Programozás házi feladat

Illyés Dávid Gyula – ZYTGYT

A program két fájlal dolgozik az indulástól kezdve, az egyik fájlban termékek vannak felsorolva, azonosítóval, árral és névvel, a másik fájlban eladások kosarainak tartalma van felsorolva úgy, hogy minden sor egy vásárlás tartalmát reprezentálja, és ilyen sorokból akármennyi lehet a fájlban. A program először végig megy a vásárlások fájlon és minden elemnél (egy egész szám, ami egy termék azonosítója) végig megy a termékek fájl tartalmán és minden elemnél összehasonlítja az épp vizsgált azonosítót a fájlban lévő összes termék azonosítójával, ha egyezést talál akkor egy függvény használatával hozzáadja egy termékek (termek\_lista) típusú hátulról strázsált láncolt listához, ez addig megy amíg a sor végére ér. Amikor eléri a sor végét, a most elkészített termek\_lista -t hozzáadja egy szintén strázsált vásárlások (vasarlasok) típusú láncolt listához. Ezt addig ismételi amíg a vásárlások fájl végére nem ér. A beolvasás után egy függvénnyel kiszámolja minden korsár teljes értékét, és bele írja a vásárlások (vasarlasok) típusú lista elem sum\_price változójába. Miután és csak is miután lefutott a summázó függvény, kiszámolja az összes vásárlás teljes értékének az átlagát. Miután ez a két függvény lefutott megkeresi az átlaghoz legközelebbi summázott kosárértéket, és kiírja az adott kosár teljes értékét és a kosár tartalmát.

A program által feldolgozott fileokban lévő adatszerkezetek így néznek ki:

* Termékek fájl minden sora egy terméket jelent, a sorokban pedig az id (egész szám) szóköz, ár (lebegő pontos szám) szóköz, név (karakter [150]), szintaktika alapján vannak az adatok, (termekek.txt) példa:

1 500.0 husleves  
2 15000.0 gérvágó  
3 1000.0 kukazsák  
4 533.5 levegő  
5 748.4 hell energia ital  
6 6584.23 laptop  
7 6139.345 kávé  
8 23.2 dianás cukor  
9 2374.34 műzli  
10 1245.245 pohár  
11 8399.45 víz  
12 12345.843 föld

* Vásárlások fájl sorai a különböző vásárlások tételeinek id -jeivel van tele, minden sor végén van egy lezáró „n” karakter, egy sorban maximum 1000 elem van a lezáró karaktert beleszámolva, a sorok száma nincs limitálva, (vasarlasok.txt) példa:

1 5 3 6 8 42 n  
2 8 4 16 27 n  
2 4 8 10 12 n  
1 3 5 7 9 11 n

Diagram

Description automatically generated with medium confidence

Az alábbiakban le vannak írva a programban lévő fő függvények működései:

Egy eladás teljes összege (sum\_price):

A függvény a vásárlások lista fejének pointerét kéri paraméterként (vasarlasok\_prt), ez alapján végig megy fésű gerincének összes „fogán” (termek\_lista), mindegyiknél összeadja az összes termék árát egy ideiglenes double változóba majd, amikor a lista végére ér, az summázott értékre megváltoztatja az adott gerinc elem sum\_price változójának értékét. Ezt minden gerinc elemnél megcsinálja, amíg a lista végére nem ér. A függvénynek nincs visszatérési értéke.

void sum\_price(vasarlasok\_ptr head) { // a vásárlás lista fejének pointerét oda adjuk  
 double sum = 0.0; // az épp summázott termék lista ideiglenes értékének tárolója  
 for (vasarlasok\_ptr v = head; v != NULL; v = v->next) { // a vásárlások lista bejárása  
 for (termekek\_ptr t = v->termek\_lista; t != NULL; t = t->next) { // a termékek lista bejárása  
 sum += t->ar; // az összes árat hozzá adjuk a sum ideiglenes változóhoz  
 }  
 v->sum\_price = sum; // a summázott árra átálítjuk a sum\_price változót a gerincben  
 sum = 0.0;  
 }  
}

Átlag összeg (avg\_price):

A függvény egy vasarlasok\_ptr típusú pointert kér paraméterként, ez alapján végig fut az összes gerinc elemen, eközben minden elemnél számolja, hogy hány darab volt eddig és a sum\_price értékeiket egy változóba összeadja, amikor a lista végére ér a summázott értéket elosztja a darabszámmal. A visszatérési értéke egy double, ami az átlagolás értéke.

// kell mellé egy aránylag rövid tömör leírás  
  
// az avg\_price mindenképp a sum\_price fgv után szabad csak meghívni  
double avg\_price(vasarlasok\_ptr head) {  
 double sum = 0, avg = 0.0; // alapértékek megadása a gyüjtő summának és az átlagnak  
 int i = 0; //elemek darabszáma  
 for (vasarlasok\_ptr v = head; v != NULL; v = v->next) { // a vásárlások lista bejárása  
 i++; // minden vasarlasok lista elemnél hozzá adunk egyet az elem számlálóhoz  
 sum += v->sum\_price; // hozzá adjuk a jelenlegi lista elemben tárolt sum\_price értékét a sum -hoz  
 }  
 //printf("sum:%0.2f \ni:%d\n", sum, i);  
 avg = (sum/i);  
 return avg; // vissza adja a kiszámolt átlag eladási árat  
}

Az átlaghoz legközelebbi kosárérték (closest\_to\_avg):

A függvény paramértere egy vasarlasok\_ptr, ami a lista fejére mutat, végig megy a vásárlások listán és minden elemnél megvizsgálja, hogy a teljes átlag (avg\_price) a nagyobb vagy a jelenlegi lista elem értéke. A nagyobból kivonja a kissebbet, majd megnézi, hogy az így kapott érték kisebb-e, mint az előző (ha ez az első elem és az ehhez használt változó az alap nulla értékén van akkor azt felülírja), ha kisebb akkor elmenti az ahhoz a vásárláshoz tartozó termek\_lista pointert és a vásárlás értékét egy-egy ideiglenes pointerbe. Amikor a lista végére ér kiírja az átlaghoz legközelebbi kosár értékét és ki listázza az adott kosár tartalmát. Visszatérési értéke nincs a függvénynek.

// kell mellé egy aránylag rövid tömör leírás  
  
void closest\_to\_avg(vasarlasok\_ptr head) {  
 double curr\_closest = 0;  
 termekek\_ptr out;  
 vasarlasok\_ptr value;  
 double avg = avg\_price(head);  
 for (vasarlasok\_ptr tmp = head; tmp != NULL; tmp = tmp->next) {  
 if (avg < tmp->sum\_price) { // ha az átlag értéke nagyobb mint az teljes kosárérték  
 double r = fabs(tmp->sum\_price - avg); // kivonjuk az átlagot a teljes értékből majd abszolútértéket vonunk  
 if (curr\_closest == 0) { // ha még nem volt megváltoztatva a legközelebbi érték  
 curr\_closest = r; // akkor ezt most megtesszük  
 out = tmp->termek\_lista; // elmentjük a termék listát  
 value = tmp; // és a kosár értékét is ez mindenhol ugyan így néz ki  
 } else if (curr\_closest > r) { // ha az új érték kisebb mint a régi akkor az új átveszi a régi helyét  
 curr\_closest = r;  
 out = tmp->termek\_lista;  
 value = tmp;  
 }  
 } else if (avg > tmp->sum\_price) {  
 double r = fabs(avg - tmp->sum\_price);  
 if (curr\_closest == 0) { //  
 curr\_closest = r;  
 out = tmp->termek\_lista;  
 value = tmp;  
 } else if (curr\_closest > r) {  
 curr\_closest = r;  
 out = tmp->termek\_lista;  
 value = tmp;  
 }  
 }  
 }  
 printf("Az átlag kosárértékhez legközelebbi kosár értéke: %0.2f Ft, tartalma:\n", value->sum\_price);  
 for (out; out->next != NULL; out = out->next) { // kiírjuk a kosár tartalmát  
 printf("Termék név: %s, \f\tár: %0.2f Ft\n", out->nev, out->ar);  
 }  
}

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <math.h>

// kell mellé egy aránylag rövid tömör leírás  
  
termekek\_ptr push\_product(termekek\_ptr head, char nev[150], double ar) { // paraméterek  
 termekek\_ptr tp = malloc(sizeof(termekek)); // memória hely foglalás és annak címének átadása a pointernek  
 if (tp == NULL) { // ha nem tud memóriát foglalni hibát ad és kilép  
 printf("Couldn\'t allocate memory for new list item\n");  
 return 0;  
 }  
 strcpy(tp->nev, nev); // az új lista elemnek oda adjuk a termék nevét  
 tp->ar = ar; // és az árat  
 tp->next = head; // az eddigi lista fejét utánna láncoljuk az új elemnek  
 head = tp; // a lista utolsó elemének fejének címévé tesszük az új elemet  
 return head; // vissza adjuk az új kezdőcímet tp=>head  
}  
// kell mellé egy aránylag rövid tömör leírás  
  
vasarlasok\_ptr push\_purchase(vasarlasok\_ptr head, termekek\_ptr termek\_p) { // fejléc  
 vasarlasok\_ptr vp = malloc(sizeof(vasarlasok)); // memória hely foglalás és annak címének átadása a pointernek  
 if (vp == NULL) { // ha nem tud memóriát foglalni hibát ad és kilép  
 printf("Couldn\'t allocate memory for new list item\n");  
 return 0;  
 }  
 vp->next = head; // az eddigi lista fejét utánna láncoljuk az új elemnek  
 vp->termek\_lista = termek\_p; // a már elkészített termékek listát hozzáfűzzük a listához  
 head = vp; // a lista utolsó elemének fejének címévé tesszük az új elemet  
 return head; // vissza adjuk az új kezdőcímet vp=>head  
}  
// kell mellé egy aránylag rövid tömör leírás  
  
void delete\_products(termekek\_ptr head) { // egy termék lista összes elemének törlése  
 if (head != NULL) { // ha eleve nincs benne semmi akkor vissza adjuk a null pointert  
 while (head == NULL) { // amég nem érünk a lista végére  
 termekek\_ptr tmp = head->next; // a következő elem címét elmentjük egy ideiglenes termekek pointerbe  
 free(head); // felszabadítjuk a verem tetején lévő elemet  
 head = tmp; // megadjuk a verem új fej elemét  
 }  
 }  
}  
// kell mellé egy aránylag rövid tömör leírás  
  
void delete\_purchases(vasarlasok\_ptr head) { // az összes vásárlások lista elem törlése az összes termék listával együtt  
 if (head != NULL) { // ha eleve nincs benne semmi akkor vissza adjuk a null pointert  
 while (head == NULL) { // amég nem érünk a lista végére  
 vasarlasok\_ptr tmp = head->next; // a következő elem címét elmentjük egy ideiglenes termekek pointerbe  
 delete\_products(head->termek\_lista); // a termek listát átadjuk a delete\_products fgv -nek  
 free(head); // felszabadítjuk a verem tetején lévő elemet  
 head = tmp; // megadjuk a verem új fej elemét  
 }  
 }  
}

//////////////////////////////////////////////////////////////////  
//////////////////////////// BEOLVASÁS ///////////////////////////  
//////////////////////////////////////////////////////////////////  
  
///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/// FILEOK MEGNYITÁSA ///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///  
  
// kell mellé egy aránylag rövid tömör leírás  
  
vasarlasok\_ptr file\_to\_list(vasarlasok\_ptr head, char purchase\_file[], char product\_file[]) {  
 FILE \*purchase\_p = fopen(purchase\_file, "r"); // megnyitjuk a vásárlások filet szöveges olvasás módban  
 if (purchase\_p == NULL) { // ha nem sikerül megnyitni a filet akkor hibaüzenetet adunk és kilépünk  
 printf("Couldn\'t open purchases file\n");  
 return 0;  
 }  
  
 FILE \*product\_p = fopen(product\_file, "r"); // megnyitjuk a termékek filet szöveges olvasás módban  
 if (product\_p == NULL) { // ha nem sikerül megnyitni a filet akkor hibaüzenetet adunk és kilépünk  
 printf("Couldn\'t open products file\n");  
 return 0;  
 }  
  
//////////////////////////////////////////////////////////////////  
  
 int tmp\_id, eq; // ideiglenes változók deklarálása  
 double tmp\_ar;  
 char tmp\_nev[150];  
 char tmp\_line[1000], split[] = " "; // változó deklaráció  
 while (fgets(tmp\_line, sizeof(tmp\_line), purchase\_p) != EOF) { // sorról sorra olvasunk a file végéig  
 termekek\_ptr product\_list = malloc(sizeof(termekek)); // új termék lista kezdő elem  
 product\_list->next = NULL; // beállítjuk a lista végének  
 char \*char\_int = strtok(tmp\_line, split); // egy új pointerbe leválasztjuk a string első elemét  
 eq = strcmp(char\_int, "n\n"); // megnézzük hogy a sor végére értünk e  
 while (char\_int != NULL && eq != 0) { // addig megyünk a stringen keresztül amég nem találkozunk a lezáró n karakterrel  
 while (fscanf(product\_p, "%d %lf %[^\n]%\*c", &tmp\_id, &tmp\_ar, tmp\_nev) != EOF) { // végig megyünk a termékek fileon  
 if (tmp\_id == atoi(char\_int)) { // ha a termék id egyezik az eladások file épp olvasott sorában vizsgált id -val  
 product\_list = push\_product(product\_list, tmp\_nev, tmp\_ar); // ha van egyezzés az id -k között akkor az berakjuk a listába  
 break; // azért hogy ne menjen tovább értelmetlenül a ciklus ha már találtunk egyezést kilépünk, ha nincs egyezés akkor nem csinálunk semmit  
 }  
 }  
 char\_int = strtok(NULL, split); // a split változóban megadott szóközös pontokon szétválasztjuk a stringet  
 if (char\_int == NULL) // ha a vásárlások file legvégére értünk akkor kilépünk a ciklusból  
 break;  
 else  
 eq = strcmp(char\_int, "n\n"); // megnézzük hogy a sor végére értünk e  
 rewind(product\_p); // a termékek file elejére megyünk  
 }  
 if (char\_int == NULL) // ha a vásárlások file legvégére értünk akkor kilépünk a ciklusból  
 break;  
 head = push\_purchase(head, product\_list); // ha n karakterrel találkozik a while kilép és a termék lista bekerül a vásárlás gerincbe  
 }  
 fclose(purchase\_p); // bezárjuk a vásárlások filet  
 fclose(product\_p); // bezárjuk a termékek filet  
 return head;  
}