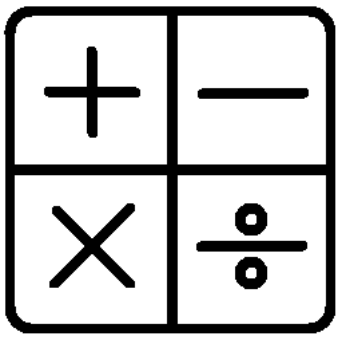


Seda

BrainUp inc.

Soutenance 2

Hassam Daya Rohan - Kaddour Matthias
Ledoux Louis - Priso-Totto Guillaume Alain

Contents

1 Introduction

Seda, ou calculatrice en coréen, est comme son nom l'indique un outil de calcul. Le projet Seda a pour objectif de créer une calculatrice capable de réaliser des opérations mathématiques de base, telles que les additions, les soustractions, les multiplications et les divisions.

Cependant, le projet Seda ne se limite pas à ces fonctionnalités de base. En effet, il est également conçu pour calculer des fonctions mathématiques complexes, telles que la fonction exponentielle, la fonction logarithmique et d'autres fonctions mathématiques.

Le projet Seda va également plus loin en permettant de réaliser des calculs matriciels. Il sera possible de travailler avec des matrices et de les manipuler en utilisant les opérations matricielles de base. Par ailleurs, Seda pourra afficher les courbes associées à des polynômes, ce qui en fait un outil très pratique pour les mathématiques.

En résumé, le projet Seda vise à créer une calculatrice complète et performante qui pourra répondre aux besoins de différents utilisateurs. Avec ses nombreuses fonctionnalités, Seda deviendra un outil essentiel pour les étudiants en mathématiques, les chercheurs et les ingénieurs qui travaillent avec des données mathématiques complexes.

2 Inspiration

Notre inspiration pour ce projet Seda provient principalement de la calculatrice de base de Windows 11. Nous souhaitons recréer cet outil avec une attention particulière portée à la qualité et à la fiabilité des calculs. Notre ambition est de reproduire la plupart des fonctionnalités de la calculatrice de Windows 11, ce qui permettra aux utilisateurs de bénéficier d'un outil pratique et efficace pour réaliser des calculs simples ou plus complexes. Cependant, nous souhaitons également améliorer cette calculatrice en y ajoutant des fonctionnalités supplémentaires, telles que la possibilité de réaliser des calculs matriciels et d'afficher les courbes associées à des polynômes. L'objectif final de Seda est de devenir une calculatrice puissante et fiable, capable de répondre aux besoins de tout type d'utilisateur.

3 Répartition des tâches

	RHD	LED	GLM	MM
expressions mathématiques			✓	
calcul dans différentes bases	✓		✓	
opérations sur les matrices		✓	✓	✓
résolution de polynômes	✓			
traçage de courbe		✓		
conversions	✓			✓
gestion mémoire	✓			
ui		✓		
ux		✓		
site web				✓
mini-jeux	✓	✓	✓	✓
compilateur python	✓			

4 Avancement des tâches

Le tableau tel que déterminé à la fin de la Soutenance 1

	Soutenance 1	Soutenance 2	Soutenance 3
expressions mathématiques	80%	100%	100%
calcul dans différentes bases	/	100%	100%
opérations sur les matrices	/	30%	100%
résolution de polynômes	/	50%	100%
traçage de courbe	/	30%	100%
conversions	/	20%	100%
gestion mémoire	80% (+20%)	100%	100%
ui	70%	80%	100%
ux	70%	80%	100%
site web	80%	100%	100%
mini-jeux	/	/	100%
compilateur python	80% (+60%)	100%	100%

Le nouveau tableau pour la Soutenance 2 La majorité des objectifs ont été atteints voir dépassés. Les opérations de matrices et traçage de courbe ont été déplacés pour la Soutenance 3 pour prioriser d'autres tâches pour la Soutenance 2

	Soutenance 1	Soutenance 2	Soutenance 3
expressions mathématiques	80%	100%	100%
calcul dans différentes bases	/	100%	100%
opérations sur les matrices	/	0% (-30%)	100%
résolution de polynômes	/	100% (+50%)	100%
traçage de courbe	/	0% (-30%)	100%
conversions	/	20%	100%
gestion mémoire	80% (+20%)	100%	100%
ui	70%	100% (+20%)	100%
ux	70%	100% (+20%)	100%
site web	80%	100%	100%
mini-jeux	/	/	100%
compilateur python	80% (+60%)	100%	100%

5 Fonctionnalités calculatoires

5.1 Shunting Yard

L'algorithme de Shunting Yard est un algorithme utilisé pour convertir une expression mathématique écrite en notation infixée (où les opérateurs sont entre les opérandes) en notation postfixée (où les opérateurs se trouvent après les opérandes). Il permet de convertir les expressions mathématiques de manière à les rendre plus faciles à évaluer par une machine. L'algorithme de Shunting Yard utilise deux piles : une pile d'opérandes et une pile d'opérateurs. Il parcourt l'expression infixée caractère par caractère, et agit en fonction du type de caractère rencontrée : Si le caractère est un opérande (comme un nombre ou une variable), il est ajouté à la pile d'opérandes. Si le caractère est un opérateur, il est ajouté à la pile d'opérateurs, à condition que les opérateurs déjà présents dans la pile aient une priorité inférieure ou égale à celle de l'opérateur en cours. Si un opérateur de priorité supérieure est présent, il est dépilé et ajouté à la pile d'opérandes avant d'ajouter l'opérateur en cours à la pile d'opérateurs. Si le caractère est un parenthese ouvrante, il est ajouté à la pile d'opérateurs. Si le caractère est un parenthese fermante, les opérateurs sont dépilés de la pile d'opérateurs et ajoutés à la pile d'opérandes jusqu'à ce que la parenthese ouvrante correspondante soit rencontrée. Une fois que l'expression infixée a été entièrement parcourue, les opérateurs restants dans la pile d'opérateurs sont dépilés et ajoutés à la pile d'opérandes. La pile d'opérandes contient alors l'expression postfixée, dans laquelle les opérateurs se trouvent après les opérandes. Il est important de noter que l'algorithme de Shunting Yard est

utilisé pour évaluer les expressions mathématiques de manière à garantir la priorité des opérateurs, en respectant les règles de précédence des opérateurs mathématiques standard. Il permet également de gérer les parenthèses dans les expressions mathématiques de manière à garantir que les calculs sont effectués dans l'ordre correct. En résumé, l'algorithme de Shunting Yard est un algorithme utilisé pour convertir une expression mathématique écrite en notation infixée en notation postfixée, en utilisant deux piles : une pile d'opérandes et une pile d'opérateurs. Il garantit la priorité des opérateurs en respectant les règles de précédence des opérateurs mathématiques standard et gère les parenthèses pour assurer que les calculs sont effectués dans l'ordre correct. Cette notation postfixée est ensuite utilisée pour évaluer facilement l'expression mathématique par une machine.

5.2 Implémentation

Premièrement, on récupère une chaine de caractère qui a été créé à partir des de l'interface de la calculatrice. Ensuite, il faut convertir cette chaine de caractères en liste de Tokens afin de pouvoir traiter plus facilement chaque élément (opérateur, opérande, Parenthèse. . . .). Il est possible de créer une liste de Token à l'aide de la struct suivante:

Si le token est un opérateur, alors l'attribut opérateur est différent de nulle, s'il le token est un opérande, alors l'attribut opérandes sera différent de null et finalement, si le token est une parenthèse, alors l'attribut parenthèse sera différent de null.

```
struct Token_Type
{
    struct Token_Operators* operator;
    struct Token_Operands* operands;
    struct Token_Parenthese* paranthese;
};
```

une fois notre liste de token créé, il nous faut quelque structure de données afin de réaliser l'algorithme de shunting yard: pile, file et éventuellement un arbre binaire pour évaluer le résultat.

Ensuite, on realise l'algorithme de shunting yard dans le but de pouvoir passer en RPN (notation polonaise inverse), puis une fois notre liste de token transforme en RPN, on peut alors construire un arbre binaire d'expression

régulière puis il suffit de faire un parcours en profondeur en ordre infixe afin de pouvoir évaluer le résultat.

6 Résolution de polynômes

6.1 Définitions

TOKEN: Structure représentant un élément. Un TOKEN contient un champ 'value', correspondant à un nombre ou bien au nombre ascii du caractère en cours de traitement (dans le cas d'une variable inconnue 'x', ce champ correspond au coefficient de cette variable). S'en viennent ensuite les champs ; 'préc', qui correspond à la précédence (ordre d'importance) de la donnée, ainsi que asso, qui correspond à l'associativité de cette dernière (dans le cas d'une variable inconnue 'x', ce champ représente son degré)

Poly: Structure représentant un polynôme. Il s'agit d'une liste chainée (comprenant un sentinelle) dont chaque élément correspond à un monôme du polynôme. Cette structure contient le champ 'value' qui correspond au coefficient du monôme en question, le champ 'degree' correspondant au degré de celui-ci ainsi que les champs 'next' et 'last' qui correspondent respectivement aux prochain et dernier monôme du polynôme

6.2 Shunting Yard 2.0

Afin de permettre la bonne manipulation de la donnée entrée, une nouvelle implémentation de l'algorithme de Shunting Yard a été réalisée. Cette version permet la manipulation d'expressions contenant une unique variable inconnue, représentée dans ce cas-ci par la lettre x, en plus de la manipulation d'expressions arithmétiques parenthésées. En premier, a lieu l'interprétation de la chaine de caractères entrée par l'utilisateur. A chaque élément est alors associé un TOKENqui est enfilé dans une file (représentée ici par la structure 'queue') . Une fois la chaine de caractères transformée en suite de 'TOKENs', l'algorithme principal de cette partie est entamé. Le but étant d'obtenir une notation polonaise inversée (ou RPN en anglais) grâce à une file contenant la nouvelle disposition des 'TOKENs' et une pile (représentée par la structure 'stack') qui sert de contenant pour les opérateurs et ainsi à gérer l'ordre d'usage de ceux-ci (grâce à leurs valeurs de précédence). Vient ensuite l'interprétation de l'expression en notation polonaise inversée. Les éléments de la file résultante sont empilés dans une nouvelle pile jusqu'à l'encontre d'un opérateur qui permettra de réaliser l'opération correspondante à celui-ci sur deux éléments se trouvant au sommet de la pile, puis de rempiler le résultat et ainsi de suite jusqu'à ce qu'il ne reste qu'un seul élément dans la pile, élément qui correspond au résultat de l'opération. A l'encontre d'une variable inconnue ('X') et après réalisation de l'opération en question, le 'TOKEN' correspondant à celle-ci est transformé en structure 'poly', correspondant ainsi à une partie d'un polynôme et est ensuite concaténé à une structure polynôme qui sera décrite par la suite. Ces deux algorithmes s'arrêtent immédiatement à l'encontre d'une erreur dans l'expression donnée.

6.3 Résolution de polynômes

Suite à l'algorithme de Shunting Yard est obtenue une structure Poly représentant le polynôme entré par l'utilisateur. Celui-ci passe alors par la fonction 'flattern' afin de regrouper tous le monômes de même degrés en seul puis par la fonction 'sort' dont le but est de trier le polynôme par ordre décroissant de degré. S'en vient ensuite la fonction de résolution qui se déroule selon trois cas dépendant de la valeur du plus haut degré du polynôme. La solution est alors renvoyée sous forme de structure 'solution' qui n'est autre qu'une liste chainée (sans sentinelle) étant sois vide (valeur NULL) soir remplie de solutions possibles pour l'équation donnée.

6.3.1 Fonction linéaire (polynôme de degré 1)

Celui-ci est sans doute le cas le plus simple. Le polynôme ayant un degré de 1 ou de 0, la liste chainée représentant celui-ci contient au pire deux éléments, il suffit alors de diviser la valeur du deuxième élément (monôme de degré nul) par le coefficient du premier élément.
Si le polynôme est une constante, celui-ci n'admet pas de solution et aucune valeur n'est renvoyé

6.3.2 Polynôme du second degré

Pour ce cas-ci, la méthode du discriminant est utilisée. Le polynôme ne pouvant contenir plus de 3 éléments, un vecteur de cette taille est créé, chaque case de celui-ci contient le coefficient du monôme de degré associé à l'indice de la case. Il suffit par la suite de calculer delta pour trouver les deux racines du polynôme grâce à la formule suivante :

$$x = \frac{-b - \sqrt{\Delta}}{2a} \tag{1}$$

avec: $\Delta = b - 4ac$

en partant d'un polynôme sous la forme: ax^2+bx+c

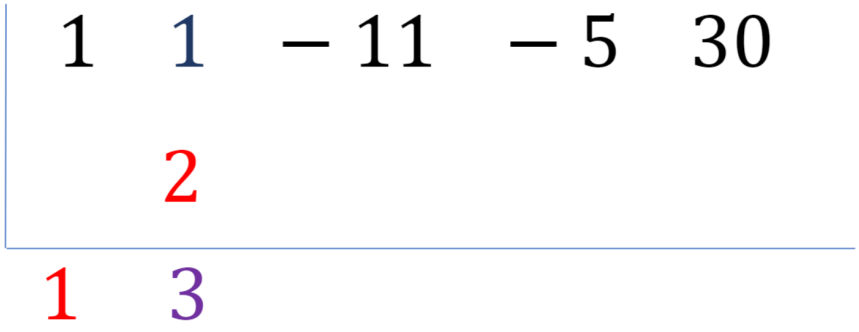
La chaine de solutions associée au polynôme ne peut pas contenir de racines non réelles

6.3.3 Polynôme de degré ≥ 3

Le principe est le suivant: factoriser le polynôme au maximum, tout comme pour les entiers. Tout entier peut s'écrire comme produit d'entiers premiers et cela est aussi applicable sur les polynômes. X^2+1 est premier et ne peut pas être factorisé. A l'opposé, X^2-1 n'est pas premier et peut s'écrire sous la forme de produit de polynômes premiers (x-1)*(x+1). Afin d'entamer la résolution du polynôme, il est essentiel de trouver une première racine de celui-ci. Un premier programme est alors lancé. Celui-ci à pour but de trouver soit une valeur pour laquelle le polynôme s'annule, soit deux valeurs entières pour lesquelles le polynôme admet des images de signes opposés. A la suite de ce dernier cas, la méthode de la bissectrice est utilisée affin de trouver une valeur rapprochée de la racine au dix-millionième près (7 digits après la virgule), cette méthode se rapproche fortement de la dichotomie. La racine trouvée nous sert alors à factoriser le polynôme en vertu de la division synthétique. Le procédé est le suivant.
Soit :

$$\mathbf{P(x)} = \mathbf{X}^4 + X^3 - 11X^2 - 5X + 30$$

Pour x = 2, P(2) = 0. Deux est donc une solution du polynôme et celui-ci peut être factorisé par '(x-2)' et s'écrire sous la forme (x-2)*Q(x). Afin de trouver Q(x), il suffit de réaliser la division synthétique de P(x) par 2 comme suit :
Soit la liste contenant le coefficient de chaque monôme du pôlynome : **Le résultat de l'opération correspondra aux coefficients de Q(x).**
Le premier coefficient de Q(x) est égal au premier coefficient de P(x). Le deuxième correspondra au produit du coefficient qui le précède par le facteur (2 en l'occurrence), auquel il faut ajouter le coefficient de la même case du polynôme de départ. L'opération effectuée est alors 1*2+1



Le même procédé permet l'obtention du coefficient suivant, l'opération correspondant à celui-ci est 3*2-11

1	1	−11	−5	30
	2	6		

1	3	−5		

Il suffit pour la suite de répéter cette opération sur le reste des coefficients afin d'aboutir au résultat suivant :

1	1	−11	−5	30
	2	6	−10	−30

1	3	−5	−15	0

La Valeur du dernier coefficient est nulle, ce qui implique que la division effectuée n'admet pas de reste. Cela confirme le fait que 'x-2' soit un facteur de P(x). On obtient alors :

$$\mathrm{Q(x)} = \mathrm{X}^3 + 3X^2 - 5X - 1$$

et:

$$\mathrm{P(x)} = \mathrm{(X\text{-}2)}(\mathrm{X}^3 + 3X^2 - 5X - 1)$$

L'opération est ensuite répétée ensuite répétée sur Q(x) afin d'obtenir un deuxième facteur du polynôme, puis sur le nouveau polynôme obtenu, cela jusqu'à l'obtention d'un polynôme de degré 2 au plus dont la résolution est enfantine.

Le calcul des coefficients peut être généralisé par le principe suivant :

- Soit A, le vecteur contenant les coefficients du polynôme de départ (le polynôme est supposé trié en ordre décroissant de degré)
- Soit f, le coefficient diviseur
- Soit B, le vecteur résultant de la division synthétique de A par f

En partant de 0:

$$B[0] = A[0]$$

$$B[i] = B[i\text{-}1]*f+A[i] \text{ avec i¿0}$$

A la suite de cette partie se trouvent les structures utilisées.

```
struct queue{
    struct queue* next;
    struct queue* previous;
    struct queue* last;
    struct token* value;
};
```

```
struct stack{
    struct stack* next;
    struct token* value;
};
```

```
struct token{
    double value;//degree for poly
    int prec;
    int asso;//coeff for poly (starts at 1)
};
```

```
struct poly{
    struct poly* next;
    struct poly* last;
    double value;
    double degree;
};

struct solution{
    struct solution* next;
    double value;
};
```

7 Fonctionnalitées mémoires

7.1 sauvegarde de l'historique

L'intégralité des calculs effectués et des résultats sont sauvegardés et peuvent être affichés à tout moment par l'utilisateur. Il suffit pour cela de cliquer sur le bouton "historique", l'historique complet depuis la création de la calculatrice sera alors affiché dans le terminal de l'utilisateur.

7.2 sauvegarde mémoire

Il est possible de sauvegarder un résultat pour le réutiliser dans une opération plus tard. Il faut pour cela cliquer sur le bouton "M+", le dernier résultat calculé sera alors enregistré. Cliquer sur le bouton "M-" permettra de récupérer cette valeur et de l'utiliser dans le calcul actuel.

8 compilateur python

Il est possible pour l'utilisateur d'écrire et voir le résultat de l'exécution d'un code python directement dans la calculatrice. L'implémentation dans la calculatrice n'a pas encore été faite, le compilateur doit être utilisé à part. il faut pour cela écrire son programme python dans un fichier .txt, puis lancer le compilateur avec le fichier .txt en argument. Le résultat de la compilation sera alors affiché dans la terminal de l'utilisateur.

Programme python "python test file.txt" pour tester le compilateur python

```
1   print(3+8)
2   print()
3   print('ceci est un texte')
4   print()
5   for i in range(4):
6           print('ceci est un texte puis un nombre:',i)
```

résultat de l'exécution de ce programme dans un terminal

```
◆ BrainUp_Corp. git:(main) ❯❯❯ ./python_executable python_test_file.tx
11

ceci est un texte

ceci est un texte puis un nombre: 0
ceci est un texte puis un nombre: 1
ceci est un texte puis un nombre: 2
ceci est un texte puis un nombre: 3
```

9 Site web

Concernant le site web, celui-ci à été entièrement implémenté en Javascript, CSS ainsi que HTML. Celui-ci est accessible à l'adresse suivante : Matthiaskd.github.io et est scindé en plusieurs parties. On peut alors y retrouver la section ''what is Seda '' qui décrit simplement et globalement l'application. Vient ensuite la section ''features" qui sera mise à jour tout au long du développement de ce projet. En effet, chaque partie accomplie aura un manuel d'utilisation sur le site. La page web présentera des versions simplifiées ''démos" des fonctionnalités implémentées. Pour finir, en bas de page se trouve la section ''membres" qui présente simplement l'équipe du projet ainsi que leurs différentes responsabilités.

10 Interface graphique

Un gestionnaire d'onglets a été ajouté afin de permettre à chaque fonctionnalité de profiter de l'intégralité de la fenêtre de la calulatrice dans un onglet qui lui est dédié.

Un affichage de sortie a été ajouté dans la fonctionnalité de la calculatrice standard afin d'afficher une partie de la mémoire qui contient les précédents calculs.