10.05.2022

PROGRAMARE PROCEDURALĂ
– LABORATOR NR. 12 –

1. Considerând o imagine color în format bitmap (BMP) pe 24 de biți, scrieți un program care să o transforme în tonuri de gri (*grayscale*).

Rezolvare:

Pentru a transforma un pixel color (RGB) într-unul în tonuri de gri se va înlocui valoarea fiecărui canal de culoare cu media aritmetică a valorilor intensităților luminoase inițiale. De exemplu, pixelul color (100, 210, 172) se va transforma în (160, 160, 160), unde 160 = [(100+210+172)/3].

```
#include<stdio.h>

typedef struct
{
    //deoarece imaginile BMP sunt memorate folosind
    //little-endian, inseamna ca valorile R, G si B
    //sunt memorate invers
    unsigned char Blue, Green, Red;
}Pixel;

int main()
{
    unsigned int width, height, padding, bits_per_pixel;
    unsigned int linie, coloana;
    Pixel pcrt;
    int ma;

    //se va modifica imaginea data, ci nu se crea
    //o noua imagine
    FILE* fimg = fopen("goldhill.bmp", "rb+");

    //latimea imaginii in pixeli se gaseste
    //incepand cu octetul 18 pe 4 octeti (unsigned int)
    fseek(fimg, 18, SEEK_SET);
    fread(&width, 4, 1, fimg);
    printf("Latimea imaginii in pixeli: %u\n", width);

    //inaltimea imaginii in pixeli se gaseste
    //incepand cu octetul 22 pe 4 octeti (unsigned int),
    //adica imediat dupa latimea sa in pixeli
    fread(&height, 4, 1, fimg);
    printf("Inaltimea imaginii in pixeli: %u\n", height);
```

```
//numarul de biti alocati pentru un pixel se gaseste
//incepand cu octetul 28 pe 4 octeti (unsigned int)
fseek(fimg, 28, SEEK_SET);
fread(&bits_per_pixel, 4, 1, fimg);
printf("Numarul de biti per pixel: %u\n", bits_per_pixel);

//calculam padding-ul fiecarei linii de pixeli,
//afland numarul total de pixeli de pe o linie
//1 pixel = 3 octeti (RGB)

if(((bits_per_pixel / 8) * width) % 4 == 0)
    padding = 0;
else
    padding = 4 - ((bits_per_pixel / 8) * width) % 4;

printf("Padding-ul unei linii din imagine in octeti: %u\n", padding);

//sar peste cei 54 de octeti din zona de header
fseek(fimg, 54, SEEK_SET);

//parcurg imaginea pixel cu pixel si citesc in structura pcrt
//cate un grup de 3 octeti, corespunzator unui pixel color RGB,
//dupa care inlocuiesc valorile celor 3 componente ale
//variabilei pcrt cu media lor aritmetica
for(linie = 0; linie < height; linie++)
{
    //parcurg pixelii de pe linia curenta
    for(coloana = 0; coloana < width; coloana++)
    {
        //citesc pixelul curent in variabile pcrt
        fread(&pcrt, sizeof(Pixel), 1, fimg);

        //inlocuiesc valorile RGB ale pixelului curent
        //cu media lor aritmetica
        ma = (pcrt.Blue + pcrt.Green + pcrt.Red) / 3;
        pcrt.Blue = pcrt.Green = pcrt.Red = ma;

        //ma intorc la inceputul pixelului curent
        fseek(fimg, -sizeof(Pixel), SEEK_CUR);

        //scriu in fisier noua valoare a pixelului
        fwrite(&pcrt, sizeof(Pixel), 1, fimg);



        //golesc buffer-ul de scriere pentru a putea
        //efectua urmatoarea citire
        fflush(fimg);
    }

    //sar peste octetii de padding de la sfarsitul liniei curente
    fseek(fimg, padding, SEEK_CUR);
```

```
    }

    fclose(fimg);

    return 0;
}
```

2. Scrieți funcții care să permită efectuarea operațiilor uzuale cu fracții și un program de test.

Rezolvare:

```
#include<stdio.h>
#include<math.h>

typedef struct
{
    int a,b; //consideram fractia a/b
} Fractie;

Fractie citire_fractie()
{
    Fractie f;

    printf("Numarator: ");
    scanf("%d",&f.a);

    printf("Numitor: ");
    do
    {
        scanf("%d",&f.b);
    }
    while(f.b == 0);

    if(f.a <= 0 && f.b < 0)
    {
        f.a = -f.a;
        f.b = -f.b;
    }
    else
        if(f.a >= 0 && f.b < 0)
        {
            f.a = -f.a;
            f.b = -f.b;
        }

    simplificare(&f);

    return f;
}
```

```
void afisare_fractie(Fractie z)
{
    printf("%d / %d\n", z.a, z.b);
}

void simplificare(Fractie *f)
{
    int x = abs(f->a);
    int y = abs(f->b);

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

    f->a /= x;
    f->b /= x;
}

Fractie produs(Fractie x, Fractie y)
{
    Fractie rez;

    rez.a = x.a * y.a;
    rez.b = x.b * y.b;

    simplificare(&rez);

    return rez;
}

int main()
{
    Fractie x = citire_fractie();
    Fractie y = citire_fractie();

    printf("\nPrima fractie:\n");
    afisare_fractie(x);

    printf("\nA doua fractie:\n");
    afisare_fractie(y);

    Fractie p = produs(x, y);

    printf("\nProdusul celor doua fractii:\n");
    afisare_fractie(p);

    return 0;
}
```

3. Fișierul text *test.txt* conține cuvinte despărțite prin semnele de punctuație uzuale, pe mai multe linii. Scrieți un program care să afișeze pe ecran cuvintele din fișier în ordinea descrescătoare a frecvențelor lor.

4. Definiți o structură numită *Angajat* care să permită memorarea numelui, vârstei și salariului unui angajat. Fișierul *angajati.txt* conține pe prima linie un număr natural nenul *n* și pe următoarele *n* linii informații despre câte un angajat, despărțite prin virgule. Scrieți un program care să încarce datele din fișierul text într-un tablou unidimensional alocat dinamic și să afișeze următoarele informații:

a) informațiile despre un angajat al cărui nume se citește de la tastatură (pe ecran);
b) salariul maxim și numele angajaților care au salariul respectiv (pe ecran);
c) salariul mediu din firmă (pe ecran);
d) angajații sortați alfabetic după nume (în fișierul text *angajati_nume.txt*).