プログラミング実習 II レポート課題第 7 回

西田直人

2019 年 1 月 18 日

1　課題 7-1

1.1　source

a.

Listing 1　Makefile1a

```
a7-1: a7-1.o aLoadSave.o aStructs.o
        cc -o a7-1 a7-1.o aLoadSave.o aStructs.o

aLoadSave.o: aLoadSave.c
        cc -c aLoadSave.c

gauss.o: gauss.c
        cc -c gauss.c

aStructs.o: aStructs.c
        cc -c aStructs.c

a7-1.o: a7-1.c headerA.h
        cc -c a7-1.c headerA.h

b7-1.o: b7-1.c headerA.h
        cc -c b7-1.c headerA.h
runA:
        ./a7-1
```

Listing 2　a7-1.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "headerA.h"



int main(){
  ImageData image[8];
  char title[30];


  for(int i=0; i<4; i++){

    switch(i){
    case 0:
      LoadImage("checker4x4_ascii.pgm", &image[i]);
      strcpy(title, "Copied_Asc_gray.pgm");
      break;
    case 1:
```

```
      LoadImage("color4x4_ascii.ppm", &image[i]);
      strcpy(title, "Copied_Asc_color.ppm");
      break;
    case 2:
      LoadImage("lenna256x256binary.ppm", &image[i]);
      strcpy(title, "Copied_Bina_color.ppm");
      break;
    case 3:
      LoadImage("sutehage400x400binary.pgm", &image[i]);
      strcpy(title, "Copied_Bina_gray.pgm");
      break;
    default:
      printf("something is wrong\n");
      return -5;
      break;
    }

    CopyImage(&image[i], &image[i+4]);
    SaveImage(title, &image[i+4]);

    ReleaseImage(&image[i]);
    ReleaseImage(&image[i+4]);
  }

  return 0;
}
```

Listing 3　aLoadSave.c

```
#include <stdio.h>
#include <stdlib.h>
#include "headerA.h"


int LoadImage(const char *filename, ImageData *img){
  FILE *fp;
  char header[32];
  int _w, _h, _dummy;
  int is_binary = 0;
  int is_color = 0;
  int _channels;

  fp = fopen(filename, "rb");

  if (fp == NULL)
    {
      fprintf(stderr, "Error: cannot open file: %s\n", filename);
      return 0;
    }

  fscanf(fp, "%s", header);

  /* ヘッダが P2 アスキー(gray) なのか P5 バイナリ(gray) なのかP3(ASCIIcolor)orP6(BINARYcolor)判定
     */
  if ( ! (header[0] == 'P' && (header[1] == '2' || header[1] == '5' || header[1] == '3'
      || header[1] == '6')) )
    {
      fprintf(stderr, "Error: invalid header: %s\n", header);
```

```
    fclose(fp);
    return 0;
  }

/* バイナリなら 1 を代入 */
is_binary = (header[1] == '5' || header[1] == '6');

//if it is color, 1 is gonna be input into "is_color"
if((is_color = (header[1] == '3' || header[1] == '6'))){
  _channels = 3;
}
else{
  _channels = 1;
}

fscanf(fp, "%d %d\n", &_w, &_h);
fscanf(fp, "%d\n", &_dummy);


//allocate an enough amount of heap memory
if(AllocateImage(img, _w, _h, _channels) == 0){
   printf("Can't allocate specified amount of memory\n");
}


if(is_binary){
    fread(img->data, sizeof(unsigned char), (img->channels)*_w*_h, fp);
}
else{
  int xi, yi;
  if(is_color){ //ascii color case
    for (yi=0; yi<_h; yi++)
      {
        for (xi=0; xi<_w; xi++)
          {
            int r, g, b;
            int idx = 3 * (xi + _w * yi);
            fscanf(fp, "%d %d %d", &r, &g, &b);
            *(img->data+idx) = (unsigned char)r;
            *(img->data+idx+1) = (unsigned char)g;
            *(img->data+idx+2) = (unsigned char)b;
          }
      }
  }
  else{ //ascii gray case
    for (yi=0; yi<_h; yi++)
      {
        for (xi=0; xi<_w; xi++)
          {
            int gray;
            int idx = (xi + _w * yi);
            fscanf(fp, "%d ", &gray);
            *(img->data+idx) = (unsigned char)gray;
          }
      }
  }
}
```

```
    return 1;

}
int SaveImage(const char *filename, ImageData *img){
    FILE *fp;
  int grayorcolor=5;

  fp = fopen(filename, "wb");

  if ( fp == NULL )
    {
      fprintf(stderr, "Error: cannot open file: %s\n", filename);
      return 0;
    }

  //P5(binarygray)orP6(binarycolor)
  if(img->channels == 3){
    grayorcolor = 6;
  }

  /* ヘッダを書き込み */
  fprintf(fp, "P%d\n%d %d\n255\n", grayorcolor, img->width, img->height);

  /* バイナリデータを書き込み */
  fwrite(img->data, sizeof(unsigned char), (img->channels)*(img->width)*(img->height),
      fp);

  fclose(fp);
  fprintf(stderr, "note: ppm image saved: %s\n", filename);
  return 1;
}
```

Listing 4　aStructs.c

```
#include <stdio.h>
#include <stdlib.h>
#include "headerA.h"

int AllocateImage(ImageData *img, int _w, int _h, int _channels){

  if((img->data = (char *)malloc(sizeof(char)*_channels*_w*_h)) == NULL){
    return 0;
  }

  img->width = _w;
  img->height = _h;
  img->channels = _channels;

  return 1;
}

void ReleaseImage(ImageData *img){
  free(img->data);
}

void CopyImage(ImageData *imgIn, ImageData *imgOut){
```

```
  imgOut->data = (char *)malloc(sizeof(char)*(imgIn->channels)*(imgIn->width)*(imgIn->
      height));//error process??

  imgOut->width = imgIn->width;
  imgOut->height = imgIn->height;
  imgOut->channels = imgIn->channels;

  memcpy(imgOut->data, imgIn->data, sizeof(char)*(imgIn->channels)*(imgIn->width)*(imgIn
      ->height));
}
```

Listing 5　headerA.h

```
#ifndef WOWWOW
#define WOWWOW

#include <stdio.h>
#include <stdlib.h>
#include <string.h>


typedef struct ImageData{
  int width;
  int height;
  int channels;
  unsigned char *data;
} ImageData;

int AllocateImage(ImageData *img, int _w, int _h, int _channels);
void ReleaseImage(ImageData *img);
void CopyImage(ImageData *imgIn, ImageData *imgOut);
int LoadImage(const char *filename, ImageData *img);
int SaveImage(const char *filename, ImageData *img);
void GaussianFilter(ImageData *imgIn, ImageData *imgOut);

#endif
```

b.

Listing 6　Makefile1b

```
b7-1: b7-1.o aLoadSave.o aStructs.o gauss.o
        cc -o b7-1 b7-1.o aLoadSave.o aStructs.o gauss.o

aLoadSave.o: aLoadSave.c
        cc -c aLoadSave.c

gauss.o: gauss.c
        cc -c gauss.c

aStructs.o: aStructs.c
        cc -c aStructs.c

a7-1.o: a7-1.c headerA.h
        cc -c a7-1.c headerA.h

b7-1.o: b7-1.c headerA.h
        cc -c b7-1.c headerA.h
```

```
runA:
        ./a7-1
```

Listing 7　b7-1.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "headerA.h"コマンドライン入力の第一引数を入力画像のファイル名、第２引数をフィルター適用回数とする。

//

int main(int argc, char *argv[]){
  //FILE *fp;
  ImageData imgIn, imgOut, imgNow;

  //printf("B!");

  if(argc != 3){
    printf("wrong command,\n ./b7-1 {filename} {the number of filters}\n");
    return -1;
  }
  /*
  if((fp = fopen(argv[1], "rb")) == NULL){
    printf("there aint such a fxxking file!\n");
    return -2;
    }*/

  if(atoi(argv[2]) == 0){
    printf("don't you wannna filter your image?\n");
    return -3;
  }

  LoadImage(argv[1], &imgIn);
  CopyImage(&imgIn, &imgOut); //for copying imgIn's channels, width, height.and to
      secure the memory of imgOut.data

  //printf("%d", atoi(argv[2]));
  //("A!");

  for(int i=0; i<atoi(argv[2]); i++){
    GaussianFilter(&imgIn, &imgOut);
    CopyImage(&imgOut, &imgIn);
    //printf("%d", i);
  }

  if(imgIn.channels == 1){
    SaveImage("FilteredImageGray.pgm", &imgOut);
  }
  else{
    SaveImage("FilteredImageColor.ppm", &imgOut);
  }

  ReleaseImage(&imgIn);
  ReleaseImage(&imgOut);

  return 0;
}
```

Listing 8　gauss.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "headerA.h"
//は続いて読んじゃうからブレイクする必要あるけど文はどうなんだ？きちんと背反で条件をわけないとダメ？switchif
void GaussianFilter(ImageData *imgIn, ImageData *imgOut){
  int xi, yi;
  int _w = imgIn->width;
  int _h = imgIn->height;
  int _channels = imgIn->channels;
  int is_color = (imgIn->channels == 3); //color:1 gray:0
  unsigned char img[(imgIn->width)*(imgIn->height)][3];
  unsigned char img2[(imgIn->width)*(imgIn->height)];
  int upper=-_w, lower=+_w, right=+1, left=-1;



  if(is_color){
    memcpy(img, imgIn->data, _channels*_h*_w);
    for(yi=0; yi< _h; yi++ ){
      for(xi=0; xi< _w; xi++){
        int i = xi + yi*_w;


        if(xi == 0) left = 0;
        if(xi == _w-1) right = 0;
        if(yi == 0) upper = 0;
        if(yi == _h-1) lower =0;

        *(imgOut->data + i*3) = (unsigned char)((2*((int)img[i+left][0]+(int)img[i+right
            ][0]+(int)img[i+upper][0]+(int)img[i+lower][0])+4*((int)img[i][0])+((int)img
            [i+upper+left][0]+(int)img[i+upper+right][0]+(int)img[i+lower+left][0]+(int)
            img[i+lower+right][0]))/16);
        *(imgOut->data + i*3+1) = (unsigned char)((2*((int)img[i+left][1]+(int)img[i+
            right][1]+(int)img[i+upper][1]+(int)img[i+lower][1])+4*((int)img[i][1])+((
            int)img[i+upper+left][1]+(int)img[i+upper+right][1]+(int)img[i+lower+left
            ][1]+(int)img[i+lower+right][1]))/16);
        *(imgOut->data + i*3+2) = (unsigned char)((2*((int)img[i+left][2]+(int)img[i+
            right][2]+(int)img[i+upper][2]+(int)img[i+lower][2])+4*((int)img[i][2])+((
            int)img[i+upper+left][2]+(int)img[i+upper+right][2]+(int)img[i+lower+left
            ][2]+(int)img[i+lower+right][2]))/16);

  upper=-_w;
  lower=+_w;
  right=+1;
  left=-1;
//printf("%d  ", i);
      }
    }
  }
  else{
    memcpy(img2, imgIn->data, _channels*_h*_w);
    for(yi=0; yi< _h; yi++ ){
      for(xi=0; xi< _w; xi++){
```

```
        int i = xi + yi*_w;

        if(xi == 0) left = 0;
        if(xi == _w-1) right = 0;
        if(yi == 0) upper = 0;
        if(yi == _h-1) lower =0;

        *(imgOut->data + i) = (unsigned char)((2*((int)img2[i+left]+(int)img2[i+right]+(
            int)img2[i+upper]+(int)img2[i+lower])+4*((int)img2[i])+((int)img2[i+upper+
            left]+(int)img2[i+upper+right]+(int)img2[i+lower+left]+(int)img2[i+lower+
            right]))/16);
//printf("%d  ", i);
  upper=-_w;
  lower=+_w;
  right=+1;
  left=-1;
      }
    }
  }
}
/*      *(imgOut->data + i*3) = (unsigned char)((2*((int)img[i-1][0]+(int)img[i+1][0]+(
    int)img[i-_w][0]+(int)img[i+_w][0])+4*((int)img[i][0])+((int)img[i-_w-1][0]+(int)img
    [i-_w+1][0]+(int)img[i+_w-1][0]+(int)img[i+_w+1][0]))/16);
        *(imgOut->data + i*3+1) = (unsigned char)((2*((int)img[i-1][1]+(int)img[i
            +1][1]+(int)img[i-_w][1]+(int)img[i+_w][1])+4*((int)img[i][1])+((int)img[i-
            _w-1][1]+(int)img[i-_w+1][1]+(int)img[i+_w-1][1]+(int)img[i+_w+1][1]))/16);
        *(imgOut->data + i*3+2) = (unsigned char)((2*((int)img[i-1][2]+(int)img[i
            +1][2]+(int)img[i-_w][2]+(int)img[i+_w][2])+4*((int)img[i][2])+((int)img[i-
            _w-1][2]+(int)img[i-_w+1][2]+(int)img[i+_w-1][2]+(int)img[i+_w+1][2]))/16);
*/
```

1.2 result

7-1a.

7-1b.

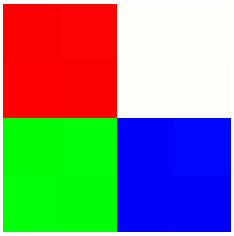

図 1　入力画像：ASCII カラー画像

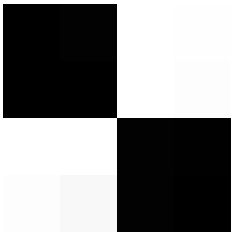

図 2　入力画像：ASCII グレースケール画像

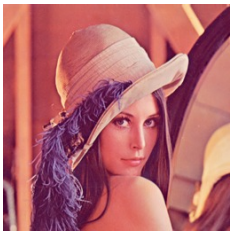

図 3　入力画像：Binary カラー画像

図 4　入力画像：Binary グレースケール画像

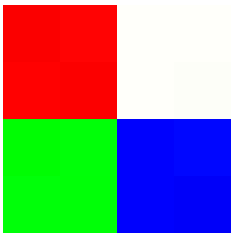

図 5　出力画像：ASCII カラー画像

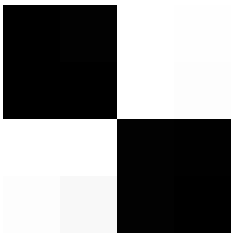

図 6　出力画像：ASCII グレースケール画像

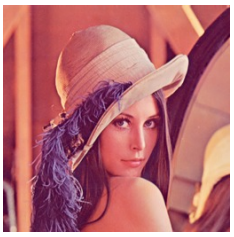

図 7　出力画像：Binary カラー画像

図 8　出力画像：Binary グレースケール画像

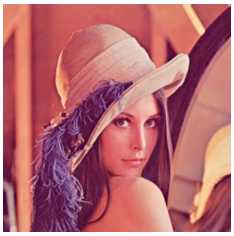

図 9　入力画像：カラー画像

図 10　入力画像：グレースケール画像

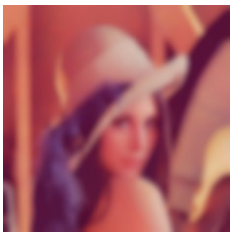

図 11　出力画像：カラー画像 30 回フィルター

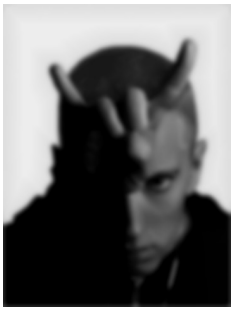

図 12　出力画像：グレースケール画像 30 回フィルター

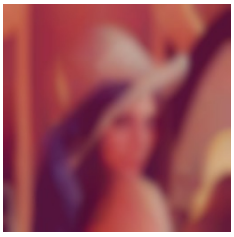

図 13　出力画像：カラー画像 70 回フィルター

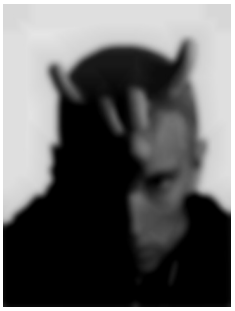

図 14　出力画像：グレースケール画像 60 回フィルター

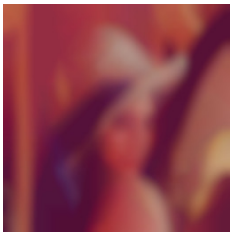

図 15　出力画像：カラー画像 100 回フィルター

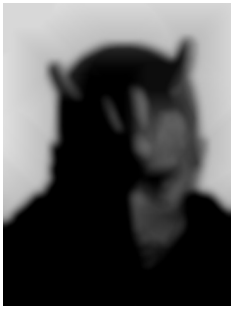

図 16　出力画像：グレースケール画像 100 回フィルター

2　課題 7-2

a.

2.1　source

Listing 9　Makefile2a

```
a7-2: a7-2.o matrix.o vector.o
        cc -o a7-2 a7-2.o matrix.o vector.o
.c.o:   # サフィックスルール。拡張子 .c から拡張子 .o のファイルを作るためのルール。
        cc -c $<
a7-2.o: a7-2.c headerB.h
        cc -c a7-2.c headerB.h
runA: a7-2
        ./a7-2
```

Listing 10　a7-2.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "headerB.h"

int main(){
  Matrix mat, mat2;
  Vector vec, vec2;
  int _rows, _cols, _n;

  printf("how's the scale of the matrix?: ");
  scanf("%d", &_rows);
  scanf("%d", &_cols);

  mat.rows = _rows;
  mat.cols = _cols;


  AllocateMatrix(&mat, _rows, _cols);

  printf("input the matrix's elements : ");
  for(int i=0; i<(_rows)*(_cols); i++){
    scanf("%lf", (mat.elems + i));
  }

  CopyMatrix(&mat, &mat2);
  printf("\nmatrix A: \n");
  FprintMatrix(stdout, &mat);
//  FprintMatrix(stdout, &mat2);



  printf("\n\nhow's the scale of the vector?: ");
  scanf("%d", &_n);
```

```
  AllocateVector(&vec, _n);

  printf("input the vector's elements : \n");
  for(int i=0; i<_n; i++){
    scanf("%lf", (vec.elems +i));
  }
//printf("aa");
  CopyVector(&vec, &vec2);
//printf("ii");
//printf("######%d######", _n);
//printf("%d %d:" vec.n, vec2.n);
printf("\nvector B: \n");
  FprintVector(stdout, &vec);
//  FprintVector(stdout, &vec2);
//printf("uu");
  MultMatrixVector(&mat, &vec, &vec2);

  printf("\n\nAxB :\n");
  FprintVector(stdout, &vec2);
  printf("\n");

  ReleaseMatrix(&mat);
  ReleaseMatrix(&mat2);
  ReleaseVector(&vec);
  ReleaseVector(&vec2);

  return 0;
}
```

Listing 11　headerB.h

```
#ifndef WOWWOW
#define WOWWOW

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct Matrix{
  int rows;
  int cols;
  double *elems;
} Matrix;

typedef struct Vector{
  int n;
  double *elems;
} Vector;


int AllocateMatrix(Matrix *mat, int rows, int cols);
void ReleaseMatrix(Matrix *mat);
void CopyMatrix(Matrix *inMat, Matrix *outMat);
void FprintMatrix(FILE *fp, Matrix *mat);

int AllocateVector(Vector *vec, int n);
void ReleaseVector(Vector *vec);
void CopyVector(Vector *inVec, Vector *outVec);
```

```
void FprintVector(FILE *fp, Vector *vec);
void MultMatrixVector(Matrix *A, Vector *V, Vector *Out);

int LoadLinearSystem(const char *filename, Matrix *A, Vector *b);
int SolveGaussJordan(Matrix *A, Vector *b);

#endif
```

Listing 12　matrix.c

```
#include "headerB.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


int AllocateMatrix(Matrix *mat, int rows, int cols){
  if((mat->elems = (double *)malloc(sizeof(double)*rows*cols + 1)) == NULL){
    printf("can't secure the memory.\n");
    return 0;
  }

  mat->rows = rows;
  mat->cols = cols;

  return 1;
}

void ReleaseMatrix(Matrix *mat){
  free(mat->elems);
}

void CopyMatrix(Matrix *inMat, Matrix *outMat){

  if((outMat->elems = (double *)malloc(sizeof(double)*(inMat->rows)*(inMat->cols))) ==
      NULL){
    printf("can't secure the memory.\n");
    exit(1);
  }

  outMat->rows = inMat->rows;
  outMat->cols = inMat->cols;

  memcpy(outMat->elems, inMat->elems, sizeof(double)*(inMat->rows)*(inMat->cols));
}


void FprintMatrix(FILE *fp, Matrix *mat){
  int _rows = mat->rows;
  int _cols = mat->cols;

  for(int i=0; i<(_rows); i++){
    for(int j=0; j<(_cols); j++){
      fprintf(fp, "%lf ", *(mat->elems+j+i*(_cols)));
    }
    fprintf(fp, "\n");
  }
```

```
}
```

Listing 13　vector.c

```
#include "headerB.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


int AllocateVector(Vector *vec, int n){
  //printf("!!!%d!1!", n);
  if((vec->elems = (double *)malloc(sizeof(double)*(n))) == NULL){
    printf("can't secure the memory.\n");
    return 0;
  }

  vec->n = n;

  return 1;
}


void ReleaseVector(Vector *vec){
  free(vec->elems);
}

void CopyVector(Vector *inVec, Vector *outVec){
  int i;

  if((outVec->elems = (double *)malloc(sizeof(double)*(inVec->n))) == NULL){
    printf("can't secure the memory.\n");
    exit(1);
  }

  outVec->n = inVec->n;

  //printf("##%d %lf## ", sizeof(inVec->n), *(inVec->elems));

  //memcpy(outVec->elems, inVec->elems, sizeof(double)*(inVec->n));これでもできる！
  //

  for(i=0; i<(inVec->n); i++){
    *(outVec->elems+i) = *(inVec->elems+i);
  }

}


void FprintVector(FILE *fp, Vector *vec){

  for(int i=0; i<(vec->n); i++){
    fprintf(fp, "%lf ", *(vec->elems + i));
  }
}


void MultMatrixVector(Matrix *A, Vector *V, Vector *Out){
```

```
  int _rows = A->rows;
  int _cols = A->cols;
  int i, j;

  if(_cols != V->n){
    printf("Can't multiply them!\n");
    exit(1);
  }

  //initialize
  for(i=0; i<(V->n); i++){
    *(Out->elems + i) = 0;
  }

  // printf("OUCH!!");

  /*
  for(int i=0; i<_cols ; i++){
    for(int j=0; j<_rows; j++){
      *(Out->elems +i) += (*(A->elems) +j + i*(_rows))*(*(V->elems)+j);
    }
    }*/

  for(i=0; i<_rows; i++){
    for(j=0; j<_cols; j++){
      *(Out->elems + i) += (*(A->elems+i*(_rows)+ j))*(*(V->elems + j));
    }
  }
}
```

2.2 result

```
  s1811433@LC2RR-P010:~/prog2/07/07kadai$ make -f Makefile2a
  cc -o a7-2 a7-2.o matrix.o vector.o
  s1811433@LC2RR-P010:~/prog2/07/07kadai$ ./a7-2
  how's the scale of the matrix?: 2 2
  input the matrix's elements : 1 2 3 4

  matrix A:
  1.000000 2.000000
  3.000000 4.000000


  how's the scale of the vector?: 2
  input the vector's elements :
  1 2

  vector B:
  1.000000 2.000000

  AxB :
  5.000000 11.000000
  s1811433@LC2RR-P010:~/prog2/07/07kadai$ ./a7-2
  how's the scale of the matrix?: 2 3
  input the matrix's elements : 1 2 3 4 5 6
```

```
matrix A:
1.000000 2.000000 3.000000
4.000000 5.000000 6.000000


how's the scale of the vector?: 2
input the vector's elements :
1 2

vector B:
1.000000 2.000000 Can't multiply them!
s1811433@LC2RR-P010:~/prog2/07/07kadai$ ./a7-2
how's the scale of the matrix?: 2 3
input the matrix's elements : 1 2 3 4 5 6

matrix A:
1.000000 2.000000 3.000000
4.000000 5.000000 6.000000


how's the scale of the vector?: 3
input the vector's elements :
1 2 3

vector B:
1.000000 2.000000 3.000000

AxB :
14.000000 26.000000 0.000000
s1811433@LC2RR-P010:~/prog2/07/07kadai$
```

b.

2.3 source

Listing 14　b7-2.c

```
#include <stdio.h>
#include <stdlib.h>
#include "headerB.h"

int LoadLinearSystem(const char *filename, Matrix *A, Vector *b);


int main(int argc, char *argv[]){
  Matrix A, A2;
  Vector b, b2, x;
  int judge, i;
  double err = 0;


  if(argc != 2){
    printf("wrong command,\n ./b7-2 {filename}\n");
    return 0;
  }
```

```
  if((judge = LoadLinearSystem(argv[1], &A, &b)) == 0){
    printfどんまい("\n");
    return 0;
  }

  printf("\nMatrix A(%d, %d)\n", A.rows, A.cols);
  FprintMatrix(stdout, &A);
  printf("\nVector b (%d)\n", b.n);
  FprintVector(stdout, &b);
  CopyVector(&b, &b2); //b2 is the copy of the original constant vector b
  CopyMatrix(&A, &A2); //A2 is the copy of the original coefficient matrix A

  if(SolveGaussJordan(&A, &b) == 1){
    printf("\n\nAnswer: x(%d)\n", b.n);
    for(i=0; i<(b.n); i++){
      printf("%lf ", *(b.elems + i));
      //printf("AAA!");
    } //b is now the answer vector(equals to x)
    printf("\n");
  }
  else{
    printf("SOMETHING IS WRONG!!\n");
  }
  //printf("BB!");


  CopyVector(&b, &x); // x is literally the answer vector


  MultMatrixVector(&A2, &x, &b); //now b is Ax
  //printf("CC!");

  double c[b.n];

  for(i=0; i<(b.n); i++){
    c[i] = (*(b.elems + i) - *(b2.elems + i));
    err += (c[i])*(c[i]);
  }
  /*ここで以下のようにしたらコアダンプ出る。ポインタでなぜできないのか質問
.初期化をしてないからでした本当にありがとうございました。
->

  double *c;

->double d;
->*c = &d; ここがいるねん。ゴミアドレスを参照しようとするとバグる。


  for(i=0; i<(b.n); i++){
   *(c+i) = (*(b.elems + i) - *(b2.elems + i));
    err += (*(c+i))*(*(c+i));
  }
  */

  printf("\nError: %lf\n", err);
```

```
  return 1;
}
```

Listing 15　linear.c

```
#include <stdio.h>
#include <stdlib.h>
#include "headerB.h"


int LoadLinearSystem(const char *filename, Matrix *A, Vector *b){
  FILE *fp;
  int _rows, _cols;
  int i, j;
  int gomi; //for fscanf("\n");

  if ((fp = fopen(filename, "r")) == NULL) {
    printf("file open error!!\n");
    return 0;
  }


  fscanf(fp, "%d %d\n", &_rows, &_cols);
  printf("LoadLinerSystem: matrix size: %dx%d\n\n", _rows, _cols);

  AllocateMatrix(A, _rows, _cols);
  AllocateVector(b, _cols);


  for(i=0; i<_rows; i++){
    for(j=0; j<_cols; j++){
      fscanf(fp, "%lf ", (A->elems+i*_cols+j));

    }
  }
  for(j=0; j<_cols; j++){
    fscanf(fp, "%lf ", (b->elems+j));
  }

  return 1;

}

int SolveGaussJordan(Matrix *A, Vector *b){
  int _rows = A->rows;
  int _cols = A->cols;
  int i, j, k;
  double a, c;//は割ったり引き算する元となる行ベクトルの行の位置（基準点）i
  //は列の位置、は割ったり引き算をする先の行ベクトルの行の位置、はかけてひく際のかける行列の要素。はの対角成分で除算する
      際の割る数jkacA

  for(i=0; i<_rows; i++){
    c = *(A->elems+i*_cols+i);
    // deviding the i-th row's components
    for(j=i; j<_cols;j++){
```

```
      *(A->elems+i*_cols+j) /= c ;
    }
    *(b->elems + i) /= c;

    // multipy and substraction

    for(k = 0; k < _rows; k++){
      if(k != i){
        a = *(A->elems + k*_cols + i);
        for(j=0; j<_cols; j++){
          *(A->elems+ k*_cols + j) -= (a)*(*(A->elems + i*_cols + j));
        }
        *(b->elems + k) -= (*(b->elems + i))*(a);
      }
    }

    if(i<_rows-1){
      //to judge whether the row or col are 0
      if(*(A->elems + (i+1)*_cols + (i+1)) == 0){
        printf("\ndiagonal matrix is zero...\n");
        return 0;
      }
    }
  }

  return 1;
}
```

2.4 result

```
s1811433@LC2RR-P010:~/prog2/07/07kadai$ make -f Makefile2b -B
cc -c b7-2.c headerB.h
cc -c matrix.c
cc -c vector.c
cc -c linear.c headerB.h
cc -o b7-2 b7-2.o matrix.o vector.o linear.o
s1811433@LC2RR-P010:~/prog2/07/07kadai$ ./b7-2 data4x4.txt
LoadLinerSystem: matrix size: 4x4


Matrix A(4, 4)
-7.000000 1.000000 -10.000000 -6.000000
7.000000 5.000000 -9.000000 2.000000
8.000000 5.000000 5.000000 3.000000
-9.000000 8.000000 -7.000000 1.000000

Vector b (4)
2.000000 9.000000 8.000000 8.000000

Answer: x(4)
0.333525 1.363479 -0.067972 -0.381912

Error: 0.000000
s1811433@LC2RR-P010:~/prog2/07/07kadai$ ./b7-2 data10x10.txt
LoadLinerSystem: matrix size: 10x10
```

```
Matrix A(10, 10)
-2.000000 2.000000 -7.000000 -7.000000 -7.000000 -1.000000 9.000000 9.000000 -5.000000
     -9.000000
-3.000000 -6.000000 6.000000 6.000000 2.000000 -4.000000 6.000000 -8.000000 6.000000
    4.000000
2.000000 9.000000 -8.000000 5.000000 3.000000 7.000000 -3.000000 1.000000 -2.000000
    4.000000
-9.000000 2.000000 2.000000 -2.000000 7.000000 -10.000000 1.000000 -1.000000 7.000000
    4.000000
1.000000 -4.000000 9.000000 2.000000 9.000000 -10.000000 7.000000 3.000000 -5.000000
    4.000000
-8.000000 -5.000000 -3.000000 4.000000 3.000000 5.000000 0.000000 6.000000 7.000000
    7.000000
6.000000 9.000000 2.000000 -2.000000 1.000000 9.000000 1.000000 6.000000 -9.000000
    -5.000000
2.000000 6.000000 8.000000 6.000000 1.000000 7.000000 6.000000 9.000000 -3.000000
    1.000000
-8.000000 2.000000 1.000000 -8.000000 -3.000000 -8.000000 -6.000000 4.000000 9.000000
    -3.000000
-6.000000 5.000000 -8.000000 -4.000000 7.000000 -3.000000 5.000000 9.000000 5.000000
    1.000000

Vector b (10)
9.000000 -4.000000 -9.000000 9.000000 2.000000 -10.000000 -2.000000 -4.000000
    -2.000000 -7.000000

Answer: x(10)
2.385402 2.742281 2.022468 -12.917047 -7.374715 1.773511 5.067214 -2.236396 -1.366858
    16.681884

Error: 0.000000
s1811433@LC2RR-P010:~/prog2/07/07kadai$
```