

int n;

int cnt;

cnt = 0;

while (cnt <= n)

printf("%d", cnt);

cnt = cnt + 1;

int cnt;

FOR (cnt = 0; cnt <= n;

cnt++) {

printf("%d", cnt);

```
int n ; cnt = 1; int a;
```

```
do{
```

```
printf("inserisci un valore:"); }
```

```
scanf("%d", &n);
```

```
}while(n < 1);
```

```
a = n;
```

```
cnt = n - 1;
```

```
while(cnt > 0){
```

```
n = n * cnt
```

```
cnt--;
```

$$n! = n(n-1) \cdot (n-2) \cdots 1$$

$$5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$$

$$n! = n \cdot (n-1) \cdot \dots \cdot 1$$

← ~~FAT~~

int ~~Fat~~=1; Fat2

int cnt=1;

int n; // INPUT

WHILE (cnt <= n)

Fat = Fat · cnt;

cnt++;

$$Fat = 1 \cdot 2 \cdot 3 \cdot \dots \cdot (n-1) \cdot n$$

int n;

int Fat2=1;

int cnt;

cnt = n;

While (cnt > 0)

Fat2 = Fat2 · cnt;

cnt--;

```
INT FAT = 1; INT CNT;  
FOR (CNT = 1; CNT <= Q; CNT++) {  
    FAT = FAT * CNT;  
}
```

ES TRIANGULAR OF FLOYD

INPUT

$$(n > 0)$$

A handwritten diagram showing the sequence of strokes for writing the numbers 1 through 10. Each number is accompanied by an arrow indicating the direction and order of the strokes. The numbers are arranged in a grid-like fashion, with some numbers having multiple strokes. The entire diagram is enclosed in a blue rectangular border.

Number	Stroke Sequence
1	1 stroke (vertical line down)
2	1 stroke (curved line from top left to bottom right)
3	2 strokes (two curved lines, one above the other)
4	1 stroke (vertical line down)
5	1 stroke (vertical line down)
6	1 stroke (curved line from top left to bottom right)
7	1 stroke (vertical line down)
8	1 stroke (vertical line down)
9	1 stroke (curved line from top left to bottom right)
10	2 strokes (vertical line down, then horizontal line to the right)

WHILE() {

WHILE() {

~~DD - KTH LE~~

A hand-drawn diagram on a white background. It features a blue curve in the upper left quadrant, starting from the left edge and curving upwards and to the right. A vertical red dashed line is drawn to the right of the blue curve. Another vertical red dashed line is further to the right, with a red bracket indicating a distance between the two dashed lines. A red line starts from the bottom left, curves upwards and to the right, and then curves back to the left, ending near the bottom center.

```
int n;
int riga;
int colonna;
```

COND.
TRUE

($n > 0$)

NOT
→

COND.
FALSE

($n \leq 0$)

```
do {
    printf("Inserisci n: ");
    scanf("%d", &n);
} while (n <= 0);
```

```
printf(".....");
scanf("%d", &n);
while (n <= 0) {
    printf(".....");
    scanf("...");
}
```

```

riga = 1;
int num = 1;
while (riga <= n) {
    colonna = 1;
    while (colonna <= riga) {
        printf("%d", num);
        num++;
        colonna++;
    }
    printf("\n");
    riga++;
}

```

```

riga = 1;
int num = 1;
for (riga = 1; riga <= n; riga++)
{
    colonna = 1;
    for (colonna = 1; colonna <= riga; colonna++)
    {
        printf("%d", num);
        num++;
    }
    printf("\n");
}

```


ES DIRE. SE UM NUMERO
E PRIMO O NO (CON FOR) ^{8/21}

```
int m;  
int cnt;  
int div = 0;  
for(cnt = 1; cnt <= h; cnt++) {  
    if(n % cnt == 0) {  
        div = div + 1;  
    }  
}
```



```
if (div == 2) {  
    printf("n e' un numero primo");  
}  
else {  
    printf("n non e' un numero primo");  
}
```

≡
↓
≡

$(a \geq 10 \ \&\& \ a < 30) \ \|\ (b \geq 5 \ \&\& \ b \% 2 \neq 0)$

$(a \leq 10 \ \|\ a \geq 30) \ \&\&$

$(b \leq 5 \ \|\ b \% 2 \neq 0)$

COMPARE 1 NUMBER
PARTY TRAINED FOR
($n \leq m$)

$n \geq 5$
+1
 $m = 10$

6
8
10

n, m, \underline{cnt}

```
FOR(cnt = n, cnt <= m; cnt++) {
    if(cnt % 2 == 0) {
        print(...);
    }
}
```

```
[ if (n % 2 == 1) { // DISPAR,  
    n++;  
} // n DIVENTA PAR.
```

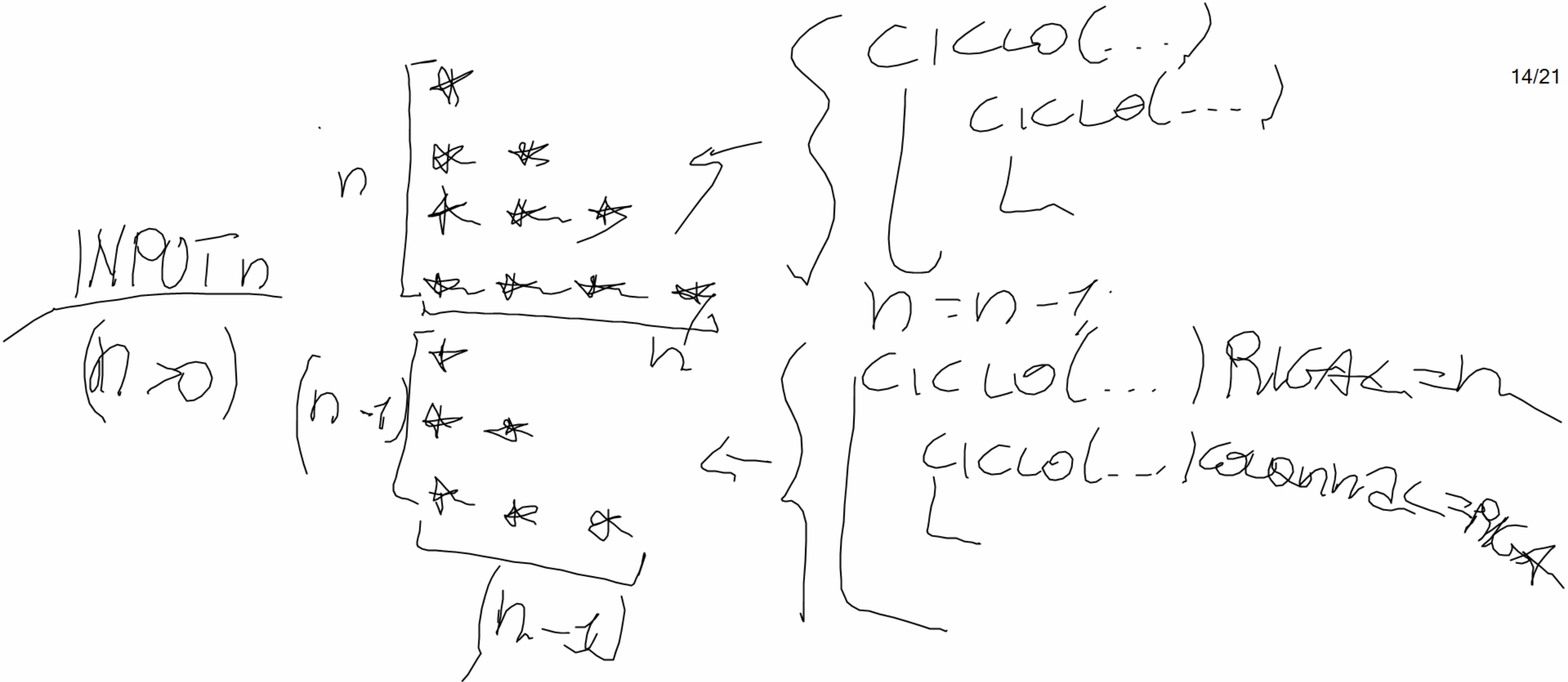
```
for (cnt = n; cnt <= m; cnt += 2) {  
    printf(...),  
}
```

```
int a = 3;
int i = 3;
int j = 6;
```

```
while (i < j) {
    a = a * i;
    i++;
}
```

~~i = 6~~
~~a = 180~~

NO
~~while (i < j) {~~




ES. DATO UN NUMERO n ($n > 0$)
SOMMARE I PRIMI n NUMERI
 DISPARI $[n \rightarrow n^2]$

ES

$$n = 5$$

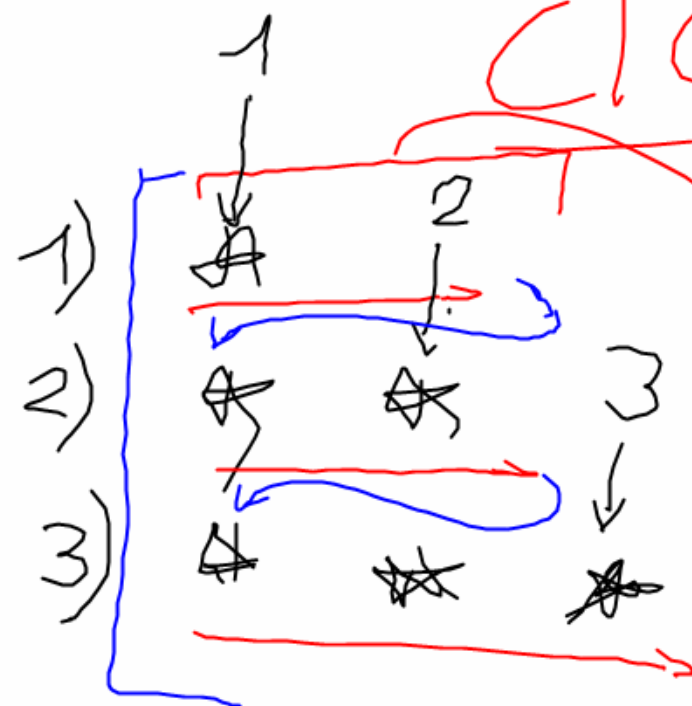
$$(n \cdot 2)$$

$$1 + 3 + 5 + 7 + 9 = 25 = (5)^2 = (n)^2$$




```
int n;  
int i;  
int som=0;  
FOR(i=1, i <= (n/2); i=i+2){  
    som=som+i;  
}
```

CICLI ANNIDATI



→ CICLO (PER COLONNE)

CICLO
(PER RIGHE)

CICLO (RIGHE)

^{COL = 1}
CICLO (COLONNE) (→)

~ (N) RIGHE

```

int RIGA=1;
int COL=1;
while (RIGA <= n) {
    COL=1;
    while (COL <= RIGA) {
        printf(...);
        COL++;
    }
    printf("\n");
    RIGA++;
}

```

(n > 0) n

*		
*	*	
*	*	*

```

int RIGA=1 int COL=1
for (RIGA=1; RIGA <= n; RIGA++) {
    for (COL=1; COL <= RIGA; COL++) {
        printf(char);
    }
    printf("\n");
}

```



CICLO (RIGHE)
CICLO (COL)

~~CICLO (RIGHE)
CICLO (COL)~~

i=1 CICLO (i <= 2) {

CICLO (RIGHE <= n) {
COL = 1;
CICLO (COL <= RIGHE) {
...
COL ++;
}
RIGHE ++;
PRINTF("n");
}
i ++; n --;
}

ES

	1	2	3	4
4	*	*	*	*
3	*	.	*	
2	*	*		
1	*			

FOR (RIGHE=N; RIGHE >= 1; RIGHE--) {

FOR (COL=RIGHE; COL >= 1; COL--) {

| PRINTF("*");

| }

PRINTF("\n");

}

n=5

1	2	3	4	5
-	2	3	4	5
-	-	3	4	5
-	-	-	4	5
-	-	-	-	5

CICLO (RIGHE)

CICLO (SPAZIO)

CICLO (NUM)

int SPAZIO = 0;

int RIGA = 1;

int COL = 1;

for (RIGA = 1; RIGA <= n; RIGA++) {

for (

) {

