TP03 – BOUCLES SIMPLES ET IMBRIQUEES

Le but des exercices de cette séance de travaux pratiques est de se familiariser avec le codage en C++ des éléments suivants :

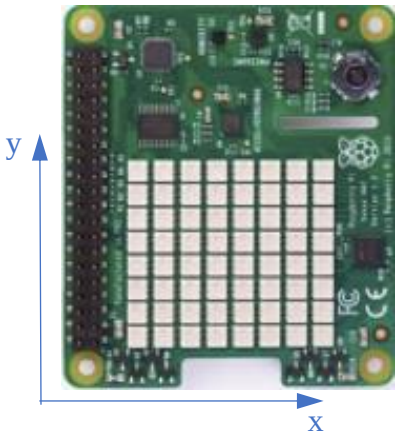
- Les boucles simples : TANT-QUE, REPETER ou POUR
- Les boucles imbriquées

Comme à chaque séance de travaux pratiques, il est nécessaire d'ouvrir un nouveau projet dans l'environnement de développement Visual Studio Code pour chaque exercice.

Les exercices peuvent être traités de façon indépendante mais il est conseillé de les aborder dans l'ordre car certaines ressources sont réutilisées. Les derniers exercices sont plus exigeants.

Exercice 1 – Allumer les pixels un par un

Écrire un programme qui allume en ROUGE successivement chacun des pixels de la première ligne (celle en bas à gauche sur l'image ci-dessous). L'allumage du pixel suivant est conditionné par l'accord de l'utilisateur.



Scénario :

Au démarrage, le pixel de coordonnées (0,0) s'allume,
Le programme interroge l'utilisateur pour savoir si Oui (appui sur la touche O) ou Non (appui sur la touche N) on doit allumer un autre pixel.
Si l'utilisateur saisit le caractère 'O', c'est au tour du pixel de coordonnées (1,0) de s'allumer.
Si l'utilisateur saisit le caractère 'N', le programme se termine.

Étape 1 – Préparation

Dessiner la vue externe du programme et proposer un jeu de tests sur les choix de l'utilisateur et les résultats visuels attendus.

Donner le type et le nom de la variable permettant de stocker le choix de l'utilisateur.

Quelle boucle allez-vous utiliser ? Justifier.

Faire l'algorithme de ce programme.

Étape 2 – Codage du programme

Ouvrir une session SSH sur la carte cible. À partir du terminal, lancer la commande :
`getLab.sh tp3-ex1`

Une fois le dossier créé, aller dans le **menu Fichier** et sélectionner **Ouvrir le dossier**. Choisir le dossier **tp3-ex1**.

Compléter le code source du fichier **tp3-ex1.cpp** :

- En ajoutant la déclaration de la variable dans la zone Début/Fin variable.
- En ajoutant la suite du code dans la zone Début/Fin Instructions.

Étape 3 – Test du programme

Vérifier la conformité des tests prévus dans l'étape 1.

Exercice 2 – Chenillard manuel

Reprendre le code de l'exercice 1 pour qu'à chaque fois qu'un nouveau pixel s'allume, le(s) pixel(s) précédemment allumé(s) passe(nt) en jaune.

Étape 1 – Préparation

Relativement à l'exercice 1, quels sont les changements à apporter sur la vue externe et le jeu de tests.

Faire l'algorithme de ce programme.

Étape 2 – Codage du programme

Commencer par copier la partie du code utile du fichier tp3-ex1.cpp.

À partir du terminal, lancer la commande : `getLab.sh tp3-ex2`

Une fois le dossier créé, aller dans le **menu Fichier** et sélectionner **Ouvrir le dossier**.

Choisir le dossier **tp3-ex2**

Coller le code du presse-papier vers le fichier **tp3-ex2.cpp**

Apporter les modifications nécessaires pour répondre au nouveau besoin.

Étape 3 – Test du programme

Vérifier la conformité des tests prévus dans l'étape 1.

Exercice 3 – Petite barre de progression automatique

Concevoir un allumage progressif des 8 pixels de la première colonne avec une temporisation d'une demi-seconde entre chaque nouvel allumage d'un pixel.

Indication : l'appel de la fonction suivante permet une temporisation de 10ms.

```
sleep_for(milliseconds(10));  // l'instruction endort le processus pendant 10ms
```

Étape 1 – Préparation

Dessiner la vue externe du programme.

Donner le type et le nom de la (des) variable(s) permettant :

- de compter le nombre d'itérations (nombre de passages dans la boucle),
- de repérer la(les) coordonnées du pixel qui doit s'allumer.

Quelle boucle allez-vous utiliser ? Justifier.

Faire l'algorithme de ce programme.

Étape 2 – Codage du programme

À partir du terminal, lancer la commande :

```
getLab.sh  tp3-ex3
```

Editer le code source du fichier **tp3-ex3.cpp** pour répondre au besoin.

Étape 3 – Test du programme

Vérifier le fonctionnement du programme.

Peut-on mesurer facilement la durée réelle entre chaque allumage de pixel ?

Exercice 4 – Remplissage automatique

Concevoir un programme qui allume successivement tous les pixels de la matrice senseHat, avec une temporisation de 200ms entre l'allumage d'un pixel et le suivant.

On propose de commencer par l'allumage de la ligne en bas (cf. photo page 1 pour l'orientation de la maquette).

Étape 1 – Préparation

Donner le type et le nom de la (des) variable(s) permettant :

- de compter le nombre d'itérations (nombre de passages dans la boucle),
- de repérer la(les) coordonnées du pixel qui doit s'allumer.

Quelle boucle allez-vous utiliser ? Justifier.

Faire l'algorithme de ce programme.

Étape 2 – Codage du programme

À partir du terminal, lancer la commande :

```
getLab.sh  tp3-ex4
```

Editer le code source du fichier **tp3-ex4.cpp** pour répondre au besoin.

Étape 3 – Test du programme

Vérifier le fonctionnement du programme.

Étape 4 – Grand Bargraphe automatique

Comment doit-on modifier le programme pour qu'il allume successivement toutes les colonnes de la matrice senseHat, avec une temporisation de 500ms entre l'allumage d'une colonne et la suivante ?

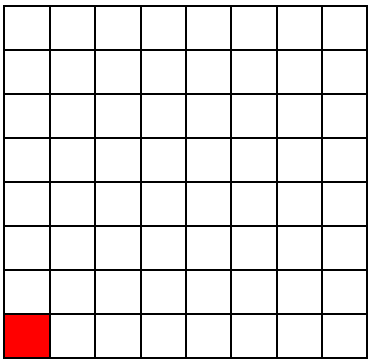
Étape 5 – Test du programme

Vérifier le fonctionnement demandé à l'étape 4.

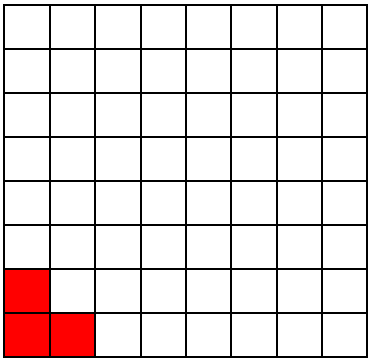
Exercice 5 – Remplissage automatique en diagonale

On souhaite maintenant que la matrice du senseHat s'allume progressivement par des diagonales successives avec 500ms d'attente entre l'allumage de 2 diagonales.

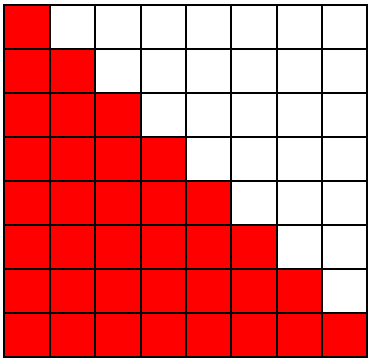
Au début c'est le "point" en bas à gauche qui doit s'allumer.



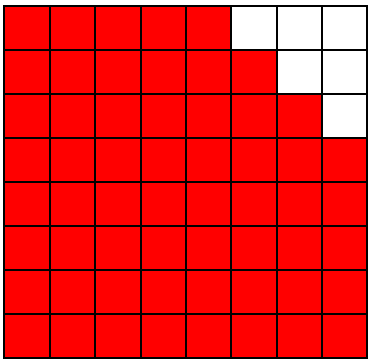
Au bout de 500ms, une 2ème diagonale apparaît :



Au bout de 3,5s, la 8ème diagonale apparaît :



Au bout de 5,5s, on obtient le remplissage suivant…



Étape 1 – Préparation

Que représente b, dans l'équation de droite `y=a.x+b` ?

Donner le type et le nom de la (des) variable(s) permettant :

- de compter le nombre d'itérations (nombre de passages dans la(les) boucle(s)),
- de repérer la(les) coordonnées du pixel qui doit s'allumer.

Étape 2 – Codage du programme

À partir du terminal, lancer la commande :

```
getLab.sh  tp3-ex5
```

Editer le code source du fichier **tp3-ex5.cpp** pour répondre au besoin.

Étape 3 – Test du programme

Vérifier le fonctionnement demandé.

Fin de séance

Éteindre le système cible

À partir du terminal, taper la commande suivante : `sudo  poweroff`