LIFAP1 – Séquence 3

TP Noté #2 - durée 1h30 mn

Jeudi 16 décembre 2021

Sujet A

Consignes

Aucun accès au WEB, aux pages de l'UE, ni à vos anciens TP n'est autorisé. Dans votre fichier, vous mettrez en commentaire vos nom et prénom ainsi que votre numéro d'étudiant.
La note tiendra compte du respect des consignes, de la qualité de la présentation et de la lisibilité du code, des algorithmes, et du bon fonctionnement du programme. **Seules les notions vues en cours devront être utilisées**.
Une fois le programme terminé et testé (ou à la fin du temps imparti), vous devrez déposer le fichier source (.cpp) via **TOMUSS** (en cliquant sur "déposer" dans la case Depot_TP_NOTE de l'UE LIFAP1). Aucun retour par mail ne sera accepté.

Travail à réaliser

Pour réaliser ce TP, vous pourrez utiliser toutes les fonctionnalités de la bibliothèque `string.h`.
Nous allons développer une application de gestion de musique.

Une `note` de musique est représentée par son `nom` ("do", "re", "mi", "fa", "sol", "la", "si") et le nombre de `temps` qu'elle dure (croche = 0.5 temps, noire = 1 temps, blanche = 2 temps, ronde = 4 temps). Une `partition` est représentée par un tableau de notes appelé `tab_notes` et le nombre de `notes` `nb_notes` qu'il contient.

1- Définir en langage C/C++ deux constantes nommées `MAXCH` et `MAXNOTES` ayant pour valeurs respectives 4 et 100.
2- Définir les structures `note` et `partition`.
3- Ecrire une **fonction** `saisir_note` qui remplit une structure `note` avec des valeurs choisies par l'utilisateur. Attention, la saisie devra être recommencée tant que les notes proposées ne sont pas parmi `do,re,mi,fa,sol,la,si` et le temps devra être égal à une valeur parmi `0.5` (croche), `1` (noire), `2` (blanche) ou `4` (ronde).
4- Ecrire une procédure `afficher_note` permettant d'afficher les valeurs contenues dans une `note`.
5- Ecrire une **procédure** `creer_melodie` qui à partir d'un nombre de notes `nb` passé en paramètre, remplit et "retourne" une structure `partition` remplie avec autant de `notes` que souhaité. On utilisera la fonction écrite en 3-.
6- Ecrire une procédure `afficher_partition` permettant d'afficher les valeurs contenues dans une `partition`. On fera appel au sous-programme `afficher_note`.
7- Ecrire un sous-programme `decrypte_partition` permettant de construire une chaine de caractères contenant chaque `note` de la partition et pour chaque `note` sa durée. On pourrait obtenir par exemple la chaine suivante : `do(croche) / fa(noire) / sol(ronde) / mi(blanche) /`
8- Ecrire une **procédure** `duree_melodie` qui calcule et "retourne" la durée totale d'une mélodie passée en paramètres.
9- Ecrire le programme principal de l'application permettant
    a. de demander à l'utilisateur le nombre de notes `nombre` qui constituent la `partition` complète (attention on devra recommencer la saisie tant que la valeur proposée n'est pas comprise entre 3 et `MAXNOTES`),
    b. de créer une mélodie en saisissant `nombre notes`,
    c. d'afficher la liste de `notes` de la `partition` en utilisant `afficher_patition`,
    d. puis en utilisant `lire_partition` de créer puis d'afficher la chaine de caractères contenant les `notes` et leur type,
    e. et enfin d'afficher la durée totale de la mélodie jouée.