Plan

Macro et inline

Les Macros

- Un macro est créé à l'aide de la directive `#define`
- Indique au compilateur de remplacer une chaîne de jetons pour chaque occurrence d'un identificateur dans le fichier source
- Le remplacement est effectué à l'étape de preprocessing
- La syntaxe d'un macro :
  `#define macro(param[, param [...]]) définition`
  Exemple :
  `#define ADD(x,y) x+y`
  `#define MAX(x,y) ((x)>(y)?(x):(y))`

Macro et inline

Les Macros

```
#define ADD(x,y) x+y
#define MAX(x,y) ((x)>(y)?(x):(y))
```

Macro et inline

Les Macros

```
#define ADD(x,y) x+y
#define MAX(x,y) ((x)>(y)?(x):(y))
```

Les macros ne sont pas sécurisés ?

- `int x=ADD(a,b)*c;` $\rightarrow x = a + b * c = a + (b * c)$
  Solution : `#define ADD(x,y) (x+y)`

Macro et inline

Les Macros

```
#define ADD(x,y) x+y
#define MAX(x,y) ((x)>(y)?(x):(y))
```

Les macros ne sont pas sécurisés ?

- `int x=ADD(a,b)*c;` $\rightarrow x = a + b * c = a + (b * c)$
  Solution : `#define ADD(x,y) (x+y)`
- Risque d'avoir plusieurs appels d'une fonction (à éviter)
  `MAX(f(3),5)` $\rightarrow$ `((f(3))>(5)?(f(3)):(5)`
  Soient $x = 5$ et $y = 3$.
  `MAX(x++,y)` donne quels résultats ?
  Résultats attendus : MAX renvoie 5 et x=6

Macro et inline

Les Macros

```
#define ADD(x,y) x+y
#define MAX(x,y) ((x)>(y)?(x):(y))
```

Les macros ne sont pas sécurisés ?

- `int x=ADD(a,b)*c;` $\rightarrow x = a + b * c = a + (b * c)$
  Solution : `#define ADD(x,y) (x+y)`
- Risque d'avoir plusieurs appels d'une fonction (à éviter)
  `MAX(f(3),5)` $\rightarrow$ `((f(3))>(5)?(f(3)):(5)`
  Soient $x = 5$ et $y = 3$.
  `MAX(x++,y)` donne quels résultats ?
  Résultats attendus : MAX renvoie 5 et x=6
- Pas de contrôle sur les types : il s'agit d'une simple substitution
  `NimporteQuelType x,y;`
  `MAX(x,y)` est autorisé

Macro et inline

Les Macros

```
#define ADD(x,y) x+y
#define MAX(x,y) ((x)>(y)?(x):(y))
```

Les macros ne sont pas sécurisés ?

- `int x=ADD(a,b)*c;` $\rightarrow x = a + b * c = a + (b * c)$
  Solution : `#define ADD(x,y) (x+y)`
- Risque d'avoir plusieurs appels d'une fonction (à éviter)
  `MAX(f(3),5)` $\rightarrow$ `((f(3))>(5)?(f(3)):(5)`
  Soient $x = 5$ et $y = 3$.
  `MAX(x++,y)` donne quels résultats ?
  Résultats attendus : MAX renvoie 5 et x=6
- Pas de contrôle sur les types : il s'agit d'une simple substitution
  `NimporteQuelType x,y;`
  `MAX(x,y)` est autorisé
- Il n'est pas possible de déboguer un macro instruction par instruction

Macro et inline

Les Macros

```
#define ADD(x,y) x+y
#define MAX(x,y) ((x)>(y)?(x):(y))
```

Les macros ne sont pas sécurisés ?

- `int x=ADD(a,b)*c;` $\rightarrow x = a + b * c = a + (b * c)$
  Solution : `#define ADD(x,y) (x+y)`
- Risque d'avoir plusieurs appels d'une fonction (à éviter)
  `MAX(f(3),5)` $\rightarrow$ `((f(3))>(5)?(f(3)):(5)`
  Soient $x = 5$ et $y = 3$.
  `MAX(x++,y)` donne quels résultats ?
  Résultats attendus : MAX renvoie 5 et x=6
- Pas de contrôle sur les types : il s'agit d'une simple substitution
  `NimporteQuelType x,y;`
  `MAX(x,y)` est autorisé
- Il n'est pas possible de déboguer un macro instruction par instruction
- Les macros ne peuvent pas accèder aux attributs déclarés avec private ou protected

Macro et inline

Les fonctions inline

Le mot-clé inline est utilisé en C++ et s'applique à une fonction.
Remplacer chaque appel à la fonction inline par le corps de la fonction.

Macro et inline

Les fonctions inline

Le mot-clé inline est utilisé en C++ et s'applique à une fonction.
Remplacer chaque appel à la fonction inline par le corps de la fonction.

Les avantages des fonctions inline

- Les méthodes et les fonctions inline ont les même propriétés que les méthodes et les fonctions ordinaires (débogage, accès aux attributs private et protected, contrôle sur les types de paramètres, etc.)

Macro et inline

Les fonctions inline

Le mot-clé inline est utilisé en C++ et s'applique à une fonction.
Remplacer chaque appel à la fonction inline par le corps de la fonction.

Les avantages des fonctions inline

- Les méthodes et les fonctions inline ont les même propriétés que les méthodes et les fonctions ordinaires (débogage, accès aux attributs private et protected, contrôle sur les types de paramètres, etc.)
- Meilleure performance (pas de sauvegarde de IP, pas de `CALL`)

Les limites des fonctions inline

- Les fonctions ne sont pas toujours inlined
  - Les fonctions récursives
  - L'option -Os (optimisation de taille) du compilateur gcc/g++
    La plupart des fonctions sont inlined lorsqu'on utilise les options -O2 et -O3 du compilateur gcc/g++

Macro et inline

Les fonctions inline

Le mot-clé inline est utilisé en C++ et s'applique à une fonction.
Remplacer chaque appel à la fonction inline par le corps de la fonction.

Les avantages des fonctions inline

- Les méthodes et les fonctions inline ont les même propriétés que les méthodes et les fonctions ordinaires (débogage, accès aux attributs private et protected, contrôle sur les types de paramètres, etc.)
- Meilleure performance (pas de sauvegarde de IP, pas de `CALL`)

Les limites des fonctions inline

- Les fonctions ne sont pas toujours inlined
  - Les fonctions récursives
  - L'option -Os (optimisation de taille) du compilateur gcc/g++
    La plupart des fonctions sont inlined lorsqu'on utilise les options -O2 et -O3 du compilateur gcc/g++
- Parfois l'accélération est infime

Macro et inline

Les fonctions inline

Le mot-clé inline est utilisé en C++ et s'applique à une fonction.
Remplacer chaque appel à la fonction inline par le corps de la fonction.

Les avantages des fonctions inline

- Les méthodes et les fonctions inline ont les même propriétés que les méthodes et les fonctions ordinaires (débogage, accès aux attributs private et protected, contrôle sur les types de paramètres, etc.)
- Meilleure performance (pas de sauvegarde de IP, pas de `CALL`)

Les limites des fonctions inline

- Les fonctions ne sont pas toujours inlined
  - Les fonctions récursives
  - L'option -Os (optimisation de taille) du compilateur gcc/g++
    La plupart des fonctions sont inlined lorsqu'on utilise les options -O2 et -O3 du compilateur gcc/g++
- Parfois l'accélération est infime
- Peut augmenter sensiblement la taille de l'exécutable