και κυρίως εκπαιδευτικών εργαλείων αποτελεί τα τελευταία χρόνια αντικείμενο έντονης ερευνητικής δραστηριότητας. Συνοπτική περιγραφή ενδεικτικών ερευνητικών δραστηριοτήτων πάνω στο θέμα αυτό κυρίως όσον αφορά την ανάπτυξη εργαλείων μάθησης προγραμματισμού, παρουσιάζεται στην ενότητα του εκπαιδευτικού λογισμικού.

ΕΠΙΛΟΓΗ ΤΗΣ ΓΛΩΣΣΑΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Η πρώτη σημαντική απόφαση που θα πρέπει να λάβουμε, αφορά τη γλώσσα προγραμματισμού που θα επιλέξουμε να χρησιμοποιήσουμε. Η απόφαση αυτή είναι ιδιαίτερα σημαντική σε περιπτώσεις φοιτητών που δεν διαθέτουν την παραμικρή προγραμματιστική εμπειρία – η πρακτική έχει δείξει πως η γλώσσα με την οποία ο φοιτητής έρχεται σε επαφή για πρώτη φορά με τον προγραμματισμό ασκεί πάρα πολύ μεγάλη επίδραση στον τρόπο ανάπτυξης εφαρμογών, στις τεχνικές προγραμματισμού που χρησιμοποιεί καθώς και στην ποιότητα του κώδικα που αναπτύσσει. Χαρακτηριστικό παράδειγμα προς αποφυγή είναι η χρήση της γλώσσας Java ως την πρώτη γλώσσα προγραμματισμού. Σε μια τέτοια περίπτωση ο φοιτητής θα αναγκαστεί να έρθει σε επαφή με προχωρημένες έννοιες του αντικειμενοστραφούς προγραμματισμού όπως είναι οι κλάσεις και τα αντικείμενα και η χρήση δομητών, αποδομητών και μεθόδων και θα χάσει την ουσία που είναι η κατανόηση της δομής του προγράμματος και του τρόπου με τον οποίο χρησιμοποιούνται οι διάφορες δομικές μονάδες που περιλαμβάνονται σε αυτό (όπως είναι οι μεταβλητές, οι συναρτήσεις και οι αριθμητικοί και λογικοί τελεστές). Μιλώντας γενικά, η γλώσσα που θα χρησιμοποιηθεί για τη διδασκαλία της τεχνικής του προγραμματισμού σε άτομα που δεν διαθέτουν καμία εμπειρία επί του αντικειμένου, θα πρέπει να διαθέτει τα ακόλουθα χαρακτηριστικά (Gupta 2004).

Απλότητα: η γλώσσα θα πρέπει να είναι σχετικά απλή και να μην χαρακτηρίζεται από περίεργη σύνταξη και ροή προγράμματος που είναι δύσκολο να κατανοηθεί, όπως συμβαίνει για παράδειγμα στις συναρτησιακές γλώσσες προγραμματισμού (functional programming languages). Επιπλέον δεν θα πρέπει να χαρακτηρίζεται από συντακτικές συνωνυμίες, που επιτρέπουν την πραγματοποίηση της ίδιας διαδικασίας με πολλούς τρόπους διαφορετικούς μεταξύ τους – για παράδειγμα στη γλώσσα προγραμματισμού C η αναφορά μας στο υπ' αριθμόν m στοιχείο κάποιου πίνακα a μπορεί να γίνει γράφοντας a[m], *(a+m) ή *++a. Η ύπαρξη τέτοιων χαρακτηριστικών είναι βέβαιο πως θα δημιουργήσει σύγχυση στον άπειρο φοιτητή και εάν είναι διαθέσιμα δεν θα πρέπει σε καμιά περίπτωση να χρησιμοποιούνται τουλάχιστον κατά τα πρώτα στάδια ενασχόλησής του με τον προγραμματισμό.

Εύρος: η γλώσσα θα πρέπει να υποστηρίζει όλα τα βασικά χαρακτηριστικά των γλωσσών προγραμματισμού όπως είναι οι βρόγχοι (do, while, for), οι εντολές διακλάδωσης του προγράμματος υπό συνθήκη (if, then, else, switch, case) καθώς επίσης

οι πίνακες και οι σύνθετες δομές δεδομένων, έτσι ώστε ο φοιτητής να μπορέσει να γνωρίσει και να μάθει να χρησιμοποιεί όλες αυτές τις έννοιες. Επιπλέον θα πρέπει να χαρακτηρίζεται από συντακτική και σημασιολογική συνέπεια έτσι ώστε να αποφεύγεται η εμφάνιση προβληματικών καταστάσεων όπως είναι οι ατέρμονοι βρόγχοι. Θεωρώντας για παράδειγμα ένα κώδικα σε γλώσσα C της μορφής if (x=1) do_something(); είναι προφανές πως αυτός θα οδηγήσει σε ατέρμονο βρόγχο, καθώς η συνθήκη είναι πάντοτε αληθής, ανεξάρτητα από την τιμή της μεταβλητής x. Αυτό συμβαίνει γιατί η γλώσσα C μετατρέπει όλους σχεδόν τους τύπους δεδομένων σε ακέραιους τύπους χωρίς μάλιστα να ενημερώνει τον προγραμματιστή για αυτή την μετατροπή. Αντίθετα σε γλώσσες προγραμματισμού όπως είναι η Java που χαρακτηρίζονται από αυστηρότητα όσον αφορά τη διαχείριση των τύπων δεδομένων (strongly typed languages) η παραπάνω έκφραση θα επιστρέψει μία λογική τιμή (Boolean value) και ο ατέρμων βρόγχος δεν πρόκειται ποτέ να εμφανιστεί.

Χρόνος απόκρισης του μεταγλωττιστή: το πρόγραμμα μεταγλώττισης που θα χρησιμοποιηθεί από τους φοιτητές για τη δημιουργία του εκτελέσιμου αρχείου, θα πρέπει να είναι εύκολο στη χρήση του και κυρίως, να δημιουργεί το εκτελέσιμο αρχείο της εφαρμογής σε σχετικά μικρό χρονικό διάστημα. Όπως είναι γνωστό από την εμπειρία, ένας αρχάριος προγραμματιστής προχωρεί σε συχνές μεταγλωττίσεις του κώδικα πραγματοποιώντας μικρές αλλαγές και διορθώνοντας μόνο ένα – ή έστω λίγα – λάθη κάθε φορά. Αυτό σημαίνει πως θα πρέπει να χρησιμοποιήσουμε ένα μεταγλωττιστή που να είναι απλός και εύκολος στη χρήση του και να δημιουργεί σε πολύ μικρό χρονικό διάστημα το εκτελέσιμο αρχείο, παρά ένα πολύπλοκο ολοκληρωμένο περιβάλλον που απαιτεί τη γνώση επιπρόσθετων πληροφοριών άσχετων με τις βασικές αρχές του προγραμματισμού και καθιστά τη δημιουργία του εκτελέσιμου αρχείου μια πολύπλοκη διαδικασία που είναι σίγουρο πως θα τρομάξει τον απλό χρήστη. Για παράδειγμα, η δημιουργία μιας μικρής εφαρμογής που θα εκτυπώνει την – ιστορική πια – φράση Hello Word μπορεί να γίνει πάρα πολύ εύκολα χρησιμοποιώντας για παράδειγμα την απλή Turbo C, παρά τη Visual C++ όπου η δημιουργία έστω και μιας απλής τέτοιας εφαρμογής απαιτεί τη δημιουργία Workspace, τον καθορισμό του τρόπου σύνδεσης των βιβλιοθηκών (στατικός ή δυναμικός) και την πολύπλοκη – γενικά – διαμόρφωση των παραμέτρων του τρέχοντος project.

Σύστημα βοηθείας και κατανοητά μηνύματα σφάλματος: το περιβάλλον ανάπτυξης εφαρμογών θα πρέπει να παρέχει επαρκή βοήθεια προς το χρήστη – για παράδειγμα, εάν ο χρήστης επιλέξει κάποια εντολή ή δεσμευμένη λέξη και πατήσει το πλήκτρο F1 θα πρέπει να μπορεί να δει ένα ενημερωτικό πλαίσιο διαλόγου που θα περιέχει πληροφορίες σχετικά με το ρόλο και τον τρόπο χρήσης της λέξης που επέλεξε. Επιπλέον, τα μηνύματα σφάλματος που εμφανίζονται θα πρέπει να είναι κατανοητά και κατατοπιστικά έτσι ώστε να δίνουν τη δυνατότητα στο χρήστη να διορθώσει από μόνος του τα σφάλματα που πραγματοποίησε χωρίς να χρειαστεί να καταφύγει στη βοήθεια του διδάσκοντα.

Άλλα επιθυμητά χαρακτηριστικά που θα πρέπει να διαθέτει η επιλεγμένη γλώσσα προγραμματισμού είναι η σαφήνεια όσον αφορά τη χρήση των τύπων δεδομένων (για

παράδειγμα στη γλώσσα C οι πραγματικοί αριθμοί περιγράφονται από τους τύπους float και double γεγονός που ενδέχεται να δημιουργήσει σύγχυση σχετικά με τον τρόπο χρήσης του καθενός) ενώ τα μεγέθη των τύπων θα πρέπει να είναι τα ίδια ανεξάρτητα από το σύστημα που χρησιμοποιούνται – αυτό είναι κάτι που ισχύει στη Java αλλά όχι στη C όπου ο τύπος δεδομένων int ανάλογα από το σύστημα (platform) στο οποίο χρησιμοποιείται μπορεί να έχει μέγεθος από 2 έως και 32 bytes.

ΑΝΑΛΥΤΙΚΗ ΠΑΡΟΥΣΙΑΣΗ ΤΩΝ ΒΑΣΙΚΩΝ ΕΝΝΟΙΩΝ

Μετά την απόφαση σχετικά με τη γλώσσα που θα χρησιμοποιήσουμε για την διδασκαλία του μαθήματος θα πρέπει να εξοικειώσουμε τους φοιτητές με τις βασικές έννοιες του αντικειμένου όπως είναι οι μεταβλητές και οι συναρτήσεις του προγράμματος και οι εντολές διακλάδωσης και ελέγχου. Ο φοιτητής θα πρέπει να γνωρίσει τους διαφόρους τύπους σφαλμάτων που δύνανται να εμφανιστούν σε μία εφαρμογή (λεκτικά, συντακτικά, σημασιολογικά και χρόνου εκτέλεσης) και να μπορέσει να τα διακρίνει μεταξύ τους. Θεωρώντας για παράδειγμα τη γλώσσα προγραμματισμού C, ο φοιτητής θα πρέπει να είναι σε θέση να γνωρίζει πως: (α) η δήλωση double 3dim; περιέχει λεκτικό λάθος (lexical error) καθώς η γλώσσα C δεν επιτρέπει τη χρήση αριθμητικών χαρακτήρων ως τον πρώτο χαρακτήρα για τα ονόματα μεταβλητών, (β) η δήλωση double int; συνιστά συντακτικό λάθος (syntactic error) καθώς η λέξη int αποτελεί δεσμευμένη λέξη (keyword) της γλώσσας και δεν μπορεί να χρησιμοποιηθεί με διαφορετικό τρόπο, (γ) η καταχώρηση a=b όπου το a είναι ακέραιος και το b συμβολοσειρά συνιστά σημασιολογικό λάθος (semantic error), ενώ (δ) η εντολή a=1/b; ενδέχεται να οδηγήσει σε σφάλμα χρόνου εκτέλεσης (run-time error) ανάλογα με την τιμή της μεταβλητής b. Στη συνέχεια ο χρήστης θα πρέπει να ενημερωθεί για τον τρόπο δημιουργίας του εκτελέσιμου αρχείου – που συνίσταται από τα στάδια της μεταγλώττισης και της σύνδεσης των βιβλιοθηκών. Η παραπάνω διαδικασία αφορά τη χρήση συμβατικών και απλών προγραμματιστικών μεθόδων, ενώ για την περίπτωση του αντικειμενοστραφούς προγραμματισμού, θα πρέπει να παρουσιαστούν αναλυτικά – σε θεωρητικό επίπεδο – οι θεμελιώδεις έννοιες αυτής της προσέγγισης όπως είναι οι έννοιες του δομητή (constructor) και του αποδομητή (destructor), η χρήση προσωπικών (private), προστατευμένων (protected) και δημόσιων (public) πεδίων και μεθόδων, η υπερφόρτωση τελεστών (operator overloading) και οι έννοιες της κληρονομικότητας (inheritance), του πολυμορφισμού (polymorphism), της ενθυλάκωσης (encapsulation) και της απόκρυψης πληροφορίας (information hiding). Είναι προφανές πως η διδασκαλία του αντικειμενοστραφούς προγραμματισμού προϋποθέτει την εξοικείωση του φοιτητή με τις θεμελιώδεις αρχές του απλού διαδικαστικού (procedural) προγραμματισμού – στην αντίθετη περίπτωση η εμπειρία έχει δείξει πως προκύπτουν σοβαρά προβλήματα κατανόησης του αντικειμένου, καθώς ο φοιτητής έρχεται σε επαφή με προχωρημένες τεχνικές τη στιγμή που ακόμη δεν γνωρίζει τις θεμελιώδεις αρχές της ανάπτυξης εφαρμογών.

ΔΙΑΔΙΚΑΣΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Η διδασκαλία τεχνικών του συμβατικού διαδικαστικού προγραμματισμού προϋποθέτει την εξοικείωση του φοιτητή με τις θεμελιώδεις έννοιες που παρουσιάστηκαν προηγουμένως και θα πρέπει να οργανωθεί σε μια σειρά διαλέξεων που να καλύπτουν όλο το εύρος των χαρακτηριστικών της γλώσσας που διδάσκεται. Αυτές οι διαλέξεις μπορούν να ομαδοποιηθούν σε δύο (ή και περισσότερα) εξαμηνιαία μαθήματα (Malmi & Korhonen 2004). Στο πρώτο από αυτά τα μαθήματα η έμφαση θα δοθεί στην κατανόηση των χαρακτηριστικών της γλώσσας για τη χρήση της σε προβλήματα διαχείρισης μονοδιάστατων και δισδιάστατων (τουλάχιστον) πινάκων, την ανάγνωση και εγγραφή σε αρχείο (η παρουσίαση θα περιλαμβάνει τόσο αρχεία κειμένου όσο και δυαδικά αρχεία), τη δημιουργία και χρήση σύνθετων τύπων δεδομένων, καθώς και έννοιες που συσχετίζονται με τη διαχείριση μνήμης – αν και το τελευταίο θεωρείται πιο προχωρημένο και θα μπορούσε να διδαχθεί μόνο εφόσον το επιτρέπει το επίπεδο του τμήματος και ο βαθμός κατανόησης των εννοιών του προγραμματισμού. Στο επόμενο μάθημα μπορεί να λάβει χώρα η περιγραφή εννοιών που συσχετίζονται με αλγορίθμους (ταξινόμηση, αναζήτηση, η μέθοδος του hashing κ.λ.π.) και δομές δεδομένων (απλές και διπλές συνδεδεμένες λίστες, δέντρα και γραφήματα). Σημαντική επίσης θεωρείται η κατανόηση εννοιών που συσχετίζονται με την απόδοση (performance) των αλγορίθμων καθώς και ο τρόπος με τον οποίο θα επιλέγουμε κάποιον από αυτούς μέσα από ένα σύνολο αλγορίθμων που πραγματοποιούν την ίδια διαδικασία (για παράδειγμα ο αλγόριθμος του quick sort θα πρέπει να επιλεγεί έναντι του αλγορίθμου του bubble sort γιατί είναι πολύ πιο γρήγορος).

ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Το βασικό πρόβλημα που συσχετίζεται με τη διδασκαλία του αντικειμενοστραφούς προγραμματισμού είναι η απότομη μετάβαση του φοιτητή σε ένα καινούριο τρόπο σκέψης εντελώς διαφορετικό από εκείνο στον οποίο είχε εντρυφήσει στα πλαίσια της διδασκαλίας του συμβατικού διαδικαστικού προγραμματισμού – η έρευνα ωστόσο έχει δείξει πως οι μαθητές εκείνοι που γνωρίζουν περισσότερες από μία γλώσσες προγραμματισμού πραγματοποίησαν πιο εύκολα τη μετάβαση σε αυτό το νέο τρόπο αντίληψης, γεγονός που αποδόθηκε στη μεγαλύτερη εμπειρία και εξοικείωση που απέκτησαν.

Τα βασικά προβλήματα που εντοπίζονται κατά το στάδιο της διδασκαλίας του αντικειμενοστραφούς προγραμματισμού είναι σε γενικές γραμμές τα εξής (Ragonis & Ben-Ari 2005):

(α) δυσκολία της κατανόησης του τρόπου χρήσης των μεθόδων. Για παράδειγμα, παρατηρήθηκε περίπτωση κατά την οποία ο φοιτητής κάλεσε μια μέθοδο για την αλλαγή της τιμής κάποιας ιδιότητας του αντικειμένου και στη συνέχεια – μην έχοντας κατανοήσει πως η κλήση της μεθόδου προκάλεσε την μεταβολή της τιμής της μεταβλητής – την τροποποίησε με το χέρι. Ανάλογα προβλήματα παρατηρήθηκαν και σε περιπτώσεις κατά τις οποίες μία μέθοδος καλείται μέσα από κάποια άλλη.

(β) δυσκολία της κατανόησης της έννοιας της επιστρεφόμενης τιμής συνάρτησης, γεγονός που καταδεικνύει έλλειψη εξοικείωσης με την έννοια της ροής του προγράμματος και οδηγεί σε ερωτήσεις του τύπου «σε ποιον επιστρέφεται η τιμή;» ή «τι συμβαίνει με την επιστρεφόμενη τιμή;».
(γ) προβλήματα χρήσης των εννοιών του δομητή και του αποδομητή. Ο φοιτητής σε πολλές περιπτώσεις δεν αντιλαμβάνεται τη διαφορά που υφίσταται ανάμεσα στη δήλωση και στη χρήση του δομητή και του αποδομητή. Επιπλέον σύγχυση επιφέρει το γεγονός της δυνατότητας ύπαρξης πολλών δομητών με το ίδιο όνομα οι οποίοι δέχονται διαφορετικά ορίσματα (και επομένως αρχικοποιούν το αντικείμενο με διαφορετικό τρόπο), αλλά ενός και μοναδικού αποδομητή ο οποίος μάλιστα καλείται υποχρεωτικά χωρίς ορίσματα.
(δ) προβλήματα κατανόησης της έννοιας της επαναχρησιμοποίησης βιβλιοθηκών καθώς και του τρόπου με τον οποίο αυτή πραγματοποιείται.

Μιλώντας γενικά, η διδασκαλία του αντικειμενοστραφούς προγραμματισμού θα πρέπει να πραγματοποιηθεί σε τρία διαφορετικά στάδια, τα οποία σε γενικές γραμμές είναι τα εξής (Sheez et al. 1997):
(α) διδασκαλία των βασικών εννοιών του αντικειμενοστραφούς προγραμματισμού σε καθαρά θεωρητικό επίπεδο
(β) ανάλυση και μοντελοποίηση του προβλήματος (δημιουργία διαγραμμάτων κλάσεων (class diagrams), αντικειμένων (object diagrams) και ενεργειών (action diagrams)) (Booch 1994) χρησιμοποιώντας ειδικά εργαλεία, όπως είναι για παράδειγμα η γλώσσα UML (Unified Modeling Language) (Flower & Scott 1999) και
(γ) υλοποίηση αντικειμενοστραφών εφαρμογών στην κατάλληλη γλώσσα προγραμματισμού (π.χ. C++, Java ή Smalltalk). Πάντως η εμπειρία έχει δείξει πως οι φοιτητές θεωρούν τον αντικειμενοστραφή προγραμματισμό πιο δύσκολο σε σχέση με τον παραδοσιακό διαδικαστικό προγραμματισμό, γεγονός που θα πρέπει να ληφθεί σοβαρά υπ' όψιν κατά τη σχεδίαση των προγραμμάτων σπουδών και της διδακτέας ύλης του κάθε μαθήματος. Ενδεικτικός τρόπος οργάνωσης των διαλέξεων ενός μαθήματος αντικειμενοστραφούς προγραμματισμού μπορεί να βρεθεί στην αναφορά (Thramboulidis 2003).

ΕΚΠΑΙΔΕΥΤΙΚΟ ΛΟΓΙΣΜΙΚΟ

Η κατανόηση των θεμελιωδών προγραμματιστικών τεχνικών τα τελευταία χρόνια, υποβοηθείται από τη χρήση εκπαιδευτικών εφαρμογών ειδικά σχεδιασμένων για αυτό το σκοπό, οι οποίες επιτρέπουν στο φοιτητή να κατανοήσει σε βάθος τον αλγόριθμο που εφαρμόζεται έτσι ώστε στη συνέχεια να μπορέσει να τον υλοποιήσει πιο εύκολα. Χαρακτηριστικό παράδειγμα τέτοιων εφαρμογών, είναι η επίδειξη του τρόπου λειτουργίας κάποιου αλγορίθμου (π.χ. του αλγορίθμου ταξινόμησης quick sort ή τη μετακίνηση στους κόμβους κάποιου δέντρου) με τη βοήθεια ειδικών εφαρμογών που στηρίζονται στη χρήση video ή animation (Byrne et al. 1996). Σε μια πιο αναλυτική περιγραφή, οι εκπαιδευτικές εφαρμογές που βοηθούν τους φοιτητές να αναπτύξουν και

να εξασκήσουν τις προγραμματιστικές τους δεξιότητες, χωρίζονται στις ακόλουθες κατηγορίες (Deek & McHugh 1998):
α) Προγραμματιστικά περιβάλλοντα (programming environments) που επιτρέπουν στους φοιτητές να εξοικειωθούν με επιλεγμένα χαρακτηριστικά των γλωσσών προγραμματισμού ενώ παράλληλα επιτρέπουν τη μεταγλώττιση, σύνδεση, εκτέλεση και αποσφαλμάτωση του προγράμματος. Τυπικές εφαρμογές που ανήκουν σε αυτή την κατηγορία είναι οι εφαρμογές SUPPORT, STRUEDI, EBPS και SOMA που εκτελούνται σε κατάσταση κειμένου, καθώς και οι εφαρμογές PICT, PECAN, SCHEMACODE και ASA που χρησιμοποιούνται μέσα από γραφικό περιβάλλον.
β) Βοηθητικές εφαρμογές αποσφαλμάτωσης (debugging aids) που χρησιμοποιούνται από τους προγραμματιστές για να ελέγξουν τη λειτουργία της εφαρμογής, να παρατηρήσουν τον τρόπο με τον οποίο αυτή συμπεριφέρεται κάτω από διάφορες συνθήκες, καθώς επίσης να ανιχνεύσουν και να διορθώσουν σφάλματα Οι εφαρμογές αυτές παρέχουν πολλά και διαφορετικά εργαλεία παρακολούθησης κώδικα (code watchers, tracers, flags, breakpoints, visualization and animation utilities). Τυπικές εφαρμογές που ανήκουν σε αυτή την κατηγορία είναι οι LAURA, DA, GENIUS και VIPS.
γ) Συστήματα ευφυούς διδασκαλίας (intelligent tutoring systems) που παρέχουν στο φοιτητή πιο έξυπνες μεθόδους κατανόησης των βασικών αρχών του προγραμματισμού. Τα συστήματα αυτά αποτελούνται από τρεις συνιστώσες: τη βάση γνώσης του αντικειμένου (domain knowledge base) που περιέχει τρόπους αναπαράστασης και περιγραφής των λύσεων του προβλήματος, των σφαλμάτων που εμφανίζονται και των κανόνων που ακολουθούνται σε κάθε περίπτωση, το μοντέλο φοίτησης (student model) που επιτρέπει την αναπαράσταση της γνώσης του φοιτητή και της διαδικασίας μέσα από την οποία αυτή αποκτάται (knowledge acquisition process) και τον πράκτορα διδασκαλίας (tutoring agent) που χρησιμοποιείται για τη διανομή του οργανωμένου εκπαιδευτικού υλικού (instructional modules) στους φοιτητές. Τυπικές εφαρμογές που ανήκουν σε αυτή την κατηγορία είναι οι εφαρμογές BIP, LISP Tutor και AST.

Θα πρέπει να σημειωθεί ωστόσο πως η παραπάνω ταξινόμηση αφορά εφαρμογές παλαιότερης γενεάς, ενώ σήμερα οι παραπάνω δυνατότητες προσφέρονται πλέον από τα ολοκληρωμένα προγραμματιστικά περιβάλλοντα (όπως είναι π.χ. η Visual C++) που παρέχουν εξελιγμένα εργαλεία ανάπτυξης και αποσφαλμάτωσης κώδικα και η λειτουργία των οποίων στηρίζεται – στις πιο συνηθισμένες περιπτώσεις – στις αρχές του αντικειμενοστραφούς προγραμματισμού.

ΑΞΙΟΛΟΓΗΣΗ ΚΑΙ ΣΥΝΕΡΓΑΤΙΚΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑ

Ένας απλός και εύκολος τρόπος για την αξιολόγηση και τη μέτρηση της απόδοσης των φοιτητών είναι η συμπλήρωση ερωτηματολογίων και ερωτήσεων πολλαπλών επιλογών σε καθαρά θεωρητικό επίπεδο. Τυπικό παράδειγμα ερώτησης που εμφανίζεται σε ένα ερωτηματολόγιο είναι η παράθεση ενός τμήματος κώδικα – μεταβλητού βαθμού δυσκολίας όσον αφορά την κατανόηση – το οποίο ο φοιτητής θα πρέπει να μελετήσει και να καταγράψει στο φύλλο ερωτήσεων το αποτέλεσμα της εκτέλεσής του. Από την

άλλη πλευρά, οι ερωτήσεις πολλαπλών επιλογών μπορούν να περιέχουν για παράδειγμα πολλές γραμμές κώδικα διαφορετικές μεταξύ τους, τις οποίες ο φοιτητής θα πρέπει να μελετήσει και να εντοπίσει ποιες από αυτές οδηγούν σε σφάλμα χρόνου εκτέλεσης και κάτω από ποιες συνθήκες. Στα πρώτα στάδια των παραδόσεων όπου η έμφαση δίδεται κυρίως στην εξοικείωση με τη χρήση και το συντακτικό της γλώσσας, ο καθηγητής μπορεί να δώσει στους φοιτητές ένα κώδικα που να περιέχει λεκτικά και συντακτικά λάθη τα οποία ο φοιτητής θα πρέπει να εντοπίσει και να διορθώσει είτε από μόνος του είτε χρησιμοποιώντας το πρόγραμμα μεταγλώττισης, ενώ σε πιο προχωρημένα στάδια ο κώδικας που διανέμεται στους φοιτητές θα είναι συντακτικά σωστός αλλά θα περιέχει σημασιολογικά και λογικά λάθη τα οποία θα πρέπει να εντοπιστούν δια της εφαρμογής μεθόδων αποσφαλμάτωσης (χρήση της συνάρτησης print για την εκτύπωση των τιμών των μεταβλητών, βηματική εκτέλεση του προγράμματος (step over) από κάποιες συγκεκριμένες γραμμές κώδικα (breakpoints), κ.λ.π.).

Η συνεργατική δραστηριότητα ανάμεσα στους φοιτητές είναι ακόμη ένας παράγοντας που μπορεί να συμβάλλει στην καλύτερη κατανόηση του αντικειμένου. Οι φοιτητές μπορούν να ομαδοποιηθούν σε μικρές ομάδες και να αναλάβουν την υλοποίηση τμήματος κάποιου μεγαλύτερου έργου, ενώ εναλλακτικά μπορούν να αξιολογήσουν ο ένας την επίδοση του άλλου έτσι ώστε να κατανοήσουν τις αδυναμίες τους και να επιλύσουν τα όποια προβλήματα αντιμετωπίζουν μέσα από τις μεταξύ τους συζητήσεις. Αυτές οι ομάδες θα πρέπει σε τακτά διαστήματα να συναντώνται με τους εκπαιδευτικούς έτσι ώστε να αξιολογείται η πρόοδός τους και να τίθενται και οι επόμενοι στόχοι που θα πρέπει να προσεγγισθούν οδηγώντας τελικά σε μια επιτυχημένη διαδικασία μάθησης.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Η διδακτική του προγραμματισμού οριοθετεί τις αρχές εκείνες που υπαγορεύουν τον τρόπο διδασκαλίας της εν λόγω τεχνικής με τρόπο ώστε αυτή να μπορέσει να εμπεδωθεί και να χρησιμοποιηθεί από τους φοιτητές. Οι μέθοδοι που εφαρμόζονται εξαρτώνται από την εμπειρία και το γνωστικό υπόβαθρο των φοιτητών ενώ η επιλογή της κατάλληλης σε κάθε περίπτωση γλώσσας προγραμματισμού αποτελεί σημαντική παράμετρο που επηρεάζει το αποτέλεσμα της όλης διαδικασίας με θετικό ή αρνητικό τρόπο. Η μεθοδολογία που ακολουθείται διαφοροποιείται σε περιπτώσεις διδασκαλίας συμβατικών διαδικαστικών ή αντικειμενοστραφών τεχνικών, ενώ η διαδικασία ανάπτυξης εφαρμογών διευκολύνεται σημαντικά από τα μοντέρνα προγραμματιστικά περιβάλλοντα που παρέχουν προηγμένα εργαλεία ανάπτυξης και αποσφαλμάτωσης του πηγαίου κώδικα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Booch G. (1994), *Object-oriented analysis and design with applications*, The Benjamin/Cummings Publishing Company Inc. (2nd edition)

Byrne M. D. et al. (1996), *Do algorithm animations aid learning ?,* GVU Technical Report, GIT-GVU-96-18, Georgia Institute of Technology

Deek F. & McHugh J. (1998), A survey and critical analysis of tools for learning programming, *Computer Science Education*, 8(2), 130-178

Flower M. & Scott K. (1999), *UML distilled: a brief guide to the standard object modeling language*, Addison-Wesley Professional (2nd edition)

Gupta D. (2004), *What is a good first programming language?* http://resolute.ucsd.edu/diwaker/articles/good-first-pl.html

Malmi L. & Korhonen A. (2004), *A pedagogical approach for teaching data structures and algorithms*, Department of Computer Science and Engineering, Helsinki University of Technology, Finald, http://www.spop.dk/chapters-abstractv1/lm_ak.pdf

Ragonis N. & Ben-Ari M. (2005), *On understanding the statics and dynamics of object-oriented programs*, *ACM SIGCSE*, 226-230

Sheetz S. D. et al. (1997), Exploring the difficulties of object-oriented techniques, *Journal of Management Information Systems*, 14(2), 103-132

Thramboulidis C. (2003), A sequence of assignments to teach object-oriented programming: a constructivism design-first approach, *Informatics in Education*, 2(1), 103-122

Νικολαίδης Σ. (2004), *Γλωσσομάθεια*, http://glossomatheia.studies.gr