Pointeur de fonction

L’outil qui permet la programmation par événements pour les interfaces graphiques (GUI)

Objectifs:

- Comprendre le concept de pointeur de function
- S’en servir partiellement pour exploiter GTKmm

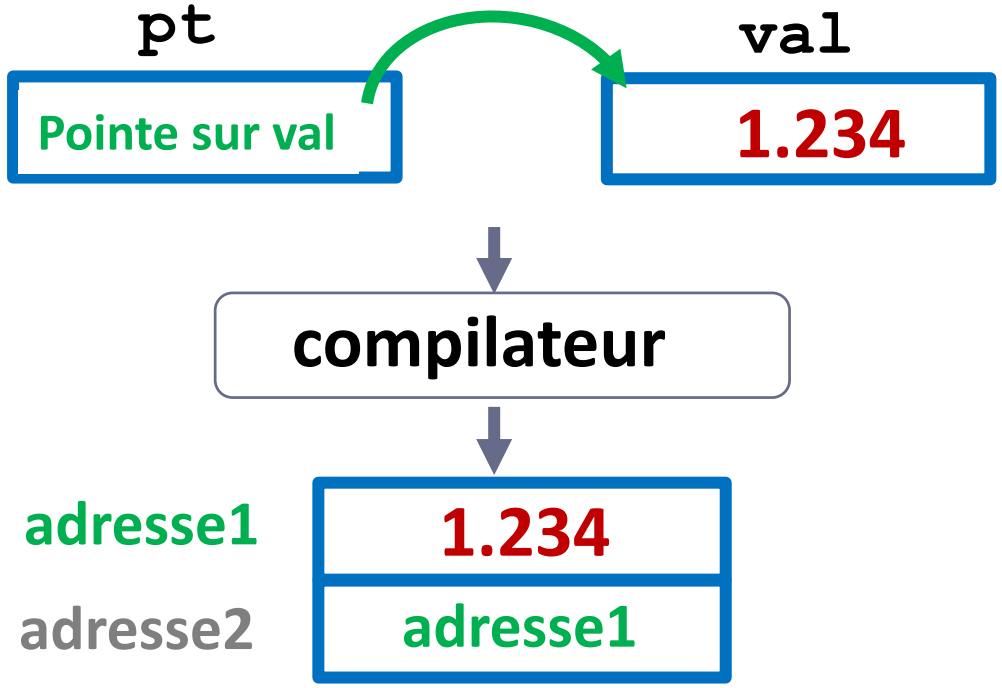
Plan:

- Rappel sur le concept de pointeur
- déclaration d'un pointeur de fonction
- Exemple: passage de fonction en paramètre
- Notre usage: passer des fonctions en parametre à GTKmm

Rappel sur le concept de pointeur

Un pointeur est une **variable** indépendante qui mémorise une **adresse** d'un octet de la mémoire

```
double val(1.234);
double *pt = &val;
```



adresse1 est l'adresse du premier octet occupé par val

Les noms de variables sont seulement destinés aux personnes qui écrivent le code.

ils n'existent plus après la production de l'exécutable.

L'exécution du code exploite seulement les **adresses** des variables et des fonctions.

Question: c'est quoi l'adresse d'une fonction ?

Pointeur sur une fonction (1)

intérêt: une fonction peut être «*passée en paramètre*» à une autre fonction simplement en transmettant son nom.

Car le *nom de la fonction,* sans les parenthèses ( ),
est **l'adresse** de sa première instruction exécutable

Remarque : l'argument formel correspondant est de type **pointeur de fonction** dont le prototype doit correspondre à celui de la fonction transmise.

Ex: soit le prototype `int f(double,int);`

`f(2.5, 9)` applique l'opérateur appel de fonction **( )** sur **f**
`&f` est l'adresse de la fonction **f**
`f` désigne AUSSI l'adresse de cette fonction **f**

Pointeur sur une fonction (2)

Le type d'un pointeur de fonction doit faire apparaitre:

- le fait qu'il mémorise une adresse
- le prototype de la fonction pointée

Ex: soit le prototype `int f(double,int);`

Déclaration d'un pointeur **adf** sur une fonction de ce prototype:

```
int (*adf)(double,int);
```

prioritaire

Les parenthèses autour de **(*adf)** sont nécessaires pour **forcer la priorité** de * qui déclare **adf** *comme pointeur*

Sinon, l'opérateur **( )** aurait déclaré **adf** comme une *fonction* car il est prioritaire par rapport à l'opérateur *

Pointeur sur une fonction (3)

L'instruction suivante ne déclare PAS **g** comme un pointeur de fonction:

```
int *g(double,int);
```

Basse priorité

Haute priorité

g est l'identificateur d'une fonction qui possède deux arguments formels de type **double** et **int**…

…et qui renvoie
une valeur de type **int***

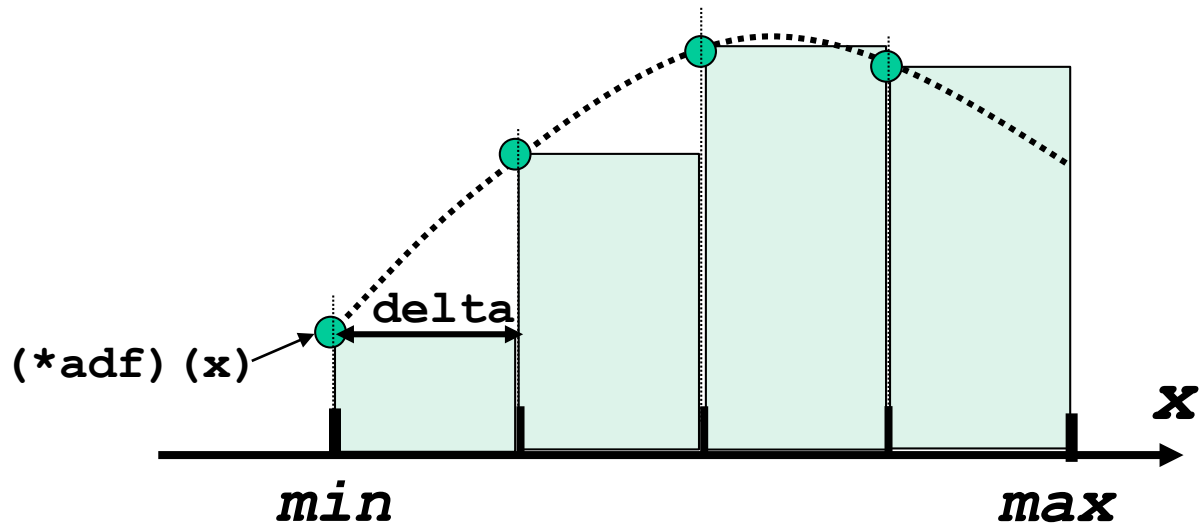
Initialisation et usage d'un pointeur de fonction

```
1  int  f(double, int);
2     …
3  f(12.3, 5);
4     …
5   int (*adf)(double, int); // n'est pas initialisé
6   …
7   adf = f ;                // ou: adf = &f ;
8   …
9   (*adf)(0.5, 2);          // appelle la fonction f
10 adf(0.5, 2);              // aussi correct en C++
11 f(0.5,2);
```

Les lignes 9 et 10 sont possibles en C++ pour appeler la fonction à l'aide de **adf**
C'est équivalent à appeler directement f comme sur la ligne 11

Passage d'une fonction en paramètre d'une fonction

```
1     double  sin(double);
2     double  sqrt(double);
      …
3     double integ( double(*adf)(double), int nb,
4                   double min, double max)
5     {
6        double delta((max-min)/nb), x(min), som(0);
7
8        for(int i(0); i<nb ; ++i, x += delta)
9           som += (*adf)(x) ;  // aussi ok avec adf(x)
10
11       return delta*som;
12    }
13    integ(sin,   100,  0.,  3.14);
14    integ(sqrt, 1000, 20., 25.  );
```



Notre usage: passer des fonctions en parametre à GTKmm

Une bibliothèque telle que GTKmm est une hiérarchie de classes qui sert à créer une interface graphique composée de **widgets**, par exemple des ***buttons***.

Avec GTKmm Il est possible d'associer une **réaction,** *sous forme d'une fonction*, au fait de cliquer la souris (ou un autre événement) sur un ***button*** en particulier.

1) Dans un phase d'initialisation, pour chaque ***button*** il suffit de *passer un nom de fonction* à une méthode dédiée.
   -> le *pointeur de fonction* est mémorisé dans un attribut de ***button*** dans GTKmm

2) Après l'initialisation, pendant la boucle infinie d'interaction, chaque fois qu'on clique sur le ***button*** la fonction associée au ***button*** est appelée à l'aide du pointeur de fonction mémorisé dans l'étape 1).