Université de Caen Normandie — Année 2023-2024
Licence d'informatique de deuxième année

Algorithmique et structures de données
Devoir à faire seul ou en binôme
Sujet de la partie 1.
La partie 1 compte pour 7/20.
Elle doit être programmée en langage C.
Elle est à rendre pour le mardi 5 décembre.
La seconde partie se fera sur machine et sera elle individuelle.

L'objectif du devoir est de définir la dérivée d'une expression arithmétique sur $\mathbb{R}$. Les expressions arithmétiques seront codées avec des arbres binaires comme vous l'avez vu en TD 6.

Voici un résumé des formules de dérivées que vous devez connaître pour la partie 1.

| $f(x)$ | $f'(x)$ |
|---|---|
| constante $k$ | $0$ |
| $x$ | $1$ |
| $(g(x) + h(x))$ | $(g'(x) + h'(x))$ |
| $(g(x) - h(x))$ | $(g'(x) - h'(x))$ |
| $(g(x) * h(x))$ | $((g'(x) * h(x)) + (g(x) * h'(x)))$ |
| $-(g(x))$ | $-(g'(x))$ |
| $(g(x)^n)$ | $((n * g'(x)) * ((g(x))^{n-1}))$ |

Nous allons distinguer quatre catégories de nœuds qui seront codés dans le nœud par un attribut `categorie` de type `char` :

1. la **variable** $x$ codée par la lettre 'v'.
2. les **coefficients (et les constantes)** codés par la lettre 'c' qui seront des entiers. Dans l'expression $3x + 5$, 3 est un coefficient et 5 est une constante.
3. les **opérateurs binaires** codés par la lettre 'b'.
4. les **opérateurs unaires** codés par la lettre 'u'.

Nous appellerons expression de catégorie $i$ un pointeur sur nœud de catégorie $i$, pour $i \in \{1, 2, 3, 4\}$.

Nous utiliserons un attribut `valeur` de type `int` pour coder la valeur du nœud. Pour un coefficient (ou une constante), il s'agira de sa valeur. Pour les trois autres catégories, nous utiliserons un caractère qui sera transtypé en entier. Par exemple, (int) 'x' pour la variable $x$ et (int) '+' pour l'opérateur binaire $+$.

Les deux premières catégories correspondent à des feuilles de l'arbre binaire et les deux dernières à des nœuds internes. Un opérateur unaire est formé à partir d'une seule expression, nous mettrons le pointeur à droite égal à `NULL`.

**Question 1**. Définissez la structure `noeud` d'un arbre binaire avec un attribut `categorie` de type `char` et un attribut `valeur` de type `int`. Définissez le type `expression` qui sera un pointeur sur `noeud`. Nous allons maintenant définir des procédures constructeurs de noeuds selon ces 4 catégories :

**Question 2**. Définissez une procédure `variable` sans argument qui retourne une expression de catégorie 1.

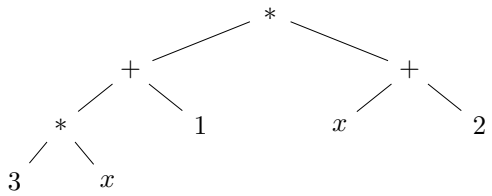


FIGURE 1 – Exemple – arbre binaire de l'expression $E1 = (3x + 1)(x + 2)$

**Question 3**. Définissez une procédure `coefficient` qui prend en argument un entier et qui retourne une expression de catégorie 2.

**Question 4**. Définissez une procédure `op_binaire` qui prend en argument un caractère $c$ et deux expressions `expression1` et `expression2` et qui renvoie une expression telle que $c$ code l'opérateur binaire, `expression1` est l'expression à gauche et `expression2` est l'expression à droite.
Cette procédure retournera une expression de catégorie 3.

**Question 5**. Définissez une procédure `op_unaire` qui prend en argument un caractère $c$ et une expression `expression1` et qui renvoie une expression telle que $c$ code l'opérateur unaire et `expression1` est l'expression à gauche.
Cette procédure retournera une expression de catégorie 4.

**Question 6**. Définissez une procédure `copie_expression` qui prend en argument une expression et qui retourne une copie de cette expression. Il n'y aura aucun nœud en commun avec l'expression en argument.

**Question 7**. Définissez une procédure `affiche_expression` qui prend en argument une expression et affiche cette expression avec des parenthèses. Par exemple, pour $E1$, on affichera $(((3 * x) + 1) * (x + 2))$.

**Question 8**. Définissez une procédure `evalue_expression` qui prend en argument une expression et une flottant et retourne l'évaluation de l'expression en remplaçant la variable $x$ par ce flottant. Pour la fonction puissance, vous utiliserez la fonction `pow` de la bibliothèque `math.h`.

**Question 9**. Définissez une procédure `derivee` qui prend en argument une expression formée avec la variable $x$,des constantes et des coefficients et avec les opérateurs binaires $+$ et $*$ et qui retourne la dérivée de cette expression. Nous appliquerons les formules données dans le tableau précédent sans effectuer de simplification.

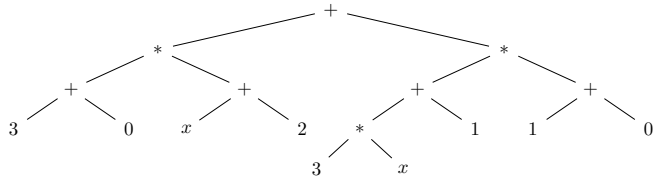


FIGURE 2 – Exemple – arbre binaire de la dérivée de $E1$

**Question 10**. Ajoutez aux procédures `derivee` et `evalue_expression` l'opérateur unaire de négation $-$, si $E1$ est une expression on forme l'expression $E = -(E1)$.

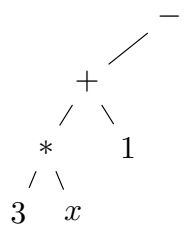


FIGURE 3 – Exemple – arbre binaire de l'expression -(3x+1)

**Question 11**. Ajoutez aux procédures `derivee` et `evalue_expression` l'opérateur binaire de la puissance codé par le caractère ^.
Par exemple pour l'expression $(1 + x)\hat{}3$ où l'expression de gauche sera $(1 + x)$ et l'expression de droite sera la constante 3.

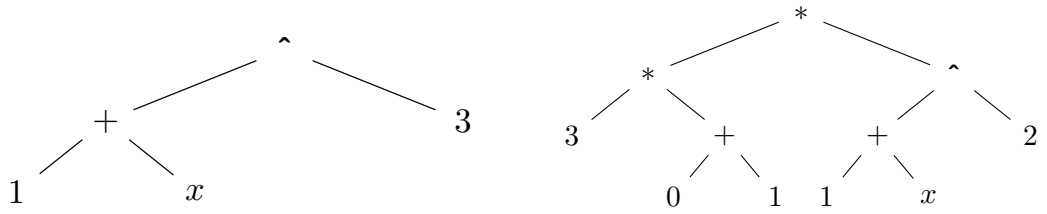


FIGURE 4 – Exemple – les arbres binaires de $(1 + x)\hat{}3$ et de sa dérivée

**Question 12**. Construisez et affichez dans le main les expressions données en exemple dans les figures ainsi que les expressions de leur dérivée.