

Curso: Lenguajes Formales de Programacion  
Auxiliar: River Anderson Ismalej Román  
Universidad San Carlos de Guatemala  
Facultad de ingeniería.  
Ingeniería en ciencias y sistemas

# MANUAL TECNICO PRACTICA UNICA

Nombre: Bily Estuardo Vallecidos Folgar  
Carnet: 202401753  
Seccion: B-  
Fecha: 27/02/2026

# **1. INTRODUCCION**

El presente manual describe la implementacion tecnica de un sistema desarrollado en C++ que lee, procesa y analiza archivos de datos academicos, generando reportes estadisticos en formato HTML. El sistema simula las fases iniciales de un compilador: lectura, separacion, validacion y analisis de datos estructurados.

El programa permite cargar tres archivos con informacion de estudiantes, cursos y notas, relacionarlos entre si, calcular estadisticas descriptivas avanzadas y presentar los resultados en reportes HTML con formato profesional.

## **2. Requerimientos Tecnicos**

### **2.1 Software**

- Lenguaje de programacion: C++17
- Compilador: MSVC (Visual Studio) o g++/clang++
- IDE: Visual Studio 2019/2022 o equivalente
- Sistema Operativo: Windows 10/11
- Navegador web para visualizar los reportes HTML generados

### **2.2 Librerias Utilizadas (STL)**

Libreria	Proposito
iostream	Entrada y salida estandar (cout, cin)
fstream	Lectura y escritura de archivos
sstream	Procesamiento de cadenas como streams
vector	Almacenamiento dinamico de datos
string	Manejo de cadenas de texto
algorithm	Algoritmos de ordenamiento (sort)
cmath	Funciones matematicas (sqrt, pow)
iomanip	Formato de salida (setprecision, fixed)

## **3. Estructuras de Datos**

El programa utiliza tres estructuras (structs) para representar los datos de los archivos de entrada:

### **3.1 Struct Estudiante**

Campo	Tipo	Descripcion
carnet	int	Identificador unico del estudiante

nombre	string	Nombre del estudiante
apellido	string	Apellido del estudiante
carrera	string	Carrera que cursa
semestre	int	Semestre actual (1-10)

### **3.2 Struct Curso**

Campo	Tipo	Descripcion
codigo	int	Codigo unico del curso
nombre	string	Nombre del curso
creditos	int	Cantidad de creditos (1-8)
semestre	int	Semestre en que se imparte
carrera	string	Carrera a la que pertenece

### **3.3 Struct Nota**

Campo	Tipo	Descripcion
carnet	int	Identificador del estudiante
codigo_curso	int	Codigo del curso cursado
nota	double	Calificacion obtenida (0-100)
ciclo	string	Ciclo academico (1S o 2S)
anio	int	Anio en que se curso

## **4. Vectores Globales**

Los datos leídos de los archivos se almacenan en tres vectores globales accesibles por todas las funciones del programa:

- vector<Estudiante> estudiantes — almacena todos los estudiantes cargados
- vector<Curso> cursos — almacena todos los cursos cargados
- vector<Nota> notas — almacena todas las notas cargadas

## **5. Funciones Principales**

Funcion	Descripcion
---------	-------------

trim(string)	Elimina espacios, \r, \n y BOM al inicio y final de una cadena
split(string, char)	Separa una cadena por un delimitador y retorna vector de tokens
cargarEstudiantes()	Lee y procesa el archivo estudiantes.lfp
cargarCursos()	Lee y procesa el archivo cursos.lfp
cargarNotas()	Lee y procesa el archivo notas.lfp
reporte1()	Genera HTML con estadísticas generales por curso
reporte2()	Genera HTML con rendimiento individual por estudiante
reporte3()	Genera HTML con el top 10 de mejores estudiantes
reporte4()	Genera HTML con cursos ordenados por índice de reprobación
reporte5()	Genera HTML con análisis estadístico por carrera
mostrarMenu()	Imprime el menú interactivo en consola
main()	Función principal con loop de menú y manejo de excepciones

## **6. Pseudocódigo**

### **6.1 Lectura de Archivos**

```

INICIO cargarEstudiantes / cargarCursos / cargarNotas
  Solicitar nombre de archivo al usuario
  Intentar abrir el archivo
  SI archivo no se puede abrir:
    Mostrar mensaje de error
    Terminar función
  Limpiar vector de datos previos
  PARA cada línea del archivo:
    SI línea inicia con BOM (0xEF): eliminar primeros 3 bytes
    SI línea termina con \r: eliminar último carácter
    SI línea vacía: continuar con siguiente
    Separar línea por comas -> obtener campos
    SI cantidad de campos incorrecta: omitir línea
    Crear objeto con los campos
    Agregar objeto al vector
    Incrementar contador
  Cerrar archivo
  Mostrar cantidad de registros cargados
FIN

```

## **6.2 Calculos Estadisticos (Reporte 1)**

PARA cada curso en vector cursos:  
  notasCurso = lista de notas donde codigo\_curso == curso.codigo  
  SI notasCurso vacia: continuar  
  promedio = suma(notasCurso) / cantidad  
  maxima = mayor valor en notasCurso  
  minima = menor valor en notasCurso  
  desviacion = raiz( suma((nota - promedio)^2) / cantidad )  
  ordenar notasCurso  
  SI cantidad es par:  
    mediana = (notasCurso[n/2-1] + notasCurso[n/2]) / 2  
  SINO:  
    mediana = notasCurso[n/2]  
  Escribir fila en HTML con todos los valores

## **6.3 Menu Principal con Manejo de Excepciones**

REPETIR:  
  Mostrar menu  
  INTENTAR:  
    Leer entrada del usuario como string  
    PARA cada caracter en entrada:  
      SI no es digito: lanzar excepcion invalid\_argument  
    Convertir entrada a entero  
    Ejecutar funcion segun opcion con switch  
  CAPTURAR invalid\_argument:  
    Mostrar 'ERROR: Ingrese un numero valido'  
    Limpiar buffer de entrada  
    Resetear opcion a 0  
  MIENTRAS opcion != 9

## **7. Formato de Archivos de Entrada**

Los archivos de entrada deben tener extension .lfp y estar codificados en UTF-8 sin BOM. Cada linea representa un registro con campos separados por comas.

### **7.1 estudiantes.lfp**

Formato: carnet,nombre,apellido,carrera,semestre  
Ejemplo: 202012345,Juan,Perez,Sistemas,5

### **7.2 cursos.lfp**

Formato: codigo,nombre,creditos,semestre,carrera

Ejemplo: 771,Lenguajes Formales y de Programacion,5,3,Sistemas

### **7.3 notas.lfp**

Formato: carnet,codigo\_curso,nota,ciclo,anio

Ejemplo: 202012345,771,85.5,1S,2024

## **8. Archivos de Salida HTML**

Archivo	Contenido
reporte1.html	Estadísticas generales por curso (promedio, max, min, desv. estandar, mediana)
reporte2.html	Rendimiento por estudiante (aprobados, reprobados, creditos acumulados)
reporte3.html	Top 10 estudiantes con mejor promedio general
reporte4.html	Cursos ordenados por porcentaje de reprobacion
reporte5.html	Analisis por carrera con distribucion por semestre

## **9. Observaciones del Desarrollo**

### **9.1 Problema con BOM en archivos .lfp**

Durante el desarrollo se detecto que los archivos creados con el Bloc de notas de Windows incluyen una marca de byte al inicio (BOM: 0xEF 0xBB 0xBF) en codificacion UTF-8. Esto causaba que el primer campo de cada linea fuera interpretado incorrectamente, resultando en errores de conversion y registros omitidos.

Solucion aplicada: los archivos .lfp deben crearse desde Visual Studio usando 'Guardar con codificacion' seleccionando 'Unicode (UTF-8 sin firma)'. Adicionalmente el codigo incluye deteccion y eliminacion del BOM al inicio de cada linea.

### **9.2 Manejo de Excepciones en el Menu**

Se implemento un bloque try/catch en el ciclo principal del menu para capturar entradas invalidas (letras, simbolos). Cuando el usuario ingresa un valor no numerico, el programa muestra un mensaje de error, limpia el buffer de entrada y regresa al menu sin cerrarse.

### **9.3 Separacion de Datos**

La funcion split() utiliza istringstream y getline() con delimitador de coma para separar los campos de cada linea. La funcion trim() complementa este proceso limpiando caracteres no visibles como espacios, retornos de carro (\r) y saltos de linea (\n) que pueden aparecer segun el sistema operativo.

### **9.4 Relacion entre Archivos**

La informacion de los tres archivos se relaciona mediante los campos carnet (entre estudiantes y notas) y codigo\_curso (entre cursos y notas). Esta relacion se realiza en memoria mediante busquedas lineales en los vectores globales durante la generacion de cada reporte.