ΚΕΦΑΛΑΙΟ 13ο

ΕΚΣΦΑΛΜΑΤΩΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

ΠΕΡΙΕΧΟΜΕΝΑ

- Κατηγορίες λαθών
- Εκσφαλμάτωση
- Εργαλεία εκσφαλμάτωσης
- Χειρισμός λαθών κατά το χρόνο εκτέλεσης

Εισαγωγή

Ανεξάρτητα από το πόσο προσεκτικά έχει γράψει κάποιος ένα πρόγραμμα, τις περισσότερες φορές απρόβλεπτες καταστάσεις, κακός χειρισμός ή λογικά λάθη στο σχεδιασμό των προγραμμάτων, οδηγούν στην εμφάνιση λαθών. Τα λάθη που παρουσιάζονται κατά την εκτέλεση ενός προγράμματος, μερικές φορές μπορεί να είναι ασήμαντα, όπως για παράδειγμα ή εσφαλμένη στοίχιση των αποτελεσμάτων. Άλλες φορές όμως μπορεί να είναι ιδιαίτερα κρίσιμα, ώστε να οδηγούν στην κατάρρευση των εφαρμογών ή του συστήματος. Ένα πρόγραμμα πριν παραδοθεί για πραγματική λειτουργία, πρέπει να ελέγχεται και να είναι βέβαιο ότι εργάζεται απρόσκοπτα και παράγει σωστά αποτελέσματα. Κάθε προγραμματιστής οφείλει κατά τη φάση της υλοποίησης, να δοκιμάζει λεπτομερώς το πρόγραμμα σε διαφορετικές συνθήκες και με ποικιλία δεδομένων, να διορθώνει όποια λάθη παρουσιαστούν και να συγκρίνει τα παραγόμενα αποτελέσματα με τα αναμενόμενα, ώστε να προλαμβάνει απρόσμενες καταστάσεις λάθους, πριν εμφανιστούν σε πραγματική λειτουργία.

Διδακτικοί στόχοι

Στόχοι του κεφαλαίου αυτού είναι οι μαθητές:

- να αναγνωρίζουν ένα λάθος.
- να ορίζουν τις κατηγορίες λάθους.
- να αναφέρουν εργαλεία εκσφαλμάτωσης και τον τρόπο χειρισμού τους.
- να εφαρμόζουν τις τεχνικές χειρισμού λαθών κατά το χρόνο εκτέλεσης.

Προερωτήσεις

- ✓ ποια είναι τα πιθανά λάθη που μπορεί να εμφανιστούν κατά την υλοποίηση και την εκτέλεση ενός προγράμματος;
- ✓ ποια είναι τα πλεονεκτήματα της εκσφαλμάτωσης;
- ✓ υπάρχουν εργαλεία που βοηθούν τον προγραμματιστή για την εκσφαλμάτωση ενός προγράμματος;
- ✓ είναι εφικτό να χειριστούμε αναπάντεχες καταστάσεις που πιθανόν θα εμφανιστούν κατά το χρόνο εκτέλεσης του προγράμματος, έτσι ώστε να μην διακόπτεται η λειτουργία του;
- ✓ η εργασία της εκσφαλμάτωσης θεωρείται ότι είναι απαραίτητη για την υλοποίηση ενός προγράμματος;

13.1 Κατηγορίες λαθών

Ένας προγραμματιστής, ανεξάρτητα από πόσο ικανός είναι, όταν δημιουργεί ένα πρόγραμμα, είναι φυσικό να κάνει κάποιο λάθος. Ειδικά σε μεγάλες εφαρμογές που κατασκευάζει πολύπλοκες υπολογιστικές ρουτίνες ή χρησιμοποιεί αρκετές συσκευές υλικού, είναι αδύνατο να αποφύγει τέτοιες ανεπιθύμητες καταστάσεις.

Σε ένα πρόγραμμα είναι δυνατό να παρουσιαστούν διαφορετικής μορφής λάθη, τα οποία μπορούν να χωριστούν σε τρεις βασικές κατηγορίες:

Λάθη κατά την υλοποίηση

Τα λάθη κατά το χρόνο υλοποίησης, προκαλούνται κυρίως από λανθασμένη σύνταξη εντολών προγράμματος. Τέτοια λάθη μπορεί να είναι η λανθασμένη συγγραφή μιας δεσμευμένης λέξης της γλώσσας προγραμματισμού ή η χρήση μιας δομής ελέγχου χωρίς την εντολή τερματισμού της.

Ένα λάθος που προκαλείται κατά τη συγγραφή του προγράμματος, ανιχνεύεται από το μεταγλωττιστή, ο οποίος εμφανίζει προς το προγραμματιστή κάποιο προειδοποιητικό μήνυμα. Αν το πρόγραμμα περιέχει ένα λάθος αυτής της μορφής, δεν επιτρέπεται η εκτέλεσή του, μέχρι να το διορθώσει ο προγραμματιστής.

Τα σύγχρονα προγραμματιστικά περιβάλλοντα μας προφυλάσσουν αυτόματα από τα λάθη κατά την υλοποίηση , αφού παρέχουν εργαλεία αυτόματου ελέγχου σύνταξης των εντολών και παρακολουθούν τον προγραμματιστή κατά τη συγγραφή του προγράμματος. Μόλις διαπιστώσουν κάποιο συντακτικό λάθος, σταματούν και απαιτούν τη διόρθωσή του. Συνήθως αντιλαμβάνονται ακριβώς το λάθος που δημιουργήθηκε και προτείνουν αναλυτικά τον τρόπο διόρθωσής του, εμφανίζοντας σε ενημερωτικό πλαίσιο την ορθή σύνταξη της εντολής που προκλήθηκε το λάθος.

Μια από τις πρώτες βλάβες που εμφανίστηκαν στους υπολογιστές ήταν ένα βραχυκύκλωμα που οφειλόνταν στον εγκλωβισμό ενός εντόμου (bug) στο εσωτερικό ενός υπολογιστή. Από το γεγονός αυτό, είναι πολύ πιθανό να προέρχεται ο όρος ***bugs****, με τον οποίο αναφερόμαστε στα λάθη που εμφανίζονται σε ένα πρόγραμμα.*

Λάθη κατά την εκτέλεση

Τα λάθη που προκαλούνται κατά το χρόνο εκτέλεσης του προγράμματος, είναι πιο επώδυνα γιατί συνήθως εμφανίζονται σε πραγματικό περιβάλλον εκτέλεσης και τις περισσότερες φορές προκαλούν τον αντικανονικό τερματισμό της εφαρμογής και το *κρέμασμα* (crash) του συστήματος.

Όταν ένα λάθος προκληθεί κατά την εκτέλεση της εφαρμογής, είναι δυνατό να αντιμετωπισθεί μόνο με τη χρήση εντολών προγράμματος που το παγιδεύουν και εκτελούν τις κατάλληλες διαδικασίες χειρισμού του.

Η πρόληψη τέτοιων λαθών είναι αρκετά δύσκολη, αφού συνήθως οφείλονται σε καταστάσεις που δεν είναι εύκολο να ελεγχθούν από τον προγραμματιστή, ενώ πολλές φορές εμφανίζονται μετά από μεγάλο χρονικό διάστημα. Τέτοια λάθη είναι δυνατό να προκληθούν από την κλήση μιας διαδικασίας με δεδομένα που δεν μπορεί να χειριστεί, όπως η αναζήτηση διαγραμμένων αρχείων, η προσπάθεια διαίρεσης ενός αριθμού με το μηδέν, η υπερχείλιση μιας αριθμητικής μεταβλητής ή από δυσλειτουργία του υλικού μέρους του υπολογιστή, όπως η καταστροφή του σκληρού δίσκου του συστήματος, ο τερματισμός μιας σύνδεσης δικτύου και η αποσύνδεση του εκτυπωτή.

Λογικά λάθη

Τα λογικά λάθη είναι συνήθως λάθη σχεδιασμού και δεν προκαλούν τη διακοπή της εκτέλεσης του προγράμματος. Ενώ ο μεταγλωττιστής της γλώσσας προγραμματισμού δεν ανιχνεύει κανένα συντακτικό λάθος και κατά την εκτέλεση του προγράμματος δεν παρουσιάζονται ανεπιθύμητες καταστάσεις σφαλμάτων, τελικά δεν παράγονται τα επιθυμητά αποτελέσματα.

Η ανίχνευση τέτοιων λαθών δεν είναι δυνατό να πραγματοποιηθεί από κάποιο εργαλείο του υπολογιστή και διαπιστώνονται μόνο με τη *διαδικασία ελέγχου* (testing) και την ανάλυση των αποτελεσμάτων των προγραμμάτων.

Το πιο δημοφιλές λάθος που παρουσιάστηκε στην ιστορία των υπολογιστών είναι το **πρόβλημα του έτους 2000** (millennium bug). Το πρόβλημα αυτό είναι ιδιόμορφο, γιατί οφείλεται σε συνδυασμένη προβληματική λειτουργία του λογισμικού και του υλικού μέρους του υπολογιστή και ακόμη απασχόλησε χρονικά την κοινωνία μας αρκετά πριν από την ουσιαστική εμφάνιση των συνεπειών του.

Άλλα σημαντικά προβλήματα που παρουσιάστηκαν στην ιστορία των υπολογιστών και αποδόθηκαν στη δυσλειτουργία του λογισμικού είναι :

- Το 1962 το διαστημικό όχημα Mariner 1 εκτοξεύτηκε από το ακρωτήρι Canaveral των Ηνωμένων Πολιτειών της Αμερικής για τον πλανήτη Αφροδίτη. Μετά την απογείωση ο πύραυλος που κατεύθυνε το διαστημικό όχημα έχασε την κατεύθυνσή του και οι τεχνικοί της διαστημικής υπηρεσίας των Η.Π.Α. (NASA) αναγκάστηκαν να τον ανατινάξουν, πριν αυτός καταστραφεί σε κάποιο σημείο της γης και κινδυ-

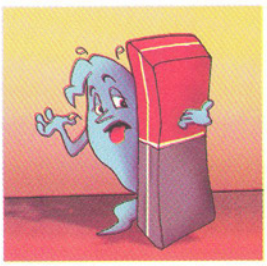
νεύσουν ανθρώπινες ζωές. Ευτυχώς το διαστημικό όχημα δεν είχε ανθρώπινο πλήρωμα. Η αναφορά της NASA απέδωσε το ατύχημα στην εσφαλμένη αντικατάσταση ενός θετικού πρόσημου με αρνητικό σε μια εντολή FORTRAN του λογισμικού πλοήγησης του πυραύλου. Το εσφαλμένο πρόσημο στοίχισε περίπου 80 εκατομμύρια δολάρια.

- Το 1990 ένα λάθος σε μια εντολή του λογισμικού των νέων συστημάτων δρομολόγησης τηλεφωνικών κλήσεων της εταιρείας τηλεπικοινωνιών AT&T, προκάλεσε το μπλοκάρισμα του μεγαλύτερου τμήματος του τηλεφωνικού δικτύου των Ηνωμένων Πολιτειών της Αμερικής. Περίπου 5 εκατομμύρια τηλεφωνικές γραμμές διακόπηκαν για εννέα ώρες.

- Ο επεξεργαστής Pentium το έτος 1994 παρουσίασε δυσλειτουργία, αφού σε σπάνιες περιπτώσεις έδινε εσφαλμένες απαντήσεις σε πολύπλοκες μαθηματικές εξισώσεις. Το πρόβλημα ανακαλύφτηκε από τον καθηγητή Thomas Nicely στο Πανεπιστήμιο Lynchburg στη Virginia των Ηνωμένων Πολιτειών της Αμερικής. Αρχικά η κατασκευάστρια εταιρεία αρνήθηκε την ύπαρξή του, αλλά στη συνέχεια υποχρεώθηκε να αντικαταστήσει τους προβληματικούς επεξεργαστές και σύμφωνα με εκτιμήσεις δαπάνησε γι αυτό το σκοπό περίπου 450 εκατομμύρια δολάρια

- Το έτος 1995 επρόκειτο να γίνουν τα εγκαίνια του νέου διεθνούς αεροδρομίου στο Denver των Ηνωμένων Πολιτειών της Αμερικής. Το αεροδρόμιο είχε κατασκευαστεί με σύγχρονη τεχνολογία και διέθετε ένα αυτόματο σύστημα μεταφοράς αποσκευών. Με την έναρξη λειτουργίας του το σύστημα μεταφοράς αποσκευών παρουσίασε λειτουργικά προβλήματα, με συνέπεια την καταστροφή αποσκευών επιβατών και του ιδίου του συστήματος. Το αεροδρόμιο διέκοψε τη λειτουργία του και επαναλειτούργησε δεκαέξι μήνες αργότερα και με κλασικό σύστημα μεταφοράς αποσκευών, επιβαρύνοντας τον προϋπολογισμό κατασκευής του κατά 3,2 εκατομμύρια δολάρια.

13.2 Εκσφαλμάτωση



Η διαδικασία ελέγχου, εντοπισμού και διόρθωσης των σφαλμάτων ενός προγράμματος καλείται *εκσφαλμάτωση* (debugging). Στόχος της διαδικασίας εκσφαλμάτωσης είναι ο εντοπισμός των σημείων του προγράμματος που προκαλούν προβλήματα στη λειτουργία του.

Η εργασία της εκσφαλμάτωσης δεν είναι εύκολη, απαιτεί βαθιά γνώση της γλώσσας προγραμματισμού και φυσικά αντίστοιχες ικανότητες από τον προγραμματιστή. Για τον εντοπισμό ενός λάθους δεν υπάρχουν ιδιαίτερα μυστικά και τρυκ. Η εκσφαλμάτωση είναι ένα πρόβλημα λογικής και όσο πιο καλά αντιλαμβάνεται ο προγραμματιστής τον τρόπο που εργάζεται το πρόγραμμα, τόσο πιο εύκολα και σύντομα θα εντοπίσει λάθη που προκαλούν δυσλειτουργίες.

Η εισαγωγή γραμμών με σχόλια σε ένα πρόγραμμα υποβοηθά σημαντικά την εκσφαλμάτωση.

Σε ένα σύγχρονο προγραμματιστικό περιβάλλον δεν χρειάζεται ιδιαίτερα μνεία για τα λάθη που παρουσιάζονται κατά το χρόνο σχεδιασμού, αφού αυτά, όπως αναφέρθηκε, είναι συντακτικά λάθη και τις περισσότερες φορές το περιβάλλον προγραμματισμού τα ανιχνεύει αυτόματα και προτείνει τη διόρθωσή τους. Ακόμη και αν το περιβάλλον δεν προτείνει τη διόρθωση, ο μεταγλωττιστής συλλαμβάνει και περιγράφει το λάθος και στη συνέχεια ο προγραμματιστής μπορεί πολύ εύκολα να το διορθώσει.

Τα λάθη που κυρίως μας απασχολούν στη φάση της εκσφαλμάτωσης είναι τα λογικά λάθη και τα λάθη που παρουσιάζονται κατά το χρόνο εκτέλεσης του προγράμματος. Η εκσφαλμάτωση τέτοιων λαθών μπορεί να γίνει μέσα από εργαλεία εκσφαλμάτωσης ή από ειδικές εντολές ή συναρτήσεις που προσφέρει το περιβάλλον προγραμματισμού.

13.3. Εργαλεία εκσφαλμάτωσης

Τα ονόματα των μεταβλητών πρέπει να ανάγουν στο περιεχόμενό τους. Έτσι διευκολύνεται η εκσφαλμάτωση.

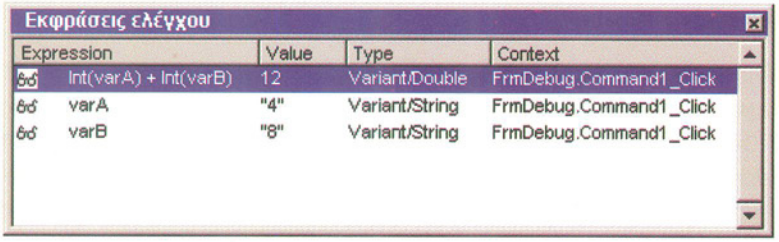
Πολλές φορές ο εντοπισμός ενός σφάλματος είναι ιδιαίτερα δύσκολος και για να επιτευχθεί χρειάζεται συστηματική παρακολούθηση και ανάλυση των δεδομένων του προγράμματος. Τα σύγχρονα εργαλεία προγραμματισμού δίνουν τη δυνατότητα στον προγραμματιστή να παρακολουθεί, τι συμβαίνει στο παρασκήνιο του προγράμματος κατά το χρόνο εκτέλεσης. Αυτό επιτυγχάνεται με τη χρήση κατάλληλων εργαλείων, που παρέχουν πολλές δυνατότητες ελέγχου κατά τη δοκιμαστική εκτέλεση ενός προγράμματος. Τα περισσότερα εργαλεία προγραμματισμού συνοδεύονται από *προγράμματα διόρθωσης* (debuggers).

Με τα εργαλεία εκσφαλμάτωσης είναι δυνατό να εμφανίζεται η τιμή μιας μεταβλητής ή μιας έκφρασης, να γίνεται δυναμική επέμβαση σε κάποιο σημείο δίνοντας διαφορετική τιμή σε μια μεταβλητή ή να εκτελείται βήμα προς βήμα το πρόγραμμα για τον εντοπισμό της εντολής που προκαλεί το σφάλμα.

Στη συνέχεια θα αναφέρουμε τις βασικές λειτουργίες εκσφαλμάτωσης που προσφέρουν τα περισσότερα προγραμματιστικά περιβάλλοντα :

Εκφράσεις ελέγχου

Οι *εκφράσεις ελέγχου* (watch expressions) είναι χρήσιμες για την άμεση παρατήρηση τιμών μεταβλητών ή εκφράσεων κατά την εκτέλεση ενός προγράμματος. Ακόμη με μια έκφραση ελέγχου, υπάρχει η δυνατότητα να διακοπεί η εκτέλεση ενός προγράμματος, όταν αλλάξει το περιεχόμενο μιας μεταβλητής ή δεχτεί συγκεκριμένη τιμή. Με τις εκφράσεις ελέγχου παρατηρούμε τη συμπεριφορά μιας μεταβλητής ή μιας έκφρασης σε όλη τη διάρκεια εκτέλεσης του προγράμματος.



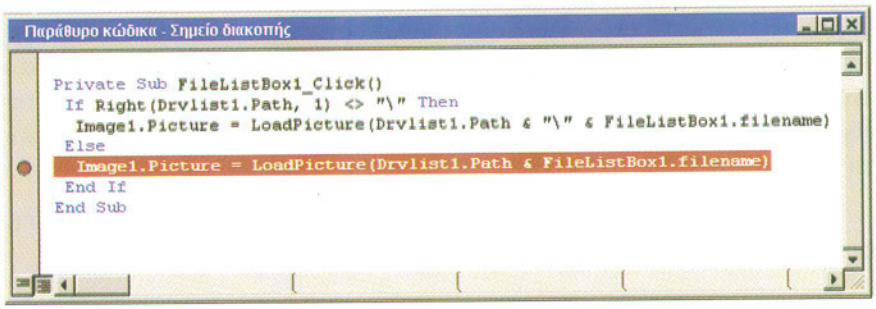
Σχ. 13.1. Εμφάνιση εκφράσεων ελέγχου

Σημεία διακοπής

Ένα *σημείο διακοπής* (breakpoint) διακόπτει την εκτέλεση ενός προγράμματος ακριβώς πριν από μια συγκεκριμένη εντολή. Τα σημεία διακοπής χρησιμοποιούνται για τον έλεγχο της κατάστασης των δεδομένων σε συγκεκριμένο σημείο του προγράμματος. Ορίζονται στον πηγαίο κώδικα με το μαρκάρισμα κάθε εντολής, στην οποία θέλουμε να διακόψουμε την εκτέλεση του προγράμματος.

Βήμα προς βήμα εκτέλεση

Μετά τη διακοπή εκτέλεσης από ένα σημείο διακοπής, παρέχεται η δυνατότητα να συνεχιστεί βήμα προς βήμα η εκτέλεση του προγράμματος. Με τη λειτουργία αυτή μπορούμε να εξετάσουμε βηματικά τα αποτελέσματα κάθε εντολής ή ρουτίνας του προγράμματος.



Σχ. 13.2. Σημείο διακοπής στον κώδικα της εφαρμογής

Ιστορικό

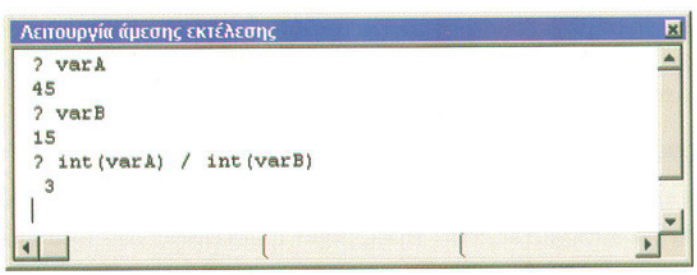
Με αυτή τη λειτουργία κατακρατείται *ιστορικό* (history) των τελευταίων εντολών που εκτελέστηκαν. Διαβάζοντας το ιστορικό αυτό μπορούμε να ελέγξουμε τη ροή του προγράμματος βαδίζοντας μπρος ή πίσω μέσα στον κώδικα.

Ιχνηλάτηση

Δίνει τη δυνατότητα αργής εκτέλεσης του προγράμματος, ενώ παράλληλα εμφανίζεται φωτισμένη η εντολή που εκτελείται κάθε φορά. Η *ιχνηλάτηση* (tracing) ενός προγράμματος ξεκινά από ένα σημείο διακοπής, που έχει εισαχθεί στο πρόγραμμα και λειτουργεί σε επίπεδο εντολής εκτελώντας κάθε φορά μια εντολή ή σε επίπεδο διαδικασίας εκτελώντας κάθε φορά όλη τη διαδικασία.

Λειτουργία άμεσης εκτέλεσης

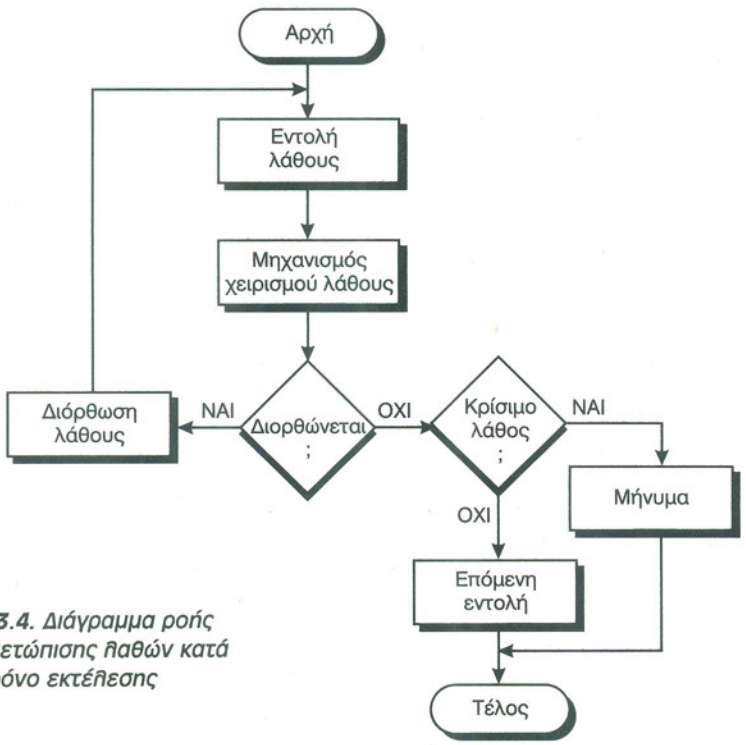
Με τη λειτουργία *άμεσης εκτέλεσης* έχουμε τη δυνατότητα σε κάποιο σημείο διακοπής να πληκτρολογήσουμε και να εκτελέσουμε άμεσα οποιαδήποτε εντολή με σκοπό τη λογική αλλαγή της ροής εκτέλεσης του προγράμματος.



Σχ. 13.3. Εμφάνιση των περιεχομένων των εκφράσεων σε λειτουργία άμεσης εκτέλεσης

Οι λειτουργίες των εργαλείων εκσφαλμάτωσης σε γραφικά περιβάλλοντα πραγματοποιούνται μέσα από ειδικά παράθυρα (π.χ. παράθυρο ελέγχου, παράθυρο άμεσης εκτέλεσης), ενώ στα υπόλοιπα εργαλεία από επιλογές μενού επιλογών που προσφέρει το περιβάλλον ανάπτυξης ή με ειδικές εντολές στον πηγαίο κώδικα.

Ένας προγραμματιστής που δεν είναι εξοικειωμένος και δεν γνωρίζει το χειρισμό των εργαλείων εκσφαλμάτωσης ή αν εργάζεται σε ένα περιβάλλον που δεν του παρέχει αντίστοιχα εργαλεία, μπορεί να υλοποιήσει αρκετές από τις λειτουργίες εκσφαλμάτωσης που αναφέρθηκαν, με την παρεμβολή εντολών εμφάνισης ή εκτύπωσης μηνυμάτων, τιμών μεταβλητών ή εκφράσεων.



Σχ. 13.4. Διάγραμμα ροής αντιμετώπισης λαθών κατά το χρόνο εκτέλεσης

13.4 Χειρισμός λαθών κατά το χρόνο εκτέλεσης

Τα λάθη που προκαλούνται κατά την εκτέλεση ενός προγράμματος μπορούν να προκληθούν από πάρα πολλές αιτίες. Ο προγραμματιστής είναι υποχρεωμένος μέσα από το πρόγραμμα του να προβλέψει κάθε πιθανό λάθος που μπορεί να συμβεί στο περιβάλλον εκτέλεσης, ώστε να το αντιμετωπίσει και να αποφύγει δυσάρεστα αποτελέσματα για το χρήστη.

Κάθε λάθος παράγει έναν κωδικό λάθους. Συνήθως ο προγραμματιστής δημιουργεί γενικές διαδικασίες χειρισμού λαθών, που τις εντάσσει σε βιβλιοθήκες.

Η διαδικασία χειρισμού λαθών κατά το χρόνο εκτέλεσης είναι απλή. Η εκτέλεση κάθε εντολής προγράμματος επιστρέφει μια ένδειξη που περιγράφει, αν ολοκληρώθηκε επιτυχώς ή παρουσιάστηκε κάποιο λάθος. Όταν παρουσιαστεί ένα λάθος, ο προγραμματιστής πρέπει να εκμεταλλευτεί αυτή την ένδειξη και μέσα από εντολές κώδικα να δώσει δυνατότητες διαφυγής στο πρόγραμμα.

Πιο αναλυτικά ο χειρισμός ενός λάθους κατά το χρόνο εκτέλεσης περιγράφεται ως εξής:

Στην ενότητα που θέλουμε να προσθέσουμε δυνατότητες χειρισμού λαθών, χρησιμοποιούμε τον αντίστοιχο μηχανισμό ανίχνευσης λαθών που προσφέρει το περιβάλλον προγραμματισμού.

Όταν προκληθεί ένα λάθος, ο μηχανισμός ανίχνευσης μεταφέρει τη ροή εκτέλεσης του προγράμματος σε ένα τμήμα κώδικα χειρισμού λάθους (exception ή error handler), στο οποίο περιγράφει το λάθος τις περισσότερες φορές με κάποιο αριθμό. Το συγκεκριμένο τμήμα κώδικα συνήθως βρίσκεται μέσα στην ίδια ενότητα που προκαλείται το λάθος.

Για την ανίχνευση και το χειρισμό των λαθών που εμφανίζονται κατά την εκτέλεση ενός προγράμματος στις γλώσσες προγραμματισμού Java, ADA και C++ χρησιμοποιείται ο μηχανισμός των εξαιρέσεων (exceptions), ενώ στη Visual Basic υπάρχει ειδική εντολή (On Error GoTo).

Η ρουτίνα χειρισμού του λάθους είναι διαφορετική κάθε φορά και δημιουργείται από τον προγραμματιστή ειδικά για το χειρισμό των λαθών που μπορεί να προκύψουν από την εργασία που εκτελείται εκείνη τη χρονική στιγμή, με στόχο φυσικά την ανώδυνη απεμπλοκή του προγράμματος από την ενδεχόμενη προβληματική κατάσταση.

Στη συνέχεια περιγράφουμε ορισμένες ενέργειες που θα μπορούσαμε να συμπεριλάβουμε στο τμήμα κώδικα που θα γίνει ο χειρισμός του λάθους:

- να διορθώσουμε μέσα από εντολές κώδικα το λάθος ή να δώσουμε τη δυνατότητα στο χρήστη να διορθώσει την αιτία πρόκλησης του λάθους. Στη συνέχεια να επιστρέψουμε τη ροή εκτέλεσης στην εντολή που προκάλεσε αυτή την κατάσταση λάθους. Μια τέτοια κατάσταση μπορεί να προκληθεί από την προσπάθεια του χρήστη να εκτυπώσει κάποιο αρχείο, ενώ ο εκτυπωτής είναι κλειστός. Μόλις ανιχνευτεί το λάθος, ενημερώνουμε το χρήστη ότι ο εκτυπωτής είναι κλειστός και αφού βεβαιωθούμε ότι διορθώθηκε το λάθος, επιστρέφουμε τη ροή εκτέλεσης του προγράμματος στην εντολή εκτύπωσης, η οποία ολοκληρώνεται με επιτυχία.
- να αξιολογήσουμε την εντολή που προκάλεσε το σφάλμα και αν θεωρήσουμε ότι η εκτέλεσή της δεν είναι κρίσιμη για την εφαρμογή, να μεταφέρουμε τη ροή εκτέλεσης στην επόμενη εντολή. Η εκτέλεση ενός αρχείου ήχου ταυτόχρονα με την εκτέλεση μιας εργασίας (π.χ. αντιγραφή αρχείων) δεν είναι σημαντική, εφόσον η συγκεκριμένη εργασία ολοκληρώνεται με επιτυχία.

- να θεωρήσουμε ιδιαίτερα κρίσιμο το σφάλμα και αφού ενημερώσουμε το χρήστη, να τερματίσουμε το πρόγραμμα. Αυτός ο τρόπος φυσικά πρέπει να αποφεύγεται, γιατί οδηγεί την εκτέλεση του προγράμματος σε απότομη διακοπή με απρόβλεπτες συνέπειες.

Όλες οι παραπάνω εργασίες πραγματοποιούνται μέσα από κατάλληλες εντολές που προσφέρουν σχεδόν όλα τα σύγχρονα περιβάλλοντα προγραμματισμού. Υπάρχουν εντολές που ανιχνεύουν ένα λάθος, δίνουν την περιγραφή του (αριθμητικά ή και με μήνυμα) και επιτρέπουν τη μεταβίβαση της ροής εκτέλεσης του προγράμματος.

Ανακεφαλαίωση

Το ζητούμενο κάθε χρήστη είναι ένα φιλικό και σταθερό πρόγραμμα που θα εργάζεται κάτω από οποιεσδήποτε συνθήκες και θα προλαβαίνει όλες τις εσφαλμένες του επιλογές. Η σωστή εκσφαλμάτωση ενός προγράμματος έχει ως αποτέλεσμα την κατασκευή τέτοιων προγραμμάτων που θα παρουσιάσουν όσο το δυνατό λιγότερα προβλήματα στο χρήστη.

Η εκσφαλμάτωση αποτελεί μια από τις βασικές εργασίες που πρέπει να κάνει ο προγραμματιστής για την ολοκλήρωση ενός προγράμματος. Κατά τη διαδικασία της εκσφαλμάτωσης ο προγραμματιστής πρέπει να εντοπίσει και να διορθώσει τα σημεία του προγράμματος που τυχόν προκαλούν λάθη. Για την επίτευξη του σκοπού αυτού πρέπει να κάνει τους κατάλληλους ελέγχους και να αναλύσει το πρόγραμμά του μέσα από τα εργαλεία εκσφαλμάτωσης που του παρέχει το περιβάλλον προγραμματισμού.

Τέλος ο προγραμματιστής για να δημιουργήσει ασφαλή και ποιοτικά προγράμματα, πρέπει να προβλέψει όλες τις καταστάσεις λάθους που μπορεί να συμβούν στο περιβάλλον εκτέλεσης από κακό χειρισμό ή προβληματικό εξοπλισμό και να δώσει στο πρόγραμμά του δυνατότητες αντιμετώπισης και τρόπους διαφυγής.

Λέξεις κλειδιά

Λάθος, έλεγχος, εκσφαλμάτωση, εργαλεία εκσφαλμάτωσης, χειρισμός λάθους.

Ερωτήσεις - Θέματα για συζήτηση

1. Να δοθούν παραδείγματα λαθών που εμφανίζονται συχνά στο λογισμικό.

2. Να γίνει συζήτηση σχετικά με τις επιπτώσεις των λαθών του λογισμικού.
3. Περιγράψτε το χειρισμό ενός λάθους που εμφανίζεται κατά το χρόνο εκτέλεσης ενός προγράμματος.
4. Περιγράψτε τη χρησιμότητα των εργαλείων εκσφαλμάτωσης.
5. Να γίνει συζήτηση για τις αιτίες που οδηγούν στην κατασκευή προβληματικού λογισμικού.

Βιβλιογραφία

1. Εγκυκλοπαίδεια Πληροφορικής και Τεχνολογίας Υπολογιστών, Εκδόσεις Νέων Τεχνολογιών, Τόμος 2, Αθήνα, 1986.
2. Παν. Πολίτης-Ηλ. Γιαννόπουλος: Προγραμματισμός με τη Visual Basic 4.0, Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1997.
3. Ivars Peterson: Fatal Defect:Chasing killer Computer Bugs, Science News, USA, 1995.