ΠΛΗΡΟΦΟΡΙΚΟΣ

Περιοδική έκδοση της Ένωσης Πληροφορικών Ελλάδας

Τεύχος 20ο Μάρτιος 2023 Διανέμεται ελεύθερα

Περιοδική έκδοση της
Ένωσης Πληροφορικών Ελλάδας
www.epe.org.gr

Τεύχος 20ο – Μάρτιος 2023

Διανέμεται ελεύθερα

Επικοινωνία:
newsletter@epe.org.gr

ΠΕΡΙΕΧΟΜΕΝΑ

✔ Ανακοίνωση σχετικά με το πολύνεκρο δυστύχημα τρένων στα Τέμπη

8-3-2023

Η **Ένωση Πληροφορικών Ελλάδας** (ΕΠΕ) συμπαρίσταται στην απώλεια και στη θλίψη για τα θύματα του πολύνεκρου -αυτή τη φορά- σιδηροδρομικού δυστυχήματος που συνέβη πριν λίγες μέρες στα Τέμπη.

Κανείς και τίποτα δεν μπορεί να αναπληρώσει το κενό που βιώνουν, όμως είναι πλέον η στιγμή να μελετήσουμε με ψυχραιμία τι δεν λειτούργησε σωστά και πώς μπορούμε να συμβάλλουμε όλοι με τον τρόπο μας στο να μην συμβεί ποτέ ξανά.

Σήμερα Πληροφορική δεν σημαίνει απλώς "έξυπνες" εφαρμογές στα κινητά τηλέφωνα ή όμορφα γραφικά σε παιχνίδια Η/Υ. Σημαίνει πρωτίστως συστήματα αυτοματισμού, τηλεματικής και τηλεμετρίας, αξιόπιστες τηλεπικοινωνίες, αυτόματη ανάλυση ρίσκου και πρόβλεψη κινδύνου σε πραγματικό χρόνο, διασφάλιση ποιότητας και διαχείρισης σφαλμάτων σε λογισμικά κρίσιμα για την ανθρώπινη ζωή (safety critical software).

Ο λόγος που σήμερα ένα τέτοιο δυστύχημα πρέπει να είναι, αν όχι αδύνατο, τουλάχιστον εξαιρετικά απίθανο να συμβεί, είναι γιατί ο ανθρώπινος παράγοντας και η σύγχρονη τεχνολογία πρέπει και μπορεί να λειτουργεί συνεργατικά και τα δύο μέρη να αλληλοσυμπληρώνονται. Είναι ακριβώς ο λόγος που, αντίστοιχα, οι αερομεταφορές θεωρούνται το ασφαλέστερο μέσο μεταφοράς, παρότι πρόκειται για εξαιρετικά σύνθετο σύστημα στην αξιοπιστία του οποίου βασίζονται καθημερινά οι ζωές εκατομμυρίων επιβατών σε όλο τον πλανήτη. Αντίστοιχες τεχνολογίες, διαδικασίες και συστήματα είναι διαθέσιμα εδώ και δεκαετίες και στα δίκτυα σταθερής τροχιάς.

Στο ποσοστό και στο πεδίο που μας αναλογεί, η ΕΠΕ δεσμεύεται να συμβάλλει στο μέγιστο δυνατό βαθμό στο έργο κάθε συναρμόδιου φορέα, μελέτη έργου και ομάδας εργασίας, εφόσον μας ζητηθεί. Είναι το ελάχιστο που μπορούμε να συνεισφέρουμε ως επιστημονικός και επαγγελματικός φορέας προς αυτή την κατεύθυνση.

Το Διοικητικό Συμβούλιο της Ένωσης Πληροφορικών Ελλάδας

✔ Συνέντευξη με τον κ. Δημήτρη Τσίγκο

// Επιχειρηματία, ιδρυτή της Starttech Ventures.
ιδρυτή της Ελληνικής Ένωσης Νεοφυών Επιχειρήσεων

Το 2000 μαζί με άλλους προπτυχιακούς φοιτητές στο Πανεπιστήμιο της Κρήτης ιδρύσατε την εταιρεία πληροφορικής Virtual Trip και ήταν το πρώτο φοιτητικό startup στην Ελλάδα! Σήμερα έχουμε φτάσει στο σημείο να εξαγοράζονται ελληνικές τεχνολογικές start-ups από παγκόσμιους κολοσσούς για ιλιγγιώδη πόσα. Μιλήστε μας γι' αυτή τη διαδρομή της ελληνικής start-up σκηνής όπως τη ζήσατε και πως βλέπετε να εξελίσσεται στο μέλλον.

Πράγματι, η Virtual Trip που ιδρύθηκε τον Σεπτέμβριο του 2000 στο Επιστημονικό & Τεχνολογικό Πάρκο Κρήτης ήταν από τα πρώτα tech startups της χώρας μας και πιθανότατα το πρώτο με πρωτοβουλία και υπό τον έλεγχο

προπτυχιακών φοιτητών. Είναι ενδιαφέρον πως τότε στην Ελλάδα η ίδια η λέξη startup δεν ήταν σε χρήση (πολύ περισσότερο η ελληνική "νεοφυής" που την εισαγάγαμε το 2011 με την Ελληνική Ένωση Νεοφυών Επιχειρήσεων). Αυτό που έχει ενδιαφέρον είναι να μελετήσει κανείς τα χαρακτηριστικά της φοιτητικής, ακαδημαϊκής και ερευνητικής κοινότητας του Τμήματος Επιστήμης Υπολογιστών εκείνης της εποχής στο Ηράκλειο, τα οποία επέτρεψαν σε κάτι τέτοιο να συμβεί.

Οι ιδρυτές της Virtual Trip, Οκτώβριος 2000. Από αριστερά: Δημήτρης Τσίγκος, Μίλτος Βασιλάκης, Νίκος Βεντούρας, Περικλής Ακριτίδης, Χάρης Γκίκας.

Νομίζω πως το σχετικά μικρό μέγεθος (δυνατότητα προσωπικών γνωριμιών), η κινητικότητα των ανθρώπων και των ιδεών (πολλοί φοιτητές στο Erasmus, πολλοί καθηγητές από Αμερική), η κουλτούρα του πειραματισμού, η - υγιής - άγνοια κινδύνου και η αυτοπεποίθηση, ήταν κάποια από τα χαρακτηριστικά της κοινότητας που την έκαναν γόνιμη για την ανάπτυξη της επιχειρηματικότητας.

Είναι χαρακτηριστικό πως από την ίδια κοινότητα, παρά το μικρό της μέγεθος, έχουν αναδειχθεί δεκάδες επιτυχημένες νεοφυείς επιχειρήσεις, τόσο από φοιτητές / αποφοίτους όσο και από καθηγητές / ερευνητές. Ενδεικτικά αναφέρω τις Cytech, Niometrics, Elorus, AbZorba Games, ORamaVR, JADBio και OpenIT, ενώ υπάρχουν πολλές ακόμα, ιδιαίτερα αξιόλογες.

Όσο για τις εξαγορές τις οποίες αναφέρονται, νομίζω δίνουμε λάθος μήνυμα εάν επικεντρώνουμε τη συζήτηση σε αυτές, πολλές φορές και σε έναν άτυπο 'διαγωνισμό' για το 'μεγαλύτερο exit' (ή, ακόμα χειρότερα, για την μεγαλύτερη χρηματοδότηση). Οι εξαγορές έχουν μια καλή διάσταση καθότι δημιουργούν τη νέα γενιά ιδιωτών επενδυτών (angel investors), οι οποίοι κατά κανόνα επανεπενδύουν σε παρόμοιες επιχειρήσεις, δημιουργώντας ένα πολλαπλασιαστικό φαινόμενο (Πρόσφατα έχει δημιουργηθεί το Ελληνικό Δίκτυο Επιχειρηματικών Αγγέλων). Πρέπει όμως να είμαστε ειλικρινείς λέγοντας πως οι εξαγορές έχουν και μια θλιβερή πλευρά, εκείνη του (συχνά πρόωρου) τέλους της διαδρομής. Ας επικεντρωθούμε λοιπόν στην ανάπτυξη πολλών, υγιών, ισχυρών οργανισμών, ανθεκτικών μεταξύ των διαφορετικών οικονομικών κύκλων, οι οποίες λύνουν πραγματικά και σημαντικά προβλήματα, προσφέρουν ποιοτικές θέσεις εργασίας και υπεραξία στους εργαζομένους τους και ανταποκρίνοντα επαρκώς και στις υποχρεώσεις τους στο κοινωνικό σύνολο. Από αυτήν την οπτική γωνία, αναμφίβολα έχει σημειωθεί σημαντική πρόοδος, επίσης αναμφίβολα όμως είμαστε ακόμα στα πρώτα βήματα της διαδρομής.

Στην τεχνολογική κοσμογονία που ζούμε, τι είναι εκείνο που κάνει μια ιδέα μεγάλη και επιδραστική;

Όσον αφορά τις επιχειρηματικές ιδέες, υπάρχει ένας ουσιώδης διαχωρισμός μεταξύ των αναγκαίων (must-have) και των επιθυμητών (nice-to-have). Όταν κάποιος ξεκινά την υλοποίηση μιας ιδέας, οι περισσότεροι θα τον ενθαρρύνουν, λέγοντάς του (ειλικρινά) πως είναι ωραία και ενδιαφέρουσα. Ένα μικρό ποσοστό όμως αυτών τελικά θα είναι *διατεθειμένο να πληρώσει* για το νέο προϊόν ή υπηρεσία. Αυτή είναι η πιο συχνή αιτία αποτυχίας νέων επιχειρήσεων και έτσι νομίζω πως αξιολογούνται οι επιχειρηματικές ιδέες.

Αν τώρα μιλήσουμε για τις ιδέες γενικότερα, πρέπει να διακρίνουμε μεταξύ αυτών που ανατρέπουν δημιουργικά ένα "κατεστημένο" και όλων των υπολοίπων. Τις πρώτες ο συγγραφέας και επενδυτής Peter Thiel τις περιγράφει ως αυτές που μα πάνε από το 0 έως το 1, σε αντιδιαστολή από τις άλλες που μας πάνε από το 1 έως το άπειρο.

Αυτό το βήμα της δημιουργίας του νέου από το "τίποτα" (που ποτέ βέβαια δεν είναι ακριβώς έτσι), είναι νομίζω εκείνο που κάνει την ιδέα μεγάλη επιδραστική (και που, αργά ή γρήγορα, θα επηρεάσει τη ζωή μεγάλου αριθμού ανθρώπων).

Ποια η δική σας γνώμη για την θέση της Πληροφορικής αυτή τη στιγμή στη χώρα μας: για τους έλληνες Πληροφορικούς, για την Εκπαίδευση που παρέχεται, για τη λειτουργία του Δημόσιου Τομέα και για τον ψηφιακό εγγραμματισμό του κόσμου.

Παρά τις φιλότιμες, άοκνες θα έλεγα, προσπάθειες οργανισμών όπως η Ένωση Πληροφορικών Ελλάδας, η Πληροφορική στη χώρα μας ακόμα δεν έχει βρει τη θέση που της αξίζει. Εξακολουθεί να αντιμετωπίζεται ως κάτι "εξωτικό" και όχι σαν τη "νέα καθημερινότητα", όπως προφανώς είναι. Εν έτει 2023 αναρωτιέμαι αν ο μη έχων πληροφορική παιδεία θα έπρεπε να χαρακτηρίζεται ως αναλφάβητος ή όχι (μάλλον θα έπρεπε). Το ελληνικό κράτος όμως, επιβεβαιώνοντας τον χαρακτηρισμό του ως περιφερειακού / δορυφορικού κράτους, εδώ και 30 χρόνια αρνείται πεισματικά να κάνει τα δέοντα ώστε η πληροφορική να βρει τη θέση της στην κοινωνία και στην οικονομία, ξεκινώντας φυσικά από την εκπαίδευση, προς όφελος όλων των κατοίκων της χώρας.

Ενδεικτικό θα έλεγα του ακραίου επαρχιωτισμού που κυριαρχεί στο ζήτημα είναι ότι παρατηρεί κανείς συμπολίτες μας να μιλούν για την "ανάγκη εισαγωγής της τεχνητής νοημοσύνης στην εκπαίδευση", όταν οι ίδιοι *μειώνουν τις ώρες πληροφορικής στο ωρολόγιο πρόγραμμα* και συνολικά υποβαθμίζουν την πληροφορική παιδεία.

Ένα άλλο παράδειγμα από την επικαιρότητα αφορά το τραγικό σιδηροδρομικό δυστύχημα των Τεμπών. Μας δίδαξε με τον πλέον τραγικό και επώδυνο τρόπο τι σημαίνει "ψηφιοποίηση" και ότι αυτή σε καμία των περιπτώσεων δεν εξαντλείται σε χαριτωμένες διαδικτυακές φόρμες για υπεύθυνες δηλώσεις - οι οποίες είναι

απαραίτητες και αυτονόητες (εδώ και 20+ χρόνια), σε καμία περίπτωση όμως επαρκείς.

Πρέπει όμως να δώσουμε ένα θετικό μήνυμα: Η χώρα διαθέτει μεγάλο αριθμό επιστημόνων & μηχανικών πληροφορικής, όπως επίσης και πολύ μεγάλο αριθμό μαθηματικών, που είναι μια ομοίως κρίσιμη περιοχή. Το ανθρώπινο κεφάλαιο αυτό είναι υπερ-πολύτιμο και και αξιοποιηθεί σωστά μπορεί να κάνει θαύματα.

Παρουσίαση στο Ηράκλειο Κρήτης, Σεπτέμβριος 2004

Πρόσφατα με αφορμή το ChatGPT έχει ανοίξει μια μεγάλη συζήτηση σε όλη την κοινωνία σχετικά με την τεχνητή νοημοσύνη και τις εκπαιδευτικές, κοινωνικές, οικονομικές και άλλες επιπτώσεις της. Ποια είναι η γνώμη σας σχετικά; Πώς μπορούν οι κοινωνίες μας να θωρακιστούν έναντι των ενδεχομένων καταχρήσεων της τεχνητής νοημοσύνης; Το ίδιο θα μπορούσε κανείς να διαγνώσει κανείς σχετικά με τις τεχνολογίες της μοριακής βιολογίας (bioengineering) που αναπτύσσονται ραγδαία βοηθούσης και της Πληροφορικής, τώρα και της τεχνητής νοημοσύνης... Ποια η γνώμη σας;

Η γνώμη μου είναι να μην πέφτουμε στην παγίδα που έπεσαν οι Λουδίτες! Κάθε νέο επιστημονικό και τεχνολογικό άλμα, από τη φωτιά και τον τροχό μέχρι την πυρηνική ενέργεια και την πληροφορική, μπορεί να χρησιμοποιηθεί τόσο για καλό όσο και για κακό. Εναπόκειται στο κοινωνικό ον, τον άνθρωπο, να κάνει την ορθή χρήση. Ως κοινωνικά όντα δημιουργήσαμε κοινωνικοπολιτικά συστήματα για να μπορέσουμε τη δύναμη του είδους μας και των δημιουργημάτων να την κατευθύνουμε για το καλό του είδους μας και του ευρύτερου περιβάλλοντος στο οποίο ζούμε. Με ανησυχεί λοιπόν βαθιά που η τεχνολογική αυτή πρόοδος έρχεται παράλληλα με μια πτώση της κοινωνικής μας διάστασης, με μια ακραία αποξένωση και απόλυτη προαγωγή του ιδιωτικού έναντι του κοινού. Ξέρετε, αν από το "κοινωνικό ον" αφαιρέσει κανείς το "κοινωνικό", μένει κάτι που μπορεί και να μην θέλει να τον χαρακτηρίζει αποκλειστικά - οδηγούμαστε τελικά στην αποκτήνωση δια μέσου της ιδιώτευσης.

Η απάντηση λοιπόν στις προκλήσεις για τη ρύθμιση και την ορθή διαχείριση των επιστημονικών και τεχνολογικών επιτευγμάτων δεν βρίσκεται πουθενά αλλού παρά στη διαφύλαξη του κοινωνικού ιστού και στην ανάδειξη του πολιτικού, κοινωνικού ανθρώπου. Για να μιλήσω και πάλι για την επικαιρότητα, αναφερόμενος στην ευρωπαϊκή σκηνή, όχι μόνο στη μικρή μας χώρα, έχει σημάνει μάλλον η ώρα

να επανέλθουμε στο επίπεδο του Πολίτη, ανεβαίνοντας από εκείνο του ψηφοφόρου στο οποίο αυτο-υποβιβαστήκαμε τις τελευταίες δεκαετίες.

Όσο για το ChatGPT στο οποίο αναφέρεστε, τον ισχυρότερο και καλύτερο παπαγάλο της ιστορίας - για να παραφράσω τον μεγάλο Noam Chomsky - είναι ένα πραγματικά μοναδικό επίτευγμα που μπορεί να βελτιώσει σημαντικά τη ζωή αμέτρητων ανθρώπων. Εάν χρησιμοποιηθεί σωστά, όπως ανέφερα παραπάνω.

Η πανδημία εξάπλωσε σε μεγάλο βαθμό το μοντέλο της τηλεργασίας, μια δυνατότητα που ταιριάζει σε μεγάλο βαθμό στις περισσότερες ειδικότητες της Πληροφορικής; Πέρα από τα προφανή οφέλη, υπάρχουν ζητήματα που έχουν προκύψει από την διευρυμένη εφαρμογή της τηλεργασίας; Θεωρείται ότι η χώρα μας θα βγει ευνοημένη από αυτήν την εξέλιξη;

Ανήκω στην (προσωρινά!) μικρή εκείνη μειοψηφία που είναι πολύ επιφυλακτική με την τηλεργασία. Η γνώμη μου είναι πως εξυπηρετεί αποκλειστικά τα συμφέροντα των πολυεθνικών επιχειρήσεων ενώ σε καμία απολύτως των περιπτώσεων δεν εξυπηρετεί τους εργαζομένους (ούτε φυσικά τις μικρές και μεσαίες επιχειρήσεις).

Αυτό διότι καταστρέφει την κοινωνική και την ανθρώπινη διάσταση της εργασίας. Οι χώροι εργασίας παύουν να είναι χώροι καινοτομίας, έρευνας, αλλά και προσωπικής και κοινωνικής αυτοπραγμάτωσης. Αν επικρατήσει η τηλεργασία, δεν θα αργήσει η στιγμή που οι περισσότερες θέσεις εργασίας θα γίνουν αναλώσιμες και θα συγκεντρωθούν σε περιοχές χαμηλού κόστους ενώ οι αποξενωμένοι εργαζόμενοι, από το "κλουβί" τους καθένας δεν θα έχουν καμία δυνατότητα παρέμβασης.

Για να το πω όσο πιο απλά μπορώ, η δική μου θεώρηση της εργασίας είναι κάτι πολύ διαφορετικό (και, επιτρέψτε μου, ανώτερο!) από ένα task-list που πρέπει να ολοκληρωθεί και μερικούς διαλόγους στο slack. Θέλω να πιστεύω πως οι περισσότεροι συνάδελφοι μοιράζονται την ίδια αντίληψη και δεν θα θελήσουν να απολέσουν την ανθρώπινη και την κοινωνική διάσταση της εργασίας τους.

Ας μου επιτραπεί να σημειώσω πάντως πως όσο δεν πιστεύω στην (αποκλειστική) τηλεργασία, άλλο τόσο δεν πιστεύω και στο μοντέλο Δευτέρα - Παρασκευή, 9:00-17:00. Το μέλλον είναι σίγουρα υβριδικό!

Παρουσιάζοντας το SOLO Gateway το 2008

Εδώ και καιρό λειτουργούν στην Ελλάδα τα λεγόμενα Coding BootCamps όπου μπορεί κάποιος να διδαχθεί ανάπτυξη εφαρμογών σε κάποιο μοντέρνο περιβάλλον/πλατφόρμα. Οι απόφοιτοι τέτοιων κέντρων δύνανται να εργασθούν ως developers σε εταιρείες

Πληροφορικής. Μήπως λοιπόν οι σπουδές Πληροφορικής σε Πανεπιστημιακό επίπεδο είναι περιττές; Επίσης, ο απόφοιτος ενός Coding BootCamp ή κάποιος/α που ξέρει π.χ. άριστα C# ή Node.js® είναι κατά τη γνώμη σας Πληροφορικός; Θα αρκούσε αυτό για να ανταπεξέλθει στις σύγχρονες απαιτήσεις των εταιρειών;

Ευχαριστώ πολύ για την ερώτηση η οποία συνδέεται άρρηκτα με τη συζήτηση παραπάνω περί πληροφορικής παιδείας. Στην εποχή μας το να "ξέρει κάποιος python ή/και Node.js" δεν είναι κάτι που το περιμένει κανείς από απόφοιτο Πανεπιστημίου. Το περιμένει από κάποιον που έχει κάνει σοβαρές σπουδές δευτεροβάθμιας εκπαίδευσης, ίσως (για λίγα χρόνια) και ένα μεταλυκειακό πρόγραμμα όπως αυτό που προσφέρουν τα ΙΕΚ.

Στις εταιρείες πληροφορικής αντίστοιχα, αναλόγως του αντικειμένου τους, υπάρχουν θέσεις εργασίας για τέτοιους εργαζόμενους (ξαναλέω, επιπέδου λυκείου ή/και ΙΕΚ), πολλοί από τους οποίους μπορούν να αμείβονται εξαιρετικά, όπως και θέσεις εργασίας για επιστήμονες και μηχανικούς πληροφορικής (που δυστυχώς στη χώρα μας είναι λιγότερες απ' όσο θα περίμενε κανείς), καθότι οι περισσότερες εταιρείες πληροφορικής παραμένουν στην ένταση εργασίας και όχι στην ένταση γνώσης.

Συνοπτικά λοιπόν, όχι μόνο τα προγράμματα αυτά δεν έχουν τίποτα κακό αλλά ίσα ίσα καλύπτουν τα κενά της δευτεροβάθμιας και της μεταλυκειακής εκπαίδευσής μας. Το πρόβλημα δημιουργείται όταν άνθρωποι σε θέσεις ευθύνης, υπό το βάρος του άκρατου επαρχιωτισμού που ανέφερα παραπάνω, αγνοούν την ουσία του ζητήματος που περιέγραψα – την οποία σε ένα στοιχειωδώς οργανωμένο κράτος θα ρύθμιζε ένας φορέας όπως το Επιμελητήριο Πληροφορικής που έχει προτείνει η Ένωση Πληροφορικών Ελλάδας.

Στον όμιλο εταιριών Startech όπου είστε επικεφαλής, υλοποιήσατε πρόσφατα ένα πρόγραμμα πρακτικής φιλοσοφίας με εισηγητή τον Θεοφάνη Τάση. Ποια είναι εμπειρία σας από αυτή τη συνεργασία. Ποια η σχέση φιλοσοφίας και ιδίως της πρακτικής φιλοσοφίας με την Πληροφορική και τους Πληροφορικούς;

Επιτρέψτε μου να ξεκινήσω με δυο διευκρινίσεις: Αφενός μεν η Starttech Ventures δεν είναι ένας 'όμιλος επιχειρήσεων' με την παραδοσιακή έννοια του όρου αλλά ενας επαγγελματίας συνιδρυτής επιχειρήσεων ανεξαρτήτων επιχειρήσεων (venture builder ή μερικές φορές και venture studio), αφετέρου δε ο Θεοφάνης Τάσης δεν χαρακτήριζε τα masterclasses που κάναμε ως "πρακτική" φιλοσοφία αλλά γενικά ως "φιλοσοφία" (μιλήσαμε μάλιστα αρκετά για τη δυσκολία χρήσης του όρου φιλόσοφος / φιλοσοφία από το ευρύ κοινό αλλά και από τους ειδικούς).

Έχουμε ήδη ολοκληρώσει πέντε masterclasses και προγραμματίζουμε ήδη ακόμα δυο, αγγίζοντας θέματα από τη φιλοσοφία της κλασικής και της ελληνιστικής περιόδου μέχρι σύγχρονους φιλοσόφους όπως ο Κορνήλιος Καστοριάδης και ο Jürgen Habermas.

Δεν σας κρύβω πως από το ευρύ φάσμα των εκπαιδευτικών και αναπτυξιακών δραστηριοτή-

των που κάνουμε, τα masterclasses φιλοσοφίας ήταν με διαφορά τα πιο επιτυχημένα, τόσο σε όρους συμμετοχής όσο και ικανοποίησης. Το γεγονός αυτό με γεμίζει ελπίδα ότι υπάρχουν οι προϋποθέσεις ώστε η φύση μας ως "κοινωνικά όντα" όχι μόνο να διατηρηθεί αλλά και να αναπτυχθεί περαιτέρω!

Ποιον άνθρωπο της Επιστήμης των Υπολογιστών αληθινά θαυμάζετε; *(Είτε ιστορικό πρόσωπο, είτε σύγχρονο)*

Αν κάποιος είναι δίκαιος, όσο κι αν θαυμάζει γίγαντες της επιστήμης μας όπως ο Alan Turing και ο Kurt Gödel, κανείς δεν μπορεί να συγκριθεί με τον John von Neumann! Παραπάνω με ρωτήσατε για μεγάλες και επιδραστικές ιδέες. Ο von Neumann είχε περισσότερες τέτοιες ιδέες από κάθε άλλον και μπόρεσε πολλές από αυτές να τις κάνει πράξη! Ο άνθρωπος αυτός καθόρισε την ζωή των ανθρώπων στον 20ο και στον 21ο αιώνα, πιθανότατα και πολύ αργότερα.

Από τους σύγχρονους θαυμάζω ιδιαίτερα τον Brian Kernighan και τον "δικό μας" Χρίστο Παπαδημητρίου.

Θα θέλατε να μας προτείνετε 3 βιβλία που αξίζει να διαβάσουμε;

Δεν νομίζω πως υπάρχει πιο δύσκολη ερώτηση από αυτή! Στη διάρκεια της πανδημίας γνώρισα καλύτερα τον τεράστιο Νίκο Καζαντζάκη και δυσκολεύομαι να βρω λέξεις που να περιγράφουν τι ένιωσα διαβάζοντάς τον.

Απαντώ λοιπόν ως εξής: "Βίος και Πολιτεία του Αλέξη Ζορμπά", "Καπετάν Μιχάλης", "Ασκητική". Αν σε όλη μου τη ζωή έπρεπε να διαβάσω μόνο τρία βιβλία, θα ήθελα να ήταν αυτά.

Ποια συμβουλή θα δίνατε σήμερα σ' έναν φοιτητή ή νέο απόφοιτο της Πληροφορικής;

Η βασικότερη συμβουλή είναι να ξεφύγει από το περιτύλιγμα. Στην εποχή μας έχουν προοδεύσει τόσο πολύ τα συστήματα ανάπτυξης εφαρμογών που είναι σχετικά εύκολο για έναν φοιτητή να κάνει εντυπωσιακές εφαρμογές. Να τα μάθει αυτά αλλά να πάει και πιο βαθιά! Δομές δεδομένων, λειτουργικά συστήματα, οργάνωση & αρχιτεκτονική υπολογιστών, γλώσσες & μεταφραστές, αλγόριθμοι. Να μη διστάσει να μπει στην ουσία της επιστήμης μας! Αυτό θα του εξασφαλίσει ένα άριστο επαγγελματικό μέλλον, θα του δώσει ένα μοναδικό αίσθημα ικανοποίησης και, το σημαντικότερο, θα τον οδηγήσει να ανακαλύψει τη βαθιά ανθρώπινη διάσταση της επιστήμης μας.

Παρουσιάζοντας την Starttech Ventures το 2016

Ανοιχτά δεδομένα και το μαύρο κουτί της Τεχνητής Νοημοσύνης

Η Τεχνητή Νοημοσύνη (AI) συγκεντρώνει το τελευταίο καιρό τα βλέμματα με νέα εργαλεία όπως το ChatGPT και το DALL-E 2 , **αλλά είναι ήδη εδώ και έχει σημαντικές επιπτώσεις στη ζωή μας**. Όλο και περισσότερο βλέπουμε την επιβολή του νόμου, την ιατρική περίθαλψη, τα σχολεία και τους χώρους εργασίας να στρέφονται στο μαύρο κουτί της τεχνητής νοημοσύνης για να λάβουν αποφάσεις που αλλάζουν τη ζωή – **μια τάση που θα πρέπει να αμφισβητούμε σε κάθε βήμα.**

Τα τεράστια και συχνά μυστικά σύνολα δεδομένων πίσω από αυτήν την τεχνολογία, που χρησιμοποιούνται για την εκπαίδευση της τεχνητής νοημοσύνης με μηχανική μάθηση, συνοδεύονται και από επιπτώσεις. **Τα δεδομένα που συλλέγονται μέσω της επιτήρησης και της εκμετάλλευσης θα αντικατοπτρίζουν συστημικές προκαταλήψεις και θα «μαθεύονται» στη διαδικασία**. Στη χειρότερη μορφή τους, τα τσιτάτα της τεχνητής νοημοσύνης και της μηχανικής μάθησης χρησιμοποιούνται για να κάνουν «tech wash» σε αυτή την προκατάληψη, επιτρέποντας στους ισχυρούς να υποστηρίξουν καταπιεστικές πρακτικές πίσω από την υποτιθέμενη αντικειμενικότητα του κώδικα.

Ήρθε η ώρα να ανοίξουμε αυτά τα μαύρα κουτιά. Η υιοθέτηση συλλογικά διατηρούμενων συνόλων Ανοιχτών Δεδομένων στην ανάπτυξη της τεχνητής νοημοσύνης δεν θα ήταν μόνο όφελος για τη διαφάνεια και την υπευθυνότητα για αυτά τα εργαλεία, **αλλά θα επιτρέψει στα επίδοξα υποκείμενα να δημιουργήσουν τη δική τους καινοτόμο και ενδυναμωτική εργασία και έρευνα. Πρέπει να διεκδικήσουμε ξανά αυτά τα δεδομένα και να αξιοποιήσουμε τη δύναμη μιας δημοκρατικής και ανοιχτής επιστήμης για να οικοδομήσουμε καλύτερα εργαλεία και έναν καλύτερο κόσμο.**

Σκουπίδια μέσα, Ευαγγέλιο έξω ?

Η Μηχανική Μάθηση είναι ένα ισχυρό εργαλείο και υπάρχουν πολλές εντυπωσια**κές περιπτώσεις χρήσης: όπως η αναζήτηση σημείων ζωής στον Άρη ή η κατασκευή συνθετικών αντισωμάτων . Αλλά στον πυρήνα τους αυτοί οι αλγόριθμοι είναι τόσο «έξυπνοι» όσο και τα δεδομένα που τροφοδοτούν.** Ξέρετε το ρητό: «σκουπίδια μέσα, σκουπίδια έξω». Η Μηχανική Μάθηση βασίζεται τελικά σε δεδομένα εκπαίδευσης για να μάθει πώς να κάνει καλές εικασίες — η λογική πίσω από την οποία είναι συνήθως άγνωστη ακόμη και στους προγραμματιστές. **Αλλά ακόμη και οι καλύτερες εικασίες δεν πρέπει να εκληφθούν ως ευαγγέλιο.**

Τα πράγματα γίνονται ζοφερά όταν αυτή η συγκαλυμμένη λογική χρησιμοποιείται για τη λήψη αποφάσεων που αλλάζουν τη ζωή. Εξετάστε τον αντίκτυπο των εργαλείων πρόβλεψης αστυνόμευσης , τα οποία είναι χτισμένα σε μια βάση διαβόητων ανακριβών και μεροληπτικών δεδομένων εγκλήματος . Αυτή η αναζήτηση για "μελλοντικά εγκλήματα" με δυνατότητα τεχνητής νοημοσύνης είναι ένα τέλειο παράδειγμα του πώς αυτό το νέο εργαλείο ξεπλένει μεροληπτικά δεδομένα της αστυνομίας σε προκατειλημμένη αστυνόμευση – με αλγόριθμους να δίνουν έμφαση σε ήδη υπερβολικά αστυνομευμένες γειτονιές. Αυτή η αυτοεκπληρούμενη προφητεία εκτυλίσσεται ακόμη και για να προβλέψει την εγκληματικότητα από το σχήμα του προσώπου σας. Στη συνέχεια, κατά τον καθορισμό της εγγύησης σε μετρητά, ένας άλλος αλγόριθμος μπορεί να ορίσει την τιμή χρησιμοποιώντας δεδομένα γεμάτα με τις ίδιες ρατσιστικές και ταξικές προκαταλήψεις.

Ευτυχώς, οι νόμοι περί διαφάνειας επιτρέπουν στους ερευνητές να εντοπίσουν και να επιστήσουν την προσοχή σε αυτά τα ζητήματα. Στοιχεία εγκλήματος, έγγραφα και όλα αυτά, διατίθενται συχνά στο κοινό. Αυτή η ίδια διαφάνεια δεν αναμένεται από ιδιωτικούς φορείς όπως ο εργοδότης σας , ο σπιτονοικοκύρης σας ή το σχολείο σας.

Η απάντηση δεν είναι απλώς να δημοσιοποιηθούν όλα αυτά τα δεδομένα. Κάποια μοντέλα τεχνητής νοημοσύνης εκπαιδεύονται σε νόμιμα ευαίσθητες πληροφορίες, ακόμη και αν είναι διαθέσιμες στο κοινό . Είναι τοξικά περιουσιακά στοιχεία που προέρχονται από ένα μείγμα επιτήρησης και υποχρεωτικών γνωστοποιήσεων δεδομένων. Η προετοιμασία αυτών των δεδομένων είναι από

μόνη της αμφίβολη, καθώς συχνά βασίζεται σε στρατούς εργαζομένων που υφίστανται υψηλή εκμετάλλευση, χωρίς τρόπους για την επισήμανση προβλημάτων με τα δεδομένα ή την επεξεργασία τους. Και παρά τους πολλούς ισχυρισμούς "μυστικής συνταγής", η ανωνυμοποίηση αυτών των μεγάλων συνόλων δεδομένων είναι πολύ δύσκολη και ίσως ακόμη και αδύνατη, και οι επιπτώσεις μιας παραβίασης θα επηρέαζαν δυσανάλογα τους ανθρώπους που παρακολουθήθηκαν και εκμεταλλεύτηκαν για την παραγωγή της.

Αντίθετα, η υιοθέτηση ανοιχτών συνόλων δεδομένων που διατηρούνται από κοινού θα ενδυναμώσει τους επιστήμονες δεδομένων, οι οποίοι είναι ήδη ειδικοί σε θέματα διαφάνειας και απορρήτου που σχετίζονται με δεδομένα, να τα διατηρούν πιο ηθικά. Με τη συγκέντρωση πόρων με αυτόν τον τρόπο, η συναινετική και διαφανής συλλογή δεδομένων θα βοηθούσε στην αντιμετώπιση αυτών των προκαταλήψεων, αλλά θα ξεκλειδώσει το δημιουργικό δυναμικό της ανοιχτής επιστήμης για το μέλλον της τεχνητής νοημοσύνης.

Ένα ανοιχτό και ενδυναμωτικό μέλλον της τεχνητής νοημοσύνης

Όπως βλέπουμε αλλού στην Ανοικτή Πρόσβαση , αυτή η κατάργηση των φραγμών και των paywalls βοηθά τα άτομα με λιγότε-ρους πόρους να έχουν πρόσβαση και να αποκτήσουν τεχνογνωσία. Το αποτέλεσμα θα μπορούσε να είναι ένα οικοσύστημα όπου η τεχνητή νοημοσύνη δεν εξυπηρετεί απλώς τους έχοντες έναντι των μη εχόντων, αλλά στο οποίο όλοι μπορούν να επωφεληθούν από την ανάπτυξη αυτών των εργαλείων.

Το λογισμικό ανοιχτού κώδικα έχει αποδείξει εδώ και καιρό τη δύναμη της συγκέντρωσης πόρων και του συλλογικού πειραματισμού. Το ίδιο ισχύει και για τα Ανοιχτά Δεδομένα—η καθιστώντας τα δεδομένα ανοιχτά προσβάσιμα μπορεί να εντοπίσει ελλείμματα και να επιτρέψει στους ανθρώπους να χτίσουν ο ένας τη δουλειά του άλλου πιο δημοκρατικά. Η σκόπιμη μεροληψία δεδομένων (ή «δηλητηρίαση δεδομένων») είναι δυνατή και αυτή η ανήθικη συμπεριφορά συμβαίνει ήδη σε λιγότε-ρο διαφανή συστήματα και είναι πιο δύσκολο να εντοπισθεί. Αν και μια κίνηση προς τη χρήση Ανοιχτών Δεδομένων στην ανάπτυξη τεχνητής νοημοσύνης θα βοηθούσε στον μετριασμό των μεροληψιών και των ψευδών ισχυρισμών, δεν είναι πανάκεια. Ακόμη και επιβλαβή και μυστικά εργαλεία μπορούν να κατασκευαστούν με καλά δεδομένα.

Ωστόσο, ένα ανοιχτό σύστημα για την ανάπτυξη τεχνητής νοημοσύνης, από δεδομένα, κώδικα, έως δημοσίευση, μπορεί να αποφέρει πολλά ανθρωπιστικά οφέλη, όπως η χρήση της τεχνητής νοημοσύνης στην ιατρική έρευνα που σώζει ζωές . Η ικανότητα επανάχρησης και γρήγορης συνεργασίας στην ιατρική έρευνα μπορεί να επιταχύνει την ερευνητική διαδικασία και να αποκαλύψει χαμένες ανακαλύψεις στα δεδομένα. Το αποτέλεσμα? Εργαλεία για σωτήρια ιατρική διάγνωση και θεραπείες για όλους τους ανθρώπους, μετριάζοντας τις φυλετικές, διακρί-

σεις φύλου και άλλες προκαταλήψεις στην ιατρική έρευνα .

Τα Ανοιχτά Δεδομένα κάνουν τα δεδομένα να λειτουργούν για τους ανθρώπους. Ενώ η τεχνογνωσία και οι πόροι που απαιτούνται για τη μηχανική μάθηση παραμένουν εμπόδιο για πολλούς, τα έργα πληθοπορισμού όπως το Open Oversight ενδυναμώνουν ήδη τις κοινότητες κάνοντας πληροφορίες σχετικά με την επιβολή του νόμου ορατή και διαφάνεια. Το να μπορούν να συλλέγουν, να χρησιμοποιούν και να αναμιγνύουν δεδομένα για να φτιάξουν τα δικά τους εργαλεία, φέρνει την έρευνα AI από τους κλειδωμένα συρτάρια στους δρόμους και καταστρέφει τις καταπιεστικές ανισορροπίες ισχύος.

Τα Ανοιχτά Δεδομένα δεν αφορούν μόνο την πρόσβαση στα δεδομένα. Πρόκειται για τον εναγκαλισμό των προοπτικών και της δημιουργικότητας όλων των ανθρώπων για να τεθούν οι βάσεις για μια πιο δίκαιη και δίκαιη κοινωνία. Πρόκειται για την κατάργηση της συλλογής δεδομένων εκμετάλλευσης και τη διασφάλιση ότι όλοι επωφελούνται από το μέλλον της τεχνητής νοημοσύνης.

Πηγή άρθρου:

https://openstandards.ellak.gr
& https://www.eff.org/

Photo: David Cassolato

✔ Peopleware

Επιμέλεια στήλης: **Νεκτάριος Μουμουτζής**

Photo: Mikhail Nilov

Η στήλη Peopleware(*) ως στόχο έχει να αναδείξει το ανθρώπινο πρόσωπο της Πληροφορικής. Ή, αν το προτιμάτε, τις ψυχοθεραπευτικές της δυνατότητες όταν καλλιεργεί και ενισχύει την δημιουργικότητα. Η στήλη θα προσπαθήσει να αναδείξει αυτή τη διάσταση της Πληροφορικής μέσα από ανθρώπινες ιστορίες που λειτουργούν ως παραβολές αφήνοντας τον αναγνώστη να βγάλει τα δικά του συμπεράσματα. Κάποιες από τις ιστορίες αυτές είναι πραγματικά περιστατικά με τροποποίηση ονομάτων προσώπων και άλλων λεπτομερειών για να μην αποκαλύπτονται ευαίσθητα προσωπικά δεδομένα. Άλλες ιστορίες θα βασίζονται στη μυθοπλασία…

Αν έχετε κι εσείς κάποια ιστορία που αναδεικνύει το ανθρώπινο πρόσωπο της Πληροφορικής, μπορείτε να επικοινωνήσετε με τον επιμελητή της στήλης στη διεύθυνση nmoumoutzis@tuc.gr για να τη μοιραστείτε με τους αναγνώστες του Πληροφορικού.

☆ Τέσσερις συν μία ώρες

Γράφει ο **Φώτης Αλεξάκος**

Όταν κατά το σχολικό έτος 2018-2019 κλήθηκα να διδάξω τέσσερις (4) ώρες Μαθηματικά συν μια (1) ώρα Πληροφορική στην Α' τάξη του απέναντι Γυμνασίου, τίποτε δεν με προϊδέαζε για όσα θα επακολουθούσαν. Ίσα ίσα που ικανοποιήθηκα, καθώς έτσι θα συμπλήρωνα το προβλεπόμενο εβδομαδιαίο διδακτικό ωράριο και θα σταματούσαν τα κακεντρεχή μουρμουρητά του τύπου «δουλεύει λιγότερο, αλλά πληρώνεται το ίδιο». Με το που υπέγραψα στο βιβλίο πράξεων, η διευθύντρια του σχολείου μου ανακοίνωσε πως θα αναλάμβανα να διδάξω Άλγεβρα και Γεωμετρία, καθώς και να κάνω μια εισαγωγή στην επιστήμη που σπούδασα, την Πληροφορική, στους μαθητές του Τμήματος Α6. Οι συνάδελφοι, γελαστοί με χτυπούσαν φιλικά στην πλάτη εκτός απ' τη φιλόλογο του εν λόγω Τμήματος, η οποία μου είπε πως τώρα θα … έχει κάποιον να μοιράζεται τα βάσανά της.

Μια ματιά στο μαθητολόγιο έλυσε άμεσα κάθε απορία μου: Είκοσι (20) μαθητές με όλα κι όλα 3-4 διαφορετικά επίθετα, ενώ σχεδόν οι μισοί είχαν στη θέση του πατρώνυμου ένα «ΑΠ»-«Αγνώστου Πατρός». Το Α6 λοιπόν ήταν το μοναδικό τμήμα του σχολείου που περιλάμβανε αποκλειστικά μαθητές Ρομά! Σε κάθε τμήμα υπήρχαν και αρκετοί Ρομά, αλλά στο «δικό μου» υπήρχαν μόνο τέτοιοι.

Photo: Dids

Δεν χρειάζεται να εξηγήσω πως πρόκειται για παιδιά με εντελώς διαφορετική κουλτούρα, συνήθειες και παραδόσεις από εμάς τους «μπαλαμούς». Ενδεικτικά να αναφέρω ότι οι ηλικίες τους –στην Α' Γυμνασίου- κάλυπταν ένα φάσμα απ' το 13 ως το 16, δεν καταλάβαιναν διαφορά ανάμεσα στο «είμαστε μέσα στην τάξη»

και «παίζουμε έξω στο διάλειμμα», ενώ σχεδόν όλα ήταν αναλφάβητα. Μόλις 4 ή 5 γνώριζαν ανάγνωση και μπορούσαν να γράψουν το όνομά τους.

Κι εγώ θα τους δίδασκα Άλγεβρα; Ή θα τους μιλούσα για τον «χελωνόκοσμο» και «το υλικό του Η/Υ»; Προς το παρόν φάνταζε αδύνατο να τους κρατήσεις 45' μέσα στην τάξη. Συνεχώς σηκωνόντουσαν απ' τις θέσεις τους, είτε για να στήσουν πηγαδάκι με φίλους/φίλες ή για να «πάνε μια βόλτα έξω για αέρα», μετά μέσα για κουβέντα και ούτω καθεξής. Τέλος πάντων, σκοπός του παρόντος κειμένου δεν είναι να χρησιμοποιηθεί ως εγχειρίδιο επιβίωσης, αφού ούτε κι ο χώρος επιτρέπει να αναφερθούν αναλυτικά τα «κόλπα» που επιστράτευσα μπας και τους μάθω κάτι. Έστω ως τον Γενάρη που οι περισσότεροι εγκατέλειπαν το σχολείο (για λόγους που δεν είναι της παρούσης). Μη φανταστεί κανείς όμως πως οι 6-7 που μένανε ήταν «εύκολο» κοινό για το υπόλοιπο της χρονιάς.

Το πρώτο που «δούλεψε» ήταν να εκμεταλλευτώ την ανάγκη τους για άμιλλα σε συνδυασμό με επίδειξη. Στην αίθουσα λοιπόν υπήρχε πάντα ένας πίνακας με την τρέχουσα «βαθμολογία» καθενός σε φθίνουσα ταξινόμηση. Ακόμη κι όσοι δεν ήξεραν να διαβάζουν, χαιρόντουσαν πολύ όταν τους έδειχνα πως το όνομά τους βρίσκεται σε σημείο πάνω απ' τη μέση της λίστας. Η βαθμολογία ήταν με άριστα το 20 και σε κάθε μάθημα έπαιρναν ή έχαναν πόντους ανάλογα με τη συμπεριφορά και την προσπάθειά τους να απαντούν σε ερωτήσεις σχετικές π.χ. με προσθαφαίρεση. Ο ίδιος ο υπολογισμός των πόντων της καθεμιάς/καθενός απαιτούσε συχνές προσθέσεις ή αφαιρέσεις. Τουλάχιστον θα μάθαιναν να διαβάζουν αριθμούς και να κάνουν στοιχειώδεις πράξεις.

Photo: Dids

Όμως η μεγάλη «επιτυχία» ήρθε απ' την Πληροφορική και μάλιστα απ' το πολύ μακρινό 1940 και το "Nimatron". Κάποιοι θεωρούν το εν λόγω μηχάνημα (μια κατασκευή που παίζει το παιχνίδι Nim) ως το πρώτο ηλεκτρονικό παιχνίδι στην ιστορία. Πρόγονος θα λέγαμε των "Pong", "Spacewar!" κλπ. Η ιδέα ήταν να χρησιμοποιήσω μια εκδοχή του παιχνιδιού που θα παιζόταν με 21 σπίρτα ως εξής: Έχουμε δυο παίκτες που μπορούν ο καθένας εναλλάξ να παίρνουν από ένα (1) έως τρία (3) σπίρτα από έναν σωρό των 21. Νικητής είναι αυτός που θα αφήσει άδειο το

τραπέζι. Δεν χρειάζεται μεγάλη προσπάθεια για να δεις πως αυτός που παίζει πρώτος κερδίζει νομοτελειακά, αρκεί να αφήνει πάντα πολλαπλάσιο του 4 σπίρτα μπροστά στον αντίπαλο. Π.χ Ένα παιχνίδι θα μπορούσε να περιγραφεί απ' την ακολουθία: αφαιρώ 1 (μένουν 20 στον σωρό), -3(17), -1(16), -2(14), -2(12), -3(9), -1(8), -3(5), -1(4) και όσα και να αφαιρέσει ο αντίπαλος (μέχρι 3 μπορεί άλλωστε), θα μείνουν 1-3 σπίρτα τα οποία σηκώνει ο νικητής 1ος παίκτης (με κόκκινο).

Τελικός μου σκοπός ήταν να καταλάβουν οι μαθητές την παραπάνω νομοτέλεια. Το βέβαιον του ποιος θα είναι ο νικητής δηλαδή. Και ίσως κάποια στιγμή να μπορούν να απαντούν σε ερωτήματα όπως: Τι θα ίσχυε αν μπορούσε ο κάθε παίκτης να αφαιρέσει ως και τέσσερα (4) σπίρτα; Αν ξεκινούσαμε από 24 σπίρτα; Και άλλα παρόμοια. Πλην όμως για να φτάσουν ως εκεί οι ημιαναλφάβητοι μαθητές μου, θα έπρεπε να παίξουν πολλές «παρτίδες» μεταξύ τους και να κάνουν πολλές αφαιρέσεις. Επίσης, κάποια στιγμή χρειάστηκε να εισάγουμε τις έννοιες του πολλαπλάσιου και του διαιρέτη! Αυτό κι αν ήθελε χρόνο και κόπο. Μόνο που πλέον απολάμβανα την αμέριστη προσοχή τους.

Βέβαια, το ενδιαφέρον των μαθητών καλώς ή κακώς είχε να κάνει και με … «εξωσχολικούς» λόγους. Ποιος δεν ήθελε να μάθει πώς να κερδίζει συνεχώς κάθε παρτίδα ενός παιχνιδιού; Φαντάζομαι πως πίσω στον τσιγγάνικο καταυλισμό, τα απογεύματα, κάμποσοι έφηβοι θα κλαίγανε το χαρτζηλίκι τους που θά 'χε καταλήξει στις τσέπες των μαθητών μου. Κι όλα αυτά χάρη στο «νέο κινέζικο παιχνίδι». Και την επομένη στο μάθημα θα μπορούσα να απαντήσω: Ορίστε κυρίες και κύριοι γιατί χρειάζεται να ξέρουμε από διαίρεση και αλγορίθμους• κι ας υπάρχει το κομπιουτεράκι στο κινητό μας τηλέφωνο.

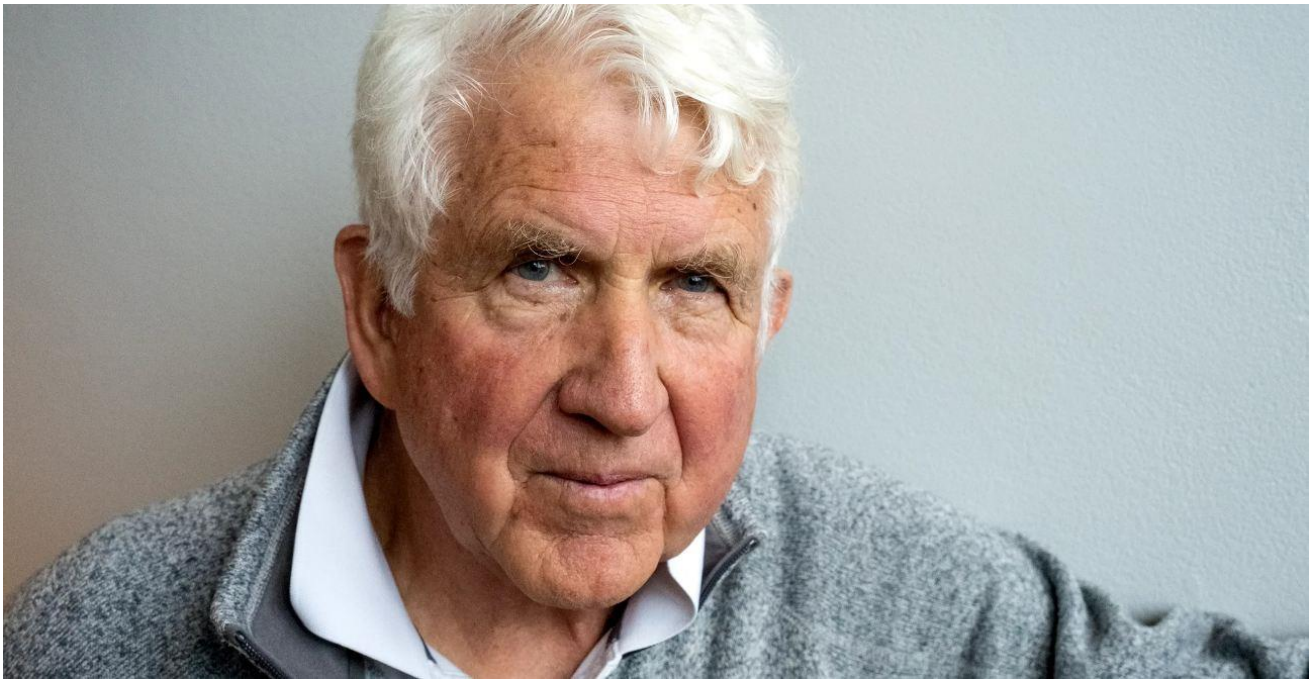
Το ρεζουμέ είναι πως αν τον Γενάρη ήταν να φύγουν οι μισοί και πλέον μαθητές, αυτήν τη φορά έφυγαν κάπως λιγότεροι (π.χ. κορίτσια κι αγόρια που … παντρεύτηκαν) ενώ και στο μάθημα φτάσαμε δειλά-δειλά να μιλάμε για διαιρέσεις(!), κλάσματα, αναλογίες κλπ. Επίσης, στην Πληροφορική είδαμε πως υπάρχουν κι άλλα παιχνίδια που η εξέλιξή τους είναι προκαθορισμένη (π.χ. τρίλιζα) και σ'αυτά είναι που διαπρέπει κυρίως ο υπολογιστής, χάρη στους περιβόητους αλγόριθμους. Σίγουρα δεν μπορούμε να μιλήσουμε για κάτι εντυπωσιακό, αλλά θα τολμούσαμε να πούμε πως οι Ρομά μαθητές, αυτήν τη φορά όντως κάτι αποκόμισαν απ' το σχολείο. Έστω και μόνο την περιέργεια σχετικά με το «τι να υπάρχει παραπέρα».

Κι αν μη τι άλλο, κάποιος εκπαιδευτικός κατάφερε να … επιβιώσει για λίγον καιρό στο Α6.

---

*(*) Ο όρος **peopleware**, σύμφωνα με το σχετικό λήμμα της αγγλικής Wikipedia, αναφέρεται σε μία από τις τρεις βασικές πτυχές της τεχνολογίας των υπολογιστών, ενώ οι άλλες δύο είναι το υλικό (hardware) και το λογισμικό (software). Ο όρος peopleware μπορεί να αναφέρεται σε οτιδήποτε έχει να κάνει με το ρόλο των ανθρώπων στην ανάπτυξη ή τη χρήση συστημάτων λογισμικού και υλικού υπολογιστών.*

Ο Bob Metcalfe, ο πρωτοπόρος του Ethernet, κερδίζει το βραβείο Turing



[Πηγή άρθρου: **Quanta Magazine**]

Ο Bob Metcalfe πίστευε πάντα στη δύναμη της δικτύωσης. Τις δεκαετίες του 1980 και του 1990 βοήθησε στη διάδοση της ιδέας ότι η αξία ενός δικτύου αυξάνεται γρήγορα με τον αριθμό των χρηστών, μια αρχή γνωστή πλέον ως νόμος του Metcalfe. Σήμερα, με το διαδίκτυο να είναι πανταχού παρόν, σκέφτεται σε μεγαλύτερη κλίμακα. «Το πιο σημαντικό νέο γεγονός για την ανθρώπινη κατάσταση είναι ότι τώρα ξαφνικά συνδεόμαστε», είπε.

Σήμερα ο Metcalfe αναδείχθηκε νικητής του βραβείου Turing, ενός ετήσιου βραβείου που θεωρείται η υψηλότερη διάκριση στην επιστήμη των υπολογιστών, για τη συμβολή του στην έναρξη της υπερσυνδεδεμένης εποχής μας. Πριν από πενήντα χρόνια, ο Metcalfe βοήθησε στην εφεύρεση του Ethernet, της τοπικής τεχνολογίας δικτύωσης που συνδέει προσωπικούς υπολογιστές σε όλο τον κόσμο με το παγκόσμιο διαδίκτυο. Έπαιξε επίσης κεντρικό ρόλο στην τυποποίηση και την εμπορευματοποίηση της εφεύρεσής του.

«Ο Μπομπ είναι ένας από τους ανθρώπους που έζησαν και στις δύο πλευρές. Μπορούσε να δει τη μεγάλη εικόνα», είπε ο Steve Crocker, ένας πρωτοπόρος δικτύωσης υπολογιστών που συνεργάστηκε με τον Metcalfe σε έναν πρόδρομο του Διαδικτύου γνωστό ως Arpanet.

Ως μεταπτυχιακός φοιτητής, ο Metcalfe κατασκεύασε αυτή τη διεπαφή για να συνδέσει τον κεντρικό υπολογιστή του MIT στο Arpanet, έναν πρόδρομο του σύγχρονου Διαδικτύου.

Η καριέρα του Metcalfe έχει αναπτυχθεί παράλληλα με την ικανότητα δικτύωσης μας. Γεννήθηκε στο Μπρούκλιν το 1946 και σπούδασε ηλεκτρολόγος μηχανικός στο Τεχνολογικό Ινστιτούτο της Μασαχουσέτης. Όταν μετακόμισε στην πόλη στο Πανεπιστήμιο του Χάρβαρντ για μεταπτυχιακές σπουδές, το Υπουργείο Άμυνας των ΗΠΑ απλώς αύξανε την επένδυσή του στην Arpanet. Ο Μέτκαλφ πρότεινε την κατασκευή μιας διεπαφής που συνδέει το δίκτυο με τον κεντρικό υπολογιστή του Χάρβαρντ, αλλά το πανεπιστήμιο τον απέρριψε. Γύρισε και έκανε την ίδια πρόταση στο MIT, όπου προσλήφθηκε ως ερευνητής ενώ ήταν ακόμα μεταπτυχιακός φοιτητής στο Χάρβαρντ. Όταν παρουσίασε μια διατριβή που περιγράφει το έργο στην επιτροπή της διατριβής του το 1972, απέτυχε να υπερασπιστεί - το θέμα δεν ήταν αρκετά θεωρητικό, είπαν.

Μέχρι τότε ο Metcalfe είχε ήδη δεχτεί μια δουλειά στο Ερευνητικό Κέντρο Palo Alto της Xerox Corporation, ή PARC, στην Καλιφόρνια. Ο διευθυντής του εργαστηρίου, Μπομπ Τέιλορ, του είπε να πάει ούτως ή άλλως και να τελειώσει τη διατριβή του από το Πάλο Άλτο. Μόλις έφτασε εκεί, ο Metcalfe άρχισε να χτίζει μια άλλη διεπαφή Arpanet για έναν νέο υπολογιστή PARC, ενώ έψαχνε για ένα θεωρητικό θέμα για να ικανοποιήσει το Χάρβαρντ.

Εκείνη την εποχή, η δικτύωση υπολογιστών ήταν τόσο μια θεωρητική πρόκληση όσο και μια μηχανική. Το θεμελιώδες πρόβλημα ήταν ο τρόπος κοινής χρήσης της πρόσβασης σε ένα δίκτυο μεταξύ πολλών χρηστών. Τα τηλεφωνικά δίκτυα αντιμετώπισαν αυτό το πρόβλημα με τον απλούστερο δυνατό τρόπο: μια σύνδεση μεταξύ δύο μερών κλείδωσε το κανάλι επικοινωνίας για όλη τη διάρκεια μιας κλήσης, καθιστώντας αυτό το κανάλι απρόσιτο σε άλλους χρήστες, ακόμη και αν δεν χρησιμοποιούνταν σε πλήρη χωρητικότητα. Αυτή η αναποτελεσματικότητα δεν είναι μεγάλο πρόβλημα για τις τηλεφωνικές συνομιλίες, οι οποίες σπάνια μένουν σε σιωπή για πολύ. Αλλά οι υπολογιστές επικοινωνούν σε σύντομες εκρήξεις, οι οποίες συχνά χωρίζονται από μεγάλες εκτάσεις νεκρού χρόνου. [Συνέχεια >>]

Βελτιστοποιήσεις κώδικα (code optimizations): Είναι πάντοτε επιθυμητές;

Γράφει ο **Φώτης Αλεξάκος**

Το παρόν άρθρο έχει γραφτεί πριν περίπου έναν χρόνο. Στο ενδιάμεσο μπορεί κάποια απ' τα προβλήματα που αναδεικνύονται εδώ να έχουν πια αντιμετωπιστεί. Πιστεύουμε όμως πως το κείμενο διατηρεί το όφελός του, αν μη τι άλλο, για ιστορικούς λόγους.

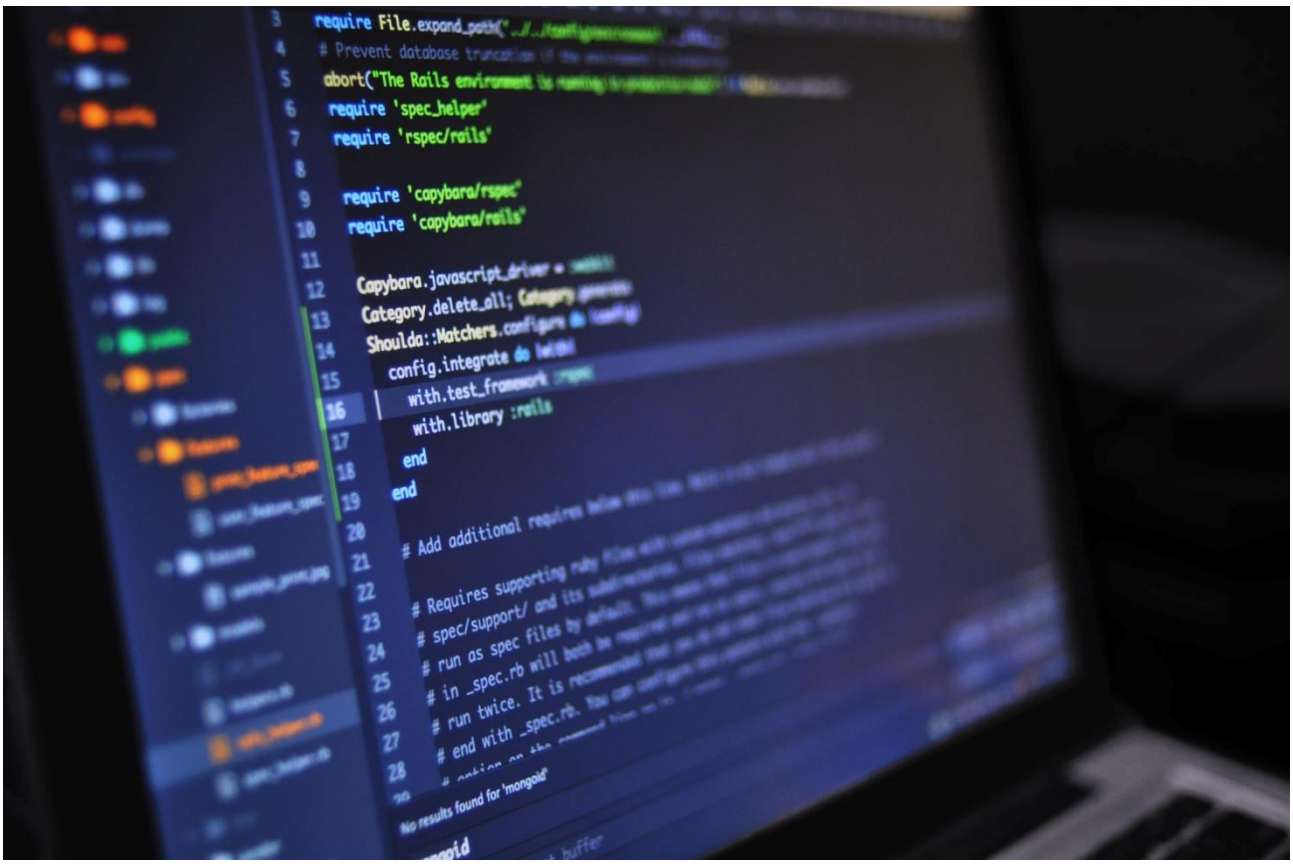


Photo: Luis Gomes

Όλοι γνωρίζουν πως εδώ και πολλά χρόνια, ακόμα κι ένα υποτυπώδες προγραμματιστικό περιβάλλον (IDE) ή -αν θέλετε- μια «σουΐτα» για εφαρμογές, περιλαμβάνει οπωσδήποτε έναν μεταγλωττιστή (compiler) με σοβαρές δυνατότητες βελτιστοποίησης του κώδικα. Πολλοί προγραμματιστές, εκόντες άκοντες, καλώς ή κακώς, παραδίδουν το εκτελέσιμο του κώδικά τους έχοντας επαναπαυθεί στις βελτιστοποιήσεις του «θεού» μεταγλωττιστή.

Είναι όμως αυτό πάντα θεμιτό; Οι Πληροφορικοί (και όχι μόνο) έχουμε μάθει «εξ απαλών ονύχων» πως ό,τι γίνεται αυτόματα, συνήθως δεν είναι βέλτιστο. Εν πάσει περιπτώσει, δεν θα βασιζόμασταν π.χ. σ' έναν αλγόριθμο για να διορθώσει τα συντακτικά λάθη ενός μαθητικού δοκιμίου (μιας δηλ. Έκθεσης). Στο παρόν πόνημα λοιπόν, θα προσπαθήσουμε να δούμε πότε και πώς οι τόσο πολύτιμες (υπό κανονικές συνθήκες) βελτιστοποιήσεις που κάνει ο μεταγλωττιστής στον κώδικά μας μπορεί να δημιουργήσουν προβλήματα στο έργο μας και ενδεχομένως πώς μπορούμε να αποφύγουμε τέτοιες καταστάσεις.

Οφείλουμε εδώ να σημειώσουμε πως το άρθρο αυτό είναι εν πολλοίς βασισμένο σε παρουσίαση του κ. ***Robert C. Seacord*** από το Πανεπιστήμιο Carnegie Mellon (CMU) των ΗΠΑ(2010 - https://bit.ly/3taSOpV). Εκεί, μαθαίνουμε εξ αρχής πως ο σωστότερος όρος δεν είναι *compiler optimization induced* ***bugs*** (σφάλματα προξενούμενα από τις βελτιστοποιήσεις του μεταγλωττιστή), αλλά: *Απροσδιόριστη συμπεριφορά* (undefined behaviour). Δηλαδή θα μιλήσουμε για προβλήματα οφειλόμενα όχι μόνο και τόσο σε «πρωτοβουλίες» του μεταγλωττιστή, αλλά και σε λάθος εκτιμήσεις-παραδοχές του ίδιου του προγραμματιστή.

Πριν ξεκινήσουμε, να σημειώσουμε ότι θα χρησιμοποιηθεί η γλώσσα C για τα παραδείγματα και κυρίως όπως αυτή υποστηρίζεται από τον πασίγνωστο μεταγλωττιστή gcc της FSF και λιγότερο από τον Microsoft Visual C compiler (MSVC). Αυτό διότι οι μεταγλωττιστές της C (και λιγότερο της C++) πέραν του ότι είναι ό,τι πιο διαδεδομένο στον χώρο, ενώ η ίδια η γλώσσα είναι πολύ «κοντά στη μηχανή» αποτελούσαν πάντα το «ινδικό χοιρίδιο» για τη μελέτη συμπεριφοράς τέτοιων λογισμικών. Επιπλέον, σε άλλες γλώσσες (π.χ. Java, Rust) έχει ληφθεί ειδική μέριμνα ώστε να προστατεύεται ο προγραμματιστής από απρόβλεπτες συμπεριφορές (ο μεταγλωττιστής δεν παίρνει πολλές «πρωτοβουλίες»), ακόμα και με θυσίες στον βωμό της ταχύτητας.

Πλήρης άδεια (shit doesn't happen)

Έχοντας αναφέρει τα παραπάνω, ας ξεκινήσουμε από τα απλά. Έστω λοιπόν ο παρακάτω κώδικας υλοποίησης μιας απλής Σειριακής Αναζήτησης:

```
#include <stdio.h>

#define N 10

int main()

{

        int x,a[N]={31,29,13,61,11,19,17,5,23,37};

        register int i=0;

        printf("Search for? ");

        scanf("%d", &x);
```

```
        //while ((i<N) && (a[i] != x))
        while ((a[i] != x) && (i<N))
                i++;
        if (i<N)
                printf("Found %d at pos: %d\n", x, i+1);
        else
                printf("%d: Not found.\n", x);
        return 0;
}
```

Αν μεταγλωττίσουμε το εν λόγω κομμάτι με τον gcc 9.3.0 και τις default επιλογές, όλα λειτουργούν όπως αναμένεται. Αν όμως ζητήσουμε τις μέγιστες δυνατές βελτιστοποιήσεις (gcc –O6) τότε, όταν επιλέξουμε προς αναζήτηση αριθμό που δεν υπάρχει στον πίνακα, θα δούμε ότι αυτός υπάρχει (πάντα) στην τελευταία θέση! Το πρόβλημα εδώ οφείλεται στον άνθρωπο και συγκεκριμένα στη γραμμή:

```
while ((a[i] != x)&& (i<N))
```

όπου στη συνθήκη, **πρώτα** προσπελαύνουμε τη θέση i του πίνακα κι έπειτα ελέγχουμε αν η θέση αυτή είναι έγκυρη (i<10). Αν τώρα εξετάσουμε με το utility *strings* το εκτελέσιμο που παρήχθη με την επιλογή –O6 (full optimizations)

```
strings a.out | grep "Not found" (Linux bash)
strings a.out | findstr "Not found" (Windows cmd)
```

θα δούμε πως δεν υπάρχει καν το κείμενο "Not found" μέσα σ' αυτό! Μάλιστα εξετάζοντας τον assembly κώδικα που έχει δημιουργήσει ο gcc, θα δούμε ότι ο μεταγλωττιστής θεωρεί πως δεν υπάρχει λόγος να ελεγχθεί ποτέ το δεύτερο σκέλος της λογικής σύζευξης: (i<N) κι έτσι έχει εξαφανίσει ως και το if-statement που έπεται της επανάληψης! Τούτο –για να το πούμε απλά- επειδή έχουμε νωρίτερα προσπελάσει θέση μνήμης δεικτοδοτούμενη από το i, άρα η τιμή του i δεν μπορεί να μην είναι εντάξει. Αυτή η στρατηγική υλοποίησης λέγεται "total license" και συνοπτικά μπορεί να περιγραφεί με τη φράση: Βελτιστοποίησε αντιμετωπίζοντας κάθε απρόβλεπτη συμπεριφορά ως κάτι που δεν μπορεί να συμβεί.

Προφανώς, αν γράψουμε διαφορετικά τη δομή *while* (όπως δηλ. στο σχόλιο της επάνω γραμμής), όλα διορθώνονται. Εδώ να σημειωθεί ότι ο MSVC (εμείς δοκιμάσαμε την εκδ. 19.29) με το

εν λόγω παράδειγμα δεν οδηγεί σε εκτελέσιμο με απρόβλεπτη συμπεριφορά, με ό,τι switch και να μεταγλωττίσουμε. Άρα, εδώ δεν έχουμε επιθετική βελτιστοποίηση (aggressive optimization).

Μη εμφανείς βελτιστοποιήσεις

Σε πολλές περιπτώσεις πάλι, τυχόν βελτιστοποιήσεις γίνονται σε υλοποιήσεις αόρατες στον άνθρωπο που γράφει τον κώδικα. Ενώ π.χ. στο παρακάτω, η εκχώρηση:

t = INT_MIN % -1; λειτουργεί κανονικά και δίνει στο t την τιμή 0, στη συνέχεια (γραμμή 14) παίρνουμε floating point exception (SIGFPE). Αν φυσικά το εκτελέσουμε με παραμέτρους γραμμής εντολών: -2147483648 και -1.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <limits.h>
4
5  int main(int argc, char *argv[])
6  {
7          int a,b,t;
8
9          t=INT_MIN % -1;
10         printf("%d\n", t);
11         if (argc != 3) return 1;
12         a=atoi(argv[1]);
13         b=atoi(argv[2]);
14         t= a % b;
15         printf("%d\n",t);
16         return 0;
17 }
```

Η ειρωνεία είναι πως ο gcc παραπονιέται για τη γραμμή 9: integer overflow in expression of type ‘int’ results in ‘0’ [-Woverflow] και όχι για την 14, αφού δεν “βλέπει” μέσα στην atoi(). Μάλιστα, το exception σηκώνεται είτε σε 32-bits σύστημα, είτε σε 64-bits, τόσο με τον MSVC όσο και με τον gcc, με και χωρίς τη χρήση optimization switch. Πιθανότατα, εδώ (στη συνάρτηση βιβλιοθήκης) υπάρχουν απρόσμενες βελτιστοποιήσεις. Το αβίαστο συμπέρασμα λοιπόν είναι: Μην εμπιστεύεσαι απόλυτα κώδικα άλλων, ιδίως αν δεν τον βλέπεις.

Πλήρης άδεια 2 (ποτέ δε γίνεται το ‘κακό’)

Έστω τώρα ο κώδικας:

```
1  #include <stdio.h>
2
3  main()
4  {
5
6  int i, j = 0;
7
8          for (i = 1; i > 0; i += i)
9                          ++j;
10         return j;
11 }
```

Αν μεταγλωττίσουμε το παραπάνω με τον gcc χρησιμοποιώντας τις default βελτιστοποιήσεις (χωρίς -O switch) ή με -O0 ή -O1 λειτουργεί ως αναμενόταν. Διαφορετικά, οδηγούμαστε σε ατέρμονα βρόγχο (το εκτελέσιμο δεν τερματίζει ποτέ). Γιατί άραγε;

Αυτήν τη φορά, όταν ζητήσουμε βελτιστοποίηση (ως προς το μέγεθος κώδικα **και** την ταχύτητα) ο μεταγλωττιστής εκτιμά πως εφόσον ο ακέραιος **i** διπλασιάζεται συνεχώς δεν θα γίνει ποτέ αρνητικός κι άρα ο βρόγχος δεν θα τελειώσει. Δηλαδή αγνοεί εντελώς την πιθανότητα ακέραιας υπερχείλισης (είπαμε: Shit never happens). Πράγματι: Αν δούμε τον assembly κώδικα που παράγει, όλη η δομή επανάληψης έχει περιοριστεί σε ένα λιτότατο (και φυσικά ατέρμον):

```
.L2:    jmp .L2
```

Για του λόγου το αληθές (γραμμές 13,14) :

```
 7 main:
 8 .LFB23:
 9         .cfi_startproc
10         endbr64
11         .p2align 4,,10
12         .p2align 3
13 .L2:
14         jmp     .L2
15         .cfi_endproc
```

Ο μεταγλωττιστής της Microsoft (MSVC) παρεμπιπτόντως, δεν φαίνεται δεν παίρνει τόσες … "πρωτοβουλίες", οπότε εκεί το παραπάνω πρόγραμμα λειτουργεί φυσιολογικά (κάνει overflow) σε κάθε περίπτωση. Ομοίως και ο LLVM compiler (Apple xcode). Σε κάθε περίπτωση, οφείλουμε να παρατηρήσουμε πως είναι μάλλον αδύνατο για τους προγραμματιστές/σχεδιαστές ενός μεταγλωττιστή να λάβουν υπ'όψιν τους το ενδεχόμενο ο συγγραφέας του πηγαίου κώδικα να βασίζεται σ' ένα overflow για τον τερματισμό ενός βρόγχου. Και να έχουμε την απαίτηση ο μεταγλωττιστής να μπορεί να κάνει και βελτιστοποιήσεις επ' αυτής της παραδοχής.

Είναι αλήθεια ότι οι προγραμματιστές ενίοτε επιδιώκουν να προκαλούνται "εξαιρέσεις" (exceptions) μέσω του κώδικά τους, τις οποίες εκμεταλλεύονται με χρήση κλήσεων τύπου *signal(2)* για τους δικούς τους λόγους. Τότε όμως, ας μην έχουμε την απαίτηση να λειτουργήσουν σωστά τα optimization switches του μεταγλωττιστή μας.

Τέλος, να παρατηρήσουμε πως αν ο διπλασιασμός της μεταβλητής **i** παραπάνω υλοποιηθεί με αριστερή ολίσθηση (shift left) κατά ένα bit (i <<= 1) που είναι και "φθηνότερο", ο μεταγλωττιστής ΔΕΝ θα κάνει τις εν λόγω προβληματικές υποθέσεις. Δε θα έχουμε δηλ. ατέρμονα βρόγχο.

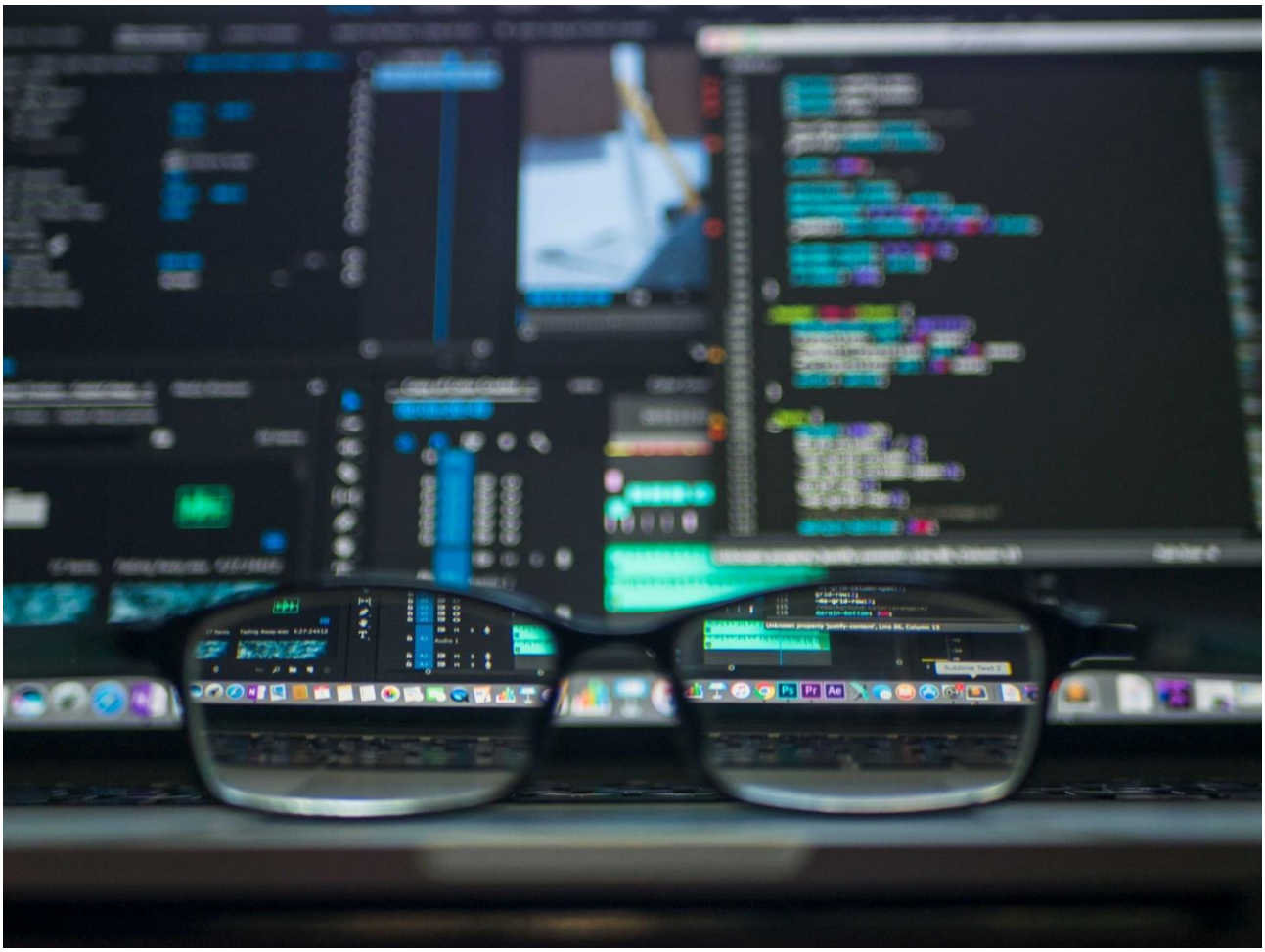


Photo: Kevin Ku

Εξάλειψη "νεκρού" κώδικα (dead code elimination)

Συχνά (επειδή κανείς δεν είναι τέλειος), τυχαίνει στον κώδικά μας να υπάρχουν κομμάτια που δεν εκτελούνται ποτέ ή που είτε εκτελεστούν, είτε όχι, δεν αλλάζει τίποτε στο αποτέλεσμα. Τα πιο συχνά είναι debugging messages που έχουν ξεχαστεί ή/και αχρείαστοι έλεγχοι. Παράδειγμα:

```
int foo(int x)
{
    if (x==0)
        return 0;
    else
        return 3*x;
}
```

Εδώ ο μεταγλωπτιστής (ενίοτε και χωρίς να ζητήσουμε καμιά βελτιστοποίηση) θα αντικαταστήσει, πολύ σωστά, όλο το παραπάνω if…statement μ' ένα απλό return 3*x; Και εννοείται πως δεν θα αλλάξει τίποτε στη συμπεριφορά της foo() και του προγράμματος μας εν γένει.

Ας δούμε όμως αυτό το κομμάτι κώδικα:

```
/* File: deadcode.c */

#include <stdio.h>

#include <stdlib.h>

#include <string.h>


int main()

{

char *pwd;


pwd=(char *)malloc(9*sizeof(char));

if (!pwd)

{

        fprintf(stderr, "malloc()");

        return 1;

}

strcpy(pwd, "abcdefgh");

memset(pwd, 0, sizeof(pwd));

//.....

free(pwd);

return 0;

}
```

Κι ας επιλέξουμε τώρα τον MSVC για την μεταγλώττισή του.

Λοιπόν, **με** optimizations: *cl /Fa.\optimized.asm deadcode.c /O2*

Χωρίς βελτιστοποιήσεις: *cl /Fa.\pure.asm deadcode.c*

Στο αρχείο `pure.asm` θα βρούμε κανονικά την κλήση της memset():

```
; Line 17
mov    r8d, 8
xor    edx, edx
mov    rcx, QWORD PTR pwd$[rsp]
call   memset
```

Κάτι που ΔΕΝ ισχύει για το `optimized.asm`

```
C:\Users\user\tmp>type optimized.asm | findstr "memset"

C:\Users\user\tmp>type pure.asm | findstr "memset"
EXTRN   memset:PROC
        call    memset

C:\Users\user\tmp>
```

Εντελώς το ίδιο συμβαίνει και με τον *gcc* (-O2 vs. -O0)

Προκύπτει φυσικά το ερώτημα: Και γιατί μας πειράζει αυτή η … "πρωτοβουλία"; Λοιπόν, η απάντηση είναι ότι σε τέτοιες περιπτώσεις υπάρχει μεγάλη πιθανότητα τα περιεχόμενα ενός buffer όπως ο `pwd` παραπάνω να παραμένουν στη μνήμη, προσπελάσιμα ενδεχομένως από κάποιο κακόβουλο λογισμικό που θα τα αντιγράψει. Αν τώρα μιλάμε για το string ενός συνθηματικού (password), καταλαβαίνουμε αμέσως γιατί **δεν**επιθυμούμε την εν λόγω βελτιστοποίηση. Έτσι, στον *gcc* μπορούμε π.χ. να επιλέξουμε το switch: *-fno-dce* (όπου DCE: Dead Code Elimination), οπότε είμαστε εμείς πλέον υπεύθυνοι, σαν καλοί προγραμματιστές, για την εξάλειψη κώδικα που -**πράγματι**- δεν έχει λόγο ύπαρξης.

Χρήση της υπόδειξης *volatile* (αφορά ξεκάθαρα C/C++)

Συχνά σήμερα τα προγράμματα μοιράζονται θέσεις στη μνήμη του υπολογιστή ή πρέπει να "ακούσουν" στο ίδιο port/socket, αν μη τι άλλο για να εξυπηρετηθούν ανάγκες διαδιεργασιακής επικοινωνίας (IPC). Σε τέτοιες μάλιστα περιπτώσεις, οι κλασικές C και C++ βρίσκονται στην κορυφή των προτιμήσεων καθώς κινούνται, όπως λέγεται, στα "χαμηλά νερά του υπολογιστικού συστήματος". Έστω λοιπόν ο παρακάτω κώδικας:

```
unsigned int *buf = (unsigned int *)0x8000;
```

```
// volatile unsigned int *buf=(unsigned int *)0x8000;

*buf=0;

while (*buf == 0)

{

        sleep(1); // `buf` is not even accessed here

}

….
```

Ο εν λόγω κώδικας φαινομενικά περιέχει έναν ατέρμονα βρόγχο (τα περιεχόμενα του δείκτη buf δε φαίνεται να αλλάζουν). Και πράγματι, αν μεταγλωττίσουμε το παραπάνω επιλέγοντας βελτιστοποιήσεις, θα έχουμε σίγουρα ένα εκτελέσιμο που δεν θα τερματίζει ποτέ. Συγκεκριμένα, βλέπουμε πως η εντολή: gcc -O2 -S παράγει τον παρακάτω assembly κώδικα:

```
.L2:

movl    $1, %edi

call    sleep@PLT

jmp     .L2
```

Το ίδιο συμβαίνει και με τον MSVC.

Φυσικά και δεν θέλουμε να έχουμε infinite loops στον κώδικά μας. Φανταστείτε τότε πως αντί για την -τυχαία- διεύθυνση 0x8000 που χρησιμοποιούμε στο παραπάνω παράδειγμα, έχουμε κάποιον δείκτη σε δ/νση μνήμης που δεσμεύτηκε π.χ. με μια κλήση στην dma_alloc_coherent() του Linux (ή στην αντίστοιχη άλλου Λ/Σ) για την εξυπηρέτηση αναγκών Direct Memory Accessing που χρειάζεται π.χ. ένας device driver. Αυτό σημαίνει πως τα περιεχόμενα της εν λόγω θέσης (όπου δείχνει ο `buf`) μπορεί ανά πάσα στιγμή να τροποποιηθούν από μια συσκευή ή διεργασία και άρα ο βρόγχος δεν είναι απαραίτητα ατέρμων. Οπότε κακώς παίρνει την παραπάνω "πρωτοβουλία" ο *gcc*(και ο *MSVC*).

Τι κάνουμε λοιπόν εδώ; Καταργούμε τις βελτιστοποιήσεις κατά τη μεταγλώττιση; Ευτυχώς όχι: Απλώς, χρησιμοποιούμε την υπόδειξη *volatile* προς τον compiler (η σχολιασμένη γραμμή). Για να το εξηγήσουμε όσο απλούστερα μπορούμε, αυτό μεταφράζεται ως οδηγία προς τον τελευταίο ότι: Η εν λόγω μεταβλητή μπορεί να αλλάξει τιμή σε οποιαδήποτε στιγμή από οποιονδήποτε, ακόμα και με "αόρατο" τρόπο. Κι έτσι το module των βελτιστοποιήσεων δεν θα ασχοληθεί με την τροποποίηση του βρόγχου που εξαρτάται από αυτήν!

Παραπομπή μέσω άκυρου δείκτη (Null pointer dereference)

Έστω το παρακάτω πρόγραμμα:

```
#include <stdio.h>
void null_deref(int *a) {
int c,*b;

b=a;
c=*b;
printf("%p OK!\n\n", b);
int k=c+2;
}
int main()
{ null_deref(NULL); }
```

Αν το μεταγλωττίσουμε και το τρέξουμε σε Linux με τις default επιλογές του gcc (όταν γραφόταν αυτές οι γραμμές χρησιμοποιήσαμε την έκδοση 9.3.0), θα πάρουμε ένα "Σφάλμα κατάτμησης" ("segmentation violation"), ήδη απ' τη γραμμή 7:c=*b, όπως φυσικά περιμέναμε. Παραδόξως, αν ζητήσουμε απ' τον μεταγλωττιστή και την ελάχιστη βελτιστοποίηση, μοιάζει να δουλεύει χωρίς προβλήματα. Πράγματι τυπώνει: "*(nil) OK!*" . Την ίδια επίσης συμπεριφορά παρουσιάζει και ο MSVC (32-bits, ver. 19.29). Αν δούμε τον assembly κώδικα που παράγει ο κάθε μεταγλωττιστής, θα παρατηρήσουμε πως στην περίπτωση που επιλέξουμε βελτιστοποίηση (π.χ. -O2 ή /O2 αντίστοιχα) οι αναφορές σε null pointer (γραμμές 7 και 9) απουσιάζουν παντελώς. Παρακάτω σε assembly (MSVC) :

```
//Χωρίς βελτιστοποιήσεις
_null_deref PROC
; Line 4
push    ebp
mov     ebp, esp
sub     esp, 12
```

```
//Με την επιλογή /O2 (MSVC)
_null_deref PROC
; Line 9
push    DWORD PTR _a$[esp-4]
push    OFFSET
??_C@_08IMKGPAFH@?$CFp?5OK?$CB?6?6@
```

```
; Line 7
mov    eax, DWORD PTR _a$[ebp]
mov    DWORD PTR _b$[ebp], eax
; Line 8 (c=*b)
mov    ecx, DWORD PTR _b$[ebp]
mov    edx, DWORD PTR [ecx]
mov    DWORD PTR _c$[ebp], edx
; Line 9
mov    eax, DWORD PTR _b$[ebp]
push   eax
push   OFFSET $SG9253
call   _printf
add    esp, 8
; Line 10 (k=c+2)
mov    ecx, DWORD PTR _c$[ebp]
add    ecx, 2
mov    DWORD PTR _k$[ebp], ecx
; Line 11
mov    esp, ebp
pop    ebp
ret    0
_null_deref ENDP
```

```
call   _printf
add    esp, 8
; Line 11
ret    0
_null_deref ENDP
```

Εδώ φαίνεται να συμβαίνουν τα εξής: Βλέποντας ο μεταγλωττιστής πως κατηγορηματικά γίνεται αναφορά (derefencing) στον δείκτη b (γραμμή 7), θεωρεί πως αυτός **δεν** μπορεί να είναι null (total license) και επιτρέπει κι άλλες αναφορές στα περιεχόμενα της θέσης μνήμης (κι ας είναι 0) παρακάτω. Μπορούμε ενδεχομένως να μιλήσουμε και για bug των εν λόγω μεταγλωττιστών. Σε κάθε περίπτωση πάντως, ο καλύτερος τρόπος να προφυλαγόμαστε από τέτοιες απρόβλεπτες

συμπεριφορές είναι να ελέγχουμε πάντα έναν δείκτη για έγκυρη τιμή (όχι NULL δηλ. όχι 0) πριν πάμε να τον χρησιμοποιήσουμε. Καθώς όμως κι αυτό μπορεί να αγνοηθεί ελέω αυτόματης βελτιστοποίησης (θυμίζουμε την αρχή πλήρους άδειας: Total license), κάποιες επιλογές κατά τη μεταγλώττιση, όπως η *-fno-delete-null-pointer-checks* του *gcc,* μπορεί ενίοτε να μας προστατεύσουν.

Ίσως τώρα ρωτήσει κανείς **γιατί** είναι τόσο ανεπιθύμητες τέτοιες απρόβλεπτες συμπεριφορές (αφού δε "σηκώνεται" exception); Η μια απάντηση λοιπόν είναι: Επειδή ακριβώς είναι απρόβλεπτες, ενώ αντιθέτως έχουμε μάθει ότι οι Η/Υ είναι **ντετερμινιστικοί**. Αν όμως μιλήσουμε πιο "τεχνικά", θα πούμε πως δεν θέλουμε γενικά ο όποιος κακοπροαίρετος να μπορεί να "αξιοποιήσει" τη σελίδα μνήμης μηδέν 0 (π.χ. με μια mmap()). Εκεί φυλάσσονται δεδομένα και κώδικας του πυρήνα (αν π.χ. μιλάμε για Linux). Επίσης, αν παρακάτω χρειαστεί να τυπωθούν ας πούμε τα -ανύπαρκτα- περιεχόμενα του null δείκτη, τότε πιθανότατα θα λάβουμε κάποιο exception σε φαινομενικά άσχετο σημείο του προγράμματος.

Επίλογος

Η παραπάνω λίστα με προβλήματα λόγω βελτιστοποίησης θα μπορούσε να επεκταθεί αρκετά, αλλά ας επιφυλαχθούμε για το μέλλον, προκειμένου -αν μη τι άλλο- να μη γίνει πολύ κουραστικό το παρόν (κείμενο). Εν κατακλείδι, αυτά που θα προτείναμε να κρατήσει κανείς είναι ακριβώς όσα διδάσκονται ως πρωτόλεια στο πρώτο εξάμηνο των σπουδών Πληροφορικής. Θα τολμήσουμε δε, ως επίλογο, να τα συμπυκνώσουμε στα εξής:

- Μπορούμε και μόνοι μας να γράψουμε κώδικα με τον βέλτιστο τρόπο.
- Ο μεταγλωττιστής δεν είναι θεός· ούτε και τον έγραψε ο … Πάπας.
- Πάντα γράφουμε κώδικα έχοντας κατά νου το χειρότερο που μπορεί να συμβεί.
- Όταν προγραμματίζουμε σε χαμηλό επίπεδο σκεφτόμαστε πως όλοι οι άλλοι θέλουν να βλάψουν το έργο μας.
- Μια ματιά στον παραγόμενο assembly κώδικα ίσως αποκαλύψει το bug που αναζητούμε εδώ και καιρό.
- Μελετούμε διαρκώς και εξονυχιστικά τα εγχειρίδια των εργαλείων ανάπτυξης. Κοινώς: RTFM.
- Τέλος: Υπάρχει όντως λόγος που αναπτύχθηκαν εργαλεία όπως profilers, code analyzers (lint, splint) και επιλογές μεταγλώττισης (flags) όπως -Wall, __STDC_ANALYZABLE__ κ.α.

✔ 40 εγχειρίδια ανοικτών τεχνολογιών ελεύθερα διαθέσιμα σε μορφή e-book

Συγκεντρωμένα σε μία λίστα στην **Ανοικτή Βιβλιοθήκη** (www.openbook.gr) όλα τα ελληνικά ψηφιακά εγχειρίδια, που αφορούν αποκλειστικά ανοικτές τεχνολογίες και ελεύθερο λογισμικό ανοικτού κώδικα:

- "Προγραμματίζοντας με τον μικροελεγκτή **Arduino**"
- "Δημιουργώ με το **Arduino** και προγραμματίζω με το **Ardublock**"
- "Εκπαιδευτική Ρομποτική με τον μικροελεγκτή **Arduino**"
- "Εγκατάσταση **Joomla**
- "Η γλώσσα προγραμματισμού **Scratch**" – Δραστηριότητες για το Δημοτικό
- "Το **Scratch** πάει Δημοτικό!"
- "**Scratch** – Οδηγός χρήσης"
- "**Scratch** – Οδηγός για Αρχάριους"
- "Δημιουργώ παιχνίδια στο **Scratch**"

- "Παίζω και προγραμματίζω με το ρομπότ **Thymio**"
- "Οδηγός για το Σύστημα Ασύγχρονης Τηλεκπαίδευσης **Moodle**"
- Συνοπτικός Οδηγός Χρήσης του **Moodle** για τον Καθηγητή
- "Ξεκινώντας με το **Ubuntu** 10.04"
- "Εγκατάσταση και Διαχείριση ΣΕΠΕΗΥ με **Ubuntu**"
- "**ΕΛ/ΛΑΚ** Λογισμικά Διαχείρισης Τάξης"
- "**Linux Mint** 17.2 – Επίσημος Οδηγός Χρήστη"
- "**UNIX**: Οδηγός αρχαρίων σε οκτώ μαθήματα"
- "**LaTeX** για βάρβαρους"
- "Εισαγωγή στον αντικειμενοστραφή προγραμματισμό με **Python**"
- "Εισαγωγή στον Προγραμματισμό με αρωγό τη γλώσσα **Python**"
- "Εισαγωγή στον προγραμματισμό με την **Python**"
- "Προγραμματισμός με **Python** στο **Raspberry Pi**"
- "Οδηγός **Python** μέσω παραδειγμάτων"
- "Προγραμματισμός σε **App Inventor**"
- "Κατασκευή δικτυακού τόπου με το **WordPress**"
- "Δημιουργία ιστοσελίδας με **WordPress**" – Βασικές λειτουργίες
- "Ελληνική μετάφραση του **OpenCart**" – Οδηγός Χρήσης
- "Οδηγός Λειτουργιών του **Firefox** OS"
- "Ελληνικός οδηγός **Mozilla Thunderbird**"
- "Στατιστική με το **OpenOffice**"
- "**Audacity** 1.3.13 – Σύντομος οδηγός χρήσης"
- "Πρόγραμμα επεξεργασίας εικόνων **GIMP**"
- "Το εγχειρίδιο του **BlueJ**"
- "Το τετράδιο της **Wikipedia**" – Σύντομος οδηγός συγγραφής άρθρων
- "**Wikipedia** – Εισαγωγή στην ελεύθερη ψηφιακή εγκυκλοπαίδεια"
- "Χρησιμοποιώντας τα **Wikibooks** στην τάξη" – Βέλτιστες πρακτικές
- "Οδηγός Χρήσης Ανοιχτού Λογισμικού Διαχείρισης Έργων **OpenProj** στην Εκπαίδευση"
- "Οδηγός **Ελεύθερου Λογισμικού** για Μικρομεσαίες Επιχειρήσεις"
- "Εγχειρίδιο **Ανοικτών Δεδομένων**"
- "Χτίζοντας μια κοινότητα **Ελεύθερου Λογισμικού**"

✔ Brain – train (και ουχί 'drain')

Γρίφοι & προβλήματα από την Επιστήμη των Υπολογιστών για μαθητές

Επιμέλεια: **Φώτης Αλεξάκος**

Photo: Meo

✔ Εξάσκηση 1

Πέντε ιδιότροποι πλην τίμιοι "κομπιουτεράδες" (έτσι είναι πάντα αυτοί), πίνουν τον καφέ τους συζητώντας περί αμοιβών στον ιδιωτικό τομέα. Θέλουν να υπολογίσουν τον μέσο μισθό τους, αλλά χωρίς να μάθει κανείς τους πόσα παίρνει ο όποιος άλλος. Διαθέτουν χαρτί και μολύβι. Πώς θα το κάνουν;

(Ανώνυμος φοιτητής στο Berkeley, 1996)

✔ Εξάσκηση 2

Ίσως γνωρίζετε ότι παλινδρομικοί (ή καρκινικοί) λέγονται οι αριθμοί που διαβάζονται το ίδιο από δεξιά κι από αριστερά. Π.χ. 575, 64746 κλπ. Αν ξεκινήσουμε από έναν διψήφιο αριθμό και αφού αντιστρέψουμε τα ψηφία του για να βρούμε τον κατοπτρικό του, τους προσθέσουμε και συνεχίσουμε έτσι, κάποια στιγμή θα καταλήξουμε σ' έναν παλινδρομικό ακέραιο. Π.χ. Αν πάρω το 28, έχουμε: 28+82=110 και 110+011=121 και βρήκα παλινδρομικό. Εδώ χρειάστηκαν δύο προσθέσεις. Θέλουμε να βρείτε τον μικρότερο διψήφιο ακέραιο που χρειάζεται τον μέγιστο αριθμό τέτοιων πράξεων για να φτάσουμε σε παλινδρομικό. Επίσης και το πόσες είναι οι προσθέσεις που απαιτήθηκαν.

Προσοχή: Θα χρειαστεί να "μπλέξετε" με πολύ μεγάλους ακέραιους (που δεν χωρούν - ας πούμε- στον τύπο unsigned long της C, ακόμα και σε Η/Υ των 64-bits).

(Περιοδικό Pixel, τ. 33, Μάϊος 1987)

✔ Εξάσκηση 3

Φτιάξτε πρόγραμμα που να δέχεται ένα εύρος ακεραίων (π.χ. [3, 250]) να εμφανίζει όλες τις **πρωτεύουσες** Πυθαγόρειες τριάδες στο διάστημα αυτό. Μια τριάδα ακεραίων α,β,γ λέγεται Πυθαγόρεια αν: β2+γ2=α2. Π.χ. {3,4,5} ή {8,15,17}. Για να είναι πρωτεύουσα *(primitive)* δεν πρέπει οι όροι της να είναι πολλαπλάσια άλλης Πυθαγόρειας τριάδας. Π.χ. Η {16, 30, 34} δεν είναι πρωτεύουσα αφού: 16=2*8, 30=2*15 και 34=2*17. Προκύπτει δηλ. Από την {8, 15, 17}.

Υπόδειξη: Υπάρχει στη διάθεσή σας ολόκληρη εργασία στα Ελληνικά (με τη φόρμουλα του Ευκλείδη κλπ): https://is.gd/4OyNRs

---

✔ *Στείλτε αν θέλετε τις δικές σας λύσεις στο* newsletter@epe.org.gr

☆ *Οι απαντήσεις των γρίφων θα δημοσιευθούν στο επόμενο τεύχος*

✔ Brain – train

Οι λύσεις των γρίφων του προηγούμενου 19ου τεύχους

Εξάσκηση 1

Προφανώς, αφού μιλάμε για μόλις δύο (2) διαφορετικά χρώματα, αρκεί να πάρουμε τρεις (3) κάλτσες απ' το συρτάρι. Οι δυο εξ αυτών θα είναι αναγκαστικά είτε κίτρινες, είτε μαύρες.

Εξάσκηση 2

Το πρόβλημα με τους πειρατές, τη μαϊμού και τις καρύδες θεωρείται πλέον κλασικό και φυσικά έχουν δημοσιευτεί πολλές λύσεις. Είτε Μαθηματικές, είτε με χρήση άπλητων αλγορίθμων.

Πληθώρα τέτοιων βρίσκονται στην ίδια την Wikipedia:

https://en.wikipedia.org/wiki/The_monkey_and_the_coconuts

Εδώ όμως έχουμε την ευκαιρία να παρουσιάσουμε έναν ακόμη τρόπο για να φτάσει κανείς στο ζητούμενο, σκέψη που ελπίζουμε να φανεί πιο κατανοητή ιδίως σε αναγνώστες μικρότερων ηλικιών: Έστω λοιπόν ότι δεν περίσσευε ποτέ καμμία καρύδα σε κανέναν πειρατή, άρα όλοι οι σωροί αποτελούνται από πλήθη που είναι πολλαπλάσια του 5. Αν Ν λοιπόν οι καρύδες στον αρχικό σωρό, υποψιαζόμαστε ότι αν το Ν είναι δύναμη του 5, θα υπάρχουν πάντα πολλαπλάσια του 5 καρύδες στον εναπομείναντα σωρό. Για να δούμε γιατί. Επιλέγουμε N=5κ και στο επόμενο βήμα:

N2=5κ - 5κ/5 =5κ – 5κ-1=5κ-1*(5-1)=4*5κ-1 , N3=4*5κ-1-4*5κ-1/5=4* 5κ-1-4*5κ-2=4*5κ-2*4=42*5κ-2 κοκ, οπότε γενικά: Nk=4k-1*56-k.

Έτσι, για κ=5 θα έχουμε αρχικά 55=3125 καρύδες στον αρχικό σωρό (θυμίζουμε πως ζητάμε το ελάχιστο πλήθος τέτοιων). Και το πρωί θα έχουν μείνει: 44*5=1280.

Τώρα: Εμείς θέλουμε να … τρώει μια καρύδα κι η μαϊμού εκτός απ' αυτές του πρωινού σωρού. Ας πάμε να βρούμε την απάντηση με δοκιμές. Εκ των παραπάνω μπορούμε να υποθέσουμε πως το 3125 θα ήταν ένα καλό νούμερο για να ξεκινήσουμε. Όμως αυτό είναι

πολλαπλάσιο του 5. Προφανώς λοιπόν και δεν μας κάνει, αλλά αν πάμε στον αμέσως μικρότερο ακέραιο (3124) αυτός μας απογοητεύει ήδη απ' τον 1ο πειρατή, καθώς 3124 mod 5=4. Αντίστοιχα: 3123 mod 5 = 3, 3122 mod 5=2 και μένει ο 3121 mod 5 = 1. Παίρνει λοιπόν ο πρώτος πονηρός 3121 div 5 = 624 καρύδες να θάψει και δίνει και μία στη μαϊμού. Άρα μείναν: 3121-624-1=2496 καρύδες. Όμως 2496 mod 5 = 1, οπότε σαν να μας βολεύει ο αριθμός αυτός και για τον επόμενο που θα ξυπνήσει. Ας μην πλατειάζουμε: Κάνοντας τους υπολογισμούς, θα δούμε πως οι 3121 καρύδες στον αρχικό σωρό είναι πράγματι η σωστή λύση και την βρήκαμε γρήγορα επειδή ψάξαμε "κοντά" στο kk, όπου k=5 είναι ο αριθμός των πειρατών. Αν απ' το 3125 πηγαίναμε προς τα πάνω, θα απογοητευόμασταν αμέσως απ' το 3126 κι ας ισχύει το, ενθαρρυντικό αρχικά: 3126 mod 5 = 1. Αφού: 3126 – 3126 div 5 – 1=2500 που ως πολλαπλάσιο του 5 προφανώς δεν μας κάνει (ο επόμενος πειρατής δεν αφήνει τίποτε για την μαϊμού).

Παραθέτουμε παρακάτω μια πρόταση για πρακτική εφαρμογή αυτού του "ψαξίματος" σε γλώσσα C και στη συνέχεια ένα Visual Basic script (να θυμόμαστε και τα παλιά) με την γενική (generic) λύση του puzzle. Στο προγραμματάκι C μπορούμε να βάλουμε δυο (2) παραμέτρους απ' τη γραμμή εντολών: Πρώτα τον ακέραιο απ' τον οποίον (και προς τα κάτω) θέλουμε να ξεκινήσουν οι δοκιμές (π.χ. 3125) κι έπειτα το πόσοι ήταν οι πειρατές (π.χ. 5)..

Trial and error (C)	***Γενική λύση (VB script)***

#include <stdio.h>
#include <stdlib.h>
 	
short checkInitCocos(long *n, long s)
{
  long i,k;
  short Ok, done=0;
 	
  i=*n;
  while ((i%s)!=1) i--;
  *n=i;
  i=i-(long)(i/s)-1;
  while ((*n>s) &&(!done))
  {
  Ok=1;
  for (k=2;k<=s;k++)
  {
  if ((i%s)!=1) {
  Ok=0; *n--; i=*n; break;
  }
  i=i-(long)(i/s)-1;
  }
  if (Ok &&(i%s)==0)
  done=1;
  else
  {
  *n--;
	dim i, sh, K, KM, N
	
	dim str
	Do
	k=cint(inputbox("How many sailors?", "Enter an integer>1","5",vbInformation))
	loop until k>1
	km=k mod 2
	if km<>0 then
	  n=k^k-k+1
	Else
	  n=(k-1)*(k^k-1)
	end if
	str="N will be: "+cstr(N)+vbCrlf
	for i=0 to k-1
	  sh=int(n/k)+1
	  N = N-sh
	str=str+"Remaining coconuts: "+cstr(N)+". Monkey gets: "+ &
	  cstr(n mod k)+vbCrlf
	next
	Msgbox str, vbOKOnly, vbInformation

```
            i=*n;
        }
    }
    return done;
}

int main(int argc, char *argv[])
{
        long sv,n=3125,s=5;

        if (argc >1)
                n=atol(argv[1]);
        if (argc >2)
                s=atol(argv[2]);
        if ((s<=0) || (n<=s)) return 1;
        while (n>s)
        {
      if (checkInitCocos(&n,s)) sv=n;
      n--;
        }
        printf("N : %ld\n",sv);
        return 0;
}
```

Εξάσκηση 3

Η απάντηση είναι κομμάτια με μάζες: 1, 3, 9, 27 και 81 κιλά αντίστοιχα, δηλ. 3^0, 3^1, 3^2, 3^3 και 3^4 κιλά. Γενικεύοντας, μπορούμε να πούμε ότι (αναφερόμαστε φυσικά σε ακέραιους), τα βάρη θα είναι πάντα δυνάμεις του 3. Μια λακωνική εξήγηση θα μπορούσε να είναι ότι κάθε ακέραιος έχει μοναδικό τρόπο αναπαράστασης στο τριαδικό σύστημα, ακόμα και στο "ισορροπημένο τριαδικό" (*balanced ternary όπως στον Σοβιετικό υπολογιστή Setun*) όπου αντί για τα ψηφία: 0,1 και 2 χρησιμοποιούνται τα -1, 0 και 1. Από εκεί και πέρα, έχουμε τρεις διακριτές καταστάσεις για κάθε κομμάτι: i) Μπορεί να τοποθετηθεί στην αριστερή βάση της πλάστιγγας, ii) στην δεξιά ή iii) να μην χρησιμοποιηθεί καθόλου. Αν έχουμε n κομμάτια (βάρη), θα έχουμε 3^n τέτοιους συνδυασμούς. Ένας εξ' αυτών θα είναι ότι δεν χρησιμοποιήθηκε κανένα βάρος. Μένουν λοιπόν: 3^n-1 συνδυασμοί που θα πρέπει να διαιρεθούν με το 2 για να μη μετράμε το ίδιο κομμάτι στη δεξιά και στην αριστερή μεριά. Έτσι λοιπόν θα έχουμε $(3^n-1)/2$ συνολικούς συνδυασμούς χρήσης βαρών. Στην περίπτωσή μας λοιπόν, πρέπει να είναι n=5 ώστε να υποστηρίζονται $(3^5-1)/2=(243-1)/2=242/2=121$ περιπτώσεις. Αν ο βράχος ζύγιζε 40Kgr, θα είχαμε n=4 κομμάτια, αφού $(3^4-1)/2=40$ με αντίστοιχες μάζες: $3^0=1$, $3^1=3$, $3^2=9$ και $3^3=27$ κιλά.

Επισκεφθείτε μας στο web
www.epe.org.gr

Γίνετε μέλος της ΕΠΕ

Για την ανάδειξη της
Πληροφορικής στη χώρα

Η Ένωση Πληροφορικών Ελλάδος υπάρχει για να δημιουργεί τις προϋποθέσεις για την προαγωγή της Πληροφορικής, αξιοποιώντας τις δυνάμεις των Πληροφορικών και ικανοποιώντας τις εργασιακές και επιστημονικές τους ανάγκες όπου και αν εργάζονται ή διαμένουν.
Είναι η κατάληξη της αναζήτησης όλων των Πληροφορικών για ένα ισχυρό φορέα του κλάδου που να αναδεικνύει αξιόπιστα τον κοινωνικό τους ρόλο και να τους εκπροσωπεί αυθεντικά σε όλα τα πεδία των ενδιαφερόντων τους.
Είναι η αφετηρία μιας μεγαλόπνοης προσπάθειας που επιδιώκει να κινητοποιήσει όλες τις ζωντανές δυνάμεις της κοινωνίας και να πορευτεί, μαζί μ᾽ αυτές, προς έναν καλύτερο κόσμο για όλους.
Σταθμός σε αυτή την πορεία και στρατηγικός στόχος της ΕΠΕ είναι η δημιουργία του Επιμελητηρίου Πληροφορικής.
Η δράση και οι παρεμβάσεις της είναι ο καταλύτης για την ωρίμανση των αναγκαίων κοινωνικών και πολιτικών συνθηκών.
Οι αξίες που καλλιεργεί θα αποτελέσουν την κληρονομιά και το όραμα του θεσμικού αυτού φορέα. Για να μπορέσουν όλοι οι πληροφορικοί να βρουν τη θέση που τους αξίζει στον κόσμο που όλοι μας οραματιζόμαστε.

LIKE US ON facebook
https://www.facebook.com/EnosiPliroforikonElladas

LinkedIn
https://www.linkedin.com/groups?gid=66328

twitter
https://twitter.com/epe_gr

YouTube
https://www.youtube.com/user/hiuaccount

RSS
http://www.epe.org.gr/index.php?id=7&type=100