Imperative programming in Dafny

Exercise: Loops  
Will Sonnex and Sophia Drossopoulou

The following exercises are about simple loops and loop invariants. We will develop invariants for simple loops, nested loops, triple nested loops, and consecutive loops.

The Dafny code can be found in the file Loops\_For\_Calculations.dfy.

## Question 1: GCD – one simple loop

Consider the following Dafny specification and code which calculate the greatest common divisor of two numbers. The function gcd defines the grates common divisor, while the method GcdCal calculates the greatest common divisor. Annotate the method body with appropriate pre-, post- conditions and invariants in order to verify the code.

function gcd(m: nat, n: nat): nat

requires m>=1;

requires n>=1;

decreases m+n;

{

if (n==m) then n

else if (m>n) then gcd(m-n, n)

else gcd(m, n-m)

}

method GcdCal(m: nat, n: nat) returns (res: nat)

requires ???

ensures res == gcd(m, n);

{

var m1: nat, n1: nat := m, n;

while (m1!=n1)

invariant ???

decreases ???

{

if (m1>n1) {

m1 := m1-n1;

}

else {

n1 := n1-m1;

}

}

return m1;

}

## Question 2: Product - Nested Loops

Consider the following Dafny specification and code which calculates the product of two numbers using only increment by one, decrement by one, and comparison with zero. Annotate the code so that it checks.

method CalcProduct(m: nat, n: nat) returns (res: nat)

ensures res == m\*n;

{

var m1: nat := m;

res := 0;

while (m1!=0)

invariant ???

decreases ???

{

var n1: nat := n;

while (n1!=0)

invariant ???

decreases ???

{

res := res+1;

n1 := n1-1;

}

m1 := m1-1;

}

}

## Question 3: Product – Triply Nested Loops

Consider the following Dafny variation of the previous exercise. The code calculates the product of three numbers using only increment by one, decrement by one, and comparison with zero. The code has three nested loops. Annotate the code so that it checks.

method CalcThreeProduct(m: nat, n: nat, p: nat)

returns (res: nat)

ensures res == m\*n\*p;

{

var m1: nat := 0;

res := 0;

while (m1!=m)

invariant ???

decreases ???

{

var n1: nat := 0;

while (n1!=n)

invariant ???

decreases ???

{

var p1: nat := 0;

while (p1!=p)

invariant ???

decreases ???

{

res := res+1;

p1 := p1+1;

}

n1 := n1+1;

}

m1:= m1+1;

}

}

## Question 4: Terms - Consecutive Loops

Consider the following Dafny specification and code which calculates the term 3\*m-2\*n using only increments by three and by two, decrement by one, comparison with zero, and negatives. Annotate the code so that it checks.

function method abs(m: int): nat

{ if m>0 then m else -m }

method CalcTerm(m: int, n: nat) returns (res: int)

ensures res == 3\*m-2\*n;

{

var m1: nat := abs(m);

var n1: nat := n;

res := 0;

while (m1!=0)

invariant ???

decreases ???

{

res := res+3;

m1 := m1-1;

}

if (m<0) { res := -res; }

while (n1!=0)

invariant ???

decreases ???

{

res := res-2;

n1 := n1-1;

}

}

## Sample Answers

In the file Loops\_For\_Calculations\_SA.dfy