C 4 : Boucles indéfinies et ruptures de boucle

Juliusz Chroboczek

11 octobre 2023

1 Boucles indéfinies

Une *boucle indéfinie*, ou *boucle while*, est une boucle dont l'exécution est contrôlée par une expression booléenne. À la différence d'une boucle définie, il n'est pas possible de prévoir à l'avance le nombre de fois que le corps d'une boucle indéfinie sera exécuté.

1.1 Boucle `while`

On appelle *racine carrée entière* d'un entier positif k le plus grand entier r tel que $r^2 \leq k$. Pour calculer la racine carrée entière de k, il suffit de trouver le plus petit entier r' tel que $r'^2 > k$, et alors $r = r' - 1$.

```
int k, rp, r;
scanf("%d", &k);
rp = 0;
while(rp * rp <= k)
    rp++;
r = rp - 1;
printf("%d\n", r);
```

Une boucle while est composée du mot-clé `while` suivi d'une expression entre parenthèses, tout cela suivi d'une instruction (qui est souvent un bloc).

`while(`*e*`)` *corps*

L'exécution procède comme suit :

1. l'expression *e* est évaluée ; si elle est fausse, on sort de la boucle ;
2. le corps de la boucle est exécuté ;
3. on recommence à l'étape 1.

1.2 Boucle `do-while`

```
int n;
do {
    printf("Devine un nombre !\n");
    scanf("%d", &n);
} while(n != 42);
printf("Gagné!\n");
```

La boucle *do-while* est analogue à la boucle *while*, mais l'expression de contrôle se trouve à la fin de la boucle ; le corps est donc exécuté au moins une fois. Elle est utile lorsque l'expression de contrôle n'a pas de sens avant la première exécution du corps.

1.3 Expressivité des boucles

Les boucles indéfinies sont strictement plus expressives que les boucles définies : toute boucle définie peut s'exprimer comme une boucle indéfinie, mais l'inverse n'est pas vrai. Par exemple, les deux boucles suivantes sont complètement équivalentes :

```
int i;
for(i = 0; i < 100; i++)
    printf("%d\n", i);
```

```
int i;
i = 0;
while(i < 100; i++) {
    printf("%d\n", i);
    i++;
}
```

Par souci de parcimonie, on utilisera toujours des boucles définies lorsqu'elles suffisent, c'est à dire lorsque le nombre d'itérations est connu *a priori*.

2 Ruptures de boucle

Il est parfois nécessaire d'interrompre l'exécution d'une boucle de façon prématurée, par exemple lors d'une situation exceptionnelle (une « erreur »).

2.1 Terminaison prématurée à l'aide de `return`

```
int
main()
{
    int i;
    double x, somme;
    somme = 0.0;
    for(i = 0; i < 4; i = i + 1) {
        scanf("%lf", &x);
        if(x > -1.0E10 && x < 1.0E-10) {
            printf("Division par zéro !\n");
```

```
            return 1;
        }
        somme = somme + 1.0 / x;
    }
    printf("La somme des %d inverses est %lf.\n", i, somme);
    return 0;
}
```

Dans l'exemple ci-dessus, dès lors qu'apparaît sur l'entrée un nombre à la valeur absolue trop basse, un message d'erreur est imprimé et la boucle est interrompue par l'exécution de l'instruction `return`, qui cause la terminaison de l'exécution de la fonction `main`.

2.2 Terminaison prématurée à l'aide de `break`

```
int
main()
{
    int n, i;
    double x, somme;

    scanf("%d", &n);
    somme = 0.0;
    for(i = 0; i < n; i = i + 1) {
        scanf("%lf", &x);
        if(x > -1.0E10 && x < 1.0E-10) {
            printf("Division par zéro !\n");
            break;
        }
        somme = somme + 1.0 / x;
    }
    printf("La somme des %d inverses est %lf\n", i, somme);
    return 0;
}
```

On peut interrompre une boucle de façon explicite à l'aide de l'instruction `break` qui a l'effet de terminer immédiatement l'exécution de la boucle la plus interne dans laquelle elle se trouve. À la différence du cas précédent, le code qui suit le corps de la boucle s'exécute normalement.

2.3 Sortie de plusieurs boucles

En C, il n'est pas possible d'utiliser `break` pour sortir de plusieurs boucles imbriquées (c'est possible par exemple en Perl ou en Go). Pour interrompre plusieurs boucles imbriquées, on peut :

- utiliser un `return` ; ou
- utiliser un *flag* qui contôle l'exécution de la boucle ; ou
- utiliser une instruction `goto`.