

Universidad San Carlos de Guatemala
Facultad de Ingeniería -FIUSAC-
Escuela de Ciencias y Sistemas
Estructura de Datos "A"



FIUSAC
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Proyecto

SISTEMA DE GESTIÓN DE AEROPUERTO

MANUAL TÉCNICO

Angel Samuel González Velásquez - 202200263

Catedrático: Ing. Rene Ornelis
Auxiliar: Daniel Monterroso

30 de junio de 2024

Índice

Índice.....	2
Interfaz gráfica.....	3
Analizador.....	4
Traducción.....	6
Reportes.....	8
Autómata Finito.....	10

Introducción

El desarrollo de software debe entenderse más que como un simple código, al íntegro y completo proceso que implica desde la generación de ideas, el análisis, el diseño y creación, hasta la propia implementación y verificación de su correcta funcionalidad y compatibilidad con cualquier sistema. Además, para mejorar la forma en la que se desarrolla y crear un software o programa, hay que tomar en cuenta las necesidades que vayan a surgir a lo largo de este proceso, por lo que las metodologías modernas y eficientes son las que nos ayudarán más.

Este manual describe de forma técnica el proyecto realizado para el curso Estructura de Datos, que trató la continuación del desarrollo de un programa con ayuda del lenguaje de programación C++ el cual permitirá tomar archivos de entrada de tipo JSON y TXT, para poder manejar, recopilar y ordenar la información de un aeropuerto y simular su funcionamiento y flujo. Para lograr esto se implementaron clases o datos abstractos para poder manejar los datos obtenidos y así llegar a la solución, la cual debe ser la generación de reportes con ayuda de la herramienta Graphviz y la posibilidad de buscar la información. Como se ha mencionado, este proyecto es la continuación de la práctica, en donde se han tenido que implementar nuevas y más complejas estructuras de datos, como un árbol binario de búsqueda, un árbol B, un grafo representado con una matriz de adyacencia y una tabla de dispersión, o Tabla Hash.



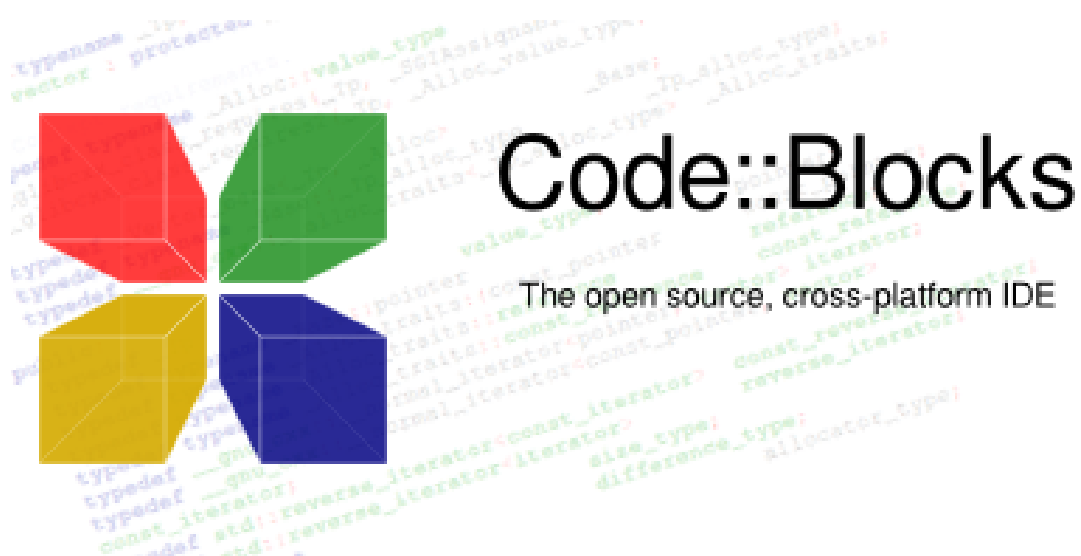
Desarrollo del programa

Sistema y software utilizado

El sistema operativo utilizado para el desarrollo de este programa es Microsoft Windows 11 Home Single, en su versión específica de OS Build 22H2.2314. Además, como ya se mencionó anteriormente, el lenguaje de programación utilizado fue C++, y para poder utilizar este lenguaje de programación se utilizó el entorno de desarrollo integrado Code::Blocks. Este es un completo entorno de desarrollo integrado (IDE) y editor de código para el lenguaje de programación C/C++. Soporta múltiples compiladores, como el GCC, Clang y Visual C++. Con las siguientes especificaciones:

- TDM-GCC 9.2.0 32/64bit
- Compatibilidad con los compiladores basados en CCG
- Depuración integrada (usando BGF)
- Perfil de GPROF
- Gerente de proyecto
- Editor de resaltado de sintaxis personalizable
- Navegador de clases
- Terminación de código
- Code Insight

Asimismo, para realizar el diseño de los diagramas definidos, que forman parte de la funcionalidad y solución del programa se utilizó el paquete Graphviz, el cual es un conjunto de herramienta de software que precisamente sirve para este motivo, se implementa a través del lenguaje descriptivo DOT.



Dependencias instaladas

Las librerías y dependencias instaladas para el buen funcionamiento del programa fueron `jsoncpp` y también distintas librerías integradas ya en el lenguaje C++, como lo puede ser `fstream`, lo que permitía el manejo de archivos, para la lectura y escritura de los mismos. De igual forma, la librería `stringstream`, `sstream`, `fstream` y `string`, para manejar las cadenas dentro del programa.

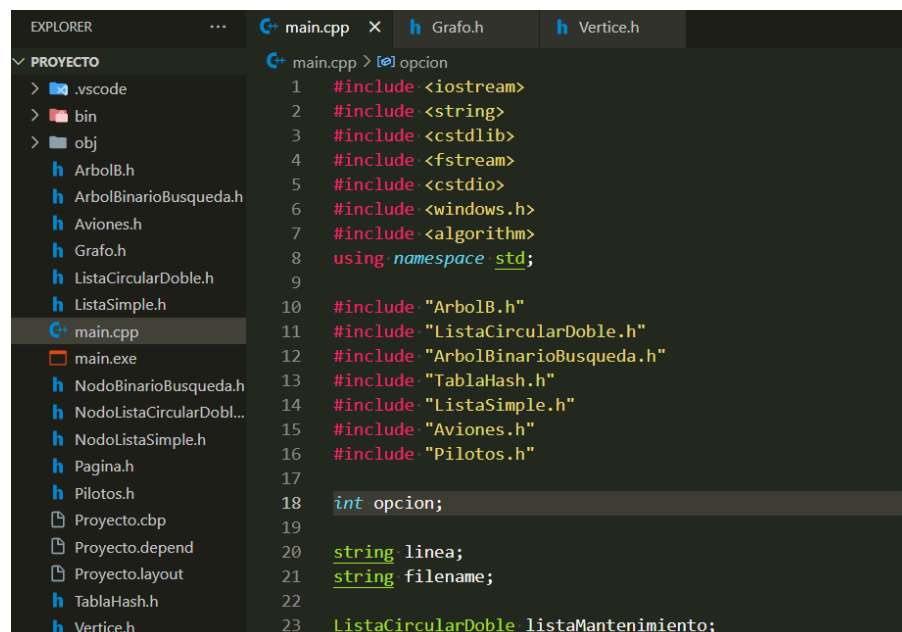
Como en este proyecto se desarrollaron estructuras de datos más sofisticadas y complejas, para algunas de estas se permitió el uso de arreglos estáticos, o hasta la utilización de la librería `vector`, como lo puede ser la tabla de dispersión y el árbol B.

Finalmente, para la creación de los reportes se utilizó la librería `graphviz` y también la instancia `system` dentro de los encabezados de los archivos fuente, esto para poder generar y abrir las imágenes de los reportes.

Clases y estructuras de Datos

Para poder tratar la información recibida de los archivos de entrada se tuvo que implementar distintas clases y estructuras de Datos, desde cero, debido a que no era posible el uso de librerías o estructuras de Datos propias del lenguaje. Por lo que las clases creadas fueron las siguientes:

- Aviones
- Pilotos
- ArbolB
- Pagina
- ArbolBinarioBusqueda
- NodoBinarioBusqueda
- Grafo
- Vertice
- ListaCircularDoble
- NodoListaCircularDoble
- ListaSimple
- NodoListaSimple
- TablaHash



```
EXPLORER    ...    C++ main.cpp X    h Grafo.h    h Vertice.h

PROYECTO
  > .vscode
  > bin
  > obj
    h ArbolB.h
    h ArbolBinarioBusqueda.h
    h Aviones.h
    h Grafo.h
    h ListaCircularDoble.h
    h ListaSimple.h
  C++ main.cpp
    main.exe
    h NodoBinarioBusqueda.h
    h NodoListaCircularDobl...
    h NodoListaSimple.h
    h Pagina.h
    h Pilotos.h
    Proyecto.cbp
    Proyecto.depend
    Proyecto.layout
    h TablaHash.h
    h Vertice.h

C++ main.cpp > [O] opcion
1  #include <iostream>
2  #include <string>
3  #include <cstdlib>
4  #include <fstream>
5  #include <cstdio>
6  #include <windows.h>
7  #include <algorithm>
8  using namespace std;
9
10 #include "ArbolB.h"
11 #include "ListaCircularDoble.h"
12 #include "ArbolBinarioBusqueda.h"
13 #include "TablaHash.h"
14 #include "ListaSimple.h"
15 #include "Aviones.h"
16 #include "Pilotos.h"
17
18 int opcion;
19
20 string linea;
21 string filename;
22
23 ListaCircularDoble listaMantenimiento;
```

Árbol B

```
h ArbolB.h > ArbolB > generarDot(Pagina *, ofstream &, int &)
1  #ifndef ARBOLB_H_INCLUDED
2  #define ARBOLB_H_INCLUDED
3
4  #include <iostream>
5  #include <fstream>
6  using namespace std;
7
8  #include "Pagina.h"
9  #include "ListaCircularDoble.h"
10
11 class ArbolB
12 {
13 private:
14     Pagina *raiz;
15
16 > void insertarNoLleno(Pagina *nodo, const Avion &avion) ...
48
49 > void dividirHijo(Pagina *nodo, int i) ...
69
70 > void generarDot(Pagina *nodo, ofstream &archivo, int &contador) ...
92
93 public:
94     ArbolB() : raiz(nullptr) {}
95
96 > void insertar(Avion &avion) ...
119
120 > void generarReporte() ...
142
143 };
144 #endif // ARBOLB_H_INCLUDED
```

```
h Pagina.h > Pagina > hijos
1  #ifndef PAGINA_H_INCLUDED
2  #define PAGINA_H_INCLUDED
3
4  #include <iostream>
5  #include <vector>
6  using namespace std;
7
8  #include "Aviones.h"
9
10 class Pagina
11 {
12 public:
13     bool esHoja;
14     vector<Avion> aviones;
15     vector<Pagina *> hijos;
16
17     Pagina(bool hoja) : esHoja(hoja)
18     {
19         aviones.reserve(4); // Máximo 4 aviones por nodo (orden 5)
20         hijos.reserve(5);   // Máximo 5 hijos por nodo (orden 5)
21     }
22 };
23
24 #endif // PAGINA_H_INCLUDED
```

Árbol Binario Búsqueda

```
h ArbolBinarioBusqueda.h > eliminarNodo(string, NodoBB *)
6  #include <sstream>
7  #include <cstdlib>
8  using namespace std;
9
10 #include "Pilotos.h"
11 #include "NodoBinarioBusqueda.h"
12
13 class ArbolBB
14 {
15 private:
16     ··NodoBB *raiz;
17     ··ofstream archivo;
18     ··string nodoDato;
19
20 public:
21     ··int recorrido = 0;
22     ··ArbolBB();
23     ··bool estaVacio();
24
25     ··void insertar(Piloto *dato);
26     ··NodoBB *insertarNodo(Piloto *dato, NodoBB *nodoPtr);
27
28     ··void eliminar(string numeroId);
29     ··NodoBB *eliminarNodo(string numeroId, NodoBB *nodoPtr);
30
31     ··void buscar(Piloto *dato);
32     ··string buscarNodo(Piloto *dato, NodoBB *nodoPtr);
33     ··NodoBB *buscarMin(NodoBB *nodoPtr);
34
35     ··void RecorridoIn();
36     ··void RecorridoIn(NodoBB *nodoPtr);
37     ··void RecorridoPre();
38     ··void RecorridoPre(NodoBB *nodoPtr);
39     ··void RecorridoPost();
40     ··void RecorridoPost(NodoBB *nodoPtr);
41
42     ··void generarReporte();
43     ··void imprimirNodo(NodoBB *nodoPtr);
```

```
3
4  #include <iostream>
5  #include <sstream>
6  using namespace std;
7
8  #include "Pilotos.h"
9
10 class NodoBB
11 {
12 private:
13     ··Piloto *piloto;
14     ··NodoBB *izq;
15     ··NodoBB *der;
16
17 public:
18     ··NodoBB(Piloto *piloto);
19     ··void setDato(Piloto *piloto);
20     ··void setIzq(NodoBB *izq);
21     ··void setDer(NodoBB *der);
22     ··NodoBB *getIzq();
23     ··NodoBB *getDer();
24     ··string getNombre();
25     ··string getNumeroId();
26     ··void setNumeroId(string numeroId);
27     ··string getVuelo();
28     ··int getHorasVuelo();
29     ··string graficarDatosArbolBinario();
30     ··~NodoBB();
31 };
```

Tabla Hash

```
h TablaHash.h > eliminar(string)
4 #include <iostream>
5 #include <fstream>
6 #include <sstream>
7 #include <cstdlib>
8 using namespace std;
9
10 #include "ListaSimple.h"
11 #include "NodoListaSimple.h"
12
13 class TablaHash
14 {
15 private:
16     int tamTabla;
17     int numElementos;
18     ListaSimple *tabla;
19
20 public:
21     TablaHash();
22     int clave(string numeroId);
23     void insertar(string numeroId);
24     void eliminar(string numeroId);
25     void generarReporte();
26     void imprimirTabla();
27     ~TablaHash();
28 };
29
30 > TablaHash::TablaHash() ...
36
37 > int TablaHash::clave(string numeroId) ...
56
57 > void TablaHash::eliminar(string numeroId) ...
65
66 > void TablaHash::insertar(string numeroId) ...
74
75 > void TablaHash::imprimirTabla() ...
83
84 > void TablaHash::generarReporte() ...

int TablaHash::clave(string numeroId)
{
    if (!numeroId.empty())
    {
        int valorAscii = static_cast<int>(numeroId[0]);
        int sumaDigitos = 0;
        for (size_t i = 1; i < numeroId.length(); ++i)
        {
            if (isdigit(numeroId[i]))
            {
                sumaDigitos += numeroId[i] - '0';
            }
        }
        int resultado = valorAscii + sumaDigitos;
        int j = (int)(resultado % tamTabla);
        return j;
    }
    return -1;
}
```


Grafo

```
h Grafo.h > ...
1  #include <iostream>
2  using namespace std;
3
4  #include "Vertice.h"
5
6  class Grafo
7  {
8  private:
9      int numVertices;
10     int maxVertices;
11     Vertice *vertices;
12     int **matrizAdy;
13
14 public:
15     Grafo();
16     Grafo(int max);
17     int getNumVertices();
18     void setNumVertices(int n);
19     void nuevoVertice(string nombre);
20     int existeVertice(string nombre);
21     void nuevoArco(string nombre1, string nombre2);
22     void imprimirMatriz();
23     ~Grafo();
24 };
25
26 Grafo::Grafo()
27 {
28 }
29
30 Grafo::Grafo(int max)
31 {
32     numVertices = 0;
33     maxVertices = max;
34     vertices = new Vertice[max];
35     matrizAdy = new int *[max];
36     for (int i = 0; i < max; i++)
37     {
```

```
h Vertice.h > graficarDatos()
1  #include <iostream>
2  #include <sstream>
3  using namespace std;
4
5  class Vertice
6  {
7  private:
8      string nombre;
9      int numVertice;
10
11 public:
12     Vertice();
13     Vertice(string nombre, int n);
14     string graficarDatos();
15     ~Vertice();
16 };
17
18 > Vertice::Vertice() ...
21
22 > Vertice::Vertice(string nombre, int n) ...
27
28 > string Vertice::graficarDatos() ...
34
35 > Vertice::~~Vertice() ...
```

Clases

```
h Aviones.h > Avion > Avion()
1  #ifndef AVIONES_H_INCLUDED
2  #define AVIONES_H_INCLUDED
3
4  #include <iostream>
5  #include <string>
6  using namespace std;
7
8  class Avion
9  {
10 public:
11     Avion();
12     ~Avion(string vuelo, string numeroRegistro, string modelo, int capacidad, string aerolinea, string ciudadDestino, string estado);
13     string vuelo;
14     string numeroRegistro;
15     int getClave() const;
16     string modelo;
17     int capacidad;
18     string aerolinea;
19     string ciudadDestino;
20     string estado;
21     ~Avion();
22 };
23
24 Avion::Avion()
25 {
26 }
27
28 Avion::~Avion(string vuelo, string numeroRegistro, string modelo, int capacidad, string aerolinea, string ciudadDestino, string estado)
29 {
30     this->vuelo = vuelo;
31     this->numeroRegistro = numeroRegistro;
32     this->modelo = modelo;
33     this->capacidad = capacidad;
34     this->aerolinea = aerolinea;
35     this->ciudadDestino = ciudadDestino;
36     this->estado = estado;
37 }
```

```
h Pilotos.h > Piloto > tipoLicencia
1  #ifndef PILOTOS_H_INCLUDED
2  #define PILOTOS_H_INCLUDED
3
4  #include <iostream>
5  #include <string>
6  using namespace std;
7
8  class Piloto
9  {
10 public:
11     Piloto();
12     ~Piloto(string nombre, string nacionalidad, string numeroId, string vuelo, int horasVuelo, string tipoLicencia);
13     string nombre;
14     string nacionalidad;
15     string numeroId;
16     string vuelo;
17     int horasVuelo;
18     string tipoLicencia;
19     ~Piloto();
20 };
21
22 Piloto::Piloto()
23 {
24 }
25
26 Piloto::~Piloto(string nombre, string nacionalidad, string numeroId, string vuelo, int horasVuelo, string tipoLicencia)
27 {
28     this->nombre = nombre;
29     this->nacionalidad = nacionalidad;
30     this->numeroId = numeroId;
31     this->vuelo = vuelo;
32     this->horasVuelo = horasVuelo;
33     this->tipoLicencia = tipoLicencia;
34 }
35
36 Piloto::~~Piloto()
37 {
38 }
```

Menú Inicial

```
363 int main()
364 {
365     int opcion;
366     do
367     {
368         mostrarMenu();
369         cin >> opcion;
370
371         switch (opcion)
372         {
373             case 1:
374             {
375                 cout << "Ingrese la ruta del archivo JSON para la carga de aviones: ";
376                 string rutaAviones;
377                 cin >> rutaAviones;
378                 cargarAviones(rutaAviones, arbolBDisponible, listaMantenimiento);
379             }
380             break;
381             case 2:
382             {
383                 cout << "Ingrese la ruta del archivo JSON para la carga de pilotos: ";
384                 string rutaPilotos;
385                 cin >> rutaPilotos;
386                 cargarPilotos(rutaPilotos, arbolBinarioPilotos, tablaPilotos);
387             }
388             break;
389             case 3:
390             {
391                 cout << "Ingrese la ruta del archivo para la carga de rutas: ";
392                 string rutaRutas;
393                 cin >> rutaRutas;
394                 cargarRutas(rutaRutas);
395             }
396             break;
397             case 4:
398             {
399
400             break;
401             case 5:
402             {
403                 mostrarMenuRecorridos();
404             }
405             break;
406             case 6:
407             {
408                 // Recomendar ruta
409             }
410             break;
411             case 7:
412             {
413                 generarReportes(arbolBinarioPilotos, tablaPilotos);
414             }
415             break;
416             case 8:
417             {
418                 cout << "Saliendo..." << endl;
419                 exit(EXIT_SUCCESS);
420             }
421             break;
422             default:
423             {
424                 cout << " -- La opcion elegida no es valida. \n"
425                 << endl;
426                 Sleep(600);
427             }
428             break;
429         }
430         cin.clear();
431     } while (opcion != 8);
432     return 0;
433 }
```