TP_s6.2: `struct`

Lien avec le MOOC Initiation à la Programmation (en C++)

Exercices semaine6 du MOOC : 2nde partie `struct`

- **Document** Tutoriel « définition et utilisation d'une structure Personne »

- **Document** Exercices semaine 6.2 du MOOC (partie struct)

  - **Exercice 19 : nombres complexes (niveau 1)**
  - passage de structure en paramètre à une fonction

  - **Exercice 20 : QCM (niveau 2)**
  - structure et vector

- **Document** Exercices additionnels semaine 6.2 du MOOC (partie struct)

  - **Exercice 17 : fractions (niveau 2)** ▪ Illustration du principe de séparation des fonctionnalités : une seule fonction est responsable de la simplification d'une fraction ; toutes les autres fonctions l'utilisent pour simplifier le résultat de l'opération dont elles sont responsables.

Exercice Complémentaire (ExC)

ExC 4 : Using C++ min() and max() functions (in English)

1) C++ offers two standard functions that achieve the comparison of two values of any type:

• The function **min**(a,b) returns the minimum of the two parameters passed by const reference

• The function **max**(a,b) returns the maximum of the two parameters passed by const reference

For example, the following code :

```
#include <iostream>
using namespace std;
int main ()
{
    cout << "min(1,2)==" << min(1,2) << '\n';
    cout << "min('z','a')==" << min('z','a') << '\n';
    cout << "max(3.14,2.72)==" << max(3.14,2.72) << '\n';
    return 0;
}
```

Print this output:

```
min(1,2)==1
min('z','a')==a
max(3.14,2.72)==3.14
```

ExC4 Exercise : read 4 values (either integer, floating point or character) and use the functions to get the min and the max of those 4 values.

ExC5: Opérateurs bit à bit (niveau 1)

L'intérêt des opérateurs bit à bit et un exemple d'extraction d'un groupe de 5 bits sont détaillés séparément.

a) Ecrire une fonction **f1** recevant en paramètre un entier non-signé **n** et un second entier non-signé **index** compris entre 0 et 31 et qui renvoie la valeur du bit de rang index dans n.

b) Ecrire une fonction **f2** recevant en paramètre :
- un entier non-signé **n**
- un entier non-signé **index** compris entre 0 et 31
- une **val** entière 0 ou 1

et qui renvoie la valeur modifiée de **n** obtenue en affectant le bit de rang **index** à la valeur **val**.

c) Ecrire une fonction **f3** recevant en paramètre :
- un entier non-signé **n**
- un entier non-signé **witdh** compris entre 1 et 32
- un entier non-signé **shift** compris entre 0 et 32-width
- une **val** entière non-signée comprise entre **0** et $\mathbf{2^{width} - 1}$

et qui renvoie la valeur modifiée **n** obtenue en insérant **val** sur les bits situés entre les rangs **shift** et **shift+width-1**