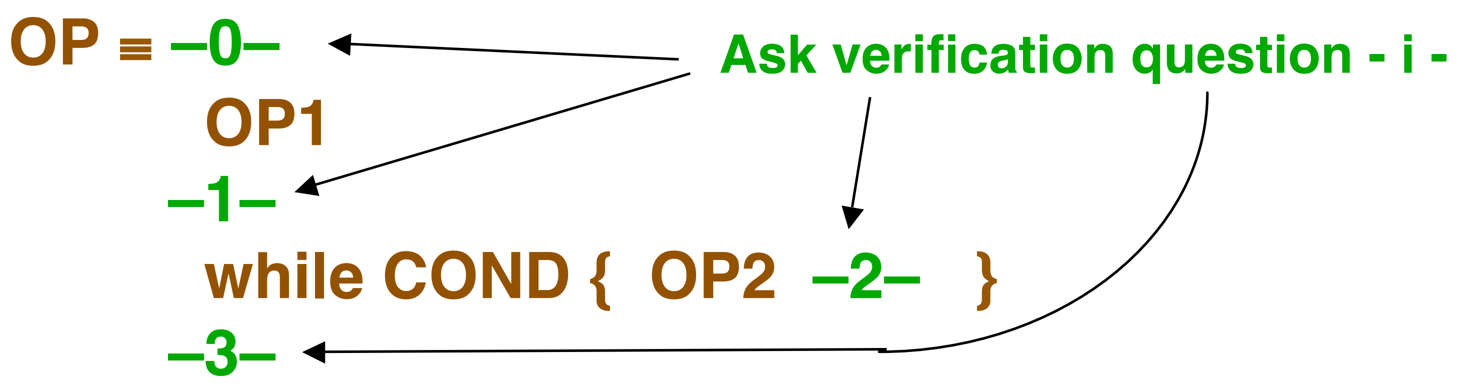
Loop invariants

Loop Questions – 1 of 4



```
OP ≡ –0–
        OP1
      –1–
        while COND {  OP2  –2–   }
      –3–
```

Let **LI** be a loop invariant, which must always be true after **OP1** is executed – except temporarily within **OP2**

Loop Questions – 2 of 4

Question 0 – What is the **LI**?

- » **In general it is an extremely difficult question to answer. It contains the essential difficulty in programming**
- » **Fundamentally it is the following**

$$\text{LI} \equiv \text{totalWork} = \text{workToDo} + \text{workDone}$$

Loop Questions – 3 of 4

```
OP ≡ –0–
        OP1
     –1–
        while COND {  OP2  –2–   }
     –3–
```

Question 1 – Is **LI** true after OP1?

precondition(OP) + execution(OP1) ⇒ LI

Question 2 – Is **LI** true after OP2?

(LI and COND) + execution(OP2) ⇒ LI

Loop Questions – 4 of 4

```
OP ≡ –0–
        OP1
      –1–
        while COND {  OP2  –2–   }
      –3–
```

Question 3a – Does the loop terminate?

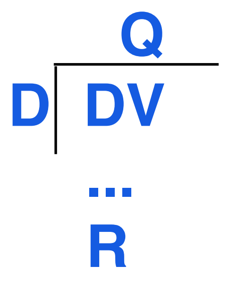
Does COND eventually become false?

Question 3b – Is postcondition of **OP** true at loop end?

(LI and (not COND)) ⇒ postcondition OP

Example Loop Design

- Consider a program loop which calculates the division of positive integers.
  - » **D is the divisor and $D > 0$**
    Q is the quotient
    R is the remainder
    DV is the dividend and $DV > 0$



- We are to compute **Q** and **R** from **D** and **DV** such that the following is true.

$$0 \leq R < D \quad \& \quad DV = D*Q + R$$

Loop Design – 1

- Question 0 – Find the loop invariant
  - » **After consulting an oracle we have determined that the following is an appropriate loop invariant**
    - > **this is the creative part of programming**

$$LI \equiv DV = D*Q + R \quad \& \quad R \geq 0$$

Loop Design – 2

```
OP ≡ –0–                LI ≡ DV = D*Q + R  & R ≥ 0
      OP1
    –1–
      while COND {  OP2  –2–  }
    –3–
```

- What we have to do is to determine **COND**, **OP1**, and **OP2** while checking that the verification questions are satisfied.
  - » **In practice we iterate between loop invariant and the program until we have a match that solves the problem.**

Loop Design – 3

LI ≡ DV = D*Q + R & R ≥ 0

- Question 1 – Make **LI** true at the start
  - **OP1 ≡ Q := 0 ; R := DV**
    - > **LI is certainly true**
  - » **DV = D*0 + DV**
  - » **DV > 0 from the precondition ⇒ R ≥ 0**

Loop Design – 4

LI ≡ DV = D*Q + R & R ≥ 0

while COND { OP2 –2– }

- Question 2 – Is **LI** still true after **OP2** is executed?

  COND ≡ R ≥ D **True before OP2 exec**

  OP2 ≡ Q := Q + 1 ; R := R – D

  Therefore Q' = Q + 1 & R' = R - D

  » **After OP2 show LI first part is true**

  ```
  > DV = D*Q' + R'                LI first part
       = D*(Q + 1) + (R - D)      Subst equality
       = D*Q + D + R - D          Rearrange
       = D*Q + R                  True before OP2, So still true
  ```

  » See effect of moving data from **workToDo (D & DV)** to **workDone (Q & R)** while maintaining the invariant.

Loop Design – 5

LI ≡ $DV = D*Q + R$ & $R \geq 0$

while COND { OP2 –2– }

- Question 2 – Is LI still true after OP2 is executed?

  COND ≡ $R \geq D$ True before OP2 exec

  OP2 ≡ $Q := Q + 1$; $R := R - D$

  Therefore $Q' = Q + 1$ & $R' = R - D$

  » After OP2 show second part of LI is still true

  > $R' \geq 0$ LI second part
  > $\Rightarrow (R - D) \geq 0$ Subst equality
  > $\Rightarrow R >= D$ Rearrangement is true from COND
  > Therefore $R' \geq 0$ is true

Loop Design – 6

LI ≡ DV = D*Q + R & R ≥ 0

```
while $R \geq D$ {
    $Q := Q + 1$
    $R := R - D$
}
```

- Question 3a – Does **COND** eventually become false?
  - » **Every time around the loop OP2 reduces the size of R by D > 0.**
  - » **In a finite number of iterations R must become less than D.**

Loop Design – 7

LI ≡ DV = D*Q + R & R ≥ 0

COND = R ≥ D

- Question 3b

  Does ~ **COND** and **LI** ⇒ postcondition for **OP** ?

  - » **~ COND ⇒ R < D**
  - » **LI ⇒ DV = D*Q + R & R ≥ 0**
  - » **Together ⇒ DV = D*Q + R & 0 ≤ R < D**
  - » **Equals Problem spec**

    0 ≤ R < D & DV = D*Q + R

Loop Invariant – Example 1a

- Copy a sequence of characters from input to output

```
read aChar from input
while aChar ≠ EOF
    write aChar to output
    read aChar from input
end while
```

- The loop invariant is the following

$$\underline{\text{In}[\,1 \,..\, N\,] \;=\; \text{Out}[\,1 \,..\, i-1\,] + \text{aChar} + \text{In}\,[\,i+1 \,..\, N\,]}$$

$$\text{totalWork} \;=\; \text{workDone} \;+\; \text{workToDo}$$

Loop Invariant – Example 1b

- The loop invariant is the following

  In[1 .. N] = Out[1.. i - 1] + aChar + In [i + 1 .. N]

- The loop invariant can be simplified by removing **Input[i+1 .. N]** from each side of the relationship

  In[1 .. i] = Out[1 .. i - 1] + aChar

- It is the simplified form that one sees most often

Loop Invariant – Example 2a

- Compute the sum of the integers 1 to N

```
sum := 0  ;  p := 0
loop exit when p = N
    p += 1 ;  sum += p
end loop
```

- The loop invariant is the following

$$\underline{\sum_{0}^{n} i} \quad = \quad \underline{\text{sum}} \quad + \quad \underline{\sum_{p+i}^{n} j}$$

totalWork = workDone + workToDo

Loop Invariant – Example 2b

- The loop invariant is the following

$$\sum_{0}^{n} i \quad = \quad sum \quad + \quad \sum_{p+1}^{n} i$$

- Simplify by removing the following expression from each side of the relationship

$$\sum_{p+1}^{n} i$$

To get

$$\sum_{0}^{p} i \quad = \quad sum$$

Loop Invariant – Example 3a

- Compare string **A[1..p]** with string **B[1..p]**. Last character in string must be **EOS**

```
i := 1
loop exit when A[i] ≠ B[i] or A[i] = EOS
    i += 1
end loop
```

A[1 .. p] ? B[1 .. p] **totalWork**

= A[1 .. i -1] = B[1 .. i -1] **workDone**

+ A[i .. n] ? B[i .. n] **workToDo**

& $i \le p$ & A[p] = B[p] = EOS

Support conditions

Loop Invariant – Example 3b

- The loop invariant is the following.

$$A[\ 1 \,..\, p\ ] \ ?\ B[\ 1 \,..\, p\ ]$$
$$= \ A[\ 1 \,..\, i-1\ ] = B[\ 1 \,..\, i-1\ ]$$
$$+\ A[\ i \,..\, n\ ] \ ?\ B[\ i \,..\, n\ ]$$
$$\&\ \ i \le p\ \ \&\ \ A[p] = B[p] = EOS$$

- The simplified loop invariant

$$A[\ 1 \,..\, i-1\ ] = B[\ 1 \,..\, i-1\ ]$$
$$\&\ \ i \le p\ \ \&\ \ A[p] = B[p] = EOS$$