Jámbor László
L3ATGK

# Tranzisztorválogató

## Program dokumentáció

## Program ismertetése:

A program a raktáron lévő tranzisztorok párba válogatását végzi. A felhasználó megadja, hogy milyen típusú és milyen gyártmányú tranzisztorokkal szeretne dolgozni. A felhasználó megadja továbbá, hogy a beválogatott tranzisztorok áramerősítési tényezőinek (bétáinak) milyen tartományba kell esniük (pl. 250-300), valamint, hogy az egy párba válogatott tranzisztorok bétái között maximum hány százalék eltérés lehet. Több ,,rendelés" is leadható.
Pl.:
Típus: BC182
Gyártó: Texas Instruments
Megengedett legkisebb béta: 300
Megengedett legnagyobb béta: 400
Megengedett maximális eltérés (%): 10

Típus: BD139
Gyártó: ST
Megengedett legkisebb béta: 100
Megengedett legnagyobb béta: 150
Megengedett maximális eltérés (%): 5

stb…

Fontos azonban, hogy egy típus – gyártó kombináció csak egyszer rendelhető.
A program a megadott értékek alapján úgy válogatja párba a tranzisztorokat, hogy azon tranzisztorok száma, amiket nem sikerült beválogatni, minimális legyen. A program párosával kilistázza a standard outputra a párba válogatott tranzisztorok sorszámait és adatait (típus, gyártó, béta), továbbá azt is kiírja, hogy hány olyan tranzisztor van, ami a megadott típusú és gyártmányú, de nem került beválogatásra (pl. túl alacsony béta miatt). (A program csak az alábbi típusú tranzisztorokat ismeri fel: BC182, BC546, BD139, BD249. A program csak az alábbi gyártókat ismeri fel: ST, Texas Instruments. A béták egész értékek.)

## Forrásfájlok:

Két forrásfájlra van szükség a program működéséhez. Az egyik a **raktarkeszlet.txt** szöveges állomány, amely a raktáron lévő tranzisztorok információit tárolja: sorszám, típusjelzés, gyártó. A másik a **betak.txt**, ami a béta-mérés eredményeit tárolja: a tranzisztor sorszáma, a mért béta, és hogy melyik alkalmazott végezte a mérést. (Ez utóbbi a program számára közömbös információ.)

**raktarkeszlet.txt**

sorszam – int
típusjelzés – string[10] (whitespace karaktert nem tartalmazhat)
gyártó – string[20] (whitespace karaktert nem tartalmazhat)

Tagolás: sorszám;típusjelzés;gyártó\n (szóközzel elválasztva)

Példa:
000 BD249 ST
001 BC182 Texas_Instruments
002 BC182 ST
003 BD139 Texas_Instruments
004 BC182 Texas_Instruments
005 BC546 ST
stb…

**betak.txt**
sorszám – int
béta – int
név – string[50] (whitespace karaktert nem tartalmazhat)
Tagolás: sorszám;béta;alkalmazott neve\n (szóközzel elválasztva)

Példa:
000 91 Dr_Fortoloczki_Imre
001 393 Nemeth_Zoltan
002 393 Nemeth_Zoltan
003 184 Nemeth_Zoltan
004 334 Nemeth_Zoltan
005 368 Dr_Fortoloczki_Imre
stb…

## Működés leírása:

A program beolvassa a raktarkeszlet.txt-ben tárolt tranzisztorokat egy láncolt listába, és a betak.txt adatait egy másik láncolt listába. A két adathalmazt összeköti a tranzisztorok sorszáma. Ez alapján a program minden tranzisztorhoz hozzárendeli a bétáját. Ezután a bétákat tartalmazó lista felszámolásra kerül, mivel többé nincs rá szükség. Létrehozásra kerülnek üres láncolt listák – az összeválogatást követően itt lesznek tárolva a tranzisztor-párok. Ezután a standard outputon ,,bejelentkezik” a program, és bekéri a felhasználótól egy tranzisztor adatait: típus, gyártó, megengedett minimális és maximális béta, megengedett maximális eltérés. Helytelen kitöltés esetén a program felszólítja a felhasználót az adatok újbóli megadására. Helyes kitöltés esetén a program elvégzi a párba válogatást az adott tranzisztorra. (Lejjebb bővebben kifejtve.) Ezután a felhasználónak lehetősége van újabb tranzisztor adatait megadni, valamint kérheti az eredmények kiírását. Amennyiben a felhasználó az eredmények kiírását kéri, a program párosával kilistázza a standard outputra a tranzisztor-párokat (sorszám, típus, gyártó, béta), külön-külön táblázatban ábrázolva a különböző típusú/gyártójú tranzisztorokat. A szortírozás és a párba válogatás során minden részfeladatnál eltárolásra kerül, hogy hány olyan tranzisztor volt, mely a megadott típusú és gyártójú volt, de nem került beválogatásra (pl. túl alacsony béta miatt). Ezeket az értékeket a

program összegzi, és legvégül kiírja az összeget a standard outputra – ennyi tranzisztort nem sikerült beválogatni a megadott típusúak/gyártmányúak közül.

A szortírozás és a párba válogatás algoritmusa:

A program az összeválogatandó tranzisztorok számára ideiglenesen létrehoz egy üres láncolt listát. Bejárja az összes tranzisztort tartalmazó listát, és annak azon elemeit, amik megfelelő típusúak és gyártmányúak, valamint bétáik a felhasználó által megadott tartományba esnek, beláncolja az újonnan létrehozott listába. Ezt a részfeladatot az **int szortiroz(…)** függvény végzi. Ezután ezen lista elemeit a **void sorbarendez(…)** függvény béta szerint növekvő sorrendbe rendezi közvetlen kiválasztással. Ezután megtörténik a tényleges párba válogatás. Ezt az **int parba_valogat(…)** függvény végzi. A függvény megkeresi a (már szortírozott és rendezett) tranzisztorokat tároló lista középső elemét (páros elemszám esetén a két középső közül a nagyobbik tekintendő középsőnek), majd az attól balra, illetve jobbra eső két olyan elemet, melyekre igaz, hogy a bétájuk és a középső elem bétájának különbsége maximális, de az eltérés még a felhasználó által megadott p%-on belül van. Ezek után a középső elemet, és a megtalált bal és jobb oldali elemek egyikét (azt, amelyiknek az eltérése nagyobb a középső elemhez képest) beláncolja a tranzisztor-párok számára korábban létrehozott listába. (A cél a lehető legtöbb pár létrehozása a megadott feltételek betartása mellett.) Amennyiben az aktuális középső elemhez nem talál a függvény egyetlen másik elemet sem – azaz bal, és jobb oldali elemnek is a középső elem adódik - a függvény eltávolítja a listából az adott elemet és eggyel növeli az n változó értékét, mely az számlálja, hogy olyan tranzisztor volt, melynek nem sikerült párt találni.

## Adattárolás:

```
/*tranzisztor struktúrája*/
typedef struct tranzisztor
{
    char sorszam[6];
    char tipus[6];
    char gyarto[20];
    int beta;
}tranzisztor;

/*sorszám - béta pár struktúrája*/
typedef struct sorszam_es_beta
{
    char sorszam[6];
    int beta;
}sorszam_es_beta;

/*raktarkeszlet.txt-t ilyen elemekből álló
láncolt listába olvassuk be*/
typedef struct raktarkeszlet
```

```
{
    tranzisztor t;
    struct raktarkeszlet *next;
}raktarkeszlet;

/*betak.txt-t ilyen elemekből álló
láncolt listába olvassuk be*/
typedef struct betak
{
    sorszam_es_beta b;
    struct betak *next;
}betak;

/*az összeválogatott párok listája ilyen elemekből áll*/
typedef struct parok
{
    tranzisztor t1, t2;
    struct parok *next;
}parok;
```

Az összes láncolt lista kétstrázsás, egy irányba láncolt.

## Főbb függvények:

**int raktarkeszlet_beolvas(char *fajlnev, raktarkeszlet *head, raktarkeszlet *tail)**

Beolvassa a megadott (szöveges) fájlt, és befűzi a tranzisztorokat a megadott listába. Sikertelen fájlmegnyitás esetén 1-et, sikeres fájlmegnyitás esetén 0-t ad vissza.

**int beta_beolvas(char *fajlnev, betak *head, betak *tail)**

Beolvassa a megadott (szöveges) fájlt, és befűzi a tranzisztorokat a megadott listába. Sikertelen fájlmegnyitás esetén 1-et, sikeres fájlmegnyitás esetén 0-t ad vissza.

**void beta_hozzarendel(raktarkeszlet *raktar_head, betak *betak_head)**

A raktar_head és a betak_head listát is bejárva sorszám alapján a raktar_head lista összes tranzisztorához hozzárendeli a bétáját. Ezután felszabadítja a betak_head által elfoglalt memóriát.

**int szortiroz(int min_beta, int max_beta, raktarkeszlet *raktar_head, raktarkeszlet *osztaly_head, raktarkeszlet *osztaly_tail, char *tipus, char *gyarto)**

Az *osztaly_head* listába beláncolja a *raktar_head* lista azon elemeit melyekre igaz, hogy *tipus* típusúak, és gyártó gyártmányúak, valamint a bétájuk >= min_beta és <= max_beta. Visszaadja azon tranzisztorok számát, melyek *tipus* típusúak, és gyártó gyártmányúak, de a bétájuk túl magas vagy túl alacsony, így a szortírozás során kihullottak.

### void sorbarendez(raktarkeszlet *head)

Közvetlen kiválasztással rendezi a *head* lista elemeit.

### int parba_valogat(parok *parok_head, parok *parok_tail, raktarkeszlet *head, double max_elteres)

Megkeresi *head* középső elemét (páros elemszám esetén a két középső közül a nagyobbik tekintendő középsőnek), majd az attól balra, illetve jobbra eső két olyan elemet, melyekre igaz, hogy a bétájuk és a középső elem bétájának különbsége maximális, de az eltérés még <= *max_elteres*. Ezek után a középső elemet, és a megtalált bal és jobb oldali elemek egyikét (azt, amelyiknek az eltérése nagyobb a középső elemhez képest) beláncolja a *parok_head* által mutatott listába, és eltávolítja a *head* listából. (A cél a lehető legtöbb pár létrehozása a megadott feltételek betartása mellett.) Amennyiben az aktuális középső elemhez nem talál a függvény egyetlen másik elemet sem – azaz bal, és jobb oldali elemnek is a középső elem adódik - a függvény eltávolítja a listából az adott elemet és eggyel növeli az n változó értékét, mely az számlálja, hogy olyan tranzisztor volt, melynek nem sikerült párt találni. A függvény az n változót adja vissza.

### void listat_kiir(parok *head)

Táblázatos formában kiírja a *head* lista tranzisztorpárajainak adatait a standard outputra. Először kiírja a táblázat fejlécét. Ezután adott estben módosítja a tranzisztorok stringjeinek tartalmát az igényesebb megjelenítés érdekében (pl. a '_' karaktert szóközre cseréli). Végül az adatolat sorszám-típus-gyártó-béta sorrendben kiírja. A tranzisztor-párokat szaggatott vonallal választja el egymástól.

### void vegrehajt(raktarkeszlet *raktar_head)

Minden tranzisztor – gyártó kombinációhoz létrehoz egy tranzisztorpárok tárolására alkalmas üres láncolt listát. Ezután kommunikál a felhasználóval, bekéri egy tranzisztor adatait. Helytelenül megadott adatok esetén az adatok újbóli megadását kéri a felhasználótól. Helyesen megadott adatok esetén átmenetileg létrehoz egy tranzisztorok tárolására alkalmas üres láncolt listát. Az **int szortiroz(…)** függvényt az adott típusú és gyártmányú tranzisztorra meghívja, *osztaly_head*-nek és *osztaly_tail*-nek a fent elmlített üres

listát megadva. Ezután a **sorbarendez(…)** függvényt meghívja, átadva annak ezt a listát. Ezután meghívja erre a **parba_valogat(…)** függvényt, melynek átadja ezt a listát, mint *head*, és a megfelelő előre létrehozott listát, mint *parok_head* és *parok_tail.* Ezután felkínálja a felhasználónak az újabb tranzisztor hozzáadásának, illetve az eredmények kiírása esetén. Ha a felhasználó az újabb tranzisztor hozzáadását választja, a fent leírt folyamat az új adatokkal is lejátszódik. Ha a felhasználó az eredmények kiírását kéri, akkor a függvény meghíja a **listat_kiir(…)** függvényt, azon típus – gyártó kombináció(k) lisáját/listáit átadva, mely(ek)et a felhasználó megadott. Emellett összegzi azon tranzisztorok számát, melyeket nem sikerült beválogatni (**szortriroz(…)** és **parba_valogat(…)** függvények visszatérési értékei), majd kiírja az összeget a standard outputra. Végül felszabadítja a listák által elfoglalt memóriát.

**int main(void)**

Létrehozza a raktárinformáció és a béta-mérés eredményeinek tárolására alkalmas láncolt litákat. Meghívja a **raktarkeszlet_beolvas(…)**, a **beta_beolvas(…)** és a **beta_hozzarendel(…)** függvényeket. Ha raktarkeszlet.txt vagy a betak.txt nem nyitható meg, hibát jelez a felhasználó felé, és visszatér nullával. Meghívja a **vegrehajt(…)** függvényt.

További, elemi feladatokat végző függvények is vannak a programban. Pl. lista középső elemének megkeresése, elem törlése listából stb.

## Tesztelési dokumentáció:

A program tesztelését a fent említett formátumú, random generált, 1000 tranzisztor adatait tartalmazó forrásfájlokkal végeztem. A bétákat tartalmazó fájl úgy lett generálva, hogy az egyes tranzisztortípusokhoz tartozó béta a gyárilag megadott tartományba esik, azaz:

BC182: 100-480

BC546: 110-450

BD139: 40-250

BD249: 25-100

A tesztelés során a program működése szélsőséges adatok megadása (pl. megengedett maximális eltérés = 0%, 100%; legkisebb megengedett béta = legnagyobb megengedett béta,

stb), illetve helytelen kitöltés (pl. elgépelés, üresen hagyott válaszmező) esetén is megfelelőnek bizonyult.