modificateur const

modificateur const

`int const j = 7;` la variable constante j,
elle ne peut pas changer de valeur

~~`j++ ;`~~ erreur de compilation

~~`j = 12;`~~ erreur de compilation

Cela ne veut pas dire que la modification de valeur est impossible, juste que on ne peut pas le faire avec j à gauche de l'affectation L = R ;

const veut dire au compilateur qu'il faut "aider" le programmeur à ne pas changer la valeur.

int *p;

p = &j; warning

*p = 25; peut provoquer l'erreur à l'exécution, cela dépend comment les const sont implémentées, sous linux et macOS ça passe mais

printf(" j= %d *p = %d\n", j , *p);

affiche sous linux (gcc) j = 25 *p = 25

mais sous macOS j = 9 *p = 25

modificateur const

```
int const *p;
```

p est un pointeur vers un int non-modifiable.
Le pointeur p n'est pas constant, c'est la valeur int pointé par p qui ne peut pas être modifiée.

```
int const i = 6;
p = &i;     --> OK, p pointe vers une valeur constante
```

```
int x = 6;
p = &x;      --> OK même si x n'est pas const
*p = 133;    -->erreur compil :
              --> read-only variable is not assignable
x = 133;          --> OK
```

donc on peut changer x mais pas en utilisant la variable p

modificateur const

```
int i = 99, k = 88;

int * const p = &i;
```

Le pointeur constant vers un `int modifiable`.

C'est le
pointeur p qui ne peut pas être modifié.
La valeur pointée par p est modifiable.

```
*p = 600 ;  OK
p  = &k;    NON

int const * const q = &i;  /* q pointeur constant vers une
                              valeur non-modifiable int */
```

Astuce: pour comprendre ce qui est constant il faut lire de droite à gauche.

const

On peut permuter const avec le type mais pas avec * :

```
int const a;      <-->      const int a;

int const *p;     <-->      const int *p;
```

Règle modifiée:

- si `const` est à gauche ou à droite d'un type alors const s'applique à ce type sinon
- `const` s'applique à `*` qui se trouve à gauche de `const`

Donc qu'est qui n'est pas modifié par `strcpy :`

```
char *strcpy(char *dst, const char *src)
```