Université Sultan Moulay Slimane

École nationale des sciences appliquées de Khouribga

DEVOIR LIBRE DE PROGRAMMATION C :

Année universitaire 2020/21

Réalisé par : **EL FATHI Zakaria**

Encadré par : Mr. **SAADI Mostafa**

Mlle **Mhammedi Sajida**

Introduction :

Dans le cadre de notre deuxième année en école d'ingénieurs à ENSA Khouribga nous avons eu pour tâche la réalisation d'un devoir informatique. Notre objectif était la réalisation d'un ensemble d'opérations sur les polynômes d'une part, et permettre à un utilisateur (architecte, menuisier, …) de mesurer certaines grandeurs géométriques d'une autre part bien sur tout en utilisant le langage C. Dans ce dossier nous allons exposer la démarche suivie pour réaliser ce projet avec des exemples bien choisis pour montrer l'intérêt.

PARTIE 1 : (calcul sur les polynomes)

Les cas d'utilisation :

Acteur : l'utilisateur

Cas d'utilisation pour l'utilisateur :

1. Initialisation (saisie de degré et des facteurs du polynôme)
2. Affichage du polynôme dans la forme habituelle
3. Saisie d'un valeur réelle pour évaluer le polynôme
4. Operations sur les polynôme (multiplication et addition)
5. Quitter le programme

Déclarations préliminaires: d'une contante de type entier et une structure de deux champs

La fonction principale (main) :

La fonction main utiliser dans cette partie permet le test des opérations sur un polynôme, cela se fait par l'intermédiaire d'un menu interactif tel qu'il offre le choix entre des valeurs entières comprises entre 0 et 5 et à l'aide d'un contrôle de saisie ce message continuera à s'afficher tant de fois que l'utilisateur entre une valeur erronée.

```
WELCOME TO MY PROGRAMME

FAIRE UN CHOIX !
  1:   pour la saisie de votre polynome :
  2:   Pour l'affichage de votre polynome' :
  3:          Pour evaluer votre polynome :
  4:          Pour additionner deux polynome :
  5:          Pour multiplier votre polynome
  0:          Pour quitter le programme

Votre choix :
```

A chaque choix elle fait appel à une fonction bien précise

Les fonctions utilisées :

- *SaisiePoly :* C'est une fonction sans paramètres *(*procédure) permettant d'entrer au clavier le degré et les coefficients d'un polynôme P passé en paramètre. Elle permet la déclaration de deux variables de nom ' n','i' de type entiers(int), affichage du message 'ENTRER LE DEGRE DE VOTRE POLYNOME', prise d'une valeur de type entier et la stocker dans p->degré, affectation du degré du pointeur a n, dans le cas ou le cas ou n est nulle, on fait appel a la fonction NulPoly (description de fonctionnement qui suit)

```
 Votre choix :1
SAISIR VOTRE 1ER POLYNOME:
ENTRER LE DEGRE DE VOTRE POLYNOME 2
saisir le 1 coefficient : 1
saisir le 2 coefficient : 2
saisir le 3 coefficient : 3
```

- *NulPoly* : c'est une fonction sans retour (procédure), elle prend en argument un polynôme et ensuite elle permet de le transformer en polynôme nul, en parcourant tous les coefficients et en même temps les remplir avec des zéro, le paramètre est un pointeur de type 'pol'

- *VoirPoly :* c'est une Procédure affichant l'expression de P(x), P étant un polynôme donné en paramètre. On fait la déclaration d'une variable de type entier 'int' et de nom 'n' , après l'affectation du degré de p a n, en suite une boucle de i(définition de type entier) débutant de n jusqu'a la condition soit satisfaite i>0 , en ordre décroissant. Test: si le coefficient numéro 'n-i' est diffèrent de 0, deuxième test de i s'elle est diffèrent de 1 et affichage du polynôme. Sinon dans le cas échéant, affichage du message (pour ne pas avoir un plus a la fin du polynôme

```
 Votre choix :2
 VOTRE POLYNOME
P1(x) = 2.00*x^2 + 2.00*x^1 + 2.00
```

- *EvalPoly* : c'est la fonction d'affichage calculant la valeur de P(x), le réel x et le polynôme P étant passés en paramètres, déclaration de la variable n de type entier et déclaration et initialisation de la variable 'racine de type réel, affectation de degré de p a n, ensuite on fait une boucle de i(initialisation) =1 jusqu'a la condition i<=n soit vérifiée de pas =1. En suite le calcul de la somme des coefficients multiplier par les puissances de x, en fin il vient l'affichage de la valeur

```
 Votre choix :3
ENTRER LA VALEUR DE X :
2

CHOISIR UN POLYNOM A EVALUER :/n 1
le resultat de P(2.00) = 14.00
```

- *AddPoly*: C'est la procédure calculant la somme de deux polynômes de paramètres p1,p2 et un pointeur p. on fait la définition des variables entiers n et a, affectation a degré de p le degré de p1 si degré p1 est supérieur a degré de p2, sinon on affecte le degré de p2, affectation a a la différence des degrés, affectation de degré de p a n. après on fait une boucle 'for' boucle de début i=0 (définition), jusqu'a i<n soit vérifier de pas égale a 1. On fait un test: si le degré de p1 est supérieure a degré de p2 , alors affectation de la somme de p2 d'indice i-a et celui de p1 d'indice i au coefficient numéro i, sinon si degré de p1 est supérieure a celui de p2, affectation de coefficient numéro i en somme de celui de p1 numero i-a au coefficient i de p. dans le

```
Votre choix :4

LA SOMME DE CES DEUX POLYNOME EST  :
2.00*x^2 + 4.00*x^1 + 4.00
```

cas échéant, affectation de la somme de coefficient i de p1 et coefficient i de p2 a p. pour l'affichage du message on fait appel de la procédure 'VoirPoly' de paramètre le pointeur p

- *MultPoly* : c'est une procédure calculant le produit de deux polynômes. On fait la déclaration des variables n et a de types entiers, affectation de la somme des degrés de p1 et p2 a degré de p puis l'affecter a n, affectation de degré de p1 a a si (degré de 1 supérieur a celui de p2) sinon on affecte le degré de p2, ensuite boucle débutant de 0 (i déclarer et initialiser) jusqu'a i<=n soit vérifier avec un pas de 1. Une boucle débutant de 0 (j déclarer et initialiser) jusqu'a j<=i soit vérifier avec un pas de 1, après on fait la sommation des coefficients et en fin appel de la procédure avec comme paramètre le pointeur p

PARTIE 2 : (mesure de certaines grandeurs géométriques)

Dans cet exercice on vise établir un programme en langage C qui permet à un utilisateur (architecte, menuisier, …) de mesurer certaines grandeurs géométriques. Un point du plan sera représenté par une structure contenant le nom du point (A, B, E, …) et deux réels x et y qui représentent respectivement l'abscisse et l'ordonné.

// ON NE PEUT PAS UTILISER UN TABLEAU POUR ENREGISTRER LES INFORMATIONS DESCRIPTIVES D'UN POINT PUISQU'IL NE PEUT PAS CONTENIR DES ELEMENTS DE TYPES DIFFERENTS //

On fait les declarations suivantes :

1. Déclaration d'une structure de nom 'point' et de trois champs
2. Déclaration de d'une structure de nom 'vecteur' de deux champs
3. Déclaration d'une structure de nom 'spt' et de champ spt
4. Déclaration d'une structure de nom 'svect' et de champ vecteur svect

Les fonctions utilisées :

- *Distance* : C'est une fonction qui calcule la distance entre deux points p1 et p2. On fait la déclaration de la fonction de nom 'distance' de paramètres p1 et p2 de types 'spt', la retournée est de type réel, notons que la distance entre deux point est la racine carrée de la différence enlevée au carré de leurs abscisses et ordonnées, cette distance sera notée comme valeur de retour de la fonction

- *Deplacer* : C'est une FONCTION qui permet de déplacer le point p. declaration de la procédure 'Deplacer' de paramètres le pointeur de type spt et dx, dy de type float, après on fait l' ajout de dx a l'abscisse et dy a l'ordonnée

- *ProdScal :* Une fonction qui calcule le produit scalaire de deux vecteurs. Donc on fait la déclaration de la fonction Prodscal de retournée de type 'float' et de paramètres de type svect, ensuite affectation du produit des abscisses a la variable de type 'float' nommée

  Abscisse et celui des ordonnées a la variable nommée 'ordonnee' et la valeur de retour est la somme

- *Colinéaires* : Une fonction qui retourne 1 si les trois points p1, p2 et p3 sont colinéaires, retourne 0 dans le cas échéant. Donc on fait la déclaration de la fonction 'Colineaire' de type de retour entier et de paramètres de types 'spt', on essaie a l'aide de la méthode du déterminant de calculer ce dernier si la condition (déterminant nul) est bien vérifier la valeur de retour sera alors 1 sinon la valeur de retour sera 0

- *AfficheEquatCar* : Une fonction affichant (procédure) sur l'écran l'équation cartésienne de la droite (p1p2).donc on procède comme suit : la différence des ordonnées est stockée dans a(variable de type float) , la différence des abscisse est stockée dans b(float), calcul de c (la constante dans l'équation cartésienne) a partir de a et b, après il vient l'étape de l'affichage de l'équation cartésienne dans sa forme canonique

- *triangleRrectangle* : une fonction qui retourne 1 si p1p2p3 est un triangle rectangle en un point, elle retourne 0 sinon. donc on fait la déclaration de la fonction de type de retour 'entier' et de paramètres de type 'spt', on calcule la distance entre chaque deux point (les trois cotes d'un triangle), puis on fait un test 'si' pour s'assurer qu'au moins la relation de Pythagore est vérifiée une fois, si oui la valeur de retour sera égale a 1 sinon on retourne 0

- *triangleIsocele* : Une fonction qui retourne 1 si le triangle p1p2p3 est isocèle, de tête p2 ; elle retourne 0 sinon. Pour cela on fait la déclaration de la fonction de type de retour entier et de paramètres de types 'spt', on calcule la distance entre p2 et p3 d'une part et celle entre p1 et p2 d'une autre part. s'il s'agit de même distance entre p1p2 et p2p3 la valeur de retour sera 1 et dans le cas échéant la valeur de retour sera 0

- *SurfaceGEF* : calcul de la surface de GEF, on fait la déclaration de la fonction 'SurfaceGEF' de type de retour réel et de paramètre de type 'spt' et 'float', déclaration de deux variables de type 'spt', ensuite on fait une affectation de l'ordonnee entre(Z) a celle ci de G car ils ont le meme ordonnee, et l'abscisse de G est celui ci de Z moins l'abscisse de une distance égale a (sqrt2)*r, affectation de l'abscisse de Q a celui de F, affectation a l'ordonnee de F la distance entre E et F plus l'ordonnee de F. enfin la valeur de retour sera le produit des deux distances entre G,F et F,Q divisée par deux

- *Trier* : pour réaliser une fonction de tri définition de la procédure de paramètre un pointeur de type 'spt', déclaration de la variable 'orig' de type 'spt', puis son initialisation avec des valeurs nulles, introduction d'une boucle parcourant le tableau, boucle parcours du tableau de l'inverse et aussi calcule de la distance entre l'origine et la j-eme case (point), la 2eme

distance sera la comparaison avec l'origine aussi mais avec le point de la case qui suit celle en j, test se basant sur la distance entre le point origine et celle de j et j+1 utilisation de la fonction déjà prototypée si la condition est vérifiée puis incrémentation des deux compteurs

- *ft_swap : c'est une fonction intermédiaire qui permet d'échanger les éléments d'un tableau cette fonction ici est déjà prototypée*