MODULE 06
SÉANCE SYSTÈME 04
TP D'INFORMATIQUE
Durée 2h30

La classe IRSnifferCirpark – détection des capteurs

BLOC DE COMPÉTENCES
U6 - VALORISATION DE LA DONNÉE ET CYBERSÉCURITÉ

COMPÉTENCE(S)
C08 - CODER

OBJECTIF PÉDAGOGIQUE
Dans cette partie, vous travaillerez sur des tableaux de données. Il faudra parcourir des tableaux, les afficher selon un format prédéfini et les enregistrer les données qu'ils contiennent dans un fichier.

CONNAISSANCES ISSUES DU RÉFÉRENTIEL
- Programmation Orientée Objet — Niveau 3

CONNAISSANCES OPÉRATIONNALISÉES
- Utiliser une classe — Niveau 2
- Créer une classe — Niveau 2

Gaëtan GEORGES
Pascal HAGOT
David IERVESE

Lycée Louis Armand
173 Bd de Strasbourg
Nogent sur Marne

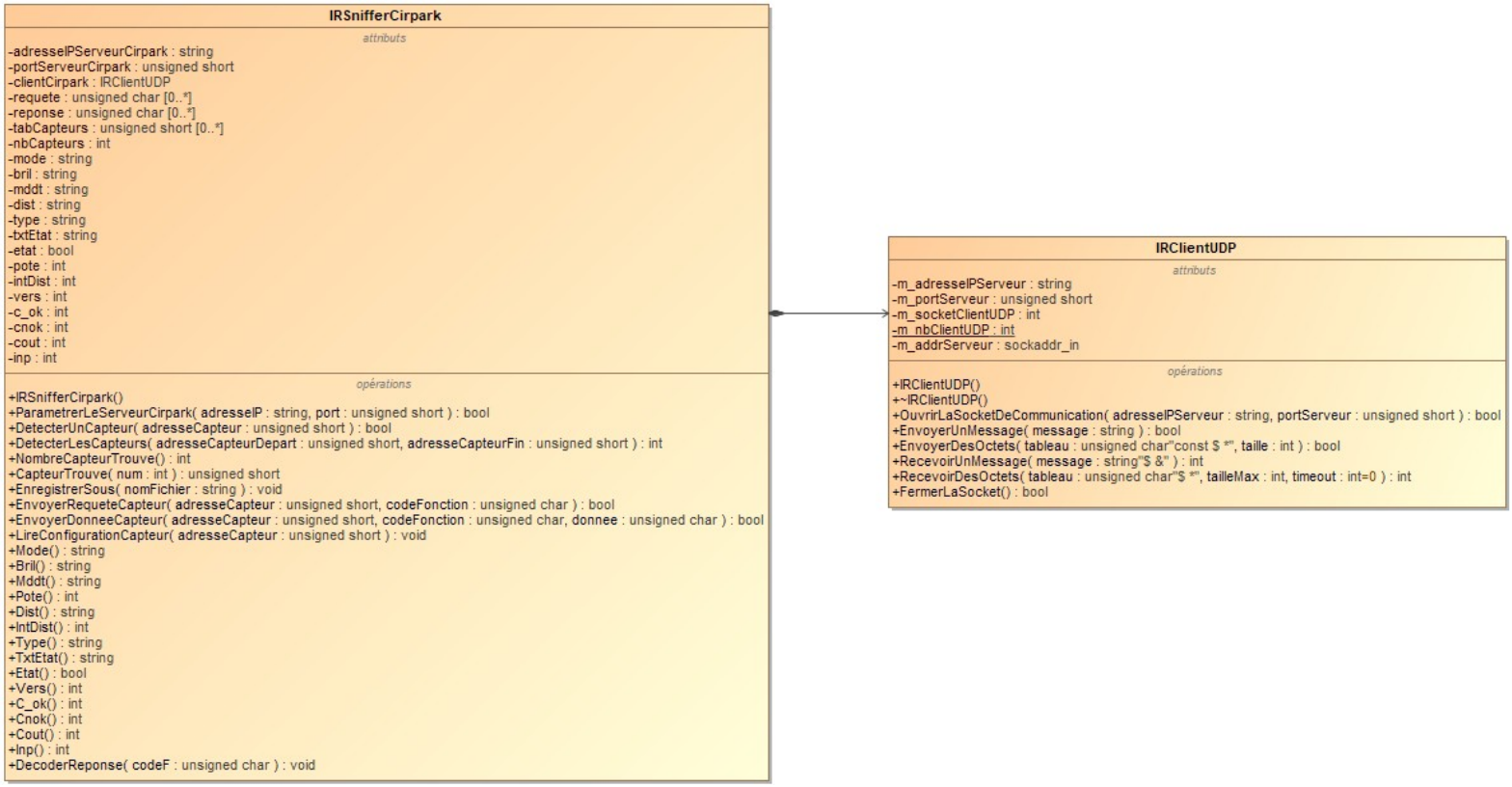
Écriture de la classe IRSnifferCirpark

Présentation

Dans cette deuxième partie, nous allons travailler sur la méthode qui permet de connaître automatiquement la liste des capteurs présents sur une plage d'adresse : c'est la méthode DetecterLesCapteurs() de la classe IRSnifferCirpark.

Le diagramme de classe

Pour rappel, voici le diagramme de classe complet :



La méthode DetecterLesCapteurs()

La définition de la méthode DetecterLesCapteurs()

Cette méthode permet à l'utilisateur de connaître la liste des capteurs présents dans un intervalle d'adresse. Par exemple, si nous souhaitons connaître le nombre de capteur entre les adresses 0x0000 et 0x1FFF, on écrira :

```
sniffer.DetecterLesCapteurs(0x0000, 0x1FFF);
```

Donner le nombre d'argument attendu par la méthode DetecterLesCapteurs() et leur type. Justifier le type choisi.

Des unsigned short on prend une plage de départ et une plage de sortie

Donner le type de retour de cette méthode.

int

Cette méthode renvoie l'attribut nbCapteurs (qui contient le nombre de capteurs trouvés). Est-ce compatible avec le type de retour de la méthode ? Justifier.

Oui c'est compatible puisque nbCapteurs est un int et la fonction doit renvoyer un int

Au début de cette méthode, vous initialiserez nbCapteurs à 0 et vous remplirez le tableau tabCapteurs de 0. Ecrire le code correspondant.

```
int nbCapteurs = 0;
int tabCapteurs[100] = {0};
```

Pour détecter combien de capteurs sont présents dans une plage d'adresse, nous allons nous aider de la méthode DetecterUnCapteur() qui nous indique si un capteur est présent... Et nous allons répéter du premier capteur demandé au dernier capteur demandé.

A votre avis, quelle boucle est préférable : for, while ou do ... while ? Justifier ?

for est mieux puisqu'on evite de bloquer le code en cas d'erreur et cela s'arrete a partir du moment ou on interroge le dernier capteur

De quel type sera la variable de boucle ? Justifier. Vous la nommerez « adr ».

int

Écrire la boucle permettant de parcourir tous les capteurs demandés. Laisser le contenu de la boucle vide pour le moment.

A l'intérieur de cette boucle :

- appeler la méthode DetecterUnCapteur() et passer lui en argument « adr ».
- si la valeur de retour de cette méthode vaut « vrai », alors :
  - affecter la valeur « adr » à l'élément « nbCapteurs » du tableau tabCapteurs.
  - incrémenter l'attribut nbCapteurs.

En C++, écrire le code de la boucle.

```
nbCapteurs = 0;

for (int i=0; i < NBCAPTMAX; i++) { tabCapteurs[i] = 0;    }

unsigned short tot = adresseCapteurFin - adresseCapteurDepart;

for (int adr = 0; adr < tot; adr++) {

bool rep = DetecterUnCapteur((adresseCapteurDepart + (unsigned short)adr));
```

> Pour terminer, une fois la boucle terminée, retourner la valeur de nbCapteurs. Ecrire le code correspondant.

```
return nbCapteurs
```

Test de la méthode DetecterLesCapteurs()

Dans le programme principal, appeler la méthode DetecterLesCapteurs() dans la plage 0x01a0 à 0x01ff. Puis afficher la valeur retournée par cette méthode.

> Ecrire le code C++ correspondant.

> Vérifier que cela correspond bien avec la réalité. Lancer Wireshark pour vérifier que vos trames émises sont correctes.

La méthode NombreCapteurTrouve()

Cette méthode est une méthode d'accès qui renvoie l'attribut nbCapteur.

> Ecrire sa définition en C++ et la tester dans votre programme principal.

La méthode CapteurTrouve()

Cette méthode est une méthode d'accès au tableau tabCapteurs. Elle renvoie le capteur qui se trouve à l'indice num passé en argument de cette méthode.

> Ecrire la définition de la méthode unsigned short CapteurTrouve(int num). Si l'argument num est inférieur à nbCapteur, cette méthode renvoie le capteur se trouvant à l'indice num dans le tablea tabCapteurs. Sinon, cette méthode renvoie 0.

> Tester cette méthode dans votre programme principal en affichant le premier capteur que votre sniffer a trouvé. Vous veillerez à afficher le numéro du capteur sur 4 chiffres en hexadécimal et en majuscule (en complétant avec des '0' devant si nécessaire). Voici comment réaliser un tel affichage :

```
   unsigned short capteur = 0x01ec;
   cout << uppercase << hex << setw(4) << setfill('0') << capteur << std::endl;
```

Pour utiliser les fonctions setw() et setfill(), il faut inclure le fichier :

```
#include <iomanip>
```

> Tester cette méthode dans votre programme principal en affichant tous les capteurs que votre sniffer a trouvé.

La méthode EnregistrerSous()

L'objectif de cette méthode est d'enregistrer dans un fichier tous les capteurs qui ont été trouvé. Le nom du fichier est passé en argument. Tous les capteurs seront enregistrés sur 4 chiffres en hexadécimal et en majuscule (en complétant avec des '0' devant si nécessaire).

> Rappeler le nom de la classe permettant d'écrire dans un fichier.

```

```

Dans la définition de la méthode EnregistrerSous() :
- créer un objet de la classe ofstream
- ouvrir le fichier passer en argument
- faire une boucle sur l'ensemble des capteurs trouvés et ajouter à chaque fois une nouvelle ligne dans le fichier en respectant le format imposé.

Tester la méthode EnregistrerSous() dans votre programme principal. Vérifier qu'un fichier est bien créé et qu'il contient bien la liste de tous les capteurs trouvés.