Exercice 1 : Série 4 — Boucles

(a) Considérons le cas d'un système de vision industrielle qui inspecte des pièces fabriquées au sein d'une ligne d'assemblage. Le programme de ce système de vision comporte certaines variables internes permettant de mémoriser le décompte des pièces analysées. Ces variables sont les suivantes :

```
int nb_parts = 2000;
int nb_parts_bad = 200;
double percent_good = (nb_parts - nb_parts_bad) / nb_parts;
```

i. Quel résultats espérait le développeur ?

> Avec 200 pièces mauvaises sur 2000 pièces produites, le programmeur attendait un résultat de 0.9, signifiant 90

ii. Qu'obtient-il dans la pratique ?

> 0.0

iii. Expliquez les défaut constaté ?

> Le calcul effectue la division entière 1800 / 2000, qui donne comme résultat 0. Ensuite, le résultat est transformé en float et donne 0.0

iv. Que corriger pour obtenir un résultat correct ?

> ```
> int nb_parts = 2000;
> int nb_parts_bad = 200;
> double percent_good = (double)(nb_parts - nb_parts_bad) / nb_parts;
> ```

(b) Indiquer si les affirmations suivantes sont justes ou fausses. Dans les cas où elles sont fausses, expliquer ce qui serait correct pour une instruction `do..while` :

i. Les instructions de la boucle sont toujours exécutées au moins une fois.

> Correct

ii. Comme un mot réservé spécifique commence et termine la boucle, on n'a pas besoin de créer un bloc lorsque l'on a plusieurs instructions.

> Incorrect, il faut utiliser un bloc s'il y a plusieurs instructions dans la boucle.

iii. La condition se trouvant en fin de boucle, on sort de la boucle lorsque la condition est vraie.

> Incorrect, on sort de la boucle lorsque la condition est fausse.

iv. Le type de la condition peut être char.

> Correct

v. Les instructions de la boucle ne peuvent pas être une autre boucle `do..while`.

> Incorrect. Une boucle do while peut se trouver à l'intérieur d'une boucle `do..while`.

(c) **i.** Écrire un programme qui affiche les nombres entiers de 1 à 100 en employant une boucle `for`.

```c
#include <stdio.h>

int main() {
    for (int i = 1; i <= 100; i++)
        printf("%d\n", i);
}
```

ii. Écrire ce même programme en utilisant une boucle `while`.

```c
#include <stdio.h>

int main() {
    int i = 1;
    while (i <= 100)
        printf("%d\n", i++);
}
```

iii. Même question avec une boucle `do..while`.

```c
#include <stdio.h>

int main() {
    int i = 1;
    do {
        printf("%d\n", i++);
    } while (i <= 100);
}
```

iv. Parmi ces trois implémentations laquelle est la plus adaptée au problème ?

Lorsque le nombre d'itérations est connu à l'avance, la boucle `for` est la plus adaptée.

(d) Boucles `for` : renseignez ce qu'affiche ces boucles

i.
```c
for (i = 'a'; i < 'd'; printf("%i-", ++i));
```

98-99-100-

ii.
```c
for (i = 'a'; i < 'd'; printf("%c-", ++i));
```

b-c-d-

iii.
```c
for (i = 'a'; i++ < 'd'; printf("%c-", i ));
```

b-c-d

iv.
```c
for (i = 'a'; i <= 'a' + 25; printf("%c-", i++ ));
```

a-b-c-d-...z-

v.
```
for (i = 1 / 3; i ; printf("%i\n", i++ ));
```

(rien)

vi.
```
for (i = 0; i != 1  ; printf("%i", i += 1 / 3 ));
```

0000 ...

vii.
```
for (i = 12, k = 1; k++ < 5 ; printf("%i ", i-- ));
```

12 11 10 9

(e) Boucles `while` : renseignez ce qu'affiche ces boucles

i.
```
int i = 0;
while (i - 10) { i += 2; printf ( "%i.", i ); }
```

2.4.6.8.10.

ii.
```
int i = 0;
while (i - 10)
i += 2; printf("%i-", i);
```

10-

iii.
```
short i = 0;
while ( i < 11 ) { i += 2; printf("%i-", i); }
```

2-4-6-8-10-12-

iv.
```
for (i = 'a'; i <= 'a' + 25; printf("%c-", i++ ));
```

a-b-c-d-...z-

v.
```
char i = 11;
while (i--) { printf("%i/", i--); }
```

10/8/6/4/2/0/-2/-4/-6/-8 ... (à l'infini)

vi.
```
long i = 12l;
while (i--) { printf("%i%%", --i); }
```

10

vii.
```
int i = 0;
while (i++ < 10) { printf("%i", i--); }
```

1111111111 ... (à l'infini)

viii.

```
int i = 1;
while (i <= 5) { printf("%i/", 2 * i++); }
```

> 2/4/6/8/10/

ix.

```
int i = 1;
while (i != 9) { printf ( "%i-", i = i + 2 ); }
```

> 3-5-7-9-

x.

```
int i = 1;
while ( i < 9 ) { printf ( "%i", i += 2 ); break; }
```

> 3