Niveau 4

TP
S4. Développement logiciel
S4.1. Principes de base
Organisation des programmes : procédures, fonctions, paramètres, valeur de retour

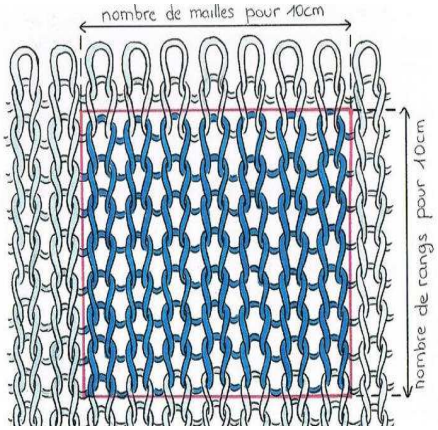
Ainsi fon(ction), fon(ction), fon(ction), les petites marionettes

TABLE DES MATIÈRES

1 - PROCÉDURES ET FONCTIONS

1) Madame Ozaur doit calculer les distances entre plusieurs points de son repère cartésien. Elle trouve la méthode très rébarbative et souhaiterait que vous lui fassiez un programme pour l'optimiser. Amandine voudrait pouvoir saisir les coordonnées des deux points concernés, `(X1,Y1)` et `(X2,Y2)` et que le programme lui calcule cette satanée distance ! Seul impératif : utiliser une procédure ou une fonction `CalculerDistance()`, à laquelle sont passées les coordonnées saisies dans le programme principal, de façon à avoir un module logiciel réutilisable.

2) En bonne tricoteuse, Isabelle adapte toujours ses modèles à sa taille ! Pour cela, elle réalise un échantillon de 10 cm x 10 cm puis compte le nombre de mailles et de rangs obtenus. Elle peut ensuite faire une petite règle de trois pour connaître le nombre de rangs à faire selon la longueur (en cm) désirée, ou encore le nombre de mailles selon la largeur (en cm) requise.
Pour cela, madame Parleurnom a besoin d'un programme qui, en fonction des nombres de mailles et de rangs de l'échantillon qu'elle aura saisis au démarrage, lui demande en boucle une longueur ou une largeur et serait capable de lui fournir les nombres de mailles et de rangs à réaliser.
Vous avez eu l'idée de génie de réaliser deux procédures ou fonctions, qui calculent, pour l'une -`CalculerRangs()`- le nombre de rangs à tricoter, pour l'autre -`CalculerMailles()`- le nombre de mailles à monter.
`CalculerRangs()` reçoit en paramètres le nombre de rangs de l'échantillon ainsi que la longueur désirée... Et quelque chose de similaire pour `CalculerMailles()` !



Par exemple, l'échantillon présenté ci-contre indique 7 mailles pour 10 cm (largeur) et 7 rangs pour 10 cm (longueur).

Isabelle veut faire une couverture de 85 cm de large et de 130 cm de haut.

Elle doit donc monter :

7×85/10 = 59,5 mailles (arrondies à 60 !) pour la largeur

et tricoter :

7×130/10 = 91 rangs pour la longueur.

3) Monsieur Kiroul, ingénieur du son, doit faire une centaine de mesures sonores en divers emplacements d'une salle de spectacle, visualiser l'ensemble des données, puis en rechercher la valeur la plus petite, la valeur la plus grande et faire une moyenne de toutes les valeurs.
Pierre requiert votre aide pour l'aider dans son travail, ou du moins, pour lui proposer un programme qui serait facile à utiliser. Votre exceptionnelle maîtrise algorithmique vous pousse à découper ce programme complexe en procédures ou fonctions :

- une pour initialiser la totalité du tableau à la valeur 0. Le prototype doit être le suivant :
```
void InitialiserTableau1D    (    float fTab1D[],
                                  unsigned int nNbreDeCases,
                                  float fValeurInitiale = 0.0 ) ;
```
- une pour réaliser l'acquisition d'une seule valeur sonore (ici, simulée et générée aléatoirement entre 20 et 150). Le prototype doit être le suivant :
```
bool AcquerirUneMesureSonore (    unsigned int nNumCase,
                                  float fTab1D[],
                                  unsigned int nNbreDeCases    ) ;
```
- une pour afficher l'ensemble des acquisitions. Le prototype doit être le suivant :
```
void AfficherTableau1D  (    const float fTab1D[],
                             unsigned int nNbreDeCases    ) ;
```

- une pour rechercher l'emplacement de la plus petite valeur. Le prototype doit être le suivant :
```
int  RechercherEmplacementPlusPetit1D (    const float fTab1D[],
                                           unsigned int nNbreDeCases    ) ;
```
- une pour celui de la plus grande. Le prototype doit être le suivant :
```
int  RechercherEmplacementPlusGrand1D (    const float fTab1D[],
                                           unsigned int nNbreDeCases    ) ;
```
- une dernière pour calculer le niveau sonore moyen. Le prototype doit être le suivant :
```
float    CalculerMoyenne1D  (    const float fTab1D[],
                                 unsigned int nNbreDeCases    ) ;
```

N'oubliez pas que Pierre doit se déplacer entre deux mesures et qu'il doit lui-même déclencher l'acquisition.

4) Monsieur Titouplin est marionnettiste indépendant. Pour son spectacle, il utilise un projecteur à LED du type PAR 56 HQ Power VDPLP56SB2, DMX 4 canaux et pilotable par ordinateur.
Jean souhaiterait de votre part un programme lui permettant de programmer l'intégralité des changements de couleur et d'intensité pour son spectacle, qui dure 30 minutes. Chaque scénette dure 5 minutes et a sa propre couleur et intensité.
Vous lui réalisez donc un programme, structuré en modules logiciels (procédures ou fonctions), un pour chaque point technique.

Chaque procédure et fonction devra être testée indépendamment avant de passer à la suivante. Pour cela, utilisez le débogueur si besoin.

Dans les prototypes proposés, adaptez les **???** selon vos choix de passages de paramètres.

Vous commencerez par préparer un couple de fichiers `.h` et `.cpp` pour gérer une structure de données. Dans ces fichiers, proposez des modules qui :

- utilisent un nouveau type structure de données (nommé `TDMX`) mémorisant la couleur, exprimée en (R,V,B), et l'intensité. Ces données sont toutes de type octet.
- initialise chaque champ de la donnée structurée passée en paramètre à la valeur 0. Le prototype doit être le suivant :
```
void      InitialiserDMX     (    TDMX ??? stDMX     ) ;
```
- permet de remplir chaque champs de la donnée structurée passée en paramètre par les valeurs saisies par l'utilisateur (ou tirées au sort). Le prototype doit être le suivant :
```
void      SaisirDMX      (    TDMX ??? stDMX     ) ;
```
- affiche chaque champ de la variable structurée passée en paramètre. Le prototype doit être le suivant :
```
void      AfficherDMX    (    TDMX ??? stDMX     ) ;
```

En saisissant R = 255, V = 127, B = 255 et 97 pour l'intensité, l'affichage devrait donner quelque chose comme :

```
Couleur : (255, 127, 255)       Intensite : 97
```

Vous continuerez en préparant un autre couple de fichiers .h et .cpp pour gérer un tableau à une dimension de structure de données. Dans ces fichiers, proposez des modules qui :

- utilisent un tableau dont chaque case serait une donnée de type `TDMX`, structuré comme précédemment.
- initialise, dans un module logiciel, tous les champs de toutes les cases du tableau à 0 Le prototype doit être le suivant :

```
void InitialiserTableauDMX   (    TDMX nTab1D[],
                                  unsigned int ??? nNbreDeCases    ) ;
```

- enregistre dans le tableau, et au bon endroit en fonction de son numéro, une scénette dont les valeurs sont acquises par un module déjà réalisé. Le prototype doit être le suivant :

```
bool EnregistrerUneScenette  (    TDMX ??? UneScenette,
                                  unsigned int ??? nNumeroScenette,
                                  TDMX nTab1D[],
                                  unsigned int ??? nNbreDeCases    ) ;
```

- (Dans le main()...) demande à Jean, dans une boucle, de saisir toutes les scénettes, en appelant le module déjà réalisé, et les enregistre dans le tableau (en appelant le module adéquat).
- affiche la couleur et l'intensité programmées de façon temporelle (ici, vous pouvez considérer des secondes plutôt que des minutes) pour simuler la commande du projecteur durant le spectacle. Le prototype doit être le suivant :

```
void AfficherScenettes  (    const TDMX nTab1D[],
                             unsigned int ??? nNbreDeCases,
                             unsigned int ??? nDureeScenette) ;
```

En affichant une ligne toutes les nDureeScenette, l'affichage final devrait donner quelque chose comme :

```
Couleur : (1, 1, 1)     Intensite : 1

Couleur : (2, 2, 2)     Intensite : 2

Couleur : (3, 3, 3)     Intensite : 3

Couleur : (4, 4, 4)     Intensite : 4

Couleur : (5, 5, 5)     Intensite : 5

Couleur : (6, 6, 6)     Intensite : 6

Ca vous a plu ???
Appuyez sur <ENTRÉE> pour fermer cette fenêtre...
```