03.05.2022

PROGRAMARE PROCEDURALĂ
– LABORATOR NR. 11 –

1.

a) Scrieți o funcție `txt2bin` care preia numerele întregi dintr-un fișier text și le scrie într-un fișier binar (funcția are doi parametri, respectiv calea fișierului binar și calea fișierului text).

Rezolvare:

```
//f_text = calea fisierului text  (de intrare)
//f_bin  = calea fisierului binar (de iesire)
void txt2bin(char *f_text, char *f_bin)
{
    FILE *fin, *fout;
    int x;

    //deschidem fisierul de intrare (text)
    fin = fopen(f_text, "r");

    //deschidem fisierul de iesire (binar)
    fout = fopen(f_bin, "wb");

    //citesc cate un numar din fisierul de intrare
    while(fscanf(fin, "%d", &x) == 1)
        //si il scriu in fisierul de iesire
        fwrite(&x, sizeof(x), 1, fout);

    //inchidem ambele fisiere
    fclose(fin);
    fclose(fout);
}
```

b) Scrieți o funcție `bin2txt` care preia numerele întregi dintr-un fișier binar și le scrie într-un fișier text (funcția are doi parametri, respectiv calea fișierului binar și calea fișierului text).

Rezolvare:

```
void bin2txt(const char* f_bin, const char* f_txt)
{
    FILE *fin, *fout;
```

```
    fin  = fopen(f_bin, "rb");
    fout = fopen(f_txt, "w");

    int x;
    while(fread(&x, sizeof(int), 1, fin) == 1)
        fprintf(fout, "%d ", x);

    fclose(fin);
    fclose(fout);
}
```

c) Scrieți o funcție schimba_semn care să schimbe semnul fiecărui număr întreg dintr-un fișier binar (funcția va avea ca parametru calea fișierului binar).

Rezolvare:

```
void schimba_semn(const char* f_bin)
{
    FILE* file;
    file = fopen(f_bin, "rb+");

    int x;
    while(fread(&x, sizeof(x), 1, file) == 1)
    {
        x *= -1;
        fseek(file, -(int)sizeof(x), SEEK_CUR);
        fwrite(&x, sizeof(x), 1, file);
        fflush(file);
    }

    fclose(file);
}
```

d) Scrieți un program de test pentru cele 3 funcții definite anterior.

Rezolvare:

```
int main()
{
    txt2bin("numere.txt", "numere.bin");
    schimba_semn("numere.bin");
    bin2txt("numere.bin", "numere1.txt");
    return 0;
}
```

2. Considerând o imagine color în format *bitmap* (BMP) pe 24 de biţi, scrieţi un program care să realizeze următoarele cerinţe:

a) să afişeze dimensiunea în octeţi, dimensiunea în pixeli şi padding-ul imaginii;

Rezolvare:

```
#include <stdio.h>

int main()
{
    FILE *fin;
    unsigned int dim_bytes, width, height, padding, bits_per_pixel;

    fin = fopen("goldhill.bmp", "rb");

    //dimensiunea imaginii in octeti se gaseste
    //incepand cu octetul 2 pe 4 octeti (unsigned int)
    fseek(fin, 2, SEEK_SET);
    fread(&dim_bytes, 4, 1, fin);
    printf("Dimensiunea imaginii in octeti: %u\n", dim_bytes);

    //latimea imaginii in pixeli se gaseste
    //incepand cu octetul 18 pe 4 octeti (unsigned int)
    fseek(fin, 18, SEEK_SET);
    fread(&width, 4, 1, fin);
    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, fin);
    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(fin, 28, SEEK_SET);
    fread(&bits_per_pixel, 4, 1, fin);
    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);

    fclose(fin);

    return 0;
}
```

b) să creeze imaginea color complementară imaginii date;

Rezolvare:

```
#include <stdio.h>

int main()
{
    FILE *fin, *fout;

    //header = header-ul imaginii initiale
    unsigned char header[54];
    //cc = canal de culoare (R, G sau B) = 1 octet fara semn
    unsigned char cc;

    fin  = fopen("goldhill.bmp", "rb");
    fout = fopen("complementara_goldhill.bmp", "wb");

    //citim header-ul imaginii initiale
    //si il scriem in imaginea complementara
    fread(header, 1, 54, fin);
    fwrite(header, 1, 54, fout);

    //citim octet cu octet din imaginea initiala
    //si scriem complementarul sau fata de 255
    //in imaginea complementara
    while(fread(&cc, 1, 1, fin) == 1)
    {
        cc = 255 - cc;
        fwrite(&cc, 1, 1, fout);
    }

    fclose(fin);
```

```
    fclose(fout);

    return 0;
}
```

c) să creeze imaginea în tonuri de gri (*grayscale*) corespunzătoare imaginii date.

Formatul BMP (BMP file format - Wikipedia)

Formatul BMP (bitmap) pe 24 de biţi este un format de fişier binar folosit pentru a stoca imagini color digitale bidimensionale având lăţime, înălţime şi rezoluţie arbitrare. Practic, imaginea este considerată ca fiind un tablou bidimensional de pixeli, iar fiecare pixel este codificat prin 3 octeţi corespunzători intensităţilor celor 3 canale de culoare R (Red), G(green) şi B(blue). Intensitatea fiecărui canal de culoare R, G sau B este dată de un număr natural cuprins între 0 şi 255. De exemplu, un pixel cu valorile (0, 0 , 0) reprezintă un pixel de culoare neagră, iar un pixel cu valorile (255, 255, 255) reprezintă un pixel de culoare albă.

Formatul BMP cuprinde o zonă cu dimensiune fixă, numita *header*, şi o zonă de date cu dimensiune variabilă care conţine pixelii imaginii propriu-zise. Header-ul, care ocupă primii 54 de octeţi ai fişierului, conţine informaţii despre formatul BMP, precum şi informaţii despre dimensiunea imaginii, numărul de octeţi utilizaţi pentru reprezentarea unui pixel etc.

Dimensiunea imaginii în octeţi este specificată în header printr-o valoare întreagă fără semn, deci memorată pe 4 octeţi, începând cu octetul cu numărul de ordine 2.

Dimensiunea imaginii în pixeli este exprimată sub forma $W \times H$, unde W reprezintă numărul de pixeli pe lăţime, iar H reprezintă numărul de pixeli pe înălţime. Lăţimea imaginii exprimată în pixeli este memorată pe patru octeţi fără semn începând cu octetul al 18-lea din header, iar înălţimea este memorată pe următorii 4 octeţi fără semn, respectiv începând cu octetul al 22- lea din header.

După cei 54 de octeţi ai header-ului, într-un fişier BMP urmează zona de date, în care sunt memorate ÎN ORDINE INVERSĂ liniile de pixeli ai imaginii, deci ultima linie de pixeli din imagine va fi memorată prima, penultima linie va fi memorată a doua,..., prima linie din imagine va fi memorată ultima. Deoarece codarea unei imagini BMP pe 24 de biţi într-un fişier binar respectă standardul little-endian, octeţii corespunzători celor 3 canale de culoare R, G şi B sunt memoraţi de la dreapta la stânga, adică în ordinea B, G şi R!

Pentru rapiditatea procesării imaginilor la citire şi scriere, imaginile în format BMP au proprietatea că fiecare linie este memorată folosind un număr de octeţi multiplu de 4. Dacă este necesar, acest lucru de realizează prin adăugarea unor octeţi de completare (*padding*) la sfârşitul fiecărei linii, astfel încât numărul total de octeţi de pe fiecare linie să devină multiplu de 4. Numărul de octeţi corespunzători unui linii este $3 \times W$ (câte 3 octeţi pentru fiecare

pixel de pe o linie). Astfel, dacă o imagine are $W = 11$ pixeli în lățime, atunci numărul de octeți de padding este 3 ($3 \times 11 = 33$ octeți pe o linie, deci se vor adăuga la sfârșitul fiecărei linii câte 3 octeți de completare, astfel încât să avem $33 + 3 = 36$ multiplu de 4 octeți). De obicei, octeții de completare au valoarea 0.