



**FACULTAD
DE INGENIERIA**

Universidad de Buenos Aires

Organización de Computadoras 66.20

Trabajo practico N°0

Integrantes:

- Alvarez Fernando 99373
- Braccelarghe Ailin 99366
- Suarez Martin 101540

1 Documentación del diseño e implementación del programa

El programa realizado es la estructura de la estructura de una matriz, *matrix_t*, la cual tuviese un método para poder multiplicarse con otra matriz.

Las matrices a multiplicar se encontraran en un archivo.txt o streams de entrada y salida, el programa las recibirá por linea de comando.

1.1 Estructura del programa

- main.c
- matrix.h
- matrix.c
- makefile

1.2 Estructura de *matrix_t*

- `matrix_t* create_matrix(size_t rows, size_t cols);`

Es el constructor de la matriz.

- `int index_value(matrix_t* m, int x, int y, double value);`

Sirve para insertar un valor (*double*) en la coordenada (x,y) de la matriz, donde **x** indica la columna e **y** la fila

- `double value_obtain(matrix_t* m, int x, int y);`

Permite obtener el valor que se encuentra en la coordenada (x,y) de la matriz

- `int complete_matrix(double* values, matrix_t* m);`

Completa la matriz a partir de un array de *double*'s

- `int destroy_matrix(matrix_t* m);`

Destructor de la matriz

- `int print_matrix(FILE* fp, matrix_t* m);`

Se encarga de imprimir la matriz resultante en un file pointer dado

- `matrix_t* matrix_multiply(matrix_t* m1, matrix_t* m2);`

Se encarga de multiplicar dos matrices, devuelve una matriz resultado

2 Comandos del makefile

\$ make build: Compila el programa.

\$ make run: Ejecuta el programa.

\$ make all: Compila y ejecuta el programa.

\$ make assembly: Compila el programa sin optimizaciones y se detiene al generar el código assembly. Utiliza además el parámetro -mrnames para que el compilador reemplaze los números de registro por sus nombres.

3 Corridas de prueba

3.1 Entrada por txt

3.1.1 input: test.txt

```
2 1 2 3 4 1 2 3 4
2 1 1 1 1 1 1 1 1
3 1 2 3 4 5 6.1 3 2 1 1 0 0 0 1 0 0 0 1
2 0 0 0 0 0 0 0 0 0 0 0 0 0 0
5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
18 19 20 21 22 23 24 25
2 1
3 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9
```

3.1.2 output: out.txt

```
Resultado de la multiplicación:
| 7 10 | 15 22
Resultado de la multiplicación:
| 2 2 | 2 2
Resultado de la multiplicación:
| 1 2 3 | 4 5 6.1 | 3 2 1
Resultado de la multiplicación:
| 215 230 245 260 275 | 490 530 570 610 650 | 765 830 895 960 1025 | 1040 1130 1220 1310 1400 | 1315 1430
1545 1660 1775
Resultado de la multiplicación:
| 0 0 | 0 0
```

3.1.3 output: consola

```
2 7 10 15 22
2 2 2 2 2
3 1 2 3 4 5 6.1 3 2 1
5 215 230 245 260 275 490 530 570 610 650 765 830 895 960 1025 1040 1130 1220 1310 1400 1315 1430 1545
1660 1775
2 0 0 0 0
ERROR EN LECTURA: Faltan valores
```

3.2 Entrada por consola

```
input: 3 ladasda asd ds
output: ERROR EN LECTURA: Valor en formato erroneo
input: 2 1/ 2 4 6 7 8 9 7 1
output: ERROR EN LECTURA: Valor en formato erroneo
input: 2 1e2 3e2 1e5 6e4 1e-6 0.25e3 3e5 4
output: 2 9e+07 26200 1.8e+10 2.524e+07
se agrega en out.txt: | 9e+07 26200 | 1.8e+10 2.524e+07
input: 2 a
output: ERROR EN LECTURA: Valor en formato erroneo
input: a
output: ERROR EN LECTURA: No se recibio un tamaño correcto de matriz
input: -2 1 2 3 4 5 6 7 8
output: ERROR EN LECTURA: No se recibio un tamaño correcto de matriz
input:
output: ERROR EN LECTURA: No se recibio un tamaño correcto de matriz
```

4 Código fuente

4.1 main.c

```
1
2 #include "matrix.c"
3 #include <stdlib.h>
4 #include <stdio.h>
5 #include <string.h>
6
7 int main(int argc, const char* argv[])
8 {
9     int size;
10    double value;
11    char buffer[4096];
12    char* delim = " ";
13    int ok;
14
15    while(fgets(buffer, sizeof(buffer), stdin)){
16
17        ok = sscanf(buffer, "%i", &size);
18        if(!ok || size <= 0){
19            fprintf(stderr, "ERROR EN LECTURA: No se recibio un tama o correcto de matriz\n");
20            return(1);
21        }
22        int ammount = 2*size*size;
23        double* values = (double*) malloc(ammount*sizeof(double));
24        if(!values) {
25            fprintf(stderr, "MEMORY ERROR\n");
26            return (1);
27        }
28        char* string = strtok(buffer, delim);
29        for(int i = 0; i < ammount; i++){
30            string = strtok(NULL, delim);
31            if(!string){
32                free(values);
33                fprintf(stderr, "%s", "ERROR EN LECTURA: Faltan valores\n");
34                return(1);
35            }
36
37            char* error = "";
38            value = strtod(string, &error);
39            if( strcmp("", error) != 0){
40                free(values);
41                fprintf(stderr, "%s", "ERROR EN LECTURA: Valor en formato erroneo\n");
42                return(1);
43            }
44
45            values[i] = value;
46        }
47
48        double* values_A = (double*) malloc(size*size*sizeof(double));
49        memcpy(values_A, values, size*size*sizeof(double));
50        double* values_B = (double*) malloc(size*size*sizeof(double));
51        memcpy(values_B, values+size*size, size*size*sizeof(double));
52
53        matrix_t* matrix_A = create_matrix(size, size);
54        matrix_t* matrix_B = create_matrix(size, size);
55
56        complete_matrix(values_A, matrix_A);
57        complete_matrix(values_B, matrix_B);
58
59        matrix_t* matrix_C = matrix_multiply(matrix_A, matrix_B);
60
61        FILE *file;
62        file = fopen("out.txt", "a");
63
64        print_matrix(file, matrix_C);
```

```

65     free(values);
66     free(values_A);
67     free(values_B);
68
69     destroy_matrix(matrix_A);
70     destroy_matrix(matrix_B);
71     destroy_matrix(matrix_C);
72
73
74     fclose(file);
75 }
76 return(0);
77 }

```

4.2 matrix.h

```

1
2 #ifndef MATRIX_H
3 #define MATRIX_H
4
5 #include <stdbool.h>
6 #include <stdlib.h>
7 #include <stdio.h>
8
9 struct matrix;
10 typedef struct matrix matrix_t;
11
12 // Constructor de matriz.
13 matrix_t* create_matrix(size_t rows, size_t cols);
14
15 // Guarda un double en la coordenada (x,y) dada.
16 int index_value(matrix_t* m, int x, int y, double value);
17
18 // Obtiene el valor (double) en la coordenada (x,y).
19 double value_obtain(matrix_t* m, int x, int y);
20
21 // Completa la matriz a partir de un array de doubles.
22 int complete_matrix(double* values, matrix_t* m);
23
24 // Destructor de matriz
25 int destroy_matrix(matrix_t* m);
26
27 // Imprime matriz sobre el file pointer fp.
28 int print_matrix(FILE* fp, matrix_t* m);
29
30 // Multiplica las matrices en m1 y m2.
31 matrix_t* matrix_multiply(matrix_t* m1, matrix_t* m2);
32
33 #endif // MATRIX_H

```

4.3 matrix.c

```
1
2 #include "matrix.h"
3 #include <stdlib.h>
4 #include <stdio.h>
5 #include <string.h>
6
7 struct matrix{
8     size_t rows;
9     size_t cols;
10    double* array;
11 };
12
13 matrix_t* create_matrix(size_t rows, size_t cols){
14     if(rows != cols) return NULL;
15     matrix_t* matrix = malloc(sizeof(matrix_t));
16     if(!matrix) {
17         fprintf(stderr, "MEMORY ERROR");
18         return NULL;
19     }
20     matrix->rows = rows;
21     matrix->cols = cols;
22     matrix->array = (double*) calloc(cols*rows, sizeof(double));
23     if(!matrix->array) {
24         free(matrix);
25         fprintf(stderr, "MEMORY ERROR");
26         return NULL;
27     }
28     return matrix;
29 }
30
31 int index_value(matrix_t* m, int x, int y, double value){
32     if(m->rows == 0 || m->cols == 0) return -1;
33     m->array[y*m->rows + x] = value;
34     return 0;
35 }
36
37 double value_obtain(matrix_t* m, int x, int y){
38     return m->array[y*m->rows + x];
39 }
40
41 int complete_matrix(double* values, matrix_t* m){
42     if(!m) {
43         fprintf(stderr, "NO MATRIX CREATED ERROR");
44         return -1;
45     }
46     for(int y = 0; y < m->rows; y++){
47         for(int x = 0; x < m->cols; x++){
48             double value = values[m->rows * y + x];
49             index_value(m, x, y, value);
50         }
51     }
52     return 0;
53 }
54
55 int destroy_matrix(matrix_t* m){
56     if(!m) {
57         fprintf(stderr, "NO MATRIX CREATED ERROR");
58         return -1;
59     }
60     free(m->array);
61     free(m);
62     return 0;
63 }
64
65 int print_matrix(FILE* fp, matrix_t* m){
66     if(!m) {
67         fprintf(stderr, "NO MATRIX CREATED ERROR");
```

```

68     return (-1);
69 }
70 if(!fp){
71     perror("Error al crear el archivo de salida");
72     return (-1);
73 }
74 else{
75     fprintf(stdout, "%li ", m->rows);
76     fprintf(fp, "%s\n", "Resultado de la multiplicacion:");
77     for(int y = 0; y < m->rows; y++){
78         fputc('|', fp);
79         fputc(' ', fp);
80         for(int x = 0; x < m->cols; x++){
81             double value = value_obtain(m, x, y);
82             fprintf(stdout, "%lg ", value);
83             fprintf(fp, "%lg", value);
84             fputc(' ', fp);
85         }
86     }
87     fprintf(stdout, "\n");
88     fputc('\n', fp);
89 }
90 fflush(fp);
91 return 0;
92 }
93
94 matrix_t* matrix_multiply(matrix_t* m1, matrix_t* m2){
95     if(m1->rows != m1->cols || m2->rows != m2->cols || m1->rows != m2->rows) {
96         fprintf(stderr, "DIMENSION ERROR");
97         return NULL;
98     }
99     matrix_t* mresult = create_matrix(m1->rows, m1->cols);
100     if(!mresult) {
101         fprintf(stderr, "NO MATRIX CREATED ERROR");
102         return NULL;
103     }
104     int N = m1->cols;
105     for(int i = 0; i < N; i++){
106         for(int x = 0; x < N; x++){
107             index_value(mresult, x, i, 0);
108             for(int y = 0; y < N; y++){
109                 double value = value_obtain(mresult, x, i);
110                 value += value_obtain(m1, y, i) * value_obtain(m2, x, y);
111                 index_value(mresult, x, i, value);
112             }
113         }
114     }
115     return mresult;
116 }

```

5 Código Assembly MIPS32

5.1 primera página de main.s

```
1  .section .mdebug.abi32
2  .previous
3  .abicalls
4  .file 1 "main.c"
5  .section .debug-abbrev,"",@progbits
6  $Ldebug_abbrev0:
7  .section .debug_info,"",@progbits
8  $Ldebug_info0:
9  .section .debug_line,"",@progbits
10 $Ldebug_line0:
11 .text
12 $Ltext0:
13 .file 2 "/usr/include/mips/int_types.h"
14 .file 3 "/usr/include/mips/types.h"
15 .file 4 "/usr/include/mips/ansi.h"
16 .file 5 "/usr/include/sys/ansi.h"
17 .file 6 "/usr/include/sys/types.h"
18 .file 7 "/usr/include/sys/endian.h"
19 .file 8 "/usr/include/pthread_types.h"
20 .file 9 "/usr/include/stdlib.h"
21 .file 10 "/usr/include/stdio.h"
22 .file 11 "matrix.h"
23 .rdata
24 .align 2
25 $LC0:
26 .ascii " \000"
27 .align 2
28 $LC1:
29 .ascii "%i\000"
30 .align 2
31 $LC2:
32 .ascii "ERROR EN LECTURA: No se recibio un tama\303\261o correct"
33 .ascii "o de matriz\n\000"
34 .align 2
35 $LC3:
36 .ascii "MEMORY ERROR\n\000"
37 .align 2
38 $LC4:
39 .ascii "%s\000"
40 .align 2
41 $LC5:
42 .ascii "ERROR EN LECTURA: Faltan valores\n\000"
43 .align 2
44 $LC6:
45 .ascii "\000"
46 .align 2
47 $LC7:
48 .ascii "\n\000"
49 .align 2
50 $LC8:
51 .ascii "ERROR EN LECTURA: Valor en formato erroneo\n\000"
52 .align 2
53 $LC9:
54 .ascii "out.txt\000"
55 .align 2
56 $LC10:
57 .ascii "a\000"
58 .text
59 .align 2
60 .globl main
61 $LFB29:
62 .loc 1 7 0
63 .ent main
```


5.2 primera página de matrix.s

```
1  .section .mdebug.abi32
2  .previous
3  .abicalls
4  .file 1 "matrix.c"
5  .section .debug_abbrev,"",@progbits
6  $Ldebug_abbrev0:
7  .section .debug_info,"",@progbits
8  $Ldebug_info0:
9  .section .debug_line,"",@progbits
10 $Ldebug_line0:
11 .text
12 $Ltext0:
13 .file 2 "/usr/include/mips/int_types.h"
14 .file 3 "/usr/include/mips/types.h"
15 .file 4 "/usr/include/mips/ansi.h"
16 .file 5 "/usr/include/sys/ansi.h"
17 .file 6 "/usr/include/sys/types.h"
18 .file 7 "/usr/include/sys/endian.h"
19 .file 8 "/usr/include/pthread_types.h"
20 .file 9 "/usr/include/stdlib.h"
21 .file 10 "/usr/include/stdio.h"
22 .file 11 "matrix.h"
23 .rdata
24 .align 2
25 $LC0:
26 .ascii "MEMORY ERROR\000"
27 .text
28 .align 2
29 .globl create_matrix
30 $LFB29:
31 .loc 1 12 0
32 .ent create_matrix
33 create_matrix:
34 .frame $fp,48,$ra # vars= 8, regs= 4/0, args= 16, extra= 8
35 .mask 0xd0010000,-4
36 .fmask 0x00000000,0
37 .set noreorder
38 .cpload $t9
39 .set reorder
40 subu $sp,$sp,48
41 .cpstore 16
42 $LCFI0:
43 sw $ra,44($sp)
44 $LCFI1:
45 sw $fp,40($sp)
46 $LCFI2:
47 sw $gp,36($sp)
48 $LCFI3:
49 sw $s0,32($sp)
50 $LCFI4:
51 move $fp,$sp
52 $LCFI5:
53 sw $a0,48($fp)
54 sw $a1,52($fp)
55 .loc 1 13 0
56 $LBB2:
57 lw $v1,48($fp)
58 lw $v0,52($fp)
59 beq $v1,$v0,$L18
60 sw $zero,28($fp)
61 b $L17
62 $L18:
63 .loc 1 14 0
64 li $a0,12 # 0xc
65 la $t9, malloc
66 jal $ra,$t9
```