Cours 2 : Les boucles for

Guillaume Aubian

Ce document a été grandement inspiré par – et largement copié sur – le document correspondant du responsable précédent de ce cours, Juliusz Chroboczek, avec son accord.

1 Syntaxe et sémantique des boucles définies

Le fragment de code suivant exécute les deux instructions `printf` pour toutes les valeurs de i allant de 1 à 10:

```
int i;
for(i = 1; i <= 10; i = i + 1) {
    printf("J'ai collé %d timbres.\n", i);
    printf("Il m'en reste %d.\n", 10 - i);
}
```

Une boucle `for` est composée du mot-clé `for` suivi de trois expressions entre parenthèses, séparées par des points-virgule, puis d'un bloc (des déclarations et des instructions entre accolades) :

```
for(e1; e2; e3) { bloc }
```

L'exécution de la boucle procède de la façon suivante :

1. l'expression e_1, dite initialisation, est exécutée ;
2. le test e_2 est évalué ; s'il est faux, on sort de la boucle, et l'exécution de la boucle est terminée ;
3. le corps de la boucle est exécuté ;
4. l'expression e_3 est exécutée ;
5. on recommence à l'étape 2.

Dans 95% des cas, e_1 assigne une valeur initiale à une variable, e_2 va vérifier que cette variable n'est pas trop grande, et e_3 va l'incrémenter. Quelques raccourcis existent donc : `i += 5` équivaut à `i = i + 5`, `i -= 5` équivaut à `i = i - 5`, `i++` équivaut à `i += 1` et `i--` équivaut à `i -= 1`.

2 Quelques exemples

```
int i;
for(i = 0; i < 10; i = i + 1)
    printf("%d est pair.\n", 2 * i);
```

peut aussi s'écrire :

```
int i;
for(i = 0; i < 20; i = i + 2)
    printf("%d est pair.\n", i);
```

Lorsque le pas est négatif, il faut inverser le sens du test de terminaison :

```
int i;
for(i = 9; i >= 0; i = i - 1)
    printf("Il me reste %d timbres.\n", i);
```