Cours MALG & MOVEX

Modélisation, spécification et vérification (II)

Dominique Méry
Telecom Nancy, Université de Lorraine

Année universitaire 2024-2025

Plan

1. Summation of the n first integers

Sommaire

Current Summary

1. Summation of the n first integers

Annotation versus commentaire de programmes ou d'algorithmes

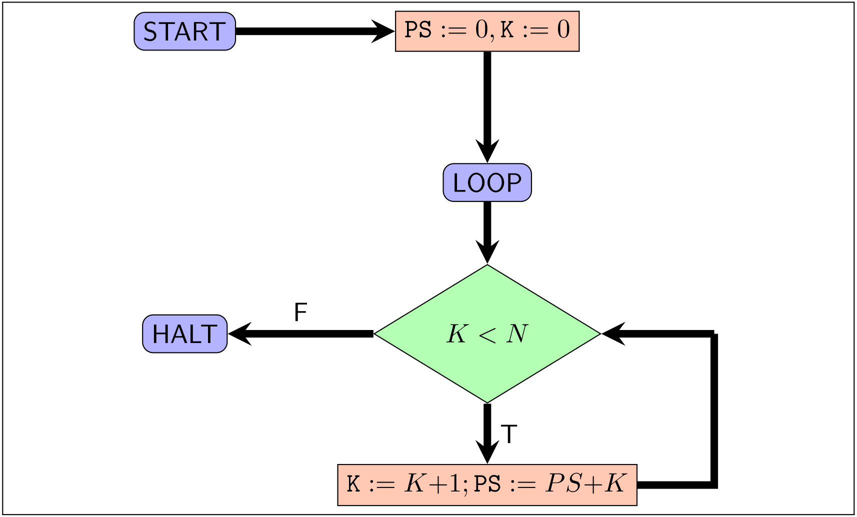
- Un programme ou un algorithme peuvent être annotés ou commentés.
- Un commentaire est une information pertinente destinée à être vue ou lue et qui a une importance relative dans l'esprit du concepteur.
- Un commentaire indique une information sur les données, sur les variables et donc sur l'état supposé du programme à l'exécution.
- Un commentaire est une annotation du texte du code qui nous permet de communiquer une information sémantique :
  - *à ce point, la variable k est plus petite sur n*
  - *l'indice e fait référence à une adresse licite de t et cette valeur est toujours positive*
  - *la somme des variables est positive*
- Les annotations peuvent être systématisées et obéir à une syntaxe spécifique définissant le langage d'annotations ;

```
/*@ assert l1: z >= 3 && y == 3; */
z = z +y;
/*@ assert l2: z >= 6 && y == 3; */
```

Calculer la somme des n premiers entiers (v0)

```c
int fS(int n) {
  int ps = 0;
  int k = 0;
  while (k < n) {
    k = k + 1;
    ps = ps + k;
  };
  return ps;
}
```

Calculer la somme des n premiers entiers (flowchart)



Calculer la somme des n premiers entiers (v0)

```
// pre n>=0;
// post ps == n*(n+1) / 2;

int fS(int n) {
  int ps = 0;
  int k = 0;
  while (k < n) {
    k = k + 1;
    ps = ps + k;
  };
  return ps;
}


int main()
{

}
```

Calculer la somme des n premiers entiers (v0)

```
// pre n>=0;
// post ps == n*(n+1) / 2;

int fS(int n) {
  int ps = 0;
  int k = 0;
  while (k < n) {
    // ps = k*(k+1) / 2;
    k = k + 1;
    ps = ps + k;
  };
    // ps = n*(n+1) / 2;
  return ps;
}


int main()
{

}
```

Calculer la somme des n premiers entiers (v0)

```
#include <stdio.h>

int fS(int n) {
  int ps = 0;
  int k = 0;
  while (k < n) {
    k = k + 1;
    ps = ps + k;

  };
  return ps;
}

int main()
{
  int   z = 3;
  printf("Value for z=%d is %d\n",z,fS(z));
  return 0;
}
```

Problème du calcul de la somme des n premiers entiers

Definition of S(n) and IS(n)

$\forall n \in \mathbb{N} : S(n) = \sum_{k=0}^{n} k$

$$\left[\begin{array}{l} IS(0) = 0 \\ n \geq 0, IS(n+1) = IS(n)+(n+1) \end{array}\right.$$

Property for S(n) and IS(n)

$\forall n \in \mathbb{N} : S(n) = IS(n)$

- base 0 : $S(0) = 0 = IS(0)$
- induction $i+1$: $\forall j \in 0..i : S(j) = IS(j)$
  - $S(i+1) = \sum_{k=0}^{i+1} k$ (definition)
  - $S(i+1) = (\sum_{k=0}^{i} k)+i+1$ (property of summation)
  - $S(i+1) = S(i)+(i+1)$ (by definition of S(i))
  - $S(i+1) = IS(i)+(i+1)$ (by inductive assumption on S(i) et IS(j))
  - $S(i+1) = IS(i)+(i+1) = IS(i+1)$ (by defintion of IS(i+1))
- conclusion : $\forall i \in \mathbb{N} : S(i) = IS(i)$ (by induction principle)

Reformulation algorithmique du calcul de la somme des n premiers entiers

- $\forall n \in \mathbb{N} : S(n) = \sum_{k=0}^{n} k$
- $\left[\begin{array}{l} IS(0) = 0 \\ n \geq 0, IS(n+1) = IS(n)+n \end{array} \right.$
- $\forall n \in \mathbb{N} : S(n) = IS(n)$
-
  - base 0 : $S(0) = 0$
  - induction $k+1$: $S(k+1) = S(k)+i+1$
  - step $k+1$: $S(k+1) = S(k)+k+1$
- $S(k) = ps$: current value of ps is S(k)
- $S(k-1) = ops$: current value of ops is $S(k-1)$
- step $k+1$: $ps = ops+k+1$

Calculer la somme des n premiers entiers (v1)

```
#include <stdio.h>

int fS(int n) {
  int ps = 0;
  int k = 0;
  int ok=k, ops = 0;
  while (k < n) {
    ok=k;ops=ps;
    k = ok + 1;
    ps = ops + k;
  };
  return ps;
}

int main()
{
  int   z = 3;
  printf("Value for z=%d is %d\n",z,fS(z));
  return 0;
}
```

Calculer la somme des n premiers entiers (v2)

```
int fS(int n) {
  int ps = 0;
  int k = 0;
  int ok=k, ops = 0;
  while (k < n) {
 /*@ assert 0 <= k && k <= n
   && ps == S(k) && ops == S(ok);     */
     ok=k;ops=ps;
     k = ok + 1;
     ps = ops + k;
 /*@ assert 0 <= k && k <= n && ps == S(k)
   && ops == S(ok);     */
  };
  return ps;
}
```

Calculer la somme des n premiers entiers

```
/*@ axiomatic S {
  @ logic integer S(integer n);
  @ axiom S_0: S(0) == 0;
  @ axiom S_i: \forall integer i; i > 0 ==> S(i) == S(i-1)+i;
  @ } */

/*@ requires n >= 0;
   assigns \nothing ;
   ensures \result == S(n);
*/
int fS(int n) {
  int ps = 0;
  int k = 0;
  int ok=k,ops=ps;
  /*@ loop invariant 0 <= k && k <= n && ps == S(k) && ops == S(ok) ;
    loop assigns ps, k,ops,ok;
   */
  while (k < n) {
/*@ assert l0: 0 <= k && k <= n && ps == S(k) && ops == S(ok);    */
    ops=ps;ok=k;
     k = ok + 1;
    ps = ops + k;
 /*@ assert l1:  0 <= k && k <= n && ps == S(k) && ops == S(ok);    */
  };
 /*@ assert  ps == S(n);    */
  return ps;
}
```

Observations sur le calcul

- Définition des fonctions mathématiques nécessaires pour exprimer le calcul de la somme des n premiers nombres entiers.
- Expression des résultats intermédiaires appelés *sommes partielles*
- Relation entre la preuve par induction et la forme du corps de l'itération.
- Induction et calcul sont liés.

Observations sur le calcul

- Définition des fonctions mathématiques nécessaires pour exprimer le calcul de la somme des n premiers nombres entiers.
- Expression des résultats intermédiaires appelés *sommes partielles*
- Relation entre la preuve par induction et la forme du corps de l'itération.
- Induction et calcul sont liés.

$$x_0 \xrightarrow{P} x \tag{1}$$

$$x_0 \xrightarrow{\star} x \tag{2}$$

$$x_0 \longrightarrow x_1 \longrightarrow \ldots \longrightarrow x \tag{3}$$

$$x_0 \longrightarrow x_1 \longrightarrow \ldots \longrightarrow x_n \xrightarrow{step} x \tag{4}$$