Sasvári Péter

Matrix Racer dokumentáció

Feladatok

racer_io.c:

`Dictionary mrReadDictionary(char* filename)` */Szavak beolvasása fájlból/*

- Programunk a .dict kiterjesztésű „szótár” fájlok beolvasásával és a <string.h> header strtok függvényével a saját Words láncolt listánkba tölti a szavakat.
- A szótárfájlban pontosvesszővel vannak elválasztva a szavak.
- Ha a beolvasás során minden sikeresen zajlik, visszakapjuk a Words láncolt lista első címét.

`Words* mrReadWordList(char* content, const char* delimiter, size_t* count)` */Stringből beolvasás láncolt listába elválasztó karakter alapján/*

- Az strtok függvény segítségével az argumentumként megadott elválasztó karakter szerint tokenekre bontja a szöveget.
- Végig iterál a tokeneken, majd az összes elemeinek dinamikusan memóriát foglal, és láncolt lista típusú adatszerkezetet valósít meg.
- Visszaadja a láncolt lista fejét, és paraméterlistán visszaadja a beolvasott szavak számát.

racer_logic.c:

`char* mrRandWord(Dictionary dic)` */Véletlenszerű szó generálása/*

- A már inicializált listából ki kell választanunk szavakat. Ehhez használjuk a C pszeudo-random számgenerátorát, ami az <stdlib.h> header-ben található.
- A Words lista kezdőcímétől kezdve annyiszor lépünk a következő elem címére, amekkora számot kisorsoltunk. Túlcsordulni nem fogunk, mert az utolsó elem címe az elsőre mutat.

`Dictionary mrRandDictionary(Dictionary dic[], int difficulty)` */Véletlenszerű szótár kiválasztása/*

- 1 és 100 között kisorsolt véletlenszerű szám alapján eldönti, hogy melyik szótárt választja a következő véletlenszerű szó kiválasztására.
- A függvény a könnyebb szavakat részesíti előnyben.

`void mrRunGame()` */Játék indítása/*

- Inicializálja a játékot, egy while ciklusban egy enter billentyűlenyomásra vár, hogy elinduljon a játék, majd egy újabb enter-re, amikor végeztünk.

- Paraméterlistán visszaadja az eredményt.

`int mrEvaluate(char* challenge, char* attempt, double stime)` */Game over/*

- Befejezi a játékot, leállítja a stoppert, statisztikát készít az elért eredményekről:
  - Words Per Minute
  - Időtartam (sec)
  - Pontosság (0%-100%)
  - Pontszám: $(\text{WPM} * \text{Pontosság})^{1.5}$
- Visszatér a pontszámmal.

Típusok

Words *(racer_types.h)*

```
1. typedef struct Words
2. {
3.     char* word; //szó memóriacíme
4.     struct Words* next; //következőelem
5. } Words;
```

Dictionary *(racer_types.h)*

```
1. typedef struct Dictionary
2. {
3.     char* content; //a beolvasott fájl nyers tartalma
4.     size_t count;  //a beolvasott fájlban található szavak száma
5.     struct Words* head; //a 'content'-ből képzett lista feje
6. } Dictionary;
```

Score *(racer_types.h)*

```
 1. typedef struct Score
 2. {
 3.       size_t total_letters_typed;
 4.       size_t total_good_letters;
 5.       size_t total_good_words;
 6.
 7.       double difSecs; double difMins; // idő differencia
 8.       double words; double avgwords;  // szavak száma
 9.       double accuracy, WPM; // pontosság és gyorsaság
10.       double w_compl; // beírt szavak és a teljes kihívás szavainak aránya
11.       int    finalScore; // teljes pontszám
12. } Score;
```

Forráskód

A forráskód könyvtárában 3 .c/.h fájlpár található, illetve még egy .h fájl.

Ezek:

- `racer_main.c`

  Ez a fájl tartalmazza a main függvényt.

- `racer_io.c`

  Itt található a szótár fájlok beolvasására szolgáló függvény, amiben pontosvesszővel vannak elválasztva a rekordok. Továbbá ez a fájl felelős a toplistába való íráshoz is.

- `racer_logic.c`

A játék logikája itt van implementálva, ezek közé tartozik például a véletlenszó-generátor, ellenőrző (statisztika generáló) függvény.

- racer_types.h */csak header/*

A program által használt struktúrák definíciói itt kapnak helyet.

Words: A pontosvessző által elválasztott szavakat tartalmazó lista.

Attempt: A játékos max. 15 karakteres nevét és elért pontszámát tartalmazó struktúra.

Dictionary: A beolvasott fájl szöveges tartalma és az abból létrehozott láncolt lista feje.

Állományok

A programhoz az alábbi fájlok tartoznak:

.c/.h: *racer_main.c, racer_io.c, racer_io.h, racer_logic.c, racer_logic.h, racer_types.h*

.dict: *eng1.dict, eng2.dict, eng3.dict, hun1.dict, hun2.dict, hun3.dict*

Futtatás

A program egyszerűen lefordítható például a GCC fordítóval. Futtatás előtt győződjünk meg arról, hogy a kimeneti mappában a .dict fájlok rendelkezésre állnak!

```
gcc -Wall -Werror -std=c99 *.c -o racer.exe
```