TP n°1

V. Ledda

9 janvier 2025

---

1 Introduction

1.1 Présentation du TP.

L'objectif du TP est d'utiliser une bibliothèque en C pour créer des fonctions sur le thème des polynômes.

Le TP est à rendre – à la fin de votre séance de TP – sur moodle sous la forme d'une archive contenant un document `pdf` qui répond aux questions posées et explique votre démarche ainsi que les sources de vos programmes. Vous aurez préalablement :

- Créer des binômes
- Préparer la première partie du TP :
  - Lire la fiche de présentation de la librairie `simplepoly`
  - Compiler et tester les exemples de la fiche de présentation

1.2 Organisation du travail

Je vous conseille de vous y mettre le plus tôt possible, c'est un travail conséquent qui vous est demandé. L'objectif est double : progresser en mathématiques et en programmation en `C`. L'évaluation de votre travail se basera sur une étude technique à l'aide de tests automatisés et la lecture de votre compte-rendu qui explique votre démarche.

La note attribuée au groupe représentera 10 % de la note du `Math2`.

1.3 Recommandations techniques

Je vous conseille d'utiliser un éditeur de texte avancé qui prend en charge la complétion de code, la navigation dans la documentation et dans le code, l'intégration d'un débogueur. Sur les postes de travail des salles informatiques vous pourrez utiliser VS Code qui possède ces fonctionnalités, mais d'autres choix sont possibles.

Télécharger l'archive disponible sur `moodle` et décompresser les fichiers dans votre répertoire personnel et travailler dans le l'arborescence ainsi créée. Dans `VScode`, il suffit d'ouvrir le répertoire de travail pour commencer à travailler.

Vérifier que le fichier `presentation.c` compile et que le programme compilé affiche les exemples de la fiche de présentation.

Créer un fichier `tp.c` dans le répertoire source et indiquer au compilateur le chemin de la bibliothèque par :

```
#include "../includes/simplepoly.h"
```

Structurer et **documenter** votre code de la manière suivante :

```
#include "../includes/simplepoly.h"
//Prototypes
int ma_belle_fonction(double * un_nom_explicite);
int* une_autre_fonction(int un_nom_explicite);


// Le point d'entrée du programme
int main(void)
  {
    //expériences et tests de votre programme (non évalués)

    return 0
 }


//Partie évaluée
/**
   * ma_belle_fonction permet de ...
   */
int ma_belle_fonction(double * un_nom_explicite)
  {

}
/**
   * une_autre_fonction permet de ...
   */
int*  ma_autre_fonction(int un_nom_explicite)
  {

}
```

2 Travail demandé - partie 1 (À faire avant le TP)

2.1 Affichage

La fonction `s_affichage_poly` est spartiate... Écrire une fonction dont le prototype est le suivant :

```
void s_poly_Affiche(polyZ *P);
```

Cette fonction affichera sur la sortie standard une expression du polynôme avec l'indéterminée X.

```
s_polyZ* P=s_polyZ_alea(10,-3,2);
s_poly_Affiche(P);
s_polyZ_free(P);
```

```
2 X^10 - X^9 + X^8 + X^7 + 2 X^6 - 2 X^5 - X^4 + X^3 - 2 X^2 -1
```

```
long valeurs[5]={-1,-1,1};
s_polyZ* P=s_polyZ_coef(2,valeurs,3);
s_poly_Affiche(P);
s_polyZ_free(P);
```

```
-X^2 -X +1
```

```
s_polyZ* P=s_polyZ_new(5);
s_poly_Affiche(P);
s_polyZ_free(P);
```

```
0
```

2.2 Algorithme de Hörner

La méthode Hörner permet entre autres de déterminer rapidement le quotient d'un polynôme par $X-a$. Elle est décrite en détail sur Wikipedia ou ici.

Le fichier `simplepoly.h` défini le type `QRz` (quotient et reste) comme une structure qui contient deux pointeurs vers deux `s_polyZ`. Créer une fonction dont le prototype est le suivant qui implémente la méthode de Hörner.

```
QRz* s_horner_Z(s_polyZ* p, long a);
```

```
s_polyZ *p;
QRz *qr;
long valeurs[8] = {5, 0, 0, 0, 1, 2, 3, 4};
p = s_polyZ_coef(7, valeurs, 8);
printf("P:\n");
s_poly_Affiche(p);

qr=s_horner_Z(p, -7);

printf("Q:\n");
s_poly_Affiche(qr->Q);
printf("R:\n");
s_poly_Affiche(qr->R);

s_polyZ_free(p);
s_qrz_free(qr);
```

P :
$5\ X^7 + X^3 + 2\ X^2 + 3\ X\ 4$
Q :
$5\ X^6 + 35\ X^5 + 245\ X^4 + 1715\ X^3 + 12006\ X^2 + 84044\ X\ 588311$
R :
4118181