
PROGRAMMATION CONCURRENTE
PrC
IX - LA GÉNÉRICITÉ (PAS À L'EXAMEN)
Fabrice.Kordon@lip6.fr
UPMC
SORBONNE UNIVERSITÉS

36 - La notion de macro

Programmation Concurrente - LI330
Université P. & M. Curie - année scolaire 2013/2014

PrC

La notion de macro

- Substituer une séquence de texte compliquée à une séquence plus simple
  - Premières utilisation en assembleur
    - Assimiler une séquence de code compliquée à quelque chose de plus simple
    - Usage de paramètres
  - C dispose de cette fonctionnalité via le pré-processeur

Exemple 1: redéfinition partielle de la syntaxe de C


```
/* Petit exemple sur l'usage de macro */
#include <stdio.h>

/* redefinition (limitee) du langage */
#define Si if (
#define Alors ){
#define FinSi }
#define Sinon } else {
#define SinonSi } else if (
```

```
int main ()
{
  int a;

  printf ("entrer une valeur\n");
  scanf ("%i", &a);
  printf ("vous avez saisi la
  valeur %i\n", a);


  Si a < 0 Alors
     printf ("a est negatif\n");
  SinonSi a == 0 Alors
     printf ("a est nul\n");
  Sinon
     printf ("a est positif\n");
  FinSi
}
```

Exemple 1: le code généré par le pré-processeur (soumis au


```
$ cpp ex-macro1.c

... le contenu de stdio.h ...

# 10 "ex-macro1.c"
int main ()
{
   int a;


   printf ("entrer une valeur\n");
   scanf ("%i", &a);
   printf ("vous avez saisi la
  valeur %i\n", a);
  if ( a < 0 ){
    printf ("a est negatif\n");
   } else if ( a == 0 ){
    printf ("a est nul\n");
   } else {
    printf ("a est positif\n");
   }
}
```

Exemple 2: macro avec paramètres

```
/* Autre petit exemple sur l'usage de macro */
#include <stdio.h>

/* macro avec des paramètres */
#define max(a,b)  (a)>(b)?(a):(b)

int main ()
{
  int a, b;

  printf ("entrer une valeur\n");
  scanf ("%i", &a);
  printf ("entrer une autre valeur\n");
  scanf ("%i", &b);
  printf ("%i est le max entre %i et %i\n", max (a,b), a, b);
}
```

Exemple 2: le code généré par le pré-processeur (soumis au


```
$ cpp ex-macro2.c
```

... le contenu de stdio.h ...

```
# 6 "ex-macro2.c"
int main ()
{
    int a, b;

    printf ("entrer une valeur\n");
    scanf ("%i", &a);
    printf ("entrer une autre valeur\n");
    scanf ("%i", &b);
    printf ("%i est le max entre %i et %i\n", (a)>(b)?(a):(b), a, b);
}
```

Observations sur les macros

- C'est un mécanisme extrêmement utile
  - Substitution de «séquences types»
  - Centralisation de modifications
    - Dimensionnement de constantes
    - Réécriture de séquences d'instructions
- Limites du système
  - Il est difficile de vérifier ce qui est substitué
    - Le compilateur travaille sur le code réécrit (c'est trop tard)
  - Il est impossible de donner un cadre strict aux substitutions
    - Spécifier qu'une constante doit être d'un type donné
    - Spécifier le prototype d'une fonction à remplacer
  - ... bref, peu de contrôle au niveau de l'expansion de macro-définitions