



*Universidad de Santiago de Chile  
Facultad de Ingeniería  
Departamento de Ingeniería Informática*

## **Estructuras de Datos y Análisis de Algoritmos**

### **Informe Laboratorio N°1**

*Fecha entrega: 17/09/2014  
Alumno: Joaquín Ignacio Villagra Pacheco  
Profesora: Jacqueline Köhler Casasempere*



## ÍNDICE

Introducción.....	2
Descripción de la solución.....	3
Análisis de los resultados obtenidos .....	8
Conclusión.....	9
Referencias.....	10
Anexo: Capturas de pantalla.....	11

## INTRODUCCIÓN

Como trabajo de laboratorio de la asignatura Análisis de algoritmos y estructura de datos, se ha solicitado desarrollar una aplicación, en el lenguaje de programación C, que genere la serie Fibonacci para valores grandes de N.

Se preguntará: ¿Cuál es el problema?

Para valores pequeños de N, es posible usar las representaciones de enteros provistas por los lenguajes de programación, sin embargo para números un poco más elevados, la serie de Fibonacci supera los 32 Bits disponibles para el tipo de dato entero, lo que imposibilita la obtención de resultados certeros de dicha serie matemática.

Este proyecto es una adaptación del código Fibonacci iterativo que conocemos, que busca ampliar el intervalo de solución, generando la efectiva realización de la serie Fibonacci para números elevados de N.

Para el desarrollo de la aplicación se sugiere representar los números enteros como un arreglo de enteros de gran tamaño y sobre este, implementar el algoritmo solicitado.

Este informe estará compuesto por la descripción de solución, en donde se indica el método de resolución de problema aplicado; Análisis de resultados obtenidos, en donde se mostrará una visión crítica del programa desarrollado; Conclusión, en donde se especificará los logros obtenidos; y finalmente las referencias bibliográficas de la información expuesta.

## DESCRIPCIÓN DE LA SOLUCIÓN

Primero que todo hay que comprender el porque surge este trabajo. El motivo, es la limitante que tiene el tipo de dato entero, a nivel de lenguajes de programación, para contener números elevados. Recordemos que un numero de tipo entero (INT) reserva 4 Bytes de memoria, a su vez un Byte son 8 bits. Aplicando la transitividad de la igualdad, se obtiene que un entero posee reservados: **4 Bytes =  $4 \times 8 = 32$  bits**

*Con 32 bits se pueden representar  $2^{32} = 4294967296$  valores*

*Sólo positivos (enteros sin signo): del 0 al 4294967295*

*Positivos y negativos (enteros con signo): del -2147483648 al 2147483647*

Teniendo claro el porque de este proyecto, se procede a describir la solución generada.

Aplicando la lógica de la división en subproblemas, aprendida en el curso de métodos de programación, se segmenta el proceso que genera la serie Fibonacci en dos funciones. Dicha división es realizada con el fin de facilitar el desarrollo de las funciones de manera independiente y al mismo tiempo, en conjunto generar la solución a nuestro problema raíz, el cual es obtener la serie Fibonacci para números elevados de N.

Ya que el problema es el tipo de dato a utilizar, se emula un nuevo tipo de dato, basándose en las estructuras de dato conocidas, que sea capaz de soportar números que estén por sobre los 4.294.967.295 valores que soporta el tipo de dato INT provisto por los lenguajes de programación. Para dicha emulación, se utiliza un puntero de tipo entero en formato de arreglo, al cual se le reserva una amplia sección de memoria, suficiente para que este arreglo soporte los valores generados por la serie Fibonacci de un numero N de gran tamaño (*Véase en línea 10, Figura 1*). A su vez, se utilizan dos arreglos de tipo entero (INT), del mismo largo definido para el puntero descrito anteriormente (*Véase línea 31, Figura 2*).

Para manejar mayor orden a nivel de desarrollo, se separa el software en dos archivos, los cuales se detallan a continuación:

1. Funciones.c  
Archivo que contiene las funciones desarrolladas (*Véase Figura 1 y Figura 2*).
2. Laboratorio.c  
Archivo que contiene el main del programa solicitado (*Véase Figura 3*).

En la figura 1 se aprecia la implementación de la función sumaDeValores, la cual realiza el proceso base de nuestro problema raíz. Realiza la suma de los valores correspondientes al Fibonacci de los números (n-1) y (n-2), tal como lo indica la definición matemática de dicha serie.

```

7  int *sumaDeValores(int numero1[], int numero2[], int largo)
8  {
9      int indice, acumulador=0;
10     int *Array = (int*) malloc (sizeof(int)*largo);
11     for(indice=largo-1;indice>=0;indice--)
12     {
13         int s = numero1[indice] + numero2[indice];
14         if(s/10>=1)
15         {
16             Array[indice]      = s % 10;
17             acumulador         = s / 10;
18             numero1[indice-1] = numero1[indice-1] + acumulador;
19         }
20         else
21         {
22             Array[indice] = s;
23         }
24     }
25     return Array;
26 }

```

*Figura 1: Implementación de función “sumaDeValores”*

A continuación, la figura 2 muestra el código de la implementación de la función “fibonacci”. Esta segunda función, utiliza dentro de su desarrollo varios llamados a “sumaDeValores” (Véase línea 43).

```

29 void fibonacci(int n)
30 {
31     int Valor1[LARGO], Valor2[LARGO], indiceAuxiliar, indice, valor = 1;
32     for(indice=0; indice < LARGO; indice++)
33     {
34         Valor1[indice]=0;
35         Valor2[indice]=0;
36         if(indice==LARGO-1)
37         {
38             Valor2[indice]=1;
39         }
40     }
41     for(indiceAuxiliar = 1; indiceAuxiliar < n; indiceAuxiliar++)
42     {
43         int *Puntero = sumaDeValores(Valor1, Valor2, LARGO);
44         for(indice=0;indice<LARGO;indice++)
45         {
46             Valor1[indice] = Valor2[indice];
47             Valor2[indice] = Puntero[indice];
48         }
49     }
50     for(indice=0;indice<LARGO;indice++)
51     {
52         if(valor==1)
53         {
54             if (Valor2[indice+1]!=0)
55             {
56                 valor=0;
57             }
58         }
59         else
60         {
61             printf("%d", Valor2[indice]);
62         }
63     }
64 }
65

```

*Figura 2: Implementación de función “fibonacci”*

La figura 3, detalla la implementación del bloque principal del software. Esta sección, es la encargada de ejecutar las funciones detalladas anteriormente y manejar la entrada y salida de datos. Particularmente en este caso, la salida de datos no esta en manos del Main, ya que la función Fibonacci imprime directamente en pantalla el resultado obtenido.

```
1
2
3 #include "funciones.h"
4
5 int main(int argc, char **argv)
6 {
7     int n = atoi(argv[1]);
8     fibonacci(n);
9     printf("\n");
10    return 0;
11 }
12
```

*Figura 3: Bloque principal del programa (MAIN).*

#### Observaciones:

- La reserva de memoria realizada para el puntero es posible gracias a la función “malloc” del lenguaje de programación C (Véase en línea 10, Figura 1).
- Para facilitar el detalle del largo de los arreglos y del puntero descrito, se utiliza un macro al comienzo del archivo “funciones.c” (Véase línea 5, Figura 4). Dicho macro, en tiempo de pre-compilación reemplaza, dentro del código, cada aparición de la palabra LARGO por el valor numérico detallado a su derecha, en este caso 1500. Esta implementación permite la rápida modificación del valor clave de las funciones, ya que las funciones especificadas solo soportarán el procesamiento de valores de N, dependiendo de la memoria que se le fue asignada.

```
3 #include <stdio.h>
4 #include <stdlib.h>
5 #define LARGO 1500
6
```

*Figura 4: Detalle de bibliotecas utilizadas y definición de Macro.*

# ANÁLISIS DE LOS RESULTADOS OBTENIDOS

Esta sección se enfocará en comparar los resultados obtenidos con la implementación efectuada con el los objetivos planteados. Primero que todo, vemos que el  $T(n)$  de la función Fibonacci descrita en la figura 2 de este documento, posee una complejidad algorítmica del orden  $O(n)$ .

En cuanto a la ejecución del software, este cumple efectivamente lo solicitado. Se logra la obtención de la serie Fibonacci para valores de N hasta 5000 (Incluso para valores mayores). El techo de la función depende únicamente de la cantidad de memoria reservada y el largo de los Arreglos auxiliares, en síntesis depende del valor especificado en el Macro definido en el archivo “funciones.c”.

En la figura 5, se visualiza la ejecución del software desarrollado. Cabe destacar que la ejecución fue desarrollada en un equipo con sistema operativo basado en UNIX (Ubuntu 14.04 LTS 64bits).

```
joaco@Machine: ~/Dropbox/Universidad/4to Semestre/Analisis de Algoritmos y estructuras de datos/Laboratorios_EDA_2s-2014/Laboratorio_N°1
65649140327510866433945175121615265453613331113140424368548051067658434935238369596534280717687753283482343455736671973139274627362910821067928078471803532913117677892465908993
86354593278945237776744061922403376386740040213303432974969020283281459341882681768389307200363479562311710310129195316979460763273758925353077255237594378843450406771555577905
645044301664011946258097221672975861502696844314695203461493229110597067624326851599284749089128470674086200858713501626031207190317208609408129832158107728207635318662461127824
5537208532365057759564300725177443150515396009051686032203491632226408852488524331580515348496224348482993809057048348244932745373262456775587908918719080366205800959474315005
240253270974699531877072437682590741993963226598414749819360928522394503970716544315642132815768890805878318340491743455627052022356484649519611246026831397097506938264870661326
450766507461151267752274862159864253071129844118262266105716351506926002986170494542504749137811515413994155067125627119713325276363193960690289565028826860836224108205056243070
1794976171121233066073310059947366875
joaco@Machine: ~/Dropbox/Universidad/4to Semestre/Analisis de Algoritmos y estructuras de datos/Laboratorios_EDA_2s-2014/Laboratorio_N°1 ./lab.o 5000
387896845438832563370191638032590531208212771464624510616059721489555013904493709701082291646221066947929345285888297381348310280895498294036143015691147893836421656394410691021
4505634133706558656238254656700712525929908385493381392883637834751890876297071203337305292310769300851809384900180384781399674888176555465378829164426091298038461377896902150229
308247566634622492307188332480328037503913035290330450584270114763524227021093463769910400671417488329842289149127310405432875329804427367682297724498774987455569190770388063704
683279481135897373999311010621930814901857081539785437919530561751076105307568878376603366735544525884488624161921055345749367589784902798823435102359984466393485325641195222185
956306047536464547076033090242080638258492915645287629157591423438091423029174910889841552098544324865940797935713168416928680395433095453880681146650826686289742063932343848
84652409887423958738019769938203171742089322654688793640026307977800587591296713896342142525791168727560036031137054775472460463987588046985178408674382863125
joaco@Machine: ~/Dropbox/Universidad/4to Semestre/Analisis de Algoritmos y estructuras de datos/Laboratorios_EDA_2s-2014/Laboratorio_N°1 ./lab.o 3000
4106158863079712603335683781926710522012510863736925240888543092690558427411340373133049166085004456083003683570694227458856936214547650267437304544685216048606292497360503469
77345373319688740584725529080204908690751262205905454219588975803110922267084927479385953913331837124479554314761107327624006673793408519173181099320170677683893476674778739502
1744702686278209185538422258583064083016618629003582668572382102358025043519514729979196765240047842363764533472683641526483462458405732142414199379172429186062639810097866942392
015404620153851867148112739830574851396421139982713640679581178458198658692285968043243650789796000
joaco@Machine: ~/Dropbox/Universidad/4to Semestre/Analisis de Algoritmos y estructuras de datos/Laboratorios_EDA_2s-2014/Laboratorio_N°1 ./lab.o 30000
232929896380036420661419994380408528529260884115483145991687978247833518703723481336017874956538836683802619774965676414243209558092089669557849196425298731790749447645639478064
39262196755845519711306684973737711042973282528237200937168441069639441943029944471479828556309954918401815303622930867567506539657625021994942965852302768005415378923031914456
4667429507895449740435270210669919977594538071882240298234778483333580247903139990071309652539729204217233879045422744683453159985074501193927574506823386755139468597285509517103
6582140355723560536009040573497122931318666092407244907750182876961042338799663939055336370739082159768612895810362877880144383781438377313578953337902118272066424399802249
54262624983854711697710023946056404910517780908036329118014527497705465662019768338437838513723761644673877045398210853752932200185088674449677149514151002565691912828698622935
08849424251841418989755513861478303262571281496374014738629819645159341072279708705968018746568998016717579318510855948093963205440067193360151451949678858084408646951928768406
3973247694790730821810443367097960000
joaco@Machine: ~/Dropbox/Universidad/4to Semestre/Analisis de Algoritmos y estructuras de datos/Laboratorios_EDA_2s-2014/Laboratorio_N°1 ./lab.o 300000
340758814482702246247775861456947982472416975315967005108042653013235190240877933898448539679515575919743966470872513440607053041036995746101725774478687843091884554486254036
677756947193527917010848466258535442936256810625049220251940512692627244142226674006465510470774492484166651355060781662700388888535742934806108125383306458801433268182514741122
205034581549428705728957977687811213964874857242535957898765463374375571233359425763080588468873779867035392290053932338236290698513823294906450994912712083470477810401482312902
620222023430437897208088049486343030174222051206870407492141761200807492998932346195765196086001778328101093521405116407454348942148137960781398798471240154640005988289912385182
7287455874674432159179028092466403804304109272125640444284884525411295205416430470582486406249934342582063124919946282892089088166763620160256154182460145872001003029672539725
2489617562794836314360398769565156234066763740052945366366462178748117993556689183391830633948906299814448352023836136752075023929191870395465875783010891560247507473558436260
696121478026476279611123367097960000
joaco@Machine: ~/Dropbox/Universidad/4to Semestre/Analisis de Algoritmos y estructuras de datos/Laboratorios_EDA_2s-2014/Laboratorio_N°1 ./lab.o 60000
51290597597710560095383416115694462505657550372719186690459734882037748803844079191682502808439996222810877523569316886439040328537845371571698582707334245989248658517390571478
7275230776737214977907501079332766784451402491325230712603217034100543075817878434714888749417124995577710146858303972500816505973648310814646719757308305648257597442603481175
04310140047589232014737710968785993999326259433742329167827540677164919559017615373957837493586077650654368483064412411033885031824263367384061314066729337037970001509797521
1202652362057580160522917703454276693633240123994520136895210553366299693282672341489137566293727281585436730108288933380143408369980485119241748921114352607717084521471862228
1870316067205867084303077322482804816610147899841053186603905482489874904459726706092410711042360378394401269272808017738663144198789424892760463875899901305445761418509313029
4584528894748770742772042255480164078334251674925073513261384048736006689033689532286082996521040853394584505394683273315884823216678316002660405781214930055149550617970815579
3413808794237002095608806734195920000
joaco@Machine: ~/Dropbox/Universidad/4to Semestre/Analisis de Algoritmos y estructuras de datos/Laboratorios_EDA_2s-2014/Laboratorio_N°1 ./lab.o 100
354224848179261915075
joaco@Machine: ~/Dropbox/Universidad/4to Semestre/Analisis de Algoritmos y estructuras de datos/Laboratorios_EDA_2s-2014/Laboratorio_N°1 ./lab.o 500
139423224561697880139724382870407283950870256587697307264108962948325571622863290691557658876222521294125
joaco@Machine: ~/Dropbox/Universidad/4to Semestre/Analisis de Algoritmos y estructuras de datos/Laboratorios_EDA_2s-2014/Laboratorio_N°1 ./lab.o 1000000
```

Figura 5: Ejecución de (./lab.o = nombreEjecutable)



## CONCLUSIÓN

Existen muchos lenguajes de programación que han surgido a lo largo del tiempo, lenguajes producto de los diferentes paradigmas o simplemente nuevos lenguajes creados para simplificar la vida a los programadores. A pesar de que C es un lenguaje “antiguo”, ofrece una ventaja considerable para quien se está adentrando en la programación, ya que obliga a trabajar a más bajo nivel permitiendo aprender desde la base y a su vez, valorar los avances que se generan con las nuevas tecnologías.

Cabe destacar que en el camino surgieron bastantes problemáticas, comenzando por la interrogante: ¿Como trabajar y operar los datos? Esta interrogante origino muchas disyuntivas, las cuales repercutieron en que la complejidad de algunas funciones se elevara considerablemente, sin perder el fin de encontrar una solución eficiente.

En términos concretos, se logró la implementación efectiva del software solicitado, el cual puede no ser óptimo al compararse con otros algoritmos, pero sin duda es una solución al problema planteado.

## REFERENCIAS

Andrés Marzal, Isabel Garcia. (Año no informado). Introducción a la programación con C. Departamento de lenguajes y sistemas informáticos: Universitat Jaume.

Brian W. Kernighan and Dennis M. Ritchie. (1988). The C programming Language. EEUU: Prentice-Hall.

## ANEXO: CAPTURAS DE PANTALLA CON FONDO BLANCO

```
int *sumaDeValores(int numero1[], int numero2[], int largo)
{
    int indice, acumulador=0;
    int *Array = (int*)malloc(sizeof(int)*largo);
    for(indice=largo-1; indice>=0; indice--)
    {
        int s = numero1[indice] + numero2[indice];
        if(s/10>=1)
        {
            Array[indice] = s % 10;
            acumulador = s / 10;
            numero1[indice-1] = numero1[indice-1] + acumulador;
        }
        else
        {
            Array[indice] = s;
        }
    }
    return Array;
}
```

Figura 1: Implementación de función "sumaDeValores"

```

void fibonacci(int n)
{
    int Valor1[LARGO], Valor2[LARGO], indiceAuxiliar, indice, valor = 1;
    for(indice=0; indice<LARGO; indice++)
    {
        Valor1[indice]=0;
        Valor2[indice]=0;
        if(indice==LARGO-1)
        {
            Valor2[indice]=1;
        }
    }
    for(indiceAuxiliar = 1; indiceAuxiliar < n; indiceAuxiliar++)
    {
        int *Puntero = sumaDeValores(Valor1, Valor2, LARGO);
        for(indice=0;indice<LARGO;indice++)
        {
            Valor1[indice] = Valor2[indice];
            Valor2[indice] = Puntero[indice];
        }
    }
    for(indice=0;indice<LARGO;indice++)
    {
        if(valor==1)
        {
            if (Valor2[indice+1]!=0)
            {
                valor=0;
            }
        }
        else
        {
            printf("%d", Valor2[indice]);
        }
    }
}

```

Figura 2: Implementación de función "fibonacci"

```

#include "funciones.c"

int main(int argc, char **argv)
{
    if(argc==2)
    {
        int n = atoi(argv[1]);
        fibonacci(n);
        printf("\n");
    }
    else
    {
        printf("Debe ingresar el numero N a evaluar como parametro,");
        printf("separado por un espacio del nombre del archivo ejecutable.\n");
    }
    return 0;
}

```

*Figura 3: Bloque principal del programa (MAIN).*

```

#include <stdio.h>
#include <stdlib.h>
#define LARGO 1500

```

•

Figura 4: Detalle de bibliotecas utilizadas y definición de Macro.