

TP Enumération, union et structure

Module : Programmation 2 : C++, POO

1. Sommaire

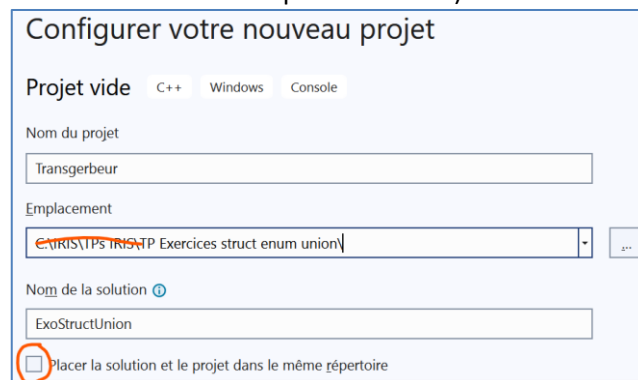
2. Objectifs.....	1
3. Préparation pour le développement des programmes.....	1
4. Transgerbeur	2
4.1. Expression du besoin.....	2
4.2. Analyse	3
4.3. Réalisation	3
5. Codage de couleur RGB.....	4
5.1. Expression du besoin.....	5
5.1. Réalisation	5
6. Gestion des employés	5
6.1. Expression du besoin.....	5
6.2. Réalisation	6
7. Notes personnelles.....	6

2. Objectifs

- Mettre en pratique les énumérations, les unions et les structures

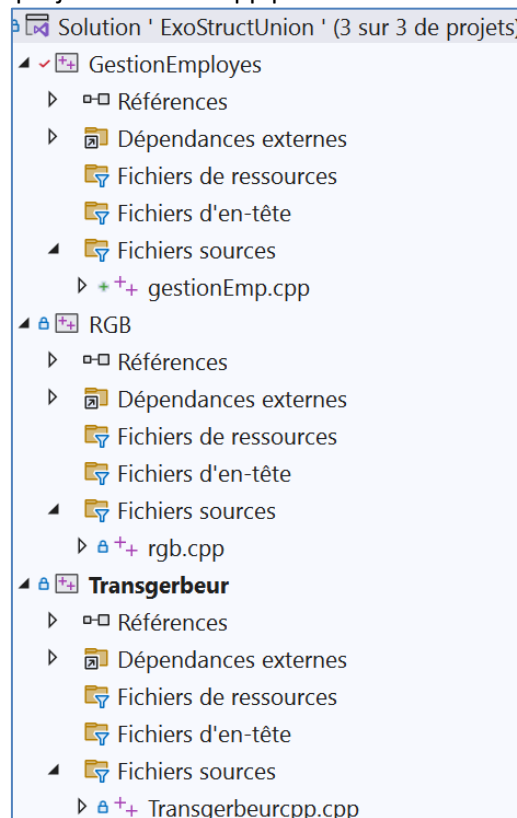
3. Préparation pour le développement des programmes.

- ⇒ Créer dans VS Studio une solution « **ExoStructUnion** » avec un premier projet **Transgerbeur** : (Décocher la case et donner le chemin sur D dans le répertoire du TP)



- ⇒ Dans cette solution ajouter deux autres projets :
 - **GestionEmployes**
 - **RGB**

⇒ Ajouter pour chaque projet un fichier .cpp pour obtenir dans l'explorateur de solution :



 **Créer un dépôt Git local et GitHub sur votre compte (xxx-CIEL) en dépôt public.**

⇒ Ajouter dans chaque fichier .cpp précédant une fonction principale qui affiche le nom du programme sur la console avec un « cout ».

 **Commit #1**



Autoévaluation

4. Transgerbeur

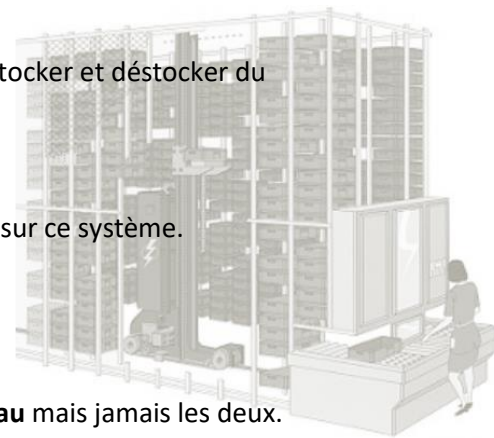
Le transgerbeur ou transtocker est un magasin automatisé qui permet de stocker et déstocker du matériel sur ordre d'un opérateur.

4.1.Expression du besoin

On désire obtenir une représentation structurée des informations à traiter sur ce système.

Les données à traiter sont les suivantes :

- Un transgerbeur comprend un ensemble de 9 **casiers**.
- Un casier est repéré par deux coordonnées **x** et **y** (nombre entier).
- Un casier a un **contenu** qui peut être soit une **palette** soit un **rouleau** mais jamais les deux.
- Un casier a un **état** de contenu (vide, palette ou rouleau).
- Une palette a un **poids** (nombre entier) et une **référence** (chaîne de 19 caractères max).
- Un rouleau a une **longueur** (nombre réel) et **numéro** de série (nombre entier)



4.2. Analyse

- ⇒ Trouver le type de donnée la mieux adapté pour chaque entité et un nom respectant la charte de nommage vu en cours et cela pour le nom de la variable et/ou pour le nom du type (structure ou énumération ou l'union).

Pour un tableau, il faut aussi préciser le type de donnée des éléments.

Données	Nom variable	Nom type	tableau	int	float	char*	union	enum	structure
Cordonnée x	px			X					
Cordonnée y	py			X					
Poids	poids			X					
Référence	reference					X			
Longueur	longueur				X				
Numéro	numero			X					
Etat du contenu	etat	EEtat						X	
Contenu du casier	contenu	UContenu					X		
Palette	paletteCasier	SPalette							X
Rouleau	rouleauCasier	SRouleau							X
Casier	casier	SCasier							X
Ensemble des casiers	tabCasier	SCasier[]	X						X



Autoévaluation

4.3. Réalisation

- ⇒ Déclarer les variables, tableaux et définir les structures, énumérations et unions des éléments précédents.



Commit #2

- ⇒ Réaliser un programme de test et vérifier pour chaque étape le contenu des variables avec le débogueur :
- Déclarer et initialiser un rouleau **ro1**
 - Déclarer une palette **pa1** et ensuite donner des valeurs aux différents membres
Utiliser **strcpy_s** pour les chaînes
 - Déclarer et initialiser un état **etatC1** à la valeur vide
 - Déclarer et initialiser un contenu **cont** avec **pa1**, puis lui affecter **ro1**

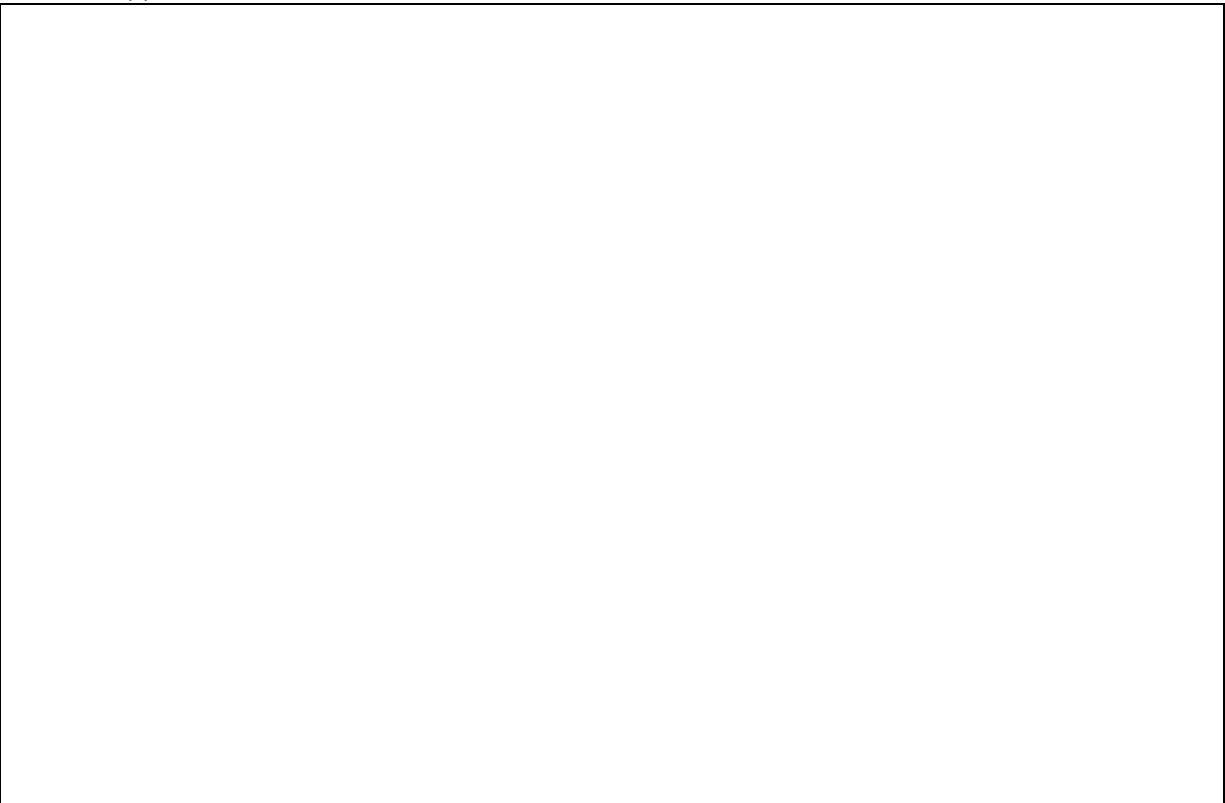
- Déclarer un casier **c1**, lui affecter le contenu **pa1**, l'état qui convient et les coordonnées 2,3
- Déclarer et initialiser un casier **c2** avec une palette
- Déclarer un tableau de casiers contenant le casier **c1** à l'indice 0 et **c2** à l'indice 2

Commit #3

- ⇒ Utiliser « **sizeof** » pour afficher sur la console la taille de chaque élément créé précédemment.

Commit #4

- ⇒ Faire une capture d'écran avec les tailles de tous les types utilisés ainsi que le tableau de casier. Mettre en évidence l'optimisation des ressources par l'utilisation d'une union par rapport à une structure.



Commit #5

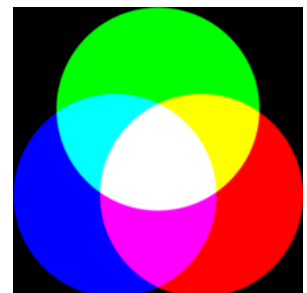


Autoévaluation

5. Codage de couleur RGB

RGB (de l'anglais « red, green, blue ») ou RVB en français, est un système de codage informatique des couleurs, le plus proche du matériel.

Le codage RVB indique une valeur pour chacune de ces couleurs primaires. Chaque valeur est codée sur un octet.



⇒ Combien de couleurs différentes (combinaison de rouge, vert et bleu) peut-on obtenir ?

5.1.Expression du besoin

On désire obtenir une représentation structurée du codage de la couleur qui pourra être utilisée avec plusieurs types :

- Une structure Srgb comportant 3 octets r, g, b
- D'un tableau de 3 octets (r,g,b)
- D'une valeur entière unique égale à $b*256^2+g*256+r$

Le programme de test permettra à un utilisateur de saisir au choix la couleur avec les 3 composantes ou la valeur unique. Le programme affichera la couleur en utilisant les 3 membres (types) de l'union

5.1.Réalisation

La structure de donnée est la suivante :

```
struct Srgb
{
    unsigned char r;
    unsigned char g;
    unsigned char b;
};

union UColor
{
    unsigned int val;
    Srgb components;
    unsigned char tabCol[3];
};
```

⇒ Faire le programme de test décrit dans l'expression du besoin. Faire en sorte que l'affichage et la saisie des nombres se fassent en hexadécimal.



Commit #6



Autoévaluation

6. Gestion des employés

6.1.Expression du besoin

On désire obtenir une représentation structurée des informations concernant les employés d'une entreprise. Les données sur l'employé sont :

- Nom
- Prénom
- Date d'embauche
- Date de prise de fonction dans le poste



Le programme à réaliser permettra de saisir les données pour un employé et affichera ces informations sur la console.

6.2.Réalisation

⇒ Définir une structure pour les dates.

Commit #7

⇒ Définir une structure pour les employés

Commit #8

⇒ Faire un programme principal de test qui déclare et initialise un employé

Commit #9

⇒ Définir une fonction pour la saisie des informations d'un employé

Commit #10

⇒ Définir une fonction pour afficher les informations d'un employé

Commit #11

⇒ Faire un programme qui utilise les deux fonctions précédentes

Commit #12



7. Notes personnelles

Cadre dans lequel vous noterez les points importants que vous devez retenir du TP et aussi tout ce que vous avez appris en informatique durant le TP.