Sprawozdanie PO2

Autor: Mikołaj Zapotoczny

14 stycznia 2020

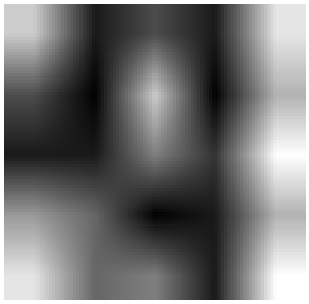
Spis treści

Spis rysunków

Listings

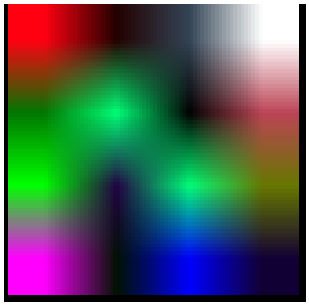
1 Zawartość paczki

- `main.c` - tutaj znajduje się funkcja 'main' oraz obsługa wyboru opcji
- `funkcje.c` - tutaj znajdują się funkcje do przetwarzania obrazów PGM i funkcja konwersji obrazu PPM do szarości
- `podstawa.c` - tutaj znajdują się funkcje: czytaj, wyświetl i zapisz
- `struktura.h` - tutaj jest struktura przechowująca informacje o obrazku
- `funkcjec.c, funkcjez.c, funkcjen.c` - funkcje do przetwarzania kolorów w obrazach PPM
- Pliki nagłówkowe - funkcje.h, podstawa.h, funkcjec.h, funkcjez.h, funkcjen.h
- `Makefile`
- `instrukcja.sh` - skrypt zawierający instrukcję obsługi programu (uruchamiany przez Makefile)
- Cztery napisane przeze mnie obrazki : 'a' 'c' (PGM), oraz 'b' 'd' (PPM)
  ('c' i 'd' są użyte tylko dla rozmywania histogramu i tam też są przedstawione)



Rys. 1: Obrazek "a" w powiększeniu

Obrazek "a" otwarty w notatniku:
P2
5 5
10
8 1 3 1 9
3 0 8 0 7
1 1 5 2 10
6 4 0 1 7
9 4 5 1 10



Rys. 2: Obrazek "b" w powiększeniu

Obrazek "b" otwarty w notatniku:
P3
4 4
15
15 0 1 2 0 0 3 4 5 15 15 15
0 7 0 0 15 7 0 0 0 11 4 5
0 15 0 2 0 4 0 15 7 6 7 0
15 0 15 0 1 0 0 0 15 1 0 3

2 Testy menu oraz podstawowego przetwarzania

2.1 Kompilacja

```
all:
  clear
  /bin/sh instrukcja.sh
  gcc -Wall -pedantic funkcje.c funkcje.h podstawa.c podstawa.h struktura.h
    main.c funkcjec.c funkcjec.h funkcjez.c funkcjez.h funkcjen.c funkcjen.h -lm
```

Listing 1: Makefile

Po uruchomieniu pliku Makefile (na diablo) program się kompiluje (bez ostrzeżeń i błędów). Ponadto uruchamia się skrypt Instrukcja.sh, który wypisuje opcje programu.

```
echo "Program do przetwarzania obrazow PGM i PPM";
echo;
echo "Podstawowe opcje:";
echo "-i <nazwa_pliku> --- nazwa pliku do wczytania (jesli zamiast nazwy podany
     bedzie znak '-' to zamiast z pliku obraz bedzie wczytywany ze standardowego
     wejscia";
echo "-o <nazwa_pliku> --- nazwa pliku do zapisu (jesli zamiast nazwy podany
    bedzie znak '-' to zamiast do pliku obraz bedzie wypisany na standardowym
    wyjsciu";
echo "-d ----------------- obraz po przetworzeniu powinien zostac wyswietlony";
echo "-m <kolor> --------- wybierz kolor do przetwarzania: r - czerwony, g -
    zielony, b - niebieski, s - konwersja do szarosci, brak opcji powoduje
    wykonanie operacji na wszyskich kolorach";
echo;
echo "Operacje szczegolowe:";
echo "-n ------------------- negatyw";
echo "-p <prog> ------------ progowanie z danym progiem, pamietaj ze prog jest
    w % (od 0 do 100)";
echo "-pc <prog> ----------- progowanie czerni z danym progiem, pamietaj ze
    prog jest w % (od 0 do 100)";
echo "-pb <prog> ----------- progowanie bieli z danym progiem, pamietaj ze prog
     jest w % (od 0 do 100)";
echo "-g <wspolczynnik> ---- korekcja gamma z danym wspolczynnikiem, pamietaj
    ze wspolczynnik musi byc nieujemny";
echo "-z <czern> <biel> ---- zmiana poziomow, pamietaj ze czern i biel sa w %
    (0,100) i ze czern < biel";
echo "-k ------------------- konturowanie";
echo "-rx ------------------ rozmywanie poziome";
echo "-ry ------------------ rozmywanie pionowe";
echo "-h ------------------- rozciaganie histogramu";
echo;
echo "Kolejnosc w jakiej moga wystapic opcje jest dowolna.";
echo "Opcja -i musi sie pojawic przy wywolaniu programu.";
echo "Znak '-' jest czescia kazdej opcji.";
echo "Przykladowe wywolanie programu: ./a.out -i nazwa_wczytaj -m g -k -o
    nazwa_zapis -d";
echo "Gdy pojawi sie opcaja -m dla obrazka PGM zostanie ona zignorowana!";
echo;
echo "Kody bledow (w razie gdyby cos poszlo nie tak): ";
echo "Niepoprawna opcja: -1";
echo "Brak nazwy:::::::: -2";
echo "Brak wartosci::::: -3";
echo "Brak pliku:::::::: -4";
```

Listing 2: Instrukcja.sh

W main.c znajdują się dwie funkcje odpowiadające za menu. Jedna z nich to 'opcje', a druga to 'argumenty'. Funkcja 'opcje' dba o to, aby zostały zaznaczone flagi odpowiednich zadań do wykonania, a funkcja argumenty dba o poprawność i przetwarzanie argumentów odpowiednich poleceń programu. Obie te funkcje zostały napisane na podstawie materiałów podanych na stronie kursu.

2.2 Listingi funkcji

```
1     /*************************************************************************/
2     /* Funkcja rozpoznaje opcje wywolania programu zapisane w tablicy argv  */
3     /* i zapisuje je w strukturze wybor                                      */
4     /* Skladnia opcji wywolania programu:                                    */
5     /*   program {[-i nazwa] [-o nazwa] [-m kolor] [-p liczba] [-pc liczba] */
6     /*   [-pb liczba] [-g liczba] [-z liczba liczba] [-n] [-rx] [-ry] [-k]  */
7     /*   [-h] [-d]}                                                          */
8     /* Argumenty funkcji:                                                    */
9     /*         argc  -  liczba argumentow wywolania wraz z nazwa programu   */
10    /*         argv  -  tablica argumentow wywolania                        */
11    /*         wybor -  struktura z informacjami o wywolanych opcjach       */
12    /* PRE:                                                                  */
13    /*      brak                                                             */
14    /* POST:                                                                 */
15    /*      funkcja otwiera odpowiednie pliki, zwraca uchwyty do nich        */
16    /*      w strukturze wybor, do tego ustawia na 1 pola, ktore             */
17    /*  poprawnie wystapily w linii wywolania programu,                      */
18    /*  pola opcji nie wystepujacych w wywolaniu ustawione sa na 0;          */
19    /*  zwraca wartosc OK, gdy wywolanie bylo poprawne                       */
20    /*  lub kod bledu zdefiniowany stalymi B_* (<0)                          */
21    /* UWAGA:                                                                */
22    /*      funkcja nie sprawdza istnienia i praw dostepu do plikow          */
23    /*      w takich przypadkach zwracane uchwyty maja wartosc NULL          */
24    /*************************************************************************/
25
26 int opcje (int argc, char **argv, w_opcje *wybor)
27 {
28     int i;
29     int  prog, progc, progb, czern, biel;  /* wartosci potrzebne w niektorych
      funkcjach */
30     double gamma;
31     char *nazwa_pliku_we, *nazwa_pliku_wy; /* zmienne przechowujace nazwy
      plikow wejsciowego i wyjsciowego */
32
33     wyzeruj_opcje(wybor);   /* wywolanie funkcji wyzerowania opcji  */
34     wybor->plik_wy=stdout;  /* na wypadek gdy nie podano opcji "-o" */
35
36     for (i=1; i<argc; i++)
37     {
38         if (argv[i][0] != '-')  /* blad: to nie jest opcja - brak znaku "-" */
39             return B_NIEPOPRAWNAOPCJA;
40
41
42         switch (argv[i][1])
43         {
44
45         //wczytywanie pliku
46         case 'i':            /* opcja z nazwa pliku wejsciowego */
```

```
47          {
48              if (++i<argc)    /* wczytujemy kolejny argument jako nazwe pliku */
49              {
50                  nazwa_pliku_we=argv[i];
51                  if (strcmp(nazwa_pliku_we,"-")==0)     /* gdy nazwa jest "-" */
52                      wybor->plik_we=stdin;       /* ustwiamy wejscie na stdin */
53                  else                            /* otwieramy wskazany plik   */
54                      wybor->plik_we=fopen(nazwa_pliku_we,"r");
55              }
56              else
57                  return B_BRAKNAZWY;    /* blad: brak nazwy pliku */
58              break;
59          }
60
61          //zapisywanie pliku
62          case 'o':           /* opcja z nazwa pliku wyjsciowego */
63          {
64              if (++i<argc)   /* wczytujemy kolejny argument jako nazwe pliku */
65              {
66                  nazwa_pliku_wy=argv[i];
67                  wybor->nazwa=argv[i];
68                  if (strcmp(nazwa_pliku_wy,"-")==0)   /* gdy nazwa jest "-"   */
69                      wybor->plik_wy=stdout;     /* ustwiamy wyjscie na stdout */
70                  else                           /* otwieramy wskazany plik    */
71                      wybor->plik_wy=fopen(nazwa_pliku_wy,"w");
72              }
73              else
74                  return B_BRAKNAZWY;       /* blad: brak nazwy pliku */
75              break;
76          }
77
78          //wybor koloru do edycji
79          case 'm':
80          {
81              if (++i<argc && (argv[i][0]=='r' || argv[i][0]=='b' || argv[i][0]==
    'g' || argv[i][0]=='s'))
82              {
83                  wybor->kolor=argv[i][0];     /* wczytujemy kolejny argument
    jako wartosc koloru */
84              }
85              else if (argv[i][0]=='-')
86                  return B_BRAKWARTOSCI;       /* blad: brak wartosci koloru */
87              else
88                  return B_NIEPOPRAWNAOPCJA;
89              break;
90          }
91
92          //wyswietlenie obrazu
93          case 'd':
94          {
95              wybor->wyswietlenie=1;
96              /* zmiana wartosci na 1, co spowoduje wykonanie */
97              break;
98          }
99
100         //progowania
101         case 'p':
```

```
102         {
103             if (argv[i][2]=='c')            /* wartosc progowania czerni */
104             {
105                 if (++i<argc)     /* wczytujemy kolejny argument jako prog */
106                 {
107                     if (sscanf(argv[i],"%d",&progc)==1)
108                     {
109                         wybor->pol_prog_czerni=1;
110                         /* zmiana wartosci na 1, co spowoduje wykonanie */
111                         wybor->progc=progc;
112                     }
113                     else
114                         return B_BRAKWARTOSCI;       /* blad: brak wartosci
    progu */
115                 }
116                 else
117                     return B_BRAKWARTOSCI;     /* blad: brak wartosci progu */
118             }
119             else if (argv[i][2]=='b')          /* wartosc progowania bieli  */
120             {
121                 if (++i<argc)       /* wczytujemy kolejny argument jako prog */
122                 {
123                     if (sscanf(argv[i],"%d",&progb)==1)
124                     {
125                         wybor->pol_prog_bieli=1;
126                         /* zmiana wartosci na 1, co spowoduje wykonanie */
127                         wybor->progb=progb;
128                     }
129                     else
130                         return B_BRAKWARTOSCI;  /* blad: brak wartosci progu */
131                 }
132                 else
133                     return B_BRAKWARTOSCI;     /* blad: brak wartosci progu */
134             }
135             else if (argv[i][2]==0)
136             {
137                 if (++i<argc)     /* wczytujemy kolejny argument jako progu */
138                 {
139                     if (sscanf(argv[i],"%d",&prog)==1)
140                     {
141                         wybor->progowanie=1;
142                         /* zmiana wartosci na 1, co spowoduje wykonanie */
143                         wybor->w_progu=prog;
144                     }
145                     else
146                         return B_BRAKWARTOSCI;   /* blad: brak progu */
147                 }
148                 else
149                     return B_BRAKWARTOSCI;  /* blad: brak progu */
150             }
151             break;
152         }
153 
154 
155         //negatyw
156         case 'n':
157         {
```

```
158             wybor->negatyw=1;
159             break;
160         }
161 
162 
163         //konturowanie
164         case 'k':
165         {
166             wybor->konturowanie=1;
167             break;
168         }
169 
170         //zmiana poziomow
171         case 'z':
172         {
173             if (++i<argc && argv[i][2]!='-')
174             {
175                 if (sscanf(argv[i],"%d",&czern)==1)
176                 {
177                     wybor->czern=czern;
178                     /* wczytujemy kolejny argument jako wartosc czerni */
179                 }
180                 else
181                     return B_BRAKWARTOSCI;
182             }
183 
184             if (++i<argc)
185             {
186                 if (sscanf(argv[i],"%d",&biel)==1)
187                 {
188                     wybor->biel=biel;
189                     /* wczytujemy kolejny argument jako wartosc bieli */
190                 }
191                 else
192                     return B_BRAKWARTOSCI;
193             }
194 
195             wybor->zmiana_poziomow=1;
196             break;
197         }
198 
199         //rozmywanie poziomow
200         case 'r':
201         {
202             if (argv[i][2]=='x')
203             {
204                 wybor->rozm_poziome=1;
205             }
206             else if (argv[i][2]=='y')
207             {
208                 wybor->rozm_pionowe=1;
209             }
210             else
211                 return B_NIEPOPRAWNAOPCJA;
212             break;
213         }
214 
```

```
215         //korekcja gamma
216         case 'g':
217         {
218             if (++i<argc)
219             {
220                 if (sscanf(argv[i],"%lf",&gamma)==1)
221                 {
222                     wybor->kor_gamma=1;
223                     wybor->gamma=gamma;
224                     /* wczytujemy kolejny argument jako wartosc gamma */
225                 }
226                 else
227                     return B_BRAKWARTOSCI;
228             }
229             else
230                 return B_BRAKWARTOSCI;
231             break;
232         }
233 
234         //rozciaganie histogramu
235         case 'h':
236         {
237             wybor->histogram=1;
238             break;
239         }
240 
241         default:
242             return B_NIEPOPRAWNAOPCJA;
243 
244         } /*koniec switch */
245     }     /* koniec for   */
246 
247     if (wybor->plik_we!=NULL)     /* ok: wej. strumien danych zainicjowany */
248         return OK;
249     else
250         return B_BRAKPLIKU;       /* blad:  nie otwarto pliku wejsciowego  */
251 }
```

Listing 3: Funkcja 'opcje'

Widzimy, że funkcja opcje nie tylko odpowiada za wyboór odpowiednich opcji, ale równierz za komunikaty błędów. Każda opcja w programie ma swoje wywołanie i tylko poprawne stosowanie sie do instrukcji daje możliwość wykonywania operacji na obrazach.

```
    /*****************************************************************/
    /* Funkcja rozpoznaje argumenty wywolania opcji i wykonuje na nich */
    /* operacje odwolujac sie do struktury obraz                       */
    /* Argumenty funkcji:                                              */
    /*         wybor - struktura z informacjami o wywolanych opcjach   */
    /* PRE:                                                            */
    /*    poprawnie wczytane opcje, aby mozna bylo odpowiednio dobrac  */
    /*    argumenty                                                    */
    /* POST:                                                           */
    /*    funkcja wykonuje wskazane operacje na odpowiednich formatach */
    /*    formatach obrazkow i odpowiednich kolorach (dla formatu PPM) */
    /*                                                                 */
    /*    dzieki strukturze obraz funkcja ma dostep do obrazka i       */
    /*    zmienia go poprzez odwolania do odpowiednich funkji          */
    /*****************************************************************/
void argumenty(w_opcje *wybor)
{
    ob obraz;    /* zmienna odwolujaca sie do struktury z pliku struktura.h */

    czytaj(wybor->plik_we,&obraz);
    fclose(wybor->plik_we);


   if(wybor->kolor == 's')
   /* dla kolor == s jest wykonywana funkcja konwersji */
    {
        if(obraz.znak != '2')
        {
            konwersja(&obraz);
        }
        else
        {
            printf("Nie wykonano konwersji, podales obrazek PGM \n");
        }
        /* warunek na wypadek checi uzytkownika do wykonania konwersji na
    szarym obrazku */
    }


    if(wybor->negatyw)          /* sprawdzenie wyboru uzytkownika */
    {
        if(obraz.znak == '2')   /* wykonanie funkcji ponizej dla obrazku PGM */
        {
            negatyw(&obraz);
        }
        else                    /* wykonanie funkcji ponizej dla obrazku PPM */
        {                       /* jezeli nie podamy dla ktorego koloru chcemy
    wykonac operacje, to operacje wykonuja sie dla wszystkich kolorow */
            if(wybor->kolor == 'r')
            negatywc(&obraz);
            /* funkcja dla koloru czerwonego (nazwa zakonczona na litere c) */
            else if(wybor->kolor == 'g')
            negatywz(&obraz);
            /* funkcja dla koloru zielonego (nazwa zakonczona na litere z) */
            else if(wybor->kolor == 'b')
            negatywn(&obraz);
```

```
55              /* funkcja dla koloru niebieskiego (nazwa zakonczona na litere n)
    */
56              else
57              {
58                  negatywc(&obraz);
59                  negatywz(&obraz);
60                  negatywn(&obraz);
61              }
62          }
63      }
64      /* Powyzsze komentarze odnosza sie do kazdej opcji znajdujacych sie
        wewnatrz funkcji argumenty */
65
66
67      if(wybor->konturowanie)
68      {
69          if(obraz.znak == '2')
70          {
71              konturowanie(&obraz);
72          }
73          else
74          {
75              if(wybor->kolor == 'r')
76              konturowaniec(&obraz);
77              else if(wybor->kolor == 'g')
78              konturowaniez(&obraz);
79              else if(wybor->kolor == 'b')
80              konturowanien(&obraz);
81              else
82              {
83                  konturowaniec(&obraz);
84                  konturowaniez(&obraz);
85                  konturowanien(&obraz);
86              }
87          }
88      }
89
90
91      if(wybor->progowanie)
92      {
93          if(obraz.znak == '2')
94          {
95              if (wybor->w_progu<=100 && wybor->w_progu>=0)
96              /* Warunek wynikajacy z podawania progu w procentach */
97              {
98                  progowanie(&obraz,wybor->w_progu);
99              }
100             else
101             {
102                 printf("Nie wykonano progowania, zla wartosc progu \n");
103             }
104         }
105         else
106         {
107             if (wybor->w_progu<=100 && wybor->w_progu>=0)
108             /* Warunek wynikajacy z podawania progu w procentach */
109             {
```

```
                if(wybor->kolor == 'r')
                progowaniec(&obraz,wybor->w_progu);
                else if(wybor->kolor == 'g')
                progowaniez(&obraz,wybor->w_progu);
                else if(wybor->kolor == 'b')
                progowanien(&obraz,wybor->w_progu);
                else
                {
                    progowaniec(&obraz,wybor->w_progu);
                    progowaniez(&obraz,wybor->w_progu);
                    progowanien(&obraz,wybor->w_progu);
                }
            }
            else
            {
                printf("Nie wykonano progowania, zla wartosc progu \n");
            }
        }
    }


    if(wybor->pol_prog_czerni)
    {
        if(obraz.znak == '2')
        {
            if (wybor->progc<=100 && wybor->progc>=0)
            /* Warunek wynikajacy z podawania progu w procentach */
            {
                pol_prog_czerni(&obraz,wybor->progc);
            }
            else
            {
                printf("Nie wykonano progowania, zla wartosc progu czerni \n");
            }
        }
        else
        {
            if (wybor->progc<=100 && wybor->progc>=0)
            /* Warunek wynikajacy z podawania progu w procentach */
            {
                if(wybor->kolor == 'r')
                pol_prog_czernic(&obraz,wybor->progc);
                else if(wybor->kolor == 'g')
                pol_prog_czerniz(&obraz,wybor->progc);
                else if(wybor->kolor == 'b')
                pol_prog_czernin(&obraz,wybor->progc);
                else
                {
                    pol_prog_czernic(&obraz,wybor->progc);
                    pol_prog_czerniz(&obraz,wybor->progc);
                    pol_prog_czernin(&obraz,wybor->progc);
                }
            }
            else
            {
                printf("Nie wykonano progowania, zla wartosc progu czerni \n");
            }
```

```
167         }
168     }
169 
170 
171     if(wybor->pol_prog_bieli)
172     {
173         if(obraz.znak == '2')
174         {
175             if (wybor->progb<=100 && wybor->progb>=0)
176             /* Warunek wynikajacy z podawania progu w procentach */
177             {
178             pol_prog_bieli(&obraz,wybor->progb);
179             }
180             else
181             {
182                 printf("Nie wykonano progowania, zla wartosc progu bieli \n");
183             }
184         }
185         else
186         {
187             if (wybor->progb<=100 && wybor->progb>=0)
188             /* Warunek wynikajacy z podawania progu w procentach */
189             {
190                 if(wybor->kolor == 'r')
191                 pol_prog_bielic(&obraz,wybor->progb);
192                 else if(wybor->kolor == 'g')
193                 pol_prog_bieliz(&obraz,wybor->progb);
194                 else if(wybor->kolor == 'b')
195                 pol_prog_bielin(&obraz,wybor->progb);
196                 else
197                 {
198                     pol_prog_bielic(&obraz,wybor->progb);
199                     pol_prog_bieliz(&obraz,wybor->progb);
200                     pol_prog_bielin(&obraz,wybor->progb);
201                 }
202             }
203             else
204             {
205                 printf("Nie wykonano progowania, zla wartosc progu bieli \n");
206             }
207         }
208     }
209 
210 
211     if(wybor->zmiana_poziomow)
212     {
213         if(obraz.znak == '2')
214         {
215             if ((wybor->czern>=0 && wybor->czern<=100) && (wybor->biel>=0 &&
    wybor->biel<=100) && wybor->czern<wybor->biel)
216             {               /* Warunki wynikajace z wlasnosci zmiany poziomow */
217                 zmiana_poziomow(&obraz,wybor->czern,wybor->biel);
218             }
219             else
220             {
221                 printf("Nie wykonano zmiany poziomow, nie spelnasz co najmniej
    jednego z warunkow:\n1. czern i biel naleza do przedzialu (0,100)\n2. czern
```

```
< biel \n");
            }
        }
        else
        {
            if ((wybor->czern>=0 && wybor->czern<=100) && (wybor->biel>=0 &&
wybor->biel<=100) && wybor->czern<wybor->biel)
            {               /* Warunki wynikajace z wlasnosci zmiany poziomow */
                if(wybor->kolor == 'r')
                zmiana_poziomowc(&obraz,wybor->czern,wybor->biel);
                else if(wybor->kolor == 'g')
                zmiana_poziomowz(&obraz,wybor->czern,wybor->biel);
                else if(wybor->kolor == 'b')
                zmiana_poziomown(&obraz,wybor->czern,wybor->biel);
                else
                {
                    zmiana_poziomowc(&obraz,wybor->czern,wybor->biel);
                    zmiana_poziomowz(&obraz,wybor->czern,wybor->biel);
                    zmiana_poziomown(&obraz,wybor->czern,wybor->biel);
                }
            }
            else
            {
                printf("Nie wykonano zmiany poziomow, nie spelnasz co najmniej
jednego z warunkow:\n1. czern i biel naleza do przedzialu (0,100)\n2. czern
< biel \n");
            }
        }
    }


    if(wybor->rozm_poziome)
    {
        if(obraz.znak == '2')
        {
            rozm_poziome(&obraz);
        }
        else
        {
            if(wybor->kolor == 'r')
            rozm_poziomec(&obraz);
            else if(wybor->kolor == 'g')
            rozm_poziomez(&obraz);
            else if(wybor->kolor == 'b')
            rozm_poziomen(&obraz);
            else
            {
                rozm_poziomec(&obraz);
                rozm_poziomez(&obraz);
                rozm_poziomen(&obraz);
            }
        }
    }


    if(wybor->rozm_pionowe)
    {
```

```
275         if(obraz.znak == '2')
276         {
277             rozm_pionowe(&obraz);
278         }
279         else
280         {
281             if(wybor->kolor == 'r')
282             rozm_pionowec(&obraz);
283             else if(wybor->kolor == 'g')
284             rozm_pionowez(&obraz);
285             else if(wybor->kolor == 'b')
286             rozm_pionowen(&obraz);
287             else
288             {
289                 rozm_pionowec(&obraz);
290                 rozm_pionowez(&obraz);
291                 rozm_pionowen(&obraz);
292             }
293         }
294     }
295
296
297     if(wybor->kor_gamma)
298     {
299         if(obraz.znak == '2')
300         {
301             if (wybor->gamma<0)
302             /* Warunek wynikajacy z wlasnosci korekcji gamma */
303             {
304                 kor_gamma(&obraz,wybor->gamma);
305             }
306             else
307             {
308                 printf("Nie wykonano korekcji gamma, zla wartosc wspolczynnika
    \n");
309             }
310         }
311         else
312         {
313             if (wybor->gamma<0)
314             /* Warunek wynikajacy z wlasnosci korekcji gamma */
315             {
316                 if(wybor->kolor == 'r')
317                 kor_gammac(&obraz,wybor->gamma);
318                 else if(wybor->kolor == 'g')
319                 kor_gammaz(&obraz,wybor->gamma);
320                 else if(wybor->kolor == 'b')
321                 kor_gamman(&obraz,wybor->gamma);
322                 else
323                 {
324                     kor_gammac(&obraz,wybor->gamma);
325                     kor_gammaz(&obraz,wybor->gamma);
326                     kor_gamman(&obraz,wybor->gamma);
327                 }
328             }
329             else
330             {
```

```
331                 printf("Nie wykonano korekcji gamma, zla wartosc wspolczynnika
    \n");
332             }
333         }
334     }
335 
336 
337     if(wybor->histogram)
338     {
339         if(obraz.znak == '2')
340         {
341             histogram(&obraz);
342         }
343         else
344         {
345             if(wybor->kolor == 'r')
346             histogramc(&obraz);
347             else if(wybor->kolor == 'g')
348             histogramz(&obraz);
349             else if(wybor->kolor == 'b')
350             histogramn(&obraz);
351             else
352             {
353                 histogramc(&obraz);
354                 histogramz(&obraz);
355                 histogramn(&obraz);
356             }
357         }
358     }
359 
360 
361     zapis(wybor->plik_wy,&obraz);
362     fclose(wybor->plik_wy);
363 
364     if(wybor->wyswietlenie)
365     {
366         wyswietl(wybor->nazwa);
367     }
368 }
```

Listing 4: Funkcja 'argumenty'

2.3 Funkcje czytaj (-i), wyświetl (-d) i zapisz (-o)

Testy mają na celu sprawdzenie działania menu dla opcji: -i, -o i -d oraz funkcji wczytaj, zapisz i wyświetl.

-Sprawdenie działania standardowego.
>Wprowadzam do programu ciąg znaków: ./a.out -i a -o nazwa -d

-Sprawdenie działania standardowego.
>Wprowadzam do programu ciąg znaków: ./a.out -i b -o nazwa -d
Obrazek został poprawnie wczytany, zapisany i wyświetlony (choć trzeba czekać chwilę na efekt działania).
Zarówno dla obrazka PGM ("a") jak i dla PPM ("b") te trzy wywołania oraz funkcje działają prawidłowo.

-Sprawdenie działania dla innej kolejności wywołania.
>Wprowadzam do programu ciąg znaków: ./a.out -d -i b -o nazwa
Widać, że kolejność wprowadzania wywołań opcji nie ma znaczenia, bo również działa poprawnie.

-Sprawdenie działania (-i) dla nazwy obrazka '-'.
>Wprowadzam do programu ciąg znaków: ./a.out -i - -o nazwa -d
Potem wprowadzam:
P2
5 5
10
8 1 3 1 9
3 0 8 0 7
1 1 5 2 10
6 4 0 1 7
9 4 5 1 10
Zgodnie z wymogami, gdy w miejscu nazwy obrazka dla -i jest '-' to jest on wczytywany ze standardowego wejścia.

-Sprawdenie działania (-o) dla nazwy obrazka '-'.
>Wprowadzam do programu ciąg znaków: ./a.out -i a -o - -d
Zostało wypisane:
P2
5 5
10
8 1 3 1 9
3 0 8 0 7
1 1 5 2 10
6 4 0 1 7
9 4 5 1 10
Zgodnie z wymogami, gdy w miejscu nazwy obrazka dla -o jest '-' to jest on wypisany na standardowe wyjście.

-Sprawdenie komunikatów o błędach.

>Wprowadzam do programu ciąg znaków: ./a.out -i -d
Wyświetlił się komunikat: "Blad. Kod bledu: -4", a więc brak pliku (nie ma pliku o nazwie '-d'), co jest poprawnym działaniem programu.

>Wprowadzam do programu ciąg znaków: ./a.out -i
Wyświetlił się komunikat: "Blad. Kod bledu: -2", a więc brak nazwy, co jest poprawnym działaniem programu, bo nie podałem nazwy pliku do wczytania.

>Wprowadzam do programu ciąg znaków: ./a.out -i a -o
Wyświetlił się komunikat: "Blad. Kod bledu: -2", a więc brak nazwy, co jest poprawnym działaniem programu, bo nie podałem nazwy pliku do zapisu.

Stwierdzam, że menu dla operacji -i, -o, -d działa poprawnie oraz funkcje zapis, czytaj i wyswietl również działają poprawnie.

```
1  int czytaj(FILE *plik_we,ob *obrazy)
2  {
3    char buf[DL_LINII]; /* bufor pomocniczy do czytania naglowka i komentarzy */
4    int znak;           /* zmienna pomocnicza do czytania komentarzy */
5    int koniec=0;       /* czy napotkano koniec danych w pliku */
6    int i,j;
7
8    /*Sprawdzenie czy podano prawidlowy uchwyt pliku */
9    if (plik_we==NULL)
10   {
11     fprintf(stderr,"Blad: Nie podano uchwytu do pliku\n");
12     return(0);
13   }
14
15   /* Sprawdzenie "numeru magicznego" - powinien byc P2 lub P3 */
16   if (fgets(buf,DL_LINII,plik_we)==NULL)
17   /* Wczytanie pierwszej linii pliku do bufora */
18     koniec=1;       /* Nie udalo sie? Koniec danych! */
19
20   if ((buf[0]!='P') || (buf[1]!='2' && buf[1]!='3') || koniec)
21   /* Czy jest magiczne "P2" lub "P3"? */
22   {
23     fprintf(stderr,"Blad: To nie jest plik PGM, ani PPM\n");
24     /* znak dla P2 i P3 */
25     return(0);
26   }
27
28   obrazy->znak=buf[1];
29   /* Przechowywanie magicznego znaku obrazka w strukturze */
30
31   /* Pominiecie komentarzy */
32   do
33   {
34     if ((znak=fgetc(plik_we))=='#')
35     /* Czy linia rozpoczyna sie od znaku '#'? */
```

```
36    {
37      if (fgets(buf,DL_LINII,plik_we)==NULL)   /* Przeczytaj ja do bufora */
38      koniec=1;          /* Zapamietaj ewentualny koniec danych    */
39    }
40    else
41    {
42      ungetc(znak,plik_we); /* Gdy przeczytany znak z poczatku linii */
43    }                       /* nie jest '#' zwroc go                 */
44  }while (znak=='#' && !koniec);    /* Powtarzaj dopoki sa linie komentarza  */
45                                    /* i nie nastapil koniec danych          */
46
47  /* Pobranie wymiarow obrazu i liczby odcieni szarosci */
48  if (fscanf(plik_we,"%d %d %d",&obrazy->wymx,&obrazy->wymy,&obrazy->szarosci)
      !=3)
49  /* Poprzez odcienie szarosci sa tutaj rozumiane rowniez odcienie innych
      kolorow */
50  {
51    fprintf(stderr,"Blad: Brak wymiarow obrazu lub liczby stopni szarosci\n");
52    return(0);   /* Brak liczby odcieni */
53  }
54
55
56  /* Tablica dynamiczna do przechowywania obrazka */
57  obrazy->tab=malloc(obrazy->wymy*sizeof(int*));
58  /* Rezerwowanie miejsca w pamieci na obrazek */
59  for (i=0;i<obrazy->wymy;i++)
60  {
61    obrazy->tab[i]=malloc(obrazy->wymx*sizeof(int));
62  }
63
64
65
66  /* Pobranie obrazu i zapisanie w tablicy */
67  /* Dla obrazu P2 */
68  if(obrazy->znak=='2')
69  {
70    for (i=0;i<obrazy->wymy;i++)
71    {
72      for (j=0;j<obrazy->wymx;j++)
73      {
74        if (fscanf(plik_we,"%d",&(obrazy->tab[i][j]))!=1)
75        {
76          fprintf(stderr,"Blad: Niewlasciwe wymiary obrazu\n");
77          return(0);
78        }
79      }
80    }
81    return obrazy->wymx* obrazy->wymy;   /* Czytanie zakonczone sukcesem */
82  }                                      /* Zwroc liczbe pikseli */
83  /* Dla obrazu P3 */
84  else
85  {   /* Tworzenie trzech roznych tablic dynamicznych na kolory RGB */
86    obrazy->czer=malloc(obrazy->wymy*sizeof(int*));
87    obrazy->ziel=malloc(obrazy->wymy*sizeof(int*));
88    obrazy->nieb=malloc(obrazy->wymy*sizeof(int*));
89
90    for (i=0;i<obrazy->wymy;i++)
```

```
91      {
92        obrazy->czer[i]=malloc(obrazy->wymx*sizeof(int));
93        obrazy->ziel[i]=malloc(obrazy->wymx*sizeof(int));
94        obrazy->nieb[i]=malloc(obrazy->wymx*sizeof(int));
95      }
96
97      for (i=0;i<obrazy->wymy;i++)
98      {
99        for (j=0;j<obrazy->wymx;j++)
100       {
101       /* Pobieranie kolejnych informacji o kolorach do trzech roznych tablic
        dynamicznych */
102         if (fscanf(plik_we,"%d %d %d",&(obrazy->czer[i][j]),&(obrazy->ziel[i][j
        ]),&(obrazy->nieb[i][j]))!=3)
103         {
104           fprintf(stderr,"Blad: Niewlasciwe wymiary obrazu\n");
105           return(0);
106         }
107       }
108     }
109   }
110   return obrazy->wymx* obrazy->wymy;    /* Czytanie zakonczone sukcesem */
111 }                                       /* Zwroc liczbe pikseli */
112
113
114
115
116
117
118 void wyswietl(char *n_pliku)
119 {
120   char polecenie[DL_LINII];       /* bufor pomocniczy do zestawienia polecenia
        */
121
122   strcpy(polecenie,"display ");   /* konstrukcja polecenia postaci */
123   strcat(polecenie,n_pliku);      /* display "nazwa_pliku" &       */
124   strcat(polecenie," &");
125   printf("%s\n",polecenie);       /* wydruk kontrolny polecenia */
126   system(polecenie);              /* wykonanie polecenia        */
127 }
128
129
130
131
132
133
134
135 void zapis (FILE* plik,ob *obrazy)
136 {
137
138   if(obrazy->znak=='2')
139   {
140   fprintf (plik, "P2\n");
141   /* Wypisanie do pliku znaku magicznego w 1 linijce */
142   fprintf (plik, "%d %d\n",obrazy->wymx, obrazy->wymy);
143   /* Wypisanie do pliku wymiarow obrazka w 2 linijce */
144   fprintf (plik, "%d\n", obrazy->szarosci);
```

```
145   /* Wypisanie do pliku liczby mowiacej o ilosci odcieni szarosci   */
146     int i,j;
147     for (i=0;i<obrazy->wymy;i++)
148     /* Wypisywanie do pliku informacji o kolorze kolejnych pikseli,   */
149     {    /* za pomoca zwyklej petli for */
150       for (j=0; j< obrazy->wymx; j++)
151       fprintf (plik, "%d ", obrazy->tab[i][j]);
152       /* Bardzo wazna w tym miejscu jest spacja po '%d', gdyz bez niej  */
153       /* nie bedzie odstepu pomiedzy kolejnymi informacjami o pikselach */
154     }            /* i obrazek nie wyswietli sie poprawnie */
155   }
156   else
157   {
158   fprintf (plik, "P3\n");
159   /* Wypisanie do pliku znaku magicznego w 1 linijce */
160   fprintf (plik, "%d %d\n",obrazy->wymx, obrazy->wymy);
161   /* Wypisanie do pliku wymiarow obrazka w 2 linijce */
162   fprintf (plik, "%d\n", obrazy->szarosci);
163   /* Wypisanie do pliku liczby mowiacej o ilosci odcieni kolorow (zmienna
       szarosci) */
164   int i,j;
165   for (i=0;i<obrazy->wymy;i++)
166     {
167       for (j=0; j< obrazy->wymx; j++)
168       fprintf (plik, "%d %d %d ", obrazy->czer[i][j],obrazy->ziel[i][j],obrazy
      ->nieb[i][j]);
169     }       /* Wypisywanie do pliku informacji o kolorze kolejnych pikseli */
170   }
171 }
```

Listing 5: Funkcje: czytaj; wyświetl; zapisz

2.4 Opcja -m i funkcja konwersji do szarości

Testy mają na celu sprawdzeniee działania menu dla opcji: -m oraz sprawdzenie funkcji konwersji do szarości.

-Sprawdzenie działania funkcji konwersji do szarości.
>Wprowadzam do programu ciąg znaków: ./a.out -i b -m s -o nazwa -d
Wyświetlił się obrazek, po konwersji do szarości, opcja działa poprawnie, bo wczytany obrazek był w formacie PPM.

Obrazek "b" przed konwersją:

```
P3
4 4
15
15 0 1      2  0  0      3  4 5      15 15 15
0  7  0     0 15  7     0  0 0      11  4  5
0 15 0      2  0  4     0 15 7      6   7  0
15 0 15     0  1  0     0  0 15     1   0  3
```

Obrazek "b" po konwersji:

```
P2
4 4
15
```

5 0 4 15 2 7 0 6 5 2 7 4 10 0 5 1

```c
/* Konwersja */
void konwersja (ob *obrazpgm)
{
  int i,j;
  obrazpgm->znak='2';   /* zmiana magicznego znaku obrazka na P2 */

  /* Tablica dynamiczna do przechowywania obrazka */
  obrazpgm->tab=malloc(obrazpgm->wymy*sizeof(int*));
  /* deklarowanie dwuwymiarowej tablicy dynamicznej */
  for (i=0;i<obrazpgm->wymy;i++)      /* 'obrazpgm' to nazwa zmiennej */
  {
    obrazpgm->tab[i]=malloc(obrazpgm->wymx*sizeof(int));
  }

  for (i=0;i<obrazpgm->wymy;i++)
  for (j=0;j<obrazpgm->wymx;j++)
  obrazpgm->tab[i][j]=(obrazpgm->czer[i][j]+obrazpgm->nieb[i][j]+obrazpgm->ziel
    [i][j])/3;
}     /* zapisanie obrazka wedlug wzoru podanego na stronie kursu */
```

Listing 6: Konwersja do szarości

-Sprawdzenie błędu, wynikającego z wprowadzenia obrazka PGM dla funkcji konwersji.
>Wprowadzam do programu ciąg znaków: ./a.out -i a -m s -o nazwa -d
Według tych opcji oczekiwałem wykonania konwersji do szarosci na obrazku PGM, ale program odpowiednio zareagował wyświetlając błąd: "Nie wykonano konwersji, podales obrazek PGM", co jest poprawną reakcją programu.

-Sprawdzenie działania funkcji -m dla pojedyńczego koloru.
>Wprowadzam do programu ciąg znaków: ./a.out -i b -m r -n -o nazwa -d
Wyświetlił się obrazek, po wykonaniu negatywu dla koloru czerwonego, czyli poprawne działanie. Dla innych kolorów edytowanych osobno ta funkcja również działa poprawnie, co będzie zobrazowane w dalszej części sprawozdania.

Obrazek "b" po wykonaniu operacji:
P3
4 4
15
0 0 1 13 0 0 12 4 5 0 15 15 15 7 0 15 15 7 15 0 0 4 4 5 15 15 0 13 0 4 15 15 7 9 7 0 0 0 15 15 1 0 15 0 15 14 0 3

Widzimy, że operacja została wykonana na co trzeciej liczbie, a więc została wykonana tylko dla koloru czerwonego, co jest poprawnym działaniem opcji -m oraz odwołania do barwy.

-Sprawdzenie działania programu dla braku opcji -m dla obrazka PPM.
>Wprowadzam do programu ciąg znaków: ./a.out -i b -n -o nazwa -d
Wyświetlił się obrazek, po wykonaniu negatywu dla wszyskich barw, czyli działanie według zaleceń prowadzącego.

Obrazek "b" po wykonaniu operacji:
P3

4 4
15
0 15 14 13 15 15 12 11 10 0 0 0 15 8 15 15 0 8 15 15 15 4 11 10 15 0 15 13 15 11 15 0 8 9 8 15 0 15 0 15 14 15 15 15 0 14 15 12

Widzimy, że podana funkcja wykonała się dla każdego koloru, według zadanego wzoru, a zatem brak opcji -m oraz odwołania do barw działa tutaj poprawnie.

-Sprawdzenie działania funkcji -m dla obrazka PGM.
>Wprowadzam do programu ciąg znaków: ./a.out -i a -m b -n -o nazwa -d
Wyświetlił się negatyw obrazka "a" (biało-czarny), bez udziału koloru niebieskiego, czyli zgodnie z oczekiwaniami opcja -m została zignorowana.

-Sprawdzenie działania komunikatów o błędach.

>Wprowadzam do programu ciąg znaków: ./a.out -i a -m -n -o nazwa -d
Wyświetlił się komunikat: "Blad. Kod bledu: -3", a więc brak wartości, co jest poprawnym działaniem programu, bo nie podałem argumentu dla opcji -m.

>Wprowadzam do programu ciąg znaków: ./a.out -i a -m j -n -o nazwa -d
Wyświetlił się komunikat: "Blad. Kod bledu: -1", a więc niepoprawna opcja, co jest poprawnym działaniem programu, bo nie ma opcji wywołania "j" dla opcji -m.

3 Testy funkcji przetwarzających obraz

Przypomnienie wyglądu obrazków "a" i "b".

Obrazek "a" otwarty w notatniku:

```
P2
5 5
10
8 1 3 1 9
3 0 8 0 7
1 1 5 2 10
6 4 0 1 7
9 4 5 1 10
```

Obrazek "b" otwarty w notatniku:

```
P3
4 4
15
15 0 1     2  0  0     3  4 5     15 15 15
0  7  0     0 15 7     0  0 0     11  4  5
0 15 0     2  0  4     0 15 7     6  7  0
15 0 15     0  1  0     0  0 15     1  0  3
```

Funkcje przetwarzania obrazków PPM będę testował dla jednego losowego koloru oraz dla wszystkich kolorów naraz.

3.1 Negatyw (-n)

Funkcja negatyw nie potrzebuje żadnych dodatkowych argumentów, tylko obrazek.

```
1  /* Negatyw dla PGM*/
2  void negatyw (ob *obrazpgm)
3  {
4    int i,j;
5    for (i=0;i<obrazpgm->wymy;i++)
6    for (j=0;j<obrazpgm->wymx;j++)
7    obrazpgm->tab[i][j]=obrazpgm->szarosci-obrazpgm->tab[i][j];
8  }
9
10 /* Negatyw dla PPM*/
11 void negatywc (ob *obraz)
12 {
13   int i,j;
14   for (i=0;i<obraz->wymy;i++)
15   for (j=0;j<obraz->wymx;j++)    /*szarosci rozumiane jako liczba odcieni */
16   obraz->czer[i][j]=obraz->szarosci-obraz->czer[i][j];
17 }
18 /* Funkcje dla obrazow PGM i PPM roznia sie tylko tablica do ktorej sa
     zapisywane piksele, wiec dalej bede podawal tylko jeden przyklad listingu */
```

Listing 7: Negatyw

Poniższe testy mają na celu sprawdzenie funkcji negatyw.

[PGM] Wprowadzam do programu ciąg znaków: ./a.out -i a -n -o nazwa -d
Obrazek "a" po wykonaniu operacji:

P2
5 5
10
2 9 7 9 1 7 10 2 10 3 9 9 5 8 0 4 6 10 9 3 1 6 5 9 0

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i b -n -m r -o nazwa -d
Wykonaie operacji na kolorze czerwonym.
Obrazek "b" po wykonaniu operacji:

P3
4 4
15
0 0 1 13 0 0 12 4 5 0 15 15 15 7 0 15 15 7 15 0 0 4 4 5 15 15 0 13 0 4 15 15 7 9 7 0 0 0 15 15 1 0
15 0 15 14 0 3

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i b -n -o nazwa -d
Obrazek "b" po wykonaniu operacji:

P3
4 4
15
0 15 14 13 15 15 12 11 10 0 0 0 15 8 15 15 0 8 15 15 15 4 11 10 15 0 15 13 15 11 15 0 8 9 8 15 0
15 0 15 14 15 15 15 0 14 15 12

Jak widać w każdym z przypadków program poprawnie zmienia wartości poszczególnych pikseli obrazków według wzoru podanego na stronie kursu. Funkcje obsługujące negatyw dla wszystkich barw działają poprawnie.

3.2 Konturowanie (-k)

Funkcja konturowanie nie potrzebuje żadnych dodatkowych argumentów, tylko obrazek.

```
/* Konturowanie */
void konturowaniec (ob *obraz)
{
  int i,j;
  for (i=0;i<obraz->wymy-1;i++)
    /* w tym miejscu '-1' w petli, bo w przeciwnym wypadku przejdziemy poza
    zakres tablicy (segmentation fault), a 1 piksel doda tylko ramke o grubosci
    1 piksela */
  for (j=0;j<obraz->wymx-1;j++)
      obraz->czer[i][j]=abs(obraz->czer[i+1][j]-obraz->czer[i][j])+abs(obraz->
    czer[i][j+1]-obraz->czer[i][j]);
}     /* 'abs' z biblioteki matematycznej to modul */
```

Listing 8: Konturowanie

Poniższe testy mają na celu sprawdzenie funkcji konturowanie.

[PGM] Wprowadzam do programu ciąg znaków: ./a.out -i a -k -o nazwa -d

Obrazek "a" po wykonaniu operacji:

P2
5 5
10
12 3 7 9 9 5 9 11 9 7 5 7 8 9 10 5 4 6 6 7 9 4 5 1 10

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i b -k -m r -o nazwa -d
Wykonaie operacji na kolorze czerwonym.

Obrazek "b" po wykonaniu operacji:

P3
4 4
15
28 0 1 3 0 0 15 4 5 15 15 15 0 7 0 2 15 7 11 0 0 11 4 5 17 15 0 4 0 4 6 15 7 6 7 0 15 0 15 0 1 0 0 0 15 1 0 3

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i b -k -o nazwa -d

Obrazek "b" po wykonaniu operacji:

P3
4 4
15
28 7 2 3 19 12 15 15 15 15 15 15 0 16 7 2 30 10 11 19 12 11 4 5 17 30 19 4 16 7 6 23 15 6 7 0 15 0 15 0 1 0 0 0 15 1 0 3

Jak widać w każdym z przypadków program poprawnie zmienia wartości poszczególnych pikseli obrazków według wzoru podanego na stronie kursu. Funkcje obsługujące konturowanie dla wszystkich barw działają poprawnie.

3.3 Progowanie (-p)

Funkcja progowanie wymaga podania argumentu - progu oraz informacji o obrazku.

```
/* Progowanie */
void progowanien (ob *obraz,int prog)
{
 int i,j;
 for (i=0;i<obraz->wymy;i++)
 for (j=0;j<obraz->wymx;j++)
  if(obraz->nieb[i][j]<=obraz->szarosci*prog/100.0)
  /* ten if sprawdza, czy dany piksel jest pod czy nad podanym progiem w % */
  {
    obraz->nieb[i][j]=0;
  }
  /* kazda wartosc, ktora jest pod progiem dostaje wartosc 0, a kazda, ktora
    jest nad progiem otrzymuje wartosc rowna maksymalnemu odcieniowi szarosci */
  else
  {
    obraz->nieb[i][j]=obraz->szarosci;
  }                  /* szarosci jest rozumiane jako liczba odcieni */
}
```

Listing 9: Progowanie

Poniższe testy mają na celu sprawdzenie funkcji progowanie.

[PGM] Wprowadzam do programu ciąg znaków: ./a.out -i a -p 20 -o nazwa -d
Obrazek "a" po wykonaniu operacji:

P2
5 5
10
10 0 10 0 10 10 0 10 0 10 0 0 10 0 10 10 10 0 0 10 10 10 10 0 10

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i b -p 20 -m b -o nazwa -d
Wykonanie operacji na kolorze niebieskim.
Obrazek "b" po wykonaniu operacji:

P3
4 4
15
15 0 0 2 0 0 3 4 15 15 15 15 0 7 0 0 15 15 0 0 0 11 4 15 0 15 0 2 0 15 0 15 15 6 7 0 15 0 15 0 1 0 0 0 15 1 0 0

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i b -p 20 -o nazwa -d
Obrazek "b" po wykonaniu operacji:

P3
4 4
15
15 0 0 0 0 0 0 15 15 15 15 15 0 15 0 0 15 15 0 0 0 15 15 15 0 15 0 0 0 15 0 15 15 15 15 0 15 0 15 0 0 0 0 0 15 0 0 0

[BŁĘDNE DANE] Wprowadzam do programu ciąg znaków: ./a.out -i b -p -o nazwa -d
Wyświetlił się komunikat: "Blad. Kod bladu: -3", a więc brak wartości, co jest prawdą, bo nie podałem wartości progu.

[BŁĘDNE DANE] Wprowadzam do programu ciąg znaków: ./a.out -i b -p 110 -o nazwa -d
Wyświetlił się komunikat: "Nie wykonano progowania, zla wartosc progu", a więc program dobrze rozpoznaje błędy.

Jak widać w każdym z przypadków program poprawnie zmienia wartości poszczególnych pikseli obrazków według wzoru podanego na stronie kursu. Funkcje obsługujące progowanie dla wszystkich barw działają poprawnie.

3.4 Progowanie bieli (-pb)

Funkcja progowanie bieli wymaga podanie argumentu - progu bieli oraz informacji o obrazku.

```
1  /* Polprogowanie bieli */
2  void pol_prog_bieli (ob *obrazpgm,int progb)
3  {
4   int i,j;
5   for (i=0; i<obrazpgm->wymy; i++)
6   for (j=0; j<obrazpgm->wymx; j++)
7    if (obrazpgm->tab[i][j]>=obrazpgm->szarosci*progb/100.0)
8    /* ten if sprawdza, czy dany piksel jest pod czy nad podanym progiem w % */
9    {
10     obrazpgm->tab[i][j]=obrazpgm->szarosci;
```

```
11      /* kazda wartosc, ktora jest nad progiem dostaje wartosc maks liczby
        odcieni */
12    }
13  }
```

Listing 10: Progowanie bieli

Poniższe testy mają na celu sprawdzenie funkcji progowania bieli.

[PGM] Wprowadzam do programu ciąg znaków: ./a.out -i a -pb 20 -o nazwa -d
Obrazek "a" po wykonaniu operacji:

P2
5 5
10
10 1 10 1 10 10 0 10 0 10 1 1 10 10 10 10 10 0 1 10 10 10 10 1 10

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i b -pb 20 -m b -o nazwa -d
Wykonanie operacji na kolorze niebieskim.
Obrazek "b" po wykonaniu operacji:

P3
4 4
15
15 0 1 2 0 0 3 4 15 15 15 15 0 7 0 0 15 15 0 0 0 11 4 15 0 15 0 2 0 15 0 15 15 6 7 0 15 0 15 0 1 0 0 0 15 1 0 15

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i b -pb 20 -o nazwa -d
Obrazek "b" po wykonaniu operacji:

P3
4 4
15
15 0 1 2 0 0 15 15 15 15 15 15 0 15 0 0 15 15 0 0 0 15 15 15 0 15 0 2 0 15 0 15 15 15 15 0 15 0 15 0 1 0 0 0 15 1 0 15

[BŁĘDNE DANE] Wprowadzam do programu ciąg znaków: ./a.out -i b -pb -o nazwa -d
Wyświetlił się komunikat: "Blad. Kod bladu: -3", a więc brak wartości, co jest prawdą, bo nie podałem wartości progu bieli.

[BŁĘDNE DANE] Wprowadzam do programu ciąg znaków: ./a.out -i b -pb 110 -o nazwa -d
Wyświetlił się komunikat: "Nie wykonano progowania, zla wartosc progu bieli", a więc program dobrze rozpoznaje błędy.

Jak widać w każdym z przypadków program poprawnie zmienia wartości poszczególnych pikseli obrazków według wzoru podanego na stronie kursu. Funkcje obsługujące progowanie bieli dla wszystkich barw działają poprawnie.

3.5 Progowanie czerni (-pc)

Funkcja progowanie czerni wymaga podanie argumentu - progu czerni oraz informacji o obrazku.

```
1  /* Polprogowanie czerni */
2  void pol_prog_czerniz (ob *obraz,int progc)
```

```
{
 int i,j;
 for (i=0; i<obraz->wymy; i++)
 for (j=0; j<obraz->wymx; j++)
  if (obraz->ziel[i][j]<=obraz->szarosci*progc/100.0)
  /* ten if sprawdza, czy dany piksel jest pod czy nad podanym progiem w % */
  {
      obraz->ziel[i][j]=0;  /* kazda wartosc, ktora jest pod progiem dostaje
    wartosc 0  */
  }
}
```

Listing 11: Progowanie czerni

Poniższe testy mają na celu sprawdzenie funkcji progowanie czerni.

[PGM] Wprowadzam do programu ciąg znaków: ./a.out -i a -pc 50 -o nazwa -d
Obrazek "a" po wykonaniu operacji:

P2
5 5
10
8 0 0 0 9 0 0 8 0 7 0 0 0 0 10 6 0 0 0 7 9 0 0 0 10

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i b -pc 40 -m g -o nazwa -d
Wykonanie operacji dla koloru zielonego.
Obrazek "b" po wykonaniu operacji:

P3
4 4
15
15 0 1 2 0 0 3 0 5 15 15 15 0 7 0 0 15 7 0 0 0 11 0 5 0 15 0 2 0 4 0 15 7 6 7 0 15 0 15 0 0 0 0 0
15 1 0 3

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i b -pc 40 -o nazwa -d
Obrazek "b" po wykonaniu operacji:

P3
4 4
15
15 0 0 0 0 0 0 0 0 15 15 15 0 7 0 0 15 7 0 0 0 11 0 0 0 15 0 0 0 0 0 15 7 0 7 0 15 0 15 0 0 0 0 0
15 0 0 0

[BŁĘDNE DANE] Wprowadzam do programu ciąg znaków: ./a.out -i b -pc -o nazwa -d
Wyświetlił się komunikat: "Blad. Kod bladu: -3", a więc brak wartości, co jest prawdą, bo nie podałem wartości progu czerni.

[BŁĘDNE DANE] Wprowadzam do programu ciąg znaków: ./a.out -i b -pc 110 -o nazwa -d
Wyświetlił się komunikat: "Nie wykonano progowania, zla wartosc progu czerni", a więc program dobrze rozpoznaje błędy.

Jak widać w każdym z przypadków program poprawnie zmienia wartości poszczególnych pikseli obrazków według wzoru podanego na stronie kursu. Funkcje obsługujące progowanie czerni dla wszystkich barw działają poprawnie.

3.6 Korekcja gamma (-g)

Funkcja korekcja gamma wymaga podania argumentu - współczynnika gamma oraz informacji o obrazku.

```
/* Korekcja gamma */
void kor_gammac (ob *obraz,double gamma)
{
  int i,j;
  double max=obraz->czer[0][0];       /* szukamy wartosci maksymalnej */

  for (i=0; i<obraz->wymy; i++)
  for (j=0; j<obraz->wymx; j++)
  {
      if (max<obraz->czer[i][j])
          max=obraz->czer[i][j];
  }

  double a,b,x;
  /* zmienne pomocnicze, uzyte do realizacji wzoru podanego na stronie kursu */
  for (i=0; i<obraz->wymy-1; i++)
  for (j=0; j<obraz->wymx-1; j++)
  {
    a=(obraz->czer[i][j])/max;
    b=(1.0)/gamma;
    /* 1.0 w tym miejscu na wszelki wypadek aby uniknac bledow rzutowania */
    x=pow(a,b);
    /* pow oznacza podnoszenie wartosci 'a' do potegi 'b' */
    obraz->czer[i][j]=x*max;
  }
}
```

Listing 12: Korekcja gamma

Poniższe testy mają na celu sprawdzenie funkcji korekcja gamma.

[PGM] Wprowadzam do programu ciąg znaków: ./a.out -i a -g 3 -o nazwa -d
Obrazek "a" po wykonaniu operacji:

P2
5 5
10
9 4 6 4 9 6 0 9 0 7 4 4 7 5 10 8 7 0 4 7 9 4 5 1 10

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i b -g 3 -m g -o nazwa -d
Wykoananie operacji na kolorze zielonym.
Obrazek "b" po wykonaniu operacji:

P3
4 4
15
15 0 1 2 0 0 3 9 5 15 15 15 0 11 0 0 15 7 0 0 0 11 4 5 0 15 0 2 0 4 0 15 7 6 7 0 15 0 15 0 1 0 0 0
15 1 0 3

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i b -g 3 -o nazwa -d
Obrazek "b" po wykonaniu operacji:

P3

4 4
15
15 0 6 7 0 0 8 9 10 15 15 15 0 11 0 0 15 11 0 0 0 11 4 5 0 15 0 7 0 9 0 15 11 6 7 0 15 0 15 0 1 0 0 0 15 1 0 3

[BŁĘDNE DANE] Wprowadzam do programu ciąg znaków: ./a.out -i b -g -o nazwa -d Wyświetlił się komunikat: "Blad. Kod bladu: -3", a więc brak wartości, co jest prawdą, bo nie podałem wartości współczynnika gamma.

[BŁĘDNE DANE] Wprowadzam do programu ciąg znaków: ./a.out -i b -g -1 -o nazwa -d Wyświetlił się komunikat: "Nie wykonano korekcji gamma, zla wartosc gamma", a więc program dobrze rozpoznaje błędy.

Jak widać w każdym z przypadków program poprawnie zmienia wartości poszczególnych pikseli obrazków według wzoru podanego na stronie kursu. Funkcje obsługujące korekcję gamma dla wszystkich barw działają poprawnie.

3.7 Zmiana poziomów (-z)

Funkcja zmiana poziomów wymaga podanie dwóch argumentów - poziomu czerni, poziomu bieli oraz standardowo informacji o obrazku.

```
1  void zmiana_poziomown (ob *obraz,int czern,int biel)
2  {
3    int i,j;
4    int cz,bie;
5
6    cz=obraz->szarosci*czern/100.0;      /* przedstawienie danych w % */
7    bie=obraz->szarosci*biel/100.0;
8
9    for (i=0; i<obraz->wymy; i++)
10   for (j=0; j<obraz->wymx; j++)
11   {
12     if (obraz->nieb[i][j]<=cz)
13     {
14       /* wartosci ponizej nowej wartosci czerni otrzymuja wartosc 0 */
15       obraz->nieb[i][j]=0;
16     }
17     else if (cz<obraz->nieb[i][j] && obraz->nieb[i][j]<bie)
18     /* wartosci posrednie sa rozciagane na wszystkie poziomy szarosci zawarte w
        obrazie */
19     {
20       obraz->nieb[i][j]=(obraz->nieb[i][j]-cz)*(obraz->szarosci/(bie-cz));
21     }
22     /* wartosci powyzej nowej wartosci bieli otrzymaja wartosc  maksymalna dla
       danego programu */
23     else
24       obraz->nieb[i][j]=obraz->szarosci;
25       /* szarosci rozumiane jako liczba odcieni */
26   }
27 }
```

Listing 13: Zmiana poziomów

Poniższe testy mają na celu sprawdzenie funkcji zmiany poziomów.

[PGM] Wprowadzam do programu ciąg znaków: ./a.out -i a -z 40 60 -o nazwa -d
Obrazek "a" po wykonaniu operacji:

P2
5 5
10
10 0 0 0 10 0 0 10 0 10 0 0 5 0 10 10 0 0 0 10 10 0 5 0 10

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i b -z 40 60 -m g -o nazwa -d
Wykonanie operacji na kolorze zieonym.
Obrazek "b" po wykonaniu operacji:

P3
4 4
15
15 0 1 2 0 0 3 0 5 15 15 15 0 5 0 0 15 7 0 0 0 11 0 5 0 15 0 2 0 4 0 15 7 6 5 0 15 0 15 0 0 0 0 0
15 1 0 3

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i b -z 40 60 -o nazwa -d
Obrazek "b" po wykonaniu operacji:

P3
4 4
15
15 0 0 0 0 0 0 0 0 15 15 15 0 5 0 0 15 5 0 0 0 15 0 0 0 15 0 0 0 0 0 15 5 0 5 0 15 0 15 0 0 0 0 0
15 0 0 0

[BŁĘDNE DANE] Wprowadzam do programu ciąg znaków: ./a.out -i b -z -o nazwa -d
Wyświetlił się komunikat: "Blad. Kod bladu: -3", a więc brak wartości, co jest prawdą, bo nie podałem wartości poziomów czerni ani bieli.

[BŁĘDNE DANE] Wprowadzam do programu ciąg znaków: ./a.out -i b -z 40 -o nazwa -d
Wyświetlił się komunikat: "Blad. Kod bladu: -3", a więc brak wartości, co jest prawdą, bo podałem tylko jedną wartość poziomu.

[BŁĘDNE DANE] Wprowadzam do programu ciąg znaków: ./a.out -i b -z -110 20 -o nazwa -d
Wyświetlił się komunikat:
"Nie wykonano zmiany poziomow, nie spelnasz co najmniej jednego z warunkow:
1. czern i biel naleza do przedzialu (0,100)
2. czern < biel",
a więc program dobrze rozpoznaje błędy.

Jak widać w każdym z przypadków program poprawnie zmienia wartości poszczególnych pikseli obrazków według wzoru podanego na stronie kursu. Funkcje obsługujące zmianę poziomów dla wszystkich barw działają poprawnie.

3.8 Rozmywanie poziome (-rx)

Funkcja rozmywanie poziome nie potrzebuje żadnych dodatkowych argumentów, tylko obrazek.

```
void rozm_poziomec (ob *obraz)
{
  int i,j;
  for (i=1; i<obraz->wymy-1; i++)     /* w tym miejscu '-1' oraz 'i=1' i 'j=1'
    w petli, bo w przeciwnym wypadku przejdziemy poza zakres */
  for (j=1; j<obraz->wymx-1; j++)     /* tablicy (segmentation fault), a 1
    piksel doda tylko ramke o grubosci 1 piksela */
      obraz->czer[i][j]=(obraz->czer[i-1][j]+obraz->czer[i][j]+obraz->czer[i
    +1][j])/3;
}
```

Listing 14: Rozmywanie poziome

Poniższe testy mają na celu sprawdzenie funkcji rozmywanie poziome.

[PGM] Wprowadzam do programu ciąg znaków: ./a.out -i a -rx -o nazwa -d
Obrazek "a" po wykonaniu operacji:

P2
5 5
10
8 1 3 1 9 3 0 5 1 7 1 1 3 1 10 6 3 2 1 7 9 4 5 1 10

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i b -rx -m b -o nazwa -d
Wykonanie operacji na kolorze niebieskim.
Obrazek "b" po wykonaniu operacji:

P3
4 4
15
15 0 1 2 0 0 3 4 5 15 15 15 0 7 0 0 15 3 0 0 4 11 4 5 0 15 0 2 0 2 0 15 8 6 7 0 15 0 15 0 1 0 0 0 15 1 0 3

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i b -rx -o nazwa -d
Obrazek "b" po wykonaniu operacji:

P3
4 4
15
15 0 1 2 0 0 3 4 5 15 15 15 0 7 0 1 5 3 1 6 4 11 4 5 0 15 0 1 2 2 0 7 8 6 7 0 15 0 15 0 1 0 0 0 15 1 0 3

Jak widać w każdym z przypadków program poprawnie zmienia wartości poszczególnych pikseli obrazków według wzoru podanego na stronie kursu (należy pamiętać o dodanej ramce, aby nie było segmentation fault). Funkcje obsługujące rozmywanie poziome dla wszystkich barw działają poprawnie.

3.9 Rozmywanie pionowe (-ry)

Funkcja rozmywanie pionowe nie potrzebuje żadnych dodatkowych argumentów, tylko obrazek.

```
void rozm_pionowen (ob *obraz)
{
  int i,j;
  for (i=0; i<obraz->wymy-1; i++)     /* w tym miejscu '-1' w petli, bo w
    przeciwnym wypadku przejdziemy poza zakres    */
  for (j=0; j<obraz->wymx-1; j++)     /* tablicy (segmentation fault), a 1
    piksel doda tylko ramke o grubosci 1 piksela */
      obraz->nieb[i][j]=(obraz->nieb[i][j-1]+obraz->nieb[i][j]+obraz->nieb[i][j
    +1])/3;
```

Listing 15: Rozmywanie pionowe

Poniższe testy mają na celu sprawdzenie funkcji rozmywanie pionowe.

[PGM] Wprowadzam do programu ciąg znaków: ./a.out -i a -ry -o nazwa -d
Obrazek "a" po wykonaniu operacji:

P2
5 5
10
3 2 2 4 9 1 3 3 3 7 0 2 3 5 10 3 2 1 3 7 9 4 5 1 10

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i b -ry -m r -o nazwa -d
Wykonanie operacji na kolorze czerwonym.
Obrazek "b" po wykonaniu operacji:

P3
4 4
15
5 0 1 3 0 0 7 4 5 15 15 15 0 7 0 0 15 7 3 0 0 11 4 5 0 15 0 0 0 4 2 15 7 6 7 0 15 0 15 0 1 0 0 0 15
1 0 3

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i b -ry -o nazwa -d
Obrazek "b" po wykonaniu operacji:

P3
4 4
15
5 0 0 3 1 1 7 6 7 15 15 15 0 7 2 0 7 3 3 3 2 11 4 5 0 5 1 0 6 4 2 9 3 6 7 0 15 0 15 0 1 0 0 0 15 1 0 3

Jak widać w każdym z przypadków program poprawnie zmienia wartości poszczególnych pikseli obrazków według wzoru podanego na stronie kursu (należy pamiętać o dodanej ramce, aby nie było segmentation fault). Funkcje obsługujące rozmywanie pionowe dla wszystkich barw działają poprawnie.

3.10 Rozciąganie histogramu (-h)

Funkcja rozciąganie histogramu nie potrzebuje żadnych dodatkowych argumentów, tylko obrazek.

```
/* Rozciaganie hitogramu */
void histogramz (ob *obraz)
/* moze byc nie zawsze widoczne, bo ta operacja ma sens gdy nie w pelni jest */
{                       /* wykorzystywany zakres odcieni szarosci obrazka */
  int i,j;
  int min=obraz->ziel[0][0],max=obraz->ziel[0][0];
```

```
  /* inicjuje w ten sposob, aby potem porownywac z kazdym innym elementem */
                  /* w ten sposob unikam bledow wartosci skrajnych */
  for (i=0; i<obraz->wymy; i++)
  for (j=0; j<obraz->wymx; j++)
  {
      if (min>obraz->ziel[i][j])
      /* szukanie wartosci minimalnej i maksymalnej odcieni szarosci w calym
    obrazku */
          min=obraz->ziel[i][j];

      if (max<obraz->ziel[i][j])
          max=obraz->ziel[i][j];
  }
  for (i=0; i<obraz->wymy; i++)
  for (j=0; j<obraz->wymx; j++)
      obraz->ziel[i][j]=(obraz->ziel[i][j]-min)*(obraz->szarosci/(max-min));
}
      /* rozciaganie wartosci wszystkich pikseli (o ile ma to sens) do wartosci
     bardziej  wysrodkowanych; w pelni wykorzystujcych zadeklarowane moliwosci
    obrazka */
```

Listing 16: Rozciąganie histogramu

Poniższe testy mają na celu sprawdzenie funkcji rozciąganie histogramu.

[PGM] Wprowadzam do programu ciąg znaków: ./a.out -i a -h -o nazwa -d
Obrazek "a" po wykonaniu operacji:

P2
5 5
10
8 1 3 1 9 3 0 8 0 7 1 1 5 2 10 6 4 0 1 7 9 4 5 1 10

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i b -h -m g -o nazwa -d
Wykonanie operacji na kolorze zielonym.
Obrazek "b" po wykonaniu operacji:

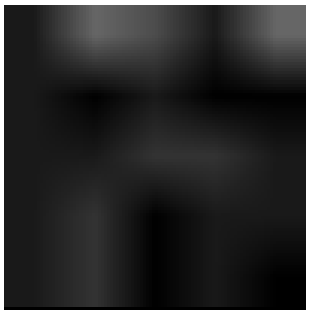
P3
4 4
15
15 0 1 2 0 0 3 4 5 15 15 15 0 7 0 0 15 7 0 0 0 11 4 5 0 15 0 2 0 4 0 15 7 6 7 0 15 0 15 0 1 0 0 0
15 1 0 3

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i b -h -o nazwa -d
Obrazek "b" po wykonaniu operacji:

P3
4 4
15
15 0 1 2 0 0 3 4 5 15 15 15 0 7 0 0 15 7 0 0 0 11 4 5 0 15 0 2 0 4 0 15 7 6 7 0 15 0 15 0 1 0 0 0
15 1 0 3

Jak widoć kompletnie nic się nie zmieniło, co w ty przypadku jest poprawnym działaniem funkcji, gdyż w tych obrazkach w pełni jest wykorzystyany potencjał odcieni barw.

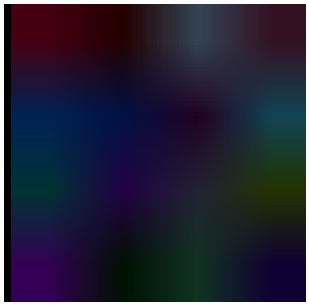
TESTY ROZCIĄGANIA HISTOGRAMU DLA INNYCH OBRAZKÓW



Rys. 3: Obrazek "c" w powiększeniu

Obrazek "c" otwarty w notatniku:

P2
5 5
10
1 4 3 1 4
1 0 1 0 0
1 1 2 2 1
1 2 0 1 1
1 2 0 1 0



Rys. 4: Obrazek "d" w powiększeniu

Obrazek "d" otwarty w notatniku:

P3
4 4
15

4 0 1	2 0 0	3 4 5	3 1 2
0 2 5	0 1 4	2 0 2	1 4 5
0 3 3	2 0 4	2 2 3	2 3 0
3 0 5	0 1 0	1 3 2	1 0 3

[PGM] Wprowadzam do programu ciąg znaków: ./a.out -i c -h -o nazwa -d

Obrazek "c" po wykonaniu operacji:

P2
5 5
10
2 8 6 2 8 2 0 2 0 0 2 2 4 4 2 2 4 0 2 2 2 4 0 2 0

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i d -h -m g -o nazwa -d
Wykonanie operacji na kolorze zielonym.

Obrazek "d" po wykonaniu operacji:

P3
4 4
15
4 0 1 2 0 0 3 4 5 3 1 12 0 7 0 0 8 7 0 0 0 11 4 5 0 3 0 2 0 4 0 2 7 6 7 0 9 0 9 0 1 0 0 0 5 1 0 3

[PPM] Wprowadzam do programu ciąg znaków: ./a.out -i d -h -o nazwa -d

Obrazek "d" po wykonaniu operacji:

P3
4 4
15

4 0 1 2 0 0 3 12 5 3 3 2 0 6 5 0 3 4 2 0 2 1 12 5 0 9 3 2 0 4 2 6 3 2 9 0 3 0 5 0 3 0 1 9 2 1 0 3

Jak widać w każdym z przypadków program poprawnie zmienia wartości poszczególnych pikseli obrazków według wzoru podanego na stronie kursu. Funkcje obsługujące histogram dla wszystkich barw działają poprawnie.

Stwierdzam, że program działa poprawnie dla wszystkich poprawnych danych wejściowych.
FINITO!