Feuille d'exercices n°2 - Autour des boucles `while`

Notions abordées

- syntaxe des boucles `while`
- simulation de boucle
- variant et invariant de boucle
- rédaction de preuve de correction et de terminaison

Fonctions mystères

Exercice 1 Mystère simple

On considère la fonction `mystere_simple` définie par le code suivant.

```
int mystere_simple(int a, int b){
  //hyp :
  //...

  int c = 1;
  int d = 0;

  //invariant :
  while(c <= b){
    d = d + a;
    c = c + 1;
  }
  //ici....

  return d;
}
```

Question 1 ✍
Effectuer la simulation de boucle pour l'appel `mystere_simple(2,3)`, puis, si besoin pour y voir bien clair, celui pour `mystere_simple(4,5)`.

Question 2 ✍
Proposer une description de la fonction `mystere_simple`. Préciser ses hypothèses et sa description.

Question 3 ✍
Proposer un jeu de test pour la fonction `mystere_simple`.

Question 4 ✍
Montrer la correction et la terminaison de la fonction `mystere_simple`

Exercice 2 Mystère double

On considère la fonction `mystere_double` définie par le code suivant.

```c
void mystere_double(int n){
  //hyp : ...
  //....

  int i = 0;
  int a = 1;
  int j, b , c;

  //...
  while (i < n){
    j = 0;
    b = 1;
    c = a;

    //....
    while (j < i){
      printf("%d  ",a/(b*c));
      c = c/(i-j);
      j = j + 1;
      b = b*j;
    }

    // ici on a ...
    printf("%d  ",a/(b*c));
    printf("\n");
    i = i + 1;
    a = a*i;

  }
}
```

Question 1 ✍
Dans le code précédent, repérer quelles variables jouent le rôle de variable de boucle, et dans quel intervalle elles varient. Pour les autres, regarder leur valeur initiales, et comment elles sont modifiées à chaque étape. En déduire ce qu'elles représentent, et formuler des invariants de boucle.

Question 2 ✍
Que fait la fonction `mystere_double` ? Proposer une description de cette fonction. Proposer quelques simulations de boucles appuyant cette proposition. Vérifier dans ces simulations que les invariants proposés semblent corrects

Question 3 ✍
Démontrer que la fonction `mystere_double` termine.

Question 4 ✍
Démontrer que la fonction `mystere_double` est correcte (au sens de la description proposée).

Question 5 ✍
Que pouvez-vous dire de l'instruction d'affichage ligne 24 ?

Exponentiation rapide

On remarque que pour un réel $a \in \mathbb{R}$ et un entier naturel $p \in \mathbb{N}^*$, on a $a^p = \begin{cases} a^{\frac{p}{2}} & \text{si } p \text{ est pair} \\ a \times a^{\frac{p}{2}} & \text{si } p \text{ est impair} \end{cases}$
où $\frac{p}{2}$ désigne le quotient de p par 2, (*i.e.* $\lfloor \frac{p}{2} \rfloor$).
On propose donc l'algorithme (informel) suivant pour l'exponentiation (*i.e.* le passage à la puissance).

> **Pour calculer** a^pc :
> - Si $p = 0$, retourner 1.
> - Si p est pair, calculer $a^{\frac{p}{2}}$ puis l'élever au carré. Retourner le résultat obtenu.
> - Si p est impair, calculer $a^{\frac{p}{2}}$, l'élever au carré puis le multiplier par a. Retourner le résultat obtenu.

Question 1 ✍
Combien faut-il de multiplications pour calculer a^p avec la méthode naïve ? Donc si $p = 2^k$ pour $k \in \mathbb{N}$, combien faut-il de multiplications ?
Toujours pour $p = 2^k$ avec $k \in \mathbb{N}$, combien faut-il de multiplications avec l'algorithme ci-dessus.

Question 2 ✍
On souhaite étudier une version **itérative** de l'algorithme proposé ci-dessus. Donner le code en C d'une fonction nommée `puissance_rapide` qui implémente cet algorithme avec une boucle `while`.

Question 3 ✍
Si ce n'est déjà fait, indiquer dans le code un variant et un invariant de boucle pertinents.

Question 4 ✍
Démontrer que la fonction `puissance_rapide` termine.

Question 5 ✍
Démontrer que la fonction `puissance_rapide` est correcte.

Calcul du PGCD

Propriété 1

Soit $(a,b)\in\mathbb{Z}^*\times\mathbb{Z}^*$. *Soit* $(q,r)\in\mathbb{Z}^2$.
Si $a=bq+r$, ***alors*** $\text{PGCD}(a,b)=\text{PGCD}(b,r)$ *si* $r\neq 0$, *et* $\text{PGCD}(a,b)=|b|$ *si* $r=0$.

Exercice 3 Avec l'algorithme d'Euclide

Question 1 ✍
Utiliser l'algorithme d'Euclide pour calculer $\text{PGCD}(48,35)$ puis $\text{PGCD}(48,36)$.

Question 2 ✍
Définir une fonction `pgcd_euclide` qui implémente l'algorithme d'Euclide en C. Préciser ses hypothèses et sa description.

Question 3 ✍
Proposer un jeu de tests pour la fonction `pgcd_euclide`

Question 4 ✍
Après avoir identifié un variant de boucle, démontrer que la fonction `pgcd_euclide` termine.

Question 5 ✍
Après avoir identifié un invariant de boucle, montrer que la fonction `pgcd_euclide` est correcte.

Exercice 4 Avec un autre algorithme proche

On la fonction `pgcd_bis` définie par le code suivant

```c
int pgcd_bis(int a, int b){
//hyp: a>0 and b>0
//retourne le pgcd de a et b
  int c = a;
  int d = b;
  int temp;

  if( d > c ){
    c = b;
    d = a;
  }

  //...
  while( d > 0 ){
    c = c - d;
    if( d > c ){
      temp = c;
      c = d;
      d = temp;
    }
  }

  return c;
}
```

Question 1 ✍
Faire la simulation des appels `pgcd_bis(48,35)` et `pgcd_bis(48,36)`.

Question 2 ✍
Proposer un jeu de tests pour la fonction `pgcd_bis`

Question 3 ✍
Après avoir identifié un variant de boucle, démontrer que la fonction `pgcd_bis` termine.

Question 4 ✍
Après avoir identifié un invariant de boucle, montrer que la fonction `pgcd_bis` est correcte.