Mehdi Fracso
Félix Dayet

Rapport TP4

Nous avons tout d'abord installé l'ensemble des composants nécessaire pour le NDK android. Tout d'abord avec l'activation du support pour NDK ainsi que l'acceptation des licences nécessaires pour autoriser le téléchargement des différentes dépendances requises pour le développement.
Pour accepter ces licences, nous avons utilisé l'outil sdkmanager (android_sdk/tools/bin/).

Création de l'application NDK

Nous avons créé un nouveau projet de type native C++. Par la suite nous sommes parti sur le layout, afin d'ajouter les boutons nécessaires de type "Button" (up, down, left, right, read, write) ainsi que le champ de saisie de type "EditText" et des zones d'affichage de texte de type "TextView" (read, write). Puis nous avons attribué à chacun d'entre eux leurs contraintes respectives, et avons procédé à une modification de leur id.
Ci-dessous une capture du layout de la MainActivity :

Capture du layout de la MainActivity

Par la suite nous avons effectué les différentes étapes demandées :

- Le bouton READ appelle une fonction native avec un nombre compris entre 0 et 10 qui va retourner le texte affiché dans le label "READ : a*a", a étant le nombre passé à la fonction.
- Le bouton WRITE appelle une fonction native avec un nombre compris entre 0 et 10 2 qui va retourner le texte affiché dans le label "WRITE : a*a*a", a étant le nombre passé à la fonction
- Les boutons LEFT, RIGHT, UP et DOWN appellent une fonction native qui va afficher le texte de la direction correspondante en Japonais en passant en paramètre le nom des boutons.

Nous avons commencé par la déclaration des signatures des fonctions natives au niveau de la classe MainActivity.java :

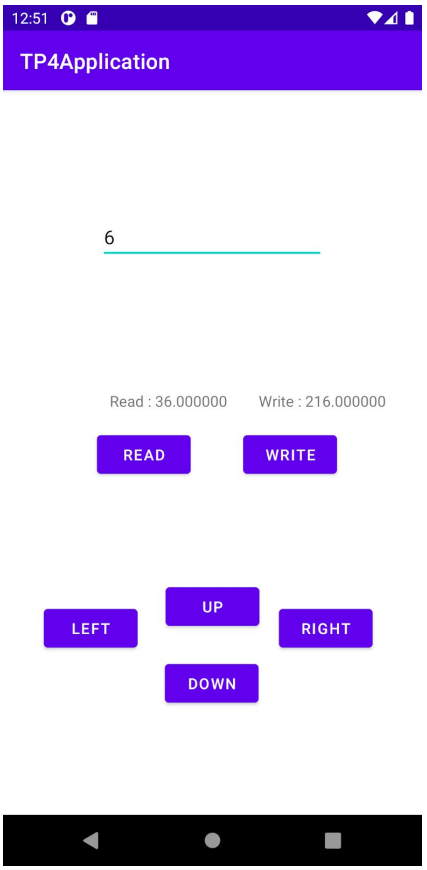
```
public native String calculateSquare(int value);
public native String calculateCube(int value);
public native String getJapaneseEquivalent(int direction);
```

Puis Android studio nous a proposé de générer les définitions de ces fonctions dans le fichier c++ (native-lib.cpp) . Nous l'avons donc appliqué et par la suite, nous nous sommes consacré à compléter ces différentes définitions avec le code c++ nécessaire.

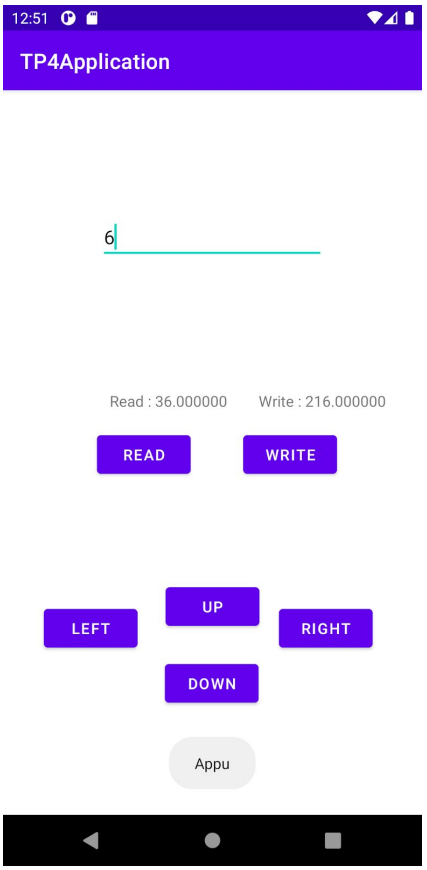
Le challenge ici a été pour nous de convertir les différents types de données échangées car pour faire les calculs en c++, il nous fallait convertir les jint en int, nous avons utilisé un cast pour cela. Pour retourner un jstring à la fin de nos fonctions qui nous donnaient un string, nous avons utilisé une fonction pour convertir le type (env->NewStringUTF(to_string(valeurEntiere).c_str())).

Pour finaliser le tp, nous avons enfin appliqué la traduction en japonais des directions souhaitées par l'utilisateur pour les afficher grâce à des toasts lors de l'appui des boutons. Nous avons aussi affecté à chaque bouton les event handlers correspondants.

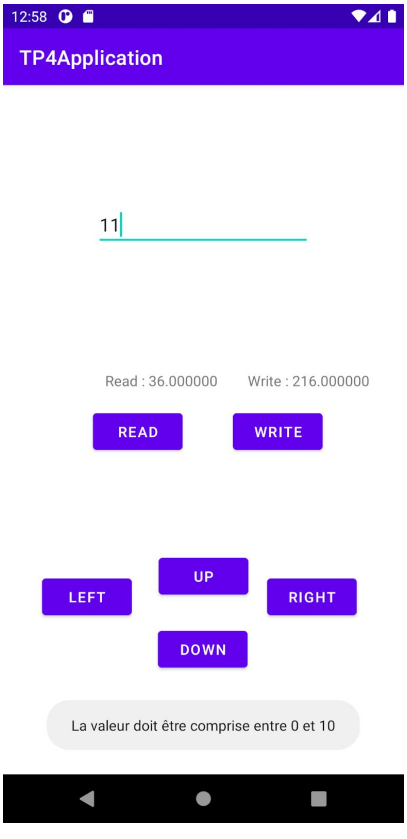
Voici différentes captures illustrants notre travail :



Appui sur les boutons read et write pour afficher le cube et le carré du nombre souhaité



Appui sur le bouton up, déclenchant l'affichage du toast Appu



Message d'erreur pour une valeur non comprise entre 0 et 10