Algorithme : InsertionTriée (liste simplement chaînée)

```
Variables :
    Liste : pointeur vers le premier nœud de la liste (peut être nul)
    N : nœud à insérer
    courant : pointeur de parcours
    x : entier à insérer

Début
    1. Créer un nouveau nœud N
    2. N.val ← x
    3. N.suivant ← NULL

    4. Si (Liste = NULL) OU (Liste.val ≥ x) Alors
           N.suivant ← Liste
           Liste ← N
           Retourner
       FinSi

    5. courant ← Liste

    6. TantQue (courant.suivant ≠ NULL) ET (courant.suivant.val < x) Faire
           courant ← courant.suivant
       FinTantQue

    7. N.suivant ← courant.suivant
    8. courant.suivant ← N
Fin
```

Algorithme : InsertionTriée_Double (liste doublement chaînée)

```
Variables :
    Liste : pointeur vers le premier nœud (peut être nul)
    N : nœud à insérer
    courant : pointeur de parcours
    x : entier à insérer

Début
    1. Créer un nouveau nœud N
    2. N.val ← x
    3. N.suivant ← NULL
    4. N.precedent ← NULL

    5. Si (Liste = NULL) Alors
           Liste ← N
           Retourner
       FinSi

    6. Si (x ≤ Liste.val) Alors
           N.suivant ← Liste
           Liste.precedent ← N
           Liste ← N
           Retourner
       FinSi

    7. courant ← Liste

    8. TantQue (courant ≠ NULL) ET (courant.val < x) Faire
           courant ← courant.suivant
       FinTantQue

    9. N.suivant ← courant

    10. Si (courant ≠ NULL) Alors
            N.precedent ← courant.precedent
            courant.precedent.suivant ← N
            courant.precedent ← N
        Sinon
            trouver dernier
            dernier.suivant ← N
            N.precedent ← dernier
        FinSi
Fin
```

Algorithme : InsertionEnTête (liste simplement chaînée)

```
Variables :
    Liste : pointeur vers le premier nœud (peut être nul)
    N : nœud à insérer
    x : entier à insérer

Début
    1. Créer un nouveau nœud N
    2. N.val ← x
    3. N.suivant ← Liste
    4. Liste ← N
Fin
```

Algorithme : InsertionEnQueue (liste simplement chaînée)

```
Variables :
    Liste : pointeur vers le premier nœud (peut être nul)
    N : nœud à insérer
    courant : pointeur de parcours
    x : entier à insérer

Début
    1. Créer un nouveau nœud N
    2. N.val ← x
    3. N.suivant ← NULL

    4. Si (Liste = NULL) Alors
           Liste ← N
           Retourner
       FinSi

    5. courant ← Liste

    6. TantQue (courant.suivant ≠ NULL) Faire
           courant ← courant.suivant
       FinTantQue

    7. courant.suivant ← N
Fin
```

Algorithme : InsertionEnTête_Circulaire (liste doublement chaînée circulaire)

```
Variables :
    tête : pointeur vers le premier nœud (peut être nul)
    N : nouveau nœud à insérer
    dernier : pointeur vers le dernier nœud
    x : entier à insérer

Début
    1. Créer un nouveau nœud N
    2. N.val ← x

    3. Si (tête = NULL) Alors
           N.suivant ← N
           N.precedent ← N
           tête ← N
           Retourner
       FinSi

    4. dernier ← tête.precedent

    5. N.suivant ← tête
    6. N.precedent ← dernier
    7. tête.precedent ← N
    8. dernier.suivant ← N
```

```
    9. tête ← N
Fin
```

Algorithme : InsertionEnQueue_Circulaire (liste doublement chaînée circulaire)

```
Variables :
    tête : pointeur vers le premier nœud (peut être nul)
    N : nouveau nœud à insérer
    dernier : pointeur vers le dernier nœud
    x : entier à insérer

Début
    1. Créer un nouveau nœud N
    2. N.val ← x

    3. Si (tête = NULL) Alors
           N.suivant ← N
           N.precedent ← N
           tête ← N
           Retourner
       FinSi

    4. dernier ← tête.precedent

    5. N.suivant ← tête
    6. N.precedent ← dernier
    7. dernier.suivant ← N
    8. tête.precedent ← N
Fin
```