Estándar de Codificación

Propósito	• Guía para codificar programas en C++
Encabezado del Programa	• Estará mi nombre acompañado por mi matricula escolar. Seguido del nombre del programa.

Formato del encabezado	Estará seccionada por región con el uso de #pragma region Encabezado. Mi nombre y el nombre del programa estará comentado. El encabezado está terminado con #pragma endregion.
	
	#pragma region Encabezado
	//Francisco Rochín Gómez
	//Gestor de Código
	#pragma endregion

Instrucciones de uso:	• No hay instrucciones. • Solamente descripción de lo que se espera que haga el programa

Ejemplo de uso	{
	El programa deberá contar las líneas de códigos, sin contar comentarios.
	}

Estándar de Codificación

Identificadores	• Los identificadores serán en ingles siempre.
Ejemplo de identificadores	num printLines() private: _name
Comentarios	• Deberá comentarse funciones que representen funcionalidades finales del programa. • Solamente se comentan variables que sea primordial su entendimiento. • Todo comentario estará situado encima de su respectiva variable, clase y/o función. • En el caso que un archivo con codificación se extendía mucho y/o tenga muchas partes, se utilizará las regiones (#pragma region) para separar estas secciones.

Comentarios correctos	// Función que cuenta líneas de código y imprime e imprime el número de líneas
	
	int countLines(string código) {
	//variable que se usara para el conteo
	int counter
	}

Comentarios incorrectos	// Función Principal
	int countLines() {
	
	}
	
	// Este esta variable hace algo
	int pointer

Espacios en blanco	• Entre cada bloque de encabezado, lista de variables (ya sea declaradas o definidas), módulos y programa principal deberá separarse con una línea en blanco para mejorar legibilidad.

Estándar de Codificación

Sangría	• Úsese espaciado por cada nivel requerido • Cada cerrado de llaves deberá respetar la sangría. • Cada declaración y/o definición de variables y funciones de una clase, debe respetar las sangrías.

Ejemplo de uso de sangrado	int main() {
	try {
	Solution sol;
	string s = "abcabcbb";
	cout << sol.lengthOfLongestSubstring(s) << endl;
	//Expected output: 3
	string s2 = "bbbbb";
	cout << sol.lengthOfLongestSubstring(s2) << endl;
	//Expected output: 1
	string s3 = "pwwkew";
	cout << sol.lengthOfLongestSubstring(s3) << endl;
	//Expected output: 3
	return 0;
	}
	catch (std::exception& e) {
	std::cerr << "Excepción: " << e.what() << std::endl;
	}
	}

Encabezado	• En una misma línea deberá estar el nombre del programa.
Seclaración de variables	• Se usará camelCase para todo identificador. Cuando se trate de variables con mismo nombre, se le añadirá el número respectivo a su serie. Toda variable de una clase privada iniciará con el carácter _. • Se usará inglés.
Sentencias	• No deberá existir sentencias que rebasen el ancho de la pantalla (80 caracteres), en su caso se pasarán al siguiente renglón si es posible.
Definición de constantes	• Se puede definir muchas constantes en una misma línea, mientras se mantenga legible y ordenado. • Se puede definir muchas constantes en una sola línea, si se trata de constantes con valores simples.
Asignación	• Deberá emplearse una línea por cada asignación
Encabezados de módulos	• Se usará #pragma region (nombre de modulo o sección)y #pragma endregion para cada uno de los módulos o secciones que ameriten separación o clasificación.

Estructuras de control de flujo	• Las condicionales y sus parámetros deben ser seguidas por apertura de llave y una sangría.
	• Los ciclos y sus parámetros deben ser seguidas por apertura de llave y una sangría.
	• Los métodos de una clase siempre deben utilizar *return.*
	• Al utilizar *try* y *catch* se debe respetar la sangría, después de haber cerrado la llave de cierre *}* pero se continuará con el código en la siguiente línea para legibilidad. Ejemplo:
	try {
	cout<<endl;
	}
	catch (){
	}