Laboratoire 3
Les structures

Étape 1 : Calculer l'âge d'une personne

Vous allez créer un petit programme qui calcule l'âge d'une personne à partir de sa date de naissance. Le programme affiche la date du jour et ensuite, demande à l'utilisateur d'entrer sa date de naissance et finalement affiche sa date de naissance en texte avec son âge.

Voici un exemple d'exécution

```
Bonjour, aujourd'hui nous sommes le 03/02/2020

Entrer votre date de naissance (jj mm aaaa) : 30 03 1995

Vous etes ne le 30 mars 1995 et vous avez maintenant 24 ans
```

1. Vous devez créer une **structure date** et déclarer des variables de type date pour manipuler la date de naissance entrée par l'utilisateur et la date du jour.

```
struct date
{
      int jour;
      int mois;
      int annee;
};
```

2. Vous devez faire une fonction qui génère la date du jour pour pouvoir calculer l'âge. Dans cette fonction vous déclarer une variable locale de type date et la retourne au main où se fera l'affichage de la première ligne à la console. Pour que cette fonction soit réutilisable, on ne fait pas l'affichage à l'intérieur, mais seulement la génération de la date. Pour afficher la date avec des 0 avant les nombres, vous devez utiliser setw et setfill.

   Voici les instructions pour manipuler les dates en C++. Bien lire les commentaires pour comprendre comment utiliser ces nouvelles variables et fonctions de la librairie <time.h>

```
struct tm newtime;          // Structure tm qui contiendra la date du jour
time_t now = time(NULL);    // now contient le nombre de secondes depuis le 01-01-1970.

localtime_s(&newtime, &now); // fait la conversion de now en structure tm

date dateJour;
dateJour.mois = newtime.tm_mon;          // mois (0- 11)
dateJour.jour = newtime.tm_mday;         // jours (1-31)
dateJour.annee = newtime.tm_year;        // Nombre d'années depuis 1900

return dateJour;
```

3. Ensuite, faîtes une fonction qui saisit une date et qui la valide adéquatement. Cette fonction retourne la date saisie et affichage se fera dans le main pour qu'elle soit réutilisable.

Exemple

```
Bonjour, aujourd'hui nous sommes le 03/02/2020

Entrer votre date de naissance (jj mm aaaa) : a 03 2020
La date doit avoir des valeurs numériques

Entrer votre date de naissance (jj mm aaaa) : 42 03 2020
Le jour doit être compris entre 1 et 31

Entrer votre date de naissance (jj mm aaaa) : 02 33 2020
Le mois doit être compris entre 1 et 12

Entrer votre date de naissance (jj mm aaaa) : 02 03 0000
L'annee doit être comprise entre 1900 et 2020

Entrer votre date de naissance (jj mm aaaa) : 30 03 2021
L'annee doit être comprise entre 1900 et 2020
```

Vous devez valider à l'intérieur d'une seule boucle do while

- le type
- le jour (entre 1 et 31)
- le mois (entre 1 et 12)
- l'année (entre 1900 et l'année actuelle). Appelez la fonction qui génère la date du jour localement pour avoir l'année en cours, comme ça, ce sera évolutif. Vive les fonctions réutilisables :o)

Valider le type

Si on veut saisir un entier et qu'une lettre est entrée, ça génère une boucle infinie. Tester ceci:

```
int number;   //nombre positif saisie au clavier
do
{
	cout << endl << "Entrer un entier: ";
	cin >> number
}while(number < 0);
```

Pourquoi? Dès que la saisie n'est pas conforme à ce qui est demandé, l'entrée est placée dans le buffer pour une prochaine saisie et le flux cin devient false. Dès que le buffer contient quelque chose, la prochaine saisie tentera de le récupérer avant de prendre les nouvelles entrées au clavier.

Nous avons 2 mécanismes à contrôler.

- Le buffer doit être vidé
- Le flux doit être remis à true

Comment faire une boucle tant que ce n'est pas un entier?

```
int number;   //nombre positif saisie au clavier
do
{
       cin.clear();         //permet de remettre le flux à son état initial
       cout << endl << "Entrer un entier: ";
       cin >> number
}while(cin.fail() && number < 0); //on tourne tant que le flux fail, donc pas un int
```

Mais la boucle est toujours infinie. Pourquoi?

Le flux fail à chaque tour de boucle, car le buffer contient toujours une mauvaise entrée qui est lue à chaque itération, nous devons donc vider le buffer avant la saisie.. La méthode cin.ignore() permet de flusher le contenu du buffer. Par contre, si le buffer est vide, le ignore attend qu'il se remplisse pour le vider, ça génère donc une pause à la console, on va entrer ENTER, il ira dans le buffer et le ignore le videra et le programme continuera. Comme on ne veut pas faire de pause inutile, on doit faire une condition pour tester si le buffer est vide ou non.

```
//vide le buffer et remet le flux valide
void viderBuffer()
{
    cin.clear();         //on reset le flux pour que la suite parte d'un flux valide
    cin.seekg(0, ios::end);//se place à la fin, si ça marche, non vide

    if(!cin.fail())      //Le flux est valide, donc le buffer est non vide
        cin.ignore(numeric_limits<streamsize>::max());
    else                 //Le flux est invalide, donc le buffer est vide
        cin.clear();
        // Le flux est dans un état invalide donc on le remet en état valide
}
```

Voici donc une boucle qui fonctionne et qui permet de bien valider le type entier et que c'est supérieur à 0

```
int number;   //nombre positif saisie au clavier
do
{
       cout << endl << "Entrer un entier: ";
       viderBuffer();
       cin >> number;
       if (!cin)                  //si le stream devient faux ou if(cin.fail())
              cout << endl << "Vous devez entrer un entier.";
       else if (number < 0)              //si le nombre est inférieur à 0
              cout << endl << "Le nombre saisi doit être supérieur ou égal à 0";
}while(!cin || number < 0);
cout << "Valeur logique";
```

On peut maintenant se coder se coder une fonction **pause** plus efficace que **system("pause");**

```
//vide le buffer et fait un ignore d'un buffer vide, donc fait une pause
void pause()
{
    viderBuffer();
    cin.ignore(numeric_limits<streamsize>::max());
}
```

4. Finalement, faites une fonction pour calculer l'âge de la personne à partir de sa date de naissance saisie précédemment. Pour calculer l'âge, vous devez faire la soustraction de l'année actuelle et de l'année de naissance de l'utilisateur. Attention, si une personne est née au mois d'avril et que sa fête n'est pas encore arrivée, elle n'a pas encore 1 an de plus. Vous devez donc vérifier le mois et le jour pour bien calculer.

   Attention si une personne est née cette année, mais avant aujourd'hui, on doit voir 0 an, mais si elle est née après aujourd'hui, on ne veut pas voir -1 an, mais «vous n'êtes pas encore né !»

```
Bonjour, aujourd'hui nous sommes le 03/02/2020

Entrer votre date de naissance (jj mm aaaa) : 03 01 2020

Vous etes ne le 30 janvier 2020 et vous avez maintenant 0 ans

Entrer votre date de naissance (jj mm aaaa) : 30 03 2020

Vous n'etes pas encore ne
```

   Pour afficher le mois en texte, vous aurez besoin d'un tableau de string avec tous les mois.

Montrez-moi cette étape dès qu'elle est terminée.

Étape 2 : Ajouter une struct au programme de l'étape 1

Dans le programme du laboratoire 2 qui calcule la note finale des étudiants d'un groupe, plusieurs paramètres sont nécessaires pour effectuer des manipulations sur les étudiants. Ce type d'organisation en mémoire est peu efficace et exige de manipuler beaucoup de variables. Pour mieux regrouper le tout et en faciliter la manipulation, nous allons faire une structure etudiant.

1. Créez un nouveau projet et copiez-y le code de l'étape 2 du lab. 2 pour faire les modifications suivantes.

2. Copiez la structure ci-dessous en haut de votre programme pour qu'elle soit globale**.**

```
struct etudiant
{
      string nom;
      string prenom;
      float noteFinale;
};
```

3. Ensuite, modifier le programme en éliminant les tableaux : nom, prénom et note finale pour ne créer qu'un seul tableau de type etudiant.

```
etudiant groupe[32];      //information du groupe d'étudiants
```

Pour manipuler ce tableau, voici les instructions de lecture et de calcul de la note final.

Par exemple :

```
fichierNote >> groupe[nbEtu].nom >> groupe[nbEtu].prenom;

while(!fichierNote.eof())    //tant que ce n'est pas la fin du fichier
{
      groupe[nbEtu].noteFinale = calculerNoteFinal(fichierNote);
      nbEtu++;
}

afficherEtudiant(groupe, nbEtu);
moyenne = calculerMoyenne(groupe, nbEtu);
afficherMoyenne(moyenne)
```

Pour bien réaliser ces modifications et ne pas avoir trop d'erreurs de compilation, commentez votre code par partie et décommenter par étape pour tester les modifications que vous faites dans l'ordre ci-dessous:

- lire les informations
- calculer la note finale
- afficher les informations
- trier les étudiants en ordre alphabétique
- calculer la moyenne

Montrez-moi ces 2 programmes terminés avant le cours où le lab.4 sera présenté. (semaine 4)