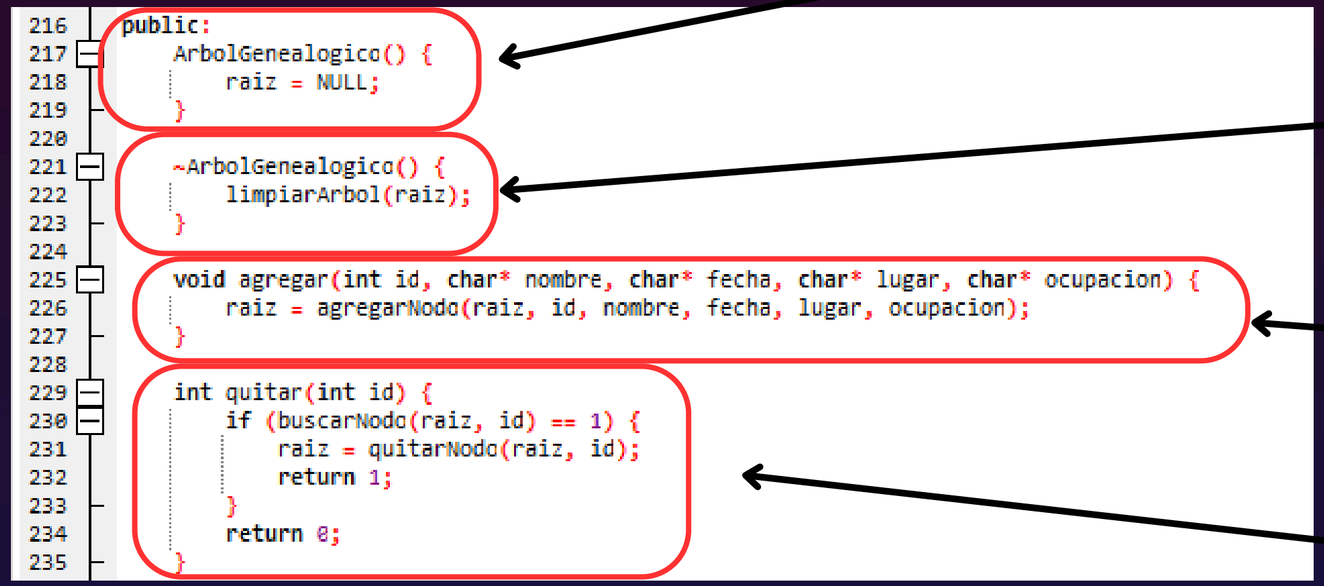
FUNCIONES DEL CÓDIGO - PARTE 3



```
216  public:
217      ArbolGenealogico() {
218          raiz = NULL;
219      }
220
221      ~ArbolGenealogico() {
222          limpiarArbol(raiz);
223      }
224
225      void agregar(int id, char* nombre, char* fecha, char* lugar, char* ocupacion) {
226          raiz = agregarNodo(raiz, id, nombre, fecha, lugar, ocupacion);
227      }
228
229      int quitar(int id) {
230          if (buscarNodo(raiz, id) == 1) {
231              raiz = quitarNodo(raiz, id);
232              return 1;
233          }
234          return 0;
235      }
```

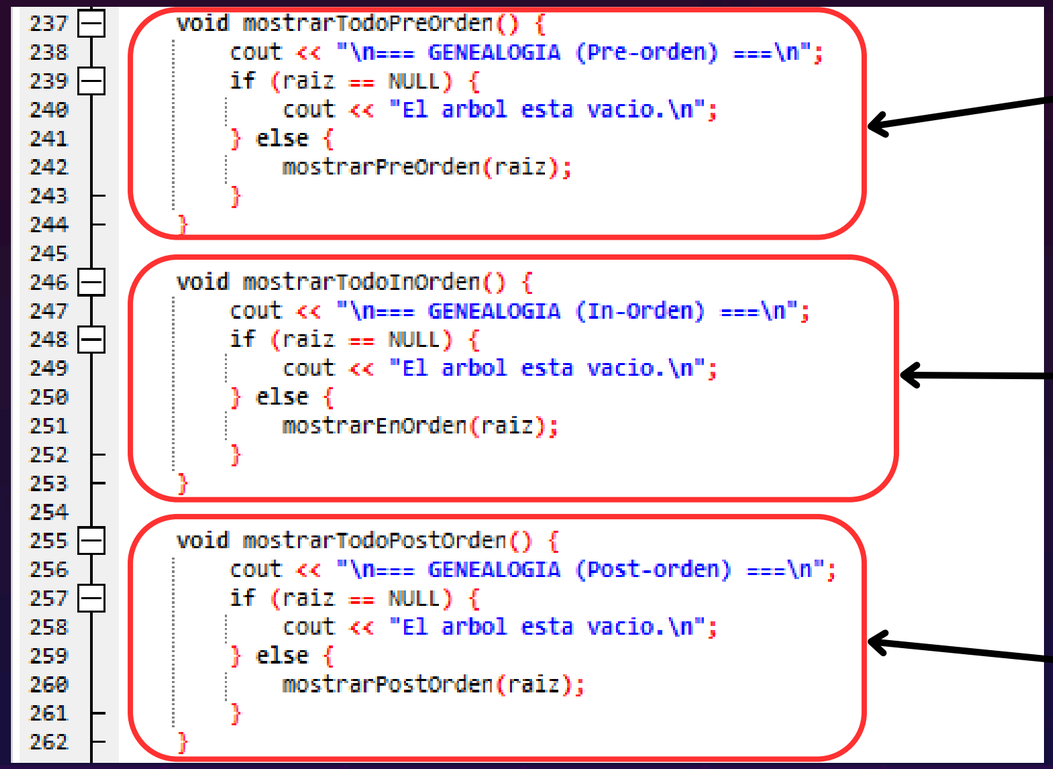
Comenzamos creando el árbol vacío e iniciamos sin nadie dentro de él.

Cuando terminamos de usar el árbol, borramos todos los datos de memoria para no dejar basura.

Pasamos los datos del nuevo miembro y lo ponemos en el árbol

Esta parte nos permite eliminar un miembro del árbol. Primero revisamos si existe, si lo encontramos, lo borramos. Si no existe, devuelve 0 para avisar que no se eliminó nada.

RECORRIDOS DEL ÁRBOL



```
237  void mostrarTodoPreOrden() {
238      cout << "\n=== GENEALOGIA (Pre-orden) ===\n";
239      if (raiz == NULL) {
240          cout << "El arbol esta vacio.\n";
241      } else {
242          mostrarPreOrden(raiz);
243      }
244  }
245
246  void mostrarTodoInOrden() {
247      cout << "\n=== GENEALOGIA (In-Orden) ===\n";
248      if (raiz == NULL) {
249          cout << "El arbol esta vacio.\n";
250      } else {
251          mostrarEnOrden(raiz);
252      }
253  }
254
255  void mostrarTodoPostOrden() {
256      cout << "\n=== GENEALOGIA (Post-orden) ===\n";
257      if (raiz == NULL) {
258          cout << "El arbol esta vacio.\n";
259      } else {
260          mostrarPostOrden(raiz);
261      }
262  }
```

Aquí mostramos todo el árbol en PreOrden: primero se mira el papá, luego la rama izquierda y luego la rama derecha. Si está vacío, mostramos un mensaje para avisar.

Esta función enseña los miembros ordenados de menor a mayor ID, revisando primero la rama izquierda, después el papá, y finalmente la rama derecha.

Aquí miramos primero los hijos, y al final el papá. Esto es bueno cuando queremos ver las generaciones jóvenes antes que las antiguas.

ESTADO DEL ÁRBOL

```
264  int buscar(int id) {
265      return buscarNodo(raiz, id);
266  }
267
268  void mostrarMiembro(int id) {
269      Nodo* nodo = obtenerNodo(raiz, id);
270      if (nodo != NULL) {
271          cout << "\n=== INFORMACION DEL MIEMBRO ===\n";
272          mostrarDatos(nodo);
273      } else {
274          cout << "Miembro con ID " << id << " no encontrado.\n";
275      }
276  }
277
278  int verificarRama(int idRama, int idMiembro) {
279      Nodo* nodoRama = obtenerNodo(raiz, idRama);
280
281      if (nodoRama == NULL) {
282          cout << "La rama con ID " << idRama << " no existe.\n";
283          return 0;
284      }
285
286      return buscarNodo(nodoRama, idMiembro);
287  }
```

Esta función nos permite saber si existe un miembro con un ID específico. Devuelve 1 si lo encuentra, o 0 si no existe.

Esta parte permite mostrar toda la información de un miembro, como su nombre, fecha de nacimiento, lugar y ocupación, usando su ID para buscarlo.

Esta función revisa si un miembro pertenece a la familia de otro miembro, es decir, si está dentro de su rama, buscando en el árbol a partir de ese ancestro.

VISUALIZACIÓN DEL ÁRBOL

```
289  void mostrarDescendientes(int idAncestro) {
290      Nodo* nodo = obtenerNodo(raiz, idAncestro);
291      if (nodo != NULL) {
292          cout << "\n=== DESCENDIENTES DE " << nodo->nombre << " ===\n";
293          cout << "Rama izquierda:\n";
294          mostrarRama(nodo->izquierda);
295          cout << "Rama derecha:\n";
296          mostrarRama(nodo->derecha);
297      } else {
298          cout << "Ancestro con ID " << idAncestro << " no encontrado.\n";
299      }
300  }
301
302  void mostrarEstadisticas() {
303      cout << "\n=== ESTADISTICAS DEL ARBOL ===\n";
304      cout << "Total de miembros: " << contarMiembros(raiz) << "\n";
305      cout << "Profundidad del arbol: " << calcularProfundidad(raiz) << "\n";
306      if (raiz == NULL) {
307          cout << "Estado: Vacio\n";
308      } else {
309          cout << "Estado: Con datos\n";
310      }
311      cout << "\n";
312  }
313
314  int estaVacio() {
315      if (raiz == NULL) {
316          return 1;
317      } else {
318          return 0;
319      }
320  }
321 };
```

Con esta función podemos ver todos los descendientes de una persona, separando la rama izquierda y la rama derecha para entender mejor cómo está distribuida la familia.

Esta función nos da un resumen con cuántos miembros hay en el árbol y qué tan profundo es, para saber cuántas generaciones tiene. También nos dice si el árbol está vacío o ya tiene información cargada.

Esta parte nos sirve para verificar si todavía no hemos puesto ningún miembro en el árbol, devolviendo 1 si está vacío, o 0 si ya hay datos guardados.

MOSTRAR DIVERSOS MENÚS

```
323  void mostrarMenu() {
324      cout << "\n========================================\n";
325      cout << "  ARBOL GENEALOGICO - CIVILIZACION\n";
326      cout << "========================================\n";
327      cout << "1. Mostrar arbol completo\n";
328      cout << "2. Agregar nuevo miembro\n";
329      cout << "3. Buscar miembro por ID\n";
330      cout << "4. Eliminar miembro\n";
331      cout << "5. Mostrar recorridos\n";
332      cout << "6. Verificar pertenencia a rama\n";
333      cout << "7. Mostrar descendientes\n";
334      cout << "8. Estadisticas del arbol\n";
335      cout << "0. Salir\n";
336      cout << "========================================\n";
337      cout << "Opcion: ";
338  }
339
340  void menuRecorridos() {
341      cout << "\n=== TIPOS DE RECORRIDO ===\n";
342      cout << "1. Pre-orden\n";
343      cout << "2. En-orden (ordenado)\n";
344      cout << "3. Post-orden\n";
345      cout << "4. Todos los recorridos\n";
346      cout << "Opcion: ";
347  }
348
349  void esperarEnter() {
350      cout << "\nPresione Enter para continuar...";
351      char c;
352      do {
353          c = cin.get();
354      } while (c != '\n');
355  }
```

Esta función se encarga de enseñar al usuario todas las opciones disponibles para trabajar con el árbol genealógico.

Esta función muestra un submenú que sirve para escoger el tipo de recorrido que queremos hacer en el árbol genealógico.

La siguiente función sirve para hacer una pausa en el programa y darle tiempo al usuario de leer lo que aparece en la pantalla antes de seguir. Le muestra un pequeño mensaje que dice “Presione Enter para continuar...”