

if (cond)  
{  
    Block  
}

stmt ← Control flow

Premise: Given: Control flow reaches to  
stmt.

We cannot conclude anything about cond!

---

Premise: Given: Control flow reaches to Block.

Conclusion:  $\text{Cond}_{\text{if}} \equiv \text{True}$ .

---

if (a > b) [ Control flow reaches to block  
{  
    Block  
}  $\rightarrow a > b ] \equiv \text{True}$ .

---

if (a > b && c < d)  
{  
    Block.  
}

Control flow reaches to block  $\rightarrow (a > b \wedge c < d)$

$$p \wedge q \equiv T \rightarrow p \equiv T$$

$$p \wedge q \equiv T \rightarrow q \equiv T.$$

Control flow reaches to Block  $\rightarrow a > b.$

Control flow reaches to Block  $\rightarrow c < d.$

---

if( $a < b \parallel c == d$ )

{  
    Block

}

---

Control flow reaches to  
Block  $\rightarrow (a < b) \vee (c = d)$

---

Either  $a < b$

or

$c = d$

or

both.

---

End of if-stmt exercises.

```

if (cond)
{
    Blockif
}
else
{
    Blockelse
}

```

Control flow reaches to  $\rightarrow$  Cond  $\equiv$  True.  
 Block<sub>if</sub>

Control flow reaches to  $\rightarrow$  Cond  $\equiv$  False  
 Block<sub>else</sub>  $\rightarrow$  Cond  $\equiv$  True

```

if (a > b)
{
    Bif
}
else
{
    Belse
}

```

Cof. @ B<sub>if</sub>  $\rightarrow$   
 $(a > b) \equiv \text{True.}$

Cof. @ B<sub>else</sub>  $\rightarrow$   
 $(a > b) \equiv \text{False}$

$$a > b \equiv \text{False}$$

$$\neg (a > b) \equiv \neg \text{False} \equiv \text{True}$$

$$\neg (a > b) \equiv \text{True}$$

$$a \leq b \equiv \text{True}$$

$$\boxed{\text{CoF}_0 @ \text{B}_{\text{else}} \rightarrow a \leq b}$$

De Morgan's law.

let  $p$  be a proposition.

let  $q$  be a proposition.

$$\text{not } (p \text{ and } q) \equiv (\text{not } p) \text{ or } (\text{not } q).$$

$p$	$q$	$p \text{ and } q$	$\text{not } (p \text{ and } q)$
F	F	F	T
F	T	F	T
T	F	F	T
T	T	T	F

$p$	$q$	$\text{not } p$	$\text{not } q$	$(\text{not } p) \text{ or } (\text{not } q)$
F	F	T	T	T
F	T	T	F	T
T	F	F	T	T
T	T	F	F	F

$$\text{not } (p \text{ and } q) \equiv (\text{not } p) \text{ or } (\text{not } q)$$

$$\neg(p \wedge q) \equiv \neg p \vee \neg q. \quad \text{--- (1)}$$

$$\text{not } (p \text{ or } q) \equiv (\text{not } p) \text{ and } (\text{not } q)$$

$$\neg(p \vee q) \equiv \neg p \wedge \neg q.$$

$p$	$q$	$p \vee q$	$\neg(p \vee q)$	$\neg p$	$\neg q$	$\neg p \wedge \neg q$
F	F	F	T	T	T	T
F	T	T	F	T	F	F
T	F	T	F	F	T	F
T	T	T	F	F	F	F

$$\neg(p \vee q) \equiv \neg p \wedge \neg q.$$

∃ else examp #2.

if ( $a \geq b$  &&  $c < d$ )

{ Bif

}

else

{

Belse

};

---

C.F. @ Bif  $\rightarrow (a \geq b) \wedge (c < d)$

C.F. @ Belse  $\rightarrow [(a \geq b) \wedge (c < d)] \in F$

$\neg [\underbrace{(a \geq b)}_p \wedge \underbrace{(c < d)}_q] \in T.$

Appl<sub>7</sub><sup>p</sup> De Morgan.

$\neg (a \geq b) \wedge (c < d) \equiv \neg (a \geq b) \vee$   
 $\neg (c < d)$

$\equiv (a < b) \vee (c \geq d)$

C.F. @ Belse:

Either  $a < b$

OR  $c \geq d$

OR Both.

---

if  $((a \leq b) \vee (c \neq d))$

{

Bif

}

else

{

Belse

}

C.F. reaches to Bif  $\rightarrow a \leq b \vee c \neq d$ .

C.F. reaches to Belse  $\rightarrow \neg(a \leq b \vee c \neq d)$

$\equiv \neg(a \leq b) \wedge \neg(c \neq d)$

$\equiv (a > b) \wedge (c = d)$

ALG: Find maximum of two numbers  $a$  &  $b$ .

```
int max(int a, int b)
{ int m;
```

P. M.

```
    if ( $a > b$ )
```

```
         $m = a$ ;
```

```
    else
```

```
         $m = b$ ;
```

```
    return ( $m$ );
```

```
}
```

Control flow reaches true Bif.

$\rightarrow a > b$ .

Execute  $m = a$ .

$a > b \rightarrow a$  is max amongst  $a$  &  $b$

State of var  $m$  = state of var  $a$ .

∴ State of var  $m$  = state of max.  
of  $a$  &  $b$ .

---

C.F. reaches to Belse  $\rightarrow a > b \equiv F$ .

$a > b \equiv \text{False}$



e.g.  $\neg (a > b) \equiv \text{True}$ .

$a \leq b \equiv \text{True}$ .

$a \leq b \equiv a < b \quad \text{or} \quad a = b$ .

In both cases  $b$  is max of  $a$  &  $b$ .

e.g. Execute  $(m = b)$

→ State of var  $m = \text{state of var } b$

$\therefore$  State var  $m$  is max. of  $a$  &  $b$

---

```
void search(int *pa, int N, int sd)
```

```
{  
    int i;
```

```
    for (i = 0; i < N; i++)
```

```
        if (pa[i] == sd)
```

```
            break;
```

```
    if (i < N)
```

```
        print("Data found")
```

```
    else
```

```
        print("Data not found")
```

```
}
```

Algorithm terminates  $\rightarrow$

Control flow has come out of  
for loop.

C.F. out for loop  $\rightarrow$

loop cond False or break executed.

Possibility #1:

loop condition false.

$$i < N \equiv \text{False.}$$

$$\neg (i < N) \equiv \text{True.}$$

$$i \geq N \equiv \text{True.}$$

$$\boxed{i == N \equiv \text{True}} \rightarrow \text{Conclusion-1}$$

loop-condition  $\equiv \text{False.}$

for values of  $i$  from 0 to  $N-1$

$$(pa[i] == sd) \equiv \text{False}$$

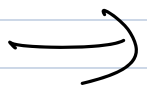
$\rightarrow$  How?

because if for any  $i_0$   
between 0 to  $N-1$  ( $0 \leq i_0 \leq N-1$ )  
 $pa[i_0] == sd$  had been true  
then break statement would  
have got executed with value  
of  $i = i_0$  which is  $< N$ .

$\therefore i_0 < N$  &  $i_0 = N$  cannot  
be satisfied together.

for  $0 \leq i < N$ ,  $pa[i] == sd$  is false  
 $\Rightarrow$  search data not found.

Exited from loop because  
loop cond is false



- 1.)  $i = N$  just after exit from  
loop
- 2.) Search data not present!

Possibility #2: break statement executed.

break statement executed  $\rightarrow$   
 $pa[i] == sd$  is true.  
 $\rightarrow 0 \leq i < N$

break stmt executed  $\rightarrow$

$0 \leq \underline{i} < N \wedge$  data found  
in array

---

void insert\_at\_sorting\_pos(int \*pa, int N)

{ /\* Precondition:  $N \geq 2$ .

$pa[0]$  to  $pa[N-2]$  is sorted  
in ascending order

\*/

int tmp = pa[N-1];

int k;

k = N-2;

while(k > 0 && pa[k] > tmp)

{

$pa[k+1] = pa[k];$

$\{ \quad k = k-1;$

$pa[k+1] = tmp;$

$\swarrow$  But  $pa[0]$  to  $pa[N-1]$  is

sorted in ascending  
order

$\swarrow$   
 $\{$

Control flow out of while loop.  
loop condition  $\equiv$  False.

$k \geq 0 \wedge pa[k] > tmp \equiv F$

$\neg(k \geq 0 \wedge pa[k] > tmp) \equiv T,$

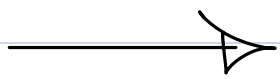
$\neg(k \geq 0) \vee \neg(pa[k] > tmp)$

$k < 0 \vee pa[k] \leq tmp,$

Case #1:  $k < 0$ .

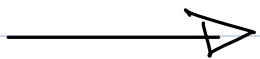
As we are decrementing  $k$  from  $N-2$  by value of 1,  $k = -1$

the loop terminated for  $k = -1$



the loop body executed for values of  $k$  from  $N-2$  to 0.

$k = N-2, N-3, N-4, \dots, 0$ .



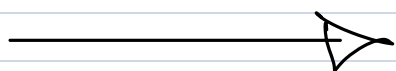
$pa[k] > tmp$  is true for

$k = N-2, N-3, \dots, 0$ .

Execution of stmt.

$pa[k+1] = pa[k]$

for values  $k = N-2, N-3, \dots, 0$   
resp.



Numbers in array from index = 0 to index = N-2 at the start of ALG have been shifted

to index = 1 to index = N-1 without changing internal order.

Precondition  $\equiv$   $pa[0]$  to  $pa[N-2]$  sorted.

We just proved above that these elements are shifted to  $pa[1]$  to  $pa[N-1]$  without changing relative order.

$\therefore$  at the end of loop  $pa[1]$  to  $pa[N-1]$  are sorted,

and

$pa[1]$  to  $pa[N-1] > tmp$ .

After loop  $pa[k+1] = tmp$ .

$pa[-k+1] = tmp$

$pa[0] = tmp$

$$= pa[N-1]$$

start

$pa[0]$  to  $pa[N-1]$  is sorted. <sup>also</sup>

Case #2:

$$k \geq 0 \text{ and } pa[k] > tmp$$

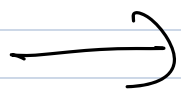
$\equiv$   
True

$\equiv$   
False.

Ans 1:  $0 \leq k \leq N-2$ .

Ans 2: Let  $k_0$  be a number between 0 to  $N-2$  ( $0 \leq k_0 \leq N-2$ ) for which  $pa[k_0] > tmp$  is False.

the loop terminated for  $k_0$



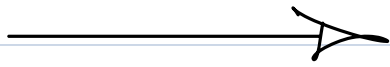
loop executed for  $k = N-2$  down to  $k = k_0 + 1$

→ Array elements from  $k = k_0 + 1$  to



$$k = N-2 > tmp.$$

loop body executes for  
 $k = N-2$  down to  $k = k_0+1$



at the end of loop

$pa[k_0+1]$  to  $pa[N-2]$

are shifted to

$pa[k_0+2]$  to  $pa[N-1]$

without changing internal order.

As  $pa[0]$  to  $pa[N-2]$  were sorted

$pa[k_0+2]$  to  $pa[N-1]$  are sorted.

$$pa[k_0+2 : N-1] > tmp.$$

---

$$pa[k_0] \leq tmp.$$

As  $pa[0]$  to  $pa[N-2]$  were sorted

$pa[0]$  to  $pa[k_0]$  are also sorted

$$pa[k_0] \leq tmp.$$

$$pa[0 : k_0] \leq tmp.$$

$$pa[0 : k_0] \leq tmp \text{ \& }$$

$$pa[k_0 + 2 : N-1] > tmp.$$

$$\text{Execution } pa[k_{st+1}] = tmp$$

→  
 $pa[0]$  to  $pa[N-1]$  is sorted.  $\square$

Systematic consid

hidden