# Rapport Compilation

Dans ce rapport, je vais énumérer les fonctionnalités de mon compilateur.

Il faut lire le README pour savoir comment compiler et exécuter les différents programmes (.txt) de test sur la machine virtuelle.

Tous les tests que j’ai réalisés afin de dire si une fonctionnalité marche ou non sont compris dans mon projet.

## Liste des fonctionnalités :

### Opérateurs unaires

Fichier de test : testOperateur1.txt

État : fonctionnel

Explications : /

### Opérateurs binaires

Fichier de test : testOperateur2.txt

État : fonctionnel

Explications : /

### Reste des expressions “constantes”

Fichier de test : testConstantes.txt

État : fonctionnel

Explications : /

### Conditionnels (if, then, else, ambiguïté sans accolades, …)

Fichier de test : testIf.txt

État : fonctionnel

Explications : /

### Boucles (continue, break⇒imbrication), while, for, do/while, …)

Fichier de test : testBoucles.txt

État : fonctionnel

Explications : /

### Variables (portée, affectation…)

Fichier de test : testVariables.txt

État : fonctionnel

Explications :

* ATTENTION à l’affectation des variables, on ne peut pas faire : « int i = 0;», à la place le faire en 2 étapes « int i ; i=0 ; »
* Les variables globales (non déclarées dans le main) crée une erreur car elles ne sont pas déclarées dans un bloc.

### Reste des instructions (structure de blocs, debug, …)

Fichier de test : testVariables.txt

État : fonctionnel

Explications :

* ATTENTION à l’affectation des variables, on ne peut pas faire : « int i = 0;», à la place le faire en 2 étapes « int i ; i=0 ; »
* Les variables globales (non déclarées dans le main) crée une erreur car elles ne sont pas déclarées dans un bloc.

### Fonctions (appels de fonctions, imbrication d’appels dans les appels, récursivité des fonctions, return, avec paramètres ?, …)

Fichier de test : testFonctions.txt

État : fonctionnel

Explications : /

### Pointeurs (\*, &, [])

Fichier de test : testPointeur1.txt, testPointeur2.txt, testPointeurTab.txt

État : fonctionnel

Explications : /

### Bibliothèque standard (malloc, free, send, recv)

Fichier : bibliothèqueStandard.txt testRecvSend.txt

État : semi-fonctionnel

Explications :

* Le print affiche un résultat correct après l’utilisation d’un read uniquement.
* Lors d’un print je dois rentrer 2 fois ce que je souhaite afficher comme s’il ne détecte pas le 1er retour à la ligne (==10) du read…

Voici la sortie sur mon mac :

coucou //read sur « coucou »

����coucou //print donne « ���� », je saisie « coucou » à nouveau

�

coucou //le print affiche le coucou

34

### Optimisations

La fonction d’optimisation se trouve ligne 280 dans node.hpp.

Dans cette fonction j’optimise les expressions constantes (ex : i =2+5).

Ce sont des expressions dont les valeurs peuvent être déterminées au moment de la compilation plutôt qu'à l'exécution.

Ici j’optimise l’addition, soustraction, multiplication, modulo, moins que, moins ou égal, plus grand que, plus grand ou égal, égalité, inégalité et les opérations logique OU et ET.