

# Un Modelo de Referencia para la Generación de Informes

Tkaczyszyn Juan Facundo, *Padrón Nro. 87.931*  
facu.tk@gmail.com

Nombre y Apellido de Autor, *Padrón Nro. 00.000*  
dirección de e-mail

Nombre y Apellido de Autor, *Padrón Nro. 00.000*  
dirección de e-mail

2do. Cuatrimestre de 2014

66.20 Organización de Computadoras – Práctica Martes  
Facultad de Ingeniería, Universidad de Buenos Aires

## Resumen

Este template es un modelo que proporciona a los alumnos las instrucciones necesarias para preparar sus informes para la asignatura *66.20 Organización de Computadoras* (práctica Viernes). El informe podrá contener (optativo) un resumen de no más de 150 palabras. La primera página del artículo deberá seguir el formato que se ilustra en el presente modelo y deberá contener el título, los nombres de los autores, sus números de padrón, sus direcciones de e-mail, y el resumen (si tuviese). La primera página del informe no debe ser numerada.

## 1. Introducción

Este artículo es un modelo que proporciona a los alumnos las instrucciones necesarias para preparar sus informes para la asignatura *66.20 Organización de Computadoras* (práctica Viernes). Además de la estructura presentada, el informe podría contener otras secciones y subsecciones, a continuación de la introducción. Podrá incluir (se recomienda) gráficos ilustrativos y/o tablas. El informe finaliza con una sección de conclusiones, y las citas bibliográficas consultadas siguiendo, rigurosamente, el formato presentado al final de este modelo. También se recomienda respetar el estilo tipográfico mostrado aquí (fuente Times Roman de 10 puntos, u otras similares)

## 2. Corridas de prueba

Texto de la sección...

2.1. Este es el Título de una Subsección

Texto de la subsección...

```
$ ./tp0 --center 0+0i --width 2 --height 2
--resolution 14x11 --output -

P2
14
11
255
  2   2   2   3   3   4  12  44   3   2   2   1   1   1
  2   3   3   3   5   9 255  24   4   3   3   2   1   1
  3   4   5  65  10 255 255 255  30   8   5   2   2   1
  4   5   8 239 255 255 255 255 255 255  6   3   2   2
255  12  52 255 255 255 255 255 255 255  8   3   2   2
255 255 255 255 255 255 255 255 255 255  5   3   2   2
255 255 255 255 255 255 255 255 255 255 14   5   3   2
255  12  52 255 255 255 255 255 255 255  8   3   2   2
  4   5   8 239 255 255 255 255 255 255  6   3   2   2
  3   4   5  65  10 255 255 255  30   8   5   2   2   1
  2   3   3   3   5   9 255  24   4   3   3   2   1   1
```

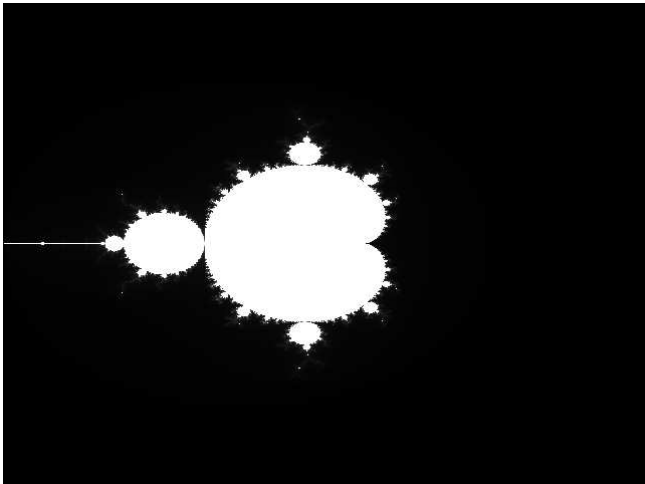


Figura 1: Mandelbrot0

## 2.2. Este es el Título de una Subsección

Texto de la subsección...

```
$ ./tp0 --width 0.006089755361389781 --height 0.006089755361389781
--center -0.16495019360389762+1.0391402340922113i
--resolution 14x11 --output -
```

```
P2
14
11
255
```

26	32	26	31	24	22	21	21	23	25	22	21	22	26
47	34	41	29	25	29	24	24	26	29	24	34	25	30
30	54	49	34	31	30	27	26	31	28	27	29	34	58
27	32	35	56	41	35	36	34	36	32	36	34	72	44
22	30	39	33	40	81	45	45	53	45	49	43	55	96
22	24	26	32	37	55	127	119	123	63	69	103	117	255
39	29	28	30	37	47	84	255	255	255	255	255	255	255
22	24	27	30	52	41	103	86	255	255	255	255	255	255
21	22	24	27	35	39	78	105	255	255	255	255	255	255
21	23	31	30	34	40	50	255	255	255	255	255	255	255
23	24	26	32	52	61	76	255	255	255	255	255	255	255

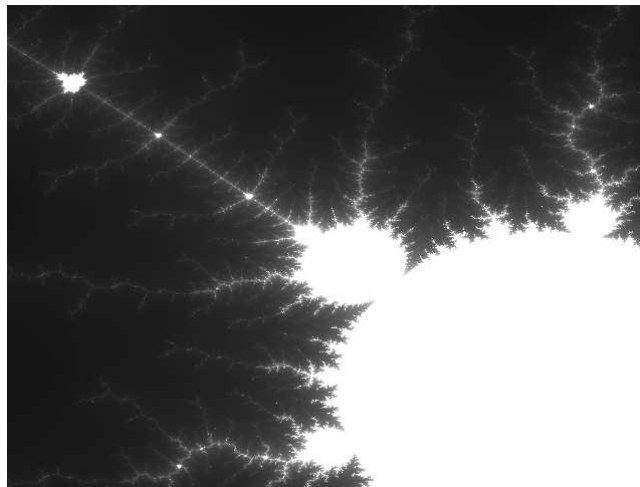


Figura 2: Mandelbrot1

### 2.3. Este es el Título de una Subsección

Texto de la subsección...

```
$ ./tp0 --width 0.00913463304208467 --height 0.00913463304208467
--center -0.027010582808902495+0.7093001367538602i
--resolution 14x11 -output -
```

```
P2
14
11
255
255 255 255 255 255 255 107 84 88 52 39 36 39 64
255 255 255 255 255 255 86 125 61 47 40 39 66 52
255 255 255 255 255 255 255 63 59 64 42 42 44 63
255 255 255 255 255 255 228 255 230 53 45 44 45 46
255 255 255 255 255 255 255 202 140 54 110 85 49 52
255 255 255 255 255 255 255 179 74 61 58 64 81 90
255 255 255 224 255 255 164 139 105 85 100 196 105 117
255 149 217 144 73 230 158 71 154 175 92 75 62 49
255 90 78 58 53 56 81 58 81 74 97 93 115 61
194 255 255 118 48 43 45 49 70 66 97 83 49 55
58 64 84 72 47 48 38 55 52 45 43 123 38 34
```

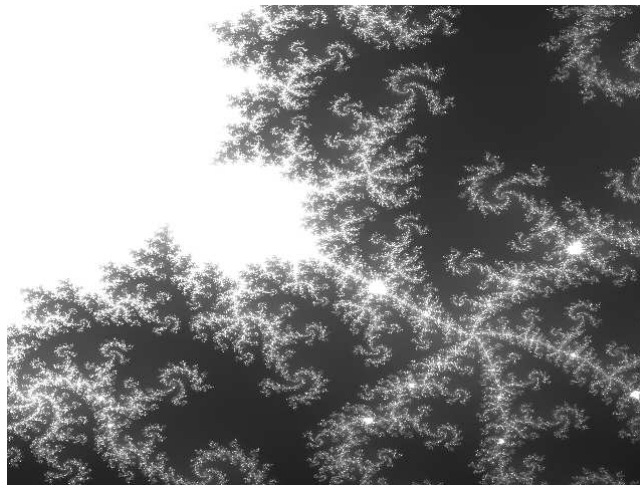


Figura 3: Mandelbrot2

## 3. Código Fuente

Texto de la sección...

### 3.1. main.c

```
#include <stdio.h>
#include "default_values.h"

int main(int argc, char** argv) {

    int    res_x    = default_res_x;
    int    res_y    = default_res_y;
    double width    = default_width;
    double height   = default_height;
    double c_re     = default_c_re;
    double c_im     = default_c_im;
    FILE * output;

    int parse_result =  parse_opts( argc,
                                    argv,
                                    &res_x,
                                    &res_y,
                                    &c_re,
                                    &c_im,
                                    &width,
                                    &height,
                                    &output );

    if ( parse_result == 0 ) {
        mandelbrot( res_x,
                    res_y,
                    c_re,
                    c_im,
                    width,
                    height,
                    output );
    }
    return 0;
}
```

### 3.2. default\_values.c

```
const int    default_res_x    = 640;
const int    default_res_y    = 480;
const double default_width    = 4;
const double default_height   = 4;
const double default_c_re     = 0;
const double default_c_im     = 0;
```

### 3.3. parse\_opt.c

```
#include <stdio.h>
#include <string.h>
#include <getopt.h>

const char* const op_cortas = "r:c:w:H:o:hV";

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

int parse_width( char * param, double * result ) {
    double width;
    int scanned = sscanf( param, "%lf", &width );
    if ( scanned == 1 ) {
        if ( width > 0 ) {
            *result = width;
            return 0;
        }
    }
    fprintf( stderr, "fatal: invalid width specification.\n" );
    return 1;
}

int parse_height( char * param, double * result ) {
    double height;
    int scanned = sscanf( param, "%lf", &height );
    if ( scanned == 1 ) {
        if ( height > 0 ) {
            *result = height;
            return 0;
        }
    }
    fprintf( stderr, "fatal: invalid height specification.\n" );
    return 1;
}

int parse_resolution( char * param, int * res_x, int * res_y ) {
    int _res_x;
    int _res_y;
    int scanned = sscanf( param, "%dx%d", &_res_x, &_res_y );
    if ( scanned == 2 ) {
        if ( ( _res_x > 0 ) && ( _res_y > 0 ) ) {
            *res_x = _res_x;
            *res_y = _res_y;
            return 0;
        }
    }
}
```

```

    }
}
fprintf( stderr, "fatal: invalid resolution specification.\n" );
return 1;
}

int parse_center( char * param, double * c_re, double * c_im ) {
    double _c_re;
    double _c_im;
    char _c_im_sign;
    int scanned = sscanf( param,
                          "%lf%c%lf" ,
                          &_c_re,
                          &_c_im_sign,
                          &_c_im );

    if ( scanned == 3 ) {
        if ( _c_im_sign == '-' )
            _c_im = _c_im * -1;
        *c_re = _c_re;
        *c_im = _c_im;
        return 0;
    }
    fprintf( stderr, "fatal: invalid center specification.\n" );
    return 1;
}

int parse_output( char * param, FILE ** output ) {
    FILE * _output = 0;

    if ( strcmp(param, "-") == 0 ) {
        _output = stdout;
    } else {
        _output = fopen( param, "wb" );
    }

    if ( _output ) {
        *output = _output;
        return 0;
    }

    fprintf( stderr, "fatal: Output file error.\n" );
    return 1;
}

void print_help( char * binary_name ) {
    printf(
        "Usage:\n"
        "  %s [options]\n"
        "\n"
        "Options:\n"
        "-r, --resolution (WxH) Image resolution (default: 640x480).\n"
        "-c, --center (a+bi) Complex plane center (default: 0+0i).\n"
        "-w, --width (w) Complex plane width (default: 4).\n"
        "-H, --height (h) Complex plane height (default: 4).\n"
    );
}

```

```

    "\n",
    binary_name );
}

void print_version() {
    printf("66.20TP0-Mandelbrot, Version 1.0\n");
}

int parse_opts( int argc,
                char * const * argv,
                int * res_x,
                int * res_y,
                double * c_re,
                double * c_im,
                double * width,
                double * height,
                FILE ** output ) {

    int output_defined = 0;
    int result;

    // getopt does not print over stderr
    opterr = 0;

    // every argument processed
    int next_opt = 0;

    while (1) {
        next_opt = getopt_long( argc,
                                argv,
                                op_cortas,
                                op_largas,
                                NULL);

        if (next_opt == -1) {
            break;
        }

        switch (next_opt) {

            case 'r': {
                if ( parse_resolution( optarg,
                                        res_x,
                                        res_y ) > 0 )
                    return 1;
                break;
            }

            case 'c': {
                if ( parse_center( optarg,

```



```

        c_re,
        c_im ) > 0 )

        return 1;
    break;
}

case 'w': {
    if( parse_width( optarg,
                    width ) > 0 )

        return 1;
    break;
}

case 'H': {
    if( parse_height( optarg,
                    height ) > 0 )

        return 1;
    break;
}

case 'o': {
    if( parse_output( optarg,
                    output ) == 0 ) {
        output_defined = 1;
    } else {
        return 1;
    }
    break;
}

case 'h': {
    print_help( argv[0] );
    return 1;
    break;
}

case 'V': {
    print_version();
    return 1;
    break;
}

default: {
    print_help( argv[0] );
    return 1;
    break;
}
}

if ( !output_defined )
    return 1;
return 0;
}

```

### 3.4. mandelbrot.c

```
#include <stdio.h>
#include <math.h>

int mandelbrot( int res_x,
                int res_y,
                double c_re,
                double c_im,
                double width,
                double height,
                FILE * output ) {

    const int    max_it = 255;
    const double escape_radius = 2;
    int it, it_x, it_y;
    double c_x, c_y;
    double c_x_min = c_re - ( width / 2 );
    double c_y_min = c_im - ( height / 2 );
    double px_width  = ( width )/res_x;
    double px_height = ( height )/res_y;
    double z_x, z_y, z_x_sq, z_y_sq;
    double er_sq = escape_radius*escape_radius;

    // PGM header
    fprintf( output, "P2\n%d\n%d\n%d\n", res_x, res_y, max_it );

    // iterate over the coordinates and write the data
    for( it_y = res_y; it_y > 0 ; it_y-- ) {
        c_y = c_y_min + it_y * px_height;
        if( fabs( c_y ) < px_height / 2 ) c_y = 0.0;
        for( it_x = 0 ; it_x < res_x ; it_x++ ) {
            c_x = c_x_min + it_x * px_width;
            z_x = c_x;
            z_y = c_y;
            z_x_sq = z_x * z_x;
            z_y_sq = z_y * z_y;
            for ( it = 0;
                  it < max_it && ((z_x_sq + z_y_sq )< er_sq );
                  it++) {
                z_y = 2 * z_x * z_y + c_y;
                z_x = z_x_sq - z_y_sq + c_x;
                z_x_sq = z_x * z_x;
                z_y_sq = z_y * z_y;
            }
            fprintf( output, "%3d", it);
        }
        fprintf( output, "\n");
    }
    return 0;
}
```

## 4. Este es el Título de Otra Sección

Texto de la otra sección. En la figura 4 se muestra un ejemplo de cómo presentar las ilustraciones del informe.



Figura 4: Facultad de Ingeniería – Universidad de Buenos Aires.

### 4.1. Este es el Título de Otra Subsección

Texto de la otra subsección...

## 5. Conclusiones

Se presentó un modelo para que los alumnos puedan tomar como referencia en la redacción de sus informes de trabajos prácticos.

## Referencias

- [1] Intel Technology & Research, “Hyper-Threading Technology,” 2006, <http://www.intel.com/technology/hyperthread/>.
- [2] J. L. Hennessy and D. A. Patterson, “Computer Architecture. A Quantitative Approach,” 3ra Edición, Morgan Kaufmann Publishers, 2000.
- [3] J. Larus and T. Ball, “Rewriting Executable Files to Measure Program Behavior,” Tech. Report 1083, Univ. of Wisconsin, 1992.