. Les pointeurs de fonction

Il reste un dernier type de pointeur que nous avons pas vu jusqu'ici : les **pointeurs de fonction**.

Jusqu'à maintenant, nous avons manipulé des pointeurs sur objet, c'est-à-dire des adresses vers des zones mémoires contenant des *données* (des entiers, des flottants, des structures, etc.). Toutefois, il est également possible de référencer des *instructions* et ceci est réalisé en C à l'aide des pointeurs de fonction.

1 Déclaration et initialisation

Un pointeur de fonction se définit à l'aide d'une syntaxe proche des prototypes de fonction. Sans plus attendre, voici ci-dessous la définition d'un pointeur sur une fonction retournant un int et attendant un int comme argument.

```
int (*pf)(int);
```

Comme vous le voyez, il est nécessaire, tout comme les pointeurs sur tableau, d'entourer le symbole * et l'identificateur de parenthèses, ici afin d'éviter que cette déclaration ne soit vue comme un prototype et non comme un pointeur de fonction. Autre particularité : le type de retour, le nombre d'arguments et leur type doivent également être spécifiés.

2 Initialisation

cela est réalisé à l'aide de l'opérateur &.
En fait, dans le cas des fonctions, il n'est pas obligatoire de recourir à cet opérateur, ainsi, les deux syntaxes suivantes sont correctes.

```
int (*pf)(int);

pf = &fonction;
pf = fonction;
```

Ceci est dû à une conversion implicite : **un identificateur de fonction est converti en un pointeur sur cette fonction**. L'utilisation de l'opérateur & est donc facultative, mais elle a le mérite de clarifier un peu les choses.

3 Passage en argument

Comme n'importe quel pointeur, un pointeur de fonction peut être passé en argument d'une autre fonction, c'est d'ailleurs tout l'intérêt de ceux-ci. Pour ce faire, il vous suffit d'employer la même syntaxe que pour une déclaration.

Exemple :

```
// Fonction f(x) à résoudre  Exemple : f(x) = x^2 – 2
double f(double x) {
    return x*x - 2;
}

// Dérivée de f(x)
double df(double x) {
    return 2*x;
}

// Fonction qui prend un pointeur de fonction en paramètre

void executer(double (*fonction)(double), double a)
{
    double res = fonction(a);
    printf(" resultat : %lf\n", res);
}


int main(int argc, char** argv) {
    executer(&f, 2);
    executer(&df, 2);
    return (EXIT_SUCCESS);
}
```

4 Exercice :

Programmer en langage C une fonction newton qui reçoit deux arguments la fonction f(x) et sa fonction dérivée df(x).

Vous pouvez vous inspirez du programme python suivant :

```
def f(x):
    return x**2 - 2

def derive(x):
    return 2*x


def newton(f, derive, a, e):
    delta = 1
    compteur = 0
    while( delta > e):
        x = - f(a) / derive(a) + a
        delta = abs( x – a)
        print(x,delta)
        a = x
        compteur += 1
    return x, compteur


print ( newton (f, derive, 2, 0.000001))
```