Εισαγωγή στον Προγραμματισμό

Σύντομες Σημειώσεις

Γιώργος Μανής

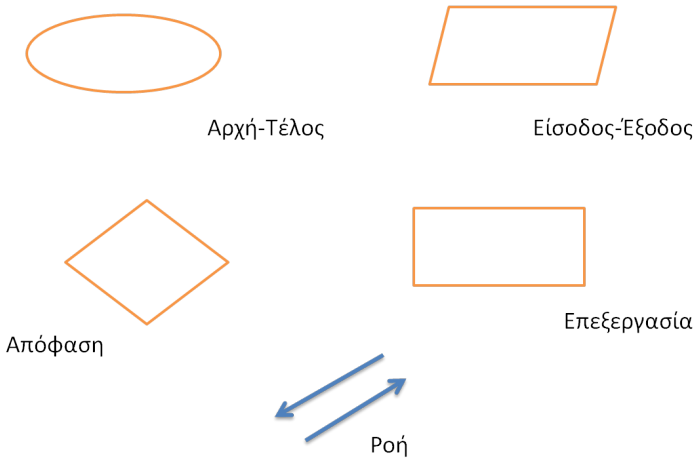
Νοέμβριος 2012

Αλγόριθμοι και Λογικά Διαγράμματα

Αλγόριθμος λέγεται μία πεπερασμένη διαδικασία καλά ορισμένων βημάτων μου ακολουθείται για τη λύση ενός προβλήματος, ενώ το *λογικό διάγραμμα (ΛΔ)* ή αλλιώς *διάγραμμα ροής προγράμματος (ΔΡΠ)* είναι μία σχηματική παράσταση του αλγορίθμου. Η έννοια του αλγορίθμου είναι πολύ βασική στην πληροφορική, ενώ το λογικό διάγραμμα ένα πολύ χρήσιμο περιγραφικό εργαλείο. Ένα λογικό διάγραμμα αποτελείται από σύμβολα και λέξεις οι οποίες είναι σε θέση να περιγράψουν με λεπτομέρεια κάθε διαδικασία επίλυσης οποιουδήποτε προβλήματος.

Η κατασκευή ενός λογικού διαγράμματος προηγείται της κωδικοποίησης. Το λογικό διάγραμμα δίνει τη δυνατότητα να μελετήσουμε καλά ένα πρόβλημα και να εμβαθύνουμε σε αυτό, επιτρέπει να το σχεδιάσουμε στην ολότητά του και να το δούμε λυμένο (ή να διαπιστώσουμε αν λύνεται) πριν ακόμα ξεκινήσουμε την επίπονη διαδικασία της κωδικοποίησης, αλλά και να συγκρίνουμε διαφορετικές μεταξύ τους λύσεις και να επιλέξουμε την καλύτερη αρκετά νωρίς, αφού οι αλλαγές κατά την κωδικοποίηση είναι πολύ πιο ακριβές από ότι οι αλλαγές κατά τη σχεδίαση ενός συστήματος.

Τα βασικότερα σύμβολα που χρησιμοποιούνται από ένα λογικό διάγραμμα είναι η έλλειψη, το ορθογώνιο και το πλάγιο παραλληλόγραμμο, ο ρόμβος και τα βέλη (Σχήμα 1). Οι ελλείψεις χρησιμοποιούνται για να συμβολίσουν την αρχή και το τέλος ενός αλγορίθμου. Τα ορθογώνια παραλληλόγραμμα συμβολίζουν επεξεργασία. Τα πλάγια παραλληλόγραμμα είσοδο και έξοδο δεδομένων, ενώ οι ρόμβοι αποτελούν σημεία στα οποία πρέπει να ληφθεί κάποια απόφαση. Με τα βέλη απεικονίζουμε τη ροή του προγράμματος. Πέρα από αυτά τα σύμβολα, υπάρχουν μερικά ακόμα, δευτερεύουσας σημασίας, αλλά



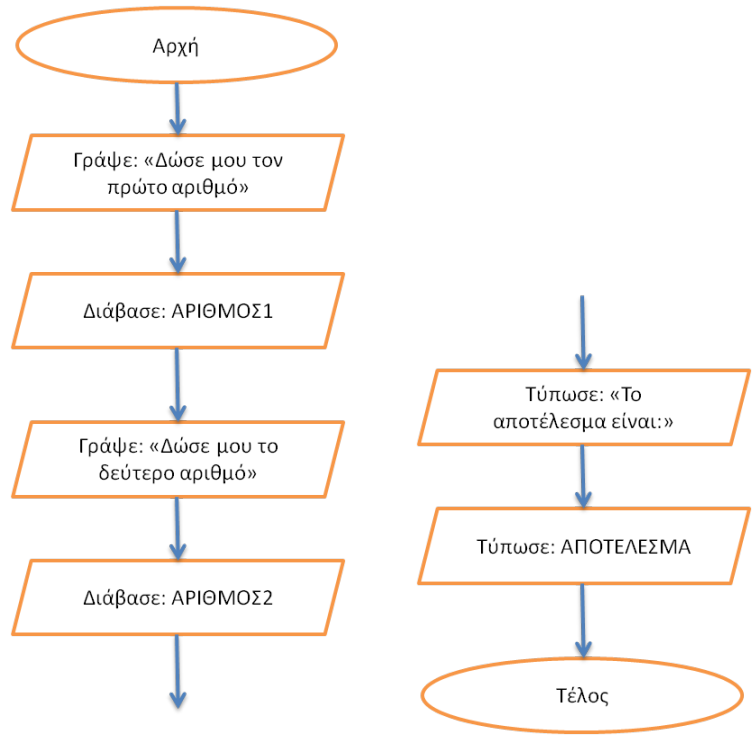
Σχήμα 1: Τα βασικότερα σύμβολα που χρησιμοποιούνται στα λογικά διαγράμματα

επίσης χρήσιμα. Ας μείνουμε τώρα σε αυτά και ας τα δούμε αναλυτικά.

Η έλλειψη χρησιμοποιείται για να σημειωθεί η αρχή και το τέλος του αλγορίθμου. Το πλάγιο παραλληλόγραμμο περιγράφει διαδικασία εισόδου ή εξόδου στοιχείων. Στο παράδειγμα της πρόσθεσης είσοδο δεδομένων έχουμε στην αρχή του προγράμματος όπου και πρέπει να δοθούν από το χρήστη στον υπολογιστή οι δύο αριθμοί που θα προστεθούν. Σε εκείνο το σημείο έχουμε και έξοδο δεδομένων, αφού για κάθε έναν από τους δύο αριθμούς που ζητούνται προηγείται ένα μήνυμα διευκρινιστικό προς το χρήστη (¨Δώσε μου τον πρώτο αριθμό¨ και ¨Δώσε μου τον δεύτερο αριθμό¨). Έξοδο έχουμε στο τέλος του προγράμματος, όπου έχουμε την εμφάνιση του αποτελέσματος στην οθόνη συνοδευόμενο πάλι από το κατάλληλο βοηθητικό μήνυμα (¨Το αποτέλεσμα είναι:¨). Τα μέρη του διαγράμματος ροής προγράμματος για το πρόβλημα της πρόσθεσης που αναλογούν στην είσοδο και έξοδο δεδομένων φαίνεται στα σχήματα 2(α΄) και 2(β΄) αντίστοιχα.

Ο ρόμβος χρησιμοποιείται για να συμβολιστούν σημεία στα οποία πρέπει να ληφθεί κάποια απόφαση. Σημεία δηλαδή, στα οποία το πρόγραμμα έχει περισσότερες από ένα δρόμο να επιλέξει ανάλογα με το αν ισχύουν ή όχι κάποιες συνθήκες. Στο πρόγραμμα της πρόσθεσης δεν απαιτείται ρόμβος, αλλά θα δούμε παρακάτω ένα άλλο πρόβλημα (την λύση του τριωνύμου) στο οποίο απαιτείται.

Το ορθογώνιο παραλληλόγραμμο χρησιμοποιείται για να περιγραφεί κά-



(α΄) Εισαγωγή δεδομένων για το πρόγραμμα της πρόσθεσης

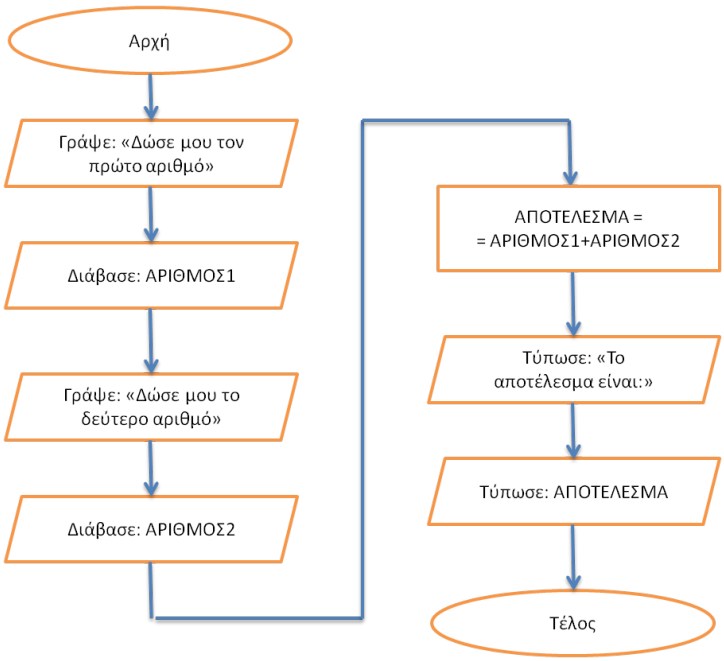
(β΄) Έξοδος δεδομένων για το πρόγραμμα της πρόσθεσης

Σχήμα 2: Είσοδος και έξοδος δεδομένων για την πρόσθεση

ποια διαδικασία. Στο παράδειγμα, θα συμβολιστεί με ορθογώνιο παραλληλόγραμμο η πρόσθεση των δύο αριθμών.

Τέλος, τα βέλη ροής δείχνουν ποιο είναι το επόμενο σχήμα στο λογικό διάγραμμα στο οποίο θα μεταβεί ο έλεγχος του προγράμματος. Από κάθε σχήμα μπορεί να μη ξεκινά κανένα βέλος (συμβαίνει όταν τερματίζεται το πρόγραμμα), μπορεί να ξεκινά ένα βέλος το οποίο καταλήγει στο επόμενο σχήμα στο οποίο θα περάσει η εκτέλεση ή περισσότερα βέλη όταν έχουμε ρόμβο απόφασης. Το διάγραμμα ροής προγράμματος για το πρόβλημα της πρόσθεσης εικονίζεται στο σχήμα 3.

Πριν όμως προχωρήσουμε στην κωδικοποίηση, ας δούμε και ένα λίγο μεγαλύτερο παράδειγμα, ένα πρόγραμμα που δέχεται σαν είσοδο ένα τριώνυμο και αποφασίζει αν το τριώνυμο έχει και πόσες πραγματικές ρίζες. Στην περίπτωση που το τριώνυμο έχει πραγματικές ρίζες τότε αυτές υπολογίζονται και τυπώνονται στην οθόνη. Στην περίπτωση που το τριώνυμο δεν έχει πραγμα-

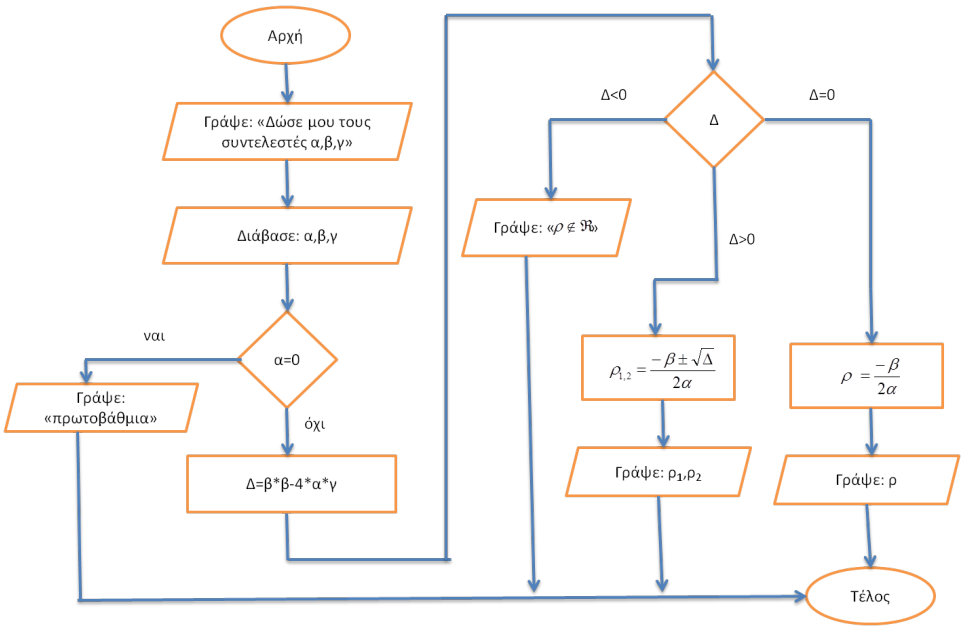


Σχήμα 3: Ολοκληρωμένο το λογικό διάγραμμα της πρόσθεσης

τικές ρίζες, τότε τυπώνεται ένα μήνυμα που μας ειδοποιεί ότι δεν υπάρχουν πραγματικές ρίζες.

Είσοδος στοιχείων απαιτείται κατά την εκκίνηση του προγράμματος όπου και ζητείται το τριώνυμο, οι συντελεστές α, β και γ δηλαδή. Έτσι, στην αρχή του λογικού διαγράμματος υπάρχουν τρία πλάγια παραλληλόγραμμα. Για να είναι το πρόγραμμα φιλικό στο χρήστη πριν από κάθε ερώτηση για τις τιμές των α, β και γ πρέπει να τυπώνεται ένα μήνυμα στην οθόνη που να πληροφορεί το χρήστη τι ακριβώς πρέπει να κάνει. Συνεπώς, πριν από κάθε πλάγιο παραλληλόγραμμο για την εισαγωγή μίας τιμής πρέπει να υπάρχει και ένα άλλο πλάγιο παραλληλόγραμμο που να τυπώνει ένα μήνυμα της μορφής: ¨Δώσε μου το συντελεστή α:¨, ή ¨Δώσε μου το συντελεστή β:¨ ή ¨Δώσε μου το συντελεστή γ:¨, αντίστοιχα. Στο τέλος του προγράμματος πρέπει να τυπώνεται το αποτέλεσμα. Έτσι, σε κάποιο σημείο του λογικού μας διαγράμματος πρέπει να υπάρχουν τα κατάλληλα πλάγια παραλληλόγραμμα για τα αποτελέσματα: ¨Το τριώνυμο δεν έχει πραγματικές ρίζες¨ και ¨Το τριώνυμο έχει μία πραγματική διπλή ρίζα την:¨ και ¨Το τριώνυμο έχει δύο πραγματικές ρίζες τις:¨. Επίσης, αν διαπιστωθεί ότι η εξίσωση είναι πρωτοβάθμια πρέπει να τυπωθεί το κατάλληλο μήνυμα ¨Η εξίσωση είναι πρωτοβάθμια¨ (Αν και αυτό μαθηματικά

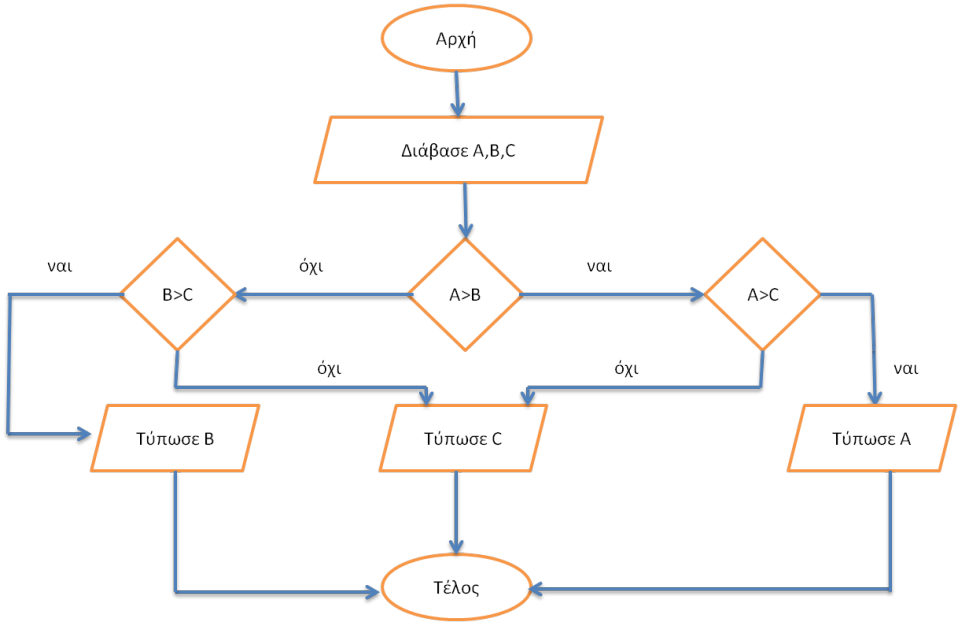
δεν είναι απόλυτα σωστό, αλλά ας κρατήσουμε το πρόγραμμα σχετικά απλό). Μπορεί κανείς να παρατηρήσει ότι σε ένα τόσο απλό πρόγραμμα και για μια τόσο απλή διαδικασία απαιτούνται τόσα πολλά πλάγια παραλληλόγραμμα. Το ίδιο πρόβλημα υπάρχει με όλα τα σχήματα του λογικού διαγράμματος. Για να μειώσουμε την έκτασή του συνηθίζουμε να ομαδοποιούμε λειτουργίες περισσοτέρων σχημάτων σε ένα, όταν αυτό δεν αποβαίνει σε βάρος της περιγραφής του αλγορίθμου. Στο συγκεκριμένο παράδειγμα μπορούμε να χρησιμοποιήσουμε ένα ή δύο πλάγια παραλληλόγραμμα αντί για έξι στην αρχή του προγράμματος, ενώ από ένα μπορεί να χρησιμοποιηθεί και για κάθε μία από τις πιθανές περιπτώσεις αποτελέσματος.



Σχήμα 4: Διάγραμμα ροής για την επίλυση του τριωνύμου

Στο παράδειγμα του τριωνύμου χρησιμοποιείται και ο ρόμβος. Ο αλγόριθμος σε κάποιο σημείο εξετάζει εάν η διακρίνουσα είναι θετική, αρνητική ή μηδέν και αποφασίζει εάν υπάρχουν και πόσες πραγματικές ρίζες. Σε κάθε μία από τις τρεις περιπτώσεις το πρόγραμμα ακολουθεί διαφορετικό δρόμο και κάνει διαφορετικούς υπολογισμούς. Είναι ένα χαρακτηριστικό παράδειγμα λογικής απόφασης. Στο ίδιο πρόγραμμα πρέπει να γίνει έλεγχος εάν το α είναι μηδέν ή όχι, γιατί σε αυτή την περίπτωση η λύση είναι πρωτοβάθμια. Πάλι δηλαδή έχουμε μία περίπτωση απόφασης με δύο δρόμους αυτή τη φορά,

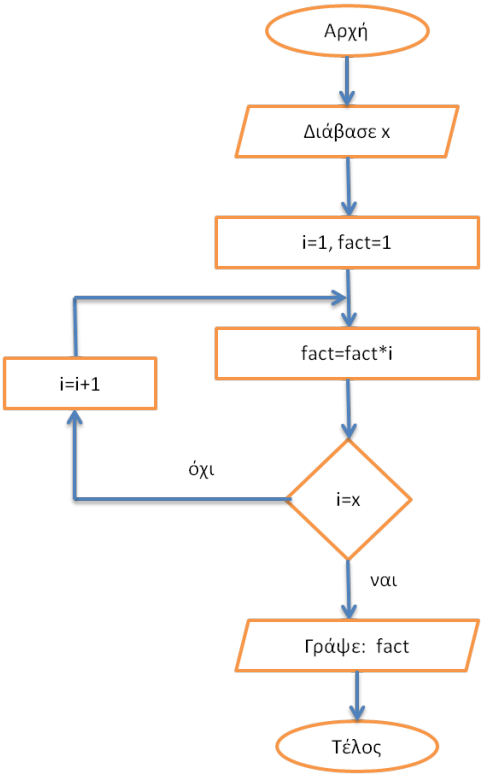
έναν που το πρόγραμμα θα συνεχίζει κανονικά στην επίλυση του τριωνύμου και έναν που θα τυπώνει το μήνυμα ¨Η επίλυση ανάγεται πρωτοβάθμια¨ και θα τερματίζεται το πρόγραμμα. Το διάγραμμα ροής για το τριώνυμο φαίνεται στο σχήμα 4.



Σχήμα 5: Διάγραμμα ροής για την εύρεση του μέγιστου τριών αριθμών

Ας δούμε λίγα ακόμα ενδιαφέροντα διαγράμματα ροής προγράμματος. Στο σχήμα 5 φαίνεται ένα διάγραμμα ροής στο οποίο τρεις αριθμοί συγκρίνονται έτσι ώστε να βρεθεί ποιος από τους τρείς είναι ο μεγαλύτερος. Δε μας ενδιαφέρει να διαπιστώσουμε ότι για παράδειγμα ο δεύτερος αριθμός είναι ο μεγαλύτερος, αλλά ποια τιμή έχει ο μεγαλύτερος από τους τρεις αριθμούς. Δηλαδή, αν έχουμε σαν είσοδο τους αριθμούς 3,8,5 η απάντηση είναι 8. Το λογικό διάγραμμα ξεκινάει εισάγωντας τους τρεις αριθμούς, έστω A,B,C. Στη συνέχεια εξετάζουμε ποιος από τους δύο πρώτους (A και B) είναι ο μεγαλύτερος. Αν ο A είναι ο μεγαλύτερος, τότε αποκλείουμε την περίπτωση ο B να είναι ο μεγαλύτερος από τους τρεις. Έτσι συνεχίζουμε την αναζήτηση ανάμεσα στους A και C. Όμοια, αν ο B ήταν ο μεγαλύτερος, τότε θα αποκλείαμε την περίπτωση ο A να ήταν ο μεγαλύτερος από τους τρεις. Έτσι θα συνεχίζουμε την αναζήτηση ανάμεσα στους B και C. Η δεύτερη σύγκριση ανάμεσα στους δύο πιθανούς αριθμούς, αφού αποκλείστηκε ο ένας από τους τρεις, μας

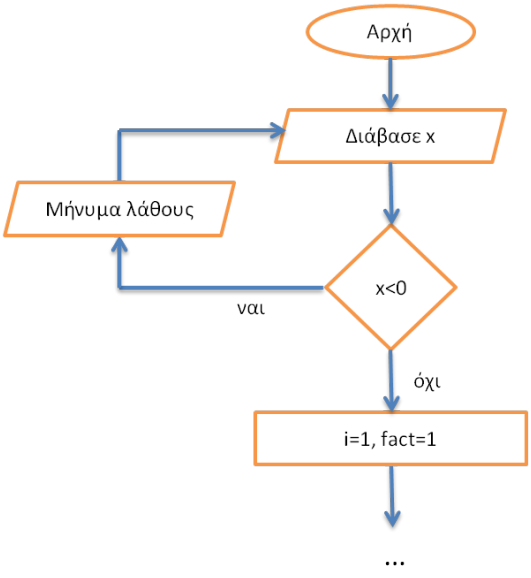
φανερώνει το μέγιστο ο οποίος και τυπώνεται. Παρατηρείστε ότι το διάγραμμα ροής προγράμματος επιστρέφει σωστό αποτέλεσμα στην περίπτωση που ο μεγαλύτερος αριθμός εμφανίζεται σε δύο ή και στις τρεις από τις μεταβλητές A,B,C. Ας δούμε γιατί: έστω $A = B$ και $A > C$ και φυσικά $B > C$. Η πρώτη σύγκριση θα μας οδηγήσει στο ρόμβο $B > C$ και στη συνέχεια θα τυπωθεί το B. Δε μας πειράζει ότι το αποτέλεσμα τυπώθηκε από το B. Θα μπορούσε να είχε τυπωθεί και από το A αν είχαμε σχεδιάσει διαφορετικά το διάγραμμα. Σε κάθε περίπτωση όμως και ανεξάρτητα από το αν θα τυπωθεί το αποτέλεσμα από το A ή το B ο αριθμός που θα τυπωθεί θα έχει τη μέγιστη τιμή, την τιμή δηλαδή που αναζητούσαμε. Σημειώνουμε πάλι ότι το διαγραμμα ροής θα ήταν διαφορετικό αν δε μας έννοιαζε να τυπωθεί η μέγιστη τιμή, αλλά μας ένοιαζε και από ποια ή ποιες απο τις A,B,C προήλθε αυτή η μέγιστη τιμή.



Σχήμα 6: Διάγραμμα ροής για την εύρεση του παραγοντικού

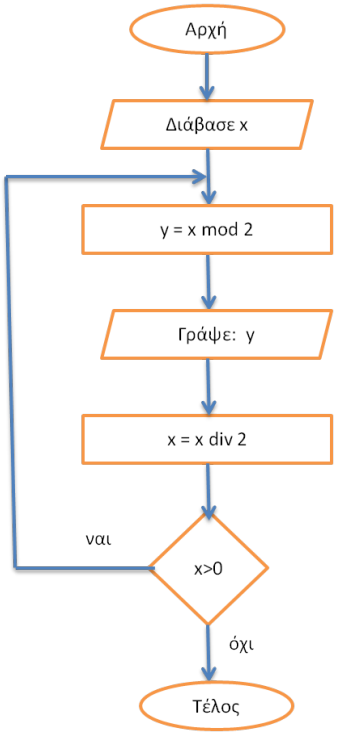
Στο σχήμα 6 παρουσιάζεται το διάγραμμα ροής προγράμματος για τον υπολογισμό του παραγοντικού. Για να υπολογίσουμε το παραγοντικό του x θα πρέπει να υπολογίσουμε την παράσταση $1 * 2 * 3 * \ldots * x$. Χρειαζόμαστε

δηλαδή μία μεταβλητή που θα ξεκινήσει από την τιμή 1, θα αυξάνεται σε κάθε βήμα κατά 1 και θα καταλήγει να έχει την τιμή x. Ας ονομάσουμε τη μεταβλητή αυτή i. Σε κάθε ένα βήμα θα πρέπει μία άλλη μεταβλητή να πολλαπλασιάζει την τιμή που είχε στο προηγούμενο βήμα με την τιμή της i. Ας την ονομάσουμε $fact$. Δείτε στη μέση του σχήματος 6 την αύξηση των μεταβλητών i και $fact$ κατά 1 και i φορές αντίστοιχα. Παρατηρήστε στην αρχή του διαγράμματος την αρχικοποίηση των μεταβλητών αυτών σε 1. Το i είναι προφανές γιατί πρέπει να αρχικοποιηθεί σε 1, αφού είπαμε ότι θέλουμε να υπολογίσουμε την παράσταση $1 * 2 * 3 * \ldots * x$. Το $fact$ πρέπει και αυτό να αρχικοποιηθεί σε 1, και ο λόγος είναι ότι θέλουμε να το αρχικοποιήσουμε στο ουδέτερο στοιχείο του πολλαπλασιασμού. Σκεφτείτε τι θα γινόταν αν το αρχικοποιούσαμε στο 0. Το $fact$ θα είχε πάντοτε την τιμή 0, αφού με ό,τι και να το πολλαπλασιάζαμε το αποτέλεσμα θα ήταν 0. Τώρα, σχεδόν έχουμε περιγράψει όλο το διάγραμμα εκτός από το ρόμβο που συγκρίνει το i με το x. Μετά την αύξηση της τιμής του $fact$, το i ελέγχεται για να διαπιστωθεί εάν έχουν γίνει όλες οι επαναλήψεις που απαιτούνται. Εάν έχουν γίνει, δηλαδή αν οι δύο μεταβλητές έχουν την ίδια τιμή, τότε ο υπολογισμός του παραγοντικού έχει τελειώσει και τυπώνουμε το αποτέλεσμα. Εάν όχι, οδηγούμε τον έλεγχο στο σημείο που αυξάνεται το i κατά 1 και εκτελούμε ακόμα μία επανάληψη.



Σχήμα 7: Έλεγχος ορθής εισόδου για το ΔΡΠ του παραγοντικού

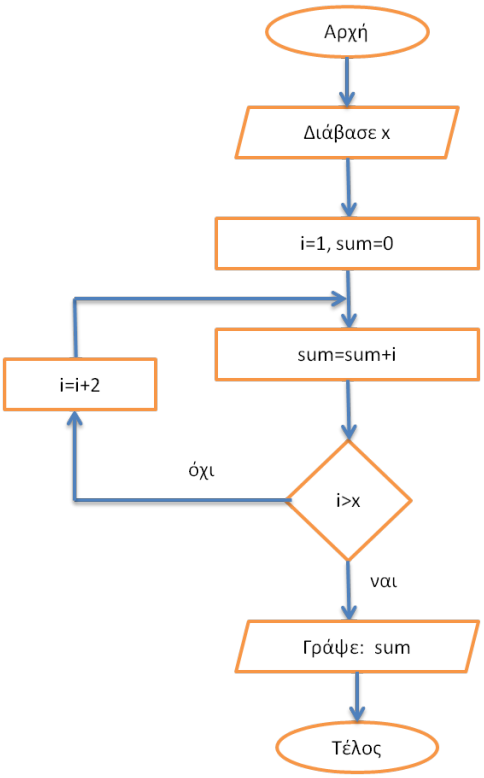
Το διάγραμμα ροής προγράμματος για τον υπολογισμό του παραγοντικού φαίνεται να λειτουργεί. Για να είμαστε όμως περισσότερο ορθοί θα πρέπει να ελέξουμε ότι η τιμή που δόθηκε από το χρήστη για τη μεταβλητή x είναι νόμιμη. Το παραγοντικό ορίζεται για αριθμούς μεγαλύτερους ή ίσους από το 0. Έτσι εάν από το χρήστη δοθεί αρνητικός αριθμός θα πρέπει να εμφανιστεί κάποιο κατάλληλο μήνυμα λάθους και να επιστρέψει ο έλεγχος στην εντολή που ζητά να δοθεί τιμή για το x. Έτσι, όσο ο χρήστης δίνει αρνητικές τιμές για το x θα του εμφανίζεται το μήνυμα λάθους και θα του ζητείται να δώσει εκ νέου τιμή. Το τμήμα του διαγράμματος ροής που υλοποιεί την παραπάνω λειτουργία φαίνεται στο σχήμα 7.



Σχήμα 8: Μετατροπή δεκαδικού αριθμού σε δυαδικό

Όλα τώρα φαίνεται να λειτουργούν σωστά. Ή μήπως όχι; Αναφέραμε προηγουμένως ότι το παραγοντικό ορίζεται για αριθμούς μεγαλύτερους ίσους από το μηδέν. Εμείς στον συλλογισμό που κάναμε για το λογικό διάγραμμα δεν εξετάσαμε καθόλου τι γίνεται για $x = 0$ και μάλιστα αρχικοποιήσαμε την τιμή του $fact$ σε ένα. Είναι σωστό αυτό που κάναμε ή πρέπει να διορθώουμε κάτι. Κατ' αρχήν ας θυμηθούμε ότι το παραγοντικό του 0 είναι ίσο με 1

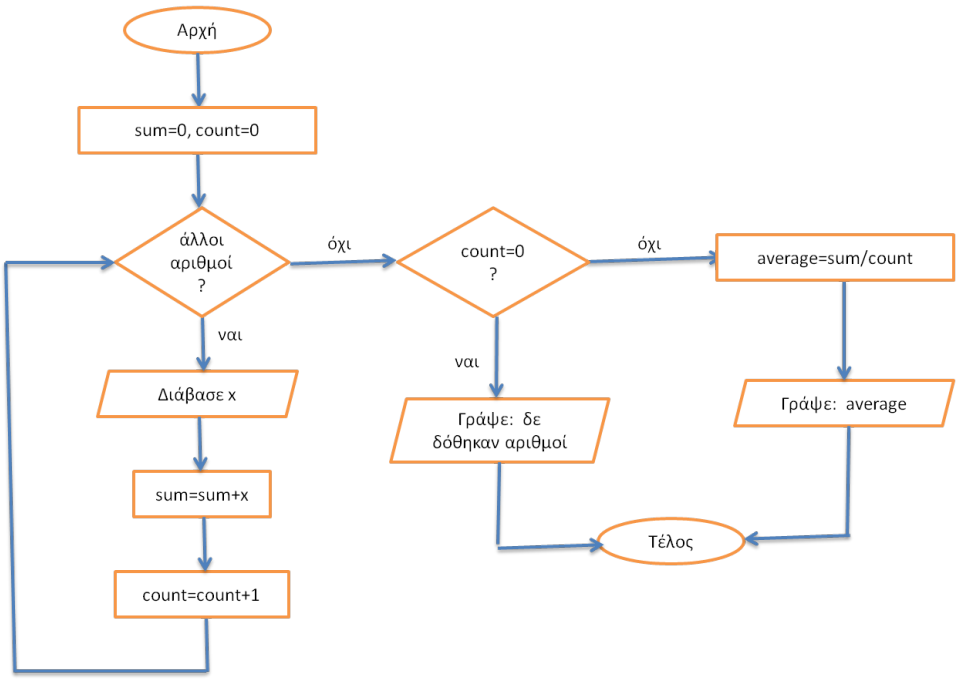
(0!=1). Ας δούμε τι θα βγάλει το διάγραμμα ροής προγράμματος αν δώσουμε στο x την τιμή 1. Το $fact$ θα αρχικοποιηθεί σε 1, θα εκτελεστεί μία φορά η $fact = fact * i$, χωρις να αλλάξει η τιμή του $fact$ αφού το i ισούται με 1 και δε θα γίνει άλλη επανάληψη αφού ο ρόμβος θα οδηγήσει την εκτέλεση έξω από το βρόχο (το i ισούται με x). Τώρα έχουμε βεβαιωθεί ότι όλα έχουν πάει καλά και μπορούμε να θεωρήσουμε ότι ολοκληρώθηκε το διάγραμμα ροής.



Σχήμα 9: Άθροισμα περιττών αριθμών

Παραθέτουμε ακόμα τρία ακόμα λογικά διαγράμματα. Στο σχήμα 9 φαίνεται το άθροισμα των περιττών αριθμών μικρότερων ή ίσων από ένα δεδομένο αριθμό. Στο σχήμα 8 φαίνεται ο τρόπος μετατροπής ενός δεκαδικού αριθμού σε δυαδικό, ενώ στο σχήμα 10 ο τρόπος υπολογισμού του μέσου όρου μίας σειράς αριθμών.

Έχοντας μελετήσει τα λογικά διαγράμματα των προηγούμενων παραδειγμάτων είμαστε σε θέση να αρχίσουμε την κωδικοποίηση. Σε ποια γλώσσα όμως; Γιατί όχι στη C, αφού είναι ευρέως χρησιμοποιούμενη και αποτελεί τη βάση για πολλές από τις περισσότερο χρησιμοποιούμενες γλώσσες, ενώ αυτή



Σχήμα 10: Μέσος όρος σειράς αριθμών

και όσες γλώσσες έχουν προκύψει από αυτήν ζητούνται στην αγορά εργασίας. Ίσως είναι λίγο νωρίς να μπούμε από τώρα στα βαθιά. Θα αναζητήσουμε τον κοινό παρονομαστή όλων των γλωσσών προγραμματισμού που ανήκουν στην κατηγορία των γλωσσών του δομημένου προγραμματισμού (όσο αυτό βέβαια είναι δυνατό), και θα διδακτούμε αυτό. Σαν βάση, λοιπόν, θα χρησιμοποιήσουμε τη γλώσσα Python αλλά σκοπός μας δεν είναι να μάθουμε Python και να εμβαθύνουμε σε αυτή, αλλά να τη χρησιμοποιήσουμε ως μέσο για να μάθουμε αυτό που καλείται *δομημένος προγραμματισμός*.