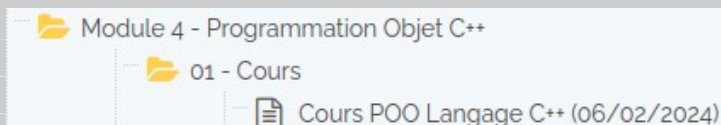


## Notion de cours abordés :

- Classes
- Attributs
- Constructeur
- Destructeur
- Méthodes de la classe.
- Instances automatique et dynamique
- Flux d'entrée/sortie standard
- Manipulateurs de flux
- Surcharge des opérateurs.



## Contexte :

La Poste propose à ses clients des emballages pré-affranchis et souhaite développer un outil de gestion de stock pour ces emballages. Chaque produit est caractérisé par plusieurs propriétés, notamment :

- Son format (par exemple : « Souple à plat », « XS en Volume », etc.)
- Sa résistance maximale
- Ses dimensions : longueur, largeur, et éventuellement hauteur
- Le nombre d'emballages disponibles en stock.

Un extrait du catalogue est présenté ci-après.

Formats	Résistance maximum de	Livraison	Dimensions (L x l x h, en mm)	Tarifs à l'unité	
				HT	TTC
Souple	1 kg	BOÎTE À LETTRES	A plat : 280 x 210	6,65 €	7,95 €
XS	1 kg		A plat : 270 x 190 En volume : 217 x 150 x 50	6,65 €	7,95 €
P	3 kg		A plat : 340 x 280 En volume : 335 x 215 x 58	8,32 €	9,95 €
M	3 kg		230 x 130 x 100	8,32 €	9,95 €
L	5 kg	CONTRE SIGNATURE	315 x 210 x 157	10,03 €	12,00 €
XL	7 kg		383 x 250 x 195	12,12 €	14,50 €
1 bouteille	2 kg		390 x 168 x 104	9,20 €	11,00 €
2 bouteilles	5 kg		390 x 297 x 106	11,29 €	13,50 €
3 bouteilles	7 kg		390 x 425 x 106	12,12 €	14,50 €

## Préparation (sur papier) :

### 1. Création d'une classe Emballage :

En vous basant sur la description précédente, proposez la déclaration de la classe Emballage. Identifiez les attributs nécessaires ainsi que leur type. Par exemple, le format de l'emballage sera représenté par une chaîne de caractères (string de la librairie standard).

### 2. Ajout d'un constructeur :

Complétez la déclaration de la classe en ajoutant un constructeur. Ce constructeur devra recevoir en paramètres les valeurs nécessaires pour initialiser les attributs. Le nombre d'emballages en stock doit être systématiquement initialisé à 0. Si l'emballage est de type « à plat », la hauteur ne sera pas fournie et sera donc initialisée par défaut à 0.

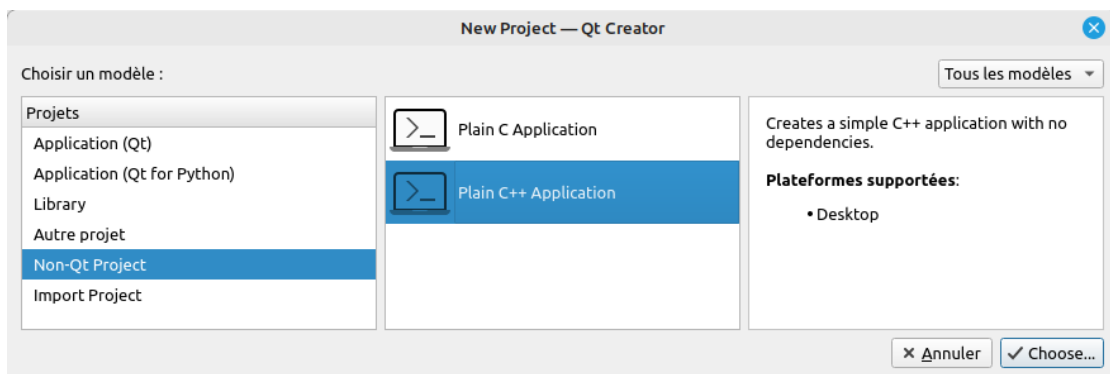
### 3. Destructeur :

Est-il nécessaire de définir un destructeur explicite dans cette classe ? Justifiez votre réponse.

## Réalisation :

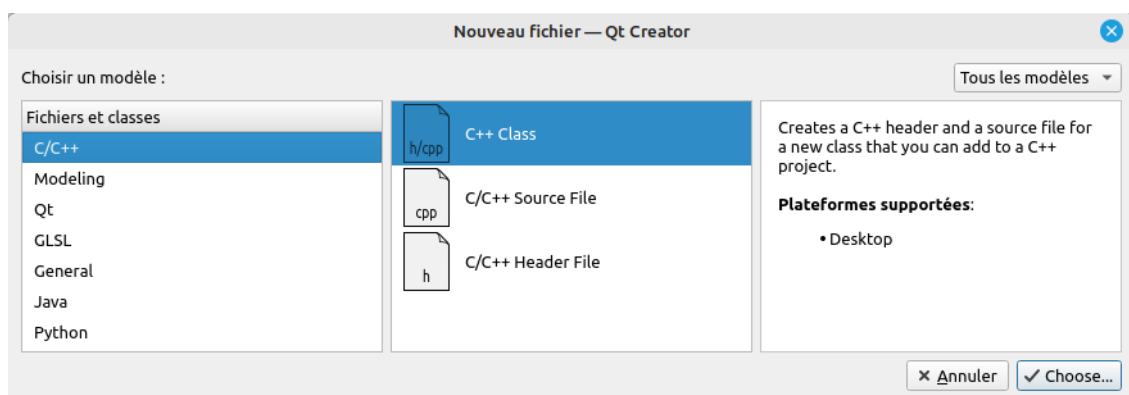
### 1. Création du projet :

Créez un nouveau projet en langage C++ (non-Qt), nommé GestionLaPoste.



### 2. Ajout de la classe Emballage :

Ajoutez la classe Emballage à ce projet.



Dans le fichier emballage.h, ajoutez les déclarations de la classe basées sur les propositions élaborées précédemment lors de la préparation.

### 3. Constructeur :

Complétez le constructeur de la classe `Emballage`. Ce constructeur devra afficher un message dans la console à l'aide de la librairie `iostream`, sous la forme suivante : "Constructeur : Emballage / <format>".

### 4. Destructeur :

Implémentez un destructeur pour la classe `Emballage`. Le destructeur doit afficher un message indiquant la destruction de l'emballage, en mentionnant ses caractéristiques pour permettre de l'identifier.

Par exemple : "Destructeur : Emballage / <format> "

### 5. Test des instances :

Dans la fonction `main()`, testez la création et la destruction de deux instances de la classe `Emballage` :

- Une instance créée automatiquement (sur la pile),
- Une instance créée dynamiquement (sur le tas, avec `new`).

Observez et notez à quel moment les destructeurs des deux instances sont appelés (pensez à libérer la mémoire allouée dynamiquement à l'aide de `delete` pour l'instance créée avec `new`).

### 6. Fonction membre `Visualiser` :

Déclarez et implémentez une fonction membre `Visualiser` dans la classe `Emballage`. Cette fonction affichera toutes les caractéristiques de l'emballage en utilisant les flux de sortie standard (`iostream`) et les manipulateurs de flux. Les informations à afficher incluent :

- Le format
- La longueur / largeur / hauteur (affichez la hauteur seulement si sa valeur n'est pas nulle)
- La résistance maximale
- Le nombre en stock

L'affichage doit ressembler à une ligne de tableau dans l'esprit d'un extrait du catalogue de la Poste, utilisant des colonnes bien alignées grâce aux manipulateurs de flux de la librairie `iostream`.

XS	1 kg	270 X 190	
XL	7 kg	383 X 250 X 195	

Testez cette nouvelle méthode en modifiant votre programme principal.

### 7. Complément pour le programme principal :

#### ● Mise en commentaire du code actuel :

Commentez le contenu actuel de la fonction `main()` pour préserver vos tests précédents tout en préparant le terrain pour cette nouvelle étape.

#### ● Tableau de pointeurs sur `Emballage` :

Déclarez un tableau de 5 pointeurs sur des objets de type `Emballage`. Ces objets devront être alloués dynamiquement dans une boucle.

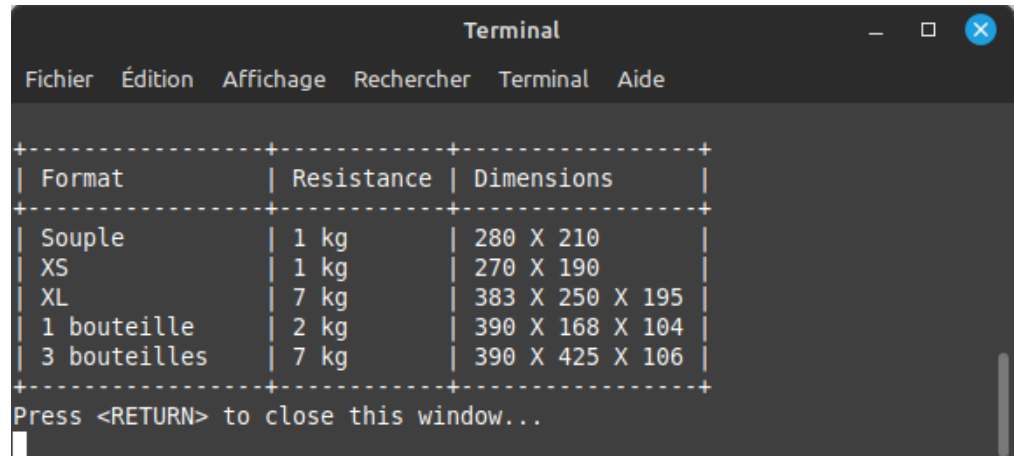
- **Saisie des caractéristiques :**

À l'intérieur de la boucle, demandez à l'utilisateur de saisir les caractéristiques (format, longueur, largeur, etc.) pour chacun des 5 emballages, et allouez dynamiquement chaque emballage à l'aide du mot-clé `new`.

- **Affichage du catalogue :**

Une fois les 5 emballages créés, parcourez à nouveau le tableau et utilisez la fonction membre `Visualiser` pour afficher les caractéristiques de chaque emballage. Cet affichage représentera un extrait du catalogue.

**Exemple :**



The screenshot shows a terminal window titled "Terminal" with a menu bar containing "Fichier", "Édition", "Affichage", "Rechercher", "Terminal", and "Aide". The terminal displays a table with three columns: "Format", "Resistance", and "Dimensions". The table is enclosed in a dashed border. The data rows are as follows:

Format	Resistance	Dimensions
Souple	1 kg	280 X 210
XS	1 kg	270 X 190
XL	7 kg	383 X 250 X 195
1 bouteille	2 kg	390 X 168 X 104
3 bouteilles	7 kg	390 X 425 X 106

Below the table, the text "Press <RETURN> to close this window..." is displayed.

## 8. Surcharge des opérateurs :

- **Comparaison du volume de deux emballages :**

Afin de comparer les volumes de deux emballages, surchargez l'opérateur `<`. Modifiez ensuite votre programme principal pour déclarer deux emballages : `colis1` de format M et `colis2` de format L, (voir extrait du catalogue), puis indiquez lequel des deux possède le plus petit volume.

**Remarque :** Le volume, exprimé en  $\text{cm}^3$ , d'un emballage est calculé comme le produit de la longueur, de la largeur, et de la hauteur (ou seulement de la longueur et de la largeur si l'emballage est à plat). **Attention, les dimensions sont exprimées en mm.**

- **Comparaison de deux emballages :**

Quel opérateur peut-on surcharger pour savoir si deux emballages sont identiques ? Un emballage est considéré comme identique à un autre s'il a le même format, les mêmes dimensions (longueur, largeur, hauteur) et la même résistance maximale. Surchargez cet opérateur et modifiez ensuite votre programme principal en introduisant une instance `colis3` possédant les mêmes caractéristiques que `colis1`, vérifiez si cet emballage est identique à `colis1` et à `colis2`.

- **Surcharge pour affecter le volume :**

Vous souhaitez écrire dans le programme principal `volume = colis1`, afin d'affecter à la variable `volume` (de type `float`) le volume du colis `colis1`. Pourquoi ne peut-on pas surcharger l'opérateur `=` pour obtenir ce résultat ? Surchargez l'opérateur de conversion approprié afin que cette affectation soit possible.