Universidad Central de Venezuela

Facultad de Ciencias

Escuela de Computación

Algoritmos y Programación

Semestre I-2023

PROYECTO – PARTE 2

Código Morse Internacional

Eduardo Bogado 26.818.715

Profa. Yusneyi Carballo Barrera

11 de agosto de 2023

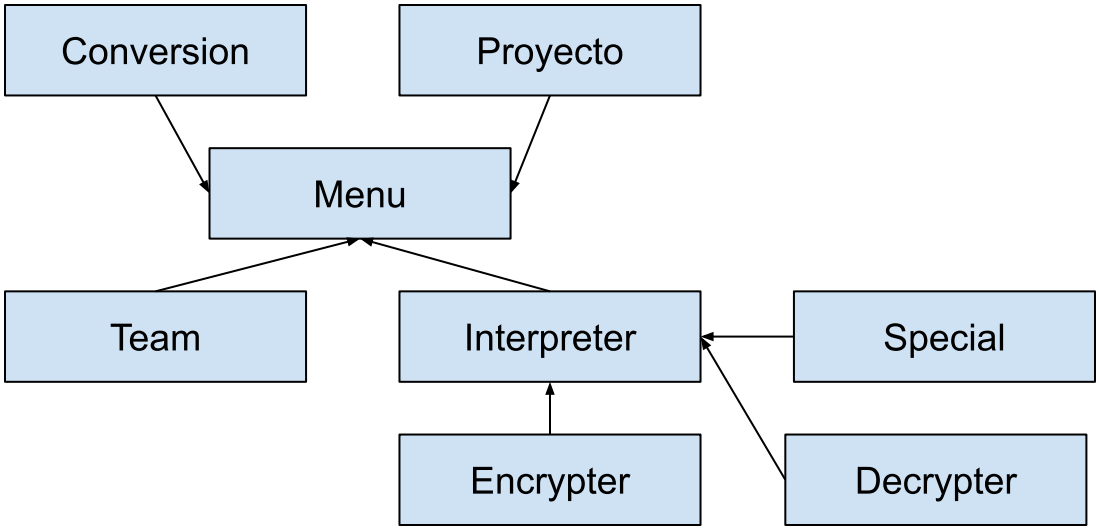
ANÁLISIS DEL PROBLEMA

Se continúa el proyecto comenzado con la función de realizar transcripciones entre texto y código morse, esta vez, añadiendo nuevos elementos incluyendo lectura de archivos, estructuras de datos y programación orientada a objetos.

En primera instancia se mantuvo el enfoque de realizar lectura y escritura de texto y morse en función de un diccionario predefinido, sin embargo, se añadió la lectura de un archivo que contiene todas las equivalencias entre caracteres de texto y cadenas de código morse. Esto se hizo con la función de facilitar la manipulación de las definiciones de caracteres predefinidas.

Otro cambio fundamental en el enfoque empleado en esta etapa del desarrollo del software es la aplicación de programación orientada a objetos, esto se hizo para aprovechar las similitudes existentes entre diversos menús del programa y reducir al mínimo el pase de parámetros el cual resultaba ser complicado para trabajar una vez el volumen de datos aumentaba.

A continuación se muestra el diagrama de clases empleado en la resolución del problema:



Puede observarse en el diagrama que todo el algoritmo está basado esencialmente en la superclase Menú a partir de la cual otras clases heredan características usadas repetidamente como métodos de entrada y mensajes al usuario.

Adicional al enfoque de transcribir textos ingresados por el usuario, también se añadió la posibilidad de suministrar textos mediante archivos para su correspondiente codificación / decodificación, así como también se incluyó un algoritmo de ordenamiento y listado para manipular temperaturas registradas en ciudades.

Esta última característica se incluyó en una clase llamada “Special” la cual tenía pocas características en común al resto de las clases. Sin embargo, esta también pertenece a la clase interpreter debido a la necesidad de realizar transcripciones a código morse por lo que los métodos de esta última resultan útiles.

El enfoque orientado a objetos facilitó en gran medida el diseño y desarrollo del programa, permitiendo el encapsulamiento de distintos módulos lo que permitía acelerar la detección de errores en el código. Sin embargo, la interdependencia entre las clases relacionadas por herencia resultó problemático en varias oportunidades debido a problemas de compatibilidad para algunos métodos compartidos.

El desarrollo de software se llevó a cabo por completo en el lenguaje de programación c++ utilizando solo librerías estándar. A fin de mantener la compatibilidad entre distintos sistemas operativos se optó por no usar librerías específicas a ningún sistema en particular esto con el costo de no poder utilizar caracteres diacríticos, acentos, tildes y caracteres ajenos al estándar UTF-8.

Se muestra el desarrollo en pseudocódigo de las nuevas características.

proyecto\ensayo.pse

```
//--------------------------------------------------------------------//
//                                                                    //
//                    Transcriptor de Código Morse                    //
//                                                                    //
//--------------------------------------------------------------------//
// Codigo finalizado el día 11/9/23
// Programa desarrollado por Eduardo Bogado
// Universidad Central de Venezuela
// Github user eduardob999
// Prof. Yusneiyi Carballo.

Clase Conversion Hereda de Menu

    // Métodos

    publico:

        // Constructor

        Acción Conversion()

            // Opción 1 - Identificación del equipo.

            string dummy;

            // Inicialización de mensjaes

            prompt = "Opcion 5 - Conversion de mensajes desde archivos.

                      Menu:
                        1.  Leer mensaje en texto
                        2.  Leer mensaje en Codigo Morse
                        3.  Ordenar y listar archivo tempEntrada.txt
                        4.  Procesamiento y calculos del archivo tempEntrada.txt
                        5.  Volver al menu principal

                      Por favor, indique su opcion:";

            condition = "Debe ingresar un numero entero entre 1 y 5.
                         Ingrese su opcion nuevamente:";

            dictionary = "  1   2   3   4   5   ";

            // Procesamiento

            Repetir

                // En este ciclo se controla la ejecicón y salida del programa
                // la variable inp se emplea como bandera

                Input( 1, 1, Falso );

```

```
                // Se inicializa el parametro input

                Selección

                    aEntero( input ) == 1 :

                        Encrypter e2.Encrypter( "entradaT.txt", "salidaT.txt" );

                    aEntero( input ) == 2 :

                        Decrypter d2.Decripter( "entradaM.txt",  "salidaM.txt" );

                    aEntero( input ) == 3 :

                        Special s1.Special1( "tempEntradaT.txt", "tempSalidaT.txt" );

                    aEntero( input ) == 4 :

                        Special s2.Special2( "tempEntradaT.txt" );

                    aEntero( input ) == 4 :

                        // Salida por defecto

                FSelección;

            Hasta ( input == "5" );

        FAcción;

FClase;

Clase Special Hereda de Interpreter

    // Atributos

    privado:

        int n;
        int m;
        Arreglo matrix de string[n,m];

    // Métodos

    publico:

        // Constructores

        Acción Special1( string filename )

            // Declaración e inicialización de variables

            int i, sum;
            i = 0;
            sum = 0;

```

```
109             // Las variables no declaradas son atributos de las superclases.
110
111             prompt = "Cual ciudad desea consultar?";
112
113             condition = "La ciudad ingresada debe coincidir de manera exacta con alguna de
    las incluidas en el archivo de entrada.";
114
115             // Procesamiento
116
117             Escribir("Opcion 5.4 - Procesamiento y calculos
118
119                     Ciudad o ciudades con la menor y la mayor temperatura en grados
    Celcius:");
120
121             ReadMatrix( filename );
122             Sort( Falso );
123
124             Repetir
125
126                 Escribir("Menor:" + matrix[0][i] + ", " + matrix[1][i] );
127                 i = i + 1;
128
129             Hasta ( matrix[1][i] == matrix[1][i-1] Y i < m );
130
131             i = m;
132
133             Repetir
134
135                 Escribir("Mayor:" + matrix[0][i-1] + ", " + matrix[1][i-1] );
136                 i = i - 1;
137
138             Hasta ( matrix[1][i-1] == matrix[1][i] Y i > 0 );
139
140             Para i = 1 hasta i = m hacer
141
142                 sum = sum + atoi( matrix[1][i].c_str() );
143
144             FPara
145
146             Escribir("- Temperatura promedio: " +
147                     sum / m + " grados Celcius.");
148
149             dictionary = "  ";
150
151             Para ( i = 0; i < m) hacer
152
153                 dictionary = dictionary + matrix[0][i] + "  ";
154
155             FPara;
156
157             Input( 1, 10, Verdadero );
158
159             i = 0;
160
161             Repetir
162
163                 i = i + 1;
```

```
164
165             Hasta ( input == matrix[0][i] );
166
167             Escribir( "La temperatura en " + input + " es: " + matrix[1][i] + " grados." );
168
169             input = "La temperatura en " + matrix[0][i] + " es: " + matrix[1][i] + "
    grados.";
170
171             dictionary = "  ";
172
173             setDictionary( 0 );
174
175             ConvertText();
176
177             Escribir( procesed );
178
179         FAcción
180
181         Acción Special2( string filename, string out )
182
183             // Declaración e inicialización de variables
184
185             string dummy, combined;
186             combined = "";
187
188
189             // Procesamiento
190
191             ReadMatrix( filename );
192             Sort( Verdadero );
193
194             combined = "Listado de ciudades y temperaturas en grados Celsius, ordenados por
    ciudad:"
195             + sorted + "";
196
197             Sort( Falso );
198
199             combined = combined + "Listado de ciudades y temperaturas en grados Celsius,
    ordenados por temperatura:"
200             + sorted;
201
202             Escribir( combined );
203             SaveFile( out, combined );
204
205             Escribir("El resultado del ordenamiento fue guardado satisfactoriamente en el
    archivo " << out <<
206             "Presione ENTER para volver al menu anterior.");
207
208             Leer( dummy );
209
210         FAcción
211
212         // Operaciones
213
214         Acción ReadMatrix( string filename ) {
215
216             // Inicializa la matriz con los datos de ciudades y temperaturas
```

```
217
218         }
219
220         Acción Sort( booleano city )
221
222             // Declaración e inicialización de veriables
223
224             int i, j;
225             string temp;
226             sorted = "";
227
228
229             Si ( city ) Entonces
230
231                 Para i = 0 hasta i = m hacer
232
233                     Para j = 0 hasta j = m hacer
234
235                         Si ( matrix[0][j] > matrix[0][j+1] )
236
237                             temp = matrix[0][j]; matrix[0][j] = matrix[0][j+1]; matrix[0]
    [j+1] = temp;
238                             temp = matrix[1][j]; matrix[1][j] = matrix[1][j+1]; matrix[1]
    [j+1] = temp;
239
240                         FSi;
241
242                     FPara
243
244                 FPara
245
246             Sino
247
248                 Para i = 0 hasta i = m hacer
249
250                     Para j = 0 hasta j = m hacer
251
252                         Si ( atoi( matrix[1][j].c_str() ) > atoi( matrix[1][j+1].c_str() ) )
253
254                             temp = matrix[0][j]; matrix[0][j] = matrix[0][j+1]; matrix[0]
    [j+1] = temp;
255                             temp = matrix[1][j]; matrix[1][j] = matrix[1][j+1]; matrix[1]
    [j+1] = temp;
256
257                         FSi;
258
259                     FPara;
260
261                 FPara;
262
263             FSi;
264
265             Para j = 0 hasta j = m hacer
266
267                     sorted = sorted + matrix[0][j] + ", " + matrix[1][j] + "";
268
269             FPara
```

```
270
271         FAcción
272
273         Acción ConvertText()
274
275             // Esta acción procesa las entradas de texto y las convierte en morse.
276             // Declaración de variables.
277
278             int i, ind, q;
279
280             // Inicialización de variables.
281
282             procesed = "";
283
284             // Procesamimiento de la cadena de entrada.
285
286             Para i = 0 hasta i < input.longitud() hacer
287
288                 // Este ciclo verifica cada letra de la entrada
289                 // se reutiliza el método search en esta acción
290
291                 Search( i, ind, 1 );
292
293                 Si ( interpreter[1][ind-1] == "/" || i - 1 > input.longitud() ) Entonces
294
295                     // Este condicional cuenta la cantidad de palabras y las almacena en
    "palen".
296
297                     palen = palen + 1;
298
299                 FSi;
300
301                 Si ( interpreter[0][ind-1] == "#") Entonces
302
303                     procesed = procesed + "";
304                     palen = palen + 1;
305                     cantmsg = cantmsg + 1;
306
307                 Sino
308
309                     procesed = procesed + interpreter[1][ind-1] + " ";
310
311                 FSi;
312
313             FPara;
314
315             // Procesamiento de datos intermedios.
316
317             cantmsg = cantmsg + 1;
318             cantpal = cantpal + "- Mensaje " + cantmsg + ": " + palen + " palabra(s)";
319             totpal = totpal + palen;
320
321         FAcción;
322
323 FClase;
```