ΕΛΛΗΝΙΚΌ ΜΕΣΟΓΕΙΑΚΌ ΠΑΝΕΠΙΣΤΉΜΙΟ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ



ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

SUDOKU ΣΕ ΓΛΏΣΣΑ C

ΝΙΚΟΛΑΚΌΠΟΥΛΟΣ ΝΙΚΌΛΑΟΣ

ΠΑΤΡΩΝΥΜΟ: ΝΙΚΟΛΑΚΟΠΟΥΛΟΣ ΓΕΩΡΓΙΟΣ

Α.Μ. : ΤΛ20262

ΕΠΙΒΛΈΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΑΝΤΩΝΙΟΣ ΚΩΝΣΤΑΝΤΑΡΑΣ

ΚΡΗΤΗ, 2025

HELLENIC MEDITERRANEAN UNIVERSITY

DEPARTMENT OF ELECTRONIC ENGINEERING



SUDOKU IN C LANGUAGE

NIKOLAKOPOULOS NIKOLAOS

FATHER'S NAME: NIKOLAKOPOULOS GEORGIOS

REG. NO. : ΤΛ20262

SUPERVISOR PROFESSOR: ANTONIOS KONSTANTARAS

CRETE, 2025

Περιεχόμενα/Contents

Ευχαριστίες 5

Περίληψη 6

Abstract 7

1.Εισαγωγή 8

1.1. Περιγραφή της διπλωματικής 8

1.2. Γλώσσα C 9

Τι είναι η γλώσσα C και πότε εμφανίστηκε; 9

Ποια οφέλη προσφέρει η γλώσσα C; 10

Τα χαρακτηριστικά της γλώσσας C 10

Πως μπορεί να μάθει κάποιος την γλώσσα c; 11

Οι μελλοντικές προοπτικές της γλώσσας C 12

1.3. Compiler 13

Τι είναι το Compiler και ποιες είναι οι λειτουργίες του; 13

Βήματα εκτέλεσης του Compiler 14

Τύποι μεταγλωττιστών 15

Πλεονεκτήματα του Compiler 17

1.4. Πρόγραμμα Sudoku 19

Τι είναι το sudoku και πότε εμφανίστηκε; 19

Κανόνες και τρόποι επίλυσης 20

Μαθηματικές ιδιότητες του Sudoku 21

NP-Complete 24

Brute Force....26

Algorithm X....27

1.5 Κώδικας του Sudoku στη γλώσσα C....29

1.6 Δεντροδιάγραμμα....37

2. Ανάλυση του Κώδικα....38

2.1. Περιγραφή του Κώδικα....38

2.2. Screenshots....53

Εικόνες από τον κώδικα....53

Όταν τρέχουμε τον κώδικα....56

3.Βιβλιογραφία/Πηγές....59

Ευχαριστίες

Θα ήθελα να εκφράσω τις ειλικρινείς μου ευχαριστίες στον επιβλέποντα καθηγητή μου, κύριο Αντώνιο Κωνσταντάρα, για την καθοδήγηση, τη διαρκή υποστήριξη και τις πολύτιμες συμβουλές του καθ' όλη τη διάρκεια της διπλωματικής μου εργασίας.

Ιδιαίτερες ευχαριστίες οφείλω στην οικογένειά μου, για την αμέριστη αγάπη, την ηθική συμπαράσταση και την ενθάρρυνση που μου προσέφερε όλα αυτά τα χρόνια. Η παρουσία και η στήριξή τους υπήρξαν καθοριστικές για την πορεία και την ολοκλήρωση των σπουδών μου.

Περίληψη

Η διπλωματική εργασία επικεντρώνεται στην υλοποίηση του δημοφιλούς παιχνιδιού Sudoku με τη χρήση της γλώσσας προγραμματισμού C. Αρχικά, παρουσιάζω μια εισαγωγή στη γλώσσα C, τα χαρακτηριστικά της, την ιστορική της εξέλιξη και τα πλεονεκτήματά της στην ανάπτυξη λογισμικού. Στη συνέχεια, αναλύω την έννοια και τις μαθηματικές ιδιότητες του Sudoku, καθώς και διάφορες μέθοδοι επίλυσης, όπως ο Brute Force, ο Algorithm X και η θεωρία NP-Completeness. Η εργασία περιλαμβάνει πλήρη υλοποίηση του παιχνιδιού σε C, με σχολιασμό κάθε εντολής, δεντροδιάγραμμα ροής, και στιγμιότυπα από την εκτέλεση του προγράμματος. Τέλος, παρουσιάζονται βασικά χαρακτηριστικά των μεταγλωττιστών και η διαδικασία μεταγλώττισης που χρησιμοποιήθηκε για την εκτέλεση του προγράμματος.

Abstract

The thesis focuses on the implementation of the popular Sudoku game using the C programming language. Initially, I present an introduction to the C language, its features, its historical development and its advantages in software development. Then, I analyze the concept and mathematical properties of Sudoku, as well as various solving methods, such as Brute Force, Algorithm X and the NP-Completeness theory. The work includes a complete implementation of the game in C, with comments on each command, a flowchart, and screenshots from the execution of the program. Finally, I present basic features of compilers and the compilation process used to execute the program.

1.Εισαγωγή

1.1. Περιγραφή της διπλωματικής

Η παρούσα διπλωματική εργασία ασχολείται με την ανάπτυξη του παιχνιδιού Sudoku στη γλώσσα προγραμματισμού C. Το Sudoku αποτελεί ένα παζλ λογικής, το οποίο έχει αποκτήσει μεγάλη απήχηση παγκοσμίως, καθώς συνδυάζει την απλότητα των κανόνων με την πολυπλοκότητα της επίλυσης. Η εργασία δεν περιορίζεται μόνο την πρακτική υλοποίηση, αλλά περιλαμβάνει και την θεωρητική ανάλυση της φύσης του παιχνιδιού, της μαθηματικής του δομής, και των υπολογιστικών τεχνικών που μπορούν να χρησιμοποιηθούν για την επίλυσή του.

Παράλληλα, γίνεται εκτενής αναφορά στη γλώσσα προγραμματισμού C, η οποία επιλέχθηκε λόγω της αποδοτικότητας, της φορητότητας και της ευρείας αποδοχής της στον χώρο της ανάπτυξης συστημάτων. Παρουσιάζω τα βασικά χαρακτηριστικά της γλώσσας, τα πλεονεκτήματα της χρήσης της, καθώς και τη διαδικασία εκμάθησής της.

Στο κύριο μέρος της εργασίας αναλύω τον κώδικα του παιχνιδιού, γραμμή προς γραμμή, με στόχο να διευκολυνθεί η κατανόηση από τον αναγνώστη, ακόμα και εάν αυτός δεν διαθέτει εμπειρία στον προγραμματισμό. Επιπλέον, παρουσιάζω τη διαδικασία μεταγλώττισης, τα εργαλεία που χρησιμοποίησα (όπως ο online compiler OnlineGDB), καθώς και στιγμιότυπα της λειτουργίας του προγράμματος.

Η εργασία στοχεύει να αποτελέσει ένα ολοκληρωμένο παράδειγμα ανάπτυξης λογισμικού με τη γλώσσα C, συνδυάζοντας θεωρία, υλοποίηση και εφαρμογή.

1.2. Γλώσσα C

Τι είναι η γλώσσα C και πότε εμφανίστηκε;



Το logo της γλώσσας C [1]

Η **γλώσσα C** είναι μια γλώσσα προγραμματισμού γενικής χρήσης.Η C αποτελεί μία από τις πιο ευρέως χρησιμοποιούμενες γλώσσες προγραμματισμού και πολλές νεώτερες γλώσσες έχουν επηρεαστεί από αυτήν,όπως C++,C#,Java, JavaScript, PHP, Python και πολλές άλλες γλώσσες.

Το C είναι μια εξαιρετική επιλογή για τον προγραμματισμό του συστήματος, όπως η ανάπτυξη λειτουργικών συστημάτων, των μεταγλωττιστών και των οδηγών δικτύων. Παρά τη δημοτικότητά του, ο C δεν είναι χωρίς τις επικρίσεις του. Κάποιοι υποστήριξαν ότι η σύνταξή της θα μπορούσε να είναι πιο περίπλοκη και ευκολότερη μάθηση, ενώ άλλοι έχουν σημειώσει την έλλειψη τυποποίησης ως σημαντικό ζήτημα. Παρ 'όλα αυτά, το C παραμένει μια ευρέως χρησιμοποιούμενη και επιρροή γλώσσα και πιθανότατα θα συνεχιστεί για πολλά χρόνια.[2]

Η γλώσσα προγραμματισμού C δημιουργήθηκε σε Bell Laboratories στις αρχές της δεκαετίας του 1970, κυρίως από τους Ken Thompson και Dennis Ritchie. Για το λειτουργικό σύστημα UNIX, το οποίο κατά το χρόνο απαιτούσε τις εφαρμογές να γράφονται στη γλώσσα συναρμολόγησης, οι προγραμματιστές χρειάζονταν ένα πιο φιλικό προς το χρήστη σύνολο οδηγιών. Τα προγράμματα συναρμολόγησης, τα οποία

επικοινωνούν απευθείας με το υλικό του υπολογιστή, είναι μακρά και πολύπλοκα για να εντοπίσουν σφάλματα και η προσθήκη νέας λειτουργικότητας απαιτεί πολύ χρόνο και προσπάθεια.[3]

Ποια οφέλη προσφέρει η γλώσσα C;

Το C είναι μια ισχυρή γλώσσα προγραμματισμού που προσφέρει πολλά οφέλη έναντι άλλων γλωσσών γιατί:

- είναι μια καθολική γλώσσα που μπορεί να χρησιμοποιηθεί για διάφορες εφαρμογές.
- είναι μια πολύ αποτελεσματική γλώσσα που μπορεί να γράψει γρήγορα και αξιόπιστο κώδικα.
- είναι μια φορητή γλώσσα, που σημαίνει ότι ο κώδικας που γράφτηκε στο C μπορεί εύκολα να καταρτιστεί και να εκτελεστεί σε διάφορες πλατφόρμες.
- είναι μια καθιερωμένη γλώσσα με μια μεγάλη και ενεργή κοινότητα προγραμματιστών που εργάζονται συνεχώς για τη βελτίωση της και τη δημιουργία νέων εργαλείων και βιβλιοθηκών.[4]

Τα χαρακτηριστικά της γλώσσας C

Το C είναι μια γλώσσα προγραμματισμού υπολογιστών γενικού σκοπού για τη διαχείριση του συστήματος, τον προγραμματισμό δικτύου και το ενσωματωμένο λογισμικό.

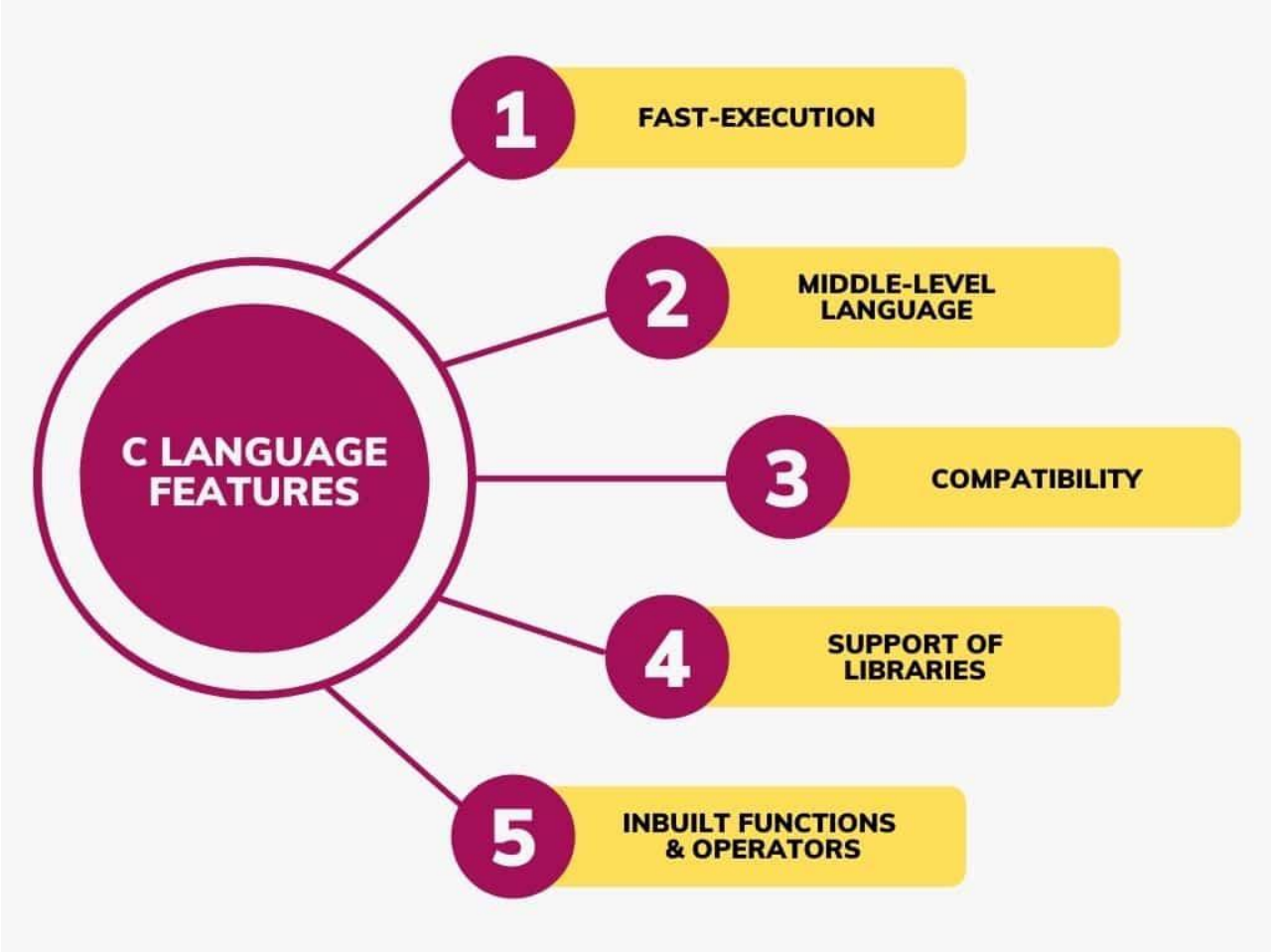
Έχει πολλά χαρακτηριστικά που το καθιστούν επιθυμητό για αυτές τις εφαρμογές:

- είναι εύκολο να μάθει και να διαβάσει ο κάθε προγραμματιστής τη σύνταξη του προγράμματος. Αυτό καθιστά τον κωδικό εντοπισμού σφαλμάτων πιο προσιτό και ταχύτερο.
- είναι σχετικά σύντομα σε σύγκριση με άλλες γλώσσες, γεγονός που μειώνει τον χρόνο που απαιτείται για την ολοκλήρωση τους.
- είναι μια ισχυρή γλώσσα προγραμματισμού που επιτρέπει στους προγραμματιστές να δημιουργούν εξελιγμένα συστήματα λογισμικού.

- η γλώσσα είναι γρήγορη, αποτελεσματική και εύκολη στην εκμάθηση, καθιστώντας την δημοφιλή επιλογή για πολλές εφαρμογές.

- είναι επίσης φορητό, που σημαίνει ότι τα προγράμματα γραμμένα στο C μπορούν εύκολα να μεταφερθούν σε άλλες πλατφόρμες.

- έχει πολλές βιβλιοθήκες και εργαλεία που διευκολύνουν τη χρήση του.[5]



Τα χαρακτηριστικά της γλώσσας C [6]

Πως μπορεί να μάθει κάποιος την γλώσσα c;

Σε περίπτωση που θέλει να μάθει κάποιος τον προγραμματισμό C, υπάρχουν μερικά πράγματα που πρέπει να έχει κατά νου.

1. Πρώτον, η εύρεση ενός καλού πόρου για να διδάξει σε κάποιον αρχάριο τα βασικά της γλώσσας είναι απαραίτητη. Μόλις έχει ένα σταθερό θεμέλιο, μπορεί να αρχίσει να ασκείτε γράφοντας μικρά προγράμματα.

2. Η συμμετοχή σε ηλεκτρονικά φόρουμ ή κοινότητες αφιερωμένες στον προγραμματισμό C είναι επίσης χρήσιμη, όπως μπορεί να μάθει κάποιος αρχάριος από άλλους που αγωνίζονται με τη γλώσσα.

3. Τέλος, δεν πειράζει να ζητήσει κάποιος αρχάριος βοήθεια όταν κολλήσει. Πολλοί άνθρωποι θα βοηθήσουν τους αρχάριους να μάθουν τον προγραμματισμό C. [7]

Οι μελλοντικές προοπτικές της γλώσσας C

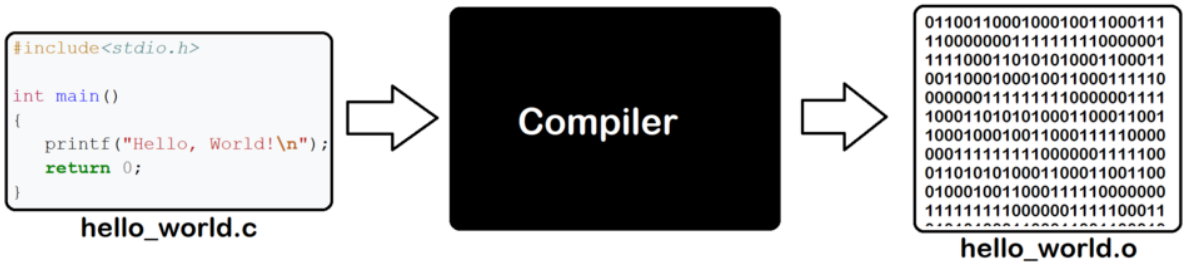
Παρά την άνοδο των σύγχρονων γλωσσών προγραμματισμού, οι προγραμματιστές C έχουν ισχυρές προοπτικές σταδιοδρομίας λόγω της αποτελεσματικότητας, της φορητότητας (portability) και της βαθιάς ενσωμάτωσης της γλώσσας σε προγραμματισμό σε επίπεδο συστήματος.

Η C παραμένει το θεμέλιο των λειτουργικών συστημάτων, τα ενσωματωμένα συστήματα, η ασφάλεια στον κυβερνοχώρο και ο υπολογιστής υψηλής απόδοσης. Με τις βιομηχανίες που βασίζονται σε συστήματα σε πραγματικό χρόνο, το IoT και το λογισμικό αυτοκινήτων, οι ειδικευμένοι προγραμματιστές C είναι σε μεγάλη ζήτηση.

Ως τεχνολογική πρόοδο, οι προγραμματιστές C θα διαδραματίσουν καθοριστικό ρόλο στον προγραμματισμό χαμηλού επιπέδου, στην ανάπτυξη του πυρήνα, στη μηχανική υλικολογισμικού και στις εφαρμογές στον κυβερνοχώρο. [8]

1.3. Compiler

Τι είναι το Compiler και ποιες είναι οι λειτουργίες του;



Σχηματικό της διαδικασίας μεταγλωττισμού [9]

Compiler ονομάζεται ένα πρόγραμμα υπολογιστή που διαβάζει ένα κώδικα γραμμένο σε μια γλώσσα προγραμματισμού και τον μεταφράζει σε ισοδύναμο κώδικα σε μια άλλη γλώσσα προγραμματισμού.

Το κείμενο της εισόδου ονομάζεται source code, ενώ η έξοδος του προγράμματος ονομάζεται object code.

Ο όρος «μεταγλωττιστής» χρησιμοποιείται κυρίως για προγράμματα που μεταφράζουν μια γλώσσα προγραμματισμού υψηλού επιπέδου σε μια γλώσσα χαμηλότερου επιπέδου.

Ένας μεταγλωττιστής μπορεί να περιλαμβάνει οποιαδήποτε από τις εξής λειτουργίες:

- λεκτική ανάλυση,
- προεπεξεργασία,
- συντακτική ανάλυση,
- σημασιολογική ανάλυση,
- παραγωγή κώδικα
- και βελτιστοποίηση κώδικα.

Επειδή τα σφάλματα των προγραμμάτων που προκύπτουν από λανθασμένη μεταγλώττιση είναι πολύ δύσκολο να εντοπιστούν και να διορθωθούν για αυτόν τον λόγο

οι κατασκευαστές των «μεταγλωττιστών» κάνουν σημαντικές προσπάθειες για να βεβαιώσουν την ορθότητα λειτουργίας του λογισμικού τους.[10]

Βήματα εκτέλεσης του Compiler

Οι μεταγλωττιστές ποικίλλουν στις μεθόδους που χρησιμοποιούν για την ανάλυση και τη μετατροπή του πηγαίου κώδικα στον κωδικό εξόδου.[11] Παρά τις διαφορές τους, συνήθως εκτελούν τα ακόλουθα βήματα:

1. **Λεξική ανάλυση:** Ο μεταγλωττιστής χωρίζει τον πηγαίο κώδικα σε λεξικά, τα οποία είναι μεμονωμένα θραύσματα κώδικα που αντιπροσωπεύουν συγκεκριμένα πρότυπα στον κώδικα. Στη συνέχεια, τα λεξικά παρασκευάζονται κατά την προετοιμασία για σύνταξη και σημασιολογικές αναλύσεις.

2. **Ανάλυση σύνταξης:** Ο μεταγλωττιστής επαληθεύει ότι η σύνταξη του κώδικα είναι σωστή, με βάση τους κανόνες για τη γλώσσα προέλευσης. Αυτή η διαδικασία αναφέρεται επίσης ως ανάλυση. Κατά τη διάρκεια αυτού του βήματος, ο μεταγλωττιστής συνήθως δημιουργεί αφηρημένα δέντρα σύνταξης που αντιπροσωπεύουν τις λογικές δομές συγκεκριμένων στοιχείων κώδικα.

3. **Σημασιολογική ανάλυση:** Ο μεταγλωττιστής επαληθεύει την εγκυρότητα της λογικής του κώδικα. Αυτό το βήμα ξεπερνά την ανάλυση σύνταξης επικυρώνοντας την ακρίβεια του κώδικα. Για παράδειγμα, η σημασιολογική ανάλυση μπορεί να ελέγξει εάν οι μεταβλητές έχουν εκχωρηθεί οι σωστοί τύποι ή έχουν δηλωθεί σωστά.

4. **Δημιουργία κώδικα IR:** Αφού περάσει ο κώδικας και στις τρεις φάσεις ανάλυσης, ο μεταγλωττιστής δημιουργεί intermediate representation (IR) δηλαδή μια ενδιάμεση αναπαράσταση του πηγαίου κώδικα. Ο κώδικας IR διευκολύνει τη μετάφραση του πηγαίου κώδικα σε διαφορετική μορφή.

5. **Βελτιστοποίηση:** Ο μεταγλωττιστής βελτιστοποιεί τον κώδικα IR κατά την προετοιμασία για την τελική παραγωγή κώδικα. Ο τύπος και η έκταση της βελτιστοποίησης εξαρτάται από τον μεταγλωττιστή. Μερικοί μεταγλωττιστές επιτρέπουν στους χρήστες να διαμορφώσουν το βαθμό βελτιστοποίησης.

6. **Δημιουργία κώδικα εξόδου:** Ο μεταγλωττιστής δημιουργεί τον τελικό κωδικό εξόδου, χρησιμοποιώντας τον βελτιστοποιημένο κώδικα IR.

Τύποι μεταγλωττιστών

1. **Μεταγλωττιστές μονής διέλευσης:** Οι μεταγλωττιστές μονής διέλευσης περνούν από τον πηγαίο κώδικα μόνο μία φορά. Μπορούν να κατανοήσουν και να το ερμηνεύσουν σε μια σάρωση ολόκληρο το πρόγραμμα. Αυτό είναι γενικά ταχύτερο, αν και δεν είναι πάντα βελτιστοποιημένο για τον αριθμό των επαναλήψεων.

2. **Μεταγλωττιστές πολλαπλής διέλευσης:** Αυτοί οι μεταγλωττιστές περνούν από τον πηγαίο κώδικα αρκετές φορές και στη συνέχεια καταρτίζουν εξαιρετικά αποτελεσματικό κώδικα μηχανής. Παρόλο που αυτοί οι μεταγλωττιστές είναι βραδύτεροι, προσφέρουν αυξημένη αποτελεσματικότητα στην εκτέλεση κώδικα και επομένως είναι κατάλληλες για εφαρμογές μεγάλης κλίμακας.

3. **Cross-compiler:** Είναι ένας μεταγλωττιστής που καταρτίζει κώδικα σε μια πλατφόρμα ή ένα σύστημα και δημιουργεί κώδικα για διαφορετική πλατφόρμα ή σύστημα, όπως το μηχάνημα προορισμού. Αυτό είναι ιδιαίτερα χρήσιμο στην ανάπτυξη των ενσωματωμένων συστημάτων. Είναι επίσης χρήσιμο όταν εργάζεται κάποιος προγραμματιστής σε μια νέα πλατφόρμα και θέλει να διασφαλίσει ότι ο κωδικός σας λειτουργεί στη νέα πλατφόρμα.

4. **Μεταγλωττιστής πηγής προς πηγής:** Ο μεταγλωττιστής πηγής-πηγής είναι ένα εργαλείο λογισμικού που μεταφράζει τον πηγαίο κώδικα σε εκτελέσιμο κώδικα. Είναι χρήσιμο για τη μετάφραση πηγαίου κώδικα γραμμένου σε διαφορετικές γλώσσες προγραμματισμού. Η διαδικασία μετάφρασης εκτελείται συνήθως από έναν μεταγλωττιστή, ο οποίος μεταφράζει τον πηγαίο κώδικα σε έναν συγκεκριμένο τύπο κώδικα μηχανής. Ο κώδικας μηχανής που δημιουργείται στη συνέχεια εκτελείται από το μηχάνημα προορισμού. Η διαδικασία μετάφρασης μπορεί να γίνει χειροκίνητα ή αυτόματα. Το εργαλείο λογισμικού μπορεί να χρησιμοποιηθεί για τη μετάφραση πηγαίου κώδικα γραμμένου σε διαφορετικές γλώσσες προγραμματισμού, όπως C ++, Java και C#.

5. **JIT (Just-In-Time) Μεταγλωττιστές**: Γλώσσες όπως οι Java και C# χρησιμοποιούν τους μεταγλωττιστές JIT που καταρτίζουν τον κωδικό κατά τη στιγμή που θα εκτελεστεί. Μετατρέπουν τον ενδιάμεσο κώδικα, όπως το Java bytecode, στον κώδικα μηχανής, όπου πρέπει να μετρήσουν την ταχύτητα και τη βελτιστοποίηση του κώδικα.Το κύριο πλεονέκτημα του μεταγλωττιστή JIT είναι ότι σας επιτρέπει να μειώσετε το μέγεθος του προγράμματος σας εξαλείφοντας τον περιττό κώδικα. Αυτό σημαίνει ότι το πρόγραμμά σας θα είναι μικρότερο και πιο αποτελεσματικό. Ένα άλλο πλεονέκτημα του μεταγλωττιστή JIT είναι ότι μπορεί να βελτιστοποιήσει το πρόγραμμά σας για διαφορετικές πλατφόρμες. Για παράδειγμα,

εάν αναπτύξετε μια εφαρμογή για κινητά, μπορείτε να χρησιμοποιήσετε τον μεταγλωττιστή JIT για να βελτιστοποιήσετε τον κώδικα για διαφορετικές συσκευές. Ο μεταγλωττιστής JIT έχει επίσης πολλά άλλα οφέλη, όπως μειωμένη χρήση μνήμης, βελτιωμένη απόδοση και αυξημένη αξιοπιστία.

6. **BYTECODE Compiler:** Ένας μεταγλωττιστής bytecode είναι ένα εργαλείο λογισμικού που μεταφράζει μια γλώσσα υψηλού επιπέδου (όπως Java) στον κώδικα μηχανής. Στη συνέχεια, ο κώδικας που προκύπτει καταρτίζεται στον κώδικα μηχανής, ο οποίος μπορεί να εκτελεστεί σε μια μηχανή προορισμού. Το κύριο πλεονέκτημα των μεταγλωττιστών ByTecode είναι ότι επιτρέπουν στους προγραμματιστές να γράφουν κώδικα σε μια γλώσσα υψηλού επιπέδου και στη συνέχεια να τον καταρρίψουν στον κώδικα μηχανής. Αυτό επιτρέπει στους προγραμματιστές να γράφουν κώδικα που είναι πιο συνοπτικό και πιο κατανοητό. Το κύριο μειονέκτημα των μεταγλωττιστών του Bytecode είναι ότι πρέπει να γράφονται σε γλώσσα υψηλού επιπέδου. Αυτό σημαίνει ότι δεν είναι κατάλληλα για την ανάπτυξη κώδικα χαμηλού επιπέδου. Οι μεταγλωττιστές bytecode χρησιμοποιούνται συνήθως κατά την ανάπτυξη εφαρμογών για κινητά.

7. **Μεταγλωττιστές υλικού (Hardware compilers):** Οι μεταγλωττιστές υλικού είναι εργαλεία λογισμικού που καταρτίζουν τον πηγαίο κώδικα στον κώδικα μηχανής. Χρησιμοποιούνται για να μετατρέψουν τον πηγαίο κώδικα σε κώδικα μηχανής, ο οποίος στη συνέχεια εκτελείται από έναν υπολογιστή. Χρησιμοποιούνται σε μια μεγάλη ποικιλία εφαρμογών, συμπεριλαμβανομένων των παιχνιδιών υπολογιστών, των ενσωματωμένων συστημάτων και των λειτουργικών συστημάτων. Ο πιο συνηθισμένος τύπος μεταγλωττιστή υλικού είναι ο συναρμολογητής. Μετατρέπει τον πηγαίο κώδικα σε κώδικα μηχανής, ο οποίος στη συνέχεια εκτελείται από τον υπολογιστή. Ο συναρμολογητής γράφεται συνήθως σε γλώσσα υψηλού επιπέδου (όπως C ή C ++) και χρησιμοποιείται για τη συναρμολόγηση προγραμμάτων που γράφονται σε γλώσσες χαμηλού επιπέδου (όπως η γλώσσα συναρμολόγησης). Άλλοι τύποι μεταγλωττιστών υλικού περιλαμβάνουν τον σύνδεσμο, ο οποίος μεταφράζει τον πηγαίο κώδικα στον κώδικα μηχανής. και ο φορτωτής, ο οποίος μεταφράζει τον κωδικό μηχανής σε πηγαίο κώδικα. Ο σύνδεσμος και ο φορτωτής χρησιμοποιούνται για τη δημιουργία δυναμικών συνδέσεων, οι οποίοι επιτρέπουν τη σύνδεση των προγραμμάτων. Οι μεταγλωττιστές υλικού χρησιμοποιούνται συχνά για τη μετάφραση πηγαίου κώδικα στον κώδικα μηχανής. Αυτή η διαδικασία μετάφρασης ονομάζεται **συλλογή**. Η συλλογή μετατρέπει τον πηγαίο κώδικα σε κώδικα μηχανής, ο οποίος στη συνέχεια εκτελείται από τον υπολογιστή. Ο μεταγλωττιστής είναι συνήθως γραμμένος σε γλώσσα υψηλού επιπέδου (όπως C ή C ++) και χρησιμοποιείται για την κατάρτιση προγραμμάτων που γράφονται σε γλώσσες χαμηλού επιπέδου (όπως η γλώσσα συναρμολόγησης).

8. **Δυαδικός Μεταγλωττιστής (Binary Compiler):** Ένας δυαδικός μεταγλωττιστής είναι ένα εργαλείο λογισμικού που μεταφράζει ένα αρχείο πηγαίου κώδικα σε δυαδική μορφή. Η δυαδική μορφή χρησιμοποιείται για την αποθήκευση των πληροφοριών του προγράμματος σε μια συμπαγή μορφή που μπορεί εύκολα να διαβαστεί και να κατανοηθεί από έναν υπολογιστή. Η δυαδική μορφή είναι επίσης γνωστή ως κώδικας μηχανής, κωδικός συναρμολόγησης ή γλώσσα μηχανής. Πρόκειται για μια γλώσσα προγραμματισμού χαμηλού επιπέδου που χρησιμοποιείται για τον έλεγχο του υλικού και του λογισμικού του μηχανήματος. Οι δυαδικοί μεταγλωττιστές χρησιμοποιούνται σε πολλές διαφορετικές καταστάσεις, συμπεριλαμβανομένης της ανάπτυξης ιστού, της διαχείρισης βάσεων δεδομένων και του προγραμματισμού δικτύου. [12]

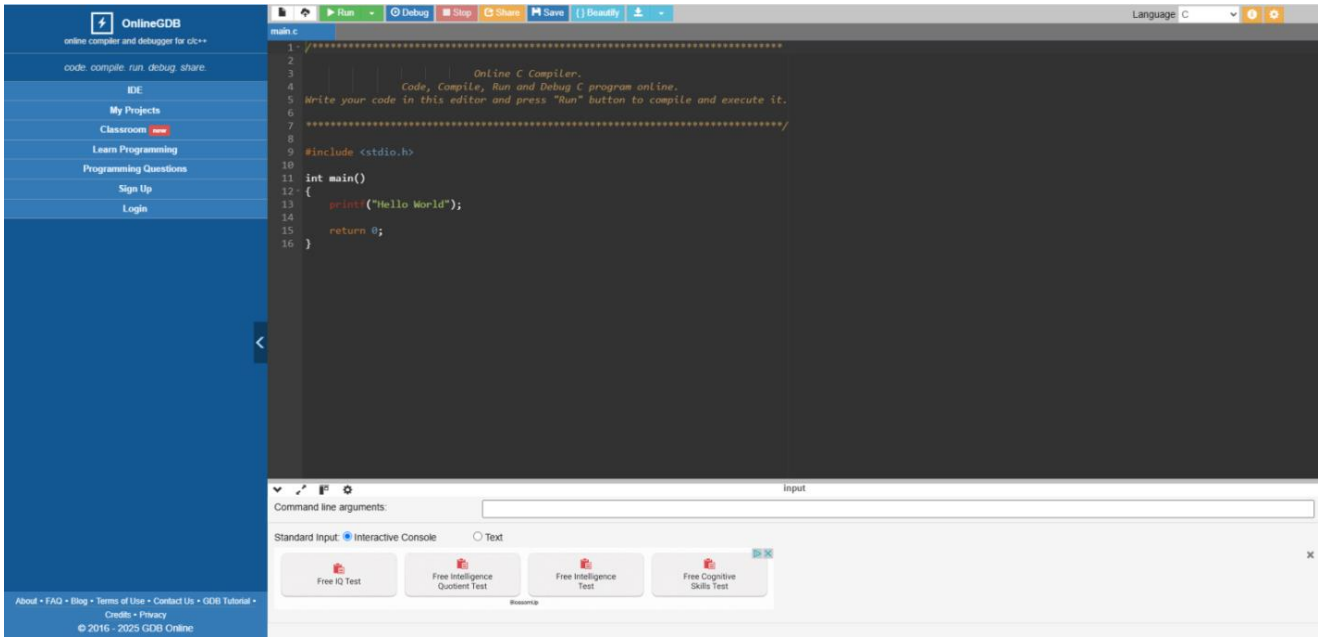
Πλεονεκτήματα του Compiler

Τα πλεονεκτήματα που έχει ένας μεταγλωττιστής

- **Ταχύτερη εκτέλεση:** Ο συντάκτης κώδικα γενικά λειτουργεί ταχύτερα από τον ερμηνευμένο κώδικα.
- **Καλύτερη βελτιστοποίηση απόδοσης:** Οι μεταγλωττιστές μπορούν να εκτελέσουν προηγμένες βελτιστοποιήσεις κατά τη διάρκεια της διαδικασίας συλλογής.
- **Βελτιωμένη ασφάλεια:** Ο πηγαίος κώδικας δεν είναι άμεσα προσβάσιμος σε προγράμματα που έχουν μεταγλωττιστεί, προσφέροντας κάποια προστασία πνευματικής ιδιοκτησίας.
- **Πρώιμη ανίχνευση σφαλμάτων:** Πολλά σφάλματα μπορούν να πιαστούν κατά τη διάρκεια της μεταγλωττιστικής χρονικής περιόδου πριν από την εκτέλεση του προγράμματος.
- **Πιο αποτελεσματική χρήση μνήμης:** Τα προγράμματα που έχουν μεταγλωττίσει συχνά έχουν καλύτερη διαχείριση μνήμης.
- **Βελτιστοποιήσεις που σχετίζονται με την πλατφόρμα:** Οι μεταγλωττιστές μπορούν να δημιουργήσουν κώδικα βελτιστοποιημένα για συγκεκριμένες αρχιτεκτονικές υλικού.
- **Μειωμένη έξοδο χρόνου εκτέλεσης:** Δεν υπάρχει ανάγκη για διερμηνέα κατά την εκτέλεση του προγράμματος.

- **Υποστήριξη για τη στατική πληκτρολόγηση:** Επιτρέπει τον έλεγχο τύπου κατά τη μεταγλώττιση, τη σύλληψη πιθανών σφαλμάτων νωρίς.[13]

Το Compiler που έτρεξα τον κώδικα είναι onlineGDB (https://www.onlinegdb.com/online_c_compiler)



Screenshot από το online gdb

1.4. Πρόγραμμα Sudoku

Τι είναι το sudoku και πότε εμφανίστηκε;

9	1	3				5		
6		7					2	4
	5			8			7	
	7	9						
		2		9			4	3
					4		9	
	4				1	9		
7		6			9			5
		1			6	4		7

Πίνακας Sudoku [14]

Το **Sudoku** είναι ένα παζλ που βασίζεται στη λογική. Στόχος είναι να συμπληρωθούν όλα τα κουτάκια στον πίνακα (9x9), ώστε κάθε στήλη, κάθε σειρά και κάθε κουτάκι 3x3 να περιέχουν όλα τα ψηφία από το 1 μέχρι το 9. Μερικά κουτάκια είναι ήδη συμπληρωμένα, ώστε να υπάρχει μόνο μία δυνατή λύση.

Εμφανίστηκε για πρώτη φορά στην Ιαπωνία το 1984, όπου του δόθηκε το όνομα "Sudoku", το οποίο είναι συντομογραφία μιας μεγαλύτερης έκφρασης στα ιαπωνικά - "Sūjib we douching in kangaroo" που σημαίνει, "τα ψηφία περιορίζονται σε ένα περιστατικό". Το Sudoku συνεχίζει να είναι ιδιαίτερα δημοφιλές στην Ιαπωνία, όπου οι άνθρωποι αγοράζουν πάνω από 600.000 περιοδικά Sudoku το μήνα.[15]

Κανόνες και τρόποι επίλυσης

Οι κανόνες για το Sudoku είναι απλοί. πρέπει ο παίκτης να συμπληρώσει ένα τετράγωνο 9 × 9 με αριθμούς από 1-9 χωρίς επαναλαμβανόμενους αριθμούς σε κάθε γραμμή, οριζόντια ή κάθετα. Για να δυσκολέψει τον παίκτη υπάρχουν 3 × 3 τετράγωνα που συμπληρωμένα στο πλέγμα και κάθε ένα από αυτά τα τετράγωνα δεν μπορεί να έχει επαναλαμβανόμενους αριθμούς. Τα παζλ θα έχουν κάποια κουτάκια συμπληρωμένα για να ξεκινήσει ο παίκτης να βάζει τους υπόλοιπους αριθμούς και όσο λιγότεροι αριθμοί είναι συμπληρωμένοι, τόσο πιο δύσκολο είναι το παζλ.

Πολλά παζλ θα αφήσουν ένα ή δύο κενά είτε σε μια γραμμή είτε σε ένα κουτί. Μια γραμμή μπορεί να λείπουν , π.χ. οι αριθμοί 5 και 6, αλλά ένα κουτί κατά μήκος αυτής της γραμμής θα έχει ήδη 6 σε αυτό, έτσι ώστε να γνωρίζετε ότι αυτό το κενό πρέπει να είναι 5, και το άλλο κενό πρέπει να είναι το 6. Το ίδιο μπορεί να συμβαίνει για έναν ή δύο αριθμούς που λείπουν στα μικρότερα κουτιά. Η συμπλήρωση αυτών των κενών θα οδηγήσει τον παίκτη σε μια λύση.

Καθώς συμπληρώνουμε τις εύκολες λύσεις, μπορούμε να αρχίσουμε να βρίσκουμε τους άλλους αριθμούς που λείπουν που είναι εύκολο να τοποθετηθούν. Στο παραπάνω παράδειγμα, το κουτί έχει τώρα 5, και αυτό μπορεί να σας βοηθήσει να λύσετε την περιοχή 3 × 3 ή μπορεί να σας βοηθήσει να λύσετε τις γραμμές που διασχίζουν αυτήν την περιοχή.[16]

9	1	3	4	2	7	5	8	6
6	8	7	9	1	5	3	2	4
2	5	4	6	8	3	1	7	9
4	7	9	1	3	2	6	5	8
1	6	2	5	9	8	7	4	3
5	3	8	7	6	4	2	9	1
3	4	5	8	7	1	9	6	2
7	2	6	3	4	9	8	1	5
8	9	1	2	5	6	4	3	7

Συμπληρωμένος πίνακας Sudoku [17]

Μαθηματικές ιδιότητες του Sudoku

Τα μαθηματικά μπορούν να χρησιμοποιηθούν για να μελετήσουν το Sudoku για να απαντήσουν σε ερωτήσεις όπως:

- "Πόσα πλέγματα (grids) sudoku είναι συμπληρωμένα;",
- "Ποιος είναι ο ελάχιστος αριθμός των ενδείξεων σε ένα έγκυρο παζλ;"
- και "Με ποιους τρόπους μπορούν να είναι συμμετρικά τα πλέγματα sudoku;"

μέσω της χρήσης των συνδυαστικών στοιχείων (combinatorics) και της θεωρίας των ομάδων (group theory).

Τα συνδυαστικά στοιχεία χρησιμοποιούνται για την καταμέτρηση των έγκυρων πλεγμάτων Sudoku και η θεωρία των ομάδων χρησιμοποιείτε για να περιγράψουν ιδέες για το πότε δύο πλέγματα είναι ισοδύναμα. [18]

			8					
4				1	5		3	
	2	9		4		5	1	8
	4					1	2	
			6		2			
	3	2					9	
6	9	3		5		8	7	
	5		4	8				1
					3			

3	1	5	8	2	7	9	4	6
4	6	8	9	1	5	7	3	2
7	2	9	3	4	6	5	1	8
9	4	6	5	3	8	1	2	7
5	7	1	6	9	2	4	8	3
8	3	2	1	7	4	6	9	5
6	9	3	2	5	1	8	7	4
2	5	7	4	8	9	3	6	1
1	8	4	7	6	3	2	5	9

Παράδειγμα για την λύση του πίνακα Sudoku [19]

NP-Complete

Στη θεωρία υπολογιστικής πολυπλοκότητας, ένα πρόβλημα είναι το NP-Complete όταν:

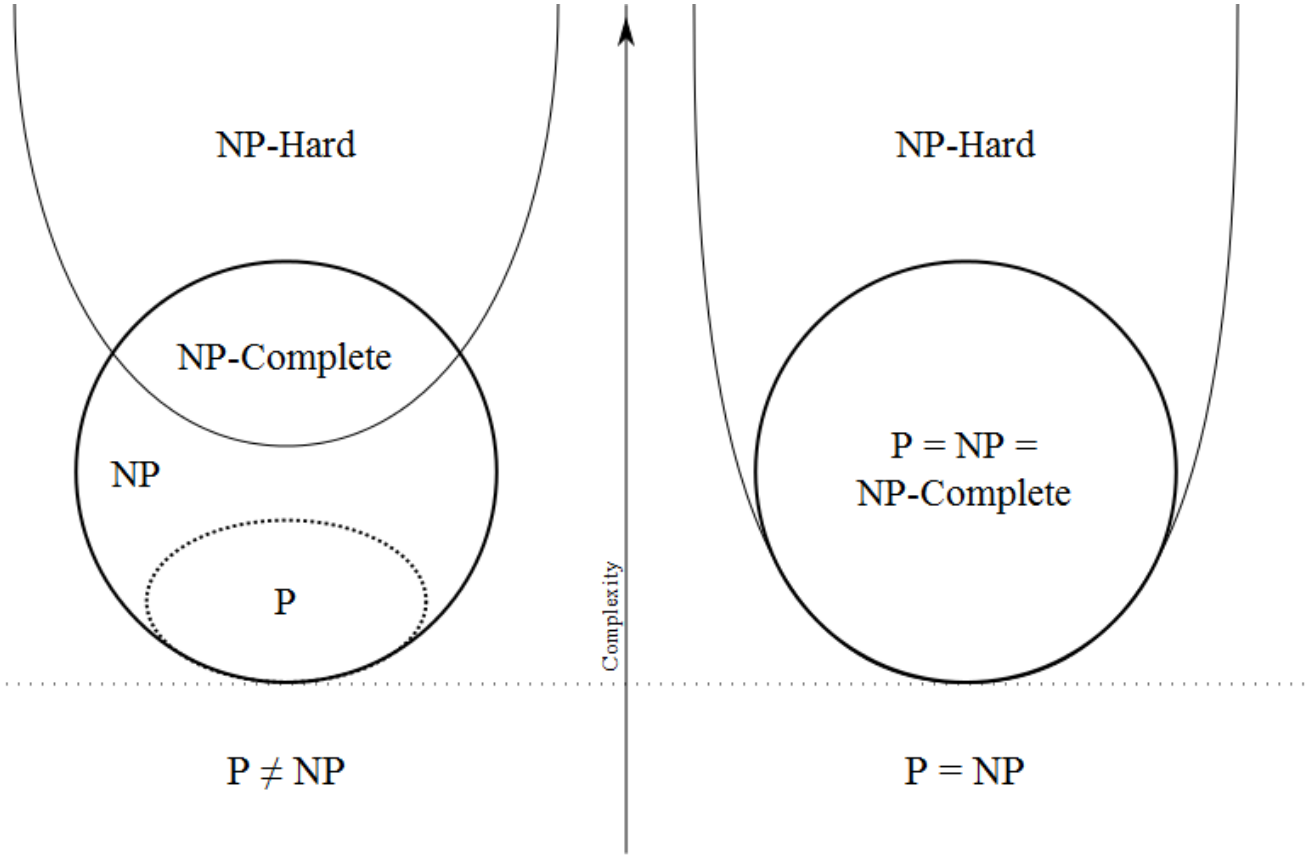
- Είναι ένα πρόβλημα απόφασης, που σημαίνει ότι για οποιαδήποτε εισροή στο πρόβλημα, η έξοδος είναι είτε "ναι" είτε "όχι".
- Όταν η απάντηση είναι "ναι", αυτό μπορεί να αποδειχθεί μέσω της ύπαρξης ενός μικρού (πολυωνυμικού μήκους) λύσης.
- Η ορθότητα κάθε λύσης μπορεί να επαληθευτεί γρήγορα (δηλαδή, σε πολυωνυμικό χρόνο) και ένας αλγόριθμος αναζήτησης βίαιης δύναμης μπορεί να βρει μια λύση δοκιμάζοντας όλες τις πιθανές λύσεις.
- Το πρόβλημα μπορεί να χρησιμοποιηθεί για την προσομοίωση κάθε άλλου προβλήματος για το οποίο μπορούμε να επαληθεύσουμε γρήγορα ότι μια λύση είναι σωστή. Με αυτή την έννοια, τα πλήρη προβλήματα NP είναι τα πιο δύσκολα από τα προβλήματα στα οποία μπορούν να επαληθευτούν γρήγορα λύσεις. Εάν μπορούσαμε να βρούμε γρήγορα λύσεις κάποιου προβλήματος NP-Complete πλήρως, θα μπορούσαμε να βρούμε γρήγορα τις λύσεις κάθε άλλου προβλήματος στις οποίες μπορεί εύκολα να επαληθευτεί μια δεδομένη λύση.

Το όνομα "NP-Complete" είναι σύντομο για το "Nondeterministic Polynomial-Time Completh".

Σε αυτό το όνομα, το "nondeterministic" αναφέρεται σε μη-nondeterministic Ture Machines, έναν τρόπο μαθηματικής επισημοποίησης της ιδέας ενός αλγόριθμου αναζήτησης βίαιης δύναμης. Ο πολυώνυμος χρόνος αναφέρεται σε ένα χρονικό διάστημα που θεωρείται "γρήγορος" για έναν ντετερμινιστικό αλγόριθμο για να ελέγξει μια μόνο λύση ή για μια μη μηχάνημα Turing για να εκτελέσει ολόκληρη την αναζήτηση. Το "Complete" αναφέρεται στην ιδιοκτησία του να είναι σε θέση να προσομοιώσει τα πάντα στην ίδια τάξη πολυπλοκότητας.

Πιο συγκεκριμένα, κάθε εισροή στο πρόβλημα θα πρέπει να συσχετίζεται με ένα σύνολο διαλυμάτων πολυωνυμικού μήκους, η εγκυρότητα του καθενός από τα οποία μπορεί να δοκιμαστεί γρήγορα (σε χρόνο πολυωνύμου) έτσι ώστε η έξοδος για οποιαδήποτε είσοδο να είναι "ναι" εάν η λύση είναι μη έντονη και όχι "αν είναι κενή.

Η τάξη πολυπλοκότητας των προβλημάτων αυτής της μορφής ονομάζεται NP, μια συντομογραφία για "nondeterministic πολυωνυμικό χρόνο". Ένα πρόβλημα λέγεται ότι είναι NP-Hard εάν όλα σε NP μπορούν να μετατραπούν σε πολυωνυμικό χρόνο σε αυτό, παρόλο που μπορεί να μην είναι σε NP. Ένα πρόβλημα είναι το NP-πλήρες εάν είναι τόσο σε NP και NP-HARD. Τα προβλήματα NP-Complete αντιπροσωπεύουν τα πιο δύσκολα προβλήματα στο NP. [20]



Διάγραμμα Euler για P, NP, NP-Complete και NP-Hard για το σύνολο των προβλημάτων [21]

Brute Force

Μερικοί χομπιστές έχουν αναπτύξει προγράμματα υπολογιστών που θα λύσουν παζλ Sudoku χρησιμοποιώντας έναν αλγόριθμο backtracking, ο οποίος είναι ένας τύπος αναλυτικής αναζήτησης brute force. Αν και έχει διαπιστωθεί ότι υπάρχουν περίπου 5.96 x 10^{26} τελικά πλέγματα, ένας αλγόριθμος βίαιης δύναμης μπορεί να είναι μια πρακτική μέθοδος για την επίλυση παζλ sudoku.

Ένας αλγόριθμος βίαιης δύναμης επισκέπτεται τα κενά κελιά με κάποια σειρά, γεμίζοντας τα ψηφία διαδοχικά ή backtracking όταν ο αριθμός διαπιστώνεται ότι δεν είναι έγκυρος.

Εν συντομία, ένα πρόγραμμα θα λύσει ένα παζλ τοποθετώντας το ψηφίο "1" στο πρώτο κελί και ελέγχοντας εάν επιτρέπεται να είναι εκεί. Εάν δεν υπάρχουν παραβιάσεις (έλεγχος σειράς, στήλης και περιορισμών κουτιού), τότε ο αλγόριθμος προχωράει στο επόμενο κύτταρο και τοποθετεί ένα "1" σε αυτό το κελί. Κατά τον έλεγχο των παραβιάσεων, εάν ανακαλυφθεί ότι δεν επιτρέπεται το "1", η τιμή προωθείται στο "2". Εάν ανακαλυφθεί ένα κελί όπου δεν επιτρέπεται κανένα από τα 9 ψηφία, τότε ο αλγόριθμος αφήνει αυτό το κυτταρικό κενό και μετακινείται πίσω στο προηγούμενο κελί. Η τιμή σε αυτό το κελί στη συνέχεια αυξάνεται κατά μία. Αυτό επαναλαμβάνεται μέχρι να ανακαλυφθεί η επιτρεπόμενη τιμή στο τελευταίο (81st) κύτταρο.

Τα πλεονεκτήματα αυτής της μεθόδου είναι:

- Μια λύση είναι εγγυημένη (εφόσον το παζλ είναι έγκυρο).
- Η επίλυση του χρόνου δεν σχετίζεται κυρίως με το βαθμό δυσκολίας.
- Ο αλγόριθμος είναι απλούστερος από άλλους αλγόριθμους, ειδικά σε σύγκριση με ισχυρούς αλγόριθμους που εξασφαλίζουν μια λύση στα πιο δύσκολα παζλ.

Το μειονέκτημα αυτής της μεθόδου είναι ότι ο χρόνος επίλυσης μπορεί να είναι αργός σε σύγκριση με τους αλγόριθμους που διαμορφώνονται μετά από παραπλανητικές μεθόδους.[22]

Algorithm X

Ο αλγόριθμος X (Algorithm X) είναι ένας αλγόριθμος για την επίλυση του ακριβούς προβλήματος κάλυψης. Πρόκειται για έναν απλό αναδρομικό, μη-nondeterministic, βάθος-πρώτο, backtracking αλγόριθμο που χρησιμοποιείται από τον Donald Knuth για να επιδείξει μια αποτελεσματική εφαρμογή που ονομάζεται DLX, η οποία χρησιμοποιεί την τεχνική Dancing Links.[23]

Το ακριβές πρόβλημα κάλυψης αντιπροσωπεύεται στον αλγόριθμο X από έναν πίνακα επιπτώσεων A που αποτελείται από 0s και 1s. Ο στόχος είναι να επιλέξετε ένα υποσύνολο των σειρών έτσι ώστε το ψηφίο 1 να εμφανίζεται σε κάθε στήλη ακριβώς μία φορά.

Ο αλγόριθμος X λειτουργεί ως εξής:

- Εάν η μήτρα (matrix) A δεν έχει στήλες, η τρέχουσα μερική λύση είναι μια έγκυρη λύση. τερματίστε με επιτυχία.
- Διαφορετικά, επιλέξτε μια στήλη C(deterministically).
- Επιλέγουμε μια σειρά r έτσι ώστε AR, C = 1(nondeterministically).
- Συμπεριλάβετε τη σειρά R στη μερική λύση.
- Για κάθε στήλη j έτσι ώστε ar, j = 1,
- Για κάθε σειρά εγώ έτσι ώστε ai, j = 1,
- Διαγράφουμε τη σειρά I από το Matrix A.
- Διαγράφουμε τη στήλη J από το Matrix A.
- Επαναλαμβάνουμε αυτόν τον αλγόριθμο αναδρομικά στη μειωμένη μήτρα A.

Η nondeterministic επιλογή του R σημαίνει ότι ο αλγόριθμος επαναλαμβάνεται πάνω από ανεξάρτητους subalgorithms. Κάθε κληρονομεί την τρέχουσα μήτρα A, αλλά το μειώνει σε σχέση με μια διαφορετική σειρά r. Εάν η στήλη C είναι εντελώς μηδενική, δεν υπάρχουν subalgorithms και η διαδικασία τερματίζεται ανεπιτυχώς.

Οι subalgorithms σχηματίζουν ένα δέντρο αναζήτησης με φυσικό τρόπο, με το αρχικό πρόβλημα στη ρίζα και με το επίπεδο K που περιέχει κάθε subalgorithm που αντιστοιχεί

σε k που επιλέγονται. Το backtracking είναι η διαδικασία διασκέδασης του δέντρου σε προετοιμασία, πρώτα βάθος.

Οποιοσδήποτε συστηματικός κανόνας για την επιλογή της στήλης C σε αυτή τη διαδικασία θα βρει όλες τις λύσεις, αλλά ορισμένοι κανόνες λειτουργούν πολύ καλύτερα από άλλους. Για να μειωθεί ο αριθμός των επαναλήψεων, ο Knuth υποδηλώνει ότι ο αλγόριθμος που επιλέγει τη στήλη επιλέξτε μια στήλη με τον μικρότερο αριθμό 1s σε αυτό.

Τα πλεονεκτήματα που έχει:

- Είναι απλό, δεν χρειάζεται να κατασκευαστεί μια περίπλοκη νέα δομή δεδομένων.
- Είναι ευέλικτο, μπορεί πολύ εύκολα να επεκταθεί για την επίλυση του sudoku.

1.5 Κώδικας του Sudoku στη γλώσσα C

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// Το N είναι το μέγεθος του πίνακα Sudoku
#define N 9

// Συνάρτηση για να ελέγξει εάν είναι ασφαλής η τοποθέτηση ενός αριθμού
int isSafe(int grid[N][N], int row, int col, int num)
{

   // Ελέγχει αν βρίσκει τον ίδιο αριθμό στην παρόμοια σειρά και επιστρέφει στο 0
   for (int x = 0; x <= N; x++) {
      if (grid[row][x] == num) {
         return 0;
      }
   }

   // Ελέγχει αν βρίσκει τον ίδιο αριθμό στην παρόμοια στήλη και επιστρέφει στο 0
   for (int x = 0; x <= N; x++) {
      if (grid[x][col] == num) {
         return 0;
      }
```

```
    }

    for (int x = 0; x <= N; x++) {
        if (grid[row - row % 3 + x % 3][col - col % 3 + x % 3] == num) {
            return 0;
        }
    }

    // Ελέγχει αν βρίσκει τον ίδιο αριθμό στον συγκεκριμένο πίνακα 3*3 και επιστρέφει 0
    int startRow = row - row % 3;
    int startCol = col - col % 3;

    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (grid[i + startRow][j + startCol] == num) {
                return 0;
            }
        }
    }
    return 1;
}

// Συνάρτηση για να γεμίσει το πίνακα Sudoku
int fillSudoku(int grid[N][N]) {
    for (int row = 0; row < N; row++) {
```

```
        for (int col = 0; col < N; col++) {
            if (grid[row][col] == 0) {
                for (int num = 1; num <= N; num++) {
                    if (isSafe(grid, row, col, num)) {
                        grid[row][col] = num;
                        if (fillSudoku(grid)) {
                            return 1;
                        }
                        // πίσω αναζήτηση
                        grid[row][col] = 0;
                    }
                }
                // Δεν βρέθηκε αριθμός
                return 0;
            }
        }
    }
    return 1;
}

/* Μια βοηθητική λειτουργία για την εκτύπωση πλέγματος. */
void printBoard(int arr[N][N])
{
    for (int row = 0; row < N; row++)
    {
```

```
        if (row % 3 == 0 && row != 0) {
            // Διαχωριστική γραμμή ανά 3 γραμμές
            printf("---------------------\n");
        }
        for (int col = 0; col < N; col++) {
            if (col % 3 == 0 && col != 0) {
                // Διαφορετικές στήλες ανά 3 στήλες
                printf("| ");
            }
            if (arr[row][col] == 0) {
                printf("0 ");
            } else {
                printf("%d ", arr[row][col]);
            }
        }
        printf("\n");
    }
}

// Συνάρτηση για να αφαιρέσει τυχαία κελιά από τον πίνακα
void removeCells(int grid[N][N], int count) {
    while (count != 0) {
        int i = rand() % N;
        int j = rand() % N;
        if (grid[i][j] != 0) {
```

```
            // Αφαιρούμε τον αριθμό
            grid[i][j] = 0;
            count--;
        }
    }
}


int main()
{
    // Αρχικοποίηση του πίνακα με μηδενικά
    int arr[N][N] = {0};
    int row, col, num;
    char choice;

    // Αρχικοποίηση της γεννήτριας τυχαιότητας
    srand(time(NULL));

    printf("Chart Sudoku:\n");
    // Γέμισμα του πίνακα με τυχαίους αριθμούς
    fillSudoku(arr);

    // Αφαιρέστε τυχαία κελιά από τον πίνακα (δηλαδή, να βάλουμε 0s)
    removeCells(arr, 40); // Αφαιρούμε 40 αριθμούς για να αφήσουμε κενά
```

```
// Εκτύπωση του αρχικού πίνακα
printBoard(arr);

while (1) {
    printf("\nEnter position (row and column) for number (e.g. 1,2) or 'q' to exit: ");
    if (scanf(" %c", &choice) == 1 && choice == 'q') {
        // Έξοδος από το παιχνίδι
        break;
    }

    // Επιστρέφει το χαρακτήρα στην ροή εισόδου
    ungetc(choice, stdin);

    // Εισάγει γραμμή και στήλη
    scanf("%d %d",&row,&col);

    // Μετατροπή σε 0-8
    row -= 1;
    // Μετατροπή σε 0-8
    col -= 1;

    if (row < 0 || row >= N || col < 0 || col >= N) {
        printf("Invalid location! Try again.\n");
    }
```

```
        printf ("\nEnter a number (1-9): ");
        scanf("%d",&num);

        if (num < 1 || num > 9) {
            printf("\nInvalid number! Numbers must be from 1-9.\n");
        }

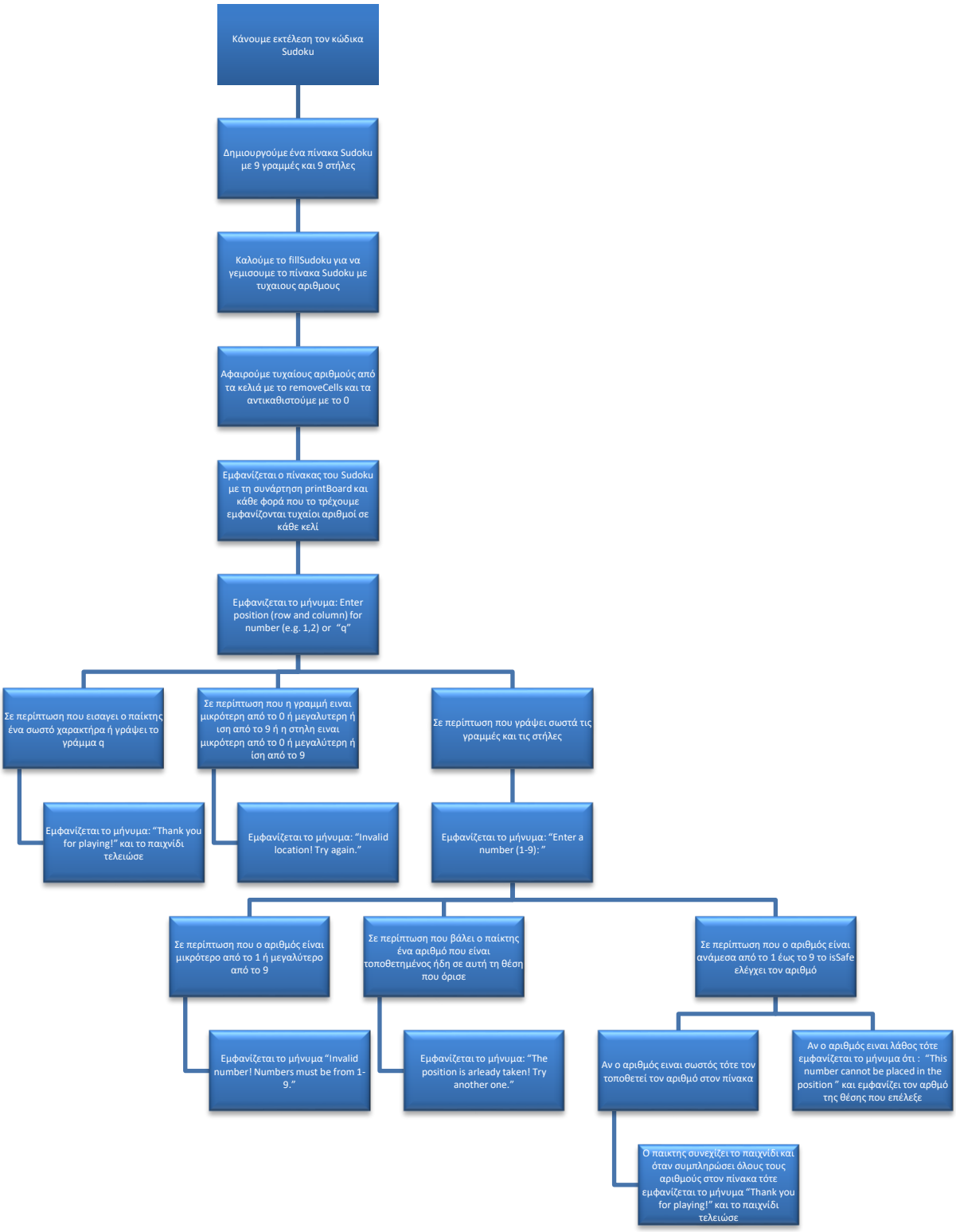
        if (arr[row][col] != 0) {
            printf("\nThe position is already taken! Try another one.\n");
        }

        // Έλεγχος αν τοποθετείται ο αριθμός
        if (isSafe(arr, row, col, num)) {
            // Τοποθέτηση αριθμού
            arr[row][col] = num;
        }
        else {
            printf("\nThis number cannot be placed in the position (%d, %d).\n", row + 1, col
+ 1);
        }

        // Εκτύπωση του πίνακα μετά από κάθε εισαγωγή
        printBoard(arr);
    }
```

```
    printf("\nThank you for playing!\n");
    return 0;
}
```

1.6 Δεντροδιάγραμμα



2. Ανάλυση του Κώδικα

2.1. Περιγραφή του Κώδικα

/*Το stdio.h χρησιμοποιείται συνήθως στον προγραμματισμό C καθώς περιλαμβάνει ένα τυπικό αρχείο κεφαλίδα εισόδου-εξόδου, το οποίο περιέχει βασικές λειτουργίες όπως το printf (), το scanf () και πολλά άλλα.*/

```
#include <stdio.h>
```

/*Το stdlib.h είναι ένα αρχείο κεφαλίδα και επίσης η τυπική βιβλιοθήκη της γλώσσας προγραμματισμού C που δηλώνει διάφορες λειτουργίες χρησιμότητας για μετατροπές τύπου, κατανομή μνήμης, αλγόριθμους και άλλες παρόμοιες περιπτώσεις χρήσεις. Έχει επίσης πολλούς τύπους δεδομένων και μακροεντολές που ορίζονται στην κεφαλίδα.*/

```
#include <stdlib.h>
```

/*Η κεφαλίδα Time.h ορίζει τέσσερις μεταβλητούς τύπους, δύο μακροεντολές και διάφορες λειτουργίες για τον χειρισμό της ημερομηνίας και της ώρας.*/

```
#include <time.h>
```

/*Το #define είναι μια οδηγία προ -επεξεργαστή που χρησιμοποιείται στα προγράμματα C για τον ορισμό των μακροεντολών. Η οδηγία #define χρησιμοποιείται για να δηλώσει σταθερές τιμές ή μια ονομαστική έκφραση που μπορεί να χρησιμοποιηθεί σε όλο το πρόγραμμα C μας. Η εντολή preprocessor #define χρησιμοποιείται στον προγραμματισμό C για τον καθορισμό συμβολικών σταθερών και μακροεντολών.*/

```
// Το N είναι το μέγεθος του πίνακα Sudoku

#define N 9
```

/*Δημιουργούμε μία συνάρτηση με το όνομα isSafe όπου ορίζουμε τις ονομασίες του πίνακα grid, με τις γραμμές (row) , τις στήλες (col) και τον αριθμό (num).*/

// Συνάρτηση για να ελέγξει εάν είναι ασφαλής η τοποθέτηση ενός αριθμού

int isSafe(int grid[N][N], int row, int col, int num)

{

/*Ορίζουμε την επανάληψη for όπου θα κάνει επανάληψη 9 φορές το x. */

// Ελέγχει αν βρίσκει τον ίδιο αριθμό στην παρόμοια σειρά και επιστρέφει στο 0

for (int x = 0; x <= N; x++) {

/*Σε περίπτωση που ο πίνακας grid με στήλη το x είναι ίσος με τον αριθμο num τότε το πρόγραμμα εκτελέστηκε σωστά.*/

if (grid[row][x] == num) {

return 0;

}

}

/* Ορίζουμε την επανάληψη for όπου θα κάνει επανάληψη 9 φορές το x. */

// Ελέγχει αν βρίσκει τον ίδιο αριθμό στην παρόμοια στήλη και επιστρέφει στο 0

for (int x = 0; x <= N; x++) {

/* Σε περίπτωση που ο πίνακας grid με γραμμή το x είναι ίσος με τον αριθμό num τότε το πρόγραμμα εκτελέστηκε σωστά.*/

```
if (grid[x][col] == num) {

return 0;

}

}
```

/*Ορίζουμε την επανάληψη for όπου θα κάνει επανάληψη του x μέχρι το 9.*/

```
for (int x = 0; x <= N; x++) {
```

/*Σε περίπτωση που ο πίνακα grid όπου οι γραμμές είναι η αφαίρεση των γραμμών με το υπόλοιπο της διαίρεση της γραμμής με το 3 και η πρόσθεση του x με το υπόλοιπο της διαίρεσης με το 3 και οι στήλες είναι είναι η αφαίρεση των στηλών με το υπόλοιπο της διαίρεση της στήλης με το 3 και η πρόσθεση του x με το υπόλοιπο της διαίρεσης με το 3 ισούται με τον αριθμό num που επιλέξαμε και το πρόγραμμα εκτελέστηκε σωστά. */

```
if (grid[row - row % 3 + x % 3][col - col % 3 + x % 3] == num) {

return 0;

}

}
```

/*Με το startRow αφαιρουμε τον αριθμό της γραμμής με το υπόλοιπο της διαίρεσης της γραμμής με το 3 και με το startCol αφαιρούμε τον αριθμό της στήλης με το υπόλοιπο της διαιρεσης της στήλης με το 3.*/

```
// Ελέγχει αν βρίσκει τον ίδιο αριθμό στον συγκεκριμένο πίνακα 3*3 και επιστρέφει 0

int startRow = row - row % 3;

int startCol = col - col % 3;


/*Ορίζουμε την επανάληψη for όπου κάνει επανάληψη 3 φορές το i.*/

for (int i = 0; i < 3; i++) {


/*Ορίζουμε την επανάληψη for όπου κάνει επανάληψη 3 φορές το j.*/

for (int j = 0; j < 3; j++) {


/*Σε περίπτωση που στον πίνακα grid όπου οι γραμμές είναι το άθροισμα του i και του startRow και οι στήλες είναι το άθροισμα του j και του startCol είναι ίσο με τον αριθμό num που γράψαμε τότε το πρόγραμμα εκτελέστηκε σωστά.*/

if (grid[i + startRow][j + startCol] == num) {

return 0;

}

}

}

/*Το return 1 σημαίνει ότι το πρόγραμμα δεν εκτελέστηκε σωστά και υπάρχει κάποιο σφάλμα.*/

return 1;

}
```

/*Ορίζουμε την συνάρτηση fillSudoku όπου συμπληρώνει τον πίνακα grid και καλεί τον πίνακα grid.*/

// Συνάρτηση για να γεμίσει το πίνακα Sudoku

int fillSudoku(int grid[N][N]) {

/*Ορίζουμε την επανάληψη 9 φορές τις γραμμές.*/

for (int row = 0; row < N; row++) {

/*Ορίζουμε την επανάληψη 9 φορές τις στήλες.*/

for (int col = 0; col < N; col++) {

/*Σε περίπτωση που ο πίνακας grid είναι ίσο με το 0 κάνουμε επανάληψη με το for 9 φορές τον αριθμό num. */

if (grid[row][col] == 0) {

for (int num = 1; num <= N; num++) {

/*Σε περίπτωση που γίνει έλεγχος με τη συνάρτηση isSafe και δεν βρεθεί λάθος τότε μπορεί να τοποθετηθεί ο αριθμός num στη γραμμή και στη στήλη που ορίσαμε. */

if (isSafe(grid, row, col, num)) {

grid[row][col] = num;

/*Σε περίπτωση που γεμίσει ο πίνακας με τη συνάρτηση fillSudoku τότε πρόγραμμα δεν εκτελέστηκε σωστά. */

```
if (fillSudoku(grid)) {

return 1;

}


/*Ο πίνακας grid ισούται με το 0 και επιστρέφει πίσω στην αναζήτηση. */

// πίσω αναζήτηση

grid[row][col] = 0;

}

}

/*Επιστρέφει το 0 που σημαίνει ότι δεν βρέθηκε κανένας αριθμός.*/

// Δεν βρέθηκε αριθμός

return 0;

}

}

}

/*Το πρόγραμμα δεν εκτελέστηκε σωστά.*/

return 1;

}


/* Δημιουργούμε μία κενή συνάρτηση με το όνομα printBoard βοηθητική λειτουργία για την εκτύπωση πλέγματος. */

void printBoard(int arr[N][N])
```

```
{



/*Με το for κάνουμε επανάληψη 9 φορές τις γραμμές.*/

for (int row = 0; row < N; row++)

{


/*Σε περίπτωση που το υπόλοιπο της διαίρεσης της γραμμής με το 3 είναι 0 και η γραμμή είναι διάφορη του 0 τότε  θα εμφανίσει διαχωριστικές οριζόντιες γραμμές.*/

if (row % 3 == 0 && row != 0) {

// Διαχωριστική γραμμή ανά 3 γραμμές

printf("----------\n");

}


/*Κάνουμε επανάληψη 9 φορές τις στήλες col.*/

for (int col = 0; col < N; col++) {


/*Σε περίπτωση που το υπόλοιπο από τη διαίρεση της στήλης με το 3 είναι 0 και ο αριθμός των στηλών είναι διάφορο του 0 τότε θα εμφανίσει κάθετες γραμμές.*/

if (col % 3 == 0 && col != 0) {

// Διαφορετικές στήλες ανά 3 στήλες

printf("| ");

}
```

```
/*Σε περίπτωση που ο πίνακας arr είναι 0 τότε θα εμφανιστεί το μήνυμα με τον αριθμό 0.*/

if (arr[row][col] == 0) {

printf("0 ");



}

/*Αλλιώς θα εμφανιστεί ο αριθμός του πίνακα.*/

else {

printf("%d ", arr[row][col]);

}

}



/*Αφήνει μια γραμμή κενή.*/

printf("\n");

}

}



/*Δημιουργούμε μία κενή συνάρτηση με το void το removeCells όπου αφαιρεί τυχαίους αριθμούς κάθε φορά που τρέχουμε τον πίνακα ξανά για να εμφανίζει κάποια κελιά με τον αριθμό 0 για να τα συμπληρώσουμε καλώντας τον πίνακα grid και την ακέραια μεταβλητή count. */

// Συνάρτηση για να αφαιρέσει τυχαία κελιά από τον πίνακα

void removeCells(int grid[N][N], int count) {
```

```
/*Ενώ το count είναι διάφορο του 0 ορίζουμε την ακεραια μεταβλητη i όπου το rand() χρησιμοποιείται για τη δημιουργία  ψευδό - τυχαίων αριθμών και υπολογιζει το υπολοιπο απο τη διαρεση με το 9.*/

while (count != 0) {

int i = rand() % N;


/*Ορίζουμε την ακεραια μεταβλητη j , το rand() χρησιμοποιείται για τη δημιουργία ψευδό - τυχαίων αριθμών και υπολογιζει το υπολοιπο απο τη διαρεση με το 9.*/

int j = rand() % N;


/*Σε περίπτωση που ο πίνακας grid με i γραμμές και j στήλες είναι διάφορος του 0 τότε*/

if (grid[i][j] != 0) {


/*Ορίζουμε τον πίνακα grid με i γραμμές και j στήλες είναι ίσο με το 0 και αφαιρούμε το count με το 1 δηλαδή count = count - 1. */

// Αφαιρούμε τον αριθμό

grid[i][j] = 0;

count--;

}

}

}


/*Φτιάχνουμε την κύρια συνάρτηση main.*/
```

```
int main()

{

/*Ορίζουμε τον πίνακα arr με 9 γραμμές και στήλες οπου όλοι οι αριθμοί θα είναι 0.*/

// Αρχικοποίηση του πίνακα με μηδενικά

int arr[N][N] = {0};
```

/*Ορίζουμε τις ακέραιες μεταβλητές δεδομένων row,col και num δηλαδή τις γραμμές, τις στήλες και τον αριθμό.*/

```
int row, col, num;
```

/*Το char είναι ένας τύπος δεδομένων που χρησιμοποιείται για την αποθήκευση ενός μόνο χαρακτήρα και με αυτό ορίζουμε το choice δηλαδή την επιλογή. */

```
char choice;
```

/*Το srand(time(NULL)) χρησιμοποιεί το εσωτερικό ρολόι του υπολογιστή για τον έλεγχο της επιλογής του σπόρου. Δεδομένου ότι ο χρόνος αλλάζει συνεχώς, ο σπόρος αλλάζει για πάντα. Σε περίπτωση που ο αριθμός σπόρων παραμένει ο ίδιος, η ακολουθία των αριθμών θα επαναληφθεί για κάθε εκτέλεση του προγράμματος.*/

```
// Αρχικοποίηση της γεννήτριας τυχαιότητας

srand(time(NULL));
```

/*Εμφανίζει το μήνυμα “Chart Sudoku:” και αλλάζει γραμμή.*/

```
printf("Chart Sudoku:\n");
```

```
/*Καλούμε την συνάρτηση fillSudoku με τον πίνακα arr.*/

// Γέμισμα του πίνακα με τυχαίους αριθμούς

fillSudoku(arr);


/*Εκτελούμε την συνάρτηση removeCells.*/

// Αφαιρέστε τυχαία κελιά από τον πίνακα (δηλαδή, να βάλουμε 0s)

removeCells(arr, 40); // Αφαιρούμε 40 αριθμούς για να αφήσουμε κενά


/*Καλούμε την συνάρτηση printBoard.*/

// Εκτύπωση του αρχικού πίνακα

printBoard(arr);


/*Το while (1) δημιουργεί ένα άπειρο βρόχο (loop) που θα συνεχίσει να τρέχει μέχρι να συναντήσει μια δήλωση break. Εμφανίζει το μήνυμα “Enter position (row and column) for number (e.g. 1,2) or 'q' to exit:” όπου γράφουμε την τιμή της γραμμής και της στήλης και σε περίπτωση που θέλουμε να σταματήσουμε γραφουμε το q. */

while (1) {

printf("\nEnter position (row and column) for number (e.g. 1,2) or 'q' to exit: ");


/*Σε περίπτωση που διαλέγει ο παίκτης να εισάγει ένα χαρακτήρα και η επιλογή είναι το q τότε το παιχνίδι θα σταματήσει.*/

if (scanf(" %c", &choice) == 1 && choice == 'q') {

// Έξοδος από το παιχνίδι
```

```
break;

}
```

/*Η συνάρτηση της βιβλιοθήκης C ungetc μεταφέρει τον χαρακτήρα char choice στο καθορισμένο ρεύμα (stream) έτσι ώστε να είναι διαθέσιμο για την επόμενη λειτουργία ανάγνωσης.Το stdin είναι μια "input stream", η οποία είναι ένας αφηρημένος όρος για κάτι που λαμβάνει εισροές (input) από το χρήστη ή από ένα αρχείο.*/

```
// Επιστρέφει το χαρακτήρα στην ροή εισόδου

ungetc(choice, stdin);
```

/*Ορίζουμε τον αριθμό των γραμμών και των στηλών.*/

```
// Εισάγει γραμμή και στήλη

scanf("%d %d",&row,&col);
```

/*Αφαιρει τον αριθμο των γραμμων με το 1.*/

```
// Μετατροπή σε 0-8

row -= 1;
```

/*Αφαιρει τον αριθμο των στηλων με το 1.*/

```
// Μετατροπή σε 0-8

col -= 1;
```

/*Σε περίπτωση που η γραμμή και είναι μικρότερη από το 0 ή μεγαλύτερη ή ίση από τον πίνακα του Sudoku ή η στήλη είναι μικρότερη απο το 0 ή μεγαλυτερη ή ιση σπο τον

πίνακα του Sudoku τότε θα εμφανιστεί το μυνημα «Invalid location! Try again.» οτι ο παίκτης βγήκε έξω από τα όρια των γραμμών και των στηλών και ζητάει να αλλάξει τους αριθμούς των γραμμών και των στηλών.*/

```
if (row < 0 || row >= N || col < 0 || col >= N) {

printf("Invalid location! Try again.\n");

}
```

/*Εμφανίζεται σε διαφορετική γραμμή το μήνυμα «Enter a number (1-9):» οπου ζητάει από το παικτη να γράψει ένα αριθμό απο το 1 έως το 9.*/

```
printf ("\nEnter a number (1-9): ");
```

/*Ο παίκτης γράφει τους αριθμούς από το 1 έως το 9.*/

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

/*Σε περίπτωση που αριθμός ειναι μικρότερος απο τον αριθμο 1 και μεγαλυτερος απο τον αριθμο 9 τοτε θα εμφανίστει σε αλλη γραμμή το μήνυμα «Invalid number! Numbers must be from 1-9.» οπου θα ζηταει οι αριθμοι να ειναι απο το 1 εως το 9. */

```
if (num < 1 || num > 9) {

printf("\nInvalid number! Numbers must be from 1-9.\n");

}
```

/*Σε περίπτωση που ο πίνακας arr είναι διάφορος του 0 τότε θα εμφανιστεί σε άλλη γραμμή το μήνυμα «The position is already taken! Try another one. » όπου θα εξηγεί στον παίκτη ότι ο αριθμός που έγραψε είναι ήδη στον πίνακα και του ζητάει να γράψει ενα άλλον αριθμό. */

```
if (arr[row][col] != 0) {

printf("\nThe position is already taken! Try another one.\n");

}


/*Σε περίπτωση που γράφουμε ένα αριθμό*/

// Έλεγχος αν τοποθετείται ο αριθμός

if (isSafe(arr, row, col, num)) {


/*Τοποθετείτε στον πίνακα arr ο αριθμός num.*/

// Τοποθέτηση αριθμού

arr[row][col] = num;

}



/*Αλλιώς θα εμφανίσει σε άλλη γραμμή το μήνυμα «This number cannot be placed in the position» με το άθροισμα των αριθμών των γραμμών με το 1 και το άθροισμα των αριθμών των στηλών με το 1.  */

else {

printf("\nThis number cannot be placed in the position (%d, %d).\n", row + 1, col + 1);

}
```

```
/*Εκτελούμε την συνάρτηση printBoard με τον πίνακα arr.*/

// Εκτύπωση του πίνακα μετά από κάθε εισαγωγή

printBoard(arr);

}


/*Όταν το παιχνίδι τελειώνει ή ο παίκτης θέλει να σταματήσει το παιχνίδι του εμφανίζει σε διαφορετική γραμμή το μήνυμα «Thank you for playing!» */

printf("\nThank you for playing!\n");


/* Το return 0 σημαίνει ότι το πρόγραμμα εκτελέστηκε με επιτυχία.*/

return 0;

}
```

2.2. Screenshots

Εικόνες από τον κώδικα

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// Το N είναι το μέγεθος του πίνακα Sudoku
#define N 9

// Συνάρτηση για να ελέγξει εάν είναι ασφαλής η τοποθέτηση ενός αριθμού
int isSafe(int grid[N][N], int row, int col, int num)
{

    // Ελέγχει αν βρίσκει τον ίδιο αριθμό στην παρόμοια σειρά και επιστρέφει στο 0
    for (int x = 0; x <= N; x++) {
        if (grid[row][x] == num) {
            return 0;
        }
    }

    // Ελέγχει αν βρίσκει τον ίδιο αριθμό στην παρόμοια στήλη και επιστρέφει στο 0
    for (int x = 0; x <= N; x++) {
        if (grid[x][col] == num) {
            return 0;
        }
    }

    for (int x = 0; x <= N; x++) {
        if (grid[row - row % 3 + x % 3][col - col % 3 + x % 3] == num) {
            return 0;
        }
    }
```

```c

    // Ελέγχει αν βρίσκει τον ίδιο αριθμό στον συγκεκριμένο πίνακα 3*3 και επιστρέφει 0
    int startRow = row - row % 3;
    int startCol = col - col % 3;

    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (grid[i + startRow][j + startCol] == num) {
                return 0;
            }
        }
    }
    return 1;
}

```

```c
// Συνάρτηση για να γεμίσει το πίνακα Sudoku
int fillSudoku(int grid[N][N]) {
    for (int row = 0; row < N; row++) {
        for (int col = 0; col < N; col++) {
            if (grid[row][col] == 0) {
                for (int num = 1; num <= N; num++) {
                    if (isSafe(grid, row, col, num)) {
                        grid[row][col] = num;
                        if (fillSudoku(grid)) {
                            return 1;
                        }
                        // πίσω αναζήτηση
                        grid[row][col] = 0;
                    }
                }
                // Δεν βρέθηκε αριθμός
                return 0;
            }
        }
    }
    return 1;
}

```

```c
/* Μια βοηθητική λειτουργία για την εκτύπωση πλέγματος. */
void printBoard(int arr[N][N])
{
    for (int row = 0; row < N; row++)
    {
        if (row % 3 == 0 && row != 0) {
            // Διαχωριστική γραμμή ανά 3 γραμμές
            printf("----------------------\n");
        }
        for (int col = 0; col < N; col++) {
            if (col % 3 == 0 && col != 0) {
                // Διαφορετικές στήλες ανά 3 στήλες
                printf("| ");
            }
            if (arr[row][col] == 0) {
                printf("0 ");
            } else {
                printf("%d ", arr[row][col]);
            }
        }
        printf("\n");
    }
}

```

```c
// Συνάρτηση για να αφαιρέσει τυχαία κελιά από τον πίνακα
void removeCells(int grid[N][N], int count) {
    while (count != 0) {
        int i = rand() % N;
        int j = rand() % N;
        if (grid[i][j] != 0) {
            // Αφαιρούμε τον αριθμό
            grid[i][j] = 0;
            count--;
        }
    }
}


```

```c
int main()
{
    // Αρχικοποίηση του πίνακα με μηδενικά
    int arr[N][N] = {0};
    int row, col, num;
    char choice;

    // Αρχικοποίηση της γεννήτριας τυχαιότητας
    srand(time(NULL));

    printf("Chart Sudoku:\n");
    // Γέμισμα του πίνακα με τυχαίους αριθμούς
    fillSudoku(arr);

    // Αφαιρέστε τυχαία κελιά από τον πίνακα (δηλαδή, να βάλουμε 0s)
    removeCells(arr, 40); // Αφαιρούμε 40 αριθμούς για να αφήσουμε κενά

    // Εκτύπωση του αρχικού πίνακα
    printBoard(arr);

    while (1) {
        printf("\nEnter position (row and column) for number (e.g. 1,2) or 'q' to exit: ");
        if (scanf(" %c", &choice) == 1 && choice == 'q') {
            // Έξοδος από το παιχνίδι
            break;
        }

```

```
        // Επιστρέφει το χαρακτήρα στην ροή εισόδου
        ungetc(choice, stdin);

        // Εισάγει γραμμή και στήλη
        scanf("%d %d",&row,&col);

        // Μετατροπή σε 0-8
        row -= 1;
        // Μετατροπή σε 0-8
        col -= 1;

        if (row < 0 || row >= N || col < 0 || col >= N) {
            printf("Invalid location! Try again.\n");
        }

        printf ("\nEnter a number (1-9): ");
        scanf("%d",&num);

        if (num < 1 || num > 9) {
            printf("\nInvalid number! Numbers must be from 1-9.\n");
        }

        if (arr[row][col] != 0) {
            printf("\nThe position is already taken! Try another one.\n");
        }

```

```
        // Ελεγχος αν τοποθετείται ο αριθμός
        if (isSafe(arr, row, col, num)) {
            // Τοποθέτηση αριθμού
            arr[row][col] = num;
        }
        else {
            printf("\nThis number cannot be placed in the position (%d, %d).\n", row + 1, col + 1);
        }

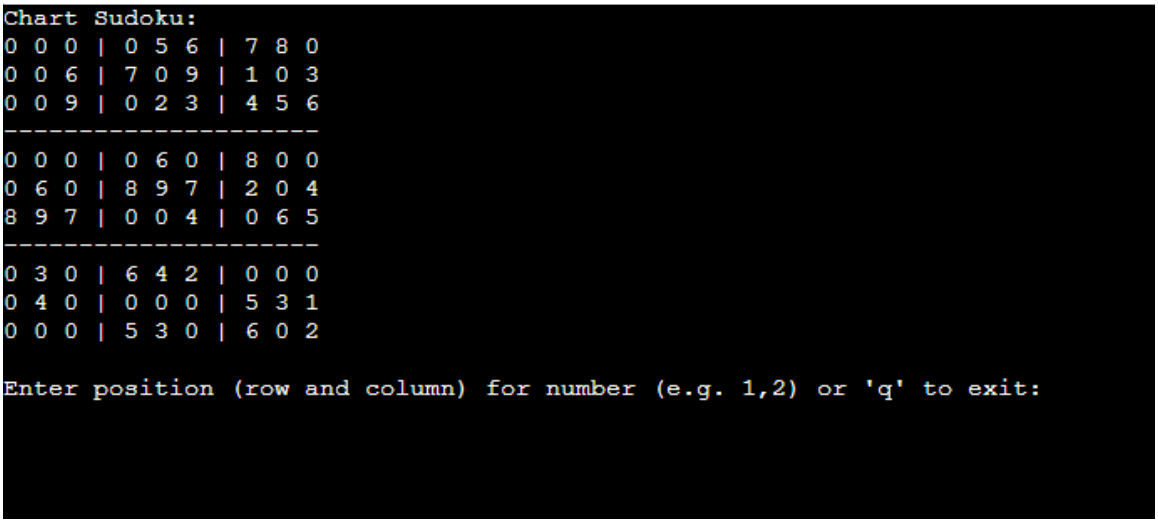
        // Εκτύπωση του πίνακα μετά από κάθε εισαγωγή
        printBoard(arr);
    }

    printf("\nThank you for playing!\n");
    return 0;
}

```

Όταν τρέχουμε τον κώδικα

Όταν τρέχουμε τον κώδικα εμφανίζεται το μήνυμα: “Chart Sudoku: ” και εμφανίζεται σε άλλη γραμμή ο πίνακας και στη συνέχεια το μήνυμα “Enter position (row and column) for number (e.g. 1,2) or ‘q’ to exit: ” όπου θα ζητάει να γράψουμε τον αριθμό της γραμμής και της στήλης αλλιώς θα γράψουμε το γράμμα q για να βγει από το παιχνίδι.



Γράφουμε για παράδειγμα την γραμμή 1 και την στήλη 2 και εμφανίζεται το μήνυμα “Enter a number (1-9): ” που ζητάει να γράψουμε ένα αριθμό από το 1 έως το 9 και γράφουμε το 2 και στην επόμενη γραμμή εμφάνιζεται ξανά ο πίνακας με τον αριθμό 2 στην θέση που επιλέξαμε.

```
Chart Sudoku:
0 0 0 | 0 5 6 | 7 8 0
0 0 6 | 7 0 9 | 1 0 3
0 0 9 | 0 2 3 | 4 5 6
---------------------
0 0 0 | 0 6 0 | 8 0 0
0 6 0 | 8 9 7 | 2 0 4
8 9 7 | 0 0 4 | 0 6 5
---------------------
0 3 0 | 6 4 2 | 0 0 0
0 4 0 | 0 0 0 | 5 3 1
0 0 0 | 5 3 0 | 6 0 2

Enter position (row and column) for number (e.g. 1,2) or 'q' to exit: 1 2

Enter a number (1-9): 2
0 2 0 | 0 5 6 | 7 8 0
0 0 6 | 7 0 9 | 1 0 3
0 0 9 | 0 2 3 | 4 5 6
---------------------
0 0 0 | 0 6 0 | 8 0 0
0 6 0 | 8 9 7 | 2 0 4
8 9 7 | 0 0 4 | 0 6 5
---------------------
0 3 0 | 6 4 2 | 0 0 0
0 4 0 | 0 0 0 | 5 3 1
0 0 0 | 5 3 0 | 6 0 2

Enter position (row and column) for number (e.g. 1,2) or 'q' to exit:
```

Όταν γράφουμε τον αριθμό 3 στη γραμμή 2 και θέση 2 τότε εμφανίζεται το μήνυμα “This number cannot be placed in the position (2, 2).” που σημαίνει ότι δεν μπορεί να τοποθετηθεί ο αριθμός στη θέση που επιλέξαμε.

```
Enter position (row and column) for number (e.g. 1,2) or 'q' to exit: 1 2

Enter a number (1-9): 2
0 2 0 | 0 5 6 | 7 8 0
0 0 6 | 7 0 9 | 1 0 3
0 0 9 | 0 2 3 | 4 5 6
---------------------
0 0 0 | 0 6 0 | 8 0 0
0 6 0 | 8 9 7 | 2 0 4
8 9 7 | 0 0 4 | 0 6 5
---------------------
0 3 0 | 6 4 2 | 0 0 0
0 4 0 | 0 0 0 | 5 3 1
0 0 0 | 5 3 0 | 6 0 2

Enter position (row and column) for number (e.g. 1,2) or 'q' to exit: 2 2

Enter a number (1-9): 3

This number cannot be placed in the position (2, 2).
0 2 0 | 0 5 6 | 7 8 0
0 0 6 | 7 0 9 | 1 0 3
0 0 9 | 0 2 3 | 4 5 6
---------------------
0 0 0 | 0 6 0 | 8 0 0
0 6 0 | 8 9 7 | 2 0 4
8 9 7 | 0 0 4 | 0 6 5
---------------------
0 3 0 | 6 4 2 | 0 0 0
0 4 0 | 0 0 0 | 5 3 1
0 0 0 | 5 3 0 | 6 0 2

Enter position (row and column) for number (e.g. 1,2) or 'q' to exit:
```

3.Βιβλιογραφία/Πηγές

[1] https://en.wikipedia.org/wiki/C_(programming_language)

[2] https://www.simplilearn.com/tutorials/c-tutorial/use-of-c-language#what_is_c_language

[3] https://www.simplilearn.com/tutorials/c-tutorial/use-of-c-language#history_of_c_language

[4] https://www.simplilearn.com/tutorials/c-tutorial/use-of-c-language#benefits_of_c_language_over_other_programming_languages

[5] https://www.simplilearn.com/tutorials/c-tutorial/use-of-c-language#key_features_of_c_programming_language

[6] https://www.mygreatlearning.com/blog/c-tutorial/

[7] https://codingzap.com/learn-c-in-easy-steps/

[8] https://www.simplilearn.com/tutorials/c-tutorial/use-of-c-language#future_of_c_programming_language

[9] https://hpc-wiki.info/hpc/Compiler

[10] https://en.wikipedia.org/wiki/Compiler

[11] https://www.techtarget.com/whatis/definition/compiler

[12] https://www.codementor.io/@ankitdixit7890/what-is-compiler-1um9dn13uc

[13] https://www.qodo.ai/question/code-compiler/

[14] https://hourglassnewspaper.com/872/features/the-history-of-sudoku/#

[15] https://en.wikipedia.org/wiki/Sudoku

[16] https://www.seniorlifestyle.com/resources/blog/5-tips-sudoku-beginners/

[17] https://hourglassnewspaper.com/872/features/the-history-of-sudoku/

[18] https://en.wikipedia.org/wiki/Mathematics_of_Sudoku

[19] https://pi.math.cornell.edu/~mec/Summer2009/Mahmood/Intro.html

[20] https://en.wikipedia.org/wiki/NP-completeness

[21] https://www.researchgate.net/figure/Euler-diagram-for-P-NP-NP-complete-and-NP-hard-set-of-problems_fig1_365943165

[22] https://en.wikipedia.org/wiki/Sudoku_solving_algorithms

[23] https://en.wikipedia.org/wiki/Knuth%27s_Algorithm_X