Actividad de Evaluación Continua 1

Práctica 1 de programación con C++

Alumno: Alexander Sebastian Kalis
Profesor: Javier Llorente Ayuso

14 de diciembre de 2025

Índice

1. Introducción y Descripción del Algoritmo

El objetivo de esta práctica es desarrollar un programa en C++ que permita detectar patrones en series de caracteres introducidas por el usuario y generar series aleatorias basadas en patrones predefinidos. El programa ha sido diseñado siguiendo los principios de la programación estructurada y cumpliendo estrictamente con los requisitos del enunciado.

1.1. Estructura del Programa

El código se ha organizado en tres bloques funcionales principales, gestionados a través de un menú interactivo:

- **Menú Principal:** Un bucle `do-while` que presenta las opciones al usuario y valida la entrada hasta que se selecciona la opción de salir.
- **Detección de Patrones (Opción 1):** Permite al usuario introducir 4 caracteres. El algoritmo analiza si estos caracteres pertenecen exclusivamente a uno de los conjuntos definidos (vocales, números, caracteres especiales) o si forman una serie general.
- **Generación de Series (Opción 2):** Utiliza la generación de números pseudoaleatorios para crear una serie de 10 elementos basada en la selección del usuario.

1.2. Decisiones de Diseño y Funciones Empleadas

Para cumplir con las especificaciones y asegurar la robustez del código, se han tomado las siguientes decisiones de implementación:

1. **Librerías Estándar:** Se han utilizado únicamente librerías básicas:
   - `iostream`: Para la entrada y salida de datos.
   - `cstdlib` y `ctime`: Para la gestión de números aleatorios (`rand`, `srand`) y la semilla de tiempo.
   - `cctype`: Para la función `tolower`, fundamental para estandarizar la entrada del usuario.
2. **Normalización de Datos:** Dado que el enunciado especifica que las vocales y consonantes deben tratarse en minúsculas, se aplica la función `tolower()` a cada carácter introducido. Además, se ha implementado un filtro para ignorar las comas (`,`) en la entrada de datos, permitiendo al usuario introducir series como "1, 2, 3, 4"sin errores.
3. **Lógica de Detección (Banderas):** Para detectar el tipo de serie en la Opción 1, se utiliza una lógica de "banderas"booleanas (`sonVocales`, `sonNumeros`, `sonEspeciales`). Se asume inicialmente que la serie cumple todas las condiciones y se descartan a medida que se analizan los caracteres. Si al final del análisis ninguna bandera específica se mantiene verdadera, o si hay una mezcla, se determina que es una **Serie General**.
4. **Generación Aleatoria:** Se emplea `rand() % N` para generar índices aleatorios. Para los caracteres especiales, se ha definido un array constante estricto: {`'#', '$', '%', '&'`}, evitando así generar caracteres no permitidos por el enunciado.

2. Código Fuente en C++

A continuación se presenta el código fuente completo desarrollado para la actividad.

```cpp
/*
 * Practica 1: Fundamentos de programacion con C++
 * Asignatura: Fundamentos de programacion (Cod. 1375)
 * Autor: Alexander Sebastian Kalis
 * Fecha: 14 dic 2025
 *
 * Descripcion:
 * Programa para detectar patrones en series de 4 caracteres introducidos
 * por el usuario y generar series aleatorias de 10 elementos basadas
 * en patrones predefinidos (vocales, numericos, especiales o general).
 */

#include <iostream>
#include <cstdlib> // Necesario para rand() y srand()
#include <ctime>   // Necesario para time()
#include <cctype>  // Necesario para tolower()

using namespace std;

// --- Prototipos de funciones ---
void mostrarMenuPrincipal();
void opcionDetectarPatron();
void opcionGenerarSerie();
bool esVocal(char c);
bool esNumero(char c);
bool esEspecialPermitido(char c);

int main() {
    // Inicializamos la semilla aleatoria una unica vez al principio
    srand(time(NULL));

    int opcion = 0;

    do {
        mostrarMenuPrincipal();
        cin >> opcion;

        switch (opcion) {
            case 1:
                opcionDetectarPatron();
                break;
            case 2:
                opcionGenerarSerie();
                break;
            case 3:
                cout << "Se cerrara la aplicacion ..." << endl;
                break;
            default:
                cout << "Opcion incorrecta. Por favor, elija 1, 2 o 3." << endl;
        }
        cout << endl; // Salto de linea estetico

    } while (opcion != 3);

    return 0;
}

// --- Implementacion de funciones ---

void mostrarMenuPrincipal() {
    cout << "-----------------------------------------------------" << endl;
    cout << "1) Detectar un patron en una serie." << endl;
    cout << "2) Generar una serie alfanumerica de entre las posibles." << endl;
    cout << "3) Salir del programa" << endl;
    cout << "Opcion elegida: ";
}

```

```
68 /**
69  * Opcion 1: El usuario introduce 4 caracteres.
70  * El programa determina si son vocales, numeros, especiales (#$%&) o una mezcla (General
    ).
71  */
72 void opcionDetectarPatron() {
73     char serie[4];
74     char entrada;
75     int contador = 0;
76 
77     cout << "Introduce cuatro caracteres de la serie (puedes separarlos por espacios o
    comas):" << endl;
78 
79     // Leemos 4 caracteres validos.
80     // El bucle ignora las comas ',' para que el caso de prueba 1.2 funcione
    correctamente.
81     while (contador < 4) {
82         cin >> entrada;
83 
84         // Convertimos a minuscula inmediatamente para estandarizar
85         entrada = tolower(entrada);
86 
87         // Si es una coma, la ignoramos y pasamos al siguiente ciclo
88         if (entrada == ',') {
89             continue;
90         }
91 
92         serie[contador] = entrada;
93         contador++;
94     }
95 
96     // Banderas para comprobar los tipos. Asumimos que son verdaderas hasta que se
    demuestre lo contrario.
97     bool sonVocales = true;
98     bool sonNumeros = true;
99     bool sonEspeciales = true;
100 
101     for (int i = 0; i < 4; i++) {
102         if (!esVocal(serie[i])) {
103             sonVocales = false;
104         }
105         if (!esNumero(serie[i])) {
106             sonNumeros = false;
107         }
108         if (!esEspecialPermitido(serie[i])) {
109             sonEspeciales = false;
110         }
111     }
112 
113     // Resultados segun las banderas
114     if (sonVocales) {
115         cout << "La serie esta formada por vocales" << endl;
116     } else if (sonNumeros) {
117         cout << "La serie esta formada por caracteres numericos" << endl;
118     } else if (sonEspeciales) {
119         cout << "La serie esta formada por caracteres especiales" << endl;
120     } else {
121         // Si no cumple ninguno de los patrones especificos, es General por descarte.
122         cout << "La serie es general" << endl;
123     }
124 }
125 
126 /**
127  * Opcion 2: Genera 10 caracteres aleatorios segun el tipo elegido.
128  */
129 void opcionGenerarSerie() {
130     int tipo;
131     cout << "Introducir el tipo de serie que queremos generar" << endl;
132     cout << "1. Serie formada por vocales exclusivamente" << endl;
133     cout << "2. Serie formada por caracteres numericos exclusivamente" << endl;
```

```
134     cout << "3. Serie formada por caracteres especiales exclusivamente" << endl;
135     cout << "4. Serie general" << endl; // Corregida la errata "geneal" del enunciado
136     cin >> tipo;
137 
138     cout << "Serie: ";
139 
140     // Definimos los conjuntos de caracteres permitidos
141     char vocales[] = {'a', 'e', 'i', 'o', 'u'};
142     char especiales[] = {'#', '$', '%', '&'};
143     // Nota: Para numeros usamos aritmetica de caracteres ('0' + n)
144     // Para general usamos una mezcla
145 
146     for (int i = 0; i < 10; i++) {
147         char generado;
148 
149         switch (tipo) {
150             case 1: // Vocales
151                 generado = vocales[rand() % 5];
152                 break;
153             case 2: // Numeros ('0' a '9')
154                 generado = '0' + (rand() % 10);
155                 break;
156             case 3: // Especiales (#, $, %, &)
157                 generado = especiales[rand() % 4];
158                 break;
159             case 4: // General
160                 // Estrategia: Elegimos aleatoriamente que tipo de caracter generar
161                 // 0: Vocal, 1: Numero, 2: Especial, 3: Consonante
162                 int subtipo = rand() % 4;
163                 if (subtipo == 0) {
164                     generado = vocales[rand() % 5];
165                 } else if (subtipo == 1) {
166                     generado = '0' + (rand() % 10);
167                 } else if (subtipo == 2) {
168                     generado = especiales[rand() % 4];
169                 } else {
170                     // Generar una letra minuscula cualquiera (a-z)
171                     // Validamos que no sea vocal para que sea estrictamente consonante (
    opcional, pero limpio)
172                     do {
173                         generado = 'a' + (rand() % 26);
174                     } while (esVocal(generado));
175                 }
176                 break;
177         }
178 
179         cout << generado;
180         // Anadimos coma y espacio si no es el ultimo elemento
181         if (i < 9) {
182             cout << ", ";
183         }
184     }
185     cout << endl;
186 }
187 // --- Funciones auxiliares ---
188 bool esVocal(char c) {
189     // Asumimos que c ya viene en minuscula
190     return (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u');
191 }
192 
193 bool esNumero(char c) {
194     // Comprobamos rango ASCII de digitos
195     return (c >= '0' && c <= '9');
196 }
197 
198 bool esEspecialPermitido(char c) {
199     // Los caracteres especiales que indica el enunciado
200     return (c == '#' || c == '$' || c == '%' || c == '&');
201 }
```

Listing 1: Código fuente main.cpp

3. Pruebas Realizadas (Casos de Prueba)

Se ha verificado el funcionamiento del programa sometiéndolo a los casos de prueba especificados en el enunciado de la práctica. A continuación se muestran las capturas de pantalla de la ejecución.

Caso 1.1: Detección de Vocales

El usuario introduce una secuencia de vocales (a, e, u, a). El programa debe detectar el patrón correctamente.

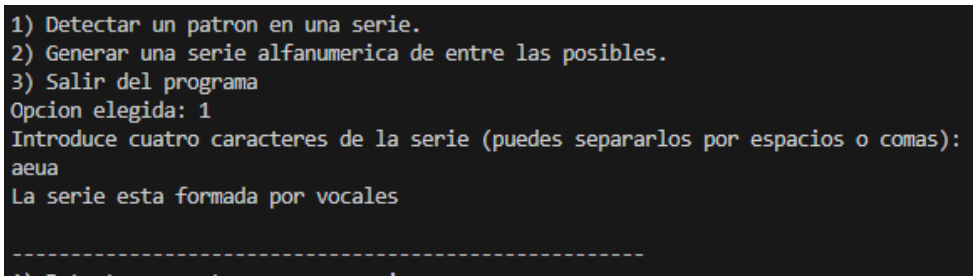

Figura 1: Ejecución del Caso 1.1

Caso 1.2: Detección de Números

El usuario introduce números separados por comas (1, 9, 8, 0). El programa filtra las comas y detecta el patrón numérico.

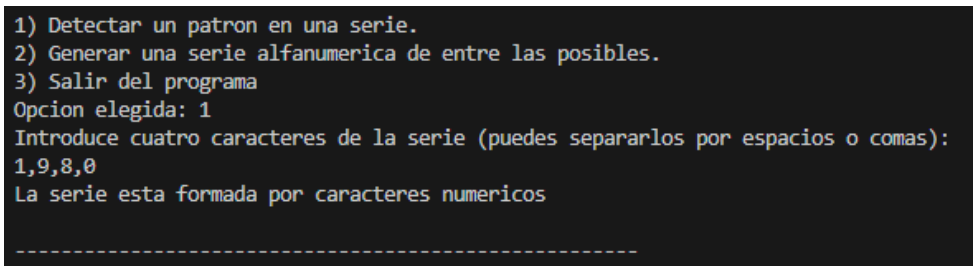

Figura 2: Ejecución del Caso 1.2

Caso 1.3: Detección de Caracteres Especiales

El usuario introduce los caracteres especiales permitidos (%, &, $, #).

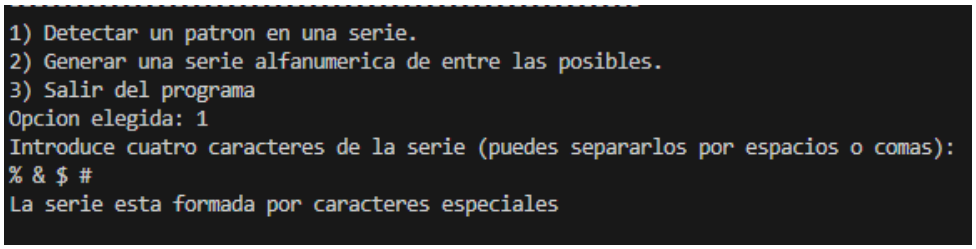

Figura 3: Ejecución del Caso 1.3

Caso 1.4: Detección de Serie General

El usuario introduce una mezcla de caracteres (1, a, b, 2). El programa determina que es una serie general.

```
--------------------------------------------------
1) Detectar un patron en una serie.
2) Generar una serie alfanumerica de entre las posibles.
3) Salir del programa
Opcion elegida: 1
Introduce cuatro caracteres de la serie (puedes separarlos por espacios o comas):
1ab2
La serie es general
```

Figura 4: Ejecución del Caso 1.4

Casos de Generación (2.1 a 2.4)

Pruebas de la opción 2 del menú para generar series aleatorias de los distintos tipos.

```
2) Generar una serie alfanumerica de entre las posibles.
3) Salir del programa
Opcion elegida: 2
Introducir el tipo de serie que queremos generar
1. Serie formada por vocales exclusivamente
2. Serie formada por caracteres numericos exclusivamente
3. Serie formada por caracteres especiales exclusivamente
4. Serie general
1
Serie: u, i, a, i, e, o, o, u, e, u

-----------------------------------------------------
1) Detectar un patron en una serie.
2) Generar una serie alfanumerica de entre las posibles.
3) Salir del programa
Opcion elegida: 2
Introducir el tipo de serie que queremos generar
1. Serie formada por vocales exclusivamente
2. Serie formada por caracteres numericos exclusivamente
3. Serie formada por caracteres especiales exclusivamente
4. Serie general
2
Serie: 6, 1, 1, 6, 4, 6, 7, 1, 4, 8

-----------------------------------------------------
1) Detectar un patron en una serie.
2) Generar una serie alfanumerica de entre las posibles.
3) Salir del programa
Opcion elegida: 2
Introducir el tipo de serie que queremos generar
1. Serie formada por vocales exclusivamente
2. Serie formada por caracteres numericos exclusivamente
3. Serie formada por caracteres especiales exclusivamente
4. Serie general
3
Serie: #, &, &, %, #, #, $, %, $, $

-----------------------------------------------------
1) Detectar un patron en una serie.
2) Generar una serie alfanumerica de entre las posibles.
3) Salir del programa
Opcion elegida: 2
Introducir el tipo de serie que queremos generar
1. Serie formada por vocales exclusivamente
2. Serie formada por caracteres numericos exclusivamente
3. Serie formada por caracteres especiales exclusivamente
4. Serie general
4
Serie: c, &, 0, %, %, $, e, e, o, d
```

Figura 5: Ejecución de los casos de generación de series