Αναφορά Εργασίας με θέμα ναρκαλιευτή ή αλλιώς Minesweeper σε γλώσσα C++

Τι αφορά το project;

-Το project αφορά την υλοποίηση του κλασικού παιχνιδιού "Ναρκαλιευτής" (Minesweeper) με τη χρήση της γλώσσας προγραμματισμού C++. Ο χρήστης καλείται να εντοπίσει τις κρυμμένες νάρκες σε ένα πλέγμα κελιών, βασιζόμενος σε αριθμητικές ενδείξεις που αποκαλύπτονται στα γειτονικά κελιά. Στόχος είναι να ολοκληρώσει την πίστα χωρίς να ενεργοποιήσει κάποια νάρκη.

Ποιο πρόβλημα λύνει;

-Το project δεν λύνει κάποιο πρακτικό πρόβλημα στον πραγματικό κόσμο, αλλά επικεντρώνεται στην επίλυση ενός προβλήματος λογικής και ανάπτυξης λογισμικού. Μέσα από την υλοποίηση, εξασκούνται θεμελιώδεις έννοιες της προγραμματιστικής λογικής όπως οι αλγόριθμοι αναζήτησης, οι δισδιάστατοι πίνακες, η διαχείριση εισόδου/εξόδου και η αναπαράσταση γραφικής πληροφορίας σε κείμενο. Παράλληλα, ενισχύεται η ικανότητα λήψης αποφάσεων με βάση δεδομένα (π.χ. πόσες νάρκες υπάρχουν γύρω από ένα κελί).

Γιατί επιλέξατε αυτό το αντικείμενο;

Επέλεξαμε αυτό το αντικείμενο γιατί ο "Ναρκαλιευτής" είναι ένα γνώριμο και κλασικό παιχνίδι που συνδυάζει διασκέδαση με προγραμματιστική πρόκληση. Προσφέρει μια εξαιρετική ευκαιρία

για εξάσκηση σε σημαντικές έννοιες της C++ όπως η χρήση πινάκων, συνθηκών,συναρτήσων, επαναλήψεων και δυναμικής διαχείρισης μνήμης.

Ποιες δυνατότητες περιλαμβάνονται;

Το παιχνίδι υποστηρίζει τις εξής δυνατότητες:

- *Επιλογή επιπέδου δυσκολίας από 1 (ΕΥΚΟΛΟ)έως 5(ΔΥΣΚΟΛΟ).*
- *Δυναμική δημιουργία πινάκων τοποθετημένες τυχαία (όχι γύρω από το πρώτο κλικ).*
- *Αποκάλυψη κελιών (r): εάν ένα κελί δεν έχει νάρκη, εμφανίζει τον αριθμό των γειτονικών ναρκών ή αποκαλύπτει κενή περιοχή αναδρομικά.*
- *Τοποθέτηση σημαίας (f): ο παίκτης μπορεί να σημειώσει πιθανά ύποπτα κελιά με σημαία.*
- *Αφαίρεση σημαίας (u): δυνατότητα διόρθωσης από λάθος τοποθέτηση σημαίας.*
- *Αποτροπή απώλειας στο πρώτο κλικ: οι νάρκες τοποθετούνται μόνο μετά την πρώτη ενέργεια.*
- *Σαφής εμφάνιση ταμπλό με ένδειξη αποκαλυμμένων κελιών, σημαδεμένων και μη αποκαλυμμένων περιοχών.*
- *Έλεγχος νίκης/ήττας: τερματισμός του παιχνιδιού με νικητήριο ή ηττημένο μήνυμα.*
- *Επαναληπτικότητα: δυνατότητα να ξεκινήσει νέο παιχνίδι χωρίς έξοδο από την εφαρμογή.*

Ποιοι είναι οι βασικοί στόχοι;

1)Να δημιουργηθεί μια πλήρως λειτουργική έκδοση του παιχνιδιού Minesweeper σε γλώσσα C++.

2) Να εξασκηθουμε σε **θεμελιώδεις δομές δεδομένων**, όπως δισδιάστατοι πίνακες, και σε **βασικές αρχές αλγοριθμικής σκέψης**.

3) Να ενσωματωθούν οι βασικοί μηχανισμοί του παιχνιδιού, όπως:

- **Τυχαία τοποθέτηση ναρκών** (με αποφυγή πρώτου κλικ πάνω σε νάρκη),
- **Αποκάλυψη κελιών** με λογική αναδρομής για κενές περιοχές,
- **Χειρισμός σημαίας** (flag/unflag),
- **Ανίχνευση νίκης ή ήττας**.

Ποιες είναι οι βασικές του συνιστώσες;

main(): Περιλαμβάνει επιλογη επιπέδου δυσκολίας (playGame(level)), επαναλαμβάνει το παιχνίδι με ερώτηση (y/n) χωρίς να τερματίζει το πρόγραμμα, Διαχειρίζεται ελέγχους εγκυρότητας επιπέδου.

printMessage():Εμφανίζει και μορφοποιείται με χρώματα

playGame(level): Αρχικοποιεί πίνακες με βάση το επίπεδο δυσκολίας.

Καθορίζει μέγεθος πλέγματος και αριθμό ναρκών.

resetGame():Επαναφορά όλων των πινάκων

plnarkes(): Τοποθέτηση 10 ναρκών τυχαία μετά το πρώτο κλικ.

printBoard(bool revealAll):Εμφανίζει αριθμούς με χρώματα. Εμφανίζει live counters (νάρκες, σημαίες, αποκαλυμμένα).

Στήλες και γραμμές αριθμούνται.

keli(): Αποκάλυψη κελιού, και αν δεν υπάρχουν γειτονικές νάρκες, αποκάλυψη όλης της γειτονιάς αναδρομικά.

isWin(): Έλεγχος αν όλα τα μη-νάρκες κελιά έχουν αποκαλυφθεί.

setColor(int): Αλλάζει το χρώμα της κονσόλας

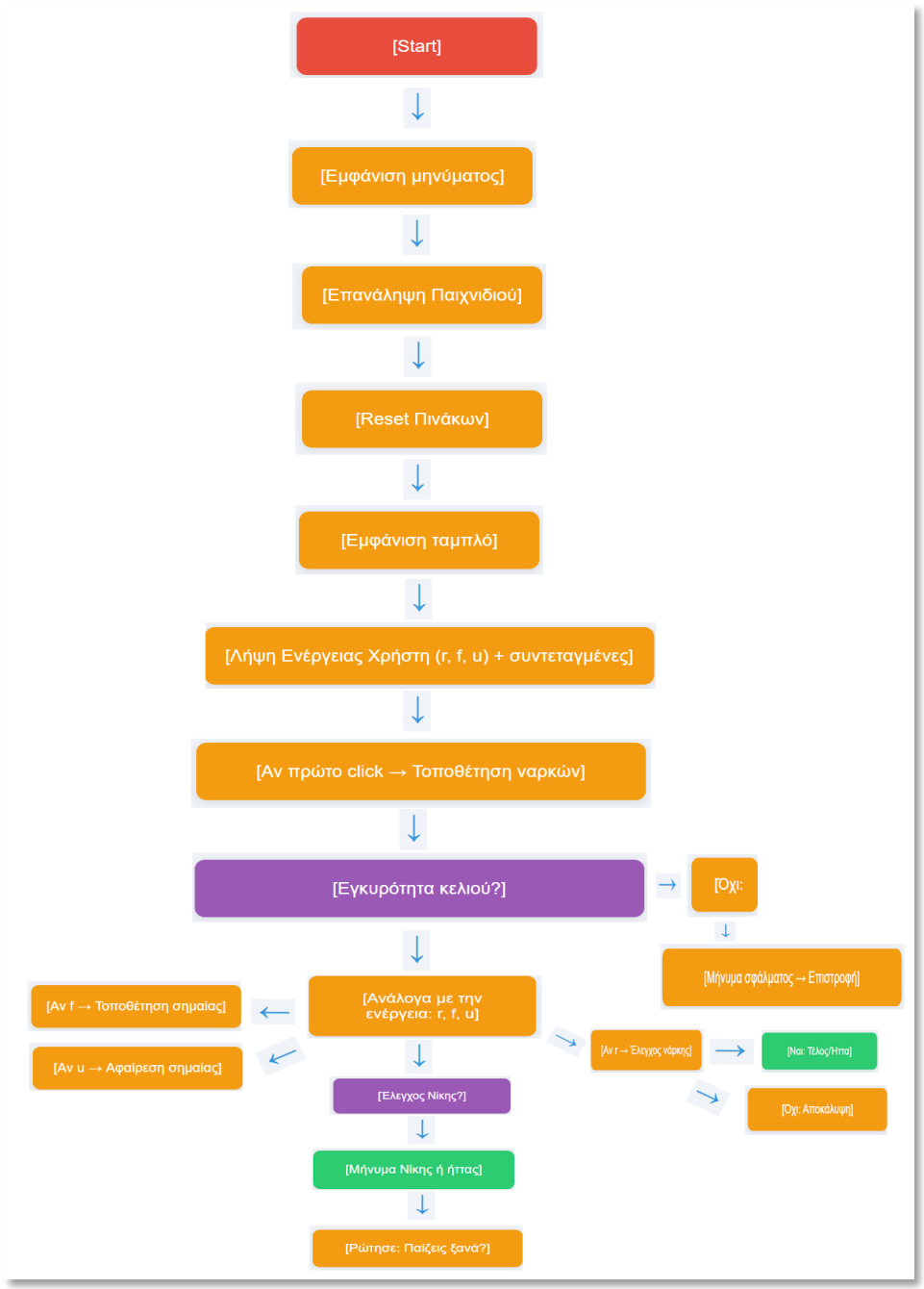
clearScreen():Καθαρισμός οθόνης

inBounds():Έλεγχος εγκυρότητας συντεταγμένων

Πώς λειτουργεί το σύστημα;

- Το σύστημα λειτουργεί ως διαδραστικό πρόγραμμα κονσόλας, το οποίο διαχειρίζεται την κατάσταση του παιχνιδιού Minesweeper μέσω πινάκων (2D vectors) και επιτρέπει στον χρήστη να αλληλεπιδρά με το παιχνίδι μέσω πληκτρολογίου.
- Το πρόγραμμα εκκινεί με ένα μήνυμα καλωσορίσματος και οδηγίες. Δημιουργούνται τρεις βασικοί πίνακες:Ορατό ταμπλό (board),πίνακας με τις πραγματικές νάρκες (narkes),πίνακας με αποκαλυμμένα κελιά (revealed).Μετά την πρώτη κίνηση του παίκτη, τοποθετούνται τυχαία οι νάρκες, εκτός της περιοχής του πρώτου κλικ.Ο χρήστης δίνει εντολές (reveal, flag, unflag), και το σύστημα:Ελέγχει εγκυρότητα,ενημερώνει τους πίνακες κατάλληλα,ελέγχει αν έχει γίνει αποκάλυψη νάρκης ή αν κερδήθηκε το παιχνίδι.Αν ολοκληρωθεί το παιχνίδι (νίκη ή ήττα), προσφέρεται η επιλογή επανεκκίνησης.

Διάγραμμα ροής(FLOWCHART):



Τεχνολογίες που Χρησιμοποιήθηκαν

Η C++ επιλέχθηκε για την ανάπτυξη του παιχνιδιού λόγω των εξής λόγων:

1. Είναι γρήγορη και αποτελεσματική όσον αφορά την απόδοση.

2. Παρέχει χαμηλού επιπέδου πρόσβαση στη μνήμη, κάτι που βοηθά στον έλεγχο των δομών δεδομένων όπως πίνακες και διανύσματα.

3. Διαθέτει πλούσιες βιβλιοθήκες και ισχυρή υποστήριξη για προγραμματισμό με δομές και αντικείμενα.

4. Είναι ευρέως διαδεδομένη και χρησιμοποιείται συχνά για την εκμάθηση των βασικών εννοιών του προγραμματισμού και της λογικής ελέγχου ροής

Βιβλιοθήκες που χρησιμοποιήθηκαν:

Βιβλιοθήκη	Περιγραφή
#include <iostream>	Είσοδος και έξοδος στην κονσόλα (cin, cout).
#include <vector>	Χρήση δυναμικών 2D πινάκων (board, revealed, flagged, κλπ).
#include <cstdlib>	Χρήση της rand() και system() για τυχαίες τιμές και καθαρισμό οθόνης.
#include <ctime>	Χρήση της time(0) ως seed για τυχαίους αριθμούς.
#include <string>	Υποστήριξη χειρισμού strings (π.χ. μηνύματα στην κονσόλα).
#include <windows.h>	Ρυθμίσεις εμφάνισης (χρώματα) στην κονσόλα Windows μέσω API.
#include <limits>	Έλεγχος μη έγκυρης εισόδου χρήστη (cin.fail() + ignore).

Πώς υλοποιήθηκε η βασική λειτουργικότητα;

Η βασική λειτουργικότητα του παιχνιδιού **Ναρκαλιευτής (Minesweeper)** υλοποιήθηκε με χρήση δομών δεδομένων όπως **πίνακες** και **δισδιάστατα διανύσματα**, που επέτρεψαν την αποτελεσματική αποθήκευση και διαχείριση της κατάστασης του παιχνιδιού. Μέσω αυτών των δομών, δημιουργήθηκε το πεδίο του παιχνιδιού, όπου κάθε κελί αντιπροσωπεύει μια θέση με ή χωρίς νάρκη, και διαχειρίζεται η εμφάνιση των αριθμών γύρω από τις νάρκες. Επιπλέον, εφαρμόστηκαν έλεγχοι ροής για την αντιμετώπιση εισόδων του χρήστη, την επιλογή κελιών, και την επικοινωνία με το χρήστη μέσω γραφικού περιβάλλοντος. Χρησιμοποιήθηκαν βασικοί αλγόριθμοι αναζήτησης και επανάληψης, όπως η **αναδρομική αναζήτηση** για το άνοιγμα κελιών όταν αποκαλύπτεται ένα κελί χωρίς γειτονικές νάρκες, καθώς και οι βρόχοι επανάληψης για τον έλεγχο της κατάστασης του παιχνιδιού και την ενημέρωση της οθόνης. Με αυτόν τον τρόπο, δημιουργήθηκε μια λειτουργική και διαδραστική υλοποίηση του παιχνιδιού, που βασίζεται σε βασικές αρχές προγραμματισμού και δομών δεδομένων.

Υπάρχουν σημαντικοί αλγόριθμοι ή δομές δεδομένων;

Δομές Δεδομένων

vector<vector<char>> board

Το «ταμπλό» που βλέπει ο χρήστης (με σύμβολα -, αριθμούς, κενά).

vector<vector<bool>> narkes

Πραγματική διάταξη ναρκών – υποδεικνύει αν ένα κελί περιέχει νάρκη.

vector<vector<bool>> revealed

Υποδεικνύει ποια κελιά έχουν ήδη αποκαλυφθεί.

vector<vector<bool>> flagged

Υποδεικνύει ποια κελιά έχουν επισημανθεί με σημαία από τον παίκτη.

Αλγόριθμοι

plnarkes(r, c)

Τοποθετεί **τυχαία νάρκες** αποφεύγοντας την περιοχή 3x3 γύρω από το πρώτο click.

keli(r, c)

Αναδρομική αποκάλυψη κελιών. Αν ένα κελί δεν έχει γειτονικές νάρκες, αποκαλύπτει και τα διπλανά του.

countAdjacentMines(r, c)

Μετρά πόσες νάρκες υπάρχουν γύρω από ένα κελί (σε 3x3 περιοχή)

isWin()

Ελέγχει αν έχουν αποκαλυφθεί όλα τα μη-ναρκοθετημένα κελιά. Επιστρέφει true σε περίπτωση νίκης.

printBoard()

Εμφανίζει το ταμπλό με αριθμούς, νάρκες, σημαίες, και counters (νάρκες που μένουν, αποκαλύψεις κλπ).

Έλεγχος Εισόδου

cin.fail() με limits

Ανίχνευση και αποφυγή λανθασμένης εισόδου από τον χρήστη (γράμματα, σύμβολα κλπ).

Αλγόριθμος ασφαλείας

inBounds(r, c)

Ελέγχει αν ένα κελί βρίσκεται εντός ορίων του πίνακα. Αποτρέπει σφάλματα out-of-bounds.

Κώδικας στο GitHub

Ο πλήρης κώδικας του project μας είναι διαθέσιμος στο παρακάτω link:

https://github.com/paidarakis/Minesweeper-C-.git

Μπορείτε να δείτε, να κάνετε clone ή να κατεβάσετε τον κώδικα από εκεί.

Αποτελέσματα και DEMO

*Το παιχνίδι Ναρκαλιευτής (Minesweeper) υλοποιήθηκε με επιτυχία στη γλώσσα **C++**, παρέχοντας πλήρως λειτουργική εμπειρία χρήστη μέσω της κονσόλας. Ο παίκτης μπορεί να:*

-Αποκαλύψει κελιά (r)

-Τοποθετήσει σημαίες (f)

-Αφαιρέσει σημαίες (u)

-Χάσει αν αποκαλύψει νάρκη

-Νικήσει αν αποκαλύψει όλα τα ασφαλή κελιά

```
========================================
         WELCOME TO MINESWEEPER!
========================================

Skopos: Apokalypste ola ta tetragwna pou den exoun narkh.
Odigies:
  r x y  -> Apokalyptete to keli sti grammi x, stin stilh y
  f x y  -> Topothethsete mia shmaia se to tetragwno sti grammi x, stin stilh y
  u x y  -> Afairete ti shmaia apo to tetragwno sti grammi x, stin stilh y
Shmape: H proti kinisi prepei na einai apokalypsi (r).

     0   1   2   3   4   5   6   7   8
   +---+---+---+---+---+---+---+---+---+
 0 | - | - | - | - | - | - | - | - | - |
   +---+---+---+---+---+---+---+---+---+
 1 | - | - | - | - | - | - | - | - | - |
   +---+---+---+---+---+---+---+---+---+
 2 | - | - | - | - | - | - | - | - | - |
   +---+---+---+---+---+---+---+---+---+
 3 | - | - | - | - | - | - | - | - | - |
   +---+---+---+---+---+---+---+---+---+
 4 | - | - | - | - | - | - | - | - | - |
   +---+---+---+---+---+---+---+---+---+
 5 | - | - | - | - | - | - | - | - | - |
   +---+---+---+---+---+---+---+---+---+
 6 | - | - | - | - | - | - | - | - | - |
   +---+---+---+---+---+---+---+---+---+
 7 | - | - | - | - | - | - | - | - | - |
   +---+---+---+---+---+---+---+---+---+
 8 | - | - | - | - | - | - | - | - | - |
   +---+---+---+---+---+---+---+---+---+

Narkes pou apokalupthikan: 0 | Shmaies: 0 | Ypolipomenes narkes: 10

Eisagete entoli (r gia apokalypsi, f gia shmaia, u gia aporiptisi shmaias) kai suntetagmenes (r x y): r 2 4
```

```
C:\Users\user\Pictures\Minesweeper.exe

Narkes pou apokalupthikan: 66 | Shmaies: 0 | Ypolipomenes narkes: 10

Eisagete entoli (r gia apokalypsi, f gia shmaia, u gia aporiptisi shmaias) kai suntetagmenes (r x y): r 5 0
     0   1   2   3   4   5   6   7   8
   +---+---+---+---+---+---+---+---+---+
 0 | - | - | * | 1 |   | 1 | - | 1 |   |
   +---+---+---+---+---+---+---+---+---+
 1 | * | 2 | 1 | 1 |   | 1 | * | 1 |   |
   +---+---+---+---+---+---+---+---+---+
 2 | 1 | 1 |   |   |   | 1 | 1 | 1 |   |
   +---+---+---+---+---+---+---+---+---+
 3 |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+
 4 | 1 | 1 |   | 1 | 1 | 1 |   |   |   |
   +---+---+---+---+---+---+---+---+---+
 5 | * | 1 |   | 2 | * | 4 | 2 | 2 | 1 |
   +---+---+---+---+---+---+---+---+---+
 6 | 1 | 1 |   | 2 | * | * | * | - | * |
   +---+---+---+---+---+---+---+---+---+
 7 |   |   |   | 1 | 2 | 3 | 2 | 3 | * |
   +---+---+---+---+---+---+---+---+---+
 8 |   |   |   |   |   |   |   | 1 | - |
   +---+---+---+---+---+---+---+---+---+

Narkes pou apokalupthikan: 66 | Shmaies: 0 | Ypolipomenes narkes: 10


BOOM! Pathses se narkh. Telos paixnidiou.
Thelete na paixete ksana?(y/n):
```

Σύγκριση με κώδικα δημιουργιμένο απο AI

Διαφορές / Πλεονεκτήματα Ανθρώπινης Υλοποίησης:

<u>Καθαρός και σταδιακά αναπτυγμένος κώδικας</u>:Ο κώδικάς μας, αν και όχι τέλειος, δεν είναι παρόμοιος με αυτόν της τεχνητής νοημοσύνης καθώς υπάρχει διαφορά στα κενά και στη δομή μεταξύ των μεταβλητών, των συναρτήσεων σε σύγκριση με ένα AI και τα ονόματα των συναρτήσεων δεν είναι στα αγγλικα αλλα στα ελληνικά .

<u>Προσαρμογή στη γλώσσα και στο κοινό</u>: Τα σχόλια είναι γραμμένα σε greeklish, κάτι που δεν θα προέκυπτε αυτόματα από AI χωρίς καθοδήγηση.

<u>Κατανόηση ροής παιχνιδιού:</u> Από την αρχική ρύθμιση μέχρι την τοποθέτηση ναρκών και την ερώτηση για την επανεκκίνηση , δείχνει κατανόηση των διαδοχικών σταδίων ενός προγράμματος. Σε αντίθεση ο AI-Generated Code προσεγγιζει αυτά τα στάδια με πιο γενικό ή βιαστικό τρόπο

Συμπεράσματα και πράγματα που μάθαμε

Η υλοποίηση του project αποτέλεσε μια εξαιρετικά εκπαιδευτική εμπειρία, καθώς συνδύασε τον σχεδιασμό,τον προγραμματισμό,τη λογική σκέψη και τη συνεργασία.

Μαθήματα που αποκομίσαμε:

1. ***Κατανόηση λογικής παιχνιδιών***
   Μάθαμε πώς σχεδιάζεται και προγραμματίζεται η λογική ενός παιχνιδιού: τι συμβαίνει σε κάθε βήμα,πώς αντιδρά το σύστημα στις ενέργειες του χρήστη και πώς διαχειριζόμαστε εξαιρέσεις ή μη έγκυρες εισόδους.
2. ***Ανάπτυξη σε κονσόλα***
   Αξιοποιήσαμε πλήρως την κονσόλα για είσοδο/έξοδο και διαμορφώσαμε ένα καθαρό περιβάλλον παιχνιδιού.
3. ***Ομαδική εργασία & Github***
   Μάθαμε να χρησιμοποιούμε το Github για συνεργασία, κατανοώντας βασικές έννοιες όπως commit, push, conflict resolution, και διαμοιρασμός κώδικα.

ΑΝΑΦΟΡΑ ΑΠΟ ΤΟΥΣ ΜΑΘΗΤΕΣ :

Από τους μαθητές:

- ***Κωσταντίνο Παιδαράκη***
- ***Αλέξανδρο Κοζάρη***
- ***Στέφανο Σκεντερίδη***