Universidad San Carlos de Guatemala
Facultad de Ingenieria
Ingenieria en Ciencias y Sistemas

INFORME DE DESARROLLO

PRACTICA UNICA

Curso: Lenguajes Formales y de Programacion
Practica Unica - 1S 2026
Nombre: Bily Estuardo Vallecidos Folgar
Carnet: 202401753
Seccion: B-
Fecha: Febrero 2026

1. Introduccion

El presente informe detalla el proceso de implementacion de la Practica Unica del curso de Lenguajes Formales y de Programacion. El objetivo fue desarrollar un sistema en C++ que simulara las fases iniciales de un compilador: lectura de archivos, separacion y validacion de datos, procesamiento estadistico y generacion de reportes analiticos.

La practica consistio en construir una aplicacion de consola capaz de leer tres archivos con datos academicos, relacionar la informacion entre ellos, calcular estadisticas descriptivas avanzadas y presentar los resultados en cinco reportes HTML con formato profesional.

2. Descripcion de la Implementacion

2.1 Arquitectura General

El programa se organiza en un unico archivo fuente C++ con las siguientes secciones bien definidas:

- Estructuras de datos (structs): Estudiante, Curso, Nota
- Vectores globales para almacenamiento en memoria
- Funciones auxiliares: trim() y split()
- Funciones de carga de archivos: cargarEstudiantes(), cargarCursos(), cargarNotas()
- Funciones de reportes: reporte1() hasta reporte5()
- Funcion de menu: mostrarMenu()
- Funcion principal main() con ciclo do-while y manejo de excepciones

2.2 Lectura y Procesamiento de Archivos

La lectura de archivos se implemento usando ifstream de la libreria fstream. Cada archivo se lee linea por linea con getline(). La separacion de campos se realiza con la funcion split() que usa istringstream para tokenizar cada linea usando la coma como delimitador.

La funcion trim() complementa el proceso limpiando caracteres no visibles como espacios, retornos de carro y saltos de linea que pueden aparecer segun el sistema operativo o la codificacion del archivo.

```
// Ejemplo de procesamiento de una linea de estudiantes.lfp
vector<string> datos = split(linea, ',');
Estudiante e;
e.carnet   = stoi(datos[0]);
e.nombre   = datos[1];
e.apellido = datos[2];
e.carrera  = datos[3];
e.semestre = stoi(datos[4]);
```

2.3 Calculos Estadisticos

Se implementaron los siguientes calculos estadisticos para el Reporte 1:

Calculo	Formula aplicada
Promedio	suma de notas / cantidad de notas
Nota maxima	Iteracion lineal buscando el mayor valor
Nota minima	Iteracion lineal buscando el menor valor
Desviacion estandar	raiz(suma((nota - promedio)^2) / n)
Mediana	Ordenar notas y tomar el valor central (o promedio de los dos centrales si n es par)

2.4 Generacion de Reportes HTML

Los cinco reportes se generan escribiendo directamente cadenas HTML en archivos .html usando ofstream. Cada reporte incluye estilos CSS embebidos para lograr un formato profesional con tablas, colores y tipografia clara.

El Reporte 3 incluye resaltado especial con colores dorado, plateado y bronce para los tres primeros lugares. El Reporte 4 colorea el porcentaje de reprobacion en rojo, naranja o verde segun el nivel de riesgo.

2.5 Menu Interactivo con Manejo de Excepciones

El menu principal utiliza un ciclo do-while que se repite hasta que el usuario selecciona la opcion 9. Se implemento un bloque try-catch para capturar entradas invalidas: si el usuario escribe letras o simbolos, el programa muestra un mensaje de error y regresa al menu sin cerrarse.

```
try {
    string entrada;
    cin >> entrada;
    for (char c : entrada)
        if (!isdigit(c)) throw invalid_argument("");
    opcion = stoi(entrada);
} catch (invalid_argument&) {
    cout << "ERROR: Ingrese un numero valido.\n";
    opcion = 0;
}
```

3. Retos Tecnicos Encontrados y Soluciones

Reto 1: Problema con BOM en archivos .lfp

Descripcion del problema: Al crear los archivos .lfp con el Bloc de notas de Windows, el sistema operativo agrega automaticamente una marca de byte al inicio del archivo conocida como BOM (Byte Order Mark): los bytes 0xEF 0xBB 0xBF. Esto causaba que el primer campo de cada linea fuera interpretado incorrectamente, resultando en errores de conversion con stoi() y en que solo se cargaran 0 o 2 registros en lugar de todos.

Diagnostico: Se agrego codigo de depuracion que imprimia cada caracter de cada linea con su valor numerico entre corchetes. Esto permitio identificar los bytes [239][187][191] al inicio de la primera linea, que corresponden al BOM en decimal.

Solucion aplicada: Se implemento deteccion del BOM al inicio de cada linea y eliminacion de los primeros 3 bytes cuando se detecta. Adicionalmente, la solucion definitiva fue recrear los archivos .lfp directamente desde Visual Studio usando la opcion 'Guardar con codificacion' seleccionando 'Unicode (UTF-8 sin firma)', eliminando el BOM por completo.

Reto 2: Ubicacion del directorio de trabajo

Descripcion del problema: Al ejecutar el programa desde Visual Studio, el directorio de trabajo no era el mismo que la carpeta donde se encontraba el ejecutable. Esto causaba que el programa no encontrara los archivos .lfp aunque estuvieran en la carpeta correcta.

Diagnostico: Se agrego temporalmente system("cd") al inicio del main() para imprimir el directorio actual. Esto revelo que el programa buscaba los archivos en la carpeta raiz del proyecto y no en x64\Debug\.

Solucion aplicada: Se identifico que Visual Studio ejecuta el programa desde la carpeta del proyecto (donde esta el .vcxproj). Se copiaron los archivos .lfp a esa carpeta y tambien a x64\Debug\ para cubrir ambos casos de ejecucion.

Reto 3: Cargar cursos con nombres que tienen espacios

Descripcion del problema: El nombre de algunos cursos contiene espacios, por ejemplo 'Lenguajes Formales y de Programacion'. Inicialmente se temia que esto pudiera causar problemas al separar por comas.

Solucion aplicada: Como el delimitador es la coma y no el espacio, los nombres con espacios se leen correctamente. La funcion split() con getline() y delimitador ',' maneja este caso sin modificaciones adicionales.

Reto 4: Reportes 2 y 3 sin datos de nombres

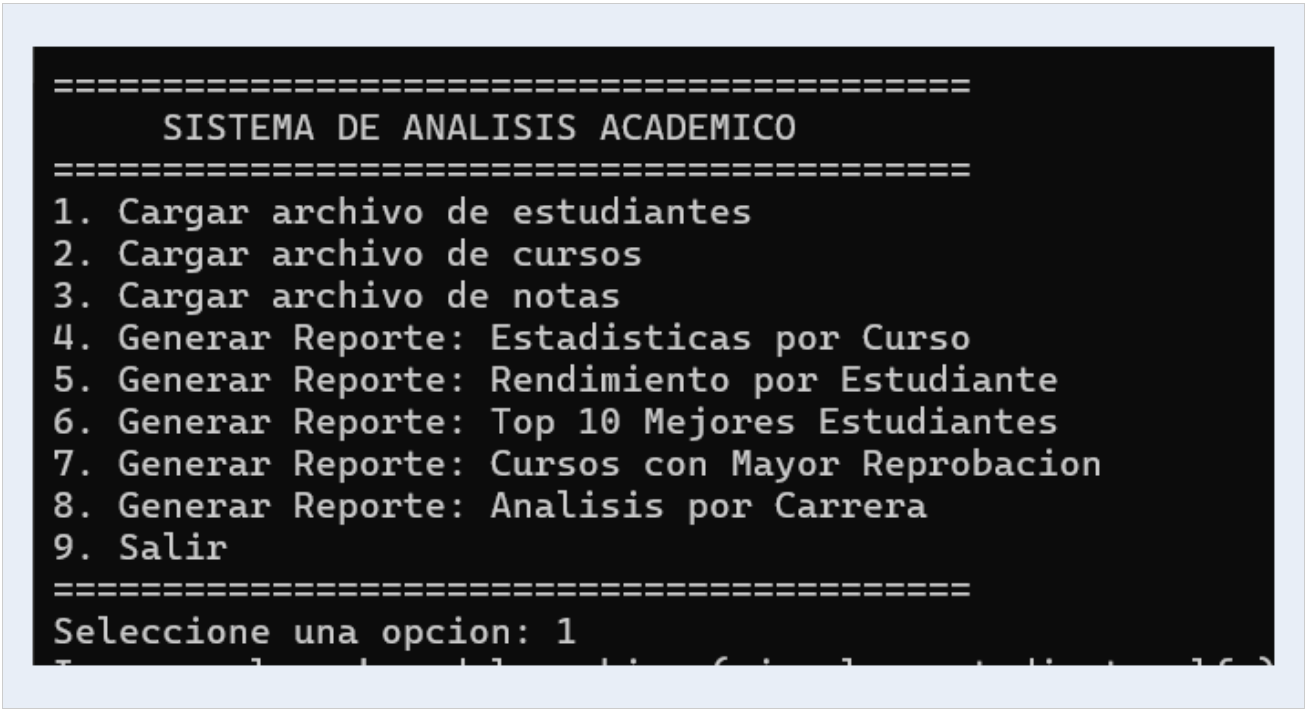
Descripcion del problema: Los reportes 2 y 3 mostraban numeros en la columna de nombres en lugar de los nombres reales de los estudiantes. Esto se debia a que los carnets en notas.lfp no coincidian con los de estudiantes.lfp por el mismo problema del BOM que corrompia la lectura.

Solucion aplicada: Al resolver el problema del BOM (Reto 1) y recrear los archivos correctamente, los carnets coincidieron y los nombres se mostraron correctamente en todos los reportes.

4. Resultados Obtenidos

4.1 Menu Principal en Ejecucion

El siguiente es el aspecto del menu principal al ejecutar el programa:



4.2 Carga de Archivos

Al cargar los tres archivos correctamente el sistema confirma la cantidad de registros procesados:

```
========================================
Seleccione una opcion: 1
Ingrese el nombre del archivo (ejemplo: estudiantes.lfp): estudiantes.lfp
Linea [31 chars]: 202012345,Juan,Perez,Sistemas,5
Linea [34 chars]: 202013456,Maria,Lopez,Industrial,3
Linea [33 chars]: 202014567,Carlos,Gonzalez,Civil,7
Linea [33 chars]: 202015678,Ana,Martinez,Sistemas,2
Linea [37 chars]: 202016789,Luis,Hernandez,Industrial,6
Linea [34 chars]: 202017890,Sofia,Ramirez,Sistemas,8
Linea [32 chars]: 202018901,Pedro,Castillo,Civil,4
Linea [35 chars]: 202019012,Laura,Flores,Industrial,1
Linea [34 chars]: 202020123,Diego,Morales,Sistemas,9
Linea [30 chars]: 202021234,Valeria,Cruz,Civil,3
Linea [36 chars]: 202022345,Roberto,Jimenez,Sistemas,6
Linea [36 chars]: 202023456,Gabriela,Ruiz,Industrial,5
>> Se cargaron 12 estudiantes correctamente.
```

4.3 Reporte 1 - Estadisticas por Curso

Reporte 1: Estadisticas Generales por Curso

Curso	Estudiantes	Promedio	Maxima	Minima	Desv. Estandar	Mediana
Lenguajes Formales y de Programacion	5	72.70	90.00	55.00	13.52	72.00
Inteligencia Artificial 1	1	92.00	92.00	92.00	0.00	92.00
Resistencia de Materiales	1	45.00	45.00	45.00	0.00	45.00
Estadistica General	2	52.75	67.50	38.00	14.75	52.75
Calculo Diferencial	2	83.00	88.00	78.00	5.00	83.00

4.4 Reporte 2 - Rendimiento por Estudiante

Reporte 2: Rendimiento por Estudiante

Carnet	Nombre	Carrera	Semestre	Promedio	Aprobados	Reprobados	Creditos
202012345	Juan Perez	Sistemas	5	85.17	3	0	15
202013456	Maria Lopez	Industrial	3	61.25	1	1	4
202014567	Carlos Gonzalez	Civil	7	67.50	1	1	5
202015678	Ana Martinez	Sistemas	2	80.00	2	0	10
202016789	Luis Hernandez	Industrial	6	49.50	1	1	5

4.5 Reporte 3 - Top 10 Mejores Estudiantes

Reporte 3: Top 10 Mejores Estudiantes

Posicion	Carnet	Nombre	Carrera	Semestre	Promedio
1	202012345	Juan Perez	Sistemas	5	85.17
2	202015678	Ana Martinez	Sistemas	2	80.00
3	202014567	Carlos Gonzalez	Civil	7	67.50
4	202013456	Maria Lopez	Industrial	3	61.25
5	202016789	Luis Hernandez	Industrial	6	49.50

4.6 Reporte 4 - Cursos con Mayor Reprobacion

Reporte 4: Cursos con Mayor Indice de Reprobacion

Codigo	Curso	Total	Aprobados	Reprobados	% Reprobacion
348	Resistencia de Materiales	1	0	1	100.00%
412	Estadistica General	2	1	1	50.00%
771	Lenguajes Formales y de Programacion	5	4	1	20.00%
795	Inteligencia Artificial 1	1	1	0	0.00%
501	Calculo Diferencial	2	2	0	0.00%

4.7 Reporte 5 - Analisis por Carrera

Reporte 5: Analisis por Carrera

Sistemas

Total Estudiantes	Promedio General	Cursos Disponibles
5	83.10	3

Distribucion por Semestre

Sem 1	Sem 2	Sem 3	Sem 4	Sem 5	Sem 6	Sem 7	Sem 8	Sem 9	Sem 10
0	1	0	0	1	1	0	1	1	0

Industrial

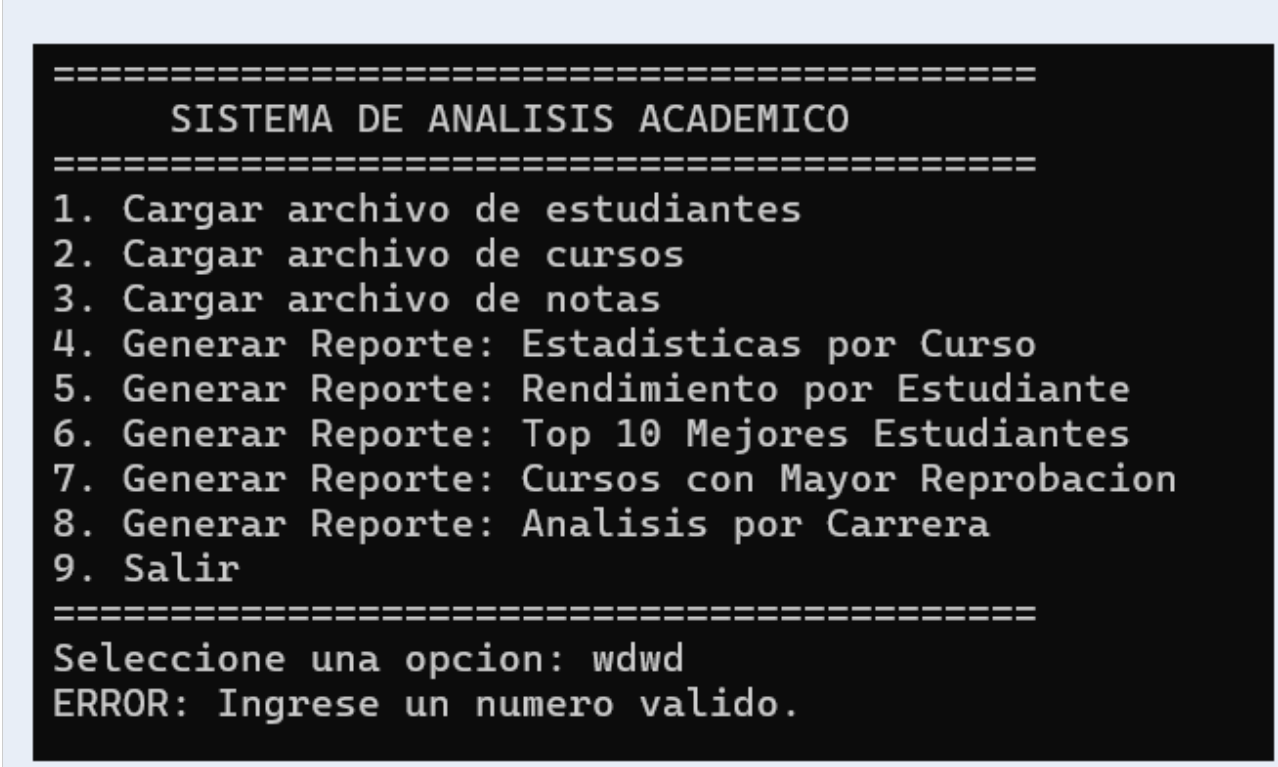
Total Estudiantes	Promedio General	Cursos Disponibles
4	55.38	1

Distribucion por Semestre

Sem 1	Sem 2	Sem 3	Sem 4	Sem 5	Sem 6	Sem 7	Sem 8	Sem 9	Sem 10

4.8 Manejo de Entradas Invalidas

Al ingresar una letra en el menu, el sistema muestra el error y regresa al menu sin cerrarse:



5. Resumen de Funcionalidades Implementadas

Funcionalidad	Estado	Observacion
Lectura de estudiantes.lfp	Completado	Lee y valida 5 campos por linea
Lectura de cursos.lfp	Completado	Lee y valida 5 campos por linea
Lectura de notas.lfp	Completado	Lee y valida 5 campos por linea
Separacion de datos por comas	Completado	Funcion split() con trim()
Calculo de promedio	Completado	Implementado en reportes 1, 2, 3 y 5
Calculo de mediana	Completado	Implementado en reporte 1
Calculo de desviacion estandar	Completado	Implementado en reporte 1
Reporte 1 - Estadisticas por curso	Completado	Genera reporte1.html
Reporte 2 - Rendimiento estudiante	Completado	Genera reporte2.html
Reporte 3 - Top 10 estudiantes	Completado	Genera reporte3.html con medallas
Reporte 4 - Cursos con reprobacion	Completado	Genera reporte4.html con colores
Reporte 5 - Analisis por carrera	Completado	Genera reporte5.html

Menu interactivo en consola	Completado	Con ciclo do-while
Manejo de excepciones en menu	Completado	try-catch para entradas invalidas
Relacion entre los tres archivos	Completado	Por carnet y codigo_curso

6. Conclusiones

1. La implementacion de un sistema de analisis de datos en C++ refuerza la comprension de las fases iniciales de un compilador, especialmente la lectura, tokenizacion y validacion de datos estructurados. El uso de split() y trim() simula directamente el proceso de analisis lexico.

2. La codificacion de archivos es un aspecto critico en el desarrollo de sistemas que leen datos externos. El problema del BOM demostro que detalles aparentemente menores en la preparacion de archivos pueden causar fallos dificiles de diagnosticar si no se cuenta con herramientas de depuracion adecuadas.

3. El uso de vectores globales y structs en C++ facilita enormemente la relacion de informacion entre multiples fuentes de datos. La decision de almacenar todos los datos en memoria antes de generar los reportes simplifica la logica de cada reporte y evita lecturas repetidas de archivos.

4. La generacion de reportes HTML directamente desde C++ sin librerias externas demuestra que con las herramientas estandar del lenguaje es posible producir salidas visuales profesionales. La inclusion de estilos CSS embebidos mejora significativamente la presentacion de los datos.

5. El manejo de excepciones con try-catch es una practica esencial para desarrollar aplicaciones robustas. Implementarlo en el menu garantiza que el programa no se cierre inesperadamente ante entradas invalidas del usuario, mejorando considerablemente la experiencia de uso.

7. Referencias

- Stroustrup, B. (2013). The C++ Programming Language (4th Edition). Addison-Wesley.
- Pressman, R. (2014). Ingenieria del Software: Un enfoque practico. McGraw-Hill.
- Walpole, R. E., Myers, R. H. (2012). Probabilidad y Estadistica para Ingenieria y Ciencias (9a edicion). Pearson.
- Documentacion oficial de C++ STL: https://cppreference.com