LOIM Fejlesztői dokumentáció

A programozás alapjai 1 BMEVIEEAA00

Választott feladat: Legyen ön is milliomos

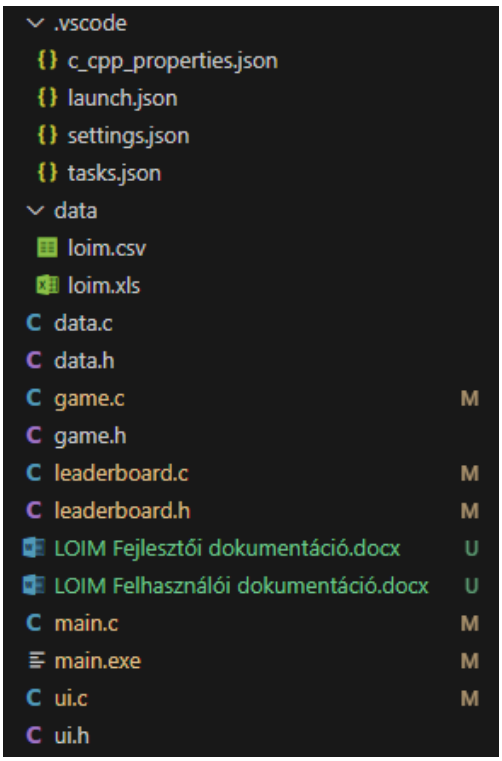
Készítette: Gombási János Márk, PACY5Z

Contents

Áttekintés

A projekt a népszerű "Legyen Ön is Milliomos!" tévés vetélkedő C nyelven, konzolos felületen megvalósított adaptációja. Filestruktúra

A program filestruktúrája a következő képpen néz ki:



A könnyebb átláthatóság és a magasabb szintű karbantarthatóság érdekében a projekt több, különálló modulra van bontva, melyek mind saját forrás (.c) és header (.h) fájllal rendelkeznek.

Main.c: A program belépési pontja. Felelős a szükséges inicializálásokér
Data: Kizárólag az adatkezelésért felelős modul. Feladata a kérdések beolvasása, memóriában való tárolása, valamint a játéklogika számára szűrőfüggvények biztosítása.
Game: Ez a modul vezérli a teljes játékmenetet a főmenütől a kérdések feltételén át a segítségek kezeléséig.
Leaderboard: A ranglista kezelésére szolgál
ui: A felhasználói felület (UI) modulja. Itt található minden, a konzolra való kiíratással kapcsolatos függvény

Adatkezelés

Erre a korábban említett data.c és data.h file felel. A program a kérdéseket a data mappában lévő loim.csv fileból olvassa, majd egy questions tömbbe rakja őket amiket a tipusa a Question struktúra.

```
typedef struct {
    int difficulty;
    char* question;
    char* answer1;
    char* answer2;
    char* answer3;
    char* answer4;
    char* correctAnswer;
    char* category;
} Question;
```

Emellett, ez a file felel a kérdések szűréséért is, amit 3 részre lehet osztani.

1. **Lépés: Szűrés Nehézség Szerint**
   A folyamat azzal kezdődik, hogy a felhasználó nehézségi szintet választ. A program ez alapján kiválogatja az összes kérdés közül azokat, amelyek a megadott nehézségi tartományba esnek.
   A szűrés eredményeként létrejön a questionPool tomb.

```
//Ez az első szűrési lépés nehézség alapján.
void buildPoolByDifficulty(int minDiff, int maxDiff) {
    poolCount = 0;
    for (int i = 0; i < questionCount; i++) {
        if (questions[i].difficulty >= minDiff && questions[i].difficulty <= maxDiff) {
            questionPool[poolCount] = &questions[i];
            poolCount++;
        }
    }

    if (poolCount == 0) {
        printf("Sajnos ezen a nehezsegi szinten nincsenek kerdesek.\n");
        return;
    }

    printf("poolCount after filtering by difficulty: %d\n", poolCount);
}
```

2. Lépés: Releváns Kategóriák Gyűjtése

Hogy a felhasználó ne választhasson üres kategóriát, a program megvizsgálja a már leszűkített questionPool-t, és kigyűjti belőle az egyedi kategórianeveket.

Ez biztosítja, hogy a felhasználónak csak olyan kategóriák jelenjenek meg, amelyekben garantáltan van kérdés a választott nehézségi szinten.

```
//ez a második lépés, ami a nehezsegi szint alapján gyűjti a kategóriákat.
void getCategoriesFromPool(Question* pool[], int poolCount) {
    categoryCount = 0;
    for (int i = 0; i < poolCount; i++) {
        if (pool[i]->category == NULL || strlen(pool[i]->category) == 0)
            continue;

        if (!isInArray(pool[i]->category) && categoryCount < MAX_CATEGORIES) {
            categories[categoryCount] = strdup(pool[i]->category);
            categoryCount++;
        }
    }
}
```

3. Lépés: Végső szűrés

Ez a függvény kapja meg a felhasználó által választott kategóriák sorszámait. A külső ciklusa végigmegy a questionPool minden elemén, míg egy belső ciklus ellenőrzi, hogy az adott kérdés kategóriája szerepel-e a felhasználó választásai között.Ha igen, a kérdés a questionPool elején marad ("megtartjuk"), ha nem, akkor "hátrahagyjuk". A folyamat végén a poolCount értéke frissül a végleges, leszűkített kérdéshalmaz méretére.

```
void filterPoolBySelectedCategories(const int selectedIndexes[], int count) {
    // Ha a felhasználó a '0'-t (Mind) választotta, nincs mit szűrni.
    if (count == 0) {
        return;
    }

    int currentPoolCount = 0;
    for (int i = 0; i < poolCount; i++) {
        bool shouldKeep = false;
        for (int j = 0; j < count; j++) {
            int selectedIndex = selectedIndexes[j];
            if (selectedIndex > 0 && selectedIndex <= categoryCount) {
                const char* selectedCategoryName = categories[selectedIndex - 1];
                if (questionPool[i]->category && strcmp(questionPool[i]->category, selectedCategoryName) == 0) {
                    shouldKeep = true;
                    break;
                }
            }
        }
        if (shouldKeep) {
            questionPool[currentPoolCount] = questionPool[i];
            currentPoolCount++;
        }
    }

    poolCount = currentPoolCount;
}
```

Játékmenet

Ezt a game fileok működtetik. A program először bekéri hogy mit szeretne csinálni a felhasználó majd a megfelelő kódreszt futtaja le.

```c
void startGame() {
    int choice = 0;

    while (1) {
        drawWelcomeScreen();

        if (scanf("%d", &choice) != 1) {
            choice = -1;
        }
        while (getchar() != '\n');

        switch (choice) {
            case 1:
                playGame();
                break;
            case 2:
                drawLeaderboard();
                printf("\nNyomj egy entert a tovabblepeshez...");
                getchar();
                continue;
            case 3:
                exit(0);
            default:
                printf("Ervenytelen valasztas! Probald ujra.\n");
                break;
        }
        break;
    }
}
```

Új játék

Új játék kezdésekör lefut a PlayGame függvény, ami inicializálja a kellő változókat, mint pl. a segítségek. Ezek után bekéri a játékos nevét, kívánt nehézségi fokozatot és a kategóriákat is. Közben meghívja a megfelő szűrő funkciókat amik a data.c fileban találhatóak, valamit a megfelelő függvényeket a UI modulból.

```
void playGame() {
    bool fiftyfifty=true;
    bool phone=true;
    bool audiance=true;
    char playerName[51];
    int diffChoice = 0;

    printf("\nKerlek, add meg a jatekos nevet (max 50 karakter): ");
    fgets(playerName, sizeof(playerName), stdin);
    playerName[strcspn(playerName, "\n")] = 0;

    drawDifficultyMenu(); //ui.c
    char inputBuffer[10];
    fgets(inputBuffer, sizeof(inputBuffer), stdin);
    sscanf(inputBuffer, "%d", &diffChoice);

    int minDiff, maxDiff;
    switch(diffChoice) {
        case 1: minDiff = 1; maxDiff = 5; break;
        case 2: minDiff = 6; maxDiff = 10; break;
        case 3: minDiff = 11; maxDiff = 15; break;
        default: printf("Ervenytelen nehezsegi szint. Visszateres a fomenube.\n"); return;
    }

    //Kérdések előszűrése nehézség szerint
    buildPoolByDifficulty(minDiff, maxDiff); //data.c

    if (poolCount == 0) {
        printf("Sajnos ezen a nehezsegi szinten nincsenek elerheto kerdesek.\n");
        return;
    }

    getCategoriesFromPool(questionPool, poolCount); //data.c
    drawCategorySelection(categories, categoryCount); //ui.c

    int selectedCategories[categoryCount];
    int selectedCount = getUserSelections(selectedCategories, categoryCount);

    //Kérdések végső szűrése kategóriák szerint
    filterPoolBySelectedCategories(selectedCategories, selectedCount); //data.c
```

Játékciklus

Ha minden adat megfelel, kezdődhet a játék! A ciklus addig megy, ameddig nem találunk el 15 kérdést, vagy ameddig nem rontunk el eggyet. Kiválasztunk egy random kérdést a tömbből, majd a ui.c drawQuestionScreen funkciójával kirajzoljuk azt. Ha a felhasználó válasza megfelel a helyes válasszal, ujrakezdődik a ciklus. Ha rossz választ adott meg, akkor pedig végetér a játék.

```c
void gameLoop(const char* player, bool* fiftyFifty, bool* phone, bool* audience) {
    int questionNumber = 1;
    int correctAnswers = 0;

    while(poolCount > 0 && correctAnswers < 15) {
        int randomIndex = rand() % poolCount;
        Question* currentQuestion = questionPool[randomIndex];

        drawQuestionScreen(player, questionNumber, correctAnswers * 1000, currentQuestion, &fiftyFifty, &phone, &audience);

        //ez meg nincs kesz annyira
        char answer;
        printf("Valaszod (A/B/C/D): ");
        scanf(" %c", &answer);
        if(answer=='1') answer='A';
        else if(answer=='2') answer='B';
        else if(answer=='3') answer='C';
        else if(answer=='4') answer='D';
        while (getchar() != '\n');
        if (toupper(answer) == toupper(currentQuestion->correctAnswer[0])) {
            printf("Helyes valasz!\n");
            correctAnswers++;
            questionNumber++;
            printf("Nyomj entert a kovetkezo kerdeshez...");
            getchar();
        } else {
            printf("Rossz valasz! A helyes valasz: %s\n", currentQuestion->correctAnswer);
            printf("Nyomj entert a menübe visszatéréshez...");
            getchar();
            startGame();
            break;
        }
        questionPool[randomIndex] = questionPool[poolCount - 1];
        poolCount--;

        /*
        questionNumber++;
        printf("Nyomj entert a kovetkezo kerdeshez...");
        getchar();*/
    }

    printf("\nJatek vege! Osszesen %d helyes valaszt adtal.\n", correctAnswers);
    // updateLeaderboard meg ilyenek
}
```

Kiírások

Az ui.c-ben található függvények nem tartalmaznak semmilyen játéklogikát. Nem hoznak döntéseket, nem módosítják a játék állapotát, és nem kezelnek adatokat. Kizárólagos feladatuk a kapott adatok formázott megjelenítése a konzolon.

```c
#ifndef UI_H
#define UI_H

#include "data.h"

void drawWelcomeScreen();
void drawDifficultyMenu();
void drawQuestionScreen(const char* player, int qNum, int money, const Question* question, bool fiftyFifty, bool phone, bool audience);
void drawLeaderboard();
void drawGameOverScreen(int prize, int minutes, int seconds);
void drawCategorySelection(char** categories, int categoryCount);

#endif
```

A függvények paraméterként kapják meg a megjelenítendő adatokat.

```c
void drawDifficultyMenu() {
    system("cls");
    printf("========================================\n");
    printf("\tVÁLASSZON NEHÉZSEGI FOKOZATOT\n");
    printf("========================================\n");
    printf("1. Konnyu\n2. Kozepes\n3. Nehez\n4. Vissza\n");
    printf("----------------------------------------\n");
}

void drawQuestionScreen(const char* player, int qNum, int money, const Question* question, bool fiftyFifty, bool phone, bool audience) {
    system("cls");
    printf("Név: %s | Kérdés #%d | Nyeremény: %d\n", player, qNum, money);
    printf("Segítségek: 50:50 (%s) | Telefonos (%s) | Közönség (%s)\n",
           fiftyFifty ? "elérhető" : "felhasználva",
           phone ? "elérhető" : "felhasználva",
           audience ? "elérhető" : "felhasználva");
    printf("--------------------------------------------------------------------------\n");
    printf("%s\n\n", question->question);
    printf("A) %s\nB) %s\nC) %s\nD) %s\n", question->answer1, question->answer2, question->answer3, question->answer4);
    printf("--------------------------------------------------------------------------\n");
    printf("Válaszod: ");
}
void drawLeaderboard() {
    system("cls");
    printf("========================================\n");
    printf("\tRANGLISTA\n");
    printf("========================================\n");
    printf("Helyezés\tNév\tNyeremény\tJátékban töltött idő\n");
    printf("----------------------------------------\n");
    printf("Nyomja meg az enter a visszalépéshez");
}
```