**Diseño e implementación:**

Antes que nada, cabe informar que se utilizó una estructura llamada “complejo” para poder representarlos. La misma consiste de dos atributos de tipo float los cuales representan a la parte real e imaginaria del numero.

El programa consiste primeramente en encontrar cada argumento indicado (pasado como parámetro al programa por línea de comandos), analizar si éstos son correctos, guardarlos en sus respectivas variables y, en caso de no ser correctos, informar el error y cortar la ejecución del programa. Todos los parámetros fueron guardados en variables estáticas, y en caso de no estar presentes se reemplazaron por aquellos dados en el enunciado por defecto. En caso de no poder crear un archivo de salida, también se sale del programa.

Se escribió un programa secuencial completamente, sin el agregado de muchas estructuras adicionales ni funciones auxiliares, para que luego el seguimiento en lenguaje ensamblador sea más simple y entendible.

El prgrama comienza con un ciclo switch en el que se almacenan todos los parámetros pasados al programa mediante línea de comandos, a través de la función getopt de la librería unistd. En caso de error, se sale del programa con EXIT\_FAILURE. Para la verificación de la validez de los mismos se utilizaron distintas funciones auxiliares, como atoi para la conversión de string a entero, o sscanf para los parámetros que combinaban decimales y otros símbolos (como los complejos). En caso de error, se imprimen mensajes de error y se sale del programa con EXIT\_FAILURE.

La parte central del algoritmo, donde se itera en el plano complejo de forma discreta, se utilizan dos ciclos definidos for para el eje real y el eje imaginario, manteniendo un acumulador para definir el brillo del pixel, el cual luego es guardado en el archivo de salida. Para enontrar el paso de avance sobre los ejes del plano, se utilizó el cociente entre el alto o ancho y la resolución de la imagen, para el eje imaginario y real, respectivamente. La operación matemática que se realiza en el interior de los ciclos se implementó de la forma más eficiente posible encontrada, generando una única línea de código para la parte real e imaginaria, y usando una única variable temporal.

**Como obtener el ejecutable y correr pruebas:**

Para mayor comodidad se dispone de un archivo llamado “script.sh” que al ejecutarlo (esto es, posicionándose sobre el directorio donde se encuentran los archivos y ejecutando el comando ./script.sh) automáticamente compilara el programa y ejecutará una seria de comandos de prueba y sus respectivas salidas.

**El código fuente:**

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

#include <ctype.h>

#include <math.h>

#include <unistd.h>

#include <getopt.h>

#define N 255

typedef struct complejo{

float parteReal;

float parteImaginaria;

}complejo;

void print\_usage() {

printf("Error de parametros ./tp0 -h para ayuda detallada.\nUso: ./tp0 -r 640x480 -c 0+0i -C 0.285-0,01i -w 4 -H 4 -o /home/user/julia.pgm (or -o - for stdout)\n");

}

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

char\* pathToExe = argv[0];

int option = -1;

int encontroRes = -1;

int encontroAlto = -1;

int encontroAncho = -1;

int encontroConst = -1;

int encontroOutput = -1;

int encontroOrigen = -1;

int salidaConsola = -1;

char\* ptrOrigen;

char\* constante;

char\* output;

char\* res;

char controli;

complejo constanteC;

complejo origen;

complejo pixel;

int resolucionHorizontal, resolucionVertical;

float anchoRectangulo;

float altoRectangulo;

while ((option = getopt(argc, argv,"r:c:C:W:H:o:h")) != -1) {

switch (option) {

case 'r' :

encontroRes = 1;

res = optarg;

break;

case 'c' :

encontroOrigen = 1;

ptrOrigen = optarg;

break;

case 'C' :

encontroConst = 1;

constante = optarg;

break;

case 'W' :

anchoRectangulo = atoi(optarg);

if (anchoRectangulo <= 0){

printf("Ancho plano complejo invalido.\n");

exit(EXIT\_FAILURE);

}

encontroAncho = 1;

break;

case 'H' :

altoRectangulo = atoi(optarg);

if (altoRectangulo <= 0){

printf("Alto plano complejo invalido.\n");

exit(EXIT\_FAILURE);

}

encontroAlto = 1;

break;

case 'o' :

encontroOutput = 1;

if (strcmp(optarg,"-") == 0) salidaConsola = 1;

output = optarg;

/\*

if ((isalpha((char)optarg[0]) != 1) && (isdigit((char)optarg[0]) != 1)){

if (strcmp(optarg, "-") == 0) salidaConsola = 1;

else {

printf("fatal: cannot open output file.\n");

exit(0);

}

}

output = optarg\*/

break;

case 'h' :

printf("AYUDA: Resolucion: -r 640x480\n Origen: -c 0+0i\n Constante imaginaria: -C 0.285-0,01i\n Ancho del plano complejo: -W 4\n Alto del plano complejo: -H 4\n Formato de salida: -o /home/julia.pgm (PARAMETRO IMPRESCINDIBLE) (or -o - for stdout)\n");

return 0;

default:

print\_usage();

exit(EXIT\_FAILURE);

}

}

if (encontroOutput != 1){ ///UNICA CONDICION NECESARIA PARA CORRER EL PROGRAMA

printf("El argumento -o es condición necesaria\n");

exit(EXIT\_FAILURE);

}

if (encontroAlto != 1) altoRectangulo = 4;

if (encontroAncho != 1) anchoRectangulo = 4;

int scanResult;

if (encontroConst == 1){

scanResult = sscanf(constante, "%f%f%c", &constanteC.parteReal, &constanteC.parteImaginaria, &controli);

if (scanResult < 3){

printf("Error obteniendo constante compleja.\n");

print\_usage();

exit(EXIT\_FAILURE);

}

} else {

constanteC.parteReal = 0.285;

constanteC.parteImaginaria = -0.01;

}

if (encontroOrigen == 1){

scanResult = sscanf(ptrOrigen, "%f%f%c", &origen.parteReal, &origen.parteImaginaria, &controli);

if (scanResult < 3){

printf("Error obteniendo origen complejo.\n");

print\_usage();

exit(EXIT\_FAILURE);

}

} else {

origen.parteReal = 0;

origen.parteImaginaria = 0;

}

if (encontroRes == 1){

int scanResult = sscanf(res, "%dx%d", &resolucionHorizontal, &resolucionVertical);

if (scanResult < 2 || resolucionVertical <= 0 || resolucionHorizontal <= 0)

{

if (resolucionVertical <= 0) printf("Resolucion vertical invalida\n");

if (resolucionHorizontal <= 0) printf("Resolucion horizontal invalida\n");

print\_usage();

exit(EXIT\_FAILURE);

}

} else {

resolucionHorizontal = 640;

resolucionVertical = 480;

}

float pasoHorizontal = ((anchoRectangulo) / (float)(resolucionHorizontal))/2;

float pasoVertical = ((altoRectangulo) / (float)(resolucionVertical))/2;

int contadorBrillo;

float temp, valorAbsoluto;

FILE \*fp;

if (salidaConsola != 1){

fp=fopen(output, "w");

if(fp == NULL) {

printf("fatal: cannot open output file\n");

print\_usage();

exit(EXIT\_FAILURE);

}

fprintf(fp, "P2 \n%d %d \n255 \n", resolucionHorizontal, resolucionVertical);

}

//debug printf("constantR %f constanteI %f orinenR %f origenI %f \n", constanteC.parteReal, constanteC.parteImaginaria, origen.parteReal, origen.parteImaginaria);

for (int y = 1; y <= resolucionVertical; y++){

for (int x = 1; x <= resolucionHorizontal; x++){

pixel.parteReal = pasoHorizontal \* (2 \* x -1) - anchoRectangulo / 2 + origen.parteReal;

pixel.parteImaginaria = -pasoVertical \* (2 \* y -1) + altoRectangulo / 2 + origen.parteImaginaria;

for (contadorBrillo = 0; contadorBrillo <= N; contadorBrillo++){

valorAbsoluto = sqrtf((pixel.parteReal \* pixel.parteReal) + (pixel.parteImaginaria \* pixel.parteImaginaria));

if (valorAbsoluto > 2) break;

temp = ((pixel.parteReal)\*(pixel.parteReal))-((pixel.parteImaginaria)\*(pixel.parteImaginaria)) + constanteC.parteReal;

pixel.parteImaginaria = ((pixel.parteImaginaria)\*(pixel.parteReal))+((pixel.parteReal)\*(pixel.parteImaginaria)) + constanteC.parteImaginaria;

pixel.parteReal = temp;

}

if (salidaConsola == 1) fprintf(stdout, "%d ", contadorBrillo); else fprintf(fp, "%d ", contadorBrillo );

}

if (salidaConsola == 1) fprintf(stdout, "\n"); else fprintf(fp, "\n");

}

if (salidaConsola != 1){

rewind(fp);

fclose(fp);

printf("Archivo guardado en %s/%s\n", pathToExe, output);

}

/\*

if (salidaConsola == 1){

fp=fopen(output, "rb+");

if(fp == NULL) return 1;

int c;

while((c=fgetc(fp)) != EOF){

printf( "%c", c );

}

fclose(fp);

remove(output);

}\*/

return 0;

}