Minta első NZH

Czirkos Zoltán · 2017.10.08.
Minta nagy ZH, előző évek feladataiból összeollózva.

Minta nagy ZH, néhány előző évekből származó feladatból összeállítva. Tudnivalók a feladatsor felhasználásával kapcsolatban: lásd itt. Ez *csak minta*. Nem minden évben ugyanakkor volt az első NZH, nem minden évben pont ugyanaddig terjedt az első NZH anyaga. A pontozás is minta, tájékoztató jellegű.

Ötöslottó

BEUGRÓ: Írj főprogramot, amelyik létrehoz egy tömböt öt lottószám számára, és a **`lotto()`** nevű függvényt meghívja rá! A függvény dolga lesz feltölteni a tömböt. Írd ki ezután a számokat a képernyőre!

Írd meg a **`lotto()`** függvényt, amelyik az ötöslottó sorsolását szimulálja, vagyis a paraméterként kapott tömbbe tesz öt, egymástól különböző, 1 és 90 közötti, véletlenszerűen választott egész számot! (A program próbálkozhat többször is a számsor előállításával.)

Egészítsd ki ezeket teljes programmá! Biztosítsd, hogy ne mindig ugyanazt az öt számot sorsolja a program!

Megoldás

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>

void lotto(int *szamok) {
    int i, j;
    for (i = 0; i < 5; ++i) {
        bool sorsol = true;
        while (sorsol) {
            szamok[i] = rand()%90 + 1;
            sorsol = false;
            for (j = 0; j < i; j++)
                if (szamok[j] == szamok[i])
                    sorsol = true;
        }
    }
}

int main(void) {
    srand(time(NULL));

    int szamok[5];
    int j;
    lotto(szamok);
    for (j = 0; j < 5; j++)
        printf("%d ", szamok[j]);

    return 0;
}
```

Pontozás, beugró értékelése:

- BEUGRÓ: a főprogramban tömb létrehozása, függvény hívása, és a tömböt bejáró ciklus helyessége.
- 1 p, fejlécfájlok, a 4-ből legalább 3 legyen meg.
- 5 p lotto(), ebből 1 p fejléc, 1 p újra próbálkozás, 1 p véletlenszám generálás, 2 p keresés.
- 4 p main(), ebből 1 p tömb, 1 p fv hívása, 1 p kiírás, 1 p srand (csak a főprogramban).

Időpontok

Definiálj típust, amelyben egy időpontot tudsz tárolni, külön óra (0...23), perc (0...59) és másodperc (0...59) értékekkel!

Írj függvényt, amelyben paraméterként egy sztringet veszel át, benne egy időponttal! Értelmezd a sztringet, majd add vissza az időpontot az előbb definiált típussal! A sztringben az időpont az alábbi három forma egyikében lesz:

- **`23:17:06`** – óra, perc, másodperc, mindegyik két számjeggyel;
- **`15h 09m 53s`** – itt is óra, perc, másodperc, mindegyik két számjeggyel;
- **`10:15 AM`** – itt az óra és a perc két-két számjeggyel, a másodperc pedig nincs megadva, 0-nak kell tekinteni. Ez a formátum 12 órás; **`12:00 AM`** = éjfél, **`08:00 AM`** = reggel 8, **`12:00 PM`** = dél, **`05:00 PM`** = a délután 5 órai tea időpontja, azaz 17 óra.

Írj főprogramot, amelyben létrehozol három időpontot tároló változót, és a megírt függvényt használva feltöltöd őket a fenti példaidőpontokkal!

Megoldás

Alább két megoldás is látható, de természetesen elég volt egy megoldást adni. Az első a **`sscanf()`** függvényen alapul; az jelzi, hogy hány számot sikerült beolvasni. Ha a formátum stimmel, akkor ez 3 kell legyen. A második megoldás nyersebb, közvetlenül a karakterekkel dolgozik. A lényege ennek is az, hogy valahogyan fel kell ismerni, melyik formátumról van szó. Ha a sztring ötödik karaktere egy kettőspont, akkor csak az első lehet; ha a sztring második karaktere egy h betű, akkor pedig a második.

```
#include <stdio.h>

typedef struct Idopont {
    int ora, perc, masodperc;
} Idopont;

Idopont megoldas1(char *szoveg) {
    Idopont i;
    char ampm;
    if (sscanf(szoveg, "%d:%d:%d", &i.ora, &i.perc, &i.masodperc) == 3)
        return i;
    if (sscanf(szoveg, "%dh %dm %ds", &i.ora, &i.perc, &i.masodperc) == 3)
        return i;
    sscanf(szoveg, "%d:%d %cM", &i.ora, &i.perc, &ampm);
    i.masodperc = 0;
    i.ora %= 12;
    if (ampm == 'P')
        i.ora += 12;
    return i;
}

Idopont megoldas2_seged(char *ora, char *perc, char *mperc) {
    Idopont i;
    i.ora = (ora[0]-'0') * 10 + (ora[1]-'0');            /* vagy stdlib.h atoi() */
    i.perc = (perc[0]-'0') * 10 + (perc[1]-'0');
```

```
    i.masodperc = (mperc[0]-'0') * 10 + (mperc[1]-'0');
    return i;
}

Idopont megoldas2(char *szoveg) {
    Idopont i;
    if (szoveg[5] == ':')
        return megoldas2_seged(szoveg+0, szoveg+3, szoveg+6);
    if (szoveg[2] == 'h')
        return megoldas2_seged(szoveg+0, szoveg+4, szoveg+8);
    i = megoldas2_seged(szoveg, szoveg+3, "00");
    i.ora %= 12;
    if (szoveg[6] == 'P')
        i.ora += 12;
    return i;
}

int main() {
    Idopont a, b, c;
    a = megoldas2("23:17:06");
    b = megoldas2("15h 09m 53s");
    c = megoldas2("10:15 AM");
    return 0;
}
```

A pontozás, az alábbi részfeladatok helyes megoldására:

- 1 p, struktúra definíciója, struktúra típusú változó létrehozása
- 1 p, char 'x' és sztring "x", aposztróf/idézőjel helyes használata
- 2 p, sztring átadása függvénynek, sztringek helyes kezelése (indexelés stb.)
- 2 p, algoritmus: időpont formátumok megkülönböztetése
- 3 p, számok kiszedése a sztringekből (bármilyen módszerrel); am/pm kezelése
- 1 p, főprogram, változók létrehozása, saját függvény helyes használata

Súlyos hibák:

- Sztringek helytelen kezelése
- Lezáró nulla fogalmának nem ismerete, sztring méretének átadása
- Túlindexelés
- Kódduplikáció

Továbbá az általános pontozási irányelvek.

Szállóvendégek

Egy hétemeletes szállodában a szobafoglalásokat tömbben tárolják. A szobák a szokásos módon vannak számozva, a százasok adják meg az emeletet, a többi pedig a szoba sorszámát (pl. 712 = 7. emelet, 12. szoba). A földszint a 0. szint, utána 1-től 7-ig az emeletek. Ennél a feladatnál nem kell teljes programot írni, csak a megadott részeket.

- BEUGRÓ: Definiálj **`Vendeg`** nevű típust, amelyik egy szállóvendég adatait (név: max. 50 karakter, szobaszám: egész) tartalmazza! Írj függvényt, amely átvesz egy vendéget, és visszaadja, hogy melyik emeleten lakik!
- Írj függvényt, amely átvesz egy **`Vendeg`** elemekből álló tömböt és egy nevet! Keresse ez meg a névhez tartozó foglalást és adja vissza a megtalált tömbelem címét vagy **`NULL`**-t, ha nincs találat!
  Írj függvényt, amely paraméterként kapja a vendégek tömbjét és egy másik, inicializálatlan tömböt, amelyet a szint sorszámával indexelünk! Írja be az utóbbi tömbbe, hogy az egyes emeleteken hány vendég lakik! Írj függvényt, amely megkapja a vendégek tömbjét, az előző függvénnyel előállítja a betöltöttségek tömbjét, és végül visszatér a legzsúfoltabb emelet sorszámával – tehát azzal, ahol a legtöbb vendég van éppen!

🗁 **Megoldás**

```
typedef struct Vendeg {
    char nev[50+1];
    int szobaszam;
} Vendeg;

int emelet(Vendeg v) {
    return v.szobaszam / 100;
}

Vendeg *keres(Vendeg *vendegek, int meret, char *nev) {
    int i;
    for (i = 0; i < meret; ++i)
        if (strcmp(vendegek[i].nev, nev) == 0)
            return &vendegek[i];
    return NULL;
}

void betoltottseg(Vendeg *vendegek, int meret, int *szintek) {
    int i;
    for (i = 0; i <= 7; ++i)
        szintek[i] = 0;
    for (i = 0; i < meret; ++i)
        szintek[emelet(vendegek[i])] += 1;
}

int legtobb_vendeg(Vendeg *vendegek, int meret) {
    int szintek[8];
    betoltottseg(vendegek, meret, szintek);
    int i, max = 0;
    for (i = 1; i < 8; ++i)
        if (szintek[i] > szintek[max])
            max = i;
    return max;
}
```

Pontozás, beugró értékelési szempontjai:

- BEUGRÓ: legyen struktúra, tartalmazzon egy sztringet és egy egész számot. (Ha 50-es a karaktertömb, akkor beugrónak elfogadható.) Legyen emelet függvény, amelynek a fejléce legyen rendben.
- 1 p, struktúra (typedef nem kell). Ha 50 a tömb mérete, nem 51, nem jár a pont.
- 1 p, emelet(). Simán **`szobaszám / 100`**, mert egész osztás van.
- 2 p, keresés: 1 p algoritmus, 1 p sztring összehasonlítás (strcmp).
- 3 p, betöltöttség: 1 p fejléc, 1 p nullázás, 1 p hisztogram. A vendégtömb méretét át kell adni, a szinttömb méretét nem, de nem baj, ha ott van. A függvény visszaadhatja az **`int *szintek`**-et, de fölösleges. Nem a függvény dolga létrehozni a szintek tömbjét!
- 3 p, legtöbb vendég: 1 p fejléc és visszatérési érték, 1 p tömb létrehozása + betoltottseg() hívása, 1 p maximum keresése. A maximum keresésében az indexet kell megjegyezni, mert az adja az emelet számát.

> Súlyos hibák:
>
> - Tömbök/pointerek keverése. Pl. nem lehet a betöltöttség függvényben a szintek tömbjének létrehozása.
> - Tömb végén NULL, EOF és hasonló konstansok feltételezése.
> - Túlindexelés.
>
> Továbbá az általános pontozási irányelvek.

Rendezés

BEUGRÓ: Írj függvényt, amely képes megcserélni két paraméterként kapott, valós típusú változó tartalmát!

Írj függvényt, amely paraméterként egy valósakat tartalmazó tömböt vesz át, továbbá két indexet, amelyek egy tartomány elejét és végét határozzák meg! Térjen vissza a függvény a megadott tartományból a legkisebb elem indexével!

Írj függvényt, amely paraméterként egy valós tömböt vesz át, és az előbbi két függvényt is felhasználva növekvő sorba rendezi a tömb számait!

Egészítsd ki ezeket egy főprogrammal, amely létrehoz egy 5 elemű tömböt, rendezi azt, majd kiírja a tömbelemeket sorszámozva!