JEGYZŐKÖNYV

Operációs rendszerek BSc

2021 tavasz féléves feladat

Készítette: **Hartman Ákos Bálint**

Neptunkód: **GZ6MDY**

A feladat leírása:

16. Írjon egy olyan C programot, mely
egy fájlból számpárokat kiolvasva meghatározza a legnagyobb közös osztóját.
A feladat megoldása során használjon nevesített csővezetéket,
valamint a kimenet kerüljön egy másik fájlba.
A kimeneti fájl struktúrája kötött!

Példa a bemeneti és kimeneti fájl struktúrájára:

Bemeneti fájl:
i (Ez jelzi a számpárok darabszámát)
x y

Kimeneti fájl(Az x,y jelzi a bemeneti adatokat a z pedig a kimenet eredményét):
x y z

A feladat elkészítésének lépései:

1.

Létrehozzuk a main függvényt, majd implementáljuk az input tömböt, ami tartalmazni fogja a beolvasott számokat, illetve a számpárok számát tartalmazó integer változót.

```
int main()
{
    int input[2];

    int szamparok;
```

2.

File bolvasás, és hibakeresés!

Valamint a számpárok számának bekérése.

```
    FILE *fp = fopen("bemenet.txt","r");

    if (fp < 0){
        perror("Hiba üres a file");
        exit(-1);
    }
    fscanf(fp,"%d",&szamparok);
```

3.

Implementáljuk a megoldás változót, illetve a 2 dimenziós tömböt.

For ciklussal beolvassuk a file-ból a számpárokat, és belementjük az input tömbbe.

majd meghívjuk a megoldás változóra az „lnko” függvényt!

Illetve belementjük a tömbbe a számpárokat, és a megoldást is.

```
    int megoldas;
    int tomb[5][3];
    printf("A fileban talalhato egyenletek szama: %d \n",szamparok);
    for(int i = 0; i < szamparok; i++){
         for(int k = 0; k < 2; k++){
             fscanf(fp,"%d",&input[k]);
         }

         megoldas = lnko(input[0], input[1]);
         tomb[i][0] = input[0];
         tomb[i][1] = input[1];
         tomb[i][2] = megoldas;

     }
```

4.

Az „lnko” függvény:

```
int lnko(double x, double y){


  if (x == 0) {
    return y;
  }

  while (y != 0) {
    if (x > y)
      x = x - y;
    else
      y = y - x;
  }

    return x;


}
```

5.

Létrehozzuk a csővezetéket, illetve feltöltjük bele a kétdimenziós tömböt, majd lezárjuk a csővezetéket!

```
    int fd;
    // FIFO fájl helye

    char * myfifo = "/tmp/myfifo";

    // Nevesített csővezeték létrehozása (FIFO)

    mkfifo(myfifo, 0666);

    // FIFO megnyitása csak írásra

    fd = open(myfifo, O_WRONLY);

    write(fd, &tomb, sizeof(tomb));

    close(fd); // Csővezeték lezárása
```

6.

Létrehozzuk a „reciver.c”-t benne a main függvényt, amiben implementáljuk a kétdimenziós tömböt, és megnyitjuk a csővezetéket, kiolvassuk a kétdimenziós tömböt belőle, és elmentjük a változónkba, majd lezárjuk a csővezetéket.

```
int main()

{

    int fd1;

    int tomb[5][3];

    // FIFO fájl helye

    char * myfifo = "/tmp/myfifo";

    // Nevesített csővezeték létrehozása (FIFO)

    mkfifo(myfifo, 0666);

    fd1 = open(myfifo, O_RDONLY);

    read(fd1, &tomb, sizeof(tomb));

    close(fd1);     // Csővezeték lezárása
```

8.

Majd létrehozzuk a kimeneti fájlt, amibe kiírjuk az adatokat, amit a végén lezárunk.

```
        FILE *file_to_write = fopen("vegeredmeny.txt","a");

    if (file_to_write < 0){
        perror("Hiba üres a file");
        exit(-1);
    }

    printf("Sikeres file kiiras!\n");

    for(int i = 0; i < 5; i++)
    {
        fprintf(file_to_write, "%d meg %d lnkoja = %d\n", tomb[i][0], t
    }

    fclose(file_to_write);
```

9.

Létrehozzuk a „bemenet.txt” file-t.

```
1 5
2 20 12
3 14 5
4 20 30
5 15 3
6 20 24
```

A futtatás eredménye:

Lefuttatjuk először a main.c-t, majd a reciever.c-t!

```
hartman@hartman-VirtualBox:~/Desktop/hazi$ gcc main.c -o main
hartman@hartman-VirtualBox:~/Desktop/hazi$ ./main
A fileban talalhato egyenletek szama: 5
hartman@hartman-VirtualBox:~/Desktop/hazi$
```

```
hartman@hartman-VirtualBox:~/Desktop/hazi$ gcc reciever.c -o reciever
hartman@hartman-VirtualBox:~/Desktop/hazi$ ./reciever
Sikeres file kiiras!
hartman@hartman-VirtualBox:~/Desktop/hazi$
```

A filekiírás vége:

```
20 meg 12 lnkoja = 4
14 meg 5 lnkoja = 1
20 meg 30 lnkoja = 10
15 meg 3 lnkoja = 3
20 meg 24 lnkoja = 4
```