Unyi Péter Álmos
Nagy házifeladat – Hufmann-kódolás
Félkész program dokumentációja

Rövid leírás:

A nagy házifeladatom egy parancssorból futtatható, szövegeket veszteségnemntesen tömörítő program, amely a Hufmann-algoritmust használja fel a tömörítésre.

Felhasznált könyvtárak és függőségek:

1. stdio.h;
2. stdlib.h;
3. stdbool.h;
4. string.h;
5. debugmalloc.h;
6. funkciok.h;
7. betomorites.

Típusok:

Gyakorisag: struct a karakterek gyakoriságának a tárolására egy szövegben;

- char betu: a karakter, aminek a gyakoriságát tárolja a .gyakorisag paraméter;
- int gyakorisag: a karakter előfordulása az adott szövegben.

Binfa: bináris fa tárolására;

- int szam: adat a bináris fa egy adatpontjában;
- struct Binfa *bal, *jobb: a bináris fa egy gyökerének a két levelére mutató pointer, melyeknek típusa szintén Binfa.

Funkciók:

FILE *fajl_letrehoz(char *fajlnev)

- Függvény fájl létrehozására. Létrehoz egy fájl, aminek az elérési útvonalát argumentumként kapta meg, majd visszatér a létrehozott filepointerrel.
- Argumentumok:
    - char *fajlnev: a fájl elérési útvonala.
- Visszatérési érték:
    - sikertelen létrehozáskor: NULL;
    - sikeres létrehozáskor: FILE *fajl: a fájlt tartalmazó pointer.

int fajlmeret(FILE *fajlp)

- Függvény egy fájl méretének a meghatározására. Eltárolja és visszatér az argumentumként kapott filepointer byteokban számolt méretével.

- Argumentumok:
    - FILE *fajlp: a fájlt tartalmazó pointer.
- Visszatérési érték:
    - int fajlmeret: a fájl mérete byteokban.

char *fajl_olvas(char *fajlnev)

- Függvény egy file tartalmának az olvasására. Megnyitja az argumentumként kapott elérési útvonalon lévő file-t és karakterenként kiolvassa belőle a tartalmát, ezt eltárolja egy char pointerben, majd ezzel tér vissza.
- Argumentumok:
    - char *fajlnev: a file elérési útvonala.
- Visszatérési érték:
    - sikertelen file megnyitáskor: NULL;
    - sikeres olvasáskor: char *string: a szöveget tartalmazó char pointer.

int fajl_ir(char *fajlnev, char *tartalom)

- Függvény egy tartalom file-ba írására. Megnyitja az argumentumként kapott elérési útvonalon található file-t és beleírja az argumentumként kapott tartalmat, majd visszatér a file-ba írt karakterek számával.
- Argumentumok:
    - char *fajlnev: a file elérési útvonala
    - char *tartalom: a file-ba írandó szöveges tartalom char pointere
- Visszatérési érték:
    - sikertelen file megnyitáskor: NULL;
    - sikeres file-ba íráskor: int kar_count: a file-ba írt karakterek száma

Gyakorisag *gyak_szamol(char *szoveg)

- Függvény egy szövegben található karakterek előforduásának a megszámolására. A függvény az argumentumként kapott szövegben megkeresi az azonos karaktereket, majd összeszámolja, hányszor találhatóak meg. Az eredményeket a karakterekkel együtt eltárolja egy Gyakorisagok típusú tömbben, és ezzel tér vissza. MÉG NINCS KÉSZ, BUG TALÁLHATÓ BENNE: HIBÁS TÖMBBEL TÉR VISSZA!
- Argumentumok:
    - char *szoveg: a szöveget tartalmazó char típusú pointer.
- Visszatérési érték:
    - üres tömb átadásakor: NULL;
    - sikeres számoláskor: Gyakorisagok *cel: a céltömb, ami tárolja a karakterek előfordulását

void bemenet_eldont(char opcio, char *arg1, char *arg2, char *arg3)

- Függvény a betömörítés kezelésére és vezérlésére. Ezt a függvényt hívja meg a huffman.c fájl main függvénye, amikor a betömörítés típusát dönti el. Egy switch

segítségével a bemeneti parancs alapján eldönti, hogy file-ból vagy vagy user inputból várja a tömörítendő szöveget. a rendezés_struct vagy rendezes_int segítségével megszámolja a betűk gyakoriságát, majd más függvények segítségével elvégzi a tömörítést. MÉG HIÁNYOS, NEM TELJES A KÓD, EGYELŐRE CSAK A BEOLVASÁS ÉS GYAKORISÁG SZÁMOLÁS TALÁLHATÓ MEG BENNE!

- Argumentumok:
    - char opcio: a user parancsa, a betömörítés iránya kapcsolóban a második karakter (file-ból vagy begépelt szövegből várja a bemenetet)
    - char *arg1: a begépelt szöveg, vagy file elérési útvonala
    - char *arg2: a kimeneti file opció parancsa, ha üres, az alapértelmezett helyre menti el a kimeneti file-t
    - char *arg3: az egyedi kimeneti file elérési útvonala
- Visszatérési érték: -

void help_screen()

- A használati utasítás képernyője, kiírja a használati opciókat és segítséget ad a program használatával kapcsolatban. -h paranccsal érhető el.
- Argumentumok: -
- Visszatérési érték: -

Gyakorisag *rendezes_struct(Gyakorisag *t, int meret)

- Függvény egy Gyakorisag típusú tömb rendezésére, a közvetlen kiválasztás algoritmusával. A .gyakorisag paraméter szerint rendezi az argumentumként kapott tömböt, kisebbtől a nagyobbig, a betűkkel együtt, majd visszatér egy ilyen típusú tömbbel.
- Argumentumok:
    - Gyakorisag *t: a rendezendő tömb;
    - int meret: a tömb mérete.
- Visszatérési érték:
    - Gyakorisag *t: a rendezett tömb.

int *rendezes_int(int *t)

- Függvény egy int tömb rendezésére, kisebbtől a nagyobbig, közvetlen kiválasztás módszerével.
- Argumentumok:
    - int *t: a számokat tartalmazó tömb.
- Visszatérési érték:
    - int *t: a rendezett tömb.

int *tombelem_torol(int *tomb, int poz, int meret)

- Függvény egy meret méretű tömb poz-adik elemének a kitörlésére. Argumentumként megkapja a tömböt és a törlés után visszatér a módosított tömbbel.

- Argumentumok:
    - int *tomb: számokból álló tömb;
    - int poz: a pozíció, ahonnan ki kell törölni az elemet;
    - int meret: a tömb mérete.
- Visszatérési réték:
    - int *tomb: a módosított tömb.

void tomb_kiir()

- Függvény egy tömb kiírására.
- Argumentumok:
    - int *n: a számokból álló tömb;
    - int meret: a tömb mérete.
- Visszatérési érték: -

Binfa *uj_adatpont(Binfa *gyoker, int szam)

- Függvény egy új Binfa típusú adatpont létrehozására az argumentumként kapott bináris fa gyökere alá. A tárolt adata a szintén argumentumként kapott int szam. Ha a kapott gyökér NULL, akkor lefoglal egy Binfa méretű területet a gyökérnek és feltölti ezt a gyökeret az adattal majd létrehoz két gyereket, jobbra és balra is. Egyébként pedig a gyerekek értékét állítja be az előző módon és visszatér a kapott bináris fával. HIBÁS ALGORITMUS, JAVÍTÁSRA SZORUL!
- Argumentumok:
    - Binfa *gyoker: a bináris fa egy adatpontjára mutató pointer;
    - int szam: az adatpont tárolt adata.
- Visszatérési érték:
    - Binfa *uj: ha a Binfa *gyoker NULL, akkor a gyökérnek lefoglalt memóriaterülettel tér vissza;
    - Binfa *gyoker: a módosított fa.

Binfa *rekurziv_tomb_osszeg(int *n, int meret)

- Függvény, amely a szövegben előforduló karakterek gyakoriságának a számaiból összegfát épít, ahol a levelek a gyakorisági értékek, a főgyökér pedig a levelek összege. Átveszi a gyakoriságokból álló rendezett tömböt és rekurzívan csinálja végig a következő ciklust: a két legkisebb elem kiválasztása és összeg képzése, ezen két elem törlése, összeg beszúrása a tömbbe, tömb rendezése, méret csökkentése egyel, függvény meghívása. Ha a tömb mérete eléri a 0-t, akkor képez egy bináris fát, majd a visszatér ezzel, és elkezdi beszúrni a tömb első két elemét (amely a rekurzív meghívás miatt mindig az a kettő lesz, amiből az összeget képeztük). Ezután a függvény visszatér a felépített összegfával. MÉG NINCS KÉSZEN, BUGOS, NEM ÉPÍTI FEL AZ ÖSSZEGFÁT!
- Argumentumok:
    - int *n: a rendezett gyakorisági értékekből álló szám tömb;
    - int meret: a tömb mérete.

- Visszatérési érték:
    - Binfa *fa: a felépített bináris összegfa.

void sorban_kiir(Binfa *gyoker)

- Függvény egy bináris fa adatainak a kiírására. A függvény rekurzívan bejárja a fát, és minden adatpontját kiírja. Ha a gyoker NULL, akkor a rekurzió megszakad az adott ágon és folytatódik a következőn.
- Argumentumok:
    - Binfa *gyoker: a bináris fára mutató pointer.
- Visszatérési érték: -