

Morze nagyházi dokumentáció

Morze kódoló / dekódoló
Gyürüs Bence 2024/2025 tanév

Forrásfájlok

main.c

Ebben a fájlban csak a `main()` függvény van, illetve a `main` függvényben használt `FunctionEnum` típus.

functions.c

Olyan általánosan használt függvények vannak benne például, mint amivel a szabványos bemenetről lehet beolvasni vagy amiben egy dinamikusan foglalt karakterlánchoz fűz hozzá egy karaktert.

morse.c

Olyan függvények vannak a modulban amik kifejezetten a morzéhoz kapcsolódnak, mint a fájl beolvasása és kezelése valamint enkódolást és dekódolást elősegítő függvények

binary_tree.c

A bináris fához kapcsolódó függvények, mint a fa felépítése, és mindkét oldali keresés a fában valamint a fa, felszabadítása

Adatszerkezetek és struktúrák:

Bináris fa:

A program a morze kódoláshoz és dekódoláshoz bináris fát használ. A fa felépítésénél karakterenként halad minden a szótárban megtalálható karakternek megfelelő morze kódban és ha a vizsgált karakter a morze kódban `.` akkor balra, ha `-` akkor jobbra fog menni a fában és ezt használja ki az enkódolás során is, hiszen olyankor a morzekódnak megfelelően megy végig a fában, majd az utolsó Node értékét adja vissza, az lesz a keresett karakter. Visszafele kereséskor pedig megkeresi az adott karaktert a fában majd az útvonalat adja vissza ahogyan eljutott a karakterig.

Morse struktúra:

A struktúra feladata, hogy eltároljon egy adott karakternek megfelelő morze kódot. A `key` az adott karakter a `value` pedig a karakternek megfelelő morze kód. A morzekódhoz a memória dinamikusán van foglalva.

Node struktúra:

A fához használat struktúra, amiben található az adott leszármazott értéke (`value`), és az őhozzá kapcsolódó jobb (`right`) és bal (`left`) gyerekének memóriacíme. A `value` egy karaktert tartalmaz.

FunctionEnum felsorolás típus:

Ebben a felsorolás típusban tárolja a program, hogy éppen melyik funkció van kiválasztva.

Függvények

`int main(int number0fArgs, char** arguments) ./main.c`

Vezérli a programot, innen indít minden függvényt. A függvény első paramétere megadja hány elemű lesz az `arguments` karakter típusú pointer pointerre. A második paraméter a terminálból kapott paraméterek tömbje. A függvény beolvassa a fájlt a `read_Morse_From_File` függvény segítségével, amitől egy `Morse` típusú pointert fog visszakapni. A programot a `main` függvény vezérli, ezt amennyiben nem a parancssorból érkezett a bemenet akkor az `inFunction` változó értéke és a szabványos bemenetről jövő adat befolyásolja. Ha a bemenet értéke `E` akkor az `inFunction` értéke `encode` lesz, ha `D` akkor `decode` lesz, minden más esetben `none`. Ha az `inFunction` értéke `none`-től eltérő (`encode` vagy `decode`) akkor a következő szabványos bemenetről érkező adat lesz az enkódolandó vagy dekódolandó szöveg, ilyenkor meghívja a funkciónak megfelelő függvényt.

`char* add_Char_To_String(char* string1, char character) ./functions.c`

A függvény egy paraméterül kapott stringhez hozzáfűz egy szintén paraméterül kapott karaktert. Első paraméternek kap egy karakter pointer, második paraméternek egy karaktert amit majd az első paraméterül kapott `string1` **dinamikusán** foglalt pointerhez fog hozzáfűzni. Lefoglal dinamikusán megfelelő mennyiségű memóriát és átmásolja a `string1` és a `character` értékét is. Fontos, hogy a függvény első

paramétere dinamikusan foglalt karakter típusú pointer, mivel a függvény futása alatt felszabadítja ezt, majd egy új dinamikusan foglalt memóriacímet ad vissza.

```
Morse* read_Morse_From_File(char* fileName, int* length) ./morse.c
```

Beolvas egy fájlt amiből kiolvasott értékekből generál egy tömböt aminek elemei a Morse típusú struktúra. Első bemenete a függvénynek a megnyitandó fájl neve, második bemenete a majd visszaadott tömb hosszához kapott pointer. A függvény beolvassa a paraméterként kapott fájlt, ha nem található ilyen fájl akkor kiír egy hibaüzenetet, majd NULL értékkel tér vissza. Ha található a fájl akkor beolvassa és előállít belőle Morse típusú struktúrát a `create_Morse` segítségével és ebből egy tömböt csinál a `morse_Linked_List` függvényt használva amivel végül visszatér a függvény.

```
Morse create_Morse(char* string) ./morse.c
```

A függvény egy karakterláncból csinál Morse típusú struktúrát. A paraméterként kapott karakterláncot úgy bontja szét, hogy a struktúra kulcsa (`key`) az első nem space karakter lesz, majd minden ezt követő nem space karakter a string végéig az értéke (`value`) lesz a struktúrának. A `value` -hoz a következő karaktert mindig az `add_Char_To_String` függvénnyel adja hozzá

```
Morse* morse_Linked_List(Morse* pointer, int length, Morse new_Morse)
./morse.c
```

Hozzáfűz egy paraméternek kapott Morse típusú tömbhöz egy új Morse típusú struktúrát. Első paraméternek kapja a Morse típusú pointert, második paraméternek ennek a hosszát és utolsó paraméterként a hozzáfűzendő struktúrát. A függvény meghívás után lefoglal a hosszának megfelelő méterű memóriát és átmásolja a kapott tömböt bele, majd utolsó elemnek értékéül az új elemet adja.

```
Node* forward_Morse_Data(Morse* morse_Array, int length) ./morse.c
```

Ez a függvény egy segédfüggvény, melynek feladata, hogy előkészítse az adatokat a bináris fa felépítéséhez. Első paramétere egy tömb, mely Morse struktúrákat tartalmaz, második paramétere az első paraméterként kapott tömb hossza. A függvény dinamikusan lefoglalja a fa gyökerét, majd meghívja a `morse_Array` minden elemére a `build_Binary_tree` függvényt. A visszatérési értéke a fa gyökerének a pointerre.

```
void build_Binary_tree(Node* node, char* string, char value) ./binary_tree.c
```

Felépíti rekurzív módon a bináris fát. Első paraméternek megkap egy Node típusú pointert, ez lesz a bináris fa következő ága, második paraméter egy karakterlánc, ami alapján építi fel a fát és az utolsó paraméter az az érték amit a fa megfelelő ágának fog majd adni. A rekurzió addig fut amíg a második paraméternek kapott `string` értéket nem `\0`, amíg ez nem teljesül addig a függvény mindig újra hívja magát az éppen legelső (nulladik) karakterének értéke alapján a függvényt. Ha az éppen soron következő paraméter `.` akkor a `left` értékéhez fogja meghívni következőnek ha `-` akkor a `right` értékez tatózó pointert adja be a függvénynek első paraméternek. Az újra meghíváskor mindig az első elemének a pointerét adja be a függvény második paraméterének így elérve, hogy az legyen az első a következőnek meghívott függvényben. A függvénynek nincs visszatérési értéke, mivel a paraméternek beadott, előre lefoglalt pointerhez adja hozzá a további ágakat.

```
char* read_From_Input() ./functions.c
```

Addig olvas be a függvény a szabványos bemenetről amíg a felhasználó nem üt enter-t. Egyesével olvassa be a szabványos bemenetről a karaktereket amíg a beolvasott karaktert nem egyenlő `\n`-el. Addig az összes beolvasott karakterhez meghívja `add_Char_To_String` függvényt és végül visszatér a string legelső elemének pointerével.

```
void encode_Morse(Node* morse_Tree, char* read_Input) ./morse.c
```

Végigmegy a függvény a beolvasott bemenet összes karakterén és meghívja az összesre a find_Morze függvényt. Az első paraméter a felépített bináris fa, a második paraméter az enkódolandó karakterlánc. A függvény ellenőrzi, hogy található-e az adott karakter a fában, ha nem akkor kilép a ciklusból és ezzel a függvény futása is véget ér.

```
bool find_Morze(char character, Node* binary_Tree) ./morse.c
```

Segédfüggvény a reverse_Search_In_Morse_Tree rekurzív függvény meghívására. A függvény írja ki a visszakapott karaktert, kezeli a `space`-t valamit az, ha nem található a karakter a szótárban.

```
char* reverse_Search_In_Morse_Tree(Node* node, char value, const char*
morse_Code) ./binary_tree.c
```

A függvény rekurzív módon megkeresi a paraméternek kapott karaktert a bináris fában, majd visszatér karakterláncként, hogy hol találta meg a fában a karaktert. Első paraméternek kap egy Node típusú pointert, ez a fa egy ága, másodiknak a keresett értéket, harmadiknak egy eleinte üres karakterláncot ami lesz később ez elérés útvonala. Ha a paraméternek kapott `at value` értéke megegyezik a paraméternek kapott `value` karakterrel akkor a függvény visszatér a `morse_Code` változóval, ellenkező esetben bejárja a jobb és a bal ágakat is, a függvényt mindig úgy hívja meg, hogy az ág irányának megfelelően hozzáad egy `.` -t vagy egy `-`-et a `add_Char_Without_Free_Memory` függvény segítségével a `morse_Code` stringhez, ami lesz a harmadik paraméter. Az első paramétere a megfelelő ágát, a második paraméter pedig a kapott `value` lesz. A visszatérési értéke a karakternek megfelelő útvonala, amennyiben nincs ilyen karakter akkor `\0` -t ad vissza.

```
char* add_Char_Without_Free_Memory(const char* path, char new_char)
./functions.c
```

Ugyanaz, mint az `add_Char_To_String` függvény, csak az első paramétere nem kötelezően dinamikus foglalát `char` pointer, így emiatt nem csinál automatikus memória felszabadítást sem. A visszatérési értéke az összefűzött karakterlánc első karakterének pointere.

```
void decode_Morse(Node* tree, char* input) ./morse.c
```

Előkészíti a keresést a bináris fában, ellenőrzi a kapott bemenetet. Elsőnek a fa gyökerének pointerét kapja bemenetnek, majd a dekódolandó szöveget. Elsőnek `control_Morse_Code` függvénnyel ellenőrzi a szöveget, hogy csak a megengedett karakterek (`.` `-` `/`) található benne, ha ennek visszatérési értéke `false` akkor a függvény nem csinál semmit, ellenkező esetben 'szavakra' bontja a kódot, amit úgy valósít meg, hogy végigmegy a betűkön és ha nem egyenlő space-el a karakter akkor hozzáadja a `word` változóhoz a karaktereket az `add_Char_To_String` függvény segítségével. Ha a soron következő karakter space akkor megkeresi a fában a karaktert `search_In_tree`-vel, majd ki is írja az általa visszaadott karaktert és törli a `word` értéket.

```
bool control_Morse_Code(char* morse) ./morse.c
```

Ellenőrzi, hogy csak a megengedett karakterek (`.` `-` `/`) vannak-e a paraméternek kapott karakterláncban. Ha olyan karaktert talál ami nincs a megengedett karakterek között úgy a visszatérési értéke `false` máskülönben `true`.

```
char search_In_tree(Node* node, char* string) ./binary_tree.c
```

Megkeresi a rekurzívan fában a paraméternek kapott karakterláncot. Első paraméternek kap egy `Node` pointert, ez az aktuális ág. A második paraméter a dekódolandó karakterlánc amit úgy keres meg a fában, hogy ha a `string` legelső (nulladik) karaktere `-` akkor a `right`-ban tárol pointert adja be a függvény újrahívásánál, ha `.` akkor pedig a `left`-ben lévő pointert adja első bemenetnek, második paraméter mindig a `string` első indexének memóriacíme lesz. Addig megy a rekurzió amíg a `string` nem lesz egy üres karakterlánc vagy a `node` nem lesz `NULL`. A függvény visszatérési értéke ha létezik akkor a megtalált karakter.

```
void free_Tree(Node* tree) ./binary_tree.c
```

Felszabadítja a bemenetül kapott fát a memóriában.

```
void free_Morse(Morse* morse_Array, int length) ./morse.c
```

Felszabadítja a bemenetül kapott `morse_Array`-t és a `value` kulcsa alatt dinamikus foglalát karakterláncot.