RAPPORT DE PROJET

# Fonctionnement général

Le code C est généré ligne par ligne lors de la lecture des instructions en langage algorithmique. L’indentation est faite automatiquement en fonction du niveau de l’instruction courante. Chaque fonction, à sa création, est attribuée une table de symbole (table de hachage) afin d’y stocker ses variables locales. Ces tables de hachage sont stockées dans une liste de tables de hachage et peuvent être cherchées lors des appels de fonctions afin de procéder aux vérifications nécessaires.

# Grammaire

Le retour à la ligne est utilisé comme symbole terminal.

La grammaire est composée de plusieurs règles. La première règle est la règle « program » qui indique que la définition des fonctions se fait toujours avant la définition de l’algorithme. La règle « function » est récursive et permet de définir plusieurs fonctions à la suite. Cette règle accepte des fonctions avec ou sans arguments, avec ou sans type de retour (procédures). Chaque fonction est ensuite composée d’expressions (règle « expression »). Une expression est soit une déclaration soit une instruction, le tout récursivement. Le programme ne se termine que lorsqu’il a lu le mot « Fin » de l’algorithme et non pas des fonctions.

## Déclaration

La déclaration d’une variable ne nécessite pas de préciser le type, celui-ci est déterminé grâce au token retourné par le lexer. Cette variable est stockée dans la table de symboles de la fonction courante, ou celle de l’algorithme.

## Instruction

Une instruction peut être une condition, un appel de fonction, une boucle ou l’instruction de retour.

### Condition

Une condition est de la forme « Si CONDITION OPERATEUR VALEUR Alors », suivie d’expressions, éventuellement suivie d’un « Sinon », lui-aussi également composé d’expressions.

### Boucle

Il est proposé deux types de boucles : Tant Que et Pour.

La boucle Tant Que accepte une expression booléenne simple (ex : vrai, !faux) ou une condition plus complexe (ex : a == 2, (a\*5/4 - b) == 0).

La boucle Pour accepte une variable entière préalablement déclarée ou non, puis deux bornes entières (de X à Y). Optionnellement, il est possible de préciser le « pas » de la boucle. Si la variable précisée n’existe pas elle est créée et attribuée le type entier.

### Appel de fonction

Il existe deux fonctions prédéfinies : Ecrire, Lire.

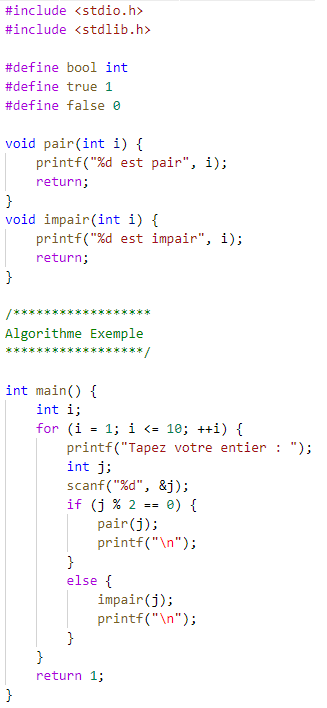
La fonction Ecrire prend en argument une chaîne de caractères possiblement concaténée avec des variables. C’est la règle « write » qui reconnaît cette chaîne de caractère, qui appelle la règle « full\_string ». Cette dernière est récursive gauche afin de pouvoir concaténer autant de variables que nécessaire. Le format qui sera passé au printf est créé en fonction des paramètres reçus, qui chaque variable est placée dans une file. Lorsque toutes les variables sont traitées, nous obtenons un format de chaîne (ex : « Test n°%d sur %d »). Les variables/fonctions nécessaires à son affichage sont ensuite ajoutées grâce à la file.

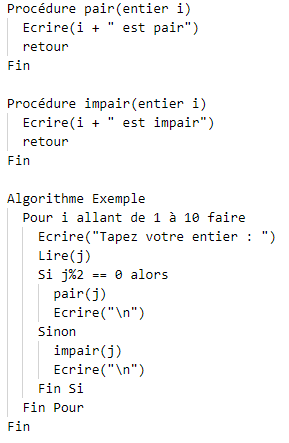
La fonction Lire prend un seul argument : une variable. Celle-ci est déclarée automatiquement en entier si non-existante et produit un scanf.

Pour toute autre fonction créée par l’utilisateur, l’existence de celle-ci est vérifiée puis formatée avec les paramètres passés. Le type des paramètres est comparé aux types des paramètres attendus, ainsi que le nombre de paramètres.

### Instruction de retour

Produit « return ; » pour les procédures et « return variable ; » pour les fonctions. La type de la valeur de retour est vérifié et comparé à celui attendu (à partir de la définition). Une procédure retournant une valeur générera une erreur.

Exemple 1 – Algorithmique et C



Exemple 2 – Algorithmique et C

