Formation : Analyste développeur 2020-2021

ParCnator

Nicolas LEJEUNE
Gabriel LEFFAD
Kevin IRLES
David HERZOG

Sommaire

- **Introduction**
  - Présentation de l’équipe
  - Demande client
  - Analyse fonctionnelle
- **Gestion de projet**
  - Ressources, environnement et tarification
  - Méthode de gestion de projet choisie
  - Outils et technologies
- **Conception**
  - Release plans
  - Architecture
  - Difficultés techniques et solutions
- **Conclusion**
  - Démonstration
  - Objectif des prochains “release plans”

Présentation de l’équipe

Nicolas LEJEUNE

JAVA, C, Python

Gabriel LEFFAD

JAVA, C, Python

David HERZOG

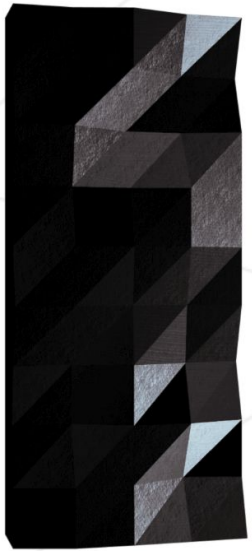
JAVA, C, Python

Kevin IRLES

PHP, JAVA, JavaScript

CFA INSTA
Institut National Supérieur des Technologies Avancées

Demande du client



La norme de 42

- ➔ Analyse et évaluation de code écrit avec le langage C
- ➔ Bloquer la livraison de code qui ne compile pas ou qui ne respecte pas la norme 42.

CFA INSTA
Institut National Supérieur des Technologies Avancées

Analyse fonctionnelle

- Tokenization
  - Lecture du fichier de code
  - Identification de chaque portion de string correspondant à un élément atomique du langage (*;, {, }, (,), ...*)
  - Transformation de ces portions de string en jetons (tokens) → *premier niveau d'abstraction*

- Création de l'arbre syntaxique (Parser)
  - Analyse des tokens créés par le Tokenizer selon les règles syntaxiques du langage C
  - Dressage d'un arbre de syntaxe abstraite (AST) → *deuxième niveau d'abstraction*

- Scoring
  - Etude du code selon la norme 42 (indentation, nombre de lignes total / dans chaque fonction)

Conditions, environnement et tarification

Ressources :

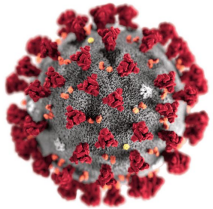
- 4 développeurs
- 4 jours
- 16 jours.homme (400euros / JH)

Deadline :

- 30.04.2021

Adaptation au distanciel :

- SRAS-COV-2
- Télétravail
- Adaptation du choix de la méthode de gestion de projet
- Établissement de méthode de communication
- Établissement du workflow
- Choix des outils adaptés
- Management en P2P



Prix TTC : <u>6400 euros</u>

Méthode de gestion de projet choisie

Extreme Programming :

→ S'adapte aux spécifications changeantes

→ Gestion des risques encourus lors de l'utilisation de nouvelles technologies

→ Releases fréquentes

→ Temps de développement court

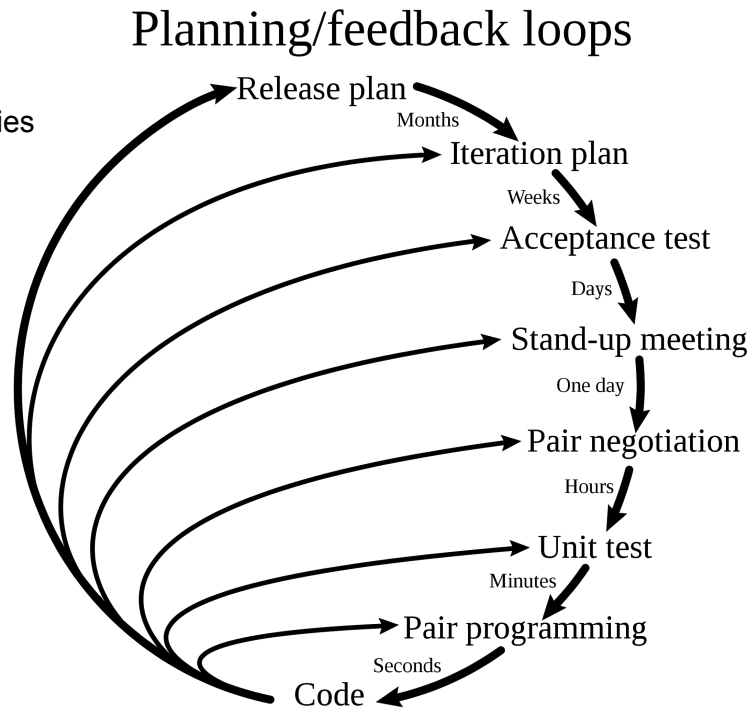
→ Nombre de checkpoints réduit

→ Tests fréquents

→ Adapté au management en P2P (meetings)

→ 5 Valeurs :

- communication (transfert de connaissance)
- simplicity (K.I.S.S. principle)
- feedback (identifier les zones d'amélioration, revoir les pratiques)
- courage ("effective action in the face of fear" Kent Beck)
- respect (fournir et accepter les retour afin de conserver des relations constructives uniquement)



Outils et technologies

Communication : Discord, TeamViewer

Gestion de version : GitHub, Git

Outils de développement : Visual Studio Code

Language : Javascript

Runtime environnement : Node.js




Outils et technologies

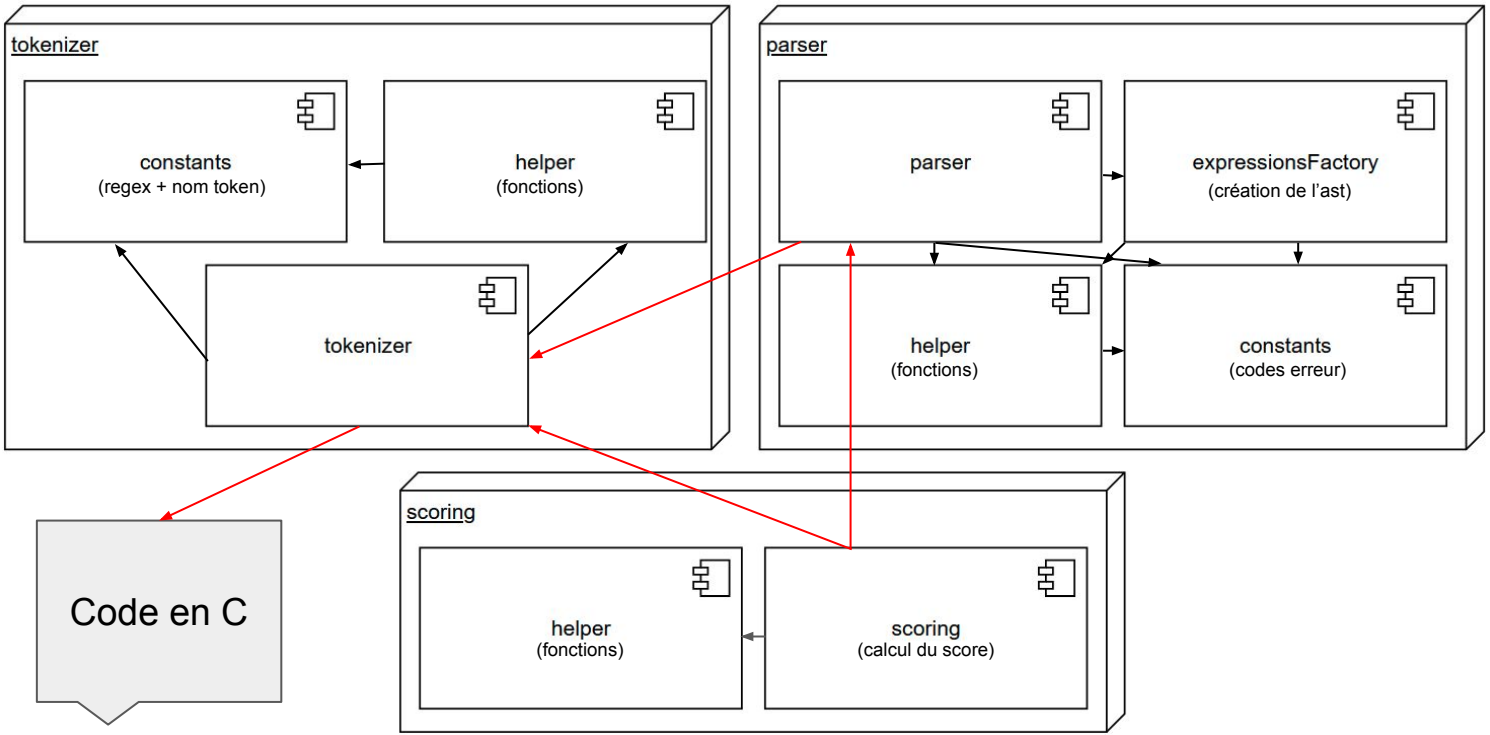
Javascript	Node.js	Alternatives
+ flexible + support + multi-paradigme - abstrait	+ environnement simple + debuggage possible + même process que le C avec la flexibilité du JS	- Java - C - Python - Deno - etc...

CFA INSTA
Institut National Supérieur des Technologies Avancées

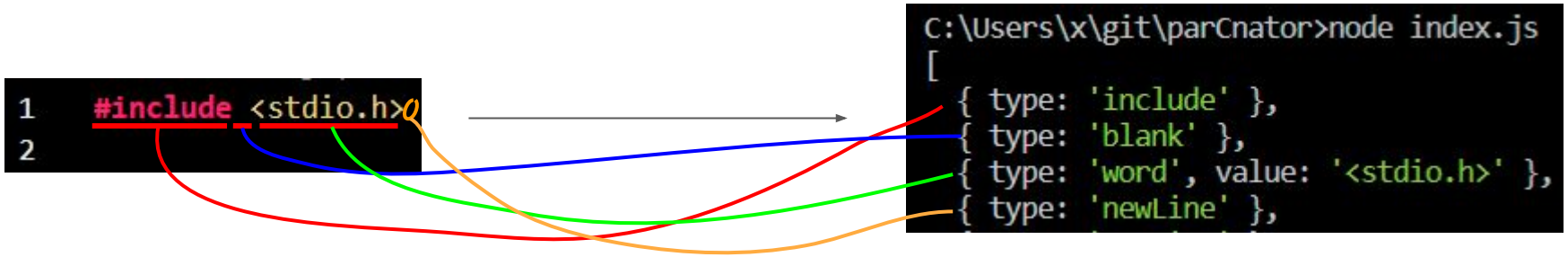
Release plans

- Analyse fonctionnelle des besoins
- Identification des différentes règles syntaxiques du langage C
- Définition des différents jetons
- Implémentation de la fonction de Tokenization
- Définition de la structure de l’arbre syntaxique
- Création de la partie permettant de créer l’arbre
- Création de la fonction de Parsing permettant de parcourir les tokens et de piloter la création de l’arbre
- Implémentation des fonctions de calcul des scores

Architecture

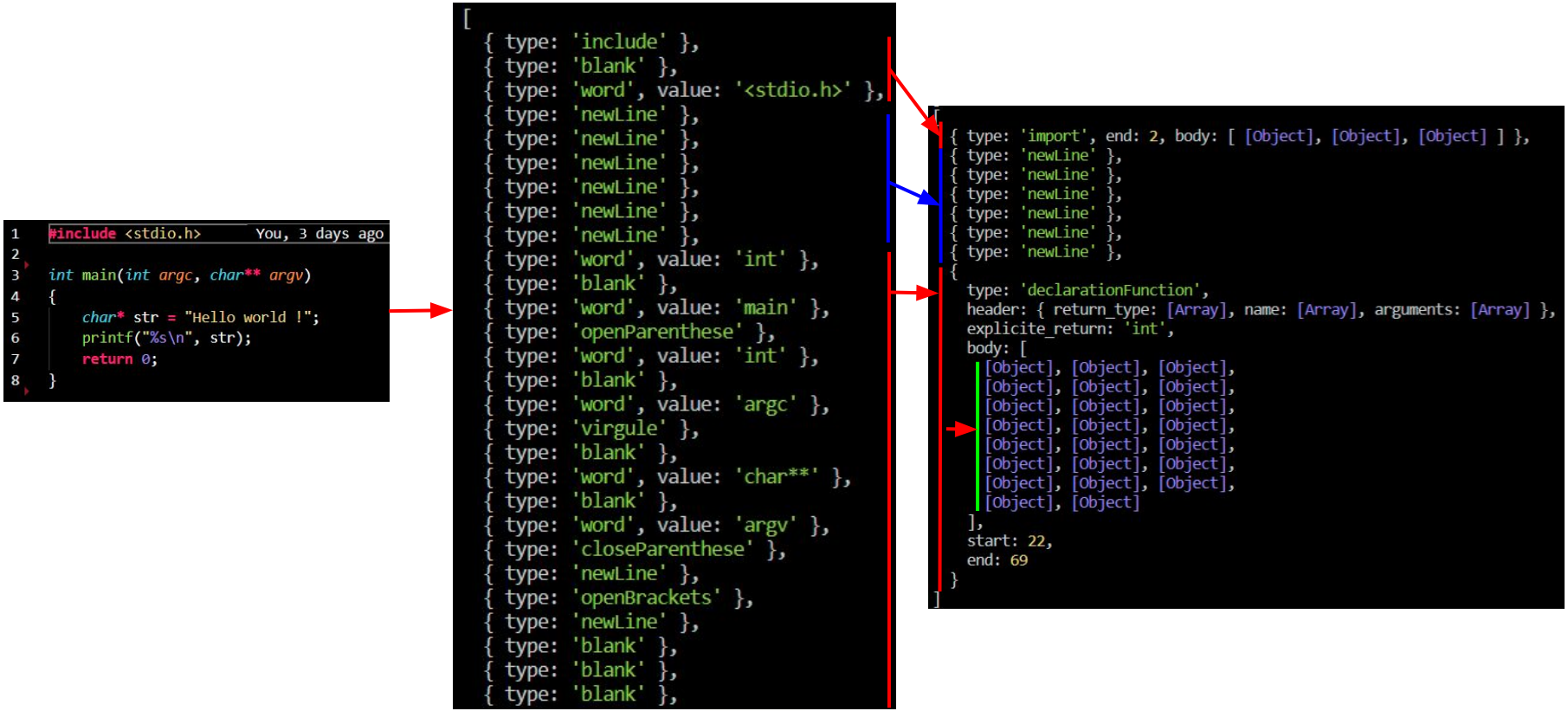


Tokenization (level 1)

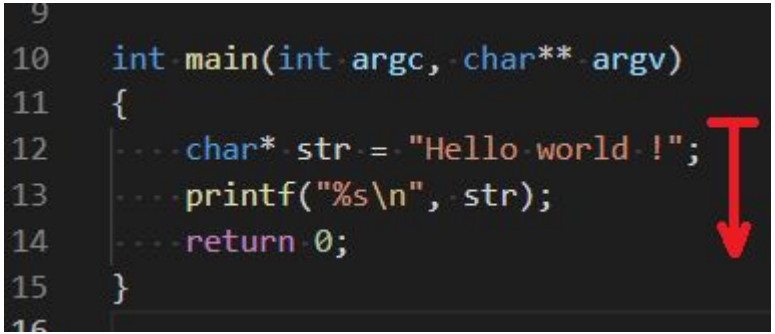


→ Transformation du code en entrée en string
→ Séparation des éléments du langage
→ Conversion de ces éléments en token
→ Création du tableau de tokens

Parser : construction de l'arbre syntaxique (level 2)



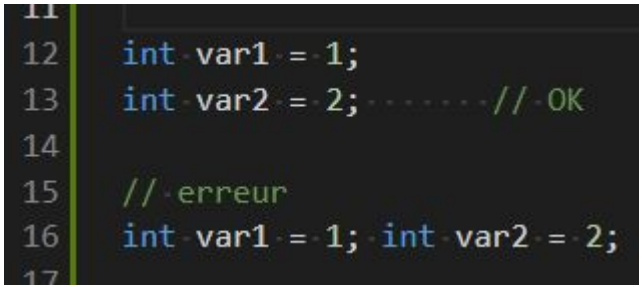
Scoring



Vérification du nombre de lignes

- **fichier** :
  - 200 lignes maximum
- **corps de fonction** :
  - parcours de l'AST:
  - 25 lignes maximum

Vérification `;` suivi par un retour à la ligne



Scoring

```
printf("%s\n", str);    // OK

printf("%s\n" , str);   // 1 erreur
printf("%s\n",str);     // 1 erreur
printf("%s\n" ,str);    // 2 erreurs
```

Vérification de la position des virgules

- une virgule est toujours:
  - précédée par un caractère (pas un espace)
  - suivie par le caractère ‘espace’

Vérification de la terminaison des expressions

- caractères { } [ ] ( ) “ ”
  - par paires
  - pas d'entrelacement

```
int tab = [ ["foo"], ["bar"] ];  // OK
int tab = [ ["foo""], ["bar"] ]; // erreur

[ () () {[]} ]                   // OK
[ { ] }                          // erreur
```

Difficultés techniques et solutions

4 principaux défis :

→ Bien comprendre les attentes du client

→ Regex pour les tokens

→ Création de l'arbre syntaxique

→ Utilisation de l'arbre syntaxique pour le scoring


DEMO

Objectif des prochains “release plan”

→ Affiner la tokenisation (pointeurs multiples, déclaration de tableau, structures)

→ Ajouter les nouveaux tokens et les structures à l’arbre syntaxique

→ Affiner le scoring afin d’avoir un barème progressif

Questions ?

Merci