Une image contenant capture d’écran, Graphique, Police, graphisme

Description générée automatiquement

|  |
| --- |
| **CER U.E. [concevoir son approche]** |
| **[Nom de l’étudiant]** |

Animateur : ADAM

Scribe : MOURAD

Gestionnaire : LAMIN

Secrétaire : YANIS

00-00-2024 A2 – Groupe 01

# Mots clés :

# Polynôme

# Monôme

# Mots à définir :

**Monôme** : Expression algébrique composée d'un seul terme, comprenant une constante multipliée par une ou plusieurs variables élevées à des puissances entières non négatives.

**Objets immuables** : Objets dont l'état ne peut pas être modifié après leur création, garantissant ainsi leur constance et facilitant le raisonnement sur le code.

**Transparence d’usage** : Propriété d'un système ou composant permettant à l'utilisateur de l'utiliser sans connaître ses détails internes ou son implémentation, assurant ainsi une utilisation simple et intuitive.

**Collision de nom** : Problème qui survient lorsque deux identificateurs (noms de variables, fonctions, etc.) identiques existent dans le même espace de noms ou contexte, entraînant des ambiguïtés ou erreurs.

**Découpage d’objet** : Processus de décomposition d'un objet en plusieurs sous-objets ou composants plus petits, facilitant ainsi la gestion, la réutilisation et la compréhension du code.

1. **Analyse du contexte :**

* **voulant écrire des exercices sur le c++, il doit concevoir la solution a son propre problème**

# Définition de la problématique :

# Quel processus de réflexion à prendre lors de la conception d’un programme .

1. **Les contraintes :**

* **Le code donné**
* **Suivre les étapes donner**
* **laisser la possibilité à l’utilisateur de dériver le type *Monomial***

# Plan d’actions

* Plan d’action 0 :

1. **Qualification :**
2. **Démonstration :**

### ***Étudier l’architecture d’un programme :***

Analyser l'organisation et la structure globale du code source d'un programme pour comprendre comment ses différentes parties interagissent, sont modifiées, et sont maintenues. Cela inclut l'étude des modules, des dépendances, et des flux de données entre les composants pour assurer une conception efficace, modulaire et évolutive.

### ***Étudier l’accouplage des classes :***

Examiner le degré de dépendance entre les classes dans un programme. Un faible accouplement (loose coupling) est souhaitable car il signifie que les classes peuvent évoluer indépendamment les unes des autres, facilitant ainsi la maintenance et l'extensibilité du code. Un accouplement fort (tight coupling) peut rendre le code plus rigide et difficile à modifier.

### ***Étudier la notion du concept de collision de nom :***

Comprendre comment les conflits d'identificateurs se produisent lorsque deux ou plusieurs éléments (variables, fonctions, classes) portent le même nom dans le même contexte ou espace de noms, entraînant des erreurs ou des ambiguïtés. Cela inclut l'examen des techniques pour éviter ces collisions, comme l'utilisation des préfixes ou des espaces de noms.

### ***Étudier le concept des namespace :***

Explorer les espaces de noms (namespaces) qui permettent de regrouper des identificateurs sous un nom commun pour éviter les conflits de noms et organiser le code de manière logique. Les namespaces sont particulièrement utiles dans les grands projets où de nombreuses bibliothèques et composants peuvent avoir des identificateurs similaires.

### ***Comprendre l’intérêt de l’implémentation en ligne et hors ligne :***

Examiner les avantages et les inconvénients de l'implémentation des fonctions et des méthodes directement dans les fichiers d'en-tête (implémentation en ligne) versus dans des fichiers de source séparés (implémentation hors ligne). L'implémentation en ligne peut offrir des optimisations de performance grâce à l'inlining, tandis que l'implémentation hors ligne peut améliorer la séparation des préoccupations et la clarté du code.

* Plan d’action 1 : Reformuler et réinterpréter les problèmes issus de sa réflexion.

1. **Qualification :**
2. **Démonstration :**

**PB 1 : Définition des Entités pour les Polynômes**

Claude Delannoy se demande quelles entités sont nécessaires pour représenter et manipuler des polynômes en C++. Il reconnaît que chaque polynôme est composé de termes, qui sont eux-mêmes des monômes. Un monôme est un polynôme à un seul terme et peut être une constante. Cette réflexion conduit à l'idée d'utiliser une agrégation de monômes pour former un polynôme, soulignant l'importance de concevoir des classes pour les deux entités.

**PB 2 : Problèmes d'Héritage entre Polynômes et Monômes**

La question se pose de savoir si un polynôme est un monôme et vice versa. Claude conclut qu'il n'est pas pertinent de modéliser un polynôme comme une extension d'un monôme car les méthodes et les opérations applicables aux polynômes ne sont pas forcément adaptées aux monômes. Une classe supplémentaire semble nécessaire pour gérer les expressions polynomiales.

**PB 3 : Immuabilité et Gestion des Objets**

Claude veut permettre aux utilisateurs de dériver la classe Monomial pour gérer des situations plus complexes, tout en maintenant l'immuabilité des objets pour assurer robustesse et lisibilité. Le passage d'objets par valeur peut provoquer le "découpage d'objet", et la gestion manuelle de la mémoire lors de l'allocation dynamique de nouveaux objets peut conduire à des fuites de mémoire. Claude envisage de passer en paramètre l'instance à initialiser avec les nouvelles valeurs, bien que cette solution ne soit pas idéale.

**PB 4 : Organisation et Collisions de Noms**

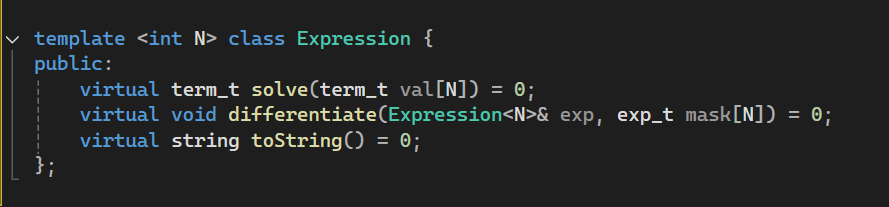
Pour éviter de complexifier le code, Claude décide de regrouper la déclaration et la définition des classes dans un même fichier. Il planifie d'utiliser des espaces de noms (namespaces) pour organiser le code et éviter les collisions de noms, en regroupant les classes ayant un lien sémantique fort, créant ainsi des modules faiblement couplés entre eux.

* Plan d’action 2 : Décortiquer le code dont un ersatz est disponible sur GIT à l’adresse du prosit (https://github.com/Supertuteur/Bloc-POO-CPI-A2/tree/main/Boucle-5/Prosit/polynomials-students)
  + 1. **Qualification :**
    2. **Démonstration :**

Pour mieux comprendre et structurer le code proposé, nous allons passer en revue les principales composantes et concepts abordés.

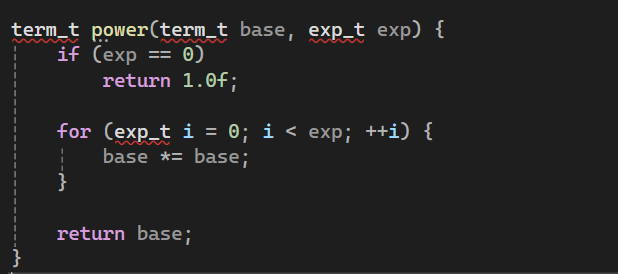
#### Templates et Classes Génériques

***Expression (classe abstraite) :***

****

Cette classe abstraite définit une expression mathématique générique avec N variables. Les méthodes sont purement virtuelles, ce qui signifie que chaque classe dérivée doit fournir des implémentations pour solve, differentiate et toString.

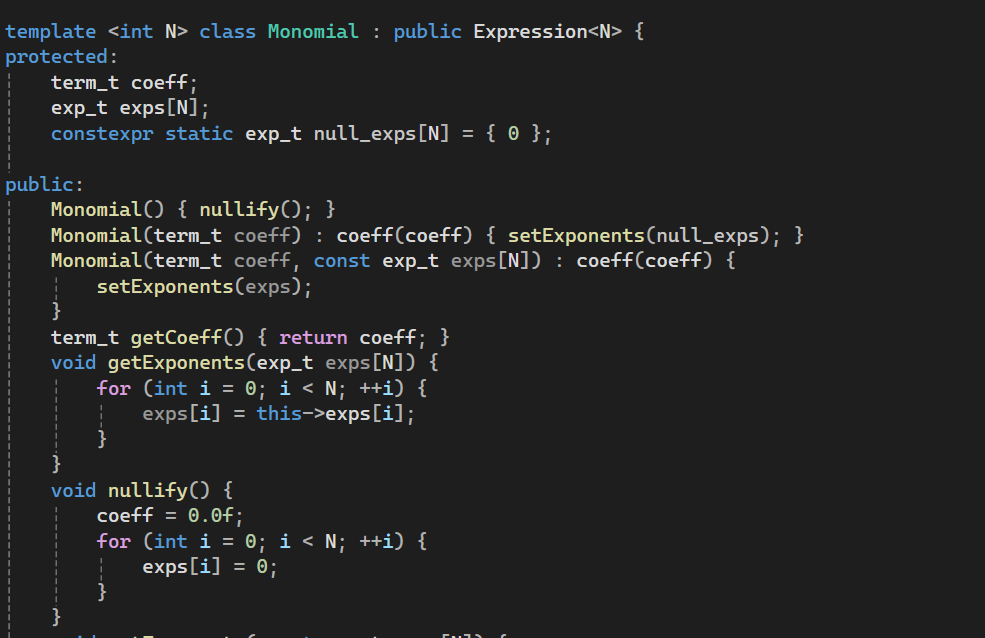
***Fonction auxiliaire power***

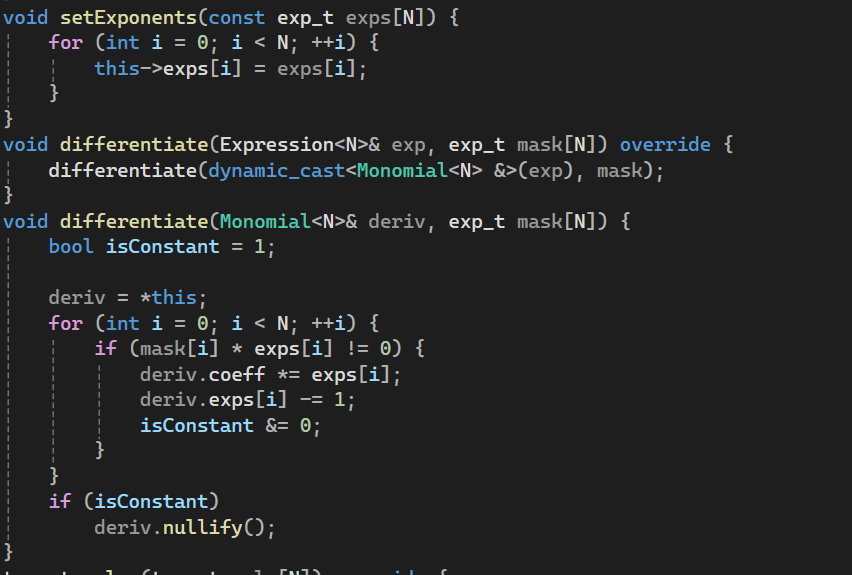


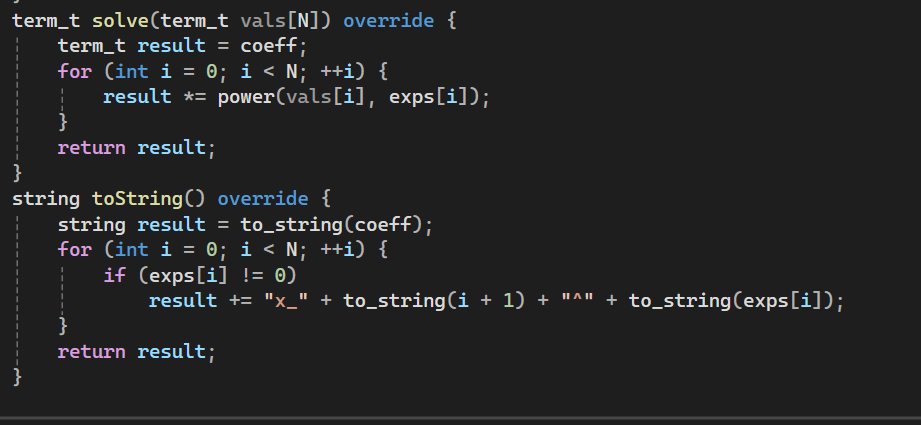
Cette fonction calcule la puissance d'un nombre. Cependant, il y a une erreur ici : la boucle devrait utiliser exp\_t comme limite pour la puissance, mais l'implémentation actuelle fait des multiplications répétées.

#### 2. Classe Monomial

***Monomial (dérivée de Expression) :***

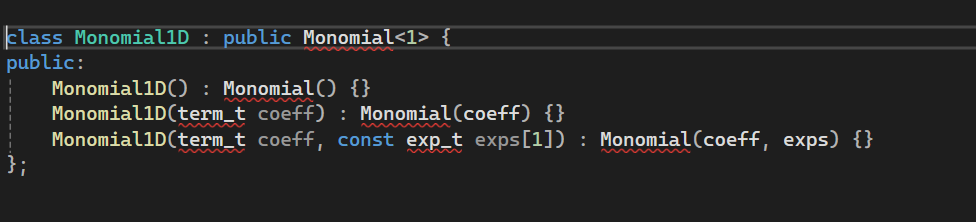






Cette classe représente un monôme avec un coefficient et des exposants. Elle implémente les méthodes pour résoudre, différencier et convertir un monôme en chaîne de caractères.

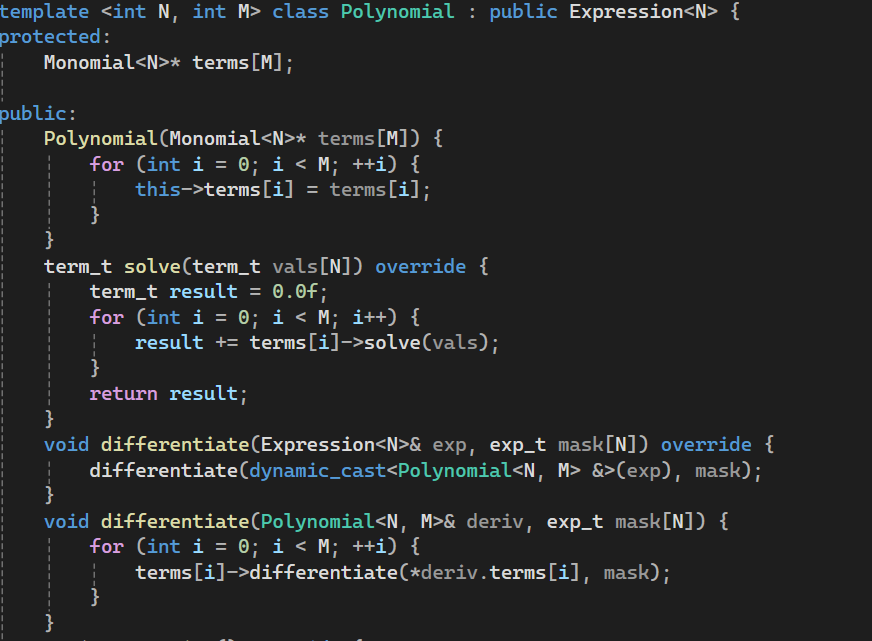
***Monomial1D (spécialisation pour N=1) :***

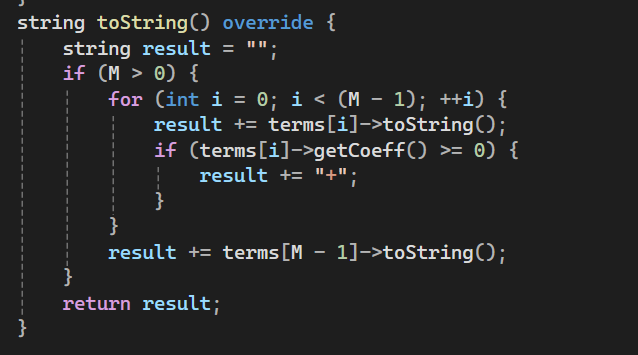
******

Cette classe est une spécialisation de Monomial pour les monômes à une seule variable.

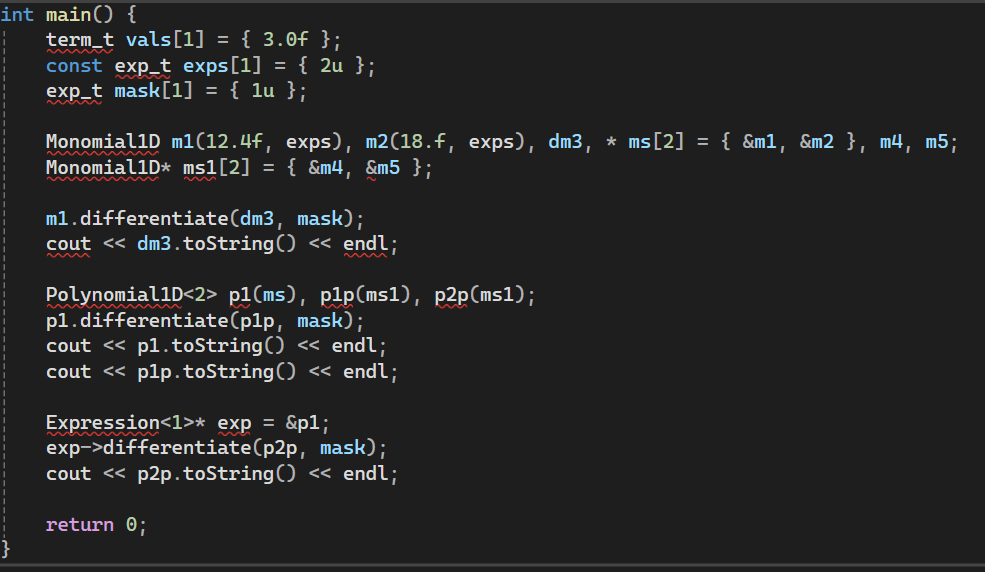
#### 3. Classe Polynomial

***Polynomial (dérivée de Expression) :***

**



Cette classe représente un polynôme comme une collection de monômes. Elle implémente les méthodes pour résoudre, différencier et convertir un polynôme en chaîne de caractères.



Cette fonction teste la différentiation de monômes et de polynômes. Elle affiche les résultats de la différentiation et de la résolution des expressions.

* Plan d’action 3 : Identifier et documenter les éléments d’implémentation liés aux problématiques soulevées (PB 1, PB 2, PB 3, PB 4). (chaque problème un par un )

1. **Qualification :**
2. **Démonstration :**

* Plan d’action 4 : modifier le code pour éviter les collisions de nom (namespace)

1. **Qualification :**
2. **Démonstration :**