FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo práctico Nº0
Infraestructura básica

Fusaro, Franco - Padrón: 95512
ffusaro94@gmail.com
Impaglione, Rocio - Padrón: 94178
rocio.impaglione@gmail.com

1er Cuatrimestre de 2018
66.20 Organización de Computadoras
Facultad de Ingeniería, Universidad de Buenos Aires

Índice

1. Introducción

El objetivo del presente trabajo práctico es lograr que los alumnos se familiaricen con las herramientas de software que se utilizarán en la materia. A tal fin, los alumnos deberán implementar un programa que resuelva el problema piloto expuesto en el enunciado, junto con la documentación correspondiente.

2. Desarrollo del Trabajo Práctico

2.1. Diseño e Implementación

La consigna del trabajo practico era desarrollar una aplicacion que genere imagenes PGM con los conjuntos de fractales de Julia. A tales fines, se utilizo el lenguaje C para programar lo pedido. Se respetaron las especificaciones dadas por el enunciado entregado por los profesores con respecto a parametros de entrada, salida y forma de procesamiento.

2.2. Hipótesis y Supuestos

- En las validaciones de entrada, asumimos que no es valido que el usuario ponga como valor de alto (parametro -H) mayor que el alto de la resolucion de la imagen.
- La misma consideracion se realizo con respecto al ancho (-w). Asumimos que no debe ser mayor que el ancho de la resolucion de la imagen.

2.3. Compilación del programa

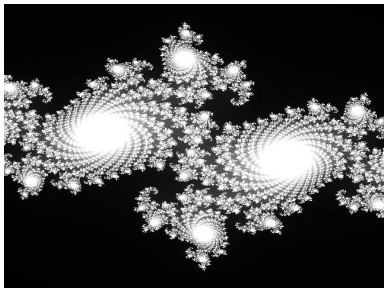
El archivo fuente del programa se denomina tp0.c y se incluye un archivo funciones.c/funciones.h en el cual se encuentran algunas funciones complementarias. Para compilar el programa, se debe utilizar el comando **make** en el directorio donde se encuentre dicho archivo, pues incluimos un Makefile para facilitar este proceso (Fuera de NetBSD). Luego, para ejecutarlo, basta con situarse en la carpeta donde haya sido compilado el programa e ingresar **./{tp0}** con los parámetros que quiera utilizar. Para ejecutar en NetBSD, se compila con el comando **gcc -Wall -O0 tp0.c funciones.c -lm -o {archivo ejecutable}** y para obtener el código assembly se ejecuta el comnado **gcc -Wall -O0 -S -mrnames tp0.c -o {archivo salida}** y **gcc -Wall -O0 -S -mrnames funciones.c -o {archivo salida}**

2.4. Funcionamiento del programa

Como ya se menciono, una vez que se realiza la compilacion detallada en el inciso 2.3, el programa esta listo para usarse.
Para correr la aplicacion con los parametros por defecto, solamente basta abrir una terminal y escribir:

./tp0

La ejecucion generara un archivo llamado output.pgm, que contendra la siguiente imagen:



Si el usuario deseara cambiar el nombre del archivo de salida por defecto, podra hacerlo a traves del parametro -o. Por ejemplo:

./tp0 -o salida.pgm

Esto generara el archivo de salida en un fichero llamado salida.pgm en vez del nombre por defecto.
Algo importante a notar, es que la aplicacion valida el ingreso de un directorio valido y de una extension de archivo correcta (.pgm). Si el usuario ingresa mal alguno de estos valores, el programa finalizara devolviendo error.

./tp0 -o salida.jpg

Output file type is invalid. Type 'tp0 -h' for help. Program terminated

./tp0 -o directorio/inexistente/archivo.pgm

Output file path is invalid. Type 'tp0 -h' for help. Program terminated

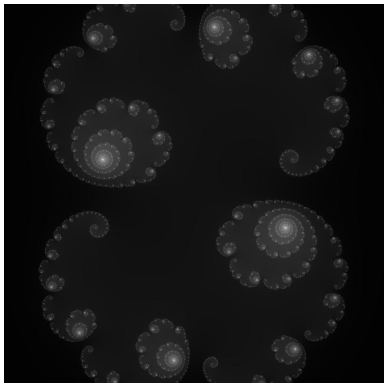
Ademas, la aplicacion admite otros parametros. Se puede generar una imagen con una resolucion distinta a la que viene por defecto (640x480) utilizando el parametro -r e ingresando un valor de la forma AnchoxAlto.

./tp0 -r 800x600

La aplicacion se utiliza para generar distintos fractales del set de Julia, por lo que el usuario puede ingresar como parametro una semilla a partir de la cual se generara la imagen:

./tp0 -s 0.282-0.007i -o salida1.pgm

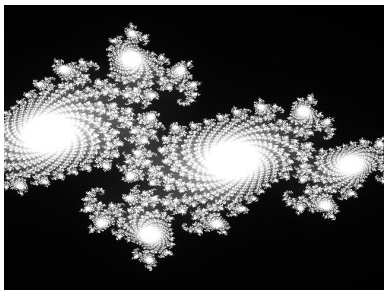
La siguiente ejecucion producira una imagen salida1.pgm con el siguiente fractal:



Ademas la aplicacion permite centrar la imagen en un punto deseado. Para ello, se utilizara el parametro -c y se especificara el centro de la imagen:

tp0 -c 0.282-0.007i

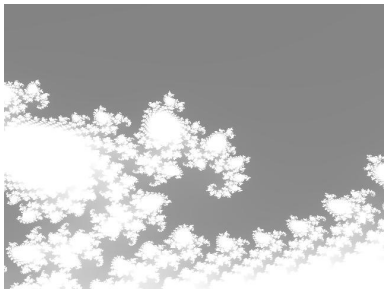
De esta forma se obtendra la siguiente imagen, que, como puede verse, es la imagen por defecto centrada en el punto 0.282-0.007i.



Si el usuario lo desea, tambien puede ampliar una parte de la imagen utilizando los parametros -H y -w:

tp0 -c 0.282-0.007i -w 0.005 -H 0.005

Habiendo ingresado estos parametros, la imagen que se obtendra es la siguiente:



En caso de que existan dudas sobre el uso de la aplicacion, el usuario puede escribir

./tp0 -h

TP0 Organizacion de computadoras - HELP
Usage:
tp0 -h Displays help and usage of the application
tp0 -V Displays version of the application
tp0 Options:
-r –resolution Set bitmap resolution to WxH pixels
-c –center Set the center of the image
-s –seed Set the seed to generate the image
-w –width Set the width of the region to be spanned
-H –height Set the height of the region to be spanned
-o –output Set path to output file
Examples:
tp0 -r 1024x768 -o example.pgm
tp0 -c 0.282-0.007i -w 0.005 -H 0.005 -o example.pgm

La version de la aplicacion puede verse utilizando el parametro -V:

tp0 -V TP0 Organizacion de computadoras - VERSION: 1.0

2.5. Corridas de Prueba

En este apartado se presentan las corridas de prueba que muestran el funcionamiento de la aplicacion desarrollada.

Lo que primero realizamos para las pruebas fue el chequeo de la línea de comandos ingresada. Para esto realizamos un archivo test_tp0.c en el cual incluímos los chequeos de las funciones realizadas de parseo de números imaginarios, de parseo de la resolución y de ingreso de valores válidos de ancho y alto. Este archivo lo incluímos en la entrega para que puedan verlo y compilarlo con "make test_tp0" (fuera de NetBSD) y correrlo con ./test_tp0 para ver su salida que es la siguiente:

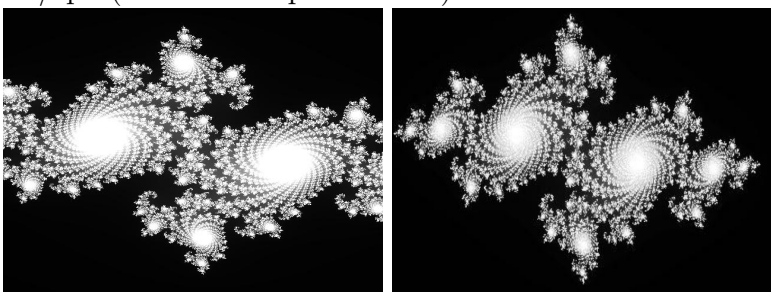
```
./test_tp0
==========================================================
START TESTS: String a numero imaginario con enteros.
==========================================================
Parseando nro imaginario:3+3i
Test OK
Parseando nro imaginario: +3+3i
Test OK
Parseando nro imaginario: +3-3i
Test OK
Parseando nro imaginario:2-5i
Test OK
Parseando nro imaginario:-2+4i
Test OK
Parseando nro imaginario: -8-4i
Test OK
END TESTS: string a numero imaginario con entero. STATUS: OK
==========================================================
START TESTS: String a numero imaginario con decimales.
==========================================================
Parseando nro imaginario: 9.25+0.75i
Test OK
Parseando nro imaginario: 2.5-7.5i
Test OK
Parseando nro imaginario: -2.25+4.5i
Test OK
Parseando nro imaginario: -8.75-4.25i
Test OK
END TESTS: string a numero imaginario con decimales. STATUS: OK
==========================================================
START TESTS: Strings inv lidos.
==========================================================
Parseando nro imaginario: '' (Vacio)
Test OK
Parseando nro imaginario: 'cualquiera'
Test OK
```

```
Parseando nro imaginario: 3 +3i
Test OK
Parseando nro imaginario: 3i+3
Test OK
Parseando nro imaginario: 0i
Test OK
Parseando nro imaginario: 0
Test OK
END TESTS: Strings inv lidos. STATUS: OK
================================================================
START TESTS: RESOLUCION.
================================================================
Parseando resolucion:800x600
Test OK
Parseando resolucion:1480x1970
Test OK
Parseando resolucion:aaaaaaaa
Test OK
Parseando resolucion:aaaaxaaaa
Test OK
Parseando resolucion:-123x456i
Test OK
Parseando resolucion:12354528
Test OK
Parseando resolucion:\00x\00
Test OK
================================================================
START TESTS: ARCHIVO SALIDA.
================================================================
Parseando archivo de salida:file.pgm
Test OK
Parseando archivo de salida:/dir/file.pgm
Test OK
Parseando archivo de salida:NULL
Test OK
Parseando archivo de salida:aa*
Test OK
Parseando archivo de salida:archivo.jpg
Test OK
================================================================
START TESTS: VALIDACION DE ANCHO/ALTO.
================================================================
Validando ancho:
Test OK
Validando alto:
Test OK
Validando ancho:2
Test OK
```

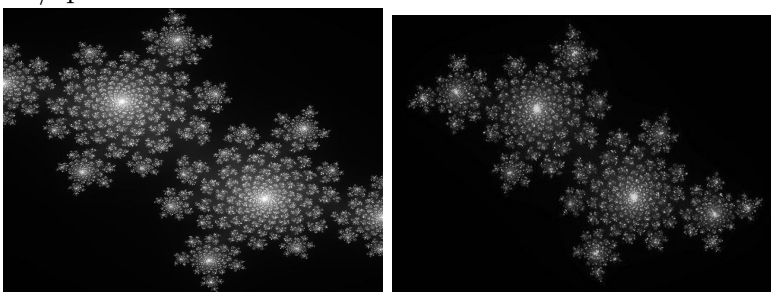
```
Validando alto:2
Test OK
Validando ancho:0.0005
Test OK
Validando alto:0.0005
Test OK
Validando ancho:aaaaaa
Test OK
Validando alto:aaaaaa
Test OK
Validando ancho:\00x\00
Test OK
Validando alto:\00x\00
Test OK
Validando ancho:****
Test OK
Validando alto:****
Test OK
Validando ancho:700
Test OK
Validando alto:500
Test OK
```

De esta manera, ya con la linea de comandos parseada y testeada comenzamos con la lógica del algoritmo y luego con sus pruebas. Primero testeamos los dos casos base dados en el enunciado y luego utilizamos un generador online para comparar con nuestros resultados (esto fue realizado a ojo). Para poder comparar los resultados, se utilizo el siguiente generador de fractales online http://www.easyfractalgenerator.com/julia-set-generator.aspx y se generaron las mismas imagenes que en nuestra aplicacion. En este apartado, se mostrara la linea de comando utilizada para generar la imagen, luego la imagen generada en la aplicacion y por ultimo la obtenida del generador online.

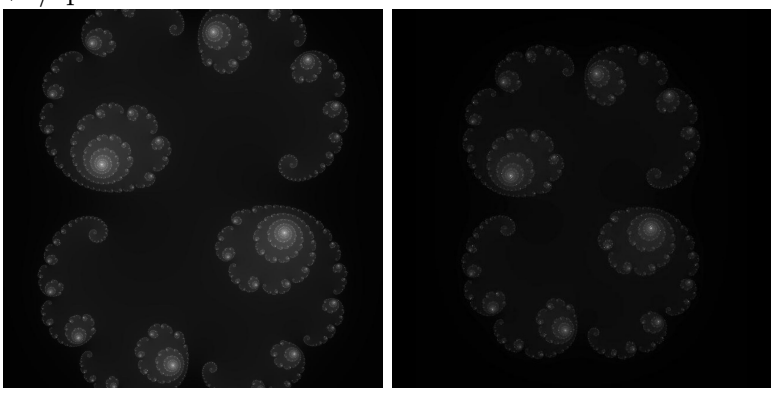
$ ./tp0 (Parametros por defecto)



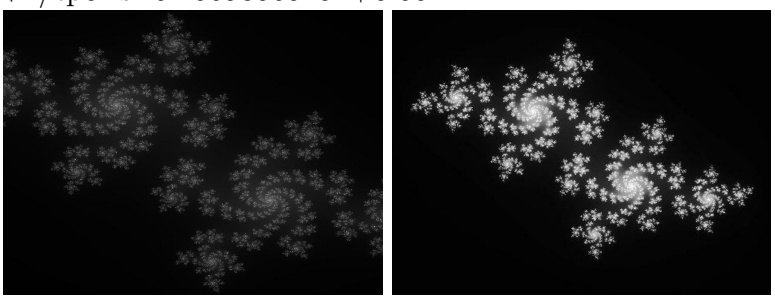
$ ./tp0 -s -0.4+0.6i



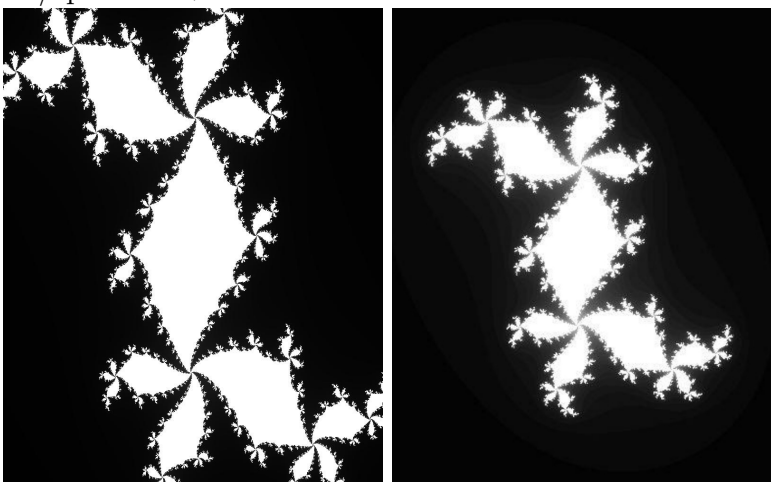
$ ./tp0 -s 0.282-0.007i -r 1000x1000



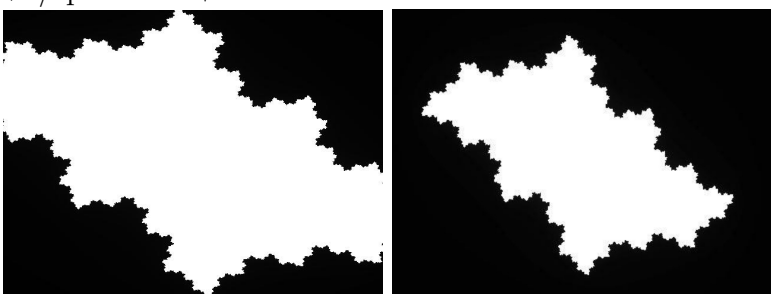
$ ./tp0 -s -0.45698565454+0.60i



$ ./tp0 -s 0.3+0.525866123i -r 800x1000



$ ./tp0 -s -0.3+0.525866123i -r 800x1000



Como puede observarse, salvando algunos detalles de la resolucion y la calidad de las imagenes, la semejanza entre las generadas a traves del generador y las generadas a partir de nuestra aplicacion es notable.

3. Código Fuente

3.1. Código C

tp0.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <stdint.h>
#include <ctype.h>
#include <unistd.h>
#include <poll.h>
#include <errno.h>
#include <getopt.h>
#include <float.h>
#include <assert.h>
#include <string.h>
#include "funciones.h"

typedef struct receivedParameters {
    char* path_to_output;
        resolution_t resolution;
        float width;
        float height;
        nro_imaginario_t center;
        nro_imaginario_t seed;
    color_t color;
} parameters_t;

static struct option long_options[] =
{
        {"resolution",required_argument,NULL,'r'},
        {"center",required_argument,NULL,'c'},
        {"width",required_argument,NULL,'w'},
        {"height",required_argument,NULL,'H'},
        {"seed",required_argument,NULL,'s'},
        {"output",required_argument,NULL,'o'},
        {"help",no_argument,NULL,'h'},
        {"version",no_argument,NULL,'V'},
    {NULL, 0, NULL, 0}
};

/* Declaracion de funciones */

void showVersion(void);
void showHelp();
void showError(int);
void returnValidation(int);
void julia(parameters_t);
void writeHeader(parameters_t,FILE*);
parameters_t getParameters(int argc, char **argv);

parameters_t getParameters(int argc, char **argv){
    int ch;
    parameters_t receivedParameters;

    //valores por defecto
```

```
receivedParameters.path_to_output = "output.pgm";
receivedParameters.resolution.width = 640;
receivedParameters.resolution.height = 480;
receivedParameters.center.real = 0;
receivedParameters.center.img = 0;
receivedParameters.width = 2;
receivedParameters.height = 2;
receivedParameters.seed.real = -0.726895347709114071439;
receivedParameters.seed.img = 0.188887129043845954792;
receivedParameters.color.max = 255;
receivedParameters.color.min = 0;

//fin valores por defecto

// loop over all of the options
while ((ch = getopt_long(argc, argv, "hVo:c:r:H:w:s:", long_options,
   NULL)) != -1) {
    // check to see if a single character or long option came through
    switch (ch){
        case 'o':
            returnValidation(checkForOutputPath(optarg,&
                receivedParameters.path_to_output));
            break;
        case 'c':
            returnValidation(parseNroImg(optarg,&receivedParameters.
                center));
            break;
        case 'r':
            returnValidation(parseResolution(optarg,&receivedParameters
                .resolution));
            break;
        case 'V':
            showVersion();
            exit(0);
            break;
        case 'h':
            showHelp();
            exit(0);
            break;
        case 'H':
            returnValidation(setHeight(optarg,&receivedParameters.
                height,receivedParameters.resolution.height));
        case 'w':
            returnValidation(setWidth(optarg,&receivedParameters.width,
                receivedParameters.resolution.width));
            break;
        case 's':
                            returnValidation(parseNroImg(optarg,&
                                receivedParameters.seed));
            break;
        case '?':
            if (optopt == 'o' || optopt == 'c' || optopt == 'H' ||
                optopt == 'w' || optopt == 's') {
                fprintf (stderr, "No arguments provided for option -%c
                    .\n", optopt);
            } else if (isprint (optopt)) {
                fprintf (stderr, "Unknown option `-%c'.\n", optopt);
            } else {
```

```
                    fprintf (stderr, "Unknown option '\\x%x'.\n", optopt);
                } //Ya se escribi un error a stderr (lo hizo la funcion
                    getopt_long.
            default:
                showHelp();
                exit(1);
        }
    }
    return receivedParameters;

}

void returnValidation(int result){
    if (result < 0){
        showError(result);
        exit(1);
    }
}

void showHelp(){
    printf("%s\n", "TP0 Organizacion de computadoras - HELP");
    printf("%s\n", "Usage: ");
    printf("%s\n", "tp0 -h    Displays help and usage of the application");
    printf("%s\n", "tp0 -V    Displays version of the application");
    printf("%s\n", "tp0 Options:");
    printf("%s\n", "-r --resolution  Set bitmap resolution to WxH pixels");
    printf("%s\n", "-c --center      Set the center of the image");
    printf("%s\n", "-s --seed        Set the seed to generate the image");
    printf("%s\n", "-w --width       Set the width of the region to be
        spanned");
    printf("%s\n", "-H --height      Set the height of the region to be
        spanned");
    printf("%s\n", "-o --output      Set path to output file");
    printf("%s\n", "Examples: ");
    printf("%s\n", "  tp0 -r 1024x768 -o example.pgm");
    printf("%s\n", "  tp0 -c 0.282-0.007i -w 0.005 -H 0.005 -o example.pgm"
        );
}

void showVersion(){
        printf("%s\n", "TP0 Organizacion de computadoras - VERSION: 1.0");
}

void showError(int errorCode) {
    if (errorCode == ERR_VACIO) {
        fprintf(stderr, "%s\n", "Parameter input is empty. Type 'tp0 -h'
            for help. Program terminated");
    }
    if (errorCode == ERR_INVALID_CHARS) {
        fprintf(stderr, "%s\n", "Argument has invalid characters. Type 'tp0
            -h' for help. Program terminated");
    }
    if (errorCode == ERR_INVALID_FORMAT) {
        fprintf(stderr, "%s\n", "Invalid arguments. Type 'tp0 -h' for help.
            Program terminated");
    }
    if (errorCode == ERR_NO_REAL_PART) {
        fprintf(stderr, "%s\n", "Parameter has no real part. Type 'tp0 -h'
```

```
            for help. Program terminated");
    }
    if (errorCode == ERR_NO_IMG_PART) {
        fprintf(stderr, "%s\n", "Parameter has no immaginary part. Type '
            tp0 -h' for help. Program terminated");
    }
    if (errorCode == ERR_INVALID_RESOLUTION) {
        fprintf(stderr, "%s\n", "Resolution input is invalid. Type 'tp0 -h'
            for help. Program terminated");
    }
    if (errorCode == ERR_INVALID_FILE_TYPE) {
        fprintf(stderr, "%s\n", "Output file type is invalid. Type 'tp0 -h'
            for help. Program terminated");
    }
    if (errorCode == ERR_INVALID_FILE_PATH) {
        fprintf(stderr, "%s\n", "Output file path is invalid. Type 'tp0 -h'
            for help. Program terminated");
    }
    if (errorCode == ERR_INVALID_PARAMETER) {
        fprintf(stderr, "%s\n", "Parameter input is invalid. Type 'tp0 -h'
            for help. Program terminated");
    }
}

void julia(parameters_t parameters){
    FILE * output;

    if(strcmp(parameters.path_to_output, "stdout") != 0){ // Abre el
        archivo y genera el header del archivo si la salida no es la
        estandar
        output = fopen(parameters.path_to_output, "w");
        writeHeader(parameters,output);
    } else {
        output = stdout;
    }

        //Calculo los incrementos en alto y ancho
        float stepWidth = parameters.width / parameters.resolution.width;
        float stepHeight = parameters.height / parameters.resolution.height
            ;
        nro_imaginario_t currentPixel;
        //Calculo el primer pixel segun el centro
        nro_imaginario_t startPixel;
        startPixel.real = parameters.center.real - (parameters.width/2);
        startPixel.img = parameters.center.img + (parameters.height/2);
        int N = parameters.color.max;
        int brillo = parameters.color.min;
        int i;
        int j;
        //para cada pixel $p {
        for (i = 0; i < parameters.resolution.height; i++)
        {
                for (j = 0; j < parameters.resolution.width; j++)
                {
                        //$f = complejo asociado a $p;
                        currentPixel.real = startPixel.real + (stepWidth *
                            j);
                        currentPixel.img = startPixel.img - (stepHeight * i
```

```
                        );
                    //for ($i = 0; $i < $N - 1; ++$i) {
                    for (brillo = 0; brillo < N; ++brillo) {
                //if (abs($f) > 2)
                if (getNroImgAbs(currentPixel) > 2) {
                    //break;
                    break;
                }
                //$f = $f * $f + $s;
                currentPixel = getNroImgSq(currentPixel);
                currentPixel.real = currentPixel.real + parameters.seed.
                    real;
                currentPixel.img = currentPixel.img + parameters.seed.img;
            }
            //dibujar el punto p con brillo $i;
            if(fprintf(output,"%d ",brillo) < 0){
                fprintf(stderr, "There was an error while writing to output
                    .\n");
                exit(1);
            };
                }
        fprintf(output,"\n");
        }

    if(strcmp(parameters.path_to_output, "stdout") != 0){
        if (fclose(output) != 0) {
            fprintf(stderr, "Unable to close output file.\n");
            exit(1);
        }
    }

}

void writeHeader(parameters_t parameters,FILE* output){
    // PGM Header - De acuerdo con la especificacion, el header de un
        archivo PGM debe contener:
    // El "numero magico" P2, el nombre del archivo con un '#' delante, el
        ancho de la imagen, el alto de la imagen
    // y el color mas oscuro que puede llegar a alcanzar (en este caso, el
        valor 255)
    fprintf(output,"P2\n");
    fprintf(output,"# %s\n",parameters.path_to_output);
    fprintf(output,"%u\n",(unsigned)parameters.resolution.width);
    fprintf(output,"%u\n",(unsigned)parameters.resolution.height);
    fprintf(output,"%u\n",(unsigned)parameters.color.max);
}

int main(int argc, char *argv[]){
    int result = 0;
    parameters_t receivedParameters = getParameters(argc,argv);
    julia(receivedParameters);
    return result;
}
```

funciones.h

```
#define ERR_VACIO -1
#define ERR_INVALID_CHARS -2
#define ERR_INVALID_FORMAT -3
```

```
#define ERR_NO_REAL_PART -4
#define ERR_NO_IMG_PART -5
#define ERR_INVALID_RESOLUTION -6
#define ERR_INVALID_FILE_TYPE -7
#define ERR_INVALID_FILE_PATH -8
#define ERR_INVALID_PARAMETER -9

typedef struct nro_imaginario
{
        long double real;
        long double img;
} nro_imaginario_t;

typedef struct resolution {
        int width;
        int height;
} resolution_t;

typedef struct color {
    int max;
    int min;
}color_t;

char** str_split(char* a_str, const char a_delim);
int stringContainsChar(char* string, const char* ch);
int parseNroImg(char* nro, nro_imaginario_t* nro_img);
int parseResolution(char* resolstr, resolution_t* resolution);
int checkForOutputPath(char* path, char** path_to_save);
char* getFileExtension(char *path);
int checkForBadCharacters(char* path);
int setWidth(char* strvalue, float* value_to_set, float value_to_check);
int setHeight(char* strvalue, float* value_to_set, float value_to_check);
long double getNroImgAbs(nro_imaginario_t nro);
nro_imaginario_t getNroImgSq(nro_imaginario_t nro);
```

funciones.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include <ctype.h>
#include <unistd.h>
#include <poll.h>
#include <errno.h>
#include <getopt.h>
#include <float.h>
#include <assert.h>
#include <math.h>
#include "funciones.h"

/* ---------------------------------
        FUNCIONES AUXILIARES
---------------------------------*/
char** str_split(char* a_str, const char a_delim)
{
    char** result    = 0;
    size_t count     = 0;
    char* tmp        = a_str;
```

```
    char* last_comma = 0;
    char delim[2];
    delim[0] = a_delim;
    delim[1] = 0;

    /* Count how many elements will be extracted. */
    while (*tmp)
    {
        if (a_delim == *tmp)
        {
            count++;
            last_comma = tmp;
        }
        tmp++;
    }

    /* Add space for trailing token. */
    count += last_comma < (a_str + strlen(a_str) - 1);

    /* Add space for terminating null string so caller
       knows where the list of returned strings ends. */
    count++;

    result = malloc(sizeof(char*) * count);

    if (result)
    {
        size_t idx  = 0;
        char* token = strtok(a_str, delim);

        while (token)
        {
            assert(idx < count);
            *(result + idx++) = strdup(token);
            token = strtok(0, delim);
        }
        assert(idx == count - 1);
        *(result + idx) = 0;
    }

    return result;
}

int stringContainsChar(char* string, const char* ch)
{
        size_t len = strlen(string);
        size_t spn = strcspn(string, ch);
        if (spn == 0){
                return 1; //si el len del segundo es 0 quiere decir que
                    contiene todos los chars.
        }
        if (len != spn){
                return 0; //si son diferentes quiere decir que lo contiene
        }
        return -1; //si no son diferentes no lo contiene
}

int parseNroImg(char* nro, nro_imaginario_t* nro_img){
```

```
    //char* nroImg = strdup(nro);
        if (!strcmp(nro, "") || nro == NULL){
                return ERR_VACIO;
        }

        long double real, img;
    char i;

        if (stringContainsChar(nro, "0123456789.+-") == -1 ||
            stringContainsChar(nro," ") == 0){
                return ERR_INVALID_CHARS;
        }

    if (sscanf(nro, "%Lf%Lf%c", &real, &img, &i) != 3 || i != 'i'){
        return ERR_INVALID_FORMAT;
    }

    nro_img->real = real;
    nro_img->img = img;
        return 0;
}

int parseResolution(char* resolstr, resolution_t* resolution) {
    int width, height;
    char separator, rest;

    if (sscanf(resolstr, "%d%c%d %c", &width, &separator, &height, &rest)
       != 3 || width <= 0 || separator != 'x' || height <= 0){
        return ERR_INVALID_RESOLUTION;
    }

    resolution->width = width;
    resolution->height = height;
    return 0;
}

int checkForOutputPath(char* path, char** path_to_save){
    if (path == NULL) return ERR_VACIO; // Chequea que el path no sea nulo

    if (strcmp(path, "-") == 0) {
        *path_to_save = "stdout";
        return 0;
    } // Si el nombre del archivo es "-" se utilizara la salida estandar

    if(checkForBadCharacters(path) == -1) return ERR_INVALID_CHARS;

    FILE* file;
    if(strcmp(getFileExtension(path), ".pgm") != 0) return
       ERR_INVALID_FILE_TYPE;
    else if (!(file = fopen(path, "w"))) return ERR_INVALID_FILE_PATH;
    fclose(file);

    *path_to_save = path;
    return 0;
}

char* getFileExtension(char* path) {
    char *ext = strrchr(path, '.');
```

```
    return (ext && ext != path) ? ext : (path + strlen(path));
}

int checkForBadCharacters(char* path){
    char bad_chars[] = "!@%^*~|";
    int i;
    for (i = 0; i < strlen(bad_chars); ++i) {
        if (strchr(path, bad_chars[i]) != NULL) {
            return -1;
        }
    }
    return 0;
}

int setHeight(char* height, float* value_to_set, float resolution_height){
    float value;
    if (sscanf(height,"%f",&value) != 1 || value < 0 || value >=
       resolution_height) return ERR_INVALID_PARAMETER;
    *value_to_set = value;
    return 0;
}

int setWidth(char* width, float* value_to_set, float resolution_width){
    float value;
    if (sscanf(width,"%f",&value) != 1 || value < 0 || value >=
       resolution_width) return ERR_INVALID_PARAMETER;
    *value_to_set = value;
    return 0;
}

long double getNroImgAbs(nro_imaginario_t nro)
{
        return sqrt(nro.real*nro.real + nro.img*nro.img);
}

nro_imaginario_t getNroImgSq(nro_imaginario_t nro)
{
        nro_imaginario_t sqrNro;
        sqrNro.real = (nro.real * nro.real) - (nro.img * nro.img);
        sqrNro.img = 2 * nro.real * nro.img;
        return sqrNro;
}
```

test_tp0.c

```
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "funciones.h"
#define ANSI_COLOR_RED     "\x1b[31m"
#define ANSI_COLOR_GREEN   "\x1b[32m"
#define ANSI_COLOR_YELLOW  "\x1b[33m"
#define ANSI_COLOR_BLUE    "\x1b[34m"
#define ANSI_COLOR_MAGENTA "\x1b[35m"
#define ANSI_COLOR_CYAN    "\x1b[36m"
#define ANSI_COLOR_WHITE   "\x1b[37m"
#define ANSI_COLOR_RESET   "\x1b[0m"
```

```
void assertPropio(int condition, char* msg);

void test_chars_validos_nro_imaginario()
{
        nro_imaginario_t nroImg;
        int result;

        /* TESTS PARA CONVERTIR STRING A NUMERO IMAGINARIO */

        /* PRIMERO TESTEAMOS LOS CASOS BASE DE COMBINACIONES DE + Y - CON
           NUMEROS ENTEROS */
    printf(ANSI_COLOR_WHITE "
       ==========================================================\n");
        printf(ANSI_COLOR_YELLOW "START TESTS: String a numero imaginario
           con enteros.\n");
    printf(ANSI_COLOR_WHITE "
       ==========================================================\n");

        result = parseNroImg("3+3i",&nroImg);
        printf(ANSI_COLOR_YELLOW "Parseando nro imaginario:"
           ANSI_COLOR_WHITE "3+3i\n");
        assertPropio(result == 0, "El parseo devolvi negativo");
        assertPropio(nroImg.real == 3, "La parte real debe ser 3");
        assertPropio(nroImg.img == 3, "La parte imaginaria debe ser 3");
        printf(ANSI_COLOR_GREEN "Test OK\n");

        printf(ANSI_COLOR_YELLOW "Parseando nro imaginario: "
           ANSI_COLOR_WHITE "+3+3i\n");
        result = parseNroImg("+3+3i",&nroImg);
        assertPropio(nroImg.real == 3, "La parte real debe ser 3");
        assertPropio(nroImg.img == 3, "La parte imaginaria debe ser 3");
        assertPropio(result == 0, "El parseo devolvi negativo.");
        printf(ANSI_COLOR_GREEN "Test OK\n");

        printf(ANSI_COLOR_YELLOW "Parseando nro imaginario: "
           ANSI_COLOR_WHITE "+3-3i\n");
        result = parseNroImg("+3-3i",&nroImg);
        assertPropio(result == 0, "El parseo devolvi negativo");
        assertPropio(nroImg.real == 3, "La parte real debe ser 3");
        assertPropio(nroImg.img == -3, "La parte imaginaria debe ser -3");
        printf(ANSI_COLOR_GREEN "Test OK\n");

        result = parseNroImg("2-5i",&nroImg);
        printf(ANSI_COLOR_YELLOW "Parseando nro imaginario:"
           ANSI_COLOR_WHITE "2-5i\n");
        assertPropio(result == 0, "El parseo devolvi negativo");
        assertPropio(nroImg.real == 2, "La parte real debe ser 2");
        assertPropio(nroImg.img == -5, "La parte imaginaria debe ser -5");
        printf(ANSI_COLOR_GREEN "Test OK\n");

        result = parseNroImg("-2+4i",&nroImg);
        printf(ANSI_COLOR_YELLOW "Parseando nro imaginario:"
           ANSI_COLOR_WHITE "-2+4i\n");
        assertPropio(result == 0, "El parseo devolvi negativo");
        assertPropio(nroImg.real == -2, "La parte real debe ser -2");
        assertPropio(nroImg.img == 4, "La parte imaginaria debe ser 4");
        printf(ANSI_COLOR_GREEN "Test OK\n");
```

```
    result = parseNroImg("-8-4i",&nroImg);
    printf(ANSI_COLOR_YELLOW "Parseando nro imaginario: "
        ANSI_COLOR_WHITE "-8-4i\n");
    assertPropio(result == 0, "El parseo devolvi negativo");
    assertPropio(nroImg.real == -8, "La parte real debe ser -8");
    assertPropio(nroImg.img == -4 , "La parte imaginaria debe ser -4");
    printf(ANSI_COLOR_GREEN "Test OK\n");

    printf(ANSI_COLOR_YELLOW "END TESTS: string a numero imaginario con
        entero. STATUS: " ANSI_COLOR_GREEN "OK\n");

    /* TESTEAMOS LOS CASOS BASE DE COMBINACIONES DE + Y - CON NUMEROS
        DECIMALES */
printf(ANSI_COLOR_WHITE "
    ==========================================================\n");
    printf(ANSI_COLOR_YELLOW "START TESTS: String a numero imaginario
        con decimales.\n");
printf(ANSI_COLOR_WHITE "
    ==========================================================\n");

    result = parseNroImg("9.25+0.75i",&nroImg);
    printf(ANSI_COLOR_YELLOW "Parseando nro imaginario: "
        ANSI_COLOR_WHITE "9.25+0.75i\n");
    assertPropio(result == 0, "El parseo devolvi negativo");
    assertPropio(nroImg.real == 9.25, "La parte real debe ser 9.25");
    assertPropio(nroImg.img == 0.75, "La parte imaginaria debe ser 0.75
        ");
    printf(ANSI_COLOR_GREEN "Test OK\n");

    result = parseNroImg("2.5-7.5i",&nroImg);
    printf(ANSI_COLOR_YELLOW "Parseando nro imaginario: "
        ANSI_COLOR_WHITE "2.5-7.5i\n");
    assertPropio(result == 0, "El parseo devolvi negativo");
    assertPropio(nroImg.real == 2.5, "La parte real debe ser 2.5");
    assertPropio(nroImg.img == -7.5, "La parte imaginaria debe ser -5")
        ;
    printf(ANSI_COLOR_GREEN "Test OK\n");

    result = parseNroImg("-2.25+4.5i",&nroImg);
    printf(ANSI_COLOR_YELLOW "Parseando nro imaginario: "
        ANSI_COLOR_WHITE "-2.25+4.5i\n");
    assertPropio(result == 0, "El parseo devolvi negativo");
    assertPropio(nroImg.real == -2.25, "La parte real debe ser -2.25");
    assertPropio(nroImg.img == 4.5, "La parte imaginaria debe ser 4.5")
        ;
    printf(ANSI_COLOR_GREEN "Test OK\n");

    result = parseNroImg("-8.75-4.25i",&nroImg);
    printf(ANSI_COLOR_YELLOW "Parseando nro imaginario: "
        ANSI_COLOR_WHITE "-8.75-4.25i\n");
    assertPropio(result == 0, "El parseo devolvi negativo");
    assertPropio(nroImg.real == -8.75, "La parte real debe ser -8.75");
    assertPropio(nroImg.img == -4.25, "La parte imaginaria debe ser
        -4.25");
    printf(ANSI_COLOR_GREEN "Test OK\n");

    printf(ANSI_COLOR_YELLOW "END TESTS: string a numero imaginario con
        decimales. STATUS: " ANSI_COLOR_GREEN "OK\n");
```

```
        /* TESTEAMOS LOS CASOS DE STRINGS INVALIDOS */
    printf(ANSI_COLOR_WHITE "
        ==========================================================\n");
        printf(ANSI_COLOR_YELLOW "START TESTS: Strings inv lidos.\n");
    printf(ANSI_COLOR_WHITE "
        ==========================================================\n");

        printf(ANSI_COLOR_YELLOW "Parseando nro imaginario: "
            ANSI_COLOR_WHITE "'' (Vacio)\n");
        result = parseNroImg("",&nroImg);
        assertPropio(result == ERR_VACIO, "Deberia devolver la constante
            ERR_VACIO");
        printf(ANSI_COLOR_GREEN "Test OK\n");

        printf(ANSI_COLOR_YELLOW "Parseando nro imaginario: "
            ANSI_COLOR_WHITE "'cualquiera'\n");
        result = parseNroImg("ERR_INVALID_CHARS",&nroImg);
        assertPropio(result == ERR_INVALID_CHARS, "Deberia devolver la
            constante ERR_INVALID_CHARS");
        printf(ANSI_COLOR_GREEN "Test OK\n");

        printf(ANSI_COLOR_YELLOW "Parseando nro imaginario: "
            ANSI_COLOR_WHITE "3 +3i\n");
        result = parseNroImg("3 +3i",&nroImg);
        assertPropio(result == ERR_INVALID_CHARS, "Deberia devolver la
            constante ERR_INVALID_CHARS.");
        printf(ANSI_COLOR_GREEN "Test OK\n");

        printf(ANSI_COLOR_YELLOW "Parseando nro imaginario: "
            ANSI_COLOR_WHITE "3i+3\n");
        result = parseNroImg("3i+3",&nroImg);
        assertPropio(result == ERR_INVALID_FORMAT, "Deberia devolver la
            constante ERR_INVALID_FORMAT.");
        printf(ANSI_COLOR_GREEN "Test OK\n");

        printf(ANSI_COLOR_YELLOW "Parseando nro imaginario: "
            ANSI_COLOR_WHITE "0i\n");
        result = parseNroImg("0i",&nroImg);
        assertPropio(result == ERR_INVALID_FORMAT, "Deberia devolver la
            constante ERR_INVALID_FORMAT.");
        printf(ANSI_COLOR_GREEN "Test OK\n");

        printf(ANSI_COLOR_YELLOW "Parseando nro imaginario: "
            ANSI_COLOR_WHITE "0\n");
        result = parseNroImg("0",&nroImg);
        assertPropio(result == ERR_INVALID_FORMAT, "Deberia devolver la
            constante ERR_INVALID_FORMAT.");
        printf(ANSI_COLOR_GREEN "Test OK\n");

        printf(ANSI_COLOR_YELLOW "END TESTS: Strings inv lidos. STATUS: "
            ANSI_COLOR_GREEN "OK\n");
}

void test_resolucion(){
    printf(ANSI_COLOR_WHITE "
        ==========================================================\n");
        printf(ANSI_COLOR_YELLOW "START TESTS: RESOLUCION.\n");
```

```
    printf(ANSI_COLOR_WHITE "
        ==========================================================\n");
    resolution_t resol;
    int result;

    result= parseResolution("800x600",&resol);
        printf(ANSI_COLOR_YELLOW "Parseando resolucion:" ANSI_COLOR_WHITE "
            800x600\n");
        assertPropio(result == 0, "El parseo fue correcto");
    printf(ANSI_COLOR_GREEN "Test OK\n");

    result = parseResolution("1480x1970",&resol);
    printf(ANSI_COLOR_YELLOW "Parseando resolucion:" ANSI_COLOR_WHITE "1480
        x1970\n");
    assertPropio(result == 0, "El parseo fue correcto");
        printf(ANSI_COLOR_GREEN "Test OK\n");

    result = parseResolution("aaaaaaaa",&resol);
    printf(ANSI_COLOR_YELLOW "Parseando resolucion:" ANSI_COLOR_WHITE "
        aaaaaaaa\n");
    assertPropio(result == ERR_INVALID_RESOLUTION, "El parseo fue correcto"
        );
    printf(ANSI_COLOR_GREEN "Test OK\n");

    result = parseResolution("aaaaxaaaa",&resol);
    printf(ANSI_COLOR_YELLOW "Parseando resolucion:" ANSI_COLOR_WHITE "
        aaaaxaaaa\n");
    assertPropio(result == ERR_INVALID_RESOLUTION, "El parseo fue correcto"
        );
    printf(ANSI_COLOR_GREEN "Test OK\n");

    result = parseResolution("-123x456i",&resol);
    printf(ANSI_COLOR_YELLOW "Parseando resolucion:" ANSI_COLOR_WHITE "-123
        x456i\n");
    assertPropio(result == ERR_INVALID_RESOLUTION, "El parseo fue correcto"
        );
    printf(ANSI_COLOR_GREEN "Test OK\n");

    result = parseResolution("12354528",&resol);
    printf(ANSI_COLOR_YELLOW "Parseando resolucion:" ANSI_COLOR_WHITE "
        12354528\n");
    assertPropio(result == ERR_INVALID_RESOLUTION, "El parseo fue correcto"
        );
    printf(ANSI_COLOR_GREEN "Test OK\n");

    result = parseResolution("\\00x\\00",&resol);
    printf(ANSI_COLOR_YELLOW "Parseando resolucion:" ANSI_COLOR_WHITE "\\00
        x\\00\n");
    assertPropio(result == ERR_INVALID_RESOLUTION, "El parseo fue correcto"
        );
    printf(ANSI_COLOR_GREEN "Test OK\n");
}

void test_archivo_salida(){
    printf(ANSI_COLOR_WHITE "
        ==========================================================\n");
    printf(ANSI_COLOR_YELLOW "START TESTS: ARCHIVO SALIDA.\n");
    printf(ANSI_COLOR_WHITE "
```

```
        ==========\n");
    int result;
        char* path = NULL;

    result = checkForOutputPath("file.pgm",&path);
    printf(ANSI_COLOR_YELLOW "Parseando archivo de salida:"
        ANSI_COLOR_WHITE "file.pgm\n");
    assertPropio(result == 0, "El parseo fue correcto");
    printf(ANSI_COLOR_GREEN "Test OK\n");

    result = checkForOutputPath("/dir/file.pgm",&path);
    printf(ANSI_COLOR_YELLOW "Parseando archivo de salida:"
        ANSI_COLOR_WHITE "/dir/file.pgm\n");
    assertPropio(result == ERR_INVALID_FILE_PATH, "El archivo de salida es
        invalido");
    printf(ANSI_COLOR_GREEN "Test OK\n");

    result = checkForOutputPath(NULL,&path);
    printf(ANSI_COLOR_YELLOW "Parseando archivo de salida:"
        ANSI_COLOR_WHITE "NULL\n");
    assertPropio(result == ERR_VACIO, "El archivo de salida es nulo");
    printf(ANSI_COLOR_GREEN "Test OK\n");

    result = checkForOutputPath("aa*",&path);
    printf(ANSI_COLOR_YELLOW "Parseando archivo de salida:"
        ANSI_COLOR_WHITE "aa*\n");
    assertPropio(result == ERR_INVALID_CHARS, "El archivo de salida tiene
        caracteres invalidos");
    printf(ANSI_COLOR_GREEN "Test OK\n");

    result = checkForOutputPath("archivo.jpg",&path);
    printf(ANSI_COLOR_YELLOW "Parseando archivo de salida:"
        ANSI_COLOR_WHITE "archivo.jpg\n");
    assertPropio(result == ERR_INVALID_FILE_TYPE, "El archivo de salida
        tiene una extension invalida");
    printf(ANSI_COLOR_GREEN "Test OK\n");
}

void test_ancho_alto(){
    printf(ANSI_COLOR_WHITE "
        ==========\n");
    printf(ANSI_COLOR_YELLOW "START TESTS: VALIDACION DE ANCHO/ALTO.\n");
    printf(ANSI_COLOR_WHITE "
        ==========\n");
    int result;
    float value;
    int res_width = 600;
    int res_height = 400;

    result = setWidth("",&value,res_width);
    printf(ANSI_COLOR_YELLOW "Validando ancho:" ANSI_COLOR_WHITE "\n");
    assertPropio(result == ERR_INVALID_PARAMETER, "No se ingreso ningun
        valor");
    printf(ANSI_COLOR_GREEN "Test OK\n");

    result = setHeight("",&value,res_height);
    printf(ANSI_COLOR_YELLOW "Validando alto:" ANSI_COLOR_WHITE "\n");
    assertPropio(result == ERR_INVALID_PARAMETER, "No se ingreso ningun
```

```
    valor");
printf(ANSI_COLOR_GREEN "Test OK\n");

result = setWidth("2",&value,res_width);
printf(ANSI_COLOR_YELLOW "Validando ancho:" ANSI_COLOR_WHITE "2\n");
assertPropio(result == 0, "El parseo fue correcto");
printf(ANSI_COLOR_GREEN "Test OK\n");

result = setHeight("2",&value,res_height);
printf(ANSI_COLOR_YELLOW "Validando alto:" ANSI_COLOR_WHITE "2\n");
assertPropio(result == 0, "El parseo fue correcto");
printf(ANSI_COLOR_GREEN "Test OK\n");

result = setWidth("0.0005",&value,res_width);
printf(ANSI_COLOR_YELLOW "Validando ancho:" ANSI_COLOR_WHITE "0.0005\n"
    );
assertPropio(result == 0, "El parseo fue correcto");
printf(ANSI_COLOR_GREEN "Test OK\n");

result = setHeight("0.0005",&value,res_height);
printf(ANSI_COLOR_YELLOW "Validando alto:" ANSI_COLOR_WHITE "0.0005\n")
    ;
assertPropio(result == 0, "El parseo fue correcto");
printf(ANSI_COLOR_GREEN "Test OK\n");

result = setWidth("aaaaaa",&value,res_width);
printf(ANSI_COLOR_YELLOW "Validando ancho:" ANSI_COLOR_WHITE "aaaaaa\n"
    );
assertPropio(result == ERR_INVALID_PARAMETER, "Se ingreso un valor
    alfabetico");
printf(ANSI_COLOR_GREEN "Test OK\n");

result = setHeight("aaaaaa",&value,res_height);
printf(ANSI_COLOR_YELLOW "Validando alto:" ANSI_COLOR_WHITE "aaaaaa\n")
    ;
assertPropio(result == ERR_INVALID_PARAMETER, "Se ingreso un valor
    alfabetico");
printf(ANSI_COLOR_GREEN "Test OK\n");

result = setWidth("\\00x\\00",&value,res_width);
printf(ANSI_COLOR_YELLOW "Validando ancho:" ANSI_COLOR_WHITE "\\00x
    \\00\n");
assertPropio(result == ERR_INVALID_PARAMETER, "Se ingreso un caracter
    invalido");
printf(ANSI_COLOR_GREEN "Test OK\n");

result = setHeight("\\00x\\00",&value,res_height);
printf(ANSI_COLOR_YELLOW "Validando alto:" ANSI_COLOR_WHITE "\\00x\\00\
    n");
assertPropio(result == ERR_INVALID_PARAMETER, "Se ingreso un caracter
    invalido");
printf(ANSI_COLOR_GREEN "Test OK\n");

result = setWidth("****",&value,res_width);
printf(ANSI_COLOR_YELLOW "Validando ancho:" ANSI_COLOR_WHITE "****\n");
assertPropio(result == ERR_INVALID_PARAMETER, "Se ingreso un caracter
    invalido");
printf(ANSI_COLOR_GREEN "Test OK\n");
```

```
    result = setHeight("****",&value,res_height);
    printf(ANSI_COLOR_YELLOW "Validando alto:" ANSI_COLOR_WHITE "****\n");
    assertPropio(result == ERR_INVALID_PARAMETER, "Se ingreso un caracter
        invalido");
    printf(ANSI_COLOR_GREEN "Test OK\n");

    result = setWidth("700",&value,res_width);
    printf(ANSI_COLOR_YELLOW "Validando ancho:" ANSI_COLOR_WHITE "700\n");
    assertPropio(result == ERR_INVALID_PARAMETER, "Se ingreso un ancho
        mayor que el ancho de la resolucion");
    printf(ANSI_COLOR_GREEN "Test OK\n");

    result = setHeight("500",&value,res_height);
    printf(ANSI_COLOR_YELLOW "Validando alto:" ANSI_COLOR_WHITE "500\n");
    assertPropio(result == ERR_INVALID_PARAMETER, "Se ingreso un alto mayor
        que el alto de la resolucion");
    printf(ANSI_COLOR_GREEN "Test OK\n");
}

void assertPropio(int condition, char* msg)
{
        if (!condition) {
                printf(ANSI_COLOR_RED "ASSERT ERROR: %s\n", msg);
                exit(7);
        }
}

int main(int argc, char *argv[]){
        test_chars_validos_nro_imaginario();
        test_resolucion();
        test_archivo_salida();
    test_ancho_alto();
    return 0;
}
```

3.2. Codigo MIPS32

```
        .file   1 "tp0.c"
        .section .mdebug.abi32
        .previous
        .abicalls
        .rdata
        .align  2
$LC0:
        .ascii  "resolution\000"
        .align  2
$LC1:
        .ascii  "center\000"
        .align  2
$LC2:
        .ascii  "width\000"
        .align  2
$LC3:
        .ascii  "height\000"
        .align  2
```

```
$LC4:
        .ascii  "seed\000"
        .align  2
$LC5:
        .ascii  "output\000"
        .align  2
$LC6:
        .ascii  "help\000"
        .align  2
$LC7:
        .ascii  "version\000"
        .data
        .align  2
        .type   long_options, @object
        .size   long_options, 144
long_options:
        .word   $LC0
        .word   1
        .word   0
        .word   114
        .word   $LC1
        .word   1
        .word   0
        .word   99
        .word   $LC2
        .word   1
        .word   0
        .word   119
        .word   $LC3
        .word   1
        .word   0
        .word   72
        .word   $LC4
        .word   1
        .word   0
        .word   115
        .word   $LC5
        .word   1
        .word   0
        .word   111
        .word   $LC6
        .word   0
        .word   0
        .word   104
        .word   $LC7
        .word   0
        .word   0
        .word   86
        .word   0
        .word   0
        .word   0
        .word   0
        .globl  memcpy
        .rdata
        .align  2
$LC8:
        .ascii  "output.pgm\000"
        .align  2
```

```
$LC12:
        .ascii  "hVo:c:r:H:w:s:\000"
        .align  2
$LC13:
        .ascii  "No arguments provided for option -%c .\n\000"
        .align  2
$LC14:
        .ascii  "Unknown option `-%c'.\n\000"
        .align  2
$LC15:
        .ascii  "Unknown option `\\x%x'.\n\000"
        .align  2
$LC9:
        .word   1073741824
        .align  3
$LC10:
        .word   138464867
        .word   -1075363142
        .align  3
$LC11:
        .word   351303579
        .word   1070083444
        .text
        .align  2
        .globl  getParameters
        .ent    getParameters
getParameters:
        .frame  $fp,128,$ra             # vars= 80, regs= 3/0, args= 24,
           extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,128
        .cprestore 24
        sw      $ra,120($sp)
        sw      $fp,116($sp)
        sw      $gp,112($sp)
        move    $fp,$sp
        sw      $a0,128($fp)
        sw      $a1,132($fp)
        sw      $a2,136($fp)
        sw      $zero,40($fp)
        la      $v0,$LC8
        sw      $v0,44($fp)
        li      $v0,640                 # 0x280
        sw      $v0,48($fp)
        li      $v0,480                 # 0x1e0
        sw      $v0,52($fp)
        sw      $zero,64($fp)
        sw      $zero,68($fp)
        sw      $zero,72($fp)
        sw      $zero,76($fp)
        l.s     $f0,$LC9
        s.s     $f0,56($fp)
        l.s     $f0,$LC9
        s.s     $f0,60($fp)
```

```
        l.d     $f0,$LC10
        s.d     $f0,80($fp)
        l.d     $f0,$LC11
        s.d     $f0,88($fp)
        li      $v0,255                 # 0xff
        sw      $v0,96($fp)
        sw      $zero,100($fp)
$L18:
        sw      $zero,16($sp)
        lw      $a0,132($fp)
        lw      $a1,136($fp)
        la      $a2,$LC12
        la      $a3,long_options
        la      $t9,getopt_long
        jal     $ra,$t9
        sw      $v0,32($fp)
        lw      $v1,32($fp)
        li      $v0,-1                  # 0xffffffffffffffff
        bne     $v1,$v0,$L20
        b       $L19
$L20:
        lw      $v0,32($fp)
        addu    $v0,$v0,-63
        sw      $v0,104($fp)
        lw      $v1,104($fp)
        sltu    $v0,$v1,57
        beq     $v0,$zero,$L36
        lw      $v0,104($fp)
        sll     $v1,$v0,2
        la      $v0,$L37
        addu    $v0,$v1,$v0
        lw      $v0,0($v0)
        .cpadd  $v0
        j       $v0
        .rdata
        .align  2
$L37:
        .gpword $L30
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L27
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
```

```
        .gpword $L36
        .gpword $L36
        .gpword $L25
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L23
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L26
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L22
        .gpword $L36
        .gpword $L36
        .gpword $L24
        .gpword $L29
        .gpword $L36
        .gpword $L36
        .gpword $L36
        .gpword $L28
        .text
$L22:
        addu    $v0,$fp,40
        addu    $v0,$v0,4
        lw      $a0,optarg
        lw      $a1,40($fp)
        move    $a2,$v0
        la      $t9,checkForOutputPath
        jal     $ra,$t9
        move    $a0,$v0
        la      $t9,returnValidation
        jal     $ra,$t9
        b       $L18
$L23:
        addu    $v0,$fp,40
        addu    $v0,$v0,24
        lw      $a0,optarg
        move    $a1,$v0
        la      $t9,parseNroImg
        jal     $ra,$t9
        move    $a0,$v0
        la      $t9,returnValidation
```

```
        jal     $ra,$t9
        b       $L18
$L24:
        addu    $v0,$fp,40
        addu    $v0,$v0,8
        lw      $a0,optarg
        move    $a1,$v0
        la      $t9,parseResolution
        jal     $ra,$t9
        move    $a0,$v0
        la      $t9,returnValidation
        jal     $ra,$t9
        b       $L18
$L25:
        la      $t9,showVersion
        jal     $ra,$t9
        move    $a0,$zero
        la      $t9,exit
        jal     $ra,$t9
$L26:
        la      $t9,showHelp
        jal     $ra,$t9
        move    $a0,$zero
        la      $t9,exit
        jal     $ra,$t9
$L27:
        addu    $v0,$fp,40
        addu    $v0,$v0,20
        lw      $a0,optarg
        move    $a1,$v0
        la      $t9,setValue
        jal     $ra,$t9
        move    $a0,$v0
        la      $t9,returnValidation
        jal     $ra,$t9
$L28:
        addu    $v0,$fp,40
        addu    $v0,$v0,16
        lw      $a0,optarg
        move    $a1,$v0
        la      $t9,setValue
        jal     $ra,$t9
        move    $a0,$v0
        la      $t9,returnValidation
        jal     $ra,$t9
        b       $L18
$L29:
        addu    $v0,$fp,40
        addu    $v0,$v0,40
        lw      $a0,optarg
        move    $a1,$v0
        la      $t9,parseNroImg
        jal     $ra,$t9
        move    $a0,$v0
        la      $t9,returnValidation
        jal     $ra,$t9
        b       $L18
$L30:
```

```
        lw      $v1,optopt
        li      $v0,111                 # 0x6f
        beq     $v1,$v0,$L32
        lw      $v1,optopt
        li      $v0,99                  # 0x63
        beq     $v1,$v0,$L32
        lw      $v1,optopt
        li      $v0,72                  # 0x48
        beq     $v1,$v0,$L32
        lw      $v1,optopt
        li      $v0,119                 # 0x77
        beq     $v1,$v0,$L32
        lw      $v1,optopt
        li      $v0,115                 # 0x73
        beq     $v1,$v0,$L32
        b       $L31
$L32:
        la      $a0,__sF+176
        la      $a1,$LC13
        lw      $a2,optopt
        la      $t9,fprintf
        jal     $ra,$t9
        b       $L36
$L31:
        lw      $v1,_ctype_
        lw      $v0,optopt
        addu    $v0,$v1,$v0
        addu    $v0,$v0,1
        lbu     $v0,0($v0)
        andi    $v0,$v0,0x97
        beq     $v0,$zero,$L34
        la      $a0,__sF+176
        la      $a1,$LC14
        lw      $a2,optopt
        la      $t9,fprintf
        jal     $ra,$t9
        b       $L36
$L34:
        la      $a0,__sF+176
        la      $a1,$LC15
        lw      $a2,optopt
        la      $t9,fprintf
        jal     $ra,$t9
$L36:
        la      $t9,showHelp
        jal     $ra,$t9
        li      $a0,1                   # 0x1
        la      $t9,exit
        jal     $ra,$t9
$L19:
        lw      $v0,128($fp)
        addu    $v1,$fp,40
        move    $a0,$v0
        move    $a1,$v1
        li      $a2,64                  # 0x40
        la      $t9,memcpy
        jal     $ra,$t9
        lw      $v0,128($fp)
```

```
        move    $sp,$fp
        lw      $ra,120($sp)
        lw      $fp,116($sp)
        addu    $sp,$sp,128
        j       $ra
        .end    getParameters
        .size   getParameters, .-getParameters
        .align  2
        .globl  returnValidation
        .ent    returnValidation
returnValidation:
        .frame  $fp,40,$ra              # vars= 0, regs= 3/0, args= 16,
           extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,40
        .cprestore 16
        sw      $ra,32($sp)
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        sw      $a0,40($fp)
        lw      $v0,40($fp)
        bgez    $v0,$L38
        lw      $a0,40($fp)
        la      $t9,showError
        jal     $ra,$t9
        li      $a0,1                   # 0x1
        la      $t9,exit
        jal     $ra,$t9
$L38:
        move    $sp,$fp
        lw      $ra,32($sp)
        lw      $fp,28($sp)
        addu    $sp,$sp,40
        j       $ra
        .end    returnValidation
        .size   returnValidation, .-returnValidation
        .rdata
        .align  2
$LC16:
        .ascii  "%s\n\000"
        .align  2
$LC17:
        .ascii  "TP0 Organizacion de computadoras - HELP\000"
        .align  2
$LC18:
        .ascii  "Usage: \000"
        .align  2
$LC19:
        .ascii  "tp0 -h   Displays help and usage of the application\000"
        .align  2
$LC20:
        .ascii  "tp0 -V   Displays version of the application\000"
        .align  2
```

```
$LC21:
        .ascii  "tp0 Options:\000"
        .align  2
$LC22:
        .ascii  "-r --resolution  Set bitmap resolution to WxH pixels\000"
        .align  2
$LC23:
        .ascii  "-c --center      Set the center of the image\000"
        .align  2
$LC24:
        .ascii  "-w --width       Set the width of the region to be spann"
        .ascii  "ed\000"
        .align  2
$LC25:
        .ascii  "-H --height      Set the height of the region to be span"
        .ascii  "ned\000"
        .align  2
$LC26:
        .ascii  "-o --output      Set path to output file\000"
        .align  2
$LC27:
        .ascii  "Examples: \000"
        .align  2
$LC28:
        .ascii  "  tp0 -r 1024x768 -o example.pgm\000"
        .align  2
$LC29:
        .ascii  "  tp0 -c 0.282-0.007i -w 0.005 -H 0.005 -o example.pgm
           \000"
        .text
        .align  2
        .globl  showHelp
        .ent    showHelp
showHelp:
        .frame  $fp,40,$ra              # vars= 0, regs= 3/0, args= 16,
           extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,40
        .cprestore 16
        sw      $ra,32($sp)
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        la      $a0,$LC16
        la      $a1,$LC17
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC16
        la      $a1,$LC18
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC16
        la      $a1,$LC19
        la      $t9,printf
```

```
        jal     $ra,$t9
        la      $a0,$LC16
        la      $a1,$LC20
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC16
        la      $a1,$LC21
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC16
        la      $a1,$LC22
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC16
        la      $a1,$LC23
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC16
        la      $a1,$LC24
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC16
        la      $a1,$LC25
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC16
        la      $a1,$LC26
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC16
        la      $a1,$LC27
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC16
        la      $a1,$LC28
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC16
        la      $a1,$LC29
        la      $t9,printf
        jal     $ra,$t9
        move    $sp,$fp
        lw      $ra,32($sp)
        lw      $fp,28($sp)
        addu    $sp,$sp,40
        j       $ra
        .end    showHelp
        .size   showHelp, .-showHelp
        .rdata
        .align  2
$LC30:
        .ascii  "TP0 Organizacion de computadoras - VERSION: 1.0\000"
        .text
        .align  2
        .globl  showVersion
        .ent    showVersion
showVersion:
```

```
        .frame  $fp,40,$ra              # vars= 0, regs= 3/0, args= 16,
           extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,40
        .cprestore 16
        sw      $ra,32($sp)
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        la      $a0,$LC16
        la      $a1,$LC30
        la      $t9,printf
        jal     $ra,$t9
        move    $sp,$fp
        lw      $ra,32($sp)
        lw      $fp,28($sp)
        addu    $sp,$sp,40
        j       $ra
        .end    showVersion
        .size   showVersion, .-showVersion
        .rdata
        .align  2
$LC31:
        .ascii  "Parameter input is empty. Type 'tp0 -h' for help. Progra"
        .ascii  "m terminated\000"
        .align  2
$LC32:
        .ascii  "Argument has invalid characters. Type 'tp0 -h' for help."
        .ascii  " Program terminated\000"
        .align  2
$LC33:
        .ascii  "Invalid arguments. Type 'tp0 -h' for help. Program termi"
        .ascii  "nated\000"
        .align  2
$LC34:
        .ascii  "Parameter has no real part. Type 'tp0 -h' for help. Prog"
        .ascii  "ram terminated\000"
        .align  2
$LC35:
        .ascii  "Parameter has no immaginary part. Type 'tp0 -h' for help"
        .ascii  ". Program terminated\000"
        .align  2
$LC36:
        .ascii  "Resolution input is invalid. Type 'tp0 -h' for help. Pro"
        .ascii  "gram terminated\000"
        .align  2
$LC37:
        .ascii  "Output file type is invalid. Type 'tp0 -h' for help. Pro"
        .ascii  "gram terminated\000"
        .align  2
$LC38:
        .ascii  "Output file path is invalid. Type 'tp0 -h' for help. Pro"
        .ascii  "gram terminated\000"
        .align  2
```

```
$LC39:
        .ascii  "Parameter input is invalid. Type 'tp0 -h' for help. Prog"
        .ascii  "ram terminated\000"
        .text
        .align  2
        .globl  showError
        .ent    showError
showError:
        .frame  $fp,40,$ra              # vars= 0, regs= 3/0, args= 16,
           extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,40
        .cprestore 16
        sw      $ra,32($sp)
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        sw      $a0,40($fp)
        lw      $v1,40($fp)
        li      $v0,-1                  # 0xffffffffffffffff
        bne     $v1,$v0,$L43
        la      $a0,__sF+176
        la      $a1,$LC16
        la      $a2,$LC31
        la      $t9,fprintf
        jal     $ra,$t9
$L43:
        lw      $v1,40($fp)
        li      $v0,-2                  # 0xfffffffffffffffe
        bne     $v1,$v0,$L44
        la      $a0,__sF+176
        la      $a1,$LC16
        la      $a2,$LC32
        la      $t9,fprintf
        jal     $ra,$t9
$L44:
        lw      $v1,40($fp)
        li      $v0,-3                  # 0xfffffffffffffffd
        bne     $v1,$v0,$L45
        la      $a0,__sF+176
        la      $a1,$LC16
        la      $a2,$LC33
        la      $t9,fprintf
        jal     $ra,$t9
$L45:
        lw      $v1,40($fp)
        li      $v0,-4                  # 0xfffffffffffffffc
        bne     $v1,$v0,$L46
        la      $a0,__sF+176
        la      $a1,$LC16
        la      $a2,$LC34
        la      $t9,fprintf
        jal     $ra,$t9
$L46:
```

```
        lw      $v1,40($fp)
        li      $v0,-5                  # 0xfffffffffffffffb
        bne     $v1,$v0,$L47
        la      $a0,__sF+176
        la      $a1,$LC16
        la      $a2,$LC35
        la      $t9,fprintf
        jal     $ra,$t9
$L47:
        lw      $v1,40($fp)
        li      $v0,-6                  # 0xfffffffffffffffa
        bne     $v1,$v0,$L48
        la      $a0,__sF+176
        la      $a1,$LC16
        la      $a2,$LC36
        la      $t9,fprintf
        jal     $ra,$t9
$L48:
        lw      $v1,40($fp)
        li      $v0,-7                  # 0xfffffffffffffff9
        bne     $v1,$v0,$L49
        la      $a0,__sF+176
        la      $a1,$LC16
        la      $a2,$LC37
        la      $t9,fprintf
        jal     $ra,$t9
$L49:
        lw      $v1,40($fp)
        li      $v0,-8                  # 0xfffffffffffffff8
        bne     $v1,$v0,$L50
        la      $a0,__sF+176
        la      $a1,$LC16
        la      $a2,$LC38
        la      $t9,fprintf
        jal     $ra,$t9
$L50:
        lw      $v1,40($fp)
        li      $v0,-9                  # 0xfffffffffffffff7
        bne     $v1,$v0,$L42
        la      $a0,__sF+176
        la      $a1,$LC16
        la      $a2,$LC39
        la      $t9,fprintf
        jal     $ra,$t9
$L42:
        move    $sp,$fp
        lw      $ra,32($sp)
        lw      $fp,28($sp)
        addu    $sp,$sp,40
        j       $ra
        .end    showError
        .size   showError, .-showError
        .globl  memcpy
        .rdata
        .align  2
$LC40:
        .ascii  "w\000"
        .align  2
```

```
$LC43:
        .ascii  "%d \000"
        .align  2
$LC44:
        .ascii  "There was an error while writing to output.\n\000"
        .align  2
$LC45:
        .ascii  "\n\000"
        .align  2
$LC46:
        .ascii  "Unable to close output file.\n\000"
        .align  2
$LC41:
        .word   1073741824
        .align  3
$LC42:
        .word   0
        .word   1073741824
        .text
        .align  2
        .globl  julia
        .ent    julia
julia:
        .frame  $fp,144,$ra             # vars= 56, regs= 3/0, args= 64,
           extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,144
        .cprestore 64
        sw      $ra,136($sp)
        sw      $fp,132($sp)
        sw      $gp,128($sp)
        move    $fp,$sp
        sw      $a0,144($fp)
        sw      $a1,148($fp)
        sw      $a2,152($fp)
        sw      $a3,156($fp)
        lw      $a0,148($fp)
        la      $a1,$LC40
        la      $t9,fopen
        jal     $ra,$t9
        sw      $v0,144($fp)
        addu    $v0,$sp,16
        addu    $v1,$fp,160
        move    $a0,$v0
        move    $a1,$v1
        li      $a2,48                  # 0x30
        la      $t9,memcpy
        jal     $ra,$t9
        lw      $a0,144($fp)
        lw      $a1,148($fp)
        lw      $a2,152($fp)
        lw      $a3,156($fp)
        la      $t9,writeHeader
        jal     $ra,$t9
```

```
        l.s     $f0,152($fp)
        cvt.s.w $f2,$f0
        l.s     $f0,160($fp)
        div.s   $f0,$f0,$f2
        s.s     $f0,72($fp)
        l.s     $f0,156($fp)
        cvt.s.w $f2,$f0
        l.s     $f0,164($fp)
        div.s   $f0,$f0,$f2
        s.s     $f0,76($fp)
        l.s     $f2,160($fp)
        l.s     $f0,$LC41
        div.s   $f0,$f2,$f0
        cvt.d.s $f2,$f0
        l.d     $f0,168($fp)
        sub.d   $f0,$f0,$f2
        s.d     $f0,96($fp)
        l.s     $f2,164($fp)
        l.s     $f0,$LC41
        div.s   $f0,$f2,$f0
        cvt.d.s $f2,$f0
        l.d     $f0,176($fp)
        add.d   $f0,$f0,$f2
        s.d     $f0,104($fp)
        lw      $v0,200($fp)
        sw      $v0,112($fp)
        lw      $v0,204($fp)
        sw      $v0,116($fp)
        sw      $zero,120($fp)
$L53:
        lw      $v0,120($fp)
        lw      $v1,156($fp)
        slt     $v0,$v0,$v1
        bne     $v0,$zero,$L56
        b       $L54
$L56:
        sw      $zero,124($fp)
$L57:
        lw      $v0,124($fp)
        lw      $v1,152($fp)
        slt     $v0,$v0,$v1
        bne     $v0,$zero,$L60
        b       $L58
$L60:
        l.s     $f0,124($fp)
        cvt.s.w $f2,$f0
        l.s     $f0,72($fp)
        mul.s   $f0,$f2,$f0
        cvt.d.s $f2,$f0
        l.d     $f0,96($fp)
        add.d   $f0,$f0,$f2
        s.d     $f0,80($fp)
        l.s     $f0,120($fp)
        cvt.s.w $f2,$f0
        l.s     $f0,76($fp)
        mul.s   $f0,$f2,$f0
        cvt.d.s $f2,$f0
        l.d     $f0,104($fp)
```

```
        sub.d   $f0,$f0,$f2
        s.d     $f0,88($fp)
        sw      $zero,116($fp)
$L61:
        lw      $v0,116($fp)
        lw      $v1,112($fp)
        slt     $v0,$v0,$v1
        bne     $v0,$zero,$L64
        b       $L62
$L64:
        lw      $a0,80($fp)
        lw      $a1,84($fp)
        lw      $a2,88($fp)
        lw      $a3,92($fp)
        la      $t9,getNroImgAbs
        jal     $ra,$t9
        mov.d   $f2,$f0
        l.d     $f0,$LC42
        c.lt.d  $f0,$f2
        bc1t    $L62
        addu    $v1,$fp,80
        lw      $v0,88($fp)
        sw      $v0,16($sp)
        lw      $v0,92($fp)
        sw      $v0,20($sp)
        lw      $a2,80($fp)
        lw      $a3,84($fp)
        move    $a0,$v1
        la      $t9,getNroImgSq
        jal     $ra,$t9
        l.d     $f2,80($fp)
        l.d     $f0,184($fp)
        add.d   $f0,$f2,$f0
        s.d     $f0,80($fp)
        l.d     $f2,88($fp)
        l.d     $f0,192($fp)
        add.d   $f0,$f2,$f0
        s.d     $f0,88($fp)
        lw      $v0,116($fp)
        addu    $v0,$v0,1
        sw      $v0,116($fp)
        b       $L61
$L62:
        lw      $a0,144($fp)
        la      $a1,$LC43
        lw      $a2,116($fp)
        la      $t9,fprintf
        jal     $ra,$t9
        bgez    $v0,$L59
        la      $a0,__sF+176
        la      $a1,$LC44
        la      $t9,fprintf
        jal     $ra,$t9
        li      $a0,1                   # 0x1
        la      $t9,exit
        jal     $ra,$t9
$L59:
        lw      $v0,124($fp)
```

```
        addu    $v0,$v0,1
        sw      $v0,124($fp)
        b       $L57
$L58:
        lw      $a0,144($fp)
        la      $a1,$LC45
        la      $t9,fprintf
        jal     $ra,$t9
        lw      $v0,120($fp)
        addu    $v0,$v0,1
        sw      $v0,120($fp)
        b       $L53
$L54:
        lw      $a0,144($fp)
        la      $t9,fclose
        jal     $ra,$t9
        beq     $v0,$zero,$L52
        la      $a0,__sF+176
        la      $a1,$LC46
        la      $t9,fprintf
        jal     $ra,$t9
        li      $a0,1                   # 0x1
        la      $t9,exit
        jal     $ra,$t9
$L52:
        move    $sp,$fp
        lw      $ra,136($sp)
        lw      $fp,132($sp)
        addu    $sp,$sp,144
        j       $ra
        .end    julia
        .size   julia, .-julia
        .rdata
        .align  2
$LC47:
        .ascii  "P2\n\000"
        .align  2
$LC48:
        .ascii  "# %s\n\000"
        .align  2
$LC49:
        .ascii  "%u\n\000"
        .text
        .align  2
        .globl  writeHeader
        .ent    writeHeader
writeHeader:
        .frame  $fp,40,$ra              # vars= 0, regs= 3/0, args= 16,
           extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,40
        .cprestore 16
        sw      $ra,32($sp)
        sw      $fp,28($sp)
```

```
        sw      $gp,24($sp)
        move    $fp,$sp
        sw      $a0,40($fp)
        sw      $a1,44($fp)
        sw      $a2,48($fp)
        sw      $a3,52($fp)
        lw      $a0,40($fp)
        la      $a1,$LC47
        la      $t9,fprintf
        jal     $ra,$t9
        lw      $a0,40($fp)
        la      $a1,$LC48
        lw      $a2,44($fp)
        la      $t9,fprintf
        jal     $ra,$t9
        lw      $a0,40($fp)
        la      $a1,$LC49
        lw      $a2,48($fp)
        la      $t9,fprintf
        jal     $ra,$t9
        lw      $a0,40($fp)
        la      $a1,$LC49
        lw      $a2,52($fp)
        la      $t9,fprintf
        jal     $ra,$t9
        lw      $a0,40($fp)
        la      $a1,$LC49
        lw      $a2,96($fp)
        la      $t9,fprintf
        jal     $ra,$t9
        move    $sp,$fp
        lw      $ra,32($sp)
        lw      $fp,28($sp)
        addu    $sp,$sp,40
        j       $ra
        .end    writeHeader
        .size   writeHeader, .-writeHeader
        .globl  memcpy
        .rdata
        .align  2
$LC50:
        .ascii  "RESOLUTION WIDTH: %d\n\000"
        .align  2
$LC51:
        .ascii  "RESOLUTION HEIGHT: %d\n\000"
        .align  2
$LC52:
        .ascii  "SEED: %.16Lf\n\000"
        .align  2
$LC53:
        .ascii  "WIDTH: %f\n\000"
        .align  2
$LC54:
        .ascii  "HEIGHT: %f\n\000"
        .align  2
$LC55:
        .ascii  "OUTPUT PATH: %s\n\000"
        .text
```

```
        .align  2
        .globl  main
        .ent    main
main:
        .frame  $fp,160,$ra             # vars= 72, regs= 3/0, args= 64,
           extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,160
        .cprestore 64
        sw      $ra,152($sp)
        sw      $fp,148($sp)
        sw      $gp,144($sp)
        move    $fp,$sp
        sw      $a0,160($fp)
        sw      $a1,164($fp)
        sw      $zero,72($fp)
        addu    $v0,$fp,80
        move    $a0,$v0
        lw      $a1,160($fp)
        lw      $a2,164($fp)
        la      $t9,getParameters
        jal     $ra,$t9
        la      $a0,$LC50
        lw      $a1,88($fp)
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC51
        lw      $a1,92($fp)
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC52
        lw      $a2,120($fp)
        lw      $a3,124($fp)
        la      $t9,printf
        jal     $ra,$t9
        la      $a0,$LC52
        lw      $a2,128($fp)
        lw      $a3,132($fp)
        la      $t9,printf
        jal     $ra,$t9
        l.s     $f0,96($fp)
        cvt.d.s $f0,$f0
        la      $a0,$LC53
        mfc1    $a2,$f0
        mfc1    $a3,$f1
        la      $t9,printf
        jal     $ra,$t9
        l.s     $f0,100($fp)
        cvt.d.s $f0,$f0
        la      $a0,$LC54
        mfc1    $a2,$f0
        mfc1    $a3,$f1
        la      $t9,printf
        jal     $ra,$t9
```

```
        la      $a0,$LC55
        lw      $a1,84($fp)
        la      $t9,printf
        jal     $ra,$t9
        addu    $v0,$sp,16
        addu    $v1,$fp,96
        move    $a0,$v0
        move    $a1,$v1
        li      $a2,48                  # 0x30
        la      $t9,memcpy
        jal     $ra,$t9
        lw      $a0,80($fp)
        lw      $a1,84($fp)
        lw      $a2,88($fp)
        lw      $a3,92($fp)
        la      $t9,julia
        jal     $ra,$t9
        lw      $v0,72($fp)
        move    $sp,$fp
        lw      $ra,152($sp)
        lw      $fp,148($sp)
        addu    $sp,$sp,160
        j       $ra
        .end    main
        .size   main, .-main
        .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"


                .file   1 "funciones.c"
        .section .mdebug.abi32
        .previous
        .abicalls
        .rdata
        .align  2
$LC0:
        .ascii  "funciones.c\000"
        .align  2
$LC1:
        .ascii  "str_split\000"
        .align  2
$LC2:
        .ascii  "idx < count\000"
        .align  2
$LC3:
        .ascii  "idx == count - 1\000"
        .text
        .align  2
        .globl  str_split
        .ent    str_split
str_split:
        .frame  $fp,80,$ra              # vars= 40, regs= 3/0, args= 16,
           extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,80
```

```
        .cprestore 16
        sw      $ra,72($sp)
        sw      $fp,68($sp)
        sw      $gp,64($sp)
        move    $fp,$sp
        sw      $a0,80($fp)
        move    $v0,$a1
        sb      $v0,24($fp)
        sw      $zero,28($fp)
        sw      $zero,32($fp)
        lw      $v0,80($fp)
        sw      $v0,36($fp)
        sw      $zero,40($fp)
        lbu     $v0,24($fp)
        sb      $v0,48($fp)
        sb      $zero,49($fp)
$L18:
        lw      $v0,36($fp)
        lb      $v0,0($v0)
        bne     $v0,$zero,$L20
        b       $L19
$L20:
        lw      $v0,36($fp)
        lb      $v1,24($fp)
        lb      $v0,0($v0)
        bne     $v1,$v0,$L21
        lw      $v0,32($fp)
        addu    $v0,$v0,1
        sw      $v0,32($fp)
        lw      $v0,36($fp)
        sw      $v0,40($fp)
$L21:
        lw      $v0,36($fp)
        addu    $v0,$v0,1
        sw      $v0,36($fp)
        b       $L18
$L19:
        lw      $a0,80($fp)
        la      $t9,strlen
        jal     $ra,$t9
        move    $v1,$v0
        lw      $v0,80($fp)
        addu    $v0,$v1,$v0
        addu    $v1,$v0,-1
        lw      $v0,40($fp)
        sltu    $v0,$v0,$v1
        lw      $v1,32($fp)
        addu    $v0,$v0,$v1
        sw      $v0,32($fp)
        lw      $v0,32($fp)
        addu    $v0,$v0,1
        sw      $v0,32($fp)
        lw      $v0,32($fp)
        sll     $v0,$v0,2
        move    $a0,$v0
        la      $t9,malloc
        jal     $ra,$t9
        sw      $v0,28($fp)
```

```
        lw      $v0,28($fp)
        beq     $v0,$zero,$L22
        sw      $zero,56($fp)
        addu    $v0,$fp,48
        lw      $a0,80($fp)
        move    $a1,$v0
        la      $t9,strtok
        jal     $ra,$t9
        sw      $v0,60($fp)
$L23:
        lw      $v0,60($fp)
        bne     $v0,$zero,$L25
        b       $L24
$L25:
        lw      $v0,56($fp)
        lw      $v1,32($fp)
        sltu    $v0,$v0,$v1
        bne     $v0,$zero,$L27
        la      $a0,$LC0
        li      $a1,55                  # 0x37
        la      $a2,$LC1
        la      $a3,$LC2
        la      $t9,__assert13
        jal     $ra,$t9
$L27:
        lw      $a0,60($fp)
        la      $t9,strdup
        jal     $ra,$t9
        move    $a2,$v0
        addu    $a1,$fp,56
        lw      $v1,0($a1)
        move    $v0,$v1
        sll     $a0,$v0,2
        lw      $v0,28($fp)
        addu    $v0,$a0,$v0
        sw      $a2,0($v0)
        addu    $v1,$v1,1
        sw      $v1,0($a1)
        addu    $v0,$fp,48
        move    $a0,$zero
        move    $a1,$v0
        la      $t9,strtok
        jal     $ra,$t9
        sw      $v0,60($fp)
        b       $L23
$L24:
        lw      $v0,32($fp)
        addu    $v1,$v0,-1
        lw      $v0,56($fp)
        beq     $v0,$v1,$L29
        la      $a0,$LC0
        li      $a1,59                  # 0x3b
        la      $a2,$LC1
        la      $a3,$LC3
        la      $t9,__assert13
        jal     $ra,$t9
$L29:
        lw      $v0,56($fp)
```

```
        sll     $v1,$v0,2
        lw      $v0,28($fp)
        addu    $v0,$v1,$v0
        sw      $zero,0($v0)
$L22:
        lw      $v0,28($fp)
        move    $sp,$fp
        lw      $ra,72($sp)
        lw      $fp,68($sp)
        addu    $sp,$sp,80
        j       $ra
        .end    str_split
        .size   str_split, .-str_split
        .align  2
        .globl  stringContainsChar
        .ent    stringContainsChar
stringContainsChar:
        .frame  $fp,56,$ra              # vars= 16, regs= 3/0, args= 16,
           extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,56
        .cprestore 16
        sw      $ra,48($sp)
        sw      $fp,44($sp)
        sw      $gp,40($sp)
        move    $fp,$sp
        sw      $a0,56($fp)
        sw      $a1,60($fp)
        lw      $a0,56($fp)
        la      $t9,strlen
        jal     $ra,$t9
        sw      $v0,24($fp)
        lw      $a0,56($fp)
        lw      $a1,60($fp)
        la      $t9,strcspn
        jal     $ra,$t9
        sw      $v0,28($fp)
        lw      $v0,28($fp)
        bne     $v0,$zero,$L31
        li      $v0,1                   # 0x1
        sw      $v0,32($fp)
        b       $L30
$L31:
        lw      $v1,24($fp)
        lw      $v0,28($fp)
        beq     $v1,$v0,$L32
        sw      $zero,32($fp)
        b       $L30
$L32:
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,32($fp)
$L30:
        lw      $v0,32($fp)
        move    $sp,$fp
```

```
        lw      $ra,48($sp)
        lw      $fp,44($sp)
        addu    $sp,$sp,56
        j       $ra
        .end    stringContainsChar
        .size   stringContainsChar, .-stringContainsChar
        .rdata
        .align  2
$LC4:
        .ascii  "\000"
        .align  2
$LC5:
        .ascii  "0123456789.+-\000"
        .align  2
$LC6:
        .ascii  " \000"
        .align  2
$LC7:
        .ascii  "%Lf%Lf%c\000"
        .text
        .align  2
        .globl  parseNroImg
        .ent    parseNroImg
parseNroImg:
        .frame  $fp,72,$ra              # vars= 24, regs= 3/0, args= 24,
           extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,72
        .cprestore 24
        sw      $ra,64($sp)
        sw      $fp,60($sp)
        sw      $gp,56($sp)
        move    $fp,$sp
        sw      $a0,72($fp)
        sw      $a1,76($fp)
        lw      $a0,72($fp)
        la      $a1,$LC4
        la      $t9,strcmp
        jal     $ra,$t9
        beq     $v0,$zero,$L35
        lw      $v0,72($fp)
        bne     $v0,$zero,$L34
$L35:
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,52($fp)
        b       $L33
$L34:
        lw      $a0,72($fp)
        la      $a1,$LC5
        la      $t9,stringContainsChar
        jal     $ra,$t9
        move    $v1,$v0
        li      $v0,-1                  # 0xffffffffffffffff
        beq     $v1,$v0,$L37
```

```
        lw      $a0,72($fp)
        la      $a1,$LC6
        la      $t9,stringContainsChar
        jal     $ra,$t9
        bne     $v0,$zero,$L36
$L37:
        li      $v0,-2                  # 0xfffffffffffffffe
        sw      $v0,52($fp)
        b       $L33
$L36:
        addu    $v1,$fp,40
        addu    $v0,$fp,48
        sw      $v0,16($sp)
        lw      $a0,72($fp)
        la      $a1,$LC7
        addu    $a2,$fp,32
        move    $a3,$v1
        la      $t9,sscanf
        jal     $ra,$t9
        move    $v1,$v0
        li      $v0,3                   # 0x3
        bne     $v1,$v0,$L39
        lb      $v1,48($fp)
        li      $v0,105                 # 0x69
        bne     $v1,$v0,$L39
        b       $L38
$L39:
        li      $v0,-3                  # 0xfffffffffffffffd
        sw      $v0,52($fp)
        b       $L33
$L38:
        lw      $v0,76($fp)
        l.d     $f0,32($fp)
        s.d     $f0,0($v0)
        lw      $v0,76($fp)
        l.d     $f0,40($fp)
        s.d     $f0,8($v0)
        sw      $zero,52($fp)
$L33:
        lw      $v0,52($fp)
        move    $sp,$fp
        lw      $ra,64($sp)
        lw      $fp,60($sp)
        addu    $sp,$sp,72
        j       $ra
        .end    parseNroImg
        .size   parseNroImg, .-parseNroImg
        .rdata
        .align  2
$LC8:
        .ascii  "%d%c%d %c\000"
        .text
        .align  2
        .globl  parseResolution
        .ent    parseResolution
parseResolution:
        .frame  $fp,64,$ra              # vars= 16, regs= 3/0, args= 24,
           extra= 8
```

```
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,64
        .cprestore 24
        sw      $ra,56($sp)
        sw      $fp,52($sp)
        sw      $gp,48($sp)
        move    $fp,$sp
        sw      $a0,64($fp)
        sw      $a1,68($fp)
        addu    $v1,$fp,40
        addu    $v0,$fp,36
        sw      $v0,16($sp)
        addu    $v0,$fp,41
        sw      $v0,20($sp)
        lw      $a0,64($fp)
        la      $a1,$LC8
        addu    $a2,$fp,32
        move    $a3,$v1
        la      $t9,sscanf
        jal     $ra,$t9
        move    $v1,$v0
        li      $v0,3                   # 0x3
        bne     $v1,$v0,$L42
        lw      $v0,32($fp)
        blez    $v0,$L42
        lb      $v1,40($fp)
        li      $v0,120                 # 0x78
        bne     $v1,$v0,$L42
        lw      $v0,36($fp)
        blez    $v0,$L42
        b       $L41
$L42:
        li      $v0,-6                  # 0xfffffffffffffffa
        sw      $v0,44($fp)
        b       $L40
$L41:
        lw      $v1,68($fp)
        lw      $v0,32($fp)
        sw      $v0,0($v1)
        lw      $v1,68($fp)
        lw      $v0,36($fp)
        sw      $v0,4($v1)
        sw      $zero,44($fp)
$L40:
        lw      $v0,44($fp)
        move    $sp,$fp
        lw      $ra,56($sp)
        lw      $fp,52($sp)
        addu    $sp,$sp,64
        j       $ra
        .end    parseResolution
        .size   parseResolution, .-parseResolution
        .rdata
        .align  2
```

```
$LC9:
        .ascii  ".pgm\000"
        .align  2
$LC10:
        .ascii  "-\000"
        .align  2
$LC11:
        .ascii  "w\000"
        .text
        .align  2
        .globl  checkForOutputPath
        .ent    checkForOutputPath
checkForOutputPath:
        .frame  $fp,48,$ra              # vars= 8, regs= 3/0, args= 16,
           extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,48
        .cprestore 16
        sw      $ra,40($sp)
        sw      $fp,36($sp)
        sw      $gp,32($sp)
        move    $fp,$sp
        sw      $a0,48($fp)
        sw      $a1,52($fp)
        sw      $a2,56($fp)
        lw      $v0,48($fp)
        bne     $v0,$zero,$L44
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,24($fp)
        b       $L43
$L44:
        lw      $a0,48($fp)
        la      $t9,checkForBadCharacters
        jal     $ra,$t9
        move    $v1,$v0
        li      $v0,-1                  # 0xffffffffffffffff
        bne     $v1,$v0,$L45
        li      $v0,-2                  # 0xfffffffffffffffe
        sw      $v0,24($fp)
        b       $L43
$L45:
        lw      $a0,48($fp)
        la      $t9,getFileExtension
        jal     $ra,$t9
        move    $a0,$v0
        la      $a1,$LC9
        la      $t9,strcmp
        jal     $ra,$t9
        beq     $v0,$zero,$L46
        li      $v0,-7                  # 0xfffffffffffffff9
        sw      $v0,24($fp)
        b       $L43
$L46:
        lw      $a0,48($fp)
```

```
        la      $a1,$LC10
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L47
        la      $v0,__sF+88
        sw      $v0,52($fp)
        b       $L48
$L47:
        lw      $a0,48($fp)
        la      $a1,$LC11
        la      $t9,fopen
        jal     $ra,$t9
        sw      $v0,52($fp)
        lw      $v0,52($fp)
        bne     $v0,$zero,$L48
        li      $v0,-8                  # 0xfffffffffffffff8
        sw      $v0,24($fp)
        b       $L43
$L48:
        lw      $v1,56($fp)
        lw      $v0,48($fp)
        sw      $v0,0($v1)
        sw      $zero,24($fp)
$L43:
        lw      $v0,24($fp)
        move    $sp,$fp
        lw      $ra,40($sp)
        lw      $fp,36($sp)
        addu    $sp,$sp,48
        j       $ra
        .end    checkForOutputPath
        .size   checkForOutputPath, .-checkForOutputPath
        .align  2
        .globl  getFileExtension
        .ent    getFileExtension
getFileExtension:
        .frame  $fp,48,$ra              # vars= 8, regs= 3/0, args= 16,
           extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,48
        .cprestore 16
        sw      $ra,40($sp)
        sw      $fp,36($sp)
        sw      $gp,32($sp)
        move    $fp,$sp
        sw      $a0,48($fp)
        lw      $a0,48($fp)
        li      $a1,46                  # 0x2e
        la      $t9,strrchr
        jal     $ra,$t9
        sw      $v0,24($fp)
        lw      $v0,24($fp)
        beq     $v0,$zero,$L53
        lw      $v1,24($fp)
```

```
        lw      $v0,48($fp)
        beq     $v1,$v0,$L53
        b       $L51
$L53:
        lw      $a0,48($fp)
        la      $t9,strlen
        jal     $ra,$t9
        move    $v1,$v0
        lw      $v0,48($fp)
        addu    $v1,$v1,$v0
        sw      $v1,28($fp)
        b       $L52
$L51:
        lw      $v0,24($fp)
        sw      $v0,28($fp)
$L52:
        lw      $v0,28($fp)
        move    $sp,$fp
        lw      $ra,40($sp)
        lw      $fp,36($sp)
        addu    $sp,$sp,48
        j       $ra
        .end    getFileExtension
        .size   getFileExtension, .-getFileExtension
        .rdata
        .align  2
$LC12:
        .ascii  "!@%^*~|\000"
        .text
        .align  2
        .globl  checkForBadCharacters
        .ent    checkForBadCharacters
checkForBadCharacters:
        .frame  $fp,56,$ra              # vars= 16, regs= 3/0, args= 16,
           extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,56
        .cprestore 16
        sw      $ra,48($sp)
        sw      $fp,44($sp)
        sw      $gp,40($sp)
        move    $fp,$sp
        sw      $a0,56($fp)
        lw      $v0,$LC12
        sw      $v0,24($fp)
        lw      $v0,$LC12+4
        sw      $v0,28($fp)
        sw      $zero,32($fp)
$L55:
        addu    $a0,$fp,24
        la      $t9,strlen
        jal     $ra,$t9
        lw      $v1,32($fp)
        sltu    $v0,$v1,$v0
```

```
        bne     $v0,$zero,$L58
        b       $L56
$L58:
        lw      $v1,32($fp)
        addu    $v0,$fp,24
        addu    $v0,$v0,$v1
        lb      $v0,0($v0)
        lw      $a0,56($fp)
        move    $a1,$v0
        la      $t9,strchr
        jal     $ra,$t9
        beq     $v0,$zero,$L57
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,36($fp)
        b       $L54
$L57:
        lw      $v0,32($fp)
        addu    $v0,$v0,1
        sw      $v0,32($fp)
        b       $L55
$L56:
        sw      $zero,36($fp)
$L54:
        lw      $v0,36($fp)
        move    $sp,$fp
        lw      $ra,48($sp)
        lw      $fp,44($sp)
        addu    $sp,$sp,56
        j       $ra
        .end    checkForBadCharacters
        .size   checkForBadCharacters, .-checkForBadCharacters
        .rdata
        .align  2
$LC13:
        .ascii  "%f\000"
        .text
        .align  2
        .globl  setValue
        .ent    setValue
setValue:
        .frame  $fp,48,$ra              # vars= 8, regs= 3/0, args= 16,
           extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,48
        .cprestore 16
        sw      $ra,40($sp)
        sw      $fp,36($sp)
        sw      $gp,32($sp)
        move    $fp,$sp
        sw      $a0,48($fp)
        sw      $a1,52($fp)
        lw      $a0,48($fp)
        la      $a1,$LC13
        addu    $a2,$fp,24
```

```
        la      $t9,sscanf
        jal     $ra,$t9
        move    $v1,$v0
        li      $v0,1                   # 0x1
        bne     $v1,$v0,$L62
        l.s     $f2,24($fp)
        mtc1    $zero,$f0
        c.lt.s  $f2,$f0
        bc1t    $L62
        b       $L61
$L62:
        li      $v0,-9                  # 0xfffffffffffffff7
        sw      $v0,28($fp)
        b       $L60
$L61:
        lw      $v0,52($fp)
        l.s     $f0,24($fp)
        s.s     $f0,0($v0)
        sw      $zero,28($fp)
$L60:
        lw      $v0,28($fp)
        move    $sp,$fp
        lw      $ra,40($sp)
        lw      $fp,36($sp)
        addu    $sp,$sp,48
        j       $ra
        .end    setValue
        .size   setValue, .-setValue
        .align  2
        .globl  getNroImgAbs
        .ent    getNroImgAbs
getNroImgAbs:
        .frame  $fp,40,$ra              # vars= 0, regs= 3/0, args= 16,
            extra= 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,40
        .cprestore 16
        sw      $ra,32($sp)
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        sw      $a0,40($fp)
        sw      $a1,44($fp)
        sw      $a2,48($fp)
        sw      $a3,52($fp)
        l.d     $f2,40($fp)
        l.d     $f0,40($fp)
        mul.d   $f4,$f2,$f0
        l.d     $f2,48($fp)
        l.d     $f0,48($fp)
        mul.d   $f0,$f2,$f0
        add.d   $f0,$f4,$f0
        mov.d   $f12,$f0
        la      $t9,sqrt
```

```
        jal     $ra,$t9
        move    $sp,$fp
        lw      $ra,32($sp)
        lw      $fp,28($sp)
        addu    $sp,$sp,40
        j       $ra
        .end    getNroImgAbs
        .size   getNroImgAbs, .-getNroImgAbs
        .align  2
        .globl  getNroImgSq
        .ent    getNroImgSq
getNroImgSq:
        .frame  $fp,32,$ra              # vars= 16, regs= 2/0, args= 0,
           extra= 8
        .mask   0x50000000,-4
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,32
        .cprestore 0
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        move    $v1,$a0
        sw      $a2,40($fp)
        sw      $a3,44($fp)
        l.d     $f2,40($fp)
        l.d     $f0,40($fp)
        mul.d   $f4,$f2,$f0
        l.d     $f2,48($fp)
        l.d     $f0,48($fp)
        mul.d   $f0,$f2,$f0
        sub.d   $f0,$f4,$f0
        s.d     $f0,8($fp)
        l.d     $f0,40($fp)
        add.d   $f2,$f0,$f0
        l.d     $f0,48($fp)
        mul.d   $f0,$f2,$f0
        s.d     $f0,16($fp)
        lw      $v0,8($fp)
        sw      $v0,0($v1)
        lw      $v0,12($fp)
        sw      $v0,4($v1)
        lw      $v0,16($fp)
        sw      $v0,8($v1)
        lw      $v0,20($fp)
        sw      $v0,12($v1)
        move    $v0,$v1
        move    $sp,$fp
        lw      $fp,28($sp)
        addu    $sp,$sp,32
        j       $ra
        .end    getNroImgSq
        .size   getNroImgSq, .-getNroImgSq
        .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

4. Conclusiones

La realizacion del presente trabajo practico nos sirvio para poder familiarizarnos con las herramientas y entornos con los que trabajaremos a lo largo de la materia. Ademas, pudimos mejorar nuestros conocimientos en el lenguaje de programacion C y nuestra interaccion con entornos UNIX. Tambien, pudimos ver como obtener el codigo assembly de nuestro programa, y tener un primer acercamiento a un lenguaje de bajo nivel.

5. Referencias

Referencias

[1] GXemul, http://gavare.se/gxemul/

[2] The NetBSD project, http://www.netbsd.org/

[3] GCC, https://gcc.gnu.org/onlinedocs/gcc-3.3.6/gcc/

[4] Debugging with GDB, https://sourceware.org/gdb/onlinedocs/gdb/

[5] Librería Standard de C, https://www.tutorialspoint.com/c_standard_library/

[6] Documentación de C en Linux, https://linux.die.net/man/

[7] Conjunto de Julia (Wikipedia), http://es.wikipedia.org/wiki/Conjunto_de_Julia

[8] Conjunto de Julia (Mathworld), http://mathworld.wolfram.com/JuliaSet.html

[9] PGM format specification, http://netpbm.sourceforge.net/doc/pgm.html

[10] Julia Set Generator, http://www.easyfractalgenerator.com/julia-set-generator.aspx

6. Anexo

6.1. Enunciado provisto por los profesores del curso

Univesidad de Buenos Aires - FIUBA
66:20 Organización de Computadoras
Trabajo práctico 0: Infraestructura básica
1^{er} cuatrimestre de 2018

`$Date: 2014/09/07 14:40:52 $`

1. Objetivos

Familiarizarse con las herramientas de software que usaremos en los siguientes trabajos, implementando un programa y su correspondiente documentación que resuelvan el problema descripto más abajo.

2. Alcance

Este trabajo práctico es de elaboración grupal, evaluación individual, y de carácter obligatorio para todos alumnos del curso.

3. Requisitos

El trabajo deberá ser entregado personalmente, en la fecha estipulada, con una carátula que contenga los datos completos de todos los integrantes, un informe impreso de acuerdo con lo que mencionaremos en la sección 6, y con una copia digital de los archivos fuente necesarios para compilar el trabajo.

4. Recursos

Usaremos el programa GXemul [1] para simular el entorno de desarrollo que utilizaremos en este y otros trabajos prácticos, una máquina MIPS corriendo una versión reciente del sistema operativo NetBSD [2].

Durante la primera clase del curso presentaremos brevemente los pasos necesarios para la instalación y configuración del entorno de desarrollo.

5. Programa

Se trata de un diseñar un programa que permita dibujar el conjunto de Julia [3] [4] y sus vecindades, en lenguaje C.

El mismo recibirá, por línea de comando, una serie de parámetros describiendo la región del plano complejo y las características del archivo imagen a generar. No deberá interactuar con el usuario, ya que no se trata de un programa interactivo, sino más bien de una herramienta de procesamiento *batch*. Al finalizar la ejecución, y volver al sistema operativo, el programa habrá dibujado el fractal en el archivo de salida.

El formato gráfico a usar es PGM o *portable gray map* [6], un formato simple para describir imágenes digitales monocromáticas.

5.1. Algoritmo

El algoritmo básico es simple: para algunos puntos f_0 de la región del plano que estamos procesando haremos un cálculo repetitivo. Terminado el cálculo, asignamos el nivel de intensidad del pixel en base a la condición de corte de ese cálculo.

El color de cada punto representa la "velocidad de escape" asociada con ese número complejo: blanco para aquellos puntos que pertenecen al conjunto (y por ende la "cuenta" permanece acotada), y tonos gradualmente más oscuros para los puntos divergentes, que no pertenezcan al conjunto.

Más específicamente: para cada pixel de la pantalla, tomaremos su punto medio, expresado en coordenadas complejas, $f_0 = \mathrm{Re}(f_0) + \mathrm{Im}(f_0)i$. A continuación, iteramos sobre $f_{i+1}(c) = f_i(c)^2 + s$, cortando la iteración cuando $|f_i(c)| > 2$, o después de `N` iteraciones.

En pseudo código:

```
para cada pixel $p {
        $f = complejo asociado a $p;
        for ($i = 0; $i < $N - 1; ++$i) {
                if (abs($f) > 2)
                        break;
                $f = $f * $f + $s;
        }
        dibujar el punto p con brillo $i;
}
```

Aquí, el parámetro s representa la semilla o *seed* usada para generar el fractal. Se trata de una constante compleja que nos permite parametrizar la forma del fractal, cuyo valor por defecto está especificado en la sección 5.2.

Así tendremos, al finalizar, una representación visual de la cantidad de ciclos de cómputo realizados hasta alcanzar la condición de escape (ver figura 1).

5.2. Interfaz

A fin de facilitar el intercambio de código *ad-hoc*, normalizaremos algunas de las opciones que deberán ser provistas por el programa:

- `-r`, o `--resolution`, permite cambiar la resolución de la imagen generada. El valor por defecto será de `640x480` puntos.
- `-c`, o `--center`, para especificar las coordenadas correspondientes al punto central de la porción del plano complejo dibujada, expresado en forma binómica (i.e. $a+bi$). Por defecto usaremos `0+0i`.
- `-w`, o `--width`, especifica el ancho de la región del plano complejo que estamos por dibujar. Valor por defecto: `2`.

- `-H`, o `--height`, sirve, en forma similar, para especificar el alto del rectángulo a dibujar. Valor por defecto: `2`.
- `-s`, o `--seed`, para configurar el valor complejo de la semilla usada para generar el fractal. Valor por defecto: `-0.726895347709114071439+0.188887129043845954792i`.
- `-o`, o `--output`, permite colocar la imagen de salida, (en formato PGM [6]) en el archivo pasado como argumento; o por salida estándar `-cout-` si el argumento es "-".

5.3. Casos de prueba

Es necesario que el informe trabajo práctico incluya una sección dedicada a verificar el funcionamiento del código implementado.

En el caso del TP 0, será necesario escribir pruebas orientadas a probar el programa completo, ejercitando los casos más comunes de funcionamiento, los casos de borde, y también casos de error.

Incluimos en este enunciado dos fractales de referencia que pueden ser usados para comprobar visualmente el funcionamiento del programa.

5.4. Ejemplos

Generamos un dibujo usando los valores por defecto, barriendo la región rectangular del plano comprendida entre los vértices $-2+2i$ y $+2-2i$.

```
$ tp0 -o uno.pgm
```

La figura 1 muestra la imagen `uno.pgm`.

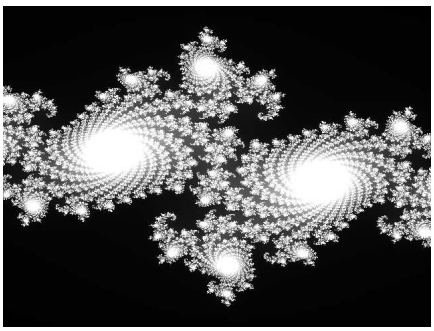


Figura 1: Región barrida por defecto.

A continuación, hacemos *zoom* sobre la región centrada en `0.282-0.007i`, usando un rectángulo de `0.005` unidades de lado.

```
$ tp0 -c 0.282-0.007i -w 0.005 -H 0.005 -o dos.pgm
```

El resultado podemos observarlo en la figura 2.

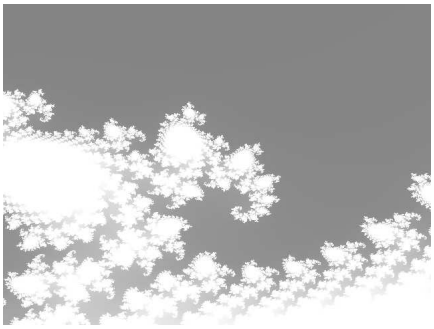


Figura 2: Región comprendida entre $0{,}2795 - 0{,}0045i$ y $0{,}2845 - 0{,}0095i$.

6. Informe

El informe deberá incluir:

- Documentación relevante al diseño e implementación del programa.
- Documentación relevante al proceso de compilación: cómo obtener el ejecutable a partir de los archivos fuente.
- Las corridas de prueba, con los comentarios pertinentes.
- El código fuente, en lenguaje C.
- Este enunciado.

7. Fecha de entrega

La última fecha de entrega y presentación sería el martes 10/4.

Referencias

[1] GXemul, `http://gavare.se/gxemul/`.

[2] The NetBSD project.
`http://www.netbsd.org/`.

[3] `http://es.wikipedia.org/wiki/Conjunto_de_Julia` (Wikipedia).

[4] `http://mathworld.wolfram.com/JuliaSet.html` (Mathworld).

[5] Smooth shading for the Mandelbrot exterior.
`http://linas.org/art-gallery/escape/smooth.html`. Linas Vepstas. October, 1997.

[6] PGM format specification.
`http://netpbm.sourceforge.net/doc/pgm.html`.