l'index de l'élément cible s'il a été trouvé.

- **AFFICHER "L'élément", element_cible, "n'a pas été trouvé dans la liste."** : Affiche un message si l'élément cible n'a pas été trouvé.
- **FIN** : Marque la fin de la section principale de l'algorithme.

3. **Exercice : Compression de données avec l'algorithme Huffman**
   - **ALGORITHME Huffman_Codage(chaine)** : Définit le début de l'algorithme avec le nom de l'algorithme et ses paramètres.
   - **freq ← CALCULER_FREQUENCE(chaine)** : Calcule la fréquence des caractères dans la chaîne donnée.
   - **pq ← CRÉER_FILE_DE_PRIORITÉ(freq)** : Crée une file de priorité à partir des fréquences des caractères.
   - **TANT_QUE LONGUEUR(pq) > 1 FAIRE ... FIN TANT_QUE** : Boucle qui fusionne les nœuds de la file de priorité jusqu'à ce qu'il ne reste qu'un seul nœud.
   - **DEBUT** : Marque le début de la section principale de l'algorithme.
   - **LIRE_CHAINE("Entrez une chaîne de caractères : ")** : Demande à l'utilisateur d'entrer une chaîne de caractères.
   - **Parcours(arbre, chemin, codage)** : Procédure récursive pour parcourir l'arbre Huffman et générer les codes binaires pour chaque caractère.
   - **codage_binaire ← " ... RETOURNER codage_binaire** : Génère la chaîne de bits pour la chaîne de caractères donnée.
   - **FIN** : Marque la fin de la section principale de l'algorithme.

Ces explications détaillent chaque partie des algorithmes, y compris les boucles, les conditions, les appels de fonctions et les structures de données utilisées pour résoudre les problèmes spécifiques de chaque exercice.