

PRIMERA PARTE

Studio Shodwe

```
357  
358 int main() {  
359     ArbolGenealogico arbol;  
360     int opcion;  
361     int id, idRama, idMiembro;  
362     char nombre[50], fecha[20], lugar[50], ocupacion[50];  
363     int resultado;  
364  
365     // Datos iniciales  
366     cout << "=====\\n";  
367     cout << " INICIALIZANDO DATOS DE EJEMPLO\\n";  
368     cout << "=====\\n";  
369  
370     char n1[] = "Emperador_Xul";  
371     char f1[] = "1500-01-01";  
372     char l1[] = "Templo_Central";  
373     char o1[] = "Gobernante";  
374     arbol.agregar(10, n1, f1, l1, o1);  
375  
376     char n2[] = "Sacerdote_Kaal";  
377     char f2[] = "1530-05-12";  
378     char l2[] = "Templo_Norte";  
379     char o2[] = "Sacerdote";  
380     arbol.agregar(5, n2, f2, l2, o2);  
381  
382     char n3[] = "Princesa_Itzel";  
383     char f3[] = "1510-03-19";  
384     char l3[] = "Palacio_Real";  
385     char o3[] = "Nobleza";  
386     arbol.agregar(15, n3, f3, l3, o3);  
387  
388     char n4[] = "Guerrero_Balam";  
389     char f4[] = "1545-08-22";  
390     char l4[] = "Fortaleza";  
391     char o4[] = "Guerrero";  
392     arbol.agregar(3, n4, f4, l4, o4);  
393  
394     char n5[] = "Escriba_Akbal";  
395     char f5[] = "1535-12-05";  
396     char l5[] = "Biblioteca";  
397     char o5[] = "Escriba";  
398     arbol.agregar(7, n5, f5, l5, o5);  
399  
400     cout << "Datos cargados correctamente.\\n";  
401     esperarEnter();  
402
```

Aquí comienza la función principal main().

Se declara un objeto arbol de tipo ArbolGenealogico, el cual representa nuestro árbol binario.

Luego, se definen variables que usaremos más adelante para capturar datos del usuario.

Se insertan cinco miembros de ejemplo usando la función agregar. Cada uno tiene un nombre, fecha, lugar, ocupación e ID único.

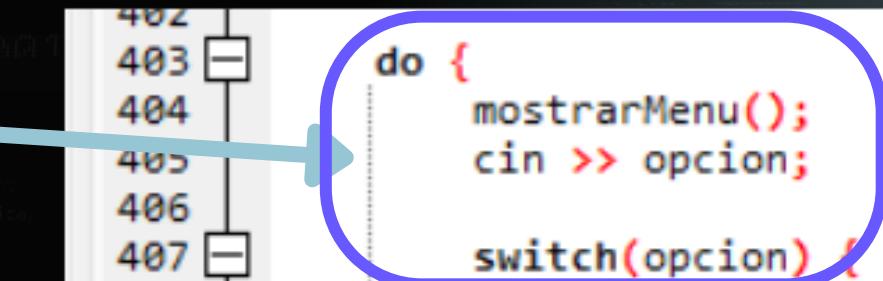


WWW.REALLYGREATSITE.COM



SEGUNDA PARTE

Comienza un bucle do { ... } while(opcion != 0); que se repite hasta que el usuario decida salir.



Opción 1: muestra todos los miembros del árbol con un recorrido "inorden" (ordenado por ID).

Opción 2: permite al usuario agregar un nuevo miembro ingresando sus datos por consola. Luego lo inserta al árbol con `arbol.agregar()`. Pero antes compara con los nodos existentes.

Opción 3: busca un miembro por su ID. Si lo encuentra, muestra los datos; si no, lo indica.

```

408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
    
```

case 1:
`arbol.mostrarTodoEnOrden();`
`esperarEnter();`
`break;`

case 2:
`cout << "\n==== AGREGAR NUEVO MIEMBRO ===\n";`
`cout << "ID: ";`
`cin >> id;`
`cout << "Nombre: ";`
`cin >> nombre;`
`cout << "Fecha (AAAA-MM-DD): ";`
`cin >> fecha;`
`cout << "Lugar: ";`
`cin >> lugar;`
`cout << "Ocupacion: ";`
`cin >> ocupacion;`

`arbol.agregar(id, nombre, fecha, lugar, ocupacion);`
`esperarEnter();`
`break;`

case 3:
`cout << "\n==== BUSCAR MIEMBRO ===\n";`
`cout << "ID a buscar: ";`
`cin >> id;`

`resultado = arbol.buscar(id);`
`if (resultado == 1) {`
 `cout << "Miembro encontrado!\n";`
 `arbol.mostrarMiembro(id);`
`} else {`
 `cout << "Miembro no encontrado.\n";`
`}`
`esperarEnter();`
`break;`



WWW.REALLYGREATSITE.COM



TERCERA PARTE

```
445  
446     case 4:  
447         cout << "\n==> ELIMINAR MIEMBRO ==>\n";  
448         cout << "ID a eliminar: ";  
449         cin >> id;  
450  
451         resultado = arbol.quitar(id);  
452         if (resultado == 1) {  
453             cout << "Miembro eliminado.\n";  
454         } else {  
455             cout << "Miembro no encontrado.\n";  
456         }  
457         esperarEnter();  
458         break;  
459  
460     case 5:  
461         int subopcion;  
462         menuRecorridos();  
463         cin >> subopcion;  
464         switch(subopcion) {  
465             case 1:  
466                 arbol.mostrarTodoPreOrden();  
467                 break;  
468             case 2:  
469                 arbol.mostrarTodoEnOrden();  
470                 break;  
471             case 3:  
472                 arbol.mostrarTodoPostOrden();  
473                 break;  
474             case 4:  
475                 arbol.mostrarTodoPreOrden();  
476                 arbol.mostrarTodoEnOrden();  
477                 arbol.mostrarTodoPostOrden();  
478                 break;  
479             default:  
480                 cout << "Opcion invalida.\n";  
481         }  
482         esperarEnter();  
483         break;
```

Studio Shodwe

Opción 4: elimina un miembro del árbol usando su ID.
Muestra si fue exitoso o no.

Opción 5: despliega un submenú para mostrar el árbol en diferentes recorridos:

PreOrden: raíz → izquierda → derecha.

InOrden: izquierda → raíz → derecha.

PostOrden: izquierda → derecha → raíz.

También puede mostrar los tres en conjunto si se elige la opción 4
del submenú



CUARTA PARTE

WWW.REALLYGREATSITE.COM

Studio Shodwe

```
485  
486     case 6:  
487         cout << "\n==> VERIFICAR PERTENENCIA ==\n";  
488         cout << "ID de la rama: ";  
489         cin >> idRama;  
490         cout << "ID del miembro: ";  
491         cin >> idMiembro;  
492  
493         resultado = arbol.verificarRama(idRama, idMiembro);  
494         if (resultado == 1) {  
495             cout << "El miembro SI pertenece a la rama.\n";  
496         } else {  
497             cout << "El miembro NO pertenece a la rama.\n";  
498         }  
499         esperarEnter();  
500         break;  
501  
502     case 7:  
503         cout << "\n==> MOSTRAR DESCENDIENTES ==\n";  
504         cout << "ID del ancestro: ";  
505         cin >> id;  
506         arbol.mostrarDescendientes(id);  
507         esperarEnter();  
508         break;  
509  
510     case 8:  
511         arbol.mostrarEstadisticas();  
512         esperarEnter();  
513         break;  
514  
515     case 0:  
516         cout << "\nGracias por usar el sistema!\n";  
517         break;  
518  
519     default:  
520         cout << "Opcion invalida.\n";  
521         esperarEnter();  
522     } while (opcion != 0);  
523  
524     return 0;  
525 }
```

Opción 6: verifica si un miembro pertenece a la rama de otro (si es descendiente de ese ancestro).

Opción 7: muestra todos los descendientes (hijos, nietos, etc.) de un miembro.

Opción 8: muestra estadísticas generales del árbol (cantidad de nodos, profundidad, etc.).

Opción 0: sale del programa con un mensaje de despedida.