**Trabajo Practico N°1**

*Estructuras Dinámicas en*

*C y Assembler:*

*Arboles Binarios de Búsqueda*

**Materia:**

* Organización del computador II (OC2)

**Docente:**

* Carlos Holtman (choltman@ungs.edu.ar)

**Alumnos:**

* Gonzalo Pesado (gonzapesa@gmail.com)
  + LEG: 38990987 / 2014
* Lucas Vargas ([lucasjv92@gmail.com](mailto:lucasjv92@gmail.com))
  + LEG: 37369589 / 2014

**Año 2017**

**Desarrollo del proyecto**

Este trabajo práctico se basó en la creación de un programa capaz de almacenar números (positivos, 0, y negativos) dentro de lo que se conoce como Árbol Binario de Búsqueda, el cual no admite números repetidos, haciendo uso de la memoria dinamica en C y Assembler.

El software está desarrollado en 2 lenguajes de programación: Assembler de Intel IA32 (Lenguaje de código máquina) y C. El entorno de desarrollo y ejecución se realizó bajo sistema operativo Windows de 32 bits.

A su vez, el software debe poder permitir la entrada de números enteros por consola, uno a la vez. Luego almacenarlos en un árbol binario de búsqueda haciendo uso de funciones implementadas en lenguaje ensamblador de IA32, llamandolas desde el código de programación de C. Por último mostrar por pantalla los números del árbol ordenados de menor a mayor, es decir recorrerlo de forma DFS.

Al finalizar el programa, se muestra como queda el árbol luego de borrar todo el subárbol izquierdo/derecho de la raíz (borrara el izquierdo si la raíz posee un nodo izquierdo sino borrara el derecho). En caso de no poseer hijos el programa no borrará ningún subárbol.

El programa a su vez también cuenta con un .bat el cual compila el código fuente (“Main.c” y sus respectivos headers, junto con el OBJ que crea NASM del archivo que posee todas las funciones de Assembler “funcionesASM.asm”). Este .bat mostrará al usuario las rutas de NASM y GCC que utilizará para compilar, en caso de que no sean las rutas correctas, se deberá cambiar la ruta directamente del .bat a la ruta correcta.

Por último, el .bat permitirá ejecutar por 1ra vez el programa una vez compilado.

Como herramientas para desarrollar el mismo se usaron los siguientes:

* Editores: Sublime, Notepad++, Notepad, Wordpad.
* IDE: SASM para testear código de assembler, y Codeblocks para probar código de C.
* Compiladores: NASM para compilar los códigos .asm, y GCC para compilar el código de C y llamar a los objetos de código creados anteriormente por NASM para vincular las funciones de Assembler llamadas desde C.

**Código Fuente: C**

**EnunciadoCorto.c**

1. #include <stdio.h>
2. #include <io.h>
3. #include <stdlib.h>
4. #include "nodo\_abb.h"
5. #include "Funciones.h"
7. **int** main(**int** argc, **char** \*argv[])
8. {
9. **struct** nodo\_abb raiz;
10. raiz.valor = 0;
11. raiz.izq = 0;
12. raiz.der = 0;
13. **int** band = 1;
14. **int** n = 0;
15. **while**(band == 1)
16. {
17. band = scanf("%d\n",&n);
18. **if**(band == 1)
19. {
20. **if**(raiz.valor == 0)
21. {
22. raiz.valor = n;
23. }
24. **else**
25. {
26. agregar\_abb(&raiz,n);
27. }
28. }
29. }
31. mostrar\_arbol\_DFS(&raiz);
33. borrar\_abb(&raiz);
35. **return** 0;
36. }

**nodo\_abb.h**

1. #ifndef NODO\_ABB\_H\_INCLUDED
2. #define NODO\_ABB\_H\_INCLUDED
4. **struct** nodo\_abb
5. {
6. **int** valor;
7. **struct** nodo\_abb \*izq;
8. **struct** nodo\_abb \*der;
9. };
11. #endif // NODO\_ABB\_H\_INCLUDED

**Funciones.h**

1. #ifndef FUNCIONES\_H\_INCLUDED
2. #define FUNCIONES\_H\_INCLUDED
3. #include <string.h>
5. **void** mensajeBienvenida()
6. {
7. printf("Organizacion del computador II, TP1\n");
8. printf("ARBOL BINARIO DE BUSQUEDA CON C Y ASM\n");
9. printf("Docente: Carlos Holtman\n");
10. printf("Alumnos: Lucas Vargas y Gonzalo Pesado\n");
11. printf("\n");
12. }
14. **void** finPrograma()
15. {
16. printf("Presiona una tecla para salir");
17. }
19. **char** \* recibirCadena( **char** \*cadena )
20. {
21. scanf("%9s",cadena);
22. **return** cadena;
23. }
25. **int** esNumero(**char** \* cadena)
26. {
27. **int** i = 0;
28. **while**(cadena[i] != '\0')
29. {
30. **if**( cadena[i] < 48 || cadena[i] > 57)
31. {
32. **if**(i == 0 && cadena[i] == 45) i = i;
33. **else** **return** 0;
34. }
35. i = i+1;
36. }
37. **return** 1;
38. }
40. **int** esSignoPeso(**char** \* cadena)
41. {
42. **if** (strlen(cadena) == 1)
43. **if**(cadena[0] == 36)
44. **return** 1;
45. **return** 0;
47. }
49. **int** stringToInt(**char** \*cadena)
50. {
51. **return** atoi(cadena);
52. }
54. **void** mostrar\_arbol(**struct** nodo\_abb \*nodo){
55. **if**(nodo != 0){
56. **if**(nodo->izq != 0){
57. mostrar\_arbol(nodo->izq);
58. }
59. printf("%d\n",nodo->valor);
60. **if**(nodo->der != 0){
61. mostrar\_arbol(nodo->der);
62. }
63. }
64. }
66. **void** mostrar\_arbol\_DFS(**struct** nodo\_abb \*nodo)
67. {
68. **if**(nodo != 0){
69. **if**(nodo->izq != 0){
70. mostrar\_arbol(nodo->izq);
71. }
72. printf("%d\n",nodo->valor);
73. **if**(nodo->der != 0){
74. mostrar\_arbol(nodo->der);
75. }
76. }
77. }
79. #endif // FUNCIONES\_H\_INCLUDED