機械ソフトウェア演習　宿題

第9回

学籍番号 : 340192C

井上　裕太

[感想]

宿題

宿題1

[shuku1.c]

———mallocを使うパターン------

#include<stdio.h>

#include<stdlib.h>

double func(int n);

int \*elements;

int main(){

int n;

double ave;

printf("number of input elements:");

scanf("%d",&n);

if(NULL==(elements=(int\*)malloc(sizeof(int)\*n))){

printf("領域確保に失敗しました\n");

}

ave = func(n);

printf("average of the elements : %lf\n",ave);

}

double func(int n){

int i,sum,temp;

sum=0;

for(i=0;i<n;i++){

scanf("%d",&(elements[i]));

}

for(i=0;i<n;i++){

sum+=elements[i];

}

return (double)sum/n;

}

———mallocを使わないパターン---

#include<stdio.h>

#include<stdlib.h>

double func(int n);

int main(){

int n;

double ave;

printf("number of input elements:");

scanf("%d",&n);

ave = func(n);

printf("average of the elements : %lf\n",ave);

}

double func(int n){

int i,sum,temp;

sum=0;

for(i=0;i<n;i++){

scanf("%d",&temp);

sum+=temp;

}

return (double)sum/n;

}

[command]

gcc shuku1.c -o shuku1 -lm

./shuku1

[input]

number of input elements:5

12

14

512

2134

21

[output]

average of the elements : 538.600000

[考察]

配列ということでmallocを利用するコードも書いたが、今回の問題に関して言えば一時保存するものを足して合計を計算していくというだけなので使わないほうが簡単である

しかし、例えば要素を区間で区切って足していく際に最大となる合計値はといった問題になった場合には動的にサイズを変えられるmallocは非常に重宝すると考える

宿題2

[shuku2.c]

// ip\_sample.c

//

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <stdbool.h>

#define X\_SIZE (1152/2)

#define Y\_SIZE (864/2)

char decide\_color(int x,int y);

void sample\_func(unsigned char\* image, int sx, int sy);

int main(int argc, char\* argv[])

{

unsigned char\* image;

int i, j;

int size\_x=X\_SIZE, size\_y=Y\_SIZE;

FILE \*fp, \*fp\_w;

unsigned char buf;

//Memory allocation

if (NULL == (image = (unsigned char \*)malloc(sizeof(unsigned char)\*size\_x\*size\_y))){

printf("Memory allocation failed. ");return (-1);

}

//Image initialize

for(j=0; j<size\_y; j++)

for(i=0; i<size\_x; i++)

image[j\*size\_x+i] = 0;

// Image read

if (NULL == (fp = fopen("test-gray.raw.data","rb"))){

printf("File test-gray.raw not found.");

fclose(fp);

free(image);

return(-1);

}

if (NULL == (fp\_w = fopen("test-gray\_out.raw.data","wb+"))){

printf("Couldn't write File test-gray\_out.");

fclose(fp);

fclose(fp\_w);

free(image);

return(-1);

}

for(j=0; j<size\_y; j++)

for(i=0; i<size\_x; i++){

fscanf(fp,"%c", &buf);

image[j\*size\_x+i] = buf;

}

sample\_func(image, size\_x, size\_y);

for(j=0; j<size\_y; j++)

for(i=0; i<size\_x; i++){

buf = image[j\*size\_x+i];

fprintf(fp\_w,"%c",buf);

}

printf("Program finished.\n");

fclose(fp);

fclose(fp\_w);

free(image);

return 0;

}

———(1)———

void sample\_func(unsigned char\* image, int sx, int sy)

{

// SAMPLE FUNCTION ２値化処理

int i,j, th;

double theta;

unsigned char buf;

unsigned char\* out\_image;

bool processed\_flag[sx][sy];

int dx[] = {-1,0,1};

printf("Input theta : ");

scanf("%d",&th);

if (NULL == (out\_image = (unsigned char \*)malloc(sizeof(unsigned char)\*sx\*sy))){

}

theta = M\_PI\*th/180.0;

//Image initialize

for(j=0; j<sy; j++)

for(i=0; i<sx; i++){

out\_image[j\*sx+i] = 0;

processed\_flag[i][j] = false;

}

for(j=0; j<sy; j++){

for(i=0; i<sx; i++){

int x = cos(theta)\*(i-sx/2)-sin(theta)\*(j-sy/2)+sx/2;

int y = sin(theta)\*(i-sx/2)+cos(theta)\*(j-sy/2)+sy/2;

int point = (int)(y\*sx+x);

if(y<sy&&x<sx&&y>0&&x>0){

out\_image[point] = image[j\*sx+i];

processed\_flag[x][y] = true;

}

}

}

int h;

for(h=0;h<100;h++)

for(j=0;j<sy;j++)

for(i=0;i<sx;i++){

if(!processed\_flag[i][j]){

int total=0,k,l,num=0;

for(k=0;k<3;k++)

for(l=0;l<3;l++){

if(!((k==0)&&(l==0))&&((j+dx[l])<sy&&(i+dx[k])<sx)){

total+=out\_image[(j+dx[l])\*sx+(i+dx[k])];

num++;

}

}

out\_image[j\*sx+i] = total/num;

}

}

for(j=0;j<sy;j++)

for(i=0;i<sx;i++)

image[j\*sx+i]=out\_image[j\*sx+i];

return;

}

————(2)————

void sample\_func(unsigned char\* image, int sx, int sy)

{

// SAMPLE FUNCTION ２値化処理

int i,j, th;

double theta;

unsigned char buf;

unsigned char\* out\_image;

bool processed\_flag[sx][sy];

int dx[] = {-1,0,1};

printf("Input theta : ");

scanf("%d",&th);

if (NULL == (out\_image = (unsigned char \*)malloc(sizeof(unsigned char)\*sx\*sy))){

}

theta = M\_PI\*th/180.0;

//Image initialize

for(j=0; j<sy; j++)

for(i=0; i<sx; i++){

out\_image[j\*sx+i] = 0;

processed\_flag[i][j] = false;

}

for(j=0; j<sy; j++){

for(i=0; i<sx; i++){

int x = cos(-theta)\*(i-sx/2)-sin(-theta)\*(j-sy/2)+sx/2;

int y = sin(-theta)\*(i-sx/2)+cos(-theta)\*(j-sy/2)+sy/2;

int point = (int)(y\*sx+x);

if(y<sy&&x<sx&&y>0&&x>0){

out\_image[j\*sx+i] = image[point];

}

}

}

for(j=0;j<sy;j++)

for(i=0;i<sx;i++)

image[j\*sx+i]=out\_image[j\*sx+i];

return;

}

[command]

gcc shuku2.c -o shuku2 -lm

./shuku2

[input]

(1)

Input theta : 40

Program finished.

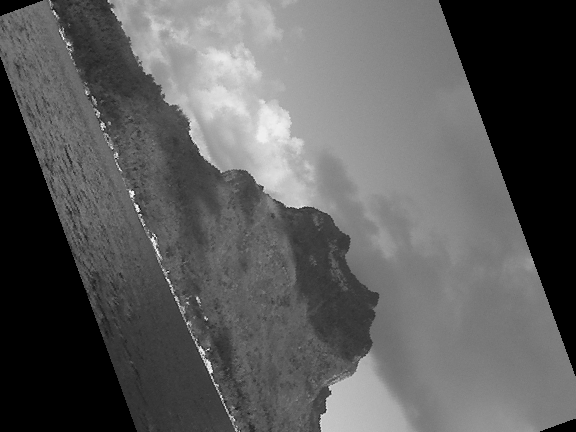
(2)

Input theta : 70

Program finished.

[output]





[考察]

(2)のほうが変換後のセルの空きが存在しないことが保証されているため簡単なコードになった

(1)については、スムージングの考え方を利用した

あらかじめそれぞれのセルに対してフラグを設定しておき、変換後に書き込まれていないセルについては自分の周り8セルの平均を代入する形式にした

これは空きセルが固まっているところに対してはある程度の回数作用させない限り、綺麗にスムージングされないが回数を経るにつれ周りの色との整合性が取れるようになってきたと考える

個人的には、for-for-if…のようにコードがネストしてしまっているのは非常に良くないため、もっと良いアルゴリズムにも挑戦してみたい

注). 更新できれば更新したいです