

C : les constantes

Le mot clé const

- Le mot clé `const` interdit de modifier une valeur par l'intermédiaire de l'identificateur concerné.

Le mot clé const

- Le mot clé `const` interdit de modifier une valeur par l'intermédiaire de l'identificateur concerné.
- Dans une déclaration, `const` s'applique à ce qui se trouve immédiatement à sa gauche.

Le mot clé const

- Le mot clé `const` interdit de modifier une valeur par l'intermédiaire de l'identificateur concerné.
- Dans une déclaration, `const` s'applique à ce qui se trouve immédiatement à sa gauche.
- S'il n'y a rien à sa gauche, il s'applique à ce qui se trouve immédiatement à sa droite.

Le mot clé const

- Le mot clé `const` interdit de modifier une valeur par l'intermédiaire de l'identificateur concerné.
- Dans une déclaration, `const` s'applique à ce qui se trouve immédiatement à sa gauche.
- S'il n'y a rien à sa gauche, il s'applique à ce qui se trouve immédiatement à sa droite.
- Ainsi, `const int x = 3;` et `int const x = 3;` sont équivalents.

Pointeur vers un entier constant

Les déclarations `const int *p;` et `int const *p;` sont équivalentes.

Elles déclarent un pointeur vers un entier constant :

- `*p = 3;` est interdit : on ne peut pas modifier l'entier par l'intermédiaire de `p` ;

Pointeur vers un entier constant

Les déclarations `const int *p;` et `int const *p;` sont équivalentes.

Elles déclarent un pointeur vers un entier constant :

- `*p = 3;` est interdit : on ne peut pas modifier l'entier par l'intermédiaire de `p` ;
- `p = q;` est autorisé : le pointeur lui-même n'est pas constant.

Pointeur constant vers un entier

La déclaration `int * const p = &x;` définit un pointeur constant vers un entier modifiable.

- `*p = 3;` est autorisé : l'entier pointé n'est pas constant ;

On initialise donc généralement le pointeur dès sa déclaration.

Pointeur constant vers un entier

La déclaration `int * const p = &x;` définit un pointeur constant vers un entier modifiable.

- `*p = 3;` est autorisé : l'entier pointé n'est pas constant ;
- `p = q;` est interdit : le pointeur `p` est constant.

On initialise donc généralement le pointeur dès sa déclaration.

Pointeur constant vers un entier constant

La déclaration `const int * const p = &x;` définit un pointeur constant vers un entier constant.

- `*p = 3;` est interdit ;

Pour lire la déclaration, on part de `p` :

Pointeur constant vers un entier constant

La déclaration `const int * const p = &x;` définit un pointeur constant vers un entier constant.

- `*p = 3;` est interdit ;
- `p = q;` est également interdit.

Pour lire la déclaration, on part de `p` :

Pointeur constant vers un entier constant

La déclaration `const int * const p = &x;` définit un pointeur constant vers un entier constant.

- `*p = 3;` est interdit ;
- `p = q;` est également interdit.

Pour lire la déclaration, on part de `p` :

- 1 `p` est constant ;

Pointeur constant vers un entier constant

La déclaration `const int * const p = &x;` définit un pointeur constant vers un entier constant.

- `*p = 3;` est interdit ;
- `p = q;` est également interdit.

Pour lire la déclaration, on part de `p` :

- 1 `p` est constant ;
- 2 c'est un pointeur ;

Pointeur constant vers un entier constant

La déclaration `const int * const p = &x;` définit un pointeur constant vers un entier constant.

- `*p = 3;` est interdit ;
- `p = q;` est également interdit.

Pour lire la déclaration, on part de `p` :

- 1 `p` est constant ;
- 2 c'est un pointeur ;
- 3 il pointe vers un `int` constant.

Objet constant ou accès en lecture seule ?

Dans le code suivant, `x` n'est pas un objet constant :

```
1 int x = 2;  
2 const int *p = &x;
```

- `x = 3;` est autorisé ;

Le type de `p` indique seulement que `p` ne peut pas servir à modifier l'objet pointé.

Objet constant ou accès en lecture seule ?

Dans le code suivant, `x` n'est pas un objet constant :

```
1 int x = 2;  
2 const int *p = &x;
```

- `x = 3;` est autorisé;
- `*p = 3;` est interdit.

Le type de `p` indique seulement que `p` ne peut pas servir à modifier l'objet pointé.

Résumé

- `int *p` : le pointeur et l'entier pointé sont modifiables.

Résumé

- `int *p` : le pointeur et l'entier pointé sont modifiables.
- `const int *p` : le pointeur est modifiable, mais pas l'entier par l'intermédiaire de ce pointeur.

Résumé

- `int *p` : le pointeur et l'entier pointé sont modifiables.
- `const int *p` : le pointeur est modifiable, mais pas l'entier par l'intermédiaire de ce pointeur.
- `int * const p` : le pointeur n'est pas modifiable, mais l'entier pointé l'est.

Résumé

- `int *p` : le pointeur et l'entier pointé sont modifiables.
- `const int *p` : le pointeur est modifiable, mais pas l'entier par l'intermédiaire de ce pointeur.
- `int * const p` : le pointeur n'est pas modifiable, mais l'entier pointé l'est.
- `const int * const p` : ni le pointeur ni l'entier pointé par son intermédiaire ne sont modifiables.

Où sont stockés les objets constants ?

Le mot clé `const` ne détermine pas à lui seul la zone mémoire utilisée.

- Une constante locale est généralement placée dans la pile, comme les autres variables locales.

L'organisation exacte dépend donc de la durée de vie de l'objet, du compilateur, des options d'optimisation et du système.

Où sont stockés les objets constants ?

Le mot clé `const` ne détermine pas à lui seul la zone mémoire utilisée.

- Une constante locale est généralement placée dans la pile, comme les autres variables locales.
- Une constante globale ou déclarée `static` est généralement placée dans la section en lecture seule `.rodata`.

L'organisation exacte dépend donc de la durée de vie de l'objet, du compilateur, des options d'optimisation et du système.

Où sont stockés les objets constants ?

Le mot clé `const` ne détermine pas à lui seul la zone mémoire utilisée.

- Une constante locale est généralement placée dans la pile, comme les autres variables locales.
- Une constante globale ou déclarée `static` est généralement placée dans la section en lecture seule `.rodata`.
- Certaines constantes peuvent être conservées dans un registre, intégrées directement dans les instructions ou même supprimées par le compilateur.

L'organisation exacte dépend donc de la durée de vie de l'objet, du compilateur, des options d'optimisation et du système.

Où sont stockés les objets constants ?

Le mot clé `const` ne détermine pas à lui seul la zone mémoire utilisée.

- Une constante locale est généralement placée dans la pile, comme les autres variables locales.
- Une constante globale ou déclarée `static` est généralement placée dans la section en lecture seule `.rodata`.
- Certaines constantes peuvent être conservées dans un registre, intégrées directement dans les instructions ou même supprimées par le compilateur.
- La section `.text` contient principalement le code machine du programme.

L'organisation exacte dépend donc de la durée de vie de l'objet, du compilateur, des options d'optimisation et du système.