Sprawozdanie

Adam Stypczyc, 259273

27 listopada 2020

Zadanie 3 przetwarzanie obrazów 1

Spis treści

Listings

1 Wstęp

Zadaniem, które musiałem wykonać, było napisanie programu przetwarzającego obrazy w formacie PGM. Po uruchomieniu programu otwiera się proste i przejrzyste menu. Do programu wczytujemy obraz, jeżeli chcemy go przetworzyć. Gdy nie wczytamy obrazu, nie możemy go przetworzyć. Po wczytaniu obrazu możemy go zapisać, wyświetlić oraz przetworzyć na różne sposoby. Do tych sposobów należą: negatyw, progowanie, korekcja gamma, konturowanie oraz rozciąganie histogramu. Samo menu zostało napisane na bazie funkcji switch. Początkowo napisałem funkcję zapis, a następnie przeszedłem do pisania funkcji przetwarzania obrazów bazując na plikach PDF, które dostarczył nam doktor Muszyński. Na sam koniec dodałem zmienne, których użyłem dodatkowo do funkcji main, oraz utworzyłem menu. Było to zadanie podobne do zadania na przygotowanie do zajęć laboratoryjnych nr 3. Po pierwszym poprawnym skompilowaniu programu przetworzony obraz nie chciał się otworzyć. Było to spowodowane tym, że w funkcji zapisu w jednym miejscu nie dałem spacji. Oddzielała ona wartości, dlatego jej brak uniemożliwiał otworzenie pliku. Był to największy problem jaki napotkałem podczas pisania tego programu, pomijając to, że początkowo nie wiedziałem zbytnio jak podejść do tego zadania. W celu testu tego programu utworzyłem bazujący na nim kod testowy. Pominąłem w nim menu i z góry przypisałem, który obraz ma być przetwarzany. Test menu został wykonany ręcznie.

2 Ważne fragmenty kodu programu

2.1 Funkcja odczytu

```
int czytaj(FILE *plik_we,int obraz_pgm[][MAX],int *wymx,int *
    wymy, int *szarosci) {
  char buf[DL_LINII];
  int znak;
  int koniec=0;

  if (plik_we==NULL) {
    fprintf(stderr,"Blad: Nie podano uchwytu do pliku\n");
    return(0);
  }

  if (fgets(buf,DL_LINII,plik_we)==NULL)
    koniec=1;

  if ( (buf[0]!='P') || (buf[1]!='2') || koniec) {
    fprintf(stderr,"Blad: To nie jest plik PGM\n");
    return(0);
  }
  do {
    if ((znak=fgetc(plik_we))=='#') {
      if (fgets(buf,DL_LINII,plik_we)==NULL)
  koniec=1;
    }  else {
      ungetc(znak,plik_we);
    }
  } while (znak=='#' && !koniec);

  if (fscanf(plik_we,"%d %d %d",wymx,wymy,szarosci)!=3) {
    fprintf(stderr,"Blad: Brak wymiarow obrazu lub liczby
     stopni szarosci\n");
    return(0);
  }
  if(*wymx>MAX)
  {
    printf("Blad: Niewlasciwe wymiary obrazu\n");
  }
  else if(*wymy>MAX)
  {
    printf("Blad: Niewlasciwe wymiary obrazu\n");
  }
```

```
39    else if(*wymy>MAX && *wymx>MAX)
40    {
41      printf("Blad: Niew Ća Żciwe wymiary obrazu\n");
42    }
43
44    for (i=0;i<*wymy;i++) {
45      for (j=0;j<*wymx;j++) {
46        if (fscanf(plik_we,"%d",&(obraz_pgm[i][j]))!=1) {
47    fprintf(stderr,"Blad: Niewlasciwe wymiary obrazu\n");
48    return(0);
49        }
50      }
51    }
52    return *wymx**wymy;
53  }
```

Listing 1: Odczyt

2.2 Funkcja zapisu

```
int zapisz(FILE *plik_wy,int obraz_pgm[][MAX],int wymx,int wymy
    , int szarosci)
{
  fprintf(plik_wy,"P2\n");
  fprintf(plik_wy,"%d %d\n%d \n", wymx, wymy, szarosci);
  for(i=0;i<wymy;i++)
  {
    for(j=0;j<wymx;j++)
    {
      fprintf(plik_wy,"%d ", obraz_pgm[i][j]);
    }
    fprintf(plik_wy,"\n");
  }
}
```

Listing 2: Zapis

2.3 Funkcja wyświetlania

```
void wyswietl(char *n_pliku)
{
  char polecenie[DL_LINII];
  strcpy(polecenie,"display ");
  strcat(polecenie,n_pliku);
  strcat(polecenie," &");
  printf("%s\n",polecenie);
  system(polecenie);
}
```

Listing 3: Wyświetlanie

2.4 Funkcja negatywu

```
void negatyw(int obraz_pgm[][MAX],int wymx,int wymy, int
    szarosci)
{
  for(i=0;i<wymy;i++)
  {
    for(j=0;j<wymx;j++)
    {
      obraz_pgm[i][j]=szarosci-obraz_pgm[i][j];
    }
  }
}
```

Listing 4: Negatyw

2.5 Funkcja progowania

```
void progowanie(float proc_p, int obraz_pgm[][MAX],int wymx,int
     wymy, int szarosci)
{
  int prog;
  prog=szarosci*proc_p/100;
  for(i=0;i<wymy;i++)
  {
    for(j=0;j<wymx;j++)
    {
      if(obraz_pgm[i][j]<=prog)
      {
        obraz_pgm[i][j]=0;
      }
      else
      {
        obraz_pgm[i][j]=szarosci;
      }
    }
  }
}
```

Listing 5: Progowanie

2.6 Funkcja korekcji gamma

```
void korekcja_gamma(float gamma1, int obraz_pgm[][MAX],int wymx
    ,int wymy, int szarosci)
{
                      /
  for(i=0;i<wymy;i++)
  {
    for(j=0;j<wymx;j++)
    {
      obraz_pgm[i][j]=pow((float)obraz_pgm[i][j]/szarosci,1/
    gamma1)*szarosci;
    }
  }
}
```

Listing 6: Korekcja gamma

2.7 Funkcja konturowania

```
void konturowanie(int obraz_pgm[][MAX],int wymx,int wymy, int
    szarosci)
{
  for(i=0;i<wymy;i++)
    {
      for(j=0;j<wymx;j
      {
        if(i==wymy-1 && j==wymx-1)
        {
          obraz_pgm[i][j]=abs(obraz_pgm[0][j]-obraz_pgm[i][j])+
    abs(obraz_pgm[i][0]-obraz_pgm[i][j]);
        }
        else if(i==wymy-1)
        {
          obraz_pgm[i][j]=abs(obraz_pgm[0][j]-obraz_pgm[i][j])+
    abs(obraz_pgm[i][j+1]-obraz_pgm[i][j]);
        }
        else if(j==wymx-1)
        {
          obraz_pgm[i][j]=abs(obraz_pgm[i+1][j]-obraz_pgm[i][j
    ])+abs(obraz_pgm[i][0]-obraz_pgm[i][j]);
        }
        else
        {
        obraz_pgm[i][j]=abs(obraz_pgm[i+1][j]-obraz_pgm[i][j])+
    abs(obraz_pgm[i][j+1]-obraz_pgm[i][j]);
        }
      }
    }
}
```

Listing 7: Konturowanie

2.8 Funkcja rozciągania histogramu

```
void histogram(int obraz_pgm[][MAX],int wymx,int wymy, int
    szarosci)
{
  int wartosc_MIN, wartosc_MAX;
  wartosc_MAX=0;
  wartosc_MIN=szarosci;
  for(i=0;i<wymy;i++)
  {
    for(j=0;j<wymx;j++)
    {
      if(obraz_pgm[i][j]>wartosc_MAX)
      {
        wartosc_MAX=obraz_pgm[i][j];
      }
      else
      {
      }
    }
  }
  for(i=0;i<wymy;i++)
  {
    for(j=0;j<wymx;j++)
    {
      if(obraz_pgm[i][j]<wartosc_MIN)
      {
        wartosc_MIN=obraz_pgm[i][j];
      }
      else
      {
      }
    }
  }
  for(i=0;i<wymy;i++)
  {
    for(j=0;j<wymx;j++)
    {
      obraz_pgm[i][j]=(obraz_pgm[i][j]-wartosc_MIN)*MAX/(
    wartosc_MAX-wartosc_MIN);
    }
  }
}
```

Listing 8: Histogram

3 Testy programu

3.1 Co dane testy ma na celu

Testy mają na celu weryfikację poprawnego działania programu.

3.1.1 Test 1

Test 1. sprawdza działanie funkcji odczytu, zapisu i wyświetlania. Plik początkowy Lena.pgm zostaje wczytany do programu, zapisany i wyświetlony. Test został wykonany na komputerze lokalnym oraz na komputerze diablo. W oryginalnym kodzie programu, gdy plik nie zostanie wczytany otrzymujemy komunikat informujący nas o tym. W kodzie testowym nie mamy takiego komunikatu.

3.1.2 Test 2

Test 2. kontroluje poprawność funkcji negatywu, sprawdza również te same funkcje co test 1., ponieważ plik zostaje wczytany, przetworzony na negatyw, zapisany oraz wyświetlony.

3.1.3 Test 3

Test 3. bada jak program zachowuje się dla podczas działania funkcji progowania. Funkcja progowania została wykonana dla progów równych 20% oraz 40%. Test został wykonany na komputerze lokalnym oraz na komputerze diablo. Poniżej zostało zamieszczone porównanie otrzymanych obrazów. Aby test był możliwy obrazy zostały wczytane, zapisane oraz wyświetlone. W oryginalnym pliku jeżeli podamy wartość procenta spoza przedziału od 0 do 100, otrzymamy komunikat o błędnej wartości procenta.

3.1.4 Test 4

Test 4. przygląda się działaniu programu, gdy chcemy wykonać korekcję gamma. Funkcja korekcji gamma została przetestowana w podobny sposób jak funkcja progowania. Została przetestowana dla różniących się od siebie wartości gamma. Porównanie również zostało umieszczone poniżej. Gdy w oryginalnym pliku wpiszemy gammę mniejszą od 0, zostaniemy poinformowani o błędzie.

3.1.5 Test 5

Test 5. ustala, czy funkcja konturowania działa bez zastrzeżeń. Ta funkcja została przetestowana w podobny sposób do funkcji negatywu. Obraz oryginalny został wczytany, przetworzony, zapisany, wyświetlony oraz obrazy otrzymane w wyniku działania programu zostały porównane ze względu na to, na którym komputerze zostały wykonane.

3.1.6 Test 6

Test 6. poddaje próbie funkcję rozciągania histogramu. Test niemalże nie różniący się niczym od testu 1. i 5.. Jedyną różnicą jest to, że testowana jest funkcja rozciągania histogramu.

3.1.7 Test 7

Test 7. sprawdza co się stanie gdy wczytamy plik, który nie jest plikiem pgm. Od programu oczekujemy komunikatu o ty, że poinformuje nas o tym, że plik nie jest plikiem pgm. Chcemy również zobaczyć czy program wyświetli nam ten obraz.

3.1.8 Test 8

Test 8. sprawdza co się stanie, gdy wczytamy plik o złych wymiarach. W tym teście testujemy, czy plik wczyta nam obraz oraz czy go wyświetli. Również oczekujemy odpowiedniego komunikatu, o tym, że plik ma złe wymiary.

3.1.9 Test 9

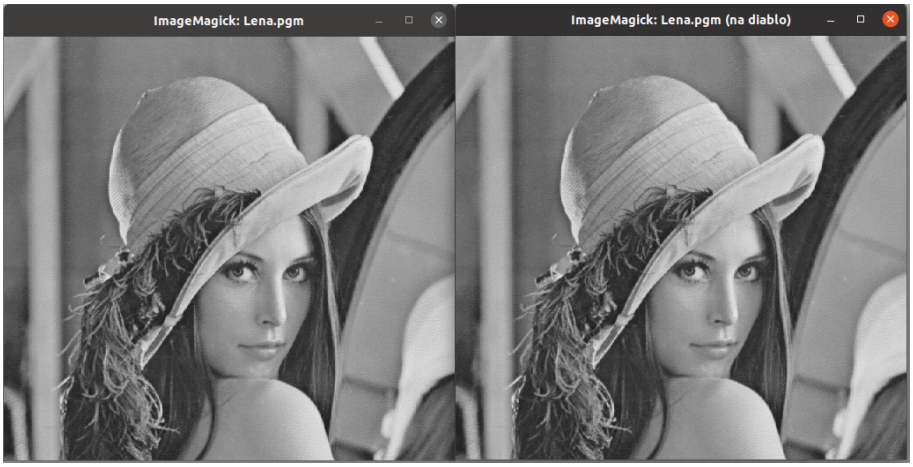
Test 9. bada poprawność działania menu użytkownika. Oryginalny program został włączony co spowodowało wyświetlenie się menu, które było do dyspozycji użytkownika.

3.2 Metoda testowania

Program został przetestowany poprzez stworzenie nowego kodu programu na komputer lokalny i na komputer diablo. W nowym programie funkcje wywołują się automatycznie, co oznacza, że po „odpaleniu" programu wyświetlają nam się gotowe, przetworzone obrazy. Jedynie menu zostało ręcznie przetestowane. Kody na komputer lokalny oraz na komputer diablo różnią się tylko nazwami plików, które zostaną zapisane. W tych nazwach zostało podkreślone na których komputerach testy zostały wykonane.

3.3 Dane wyjściowe zwrócone w teście przez program

3.3.1 Test 1

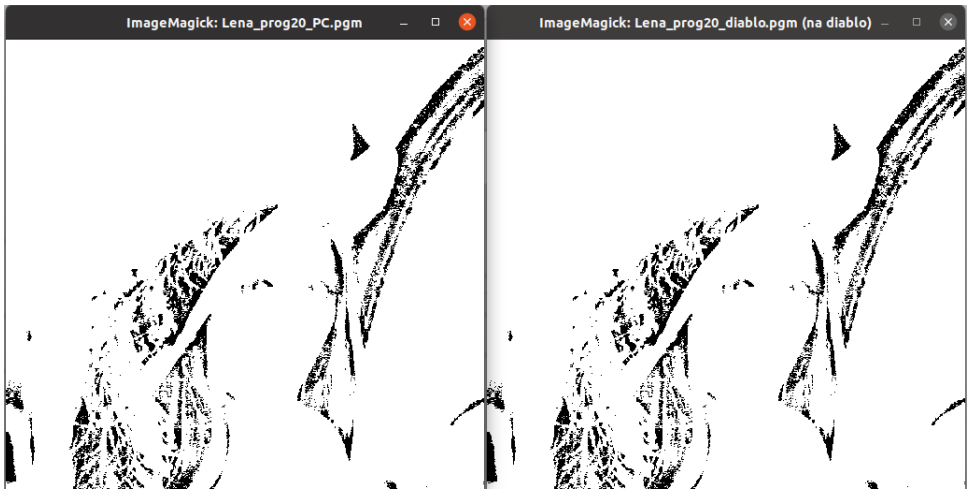


Rysunek 1: Porównanie oryginału na komputerze lokalnym oraz na komputerze diablo.

3.3.2 Test 2

Rysunek 2: Porównanie negatywu na komputerze diablo oraz na komputerze lokalnym.

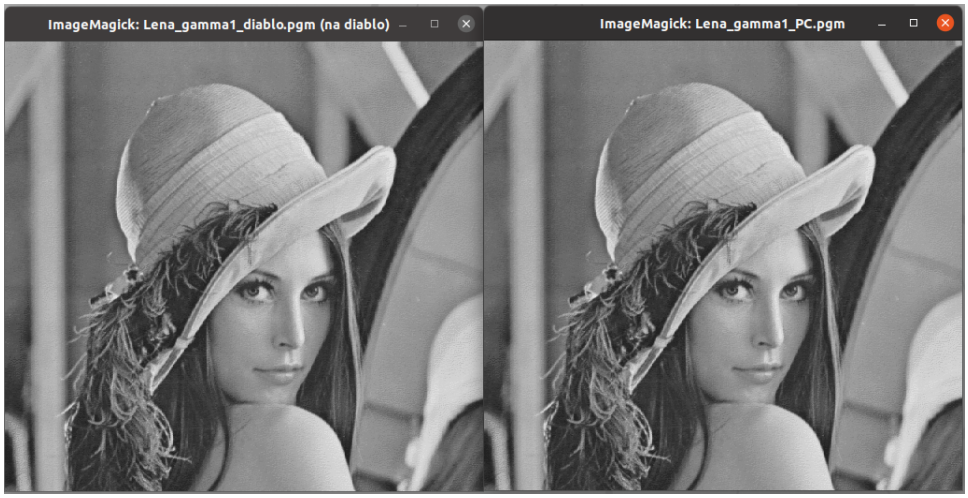
3.3.3 Test 3



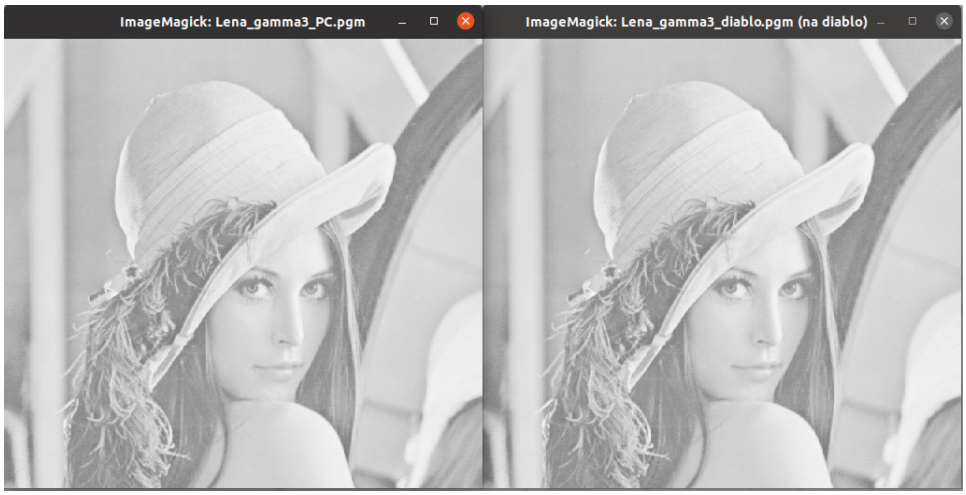
Rysunek 3: Porównanie progowania o progu 20% na komputerze diablo oraz na komputerze lokalnym.

Rysunek 4: Porównanie progowania o progu 40% na komputerze diablo oraz na komputerze lokalnym.

3.3.4 Test 4



Rysunek 5: Porównanie korekcji gamma o gamma równym 1 na komputerze diablo oraz na komputerze lokalnym.



Rysunek 6: Porównanie korekcji gamma o gamma równym 3 na komputerze diablo oraz na komputerze lokalnym.

3.3.5 Test 5

Rysunek 7: Porównanie konturowania na komputerze diablo oraz na komputerze lokalnym.

3.3.6 Test 6

Rysunek 8: Porównanie rozciągania histogramu na komputerze diablo oraz na komputerze lokalnym.

3.3.7 Test 7

```
Blad: To nie jest plik PGM
display piesel.jpg &
```

Rysunek 9: Błąd na komputerze diablo.

```
Blad: To nie jest plik PGM
display piesel.jpg &
```

Rysunek 10: Błąd na komputerze lokalnym.

Rysunek 11: Porównanie wczytywania i wyświetlania pliku piesel.jpg na komputerze diablo oraz na komputerze lokalnym.

3.3.8 Test 8

```
Blad: Niewłaściwe wymiary obrazu
Segmentation Fault
```

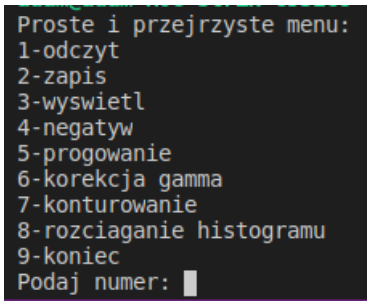
Rysunek 12: Błąd na komputerze diablo.

```
Blad: Niewłaściwe wymiary obrazu
Naruszenie ochrony pamięci (zrzut pamięci)
```

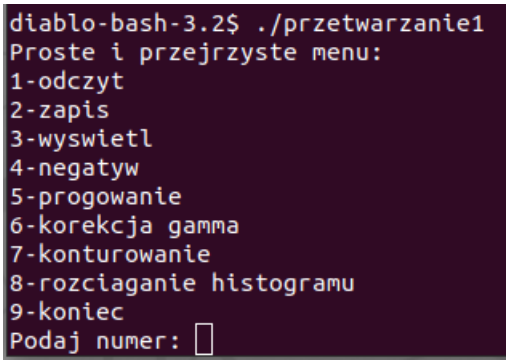
Rysunek 13: Błąd na komputerze lokalnym.

Plik nie został wyświetlony, oraz zakończył działanie programu. Początkowo test obrazu o złych wymiarach był przed testem obrazu piesel.jpg, jednakże przez błąd, który wystąpił w teście, spowodował to, że test piesel.jpg wcale się nie wykonał. Z tego powodu testy zostały zamienione miejscami, aby test w całości był zautomatyzowany.

3.3.9 Test 9



Rysunek 14: Test menu na komputerze lokalnym.



Rysunek 15: Test menu na komputerze diablo.

Menu działa poprawnie. Wyświetla się oraz daje możliwość wyboru z zakresu 1-9, w zależności od tego co chcemy zrobić. Gdy wybierzemy wartość spoza wskazanego zakresu, nic się nie stanie oraz menu wyświetli się nam jeszcze raz. Na samym początku powinniśmy wczytać obraz wybierając 1 i wpisując nazwę pliku. Dopiero wtedy możemy go przetwarzać.

4 Wnioski końcowe

Przetestowanie programu pozwoliło na przetworzenie obrazu Lena.pgm. Testy wykazały, że program bez problemu przetwarza obraz, który jest w odpowiednim formacie oraz o odpowiednim rozmiarze. Test menu pokazał, że działa ono bez zarzutów. Wyniki testów na komputerze lokalnym oraz na komputerze diablo wyszły identycznie. Program nie został przetestowany, pod tym względem czy przetworzy plik o innym formacie niż pgm. Było to zbędne, ponieważ plik o złym formacie został błędnie odczytany, więc nie byłoby możliwe poprawne przetworzenie obrazu. Również gdy próbowaliśmy odczytać i wyświetlić obraz o złych wymiarach nie było to możliwe. Otrzymywaliśmy komunikat o tym, że plik ma złe wymiary, a następnie „Segmentation fault". Kończyło to działanie programu. Było to niezależne od tego na którym miejscu znajdował się obrazu o złych wymiarach. Następne testy się nie wykonały. Aby wczytać obraz o wymiarach spoza zakresu początkowego, musielibyśmy zmienić definicję wymiaru obrazu, tak aby wymiar testowanego obrazu znajdował się w zakresie. Po tych testach możemy powiedzieć, że program działa poprawnie.