

- Apprendre à utiliser votre éditeur de texte (**jedit** ou **atom**) et les commandes systèmes pour saisir et gérer les fichiers sources des programmes en C.
- Connaître les commandes de compilation et d'exécution d'un programme.
- Traduire un algorithme en langage C.
- Utiliser la bibliothèque mathématique du système dans des programmes.
- Utiliser les fonctions d'entrées/sorties de base du langage C.
- Découvrir quelques limites du calcul sur ordinateur.

Consignes :

- 1 Créer un répertoire `tp-algo-prog` à la racine de votre compte.
- 2 Avec votre éditeur de texte, ouvrir dans le répertoire ci-dessus un nouveau fichier nommé `poly1.c` (ou tout autre nom se terminant par `.c` pour signifier à votre éditeur que le fichier est un source en langage C).
- 3 Saisir le programme fourni (voir diapo 7) dans ce fichier.
- 4 Tenter de compiler puis d'exécuter ce fichier.
- 5 Corriger les erreurs dans ce fichier source et trouver le but de ce programme.

- Pour compiler le programme :


```
gcc -Wall -o poly1 poly1.c
```

- Pour exécuter le programme :

```
./poly1
```

- En sélectionnant plusieurs lignes puis en choisissant l'item `Auto Indent` du menu `Edit >> Lines`, le code de ces lignes sera réindenté.
- Pour atteindre une ligne par son numéro, vous pouvez utiliser le raccourci `Ctrl-g`.

- Si vous sélectionnez plusieurs lignes et appuyez sur **Ctrl-i**, le code de ces lignes sera réindenté.
- Pour une meilleure présentation, vous pouvez installer le plugin **AStyle Beautifier** et l'utiliser via le menu **Plugins**.
- Afin de diminuer l'indentation par défaut, dans le menu **Utilitaires >> Options Générales**, vous pouvez choisir la rubrique **Édition** et régler la *Profondeur de tabulation* et le *Retrait* sur 4 ou même 2.
- Pour atteindre une ligne par son numéro, vous pouvez utiliser le raccourci **Ctrl-g**.

- Les flèches droite et gauche permettent de déplacer le curseur pour modifier la commande en cours de frappe.
- Les flèches haut et bas permettent de parcourir l'historique des commandes.
- La touche tabulation (**Tab**,  ou $\rightarrow|$) permet de compléter le nom d'une commande, d'un répertoire ou d'un fichier.
- Organiser l'écran pour voir à la fois votre terminal et votre éditeur de texte (**jedit**).

La commande de compilation ci-dessous :

```
gcc -Wall -o poly1 poly1.c
```

peut se détailler comme suit :

- gcc** Le nom du compilateur C de GNU (*GNU C Compiler*).
- Wall** (pour *warnings all*) Option demandant au compilateur d'afficher tous les messages d'avertissement.
- o poly1** L'option **-o** (pour *output*) suivi d'un nom de fichier (ici **poly1**) indique au compilateur le nom du programme binaire à produire.
- poly1.c** Le nom du ou des fichiers sources à compiler.

— *poly1.c* (premier programme) —

► *poly1.c*

```
/* premier programme en C */
#include <stdio.h>

int main() {
    double a, b;
    double x;
    double resultat;

    /* saisie des parametres */
    printf("Donnez la valeur de a : ");
    scanf("%lf", &a);

    printf("Donnez la valeur de b : ");
    scanf("%lf", &b);

    printf("Donnez la valeur de x : ");
    scanf("%f", &x);

    /* calcul */
    resultat = a * x + b;

    /* affichage des resultats */
    printf("a * x + b =\n");
    printf("%g\n", resultat);

    return 0;
}
```

- À quoi sert ce premier programme ?
- Que se passe-t-il pour les valeurs suivantes de a , b et x ?

a	b	x
10^9	-1×10^{18}	10^9
10^{20}	-1×10^{40}	10^{20}
10^{25}	-1×10^{50}	10^{25}
10^{200}	0	10^{200}

Expliquer les résultats obtenus.

Consignes :

- 1 Traduire l'algorithme (diapo 10) de résolution d'un polynôme du second degré en un programme C nommé **poly2.c**. Les valeurs des paramètres de l'algorithme seront demandées à l'utilisateur.
- 2 Compiler et tester ce programme.
- 3 Tester son comportement avec une valeur nulle pour a .
- 4 Compléter ce programme pour qu'il soit utilisable quelles que soient la valeur de a .
- 5 Si le temps le permet, modifier ce programme pour qu'il calcule et affiche les solutions imaginaires lorsque le discriminant est négatif.

POLYNÔME SIMPLE(a, b, c)

Résolution du polynôme $ax^2 + bx + c = 0$ (lorsque $a \neq 0$)

Paramètres : a, b, c (réels) les coefficients du polynôme.

Variables : x_1, x_2, x (réels) les solutions éventuelles du polynôme.
 delta (réel) le discriminant.

Début

$\text{delta} \leftarrow b^2 - 4 \times a \times c$

Si ($\text{delta} > 0$) **Alors**

$x_1 \leftarrow \frac{-b + \sqrt{\text{delta}}}{2 \times a}$

$x_2 \leftarrow \frac{-b - \sqrt{\text{delta}}}{2 \times a}$

Action : AFFICHER("Deux solutions réelles :", x_1, x_2)

Sinon

Si ($\text{delta} = 0$) **Alors**

$x \leftarrow \frac{-b}{2 \times a}$

Action : AFFICHER("Une solution réelle :", x)

Sinon

Action : AFFICHER("Aucune solution réelle")

Fin Si

Fin Si

Fin

Si (*condition*) **Alors**

| ...*instructions si vrai*...

Fin Si



```
if (condition) {  
    ...instructions si vrai...  
}
```

Si (*condition*) **Alors**

| ...*instructions si vrai*...

Sinon

| ...*instructions si faux*...

Fin Si



```
if (condition) {  
    ...instructions 1 si vrai...  
} else {  
    ...instructions 2 si faux...  
}
```

- Il n'existe pas d'opérateur *puissance* en C. Pour calculer le carré d'une valeur, il suffit de la multiplier par elle-même.
- Pour calculer la racine carrée d'une valeur, on peut faire appel à la fonction `sqrt` de la bibliothèque mathématique.
 - Déclarer l'utilisation de cette bibliothèque au début du fichier source :

```
#include <math.h>
```

- Utiliser la fonction `sqrt`. Exemple :

```
a = sqrt(2) + 1;
```

- Indiquer au compilateur de relier le programme exécutable à cette bibliothèque :

```
gcc -Wall -o poly2 poly2.c -lm
```

Le `-l` signifie *library* et le `m` est pour le mot mathématique.

■ Syntaxe de `printf` :

```
printf("...format...", v1, v2, ...);  
// les ... finaux indiquent qu'on peut insérer autant de valeurs que nécessaire
```

- "...format..." est un texte affiché *tel quel* sauf que chacune des séquences `%Z` est remplacée dans l'ordre par l'une des valeurs qui suivent (`v1`, `v2`, ...). Choisir `Z` selon le résultat attendu.

<code>%Z</code>	type	nature d'affichage
<code>%c</code>	char	caractère
<code>%d</code>	int	valeur entière (base 10)
<code>%x</code>	int	valeur entière (base 16)
<code>%f</code>	double	valeur réelle
<code>%e</code>	double	valeur réelle (notation scientifique)
<code>%g</code>	double	valeur réelle (au mieux de <code>%f</code> et <code>%e</code>)

- Exemple pour afficher deux valeurs (une entière et une réelle) :

```
printf("Voici deux valeurs : %d et %f\n", 103, 12.4e4);
```

Ce qui affiche :

Voici deux valeurs : 103 et 124000