1. Programul incepe prin definirea bibliotecilor necesare: stdio.h, string.h, math.h, ctype.h

PSEUDOCOD

```
START

fisier pointer FILE

DACA fisier = NULL ATUNCI
        afisare "fisierul e null"
SFARSIT DACA

nr_linie_afisata, linie_curenta intreg
linie_curenta <- 1

nume, prenume, data_nastere, aux, aux1, aux2 siruri de caractere
CAT TIMP !feof(fisier) EXECUTA
        DACA linie_curenta=nr_linie_afisata ATUNCI
                citire din fisier "%s %s %s" nume, prenume, data_nastere
        ALTFEL
                // bagam in aux, aux1, aux2 sirurile de caractere care nu ne intereseaza
                citire din fisier "%s %s %s" aux, aux1, aux2
        SFARSIT DACA

aux <- ""
aux1 <- ""
aux2 <- ""

PENTRU i=0;strlen(data_nastere);i++ EXECUTA
        DACA i < 2 ATUNCI
                concateneaza la aux data_nastere[i]
        SFARSIT DACA
        DACA i > 2 && i < 5 ATUNCI
```

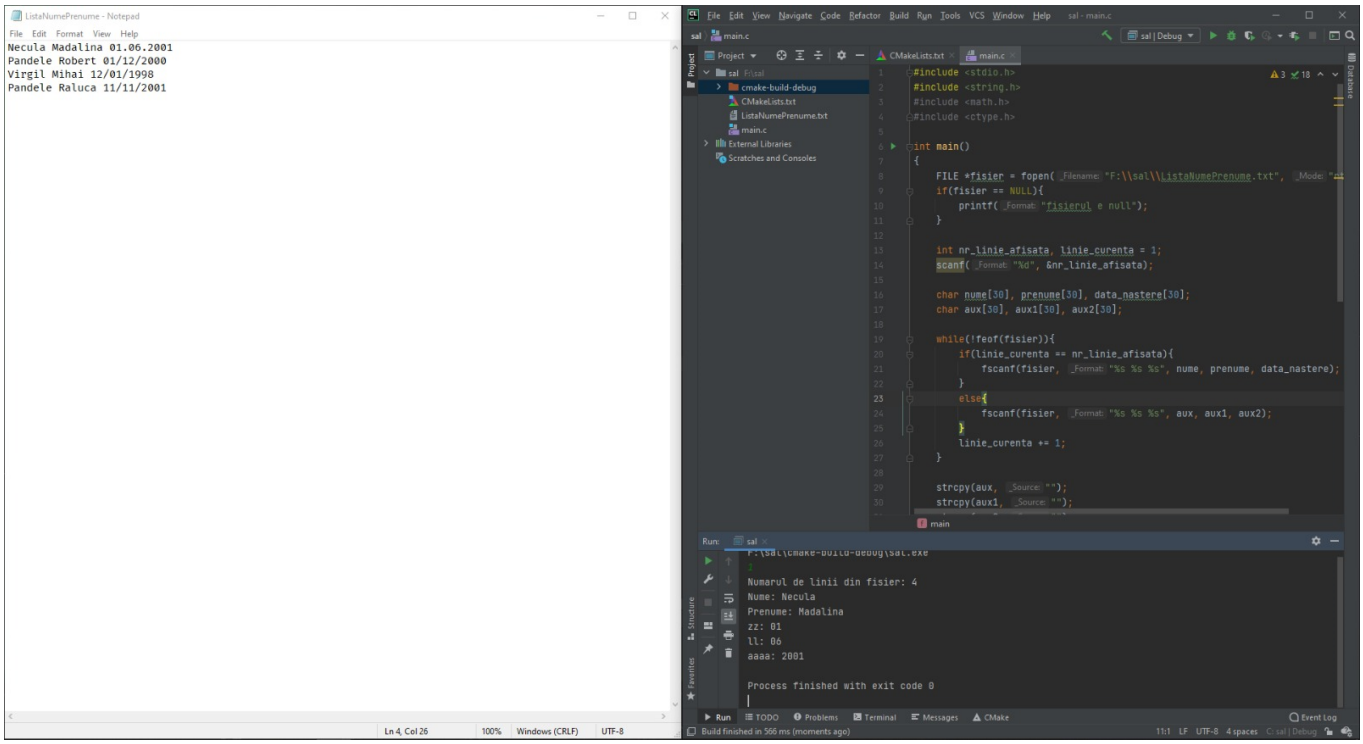
```
                concateneaza la aux1 data_nastere[i]
        SFARSIT DACA
        DACA i > 5 ATUNCI
                concateneaza la aux2 data_nastere[i]
        SFARSIT DACA

afisare linie_curenta - 1 // nr total linii
afisare nume
afisare prenume
afisare zi
afisare luna
afisare an
SFARSIT
```

COD

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

int main()
{
   FILE *fisier = fopen("F:\\sal\\ListaNumePrenume.txt", "rt");
   if(fisier == NULL){
      printf("fisierul e null");
   }

   int nr_linie_afisata, linie_curenta = 1;
   scanf("%d", &nr_linie_afisata);

   char nume[30], prenume[30], data_nastere[30];
```

```
char aux[30], aux1[30], aux2[30];

while(!feof(fisier)){
   if(linie_curenta == nr_linie_afisata){
      fscanf(fisier, "%s %s %s", nume, prenume, data_nastere);
   }
   else{
      fscanf(fisier, "%s %s %s", aux, aux1, aux2);
   }
   linie_curenta += 1;
}

strcpy(aux, "");
strcpy(aux1, "");
strcpy(aux2, "");

for(int i = 0; i < strlen(data_nastere); i++){
   if(i < 2){
      strncat(aux, &data_nastere[i], 1);
   }
   if(i > 2 && i < 5){
      strncat(aux1, &data_nastere[i], 1);
   }
   if(i > 5){
      strncat(aux2, &data_nastere[i], 1);
   }
}

printf("Numarul de linii din fisier: %d\n", linie_curenta - 1);
printf("Nume: %s\n", nume);
printf("Prenume: %s\n", prenume);
```

```
    printf("zz: %s\n", aux);
    printf("ll: %s\n", aux1);
    printf("aaaa: %s\n", aux2);

    fclose(fisier);
}
```



2.

PSEUDOCOD

START

region structura {x0, y0, x1, y1 intreg};

int Negate_Image(unsigned char* Img, int row, int col, struct region ROI)

fisier pointer FILE

```
        // afisare matrice inainte de prelucrare
        PENTRU y=ROI.y1; y<=ROI.y0; y++ EXECUTA
                PENTRU x=ROI.x0; x<=ROI.x1; x++ EXECUTA
                        afisare element matrice
                afisare newline

        PENTRU y=ROI.y1; y<=ROI.y0; y++ EXECUTA
                PENTRU x=ROI.x0; x<=ROI.x1; x++ EXECUTA
                        Img[y * col + x] = 255 - Img[y * col + x]

        // afisare matrice dupa prelucrare
        PENTRU y=ROI.y1; y<=ROI.y0; y++ EXECUTA
                PENTRU x=ROI.x0; x<=ROI.x1; x++ EXECUTA
                        afisare element matrice
                afisare newline

int BlackWhite_Image(unsigned char* Image, int row, int col, struct region ROI)
        fisier pointer FILE

        // afisare matrice inainte de prelucrare
        PENTRU y=ROI.y1; y<=ROI.y0; y++ EXECUTA
                PENTRU x=ROI.x0; x<=ROI.x1; x++ EXECUTA
                        afisare element matrice
                afisare newline

        PENTRU y=ROI.y1; y<=ROI.y0; y++ EXECUTA
                PENTRU x=ROI.x0; x<=ROI.x1; x++ EXECUTA
                        DACA Image[y * col + x] > 127 ATUNCI
                                Image[y * col + x] <- 255
                        ALTFEL
```

```
                    Image[y * col + x] <- 0
                SFARSIT DACA

    // afisare matrice dupa prelucrare
    PENTRU y=ROI.y1; y<=ROI.y0; y++ EXECUTA
        PENTRU x=ROI.x0; x<=ROI.x1; x++ EXECUTA
            afisare element matrice
        afisare newline

int Gray_Image_Processing(unsigned char* Img, int n, int m, struct region ROI, int (*
pfunct)(unsigned char*, int, int, struct region))
    pfunct(Img, n, m, ROI)

main
    matrice unsigned char
    initializare seed pentru rand()

    i,j intregi
    i <- 0
    j <- 0
    PENTRU i=0; i<256; i++ EXECUTA
        PENTRU j=0; j<256; j++ EXECUTA
            matrice[i][j] <- rand() % 256

    ROI struct region
    ROI.x0 <- 11
    ROI.x1 <- 13
    ROI.y0 <- 11
    ROI.y1 <- 11

    pfunct pointer functie
```

```
        pfunct <- BlackWhite_Image
        Gray_Image_Processing(matrice, 256, 256, ROI, pfunct)

STOP
```

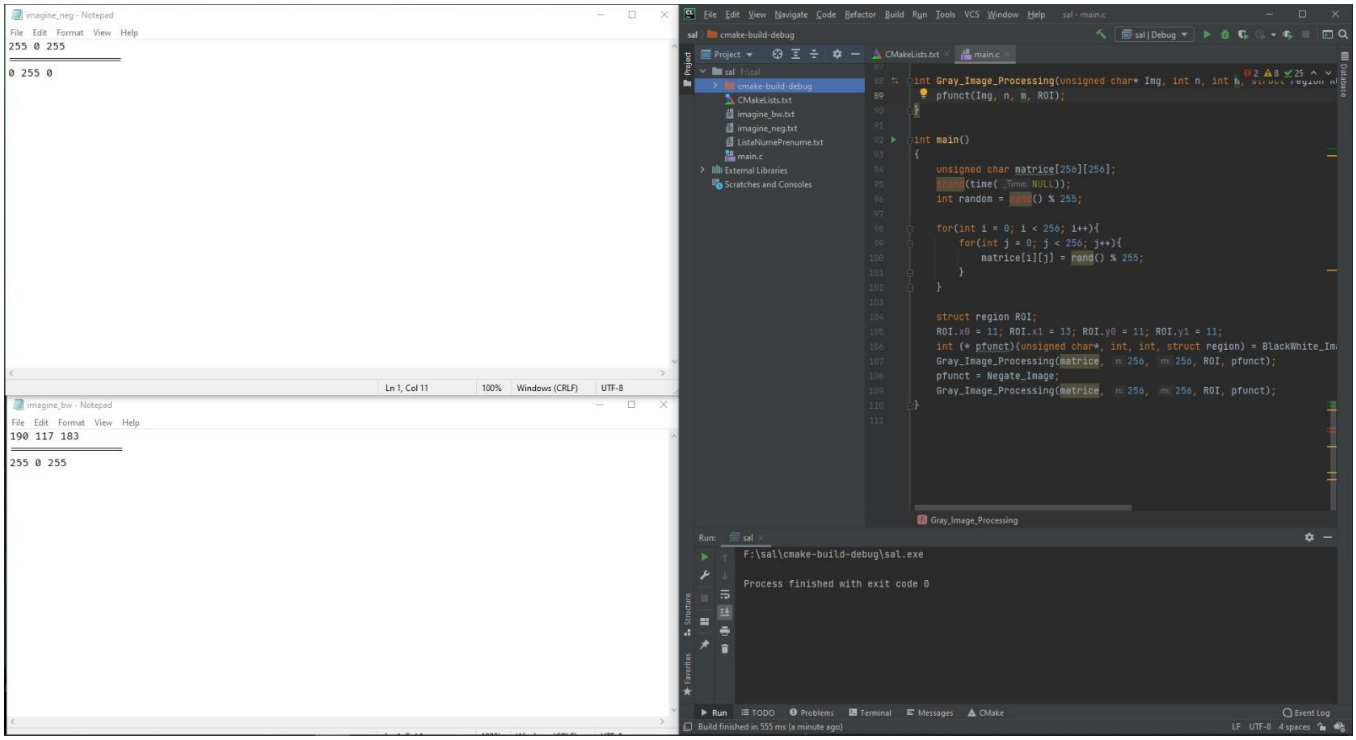
COD

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <ctype.h>
#include <time.h>

//Necula Madalina 01.06.2001
//x0 = 11; x1 = 13; y0 = 11; y1 = 11

struct region{
   int x0;
   int y0;
   int x1;
   int y1;
};

int Negate_Image(unsigned char* Img, int row, int col, struct region ROI);
int BlackWhite_Image(unsigned char* Image, int row, int col, struct region ROI);
int Gray_Image_Processing(unsigned char* Img, int n, int m, struct region ROI, int (*
pfunct)(unsigned char*, int, int, struct region));

int Negate_Image(unsigned char* Img, int row, int col, struct region ROI){
   // orice element x devine 255 - x
   FILE *fisier = fopen("F:\\sal\\imagine_neg.txt", "wt");
```

```
    // Afisare imagine inainte de prelucrare
    for(int y = ROI.y1; y <= ROI.y0; y++){
        for(int x = ROI.x0; x <= ROI.x1; x++){
            fprintf(fisier, "%d ", Img[y * col + x]);
        }
        fprintf(fisier, "\n");
    }
    fprintf(fisier, "==================\n");

    for(int y = ROI.y1; y <= ROI.y0; y++){
        for(int x = ROI.x0; x <= ROI.x1; x++){
            Img[y * col + x] = 255 - Img[y * col + x];
        }
    }

    // Afisare imagine dupa prelucrare
    for(int y = ROI.y1; y <= ROI.y0; y++){
        for(int x = ROI.x0; x <= ROI.x1; x++){
            fprintf(fisier, "%d ", Img[y * col + x]);
        }
        fprintf(fisier, "\n");
    }

    fclose(fisier);
    return 1;
}

int BlackWhite_Image(unsigned char* Image, int row, int col, struct region ROI){
    // orice element x, daca x > 127 devine 255, altfel 0
    FILE *fisier = fopen("F:\\sal\\imagine_bw.txt", "wt");
```

```
    // Afisare imagine inainte de prelucrare
    for(int y = ROI.y1; y <= ROI.y0; y++){
        for(int x = ROI.x0; x <= ROI.x1; x++){
            fprintf(fisier, "%d ", Image[y * col + x]);
        }
        fprintf(fisier, "\n");
    }
    fprintf(fisier, "==================\n");

    for(int y = ROI.y1; y <= ROI.y0; y++){
        for(int x = ROI.x0; x <= ROI.x1; x++){
            if(Image[y * col + x] > 127){
                Image[y * col + x] = 255;
            }
            else{
                Image[y * col + x] = 0;
            }
        }
    }

    // Afisare imagine dupa prelucrare
    for(int y = ROI.y1; y <= ROI.y0; y++){
        for(int x = ROI.x0; x <= ROI.x1; x++){
            fprintf(fisier, "%d ", Image[y * col + x]);
        }
        fprintf(fisier, "\n");
    }

    fclose(fisier);
    return 1;
}
```

```
int Gray_Image_Processing(unsigned char* Img, int n, int m, struct region ROI, int (*
pfunct)(unsigned char*, int, int, struct region)){
   pfunct(Img, n, m, ROI);
}

int main()
{
   unsigned char matrice[256][256];
   srand(time(NULL));
   int random = rand() % 255;

   for(int i = 0; i < 256; i++){
      for(int j = 0; j < 256; j++){
         matrice[i][j] = rand() % 255;
      }
   }

   struct region ROI;
   ROI.x0 = 11; ROI.x1 = 13; ROI.y0 = 11; ROI.y1 = 11;
   int (* pfunct)(unsigned char*, int, int, struct region) = BlackWhite_Image;
   Gray_Image_Processing(matrice, 256, 256, ROI, pfunct);
}
```



3.**COD**

```
 #include <stdio.h>
#include <string.h>
#include <math.h>
#include <ctype.h>
#include <time.h>

struct item{
   char N[20];
   int ID;
};

int main()
{
   void **vector = malloc(5 * sizeof(void *));
```

```
struct item obj;
int numar = 5;
float numar_real = 5.5;

obj.ID = 4567;
strcpy(obj.N, "AABBCC");

for(int i = 0; i < 5; i++){
   if(i == 0){
      vector[i] = malloc(sizeof(int));
      vector[i] = &numar;
   }
   if(i == 1){
      vector[i] = malloc(sizeof(float));
      vector[i] = &(numar_real);
   }
   if(i == 2){
      vector[i] = malloc(sizeof(struct item));
      vector[i] = &(obj);
   }
}

for(int i = 0; i < 3; i++){
   if(i == 0){
      printf("nr intreg: %d\n", *(int *)(vector[i]));
   }
   if(i == 1){
      printf("nr real: %f\n", *(float *)(vector[i]));
   }
   if(i == 2){
      printf("obj.N %s obj.ID %d", (*(struct item *)(vector[i])).N, (*(struct item *)(vector[i])).ID);
```

```
    }
  }
}
```