IA C.A.T

Cette liste ne concerne qu'un seul fichier source

On veut stocker pour chaque information récupérer la ligne ou elle apparait

Il reste encore à définir tout ce qui en rapport avec les langages objets

- Liste des bibliothèques (Et/ou fichiers externes) utilisés
- Liste des types (re)définis dans le code (Nouveau type ex: Une structure, typedef)
- Récupérer la liste des variables globales
- Traiter le code exécuter automatiquement comme une fonction principale (Ex: main en C, ou un code sans indentation en Python)
- Liste des classes (Définies dans le code source)
  - A définir
- Liste des fonctions (Définies dans le code source, != Pas de méthodes)
- Pour chaque classe récupérer la liste des constructeurs
- Pour chaque classe récupérer la liste des méthodes
- Pour chaque classe récupérer la liste des attributs et ses modifieurs/qualifiers
- Récupérer la visibilité des attributs (Ex: Public, Private, Protected)
- Pour chaque déclaration variable/attribut (simple) récupérer
  - son nom
  - son type
  - ses modifieur de type (unsigned, signed, short, long)
  - ses qualifieurs (const et volatile)
  - ses classes d'enregistrements (auto, extern, register, static)
- Récupérer les déclaration de pointeur
- Récupérer les déclarations structurées
  - tableau
  - dictionnaire
  - pile
  - file
  - etc
-
- Pour chaque fonctions/méthodes récupérer la profondeur maximum
- Pour chaque fonctions/méthodes récupérer la liste de :
  - des boucles dans l'ordre d'apparition (Champs commun à toutes les boucles : types et niveau)
  - if
  - else
  - else if
  - des appels de fonctions/méthode avec la liste des paramètres en chaine de caractères.
  - nombre de variable initialisées

    - nombre de switch
    - nombre de case
    - nombre d'apparition de chaque opérateur
        - "+" : plus
        - "-" : moins
        - "x" : fois
        - "/" : divise
        - "%" : modulo
        - "&&" : et_logique
        - "||" : ou_logique
        - "&" : et_binaire
        - "|" : ou_binaire
        - "^" : ou_exclusif_binaire
        - "<<" : décalage_binaire_droite
        - ">>" : décalage_binaire_gauche
        - "=" : affectation
        - "==": egalité
        - "!=" : différence
        - "<" : inferieur
        - "<=" inferieur_egale
        - ">" : supérieur
        - ">=" : supérieur_egale
        - "++X" : pre_incrementation
        - "X++" : post_incrementation
        - "--X" : pre_decrementation
        - "X--" : post_incrementation
        - "&X" : opérateur_referencement
        - "*X" : operateur_dereferencement
        - "(type)" : operateur_transtypage
        - "+="
        - "-="
        - "*="
        - "x="
        - "/="
        - "%="
        - "^="
        - "&="
        - "|="
        - "<<="
        - ">>="
        - "," : séquencement
        - "?": operateur_ternaire
- Pour chaque fonction/methode:
    - nombre de swap
    - nombre d'initialisation de tableau
    - nombre de copie de tableau
    - nombre de decalage de tableau
    - nombre de recherche dans un tableau
    - structure de la fonction (A déterminer précisément)
- Pour chaque affectation (ou affectation composé) je veux récupérer :

    - l'element gauche
    - l'element droit
    - La liste des operateurs de l'element droit
    - La liste des variable de l'element droit
-

```
a = "1"
b = 1

if a == b: True
if a is b: False
```

```
{
    "boucles": [
        "enumeration": [
            {
                "type": "for",
                "init": "i=5",
                "cond": "i<10+(i/2)",
                "step": "i=i*2"
            }
        ],
    ]
}
```

```
for(i in range(0, 10)):
    ...
    while(j < 10)

for(k in range(0, 10)):
    ...
    while(l < 10)
```

```
{
    "classes": [
        "Lampe": {
            "attributs": [
                "toto": {
                    "type": "String"
                    "visibility": "Public"
                }
            ]
        }
    ]
}
```