5 La récursivité

- Un programme récursif est tout programme qui se rappelle lui-même.
- le programme défini est utilisé pour se définir.
- un programme récursif est un programme qui fait une partie du travail et se rappelle ensuite pour terminer le reste.
- La récursivité est un moyen simple et élégant de résoudre certains problèmes de nature récurrente.
- **Remarque :** Toute boucle **pour** ou **tantque** peut être transformé en programme récursif.

Condition d'arrêt

- Puisque le programme récursif s'appelle lui-même, il est nécessaire de fournir une condition pour arrêter la récursivité, ce qui est le cas où le programme ne s'appelle pas
- Il est préférable de tester d'abord la condition d'arrêt, puis, si la condition n'est pas remplie, de rappeler le programme au fur et à mesure que l'appel conduit à la condition d'arrêt.
- **Exemple :**

```
void affiche (int i)
{
   printf("%d",i);
   affiche (i +1);
}
```

```
void affiche (int i)
{
   if (i<10) {
      printf("%d",i);
      affiche (i +1);
   }
}
```

La forme générale du programme récursif :

```
Procédure récursive
(paramètres)
début
   si (condition d'arrêt)
alors
      <instructions du
point d'arrêt>
   Sinon
      <instructions>
      Appel récursif
(paramètres changés)
      <Instructions>
   Fsi;
Fin.
```

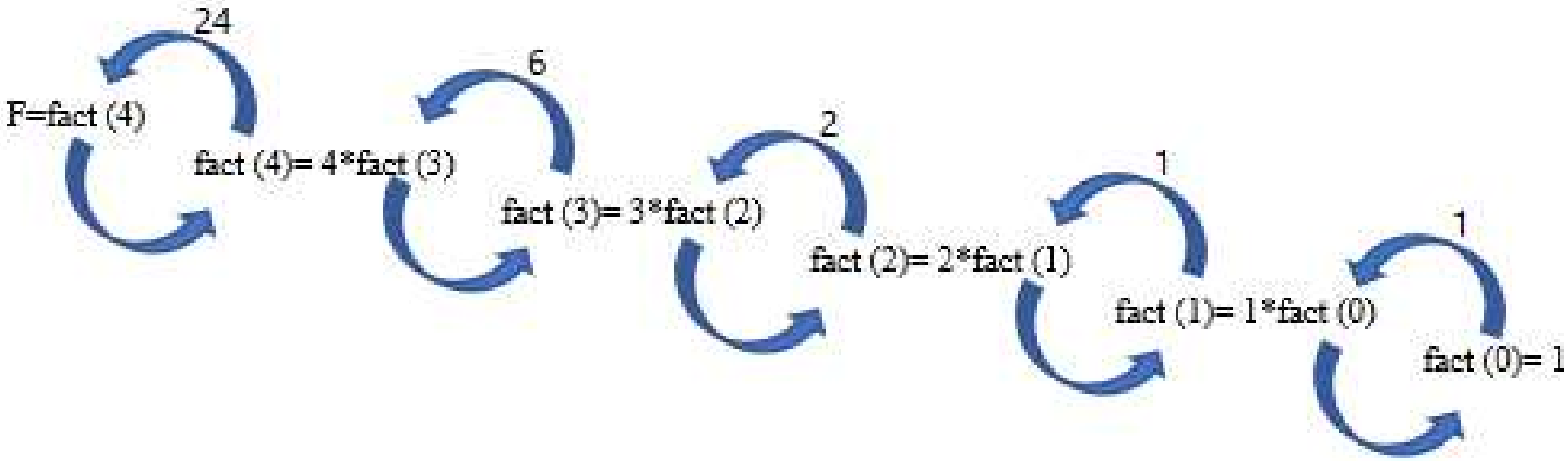
```
void recursive(paramètres)
{
   if (condition d'arrêt)
      <instructions du
point d'arrêt>
   else
   {
      <instructions>
      Appel récursif
(paramètres changés)
      <Instructions>
   }
}
```

Exemple

$$\text{fact}(n) = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot \text{fact}(n-1) & \text{if } n > 0 \end{cases}$$

```
Fonction fact (n : Entier) : Entier
début
    si (n = 0) alors
        fact←1
    sinon
        fact←n*fact (n-1)
    fsi ;
fin.
```

```
int fact (int n){
    if (n == 0) return 1;
    return n*fact(n-1);
}
```


24
F=fact (4)
fact (4)= 4*fact (3)
6
fact (3)= 3*fact (2)
2
fact (2)= 2*fact (1)
1
fact (1)= 1*fact (0)
1
fact (0)= 1

La pile d’exécution :

- Un emplacement en mémoire désigné pour contenir les paramètres et les variables locales et où le résultat est stocké pour chaque sous-programme en cours d'exécution.
- Habituellement, la programmation en mode récursif est plus facile et plus lisible, mais elle consomme beaucoup de mémoire, par exemple pour calculer 4! On réserve dans la pile une place pour mettre le résultat, une autre pour mettre le paramètre n=4, puis une autre place pour mettre le résultat de 3! Et le paramètre n = 3 et ainsi de suite jusqu'à ce que 0! soit calculé Le paramètre n=0 est supprimé, puis les paramètres et les résultats sont supprimés dans l'ordre inverse de celui dans lequel ils ont été créés.

La récursivité mutuelle

- un programme appelle un autre programme, qui à son tour appelle le premier programme.
- **Exemple**

```c
#include <stdio.h>

float f2(int n);

float f1(int n) {
  if (n <= 0) return 0;
  return 1. / n + f2(n - 2);
}

float f2(int n) {
  if (n <= 0) return 0;
  return -1. / n + f1(n - 2);
}

void main() {
  printf("%f\n", 4*f1(2*100+1) * 4);
}
```

La récursivité mutuelle

- **Remarque importante :** Comme la fonction f1 appelle la fonction f2 qui n'est pas encore définie en langage C, l'en-tête de la fonction f2 doit être ajouté sans son corps (la première ligne) avant de définir la fonction f1, sachant que sa définition vienne après .