# Proving a sum formula by induction

- **Problem:** Prove that
$$\sum_{i=1}^{n} i = 1 + 2 + \cdots + n = \frac{n(n+1)}{2}$$
- **Solution:** Define the predicate $P(n)$ as the equation above and use induction:
  - **Basis:** $P(1)$ holds because for $n = 1$ we have $\sum_{i=1}^{n} i = \sum_{i=1}^{1} i = 1 = \frac{1(1+1)}{2} = \frac{n(n+1)}{2}$
  - **Induction:**
    - **Inductive hypothesis:** Assume $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
    - **Inductive step:** Wish to prove að $\sum_{i=1}^{n+1} i = \frac{(n+1)(n+2)}{2}$.

# Proving a sum formula by induction

- **Problem:** Prove that
$$\sum_{i=1}^{n} i = 1 + 2 + \cdots + n = \frac{n(n+1)}{2}$$
- **Solution:** Define the predicate $P(n)$ as the equation above and use induction:
  - **Basis:** $P(1)$ holds because for $n = 1$ we have $\sum_{i=1}^{n} i = \sum_{i=1}^{1} i = 1 = \frac{1(1+1)}{2} = \frac{n(n+1)}{2}$
  - **Inducton:**
    - **Inductive hypothesis:** Assume $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
    - **Inductive step:** Wish to prove að $\sum_{i=1}^{n+1} i = \frac{(n+1)(n+2)}{2}$.
    $\sum_{i=1}^{n+1} i = \sum_{i=1}^{n} i + (n+1) \overset{\text{By inductive hypothesis}}{=} \frac{n(n+1)}{2} + (n+1) = \frac{n^2+n+2n+2}{2} = \frac{n^2+3n+2}{2} = \frac{(n+1)(n+2)}{2}$ sem er það sem sanna þurfti.

# General while-loop

Sauðakóði

meðan $C$
    S

Pseudocode

while $C$
    S

Java

```
while( C )
{
    S
}
```

# General while-loop

$\boxed{\{\ F\ \}}$

while $C$

$\boxed{\{\ I\ \}}$

S

$\boxed{\{\ E\ \}}$

The precondition of the loop describes the state just before the loop starts

The loop invariant makes it possible to verify the loop

The postcondition of the loop describes the state just after the loop finishes

# General while-loop

$\{ F \}$

while $C$

$\{ I \}$

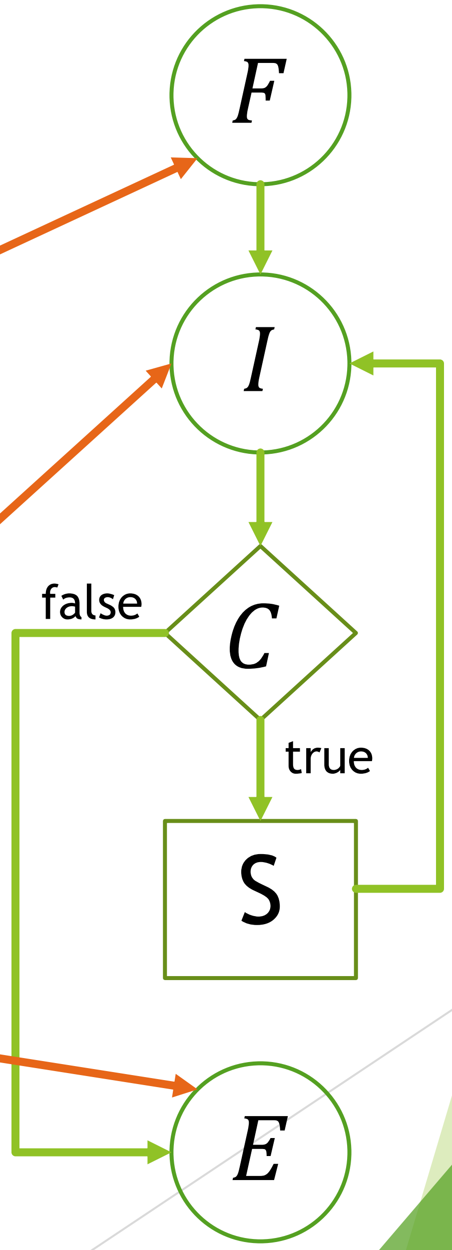
S

$\{ E \}$

The precondition of the loop describes the state just before the loop starts

The loop invariant makes it possible to verify the loop

The postcondition of the loop describes the state just after the loop finishes



# General while-loop

$\{\ F\ \}$

while $C$

$\{\ I\ \}$

S

$\{\ E\ \}$

The precondition of the loop describes the state just before the loop starts

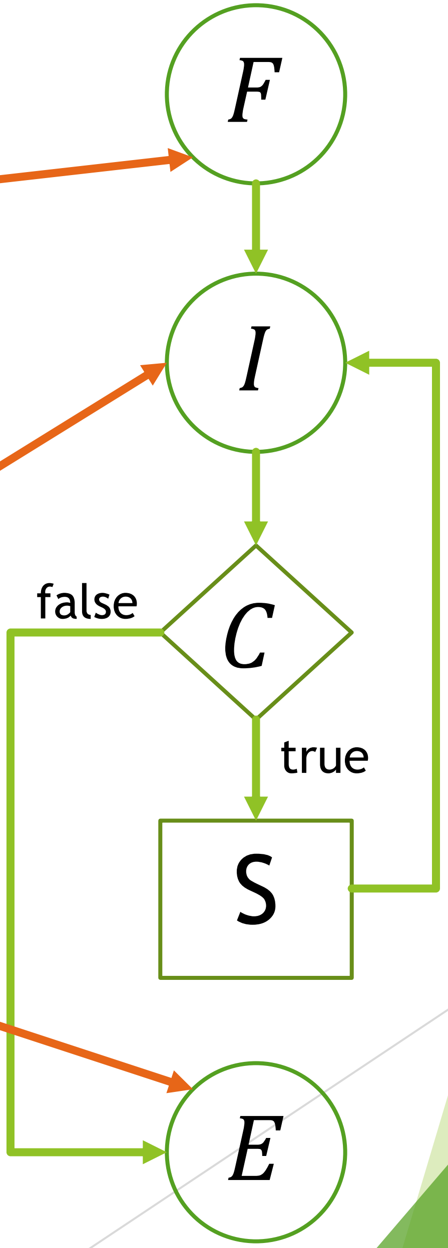
**Rule 1:** $F \rightarrow I$

The loop invariant makes it possible to verify the loop

**Rule 2:** $\{I \wedge C\}\mathbf{S}\{I\}$

The postcondition of the loop describes the state just after the loop finishes

**Rule 3:** $I \wedge \neg C \rightarrow E$



# Prove the sum formula using program verification

$\{ n \geq 0 \}$

An executable and verified sequence of code where $n$ does not change, but $s$ gets a new value

$\{ s = 1 + 2 + \cdots + n \}$
$\{ s = n(n+1)/2 \}$

- Here we prove the formula for $n \geq 0$, not just $n \geq 1$, but that is a minor point
- Is equivalent to proof by induction if and only if the termination of the sequence of code is also proven

# Prove the sum formula using program verification

$\{ n \geq 0 \}$

$\boxed{\text{Initialize } s \text{ and } k}$

**while** $k \neq n$

$\{ 0 \leq k \leq n \}$

$\{ s = 1 + 2 + \cdots + k \}$

$\{ s = k(k+1)/2 \}$

$\boxed{\text{Preserve the loop invariant}}$

$\{ s = 1 + 2 + \cdots + n \}$

$\{ s = n(n+1)/2 \}$

- Here we prove the formula for $n \geq 0$, not just $n \geq 1$, but that is a minor point
- Is equivalent to proof by induction if and only if the termination of the sequence of code is also proven

# Prove the sum formula using program verification

$\{\ n \geq 0\ \}$

$k \coloneqq 0$

$s \coloneqq 0$

**while** $k \neq n$

$\quad \{\ 0 \leq k \leq n\ \}$

$\quad \{\ s = 1 + 2 + \cdots + k\ \}$

$\quad \{\ s = k(k+1)/2\ \}$

$\quad k \coloneqq \mathrm{k} + 1$

$\quad s \coloneqq s + k$

$\{\ s = 1 + 2 + \cdots + n\ \}$

$\{\ s = n(n+1)/2\ \}$

- Here we prove the formula for $n \geq 0$, not just $n \geq 1$, but that is a minor point
- Is equivalent to proof by induction if and only if the termination of the sequence of code is also proven

# Prove the sum formula using program verification

$\{ n \geq 0 \}$
$k := 0$
$s := 0$
**while** $k \neq n$
$\quad \{ 0 \leq k \leq n \}$
$\quad \{ s = 1 + 2 + \cdots + k \}$
$\quad \{ s = k(k+1)/2 \}$
$\quad k := \mathrm{k} + 1$
$\quad s := s + k$
$\{ s = 1 + 2 + \cdots + n \}$
$\{ s = n(n+1)/2 \}$

- Here we prove the formula for $n \geq 0$, not just $n \geq 1$, but that is a minor point
- Is equivalent to proof by induction if and only if the termination of the sequence of code is also proven
- In this case termination is obvious: we will go through the loop exactly $n$ times
- The value of the formula $n - k$ is originally n and decreases by 1 each pass through the loop and cannot decrease below 0 because then the loop terminates

# Arguments

- Remember that the code
  $\{\ F\ \}$
  **while** $C$
  $\quad\{\ I\ \}$
  $\quad$S
  $\{\ E\ \}$
- needs to fulfil the following:

1. $F \rightarrow I$
2. $\{C \wedge I\}\mathrm{S}\{I\}$
3. $I \wedge \neg C \rightarrow E$

- Clearly rules 1 and 3 are fulfilled
- Rule 2 is also fulfilled since if we let $k'$ and $s'$ stand for the new values in the variables $k$ and $s$ after each general pass through the loop,we get $k' = k + 1$ and we get

$$\begin{aligned} s' &= s + k' \\ &= 1 + 2 + \cdots + k + k' \\ &= 1 + 2 + \cdots + k + (k + 1) \\ &= 1 + 2 + \cdots + k' \end{aligned}$$

and we also get

$$\begin{aligned} s' &= s + k' \\ &= k(k+1)/2 + k' \\ &= (k^2 + k)/2 + (k + 1) \\ &= (k^2 + 3k + 2)/2 \\ &= (k + 1)(k + 2)/2 \\ &= k'(k' + 1)/2 \end{aligned}$$

which is what we needed to prove that the loop invariant is preserved (in addition to $0 \leq k' \leq n$, which is obvious)

# Proving the same in Dafny (to you)

```
method SumIntsLoop( n: int ) returns ( s: int )
  requires n >= 0
  ensures s == n*(n+1)/2
{
  s := 0;
  var k := 0;
  while k != n
    decreases n-k
    invariant 0 <= k <= n
    invariant s == k*(k+1)/2
  {
    k := k+1;
    s := s+k;
  }
}
```

# Proving it recursively

```
method SumIntsRecursive( n: int ) returns ( s: int )
  requires n >= 0
  ensures s == n*(n+1)/2
{
  if n == 0 { return 0; }
  s := SumIntsRecursive(n-1);
  s := s+n;
}
```

# Proving the same in Dafny (to Dafny)

```
function SumIntsFunction( n: int ): int
  requires n >= 0
{
  if n == 0 then
    0
  else
    SumIntsFunction(n-1)+n
}

lemma SumIntsAll()
  ensures forall n | n >= 0 :: SumIntsFunction(n) == n*(n+1)/2
{}
```

# Proving the same in Dafny (to Dafny)

```
function SumIntsFunction( n: int ): int
  requires n >= 0
{
  if n == 0 then
    0
  else
    SumIntsFunction(n-1)+n
}

method Main()
{
  assert forall n | n >= 0 :: SumIntsFunction(n) == n*(n+1)/2;
  print "Success!";
}
```

# Induction without computation (1/2)

```
function SumIntsFunction( n: int ): int
  requires n >= 0
{
  if n == 0 then
    0
  else
    SumIntsFunction(n-1)+n
}

lemma SumIntsLemma( n: int )
  requires n >= 0
  ensures SumIntsFunction(n) == n*(n+1)/2
{
  if n == 0 { return; }
  SumIntsLemma(n-1);
}
```

# Induction without computation (2/2)

```
function SumIntsFunction( n: int ): int
  requires n >= 0
  decreases n
  ensures SumIntsFunction(n) == n*(n+1)/2
{
  if n == 0 then
    0
  else
    SumIntsFunction(n-1)+n
}
```