Loops / Repetition Statements

- ***Repetition statements*** **allow us to execute a statement multiple times**
- **Often they are referred to as *loops***
- **C has three kinds of repetition statements:**
  - **the *while loop***
  - **the *for loop***
  - **the *do loop***
- **The programmer should choose the right kind of loop for the situation**

Example 1: Fixing Bad Keyboard Input

- Write a program that refuses to accept a negative number as an input.
- The program must keep asking the user to enter a value until he/she enters a positive number.
- How can we do this?

Try to solve it using if-else statement

- **Example program that continuously asks for positive number as input:**

```
int n;
printf (“Please enter a positive number:”);
scanf(“%d”,&n);

if (n < 0){
  printf (“Enter positive number, BE POSITIVE!\n”);
  scanf(“%d”, &n);
}

if (n < 0){
  printf (“Enter positive number, BE POSITIVE!\n”);
  scanf(“%d”, &n);
}

 .........

 .........
```

Example 2: Grade of several students

- Write a program that continuously calculates the grade of all students’ marks and stop when the user wants.
- After calculating one student’s grade (from his marks) the program must keep asking the user whether he likes to continue or not.
- How can we do this?

while Loop

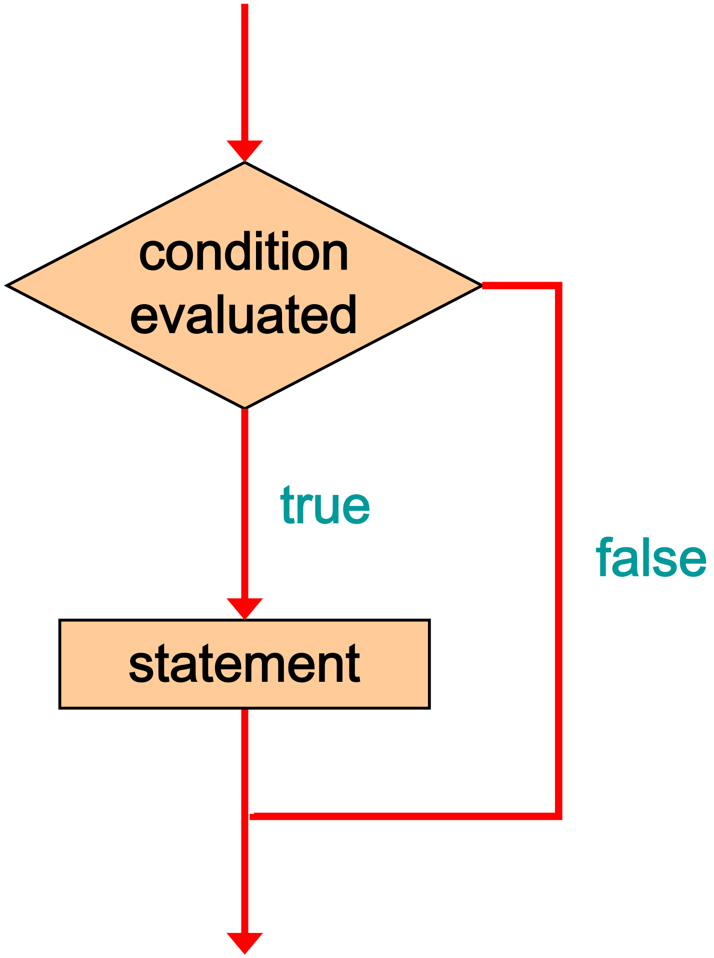
যতক্ষন যদি

while ~~**if**~~ (*condition*)
statement;

if condition is satisfied execute the statement(s)

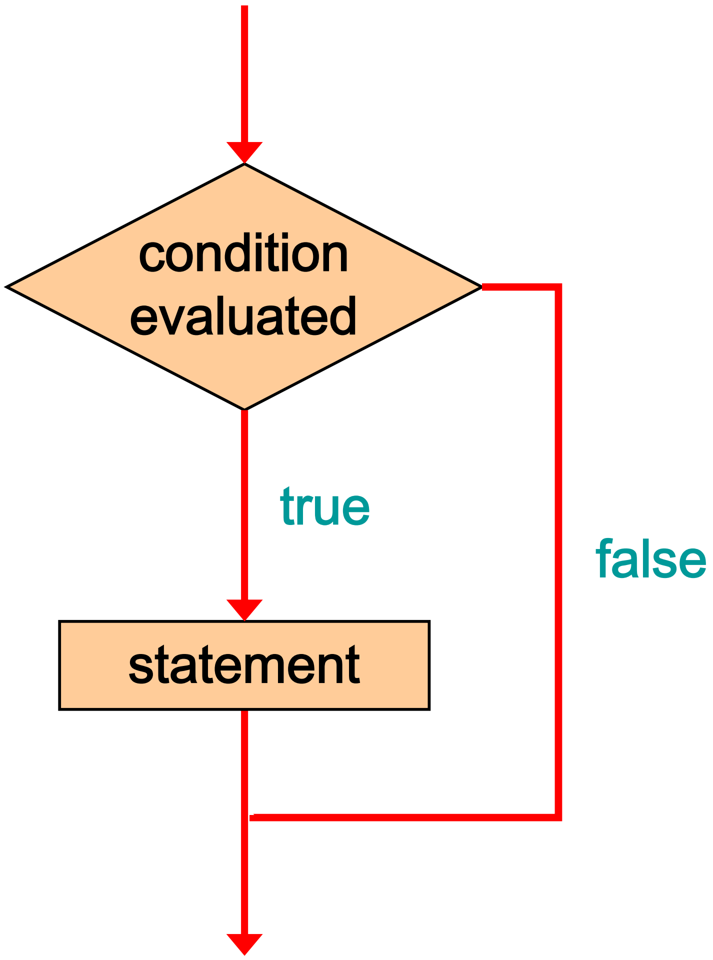
while condition is satisfied execute the statement(s)

Logic of an if statement



Logic of a while Loop

if logic



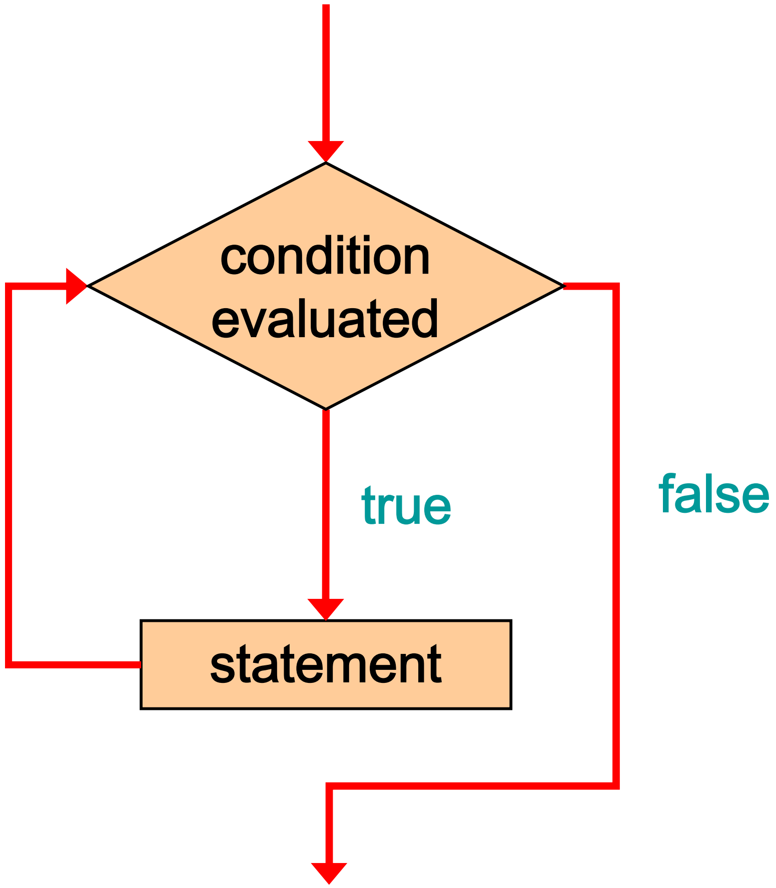
condition evaluated

true

false

statement

The while Loop



condition evaluated

true

false

statement

The while Statement formally

- A *while statement* has the following syntax:

```
while ( condition )
    statement;
```

```
while ( condition ){
    statement1;
    statement2;
……
}
```

  - If the `condition` is true, the `statement or a block of statements` is executed
  - Then the condition is evaluated again, and if it is still true, the statement/block is executed again
  - The statement/block is executed repeatedly until the condition becomes false

The while Statement

- **Example program that continuously asks for positive number as input:**

```
 int n;

 printf (“Please enter a positive number:”);
 scanf(“%d”,&n);
while(n < 0){
  if (n < 0){
   printf (“Enter positive number, BE POSITIVE!\n”);
   scanf(“%d”, &n);
 }
```

Some examples

- **Print “The sky is the limit!” 10 times.**

```
main(){
      printf (“The sky is the limit”);
}
```

Some examples

- **Print “The sky is the limit!” 10 times.**

```
main(){
      printf (“The sky is the limit”);
      printf (“The sky is the limit”);
      printf (“The sky is the limit”);
      printf (“The sky is the limit”);
      printf (“The sky is the limit”);
      printf (“The sky is the limit”);
      printf (“The sky is the limit”);
      printf (“The sky is the limit”);
      printf (“The sky is the limit”);
      printf (“The sky is the limit”);

}
```

Some examples

- **Print “The sky is the limit!” 100 times.**

```
main(){
        printf (“The sky is the limit”);
        printf (“The sky is the limit”);
        printf (“The sky is the limit”);
        printf (“The sky is the limit”);
        printf (“The sky is the limit”);
        printf (“The sky is the limit”);
        printf (“The sky is the limit”);
        printf (“The sky is the limit”);
        printf (“The sky is the limit”);
        printf (“The sky is the limit”);

}
```

Some examples

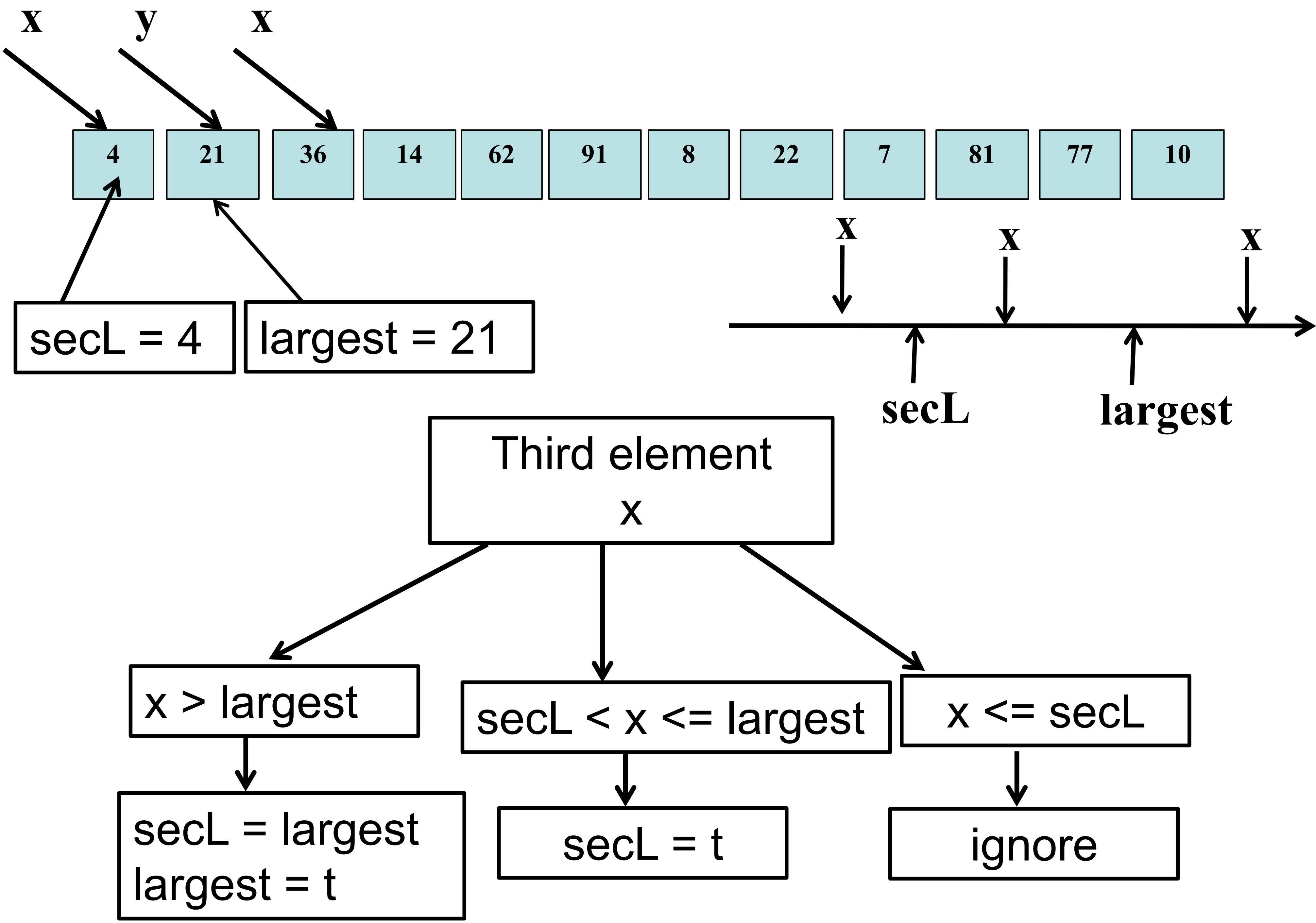
- Print “The sky is the limit!” n times. n will be user input

```
scanf(“%d”,&n);
int count = 1;
while (count <= n)
{
   printf (“The sky is the limit”);
   count++;
}
```

- If the condition of a `while` loop is false initially, the statement is never executed
- Therefore, the body of a `while` loop will execute zero or more times

Some examples

- **Print first n natural numbers.**
  - ❑ **Upwards**
  - ❑ **Downwards**
- **Print odd numbers up to n.**
- **Print even numbers up to n.**
- **Print summation of first n numbers.**
- **Print summation of all odd numbers up to n.**
- **Print summation of all even numbers up to n.**
- **Print second largest of a series of natural numbers (at least two) given as input. STOP when the user enters 0. Natural numbers are 1, 2, 3, 4……**



4	21	36	14	62	91	8	22	7	81	77	10

Third element
x

x > largest

secL = largest
largest = t

secL < x <= largest

secL = t

x <= secL

ignore

Summary of a while statement

- A `while` loop is functionally equivalent to the following structure:

```
initialization;
while ( condition )
{
    statement;
    increment;
}
```

```
initialization;
while ( condition )
{
    statement;
    increment;
}
```

- A *for statement* has the following syntax:

The *initialization* is executed once before the loop begins

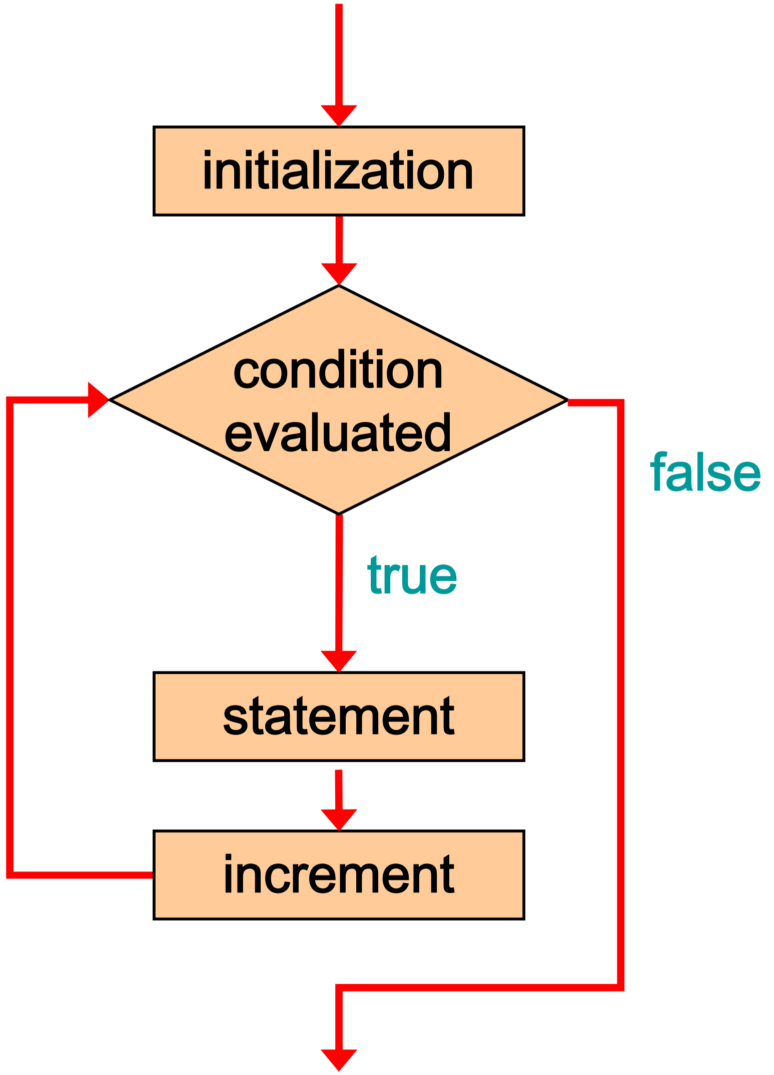
The *statement* is executed until the *condition* becomes false

```
for ( initialization ; condition ; increment )
    statement;
```

The *increment* portion is executed at the end of each iteration

Logic of a for loop

```
for ( initialization ; condition ; increment )
   statement;
```



The for Statement

$1 \le count \le n$

$1 \le count \quad \text{and} \quad count \le n$

- **An example of a `for` loop:**

```
for (count=1; count <= n; count++)
   printf (“%d\n”, count);

for (count=n; count >= 1; count--)
   printf (“%d\n”, count);
```

- **The initialization section can be used to declare a variable. The variable disappears right after loop.**

- **Like a `while` loop, the condition of a `for` loop is tested prior to executing the loop body**

- **Therefore, the body of a `for` loop will execute zero or more times**

The for Statement

- **The increment section can perform any calculation**

```
int num;
for (num=100; num > 0; num -= 5)
   printf (“%d\n”, num);
```

- **A `for` loop is well suited for executing statements a specific number of times that can be calculated or determined in advance**

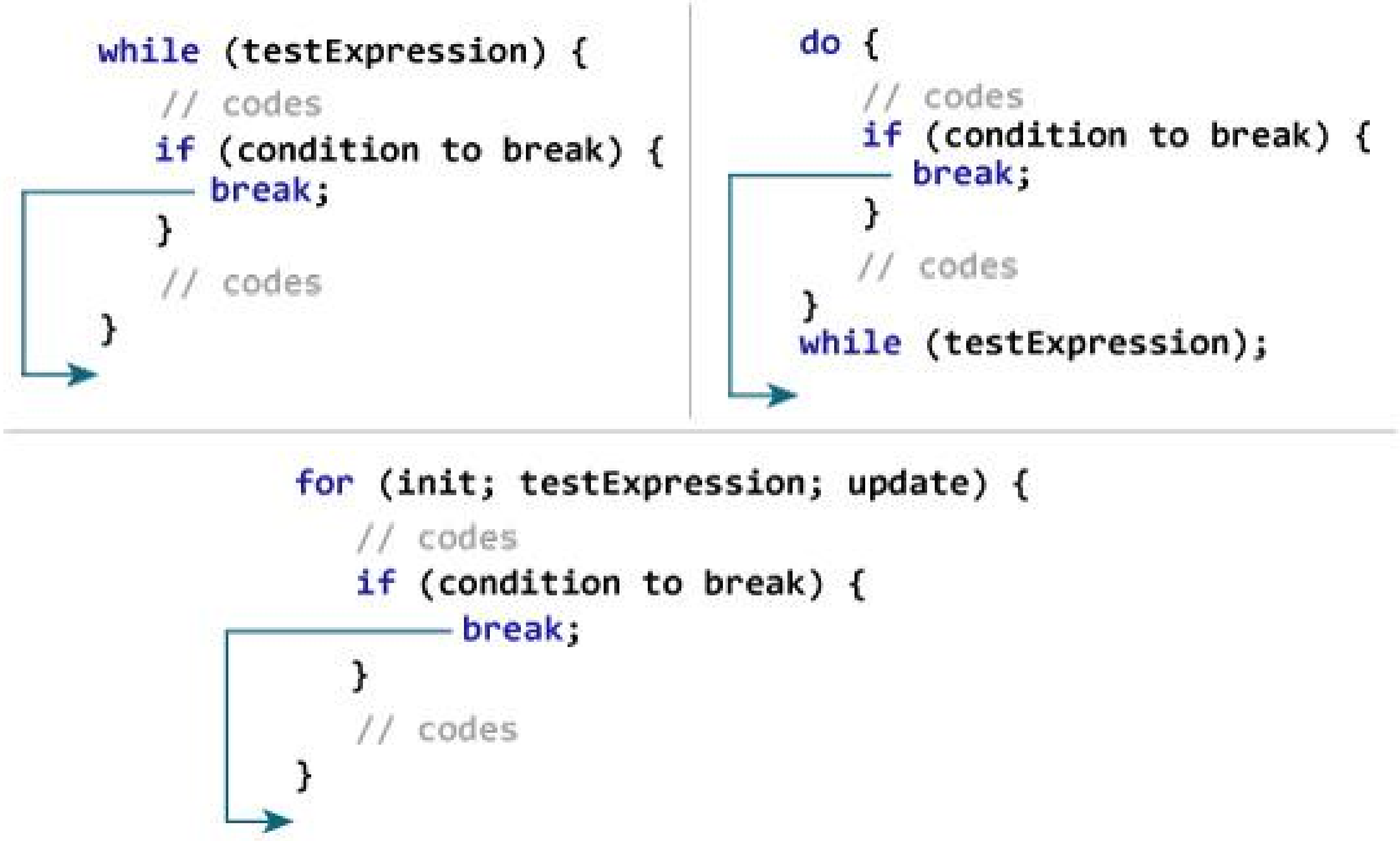
The break and coninue Statement

- Sometimes we need:
  - to skip some statements inside the loop (continue)
  - or terminate the loop immediately without checking the test condition (break).

- In such cases, break and continue statements are used.

The break Statement

The break statement terminates the loop immediately when it is encountered



```
while (testExpression) {
    // codes
    if (condition to break) {
        break;
    }
    // codes
}
```

```
do {
    // codes
    if (condition to break) {
        break;
    }
    // codes
}
while (testExpression);
```

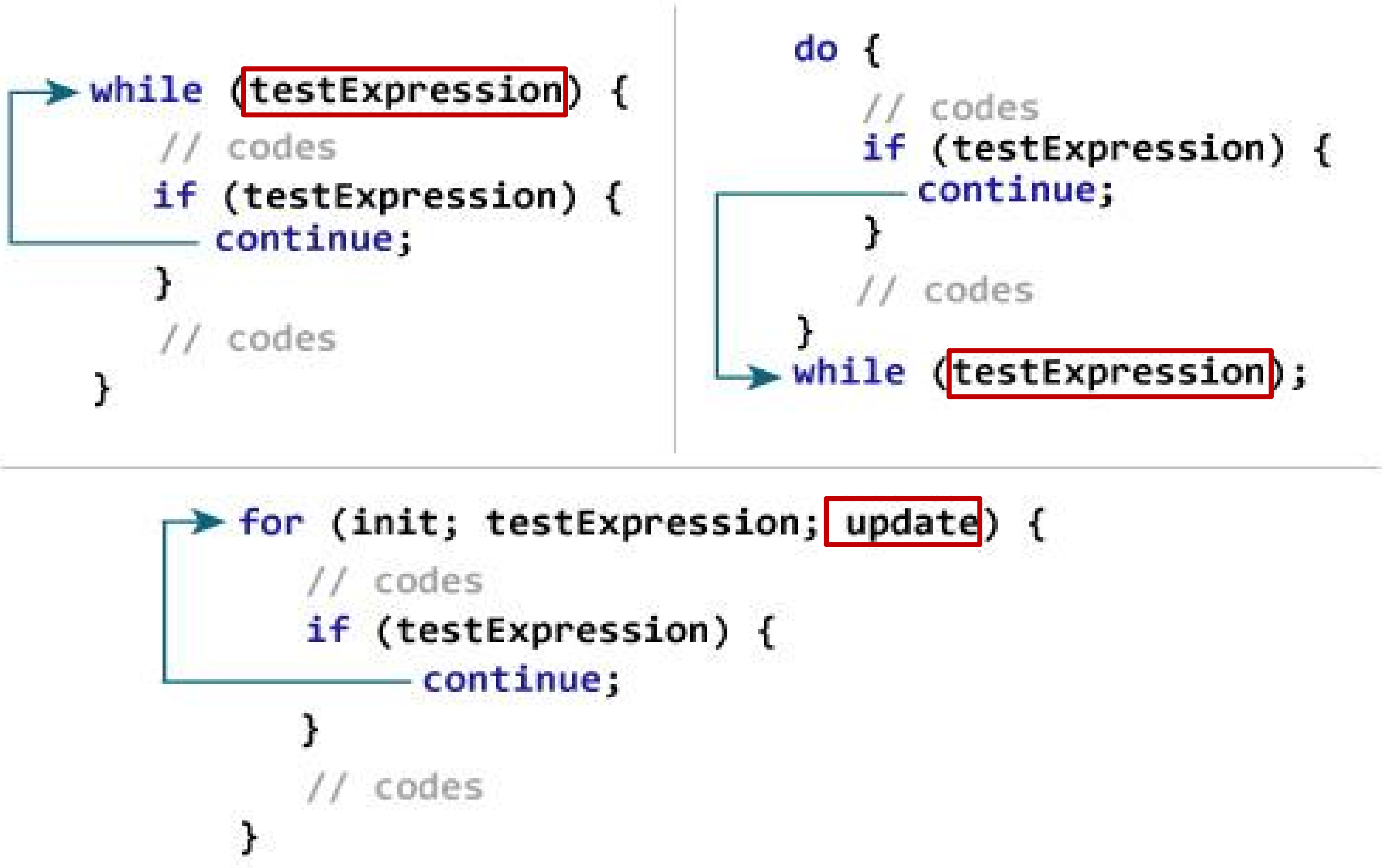
```
for (init; testExpression; update) {
    // codes
    if (condition to break) {
        break;
    }
    // codes
}
```

Example: break Statement

```
// Program to calculate the sum of maximum of 10 numbers
// If negative number is entered, loop terminates, sum is displayed
main( ) {
        int i;
        double number, sum = 0.0;
        for(i=1; i <= 10; ++i) {
                printf("Enter n%d: ",i);
                scanf("%lf", &number);
// If user enters negative number, loop is terminated
                if(number < 0.0) {
                        break;
                }
                sum += number;
        }
        printf("Sum = %.2lf",sum);
}
```

The continue Statement

The continue statement skips statements after it inside the loop.



```
while (testExpression) {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
}
```

```
do {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
}
while (testExpression);
```

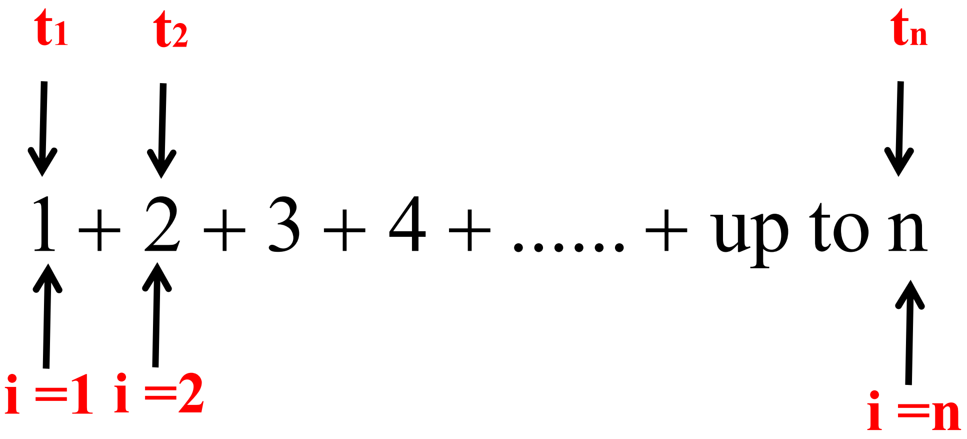
```
for (init; testExpression; update) {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
}
```

Example: continue Statement

```
// Program to calculate the sum of maximum of 10 +ve numbers
// If negative number is entered, it is ignored
main( ) {
        int i;
        double number, sum = 0.0;
        for(i=1; i <= 10; ++i) {
                printf(" Enter n%d: ", i);
                scanf("%lf", &number);
// If user enters negative number, skip it
                if(number < 0.0) {
                        continue;
                }
                sum += number;
        }
        printf("Sum = %.2lf",sum);
}
```

Some example problems

- **Write down a program to find the summation of the following series:**



t_1 t_2 t_n

1 + 2 + 3 + 4 + …… + up to n

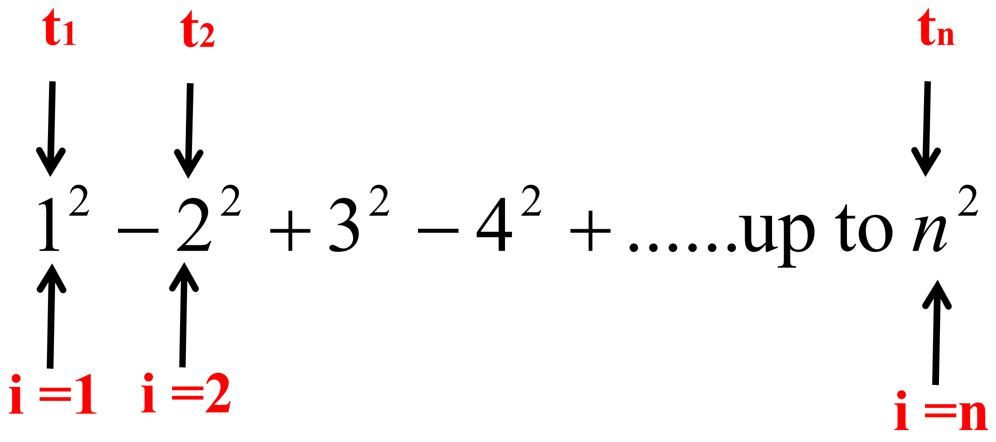
i =1 i =2 i =n

$\boxed{t = i}$

```
int main(){
        int i, n,t,s = 0;
        scanf("%d",&n);
        for(i = 1; i <= n; i++){
                t = i;
                s = s + t;
        }
        printf("%d",s);
}
```

Some example problems

- **Write down a program to find the summation of the following series:**



t_1 t_2 t_n

$$1^2 - 2^2 + 3^2 - 4^2 + \text{......up to } n^2$$

i =1 i =2 i =n

t = i^2 when i is odd

t = -i^2 when i is even

```
int main(){
        int i, n,t,s = 0;
        scanf("%d",&n);
        for(i = 1; i <= n; i++){
                if(i%2 == 0)
                        t = i*i;
                else
                        t = -i*i;
                s = s + t;
        }
        printf("%d",s);
```

Some example problems

- Write down a program to find the summation of the following series:

$$\sin(x) = \underset{i=1}{\underset{\uparrow}{\overset{\overset{t_1}{\downarrow}}{x}}} - \underset{i=2}{\underset{\uparrow}{\overset{\overset{t_2}{\downarrow}}{\frac{x^3}{3!}}}} + \frac{x^5}{5!} - \cdots \text{................ . .} \quad \overset{\overset{t_n}{\downarrow}}{\underset{\overset{\uparrow}{i=n}}{\text{N}}} \text{ terms}$$

Power of x → 2i+1

$r \rightarrow -x^2 / (2i\times(2i+1))$

$t_{new} \rightarrow r\times t_{prev}$

```
int main(){
        int i, n;
        float x,t,r,s = 0;
        scanf("%f%d",&x, &n);
        x = 22.0*x/(7*180);
        s = t = x;
        for(i = 1; i < n; i++){
                r = -x*x/(2*i*(2*i+1));
                t = r*t;
                s = s + t;
        }
        printf("%f",s);
}
```

Some example problems

- **Print factorial of n:**

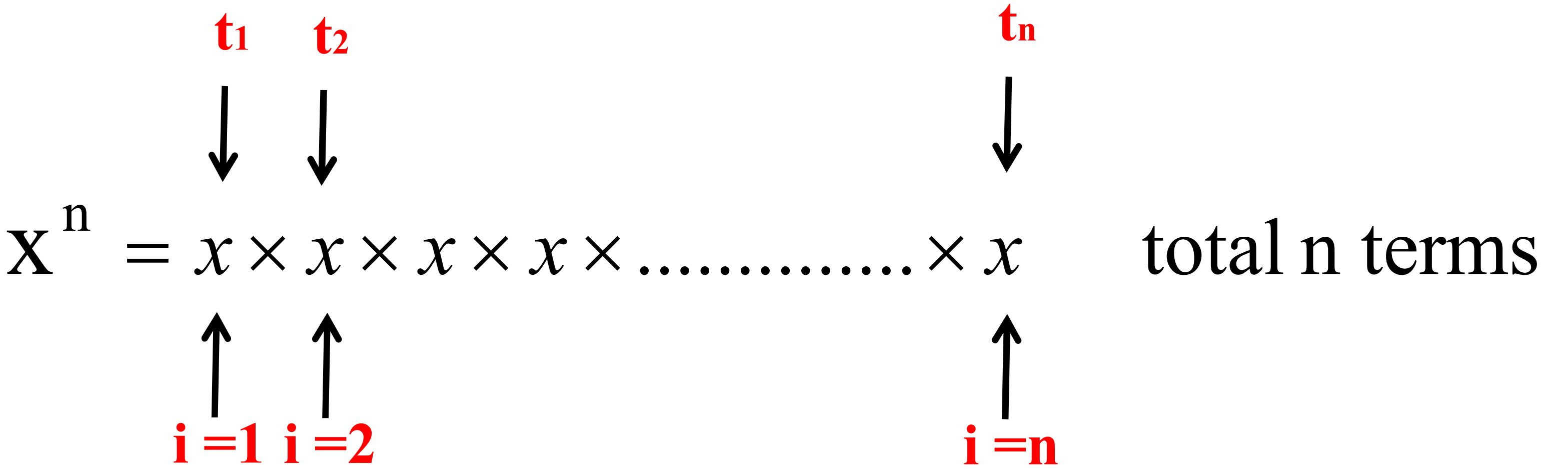
$$n! = \underset{\substack{\uparrow\\ i=1}}{\overset{\substack{t_1\\ \downarrow}}{1}} \times \underset{\substack{\uparrow\\ i=2}}{\overset{\substack{t_2\\ \downarrow}}{2}} \times 3 \times 4 \times \ldots\ldots \times \underset{\substack{\uparrow\\ i=n}}{\overset{\substack{t_n\\ \downarrow}}{\text{up to n}}}$$

$$\boxed{t = i}$$

```
int main(){
        int i, n,t,p = 1;
        scanf("%d",&n);
        for(i = 1; i <= n; i++){
                t = i;
                p = p * t;
        }
        printf("%d",p);
}
```

Some example problems

- **Print x^n:**



t_1 t_2 t_n

$$x^n = x \times x \times x \times x \times \ldots\ldots\ldots\ldots\ldots \times x \qquad \text{total n terms}$$

i =1 i =2 i =n

$$\boxed{t = x}$$

```
int main(){
        int i,x,n,t,p = 1;
        scanf("%d%d",&x,&n);
        for(i = 1; i <= n; i++){
                t = x;
                p = p * t;
        }
        printf("%d",p);
}
```

Some example problems

- **Show all factors of a number n**
  - **Candidates 1, 2, 3, 4 ………. n**

```
int main(){
        int i,n;
        scanf("%d",&n);
        for(i = 1; i <= n; i++){
                if(n%i == 0)
                        printf("%d ",i);
        }
}
```

Some example problems

- Show smallest factor of a number n (other than 1)
    - Candidates 1, 2, 3, 4 ………. n
        - Break on first candidate that becomes a factor

```
int main(){
        int i,n;
        scanf("%d",&n);
        for(i = 2; i <= n; i++){
                if(n%i == 0){
                        printf("%d",i);
                        break;
                }
        }
}
```

Some example problems

- Show largest factor of a number n (other than n)
    - Candidates 1, 2, 3, 4 ………. n
        - Break on first candidate that becomes a factor
        - Number = largest factor * smallest factor
            - largest factor = Number/smallest factor
        - Example 28 → factors 2, 4, 7, 14, smallest 2, largest 14

```
int main(){
        int i,n;
        scanf("%d",&n);
        for(i = 2; i <= n; i++){
                if(n%i == 0){
                        printf("%d",n/i);
                        break;
                }
        }
}
```

Some example problems

- Show how many factors of a number n has
  - Candidates 1, 2, 3, 4 ………. n
    - Increment a counter whenever you get a candidate which is a factor

```
int main(){
        int i,n,c=0;
        scanf("%d",&n);
        for(i = 1; i <= n; i++){
                if(n%i == 0)
                        c++;
        }
        printf("Number of factors: %d",c);
}
```

Some example problems

- Primality testing: determine whether a number n is prime or not
  - Candidates 1, 2, 3, 4 ………. n
    - Increment a counter whenever you get a candidate which is a factor
    - Prime numbers always have two factors.

```
int main(){
        int i,n,c=0;
        scanf("%d",&n);
        for(i = 1; i <= n; i++){
                if(n%i == 0)
                        c++;
        }
        if(c == 2)
                printf("Prime Number");
        else     printf("Not a Prime Number");
}
```

Some example problems

- **Primality testing: determine whether a number n is prime or not**
  - **Candidates 1, 2, 3, 4 ………. n**
    - **Increment a counter whenever you get a candidate which is a factor**
    - **Prime numbers always have two factors.**

```
int main(){
        int i,n,c=0;
        scanf("%d",&n);
        for(i = 1; i*i <= n; i++){
                if(n%i == 0)
                        c++;
        }
        if(c == 1 && n != 1)
                printf("Prime Number");
        else     printf("Not a Prime Number");
}
```

Increase efficiency by going up to the square root

Some example problems

- Perfect number testing: determine whether a number n is perfect or not
  - If a number can be made out of its factors
  - For example 6 → 1, 2, 3 → 1+ 2 +3 = 6
  - Another example 28 → 1,2,4,7,14 → 1+2+4+7+14
  - Candidates 1, 2, 3, 4 ………. n
    - Add to sum whenever you get a candidate which is a factor

```
int main(){
        int i,n,s=0;
        scanf("%d",&n);
        for(i = 1; i < n; i++){
                if(n%i == 0)
                        s = s + i;
        }
        if(s == n)
                printf("Perfect Number");
        else     printf("Not a Perfect Number");
}
```

Some example problems

- GCD of two numbers (Normal way)
  - GCD(24,54) = 6
  - Factors of 24 → 1, 2, 3, 4, 6, 8, 12, 24
  - Factors of 54 → 1, 2, 3, 6, 9, 18, 27, 54
  - Common Factors 1, 2, 3, 6
  - Greatest Common Factor 6

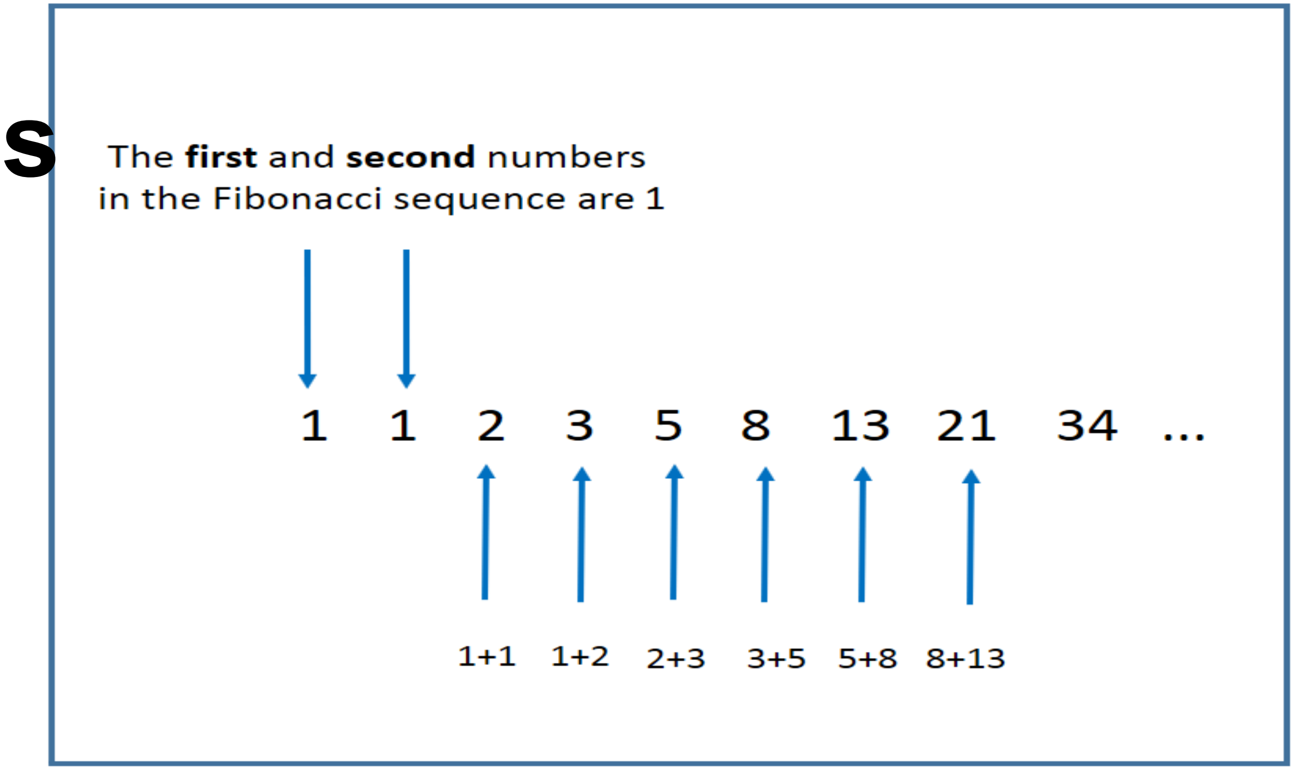
```
int main(){
        int i,a,min,b,gcd=1;
        scanf("%d%d",&a,&b);
        if(a == 0 || b == 0) gcd = a+b;
        else{
                min = (a < b)? a : b;
                for(i = 1; i <= min; i++){
                        if(a%i == 0 && b%i == 0)
                                gcd = i;
                }
        }
        printf("GCD: %d",gcd);
}
```

Some example problems

- GCD of two numbers (Efficient way)
  - gcd(a,b) = gcd (b, a%b) for b > 0
  - gcd(54,24) → gcd(24,6) → gcd(6,0) → 6

```
int main(){
       int i,a,b;
       scanf("%d%d",&a,&b);
       while(b != 0){
                      c = a%b;
                      a = b;
                      b = c;
       }
       printf("GCD: %d", a);
}
```

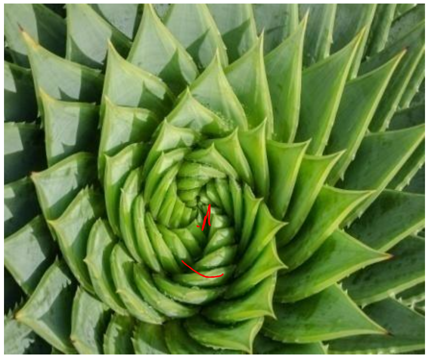
Fibonacci Series

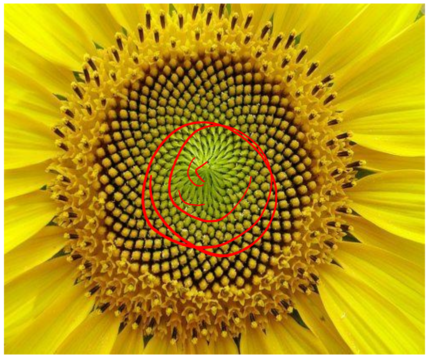


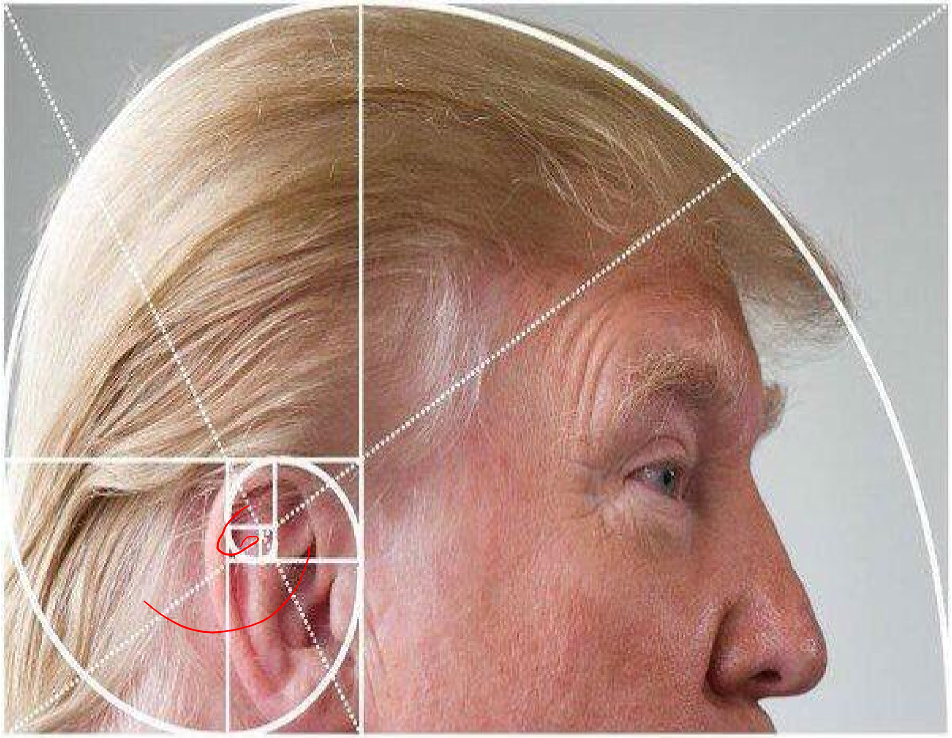
The **first** and **second** numbers in the Fibonacci sequence are 1

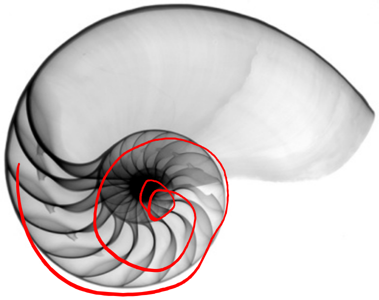
1 1 2 3 5 8 13 21 34 ...

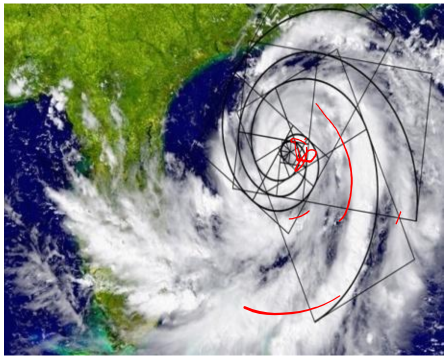
1+1 1+2 2+3 3+5 5+8 8+13





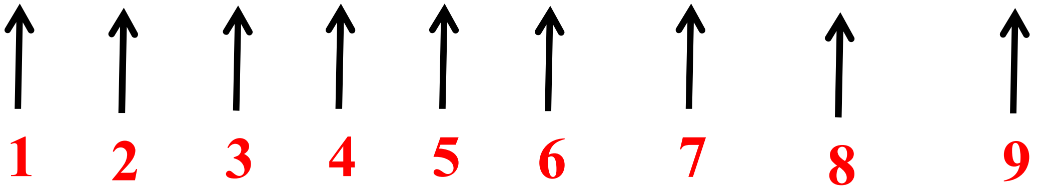






Fibonacci Series Generation

1, 1, 2, 3, 5, 8, 13, 21, 34, 55 ……….

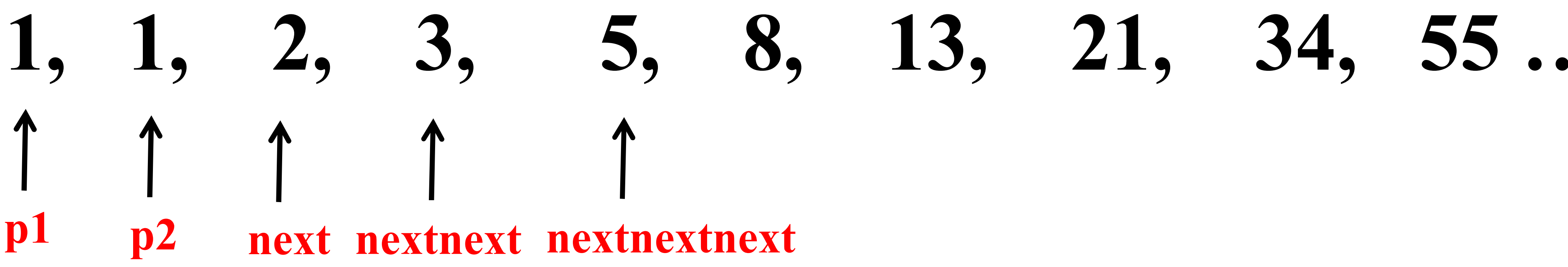


Write down a program that will print n-th Fibonacci number where n will be input to your program.

n = 4 output → 3

n = 7 output →13

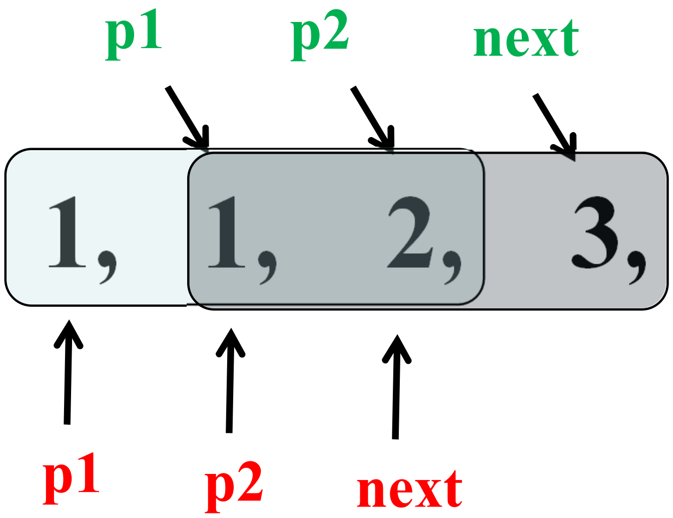
Fibonacci Series Generation



1, 1, 2, 3, 5, 8, 13, 21, 34, 55 ..

p1 p2 next nextnext nextnextnext

```
int main(){
        int p1,p2,next,n;
        scanf("%d",&n);
        p1 = 1;
        p2 = 1;
        next = p1 + p2;
        nextnext = p2 + next;
        nextnextnext = next + nextnext;
        …. .
        ….
}
```



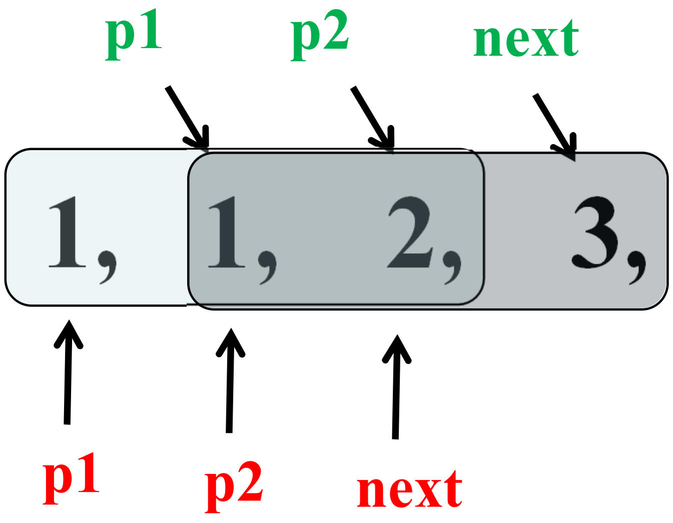
p1 p2 next

1, 1, 2, 3,

p1 p2 next

5, 8, 13, 21, 34, 55 ..

```
int main(){
        int p1,p2,next,n;
        scanf("%d",&n);
        p1 = 1;
        p2 = 1;
        for(i = ; i <= ; i++){
                next = p1 + p2;
                p1 = p2;
                p2 = next;
        }
```



p1 p2 next

1, 1, 2, 3,

p1 p2 next

5, 8, 13, 21, 34, 55 ..

```
int main(){
        int p1,p2,next,n;
        scanf("%d",&n);
        p1 = 1;
        p2 = 1;
        for(i = 3; i <= n; i++){
                next = p1 + p2;
                p1 = p2;
                p2 = next;
        }
        if(n <= 2) printf("%d", p1);
        else    printf("%d", next);
}
```

The for Statement

- **Each expression in the header of a `for` loop is optional**
- **If the initialization is left out, no initialization is performed**
- **If the condition is left out, it is always considered to be true, and therefore creates an infinite loop**
- **If the increment is left out, no increment operation is performed**

Infinite Loops

- **The body of a `while` loop eventually must make the condition false**

- **If not, it is called an *infinite loop*, which will execute until the user interrupts the program**

- **This is a common logical error**

- **You should always double check the logic of a program to ensure that your loops will terminate normally**

Infinite Loops

- An example of an infinite loop:

```
int count = 1;
while (1 == 1){
   printf (“%d\n”, count);
   count = count - 1;
}
```

```
int count = 1;
for(; ;){
   printf (“%d\n”, count);
   count = count - 1;
}
```

- This loop will continue executing until interrupted (Control-C) or until an underflow error occurs

The do Statement

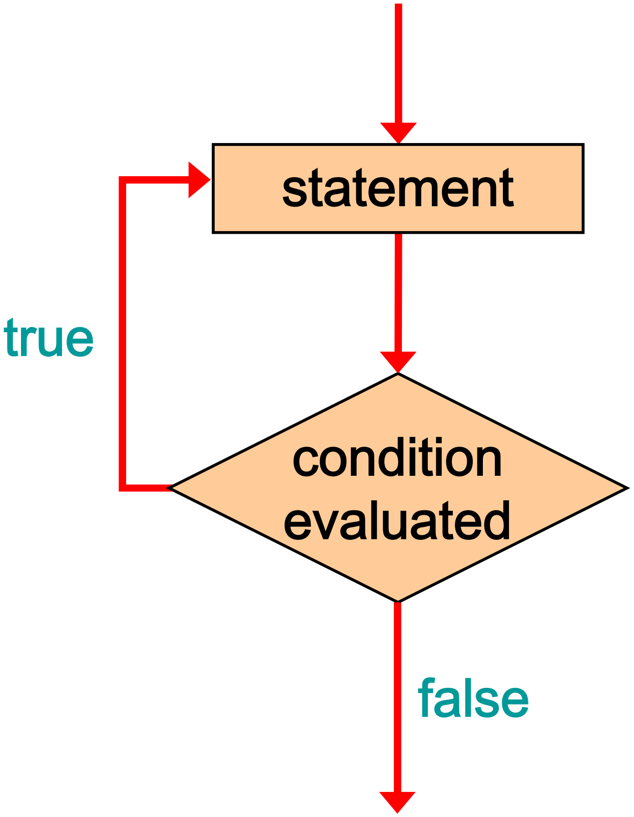
- A *do statement* has the following syntax:

```
do{
   statement;
}
while ( condition );
```

- The `statement` is executed once initially, and then the `condition` is evaluated

- The statement is executed repeatedly until the condition becomes false

Logic of a do-while Loop



The do Statement

- **An example of a `do` loop:**

```
int count = 1;
do{
   printf(“%d\n”, count);
   count++;
} while (count <= 5);
```

- **The body of a `do` loop is executed at least once**

The do Statement

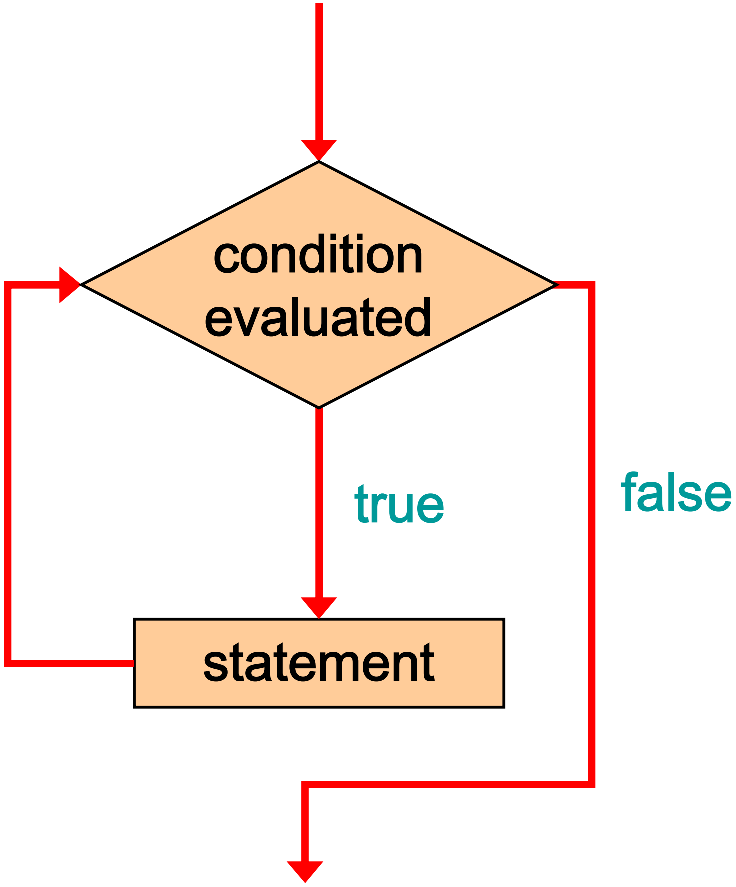
- **An example of a `do` loop:**

```
int n;
do{
   printf(“Enter a positive number: ”);
   scanf(“%d”,&n);
} while (n < 0);
```

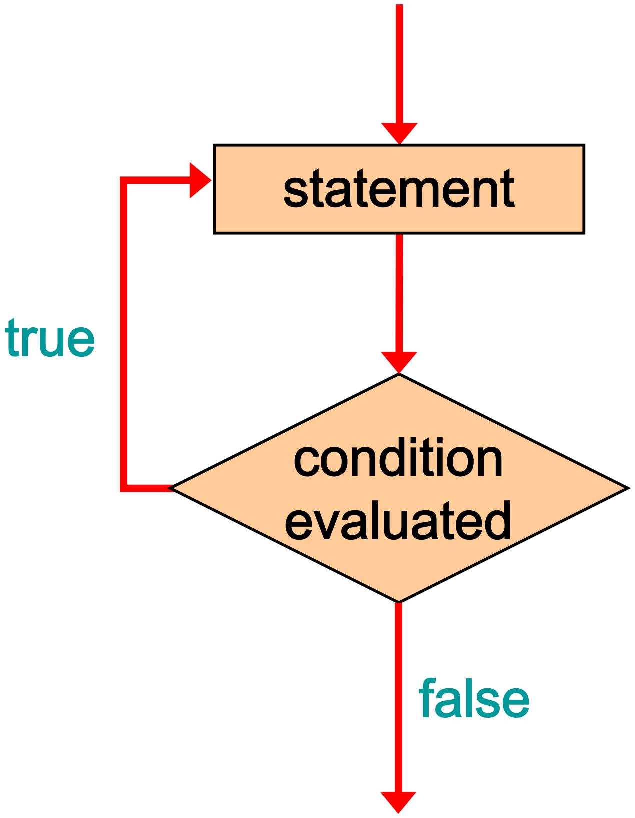
- **The body of a `do` loop is executed at least once**

Comparing while and do

The while Loop



The do Loop



Example: Printing reverse of a number

- **Write down a program that prints the digits of a number in reverse.**
- **For example:**
- **input: 6457**
- **output: 7546**

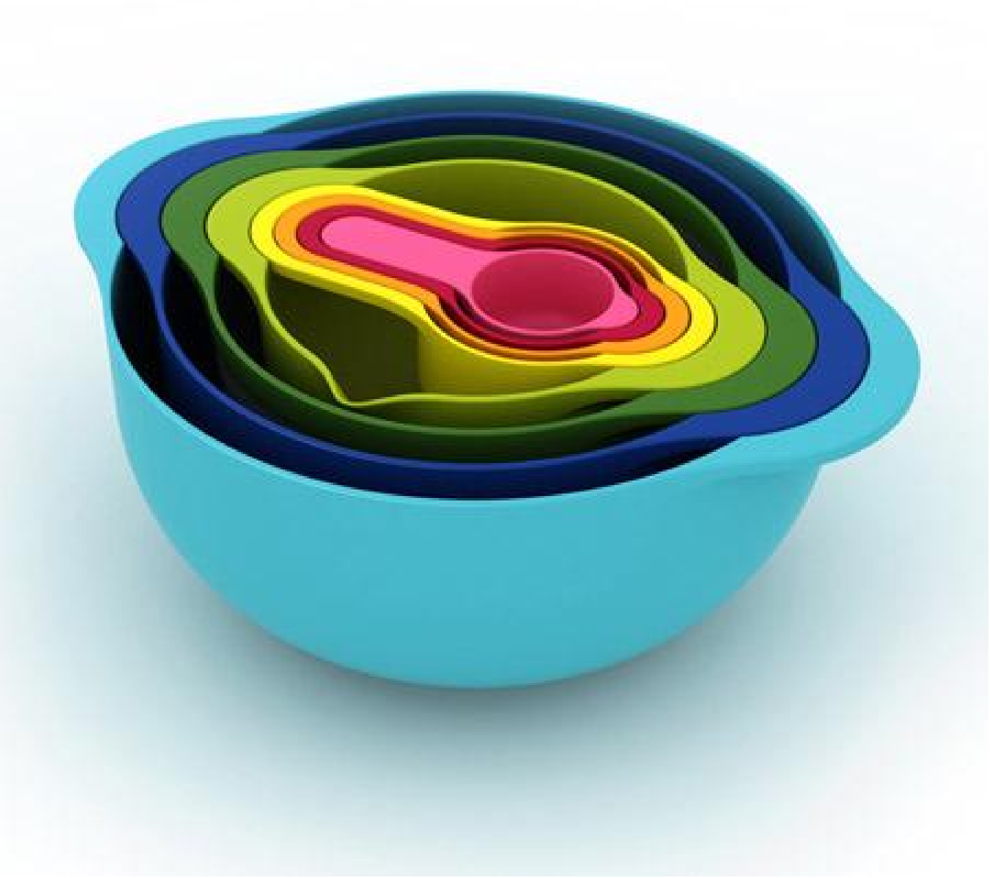
```
scanf(“%d”,&n);
do{
      a = n%10;
      printf(“%d”,a);
      n = n/10;
} while (n != 0);
```

Relevant Problem: counting number of digits of a number

- **Write down a program that prints number of digits of a number *n*.**
- **For example:**
- **input: 6457**
- **output: 4**

```
scanf(“%d”,&n);
c = 0;
do{
      n = n/10;
      c++;
} while (n != 0);
printf(“%d”,c);
```

Nested Loops



Nested Loops

- **Similar to nested `if` statements, loops can be nested as well**
- **That is, the body of a loop can contain another loop**
- **For each iteration of the outer loop, the inner loop iterates completely**

Nested Loops

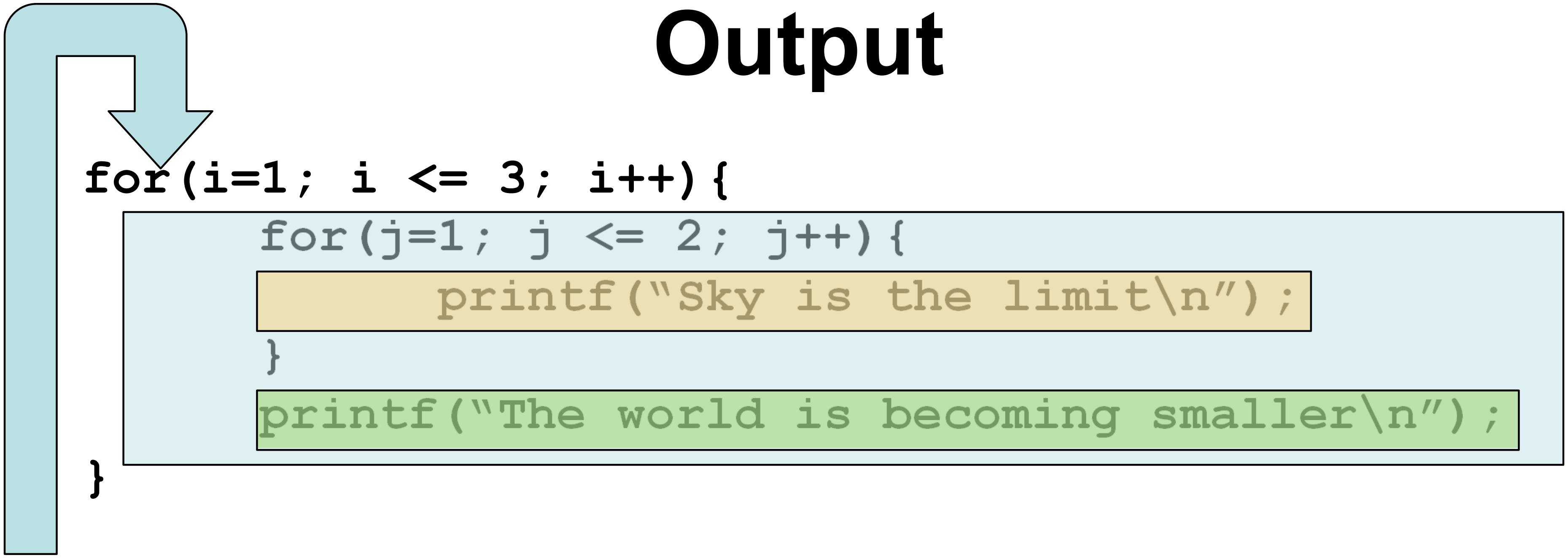
- **What will be the output?**

Outer loop

Inner loop

```
for(i=1; i <= 3; i++){
      for(j=1; j <= 2; j++){
            printf("Sky is the limit\n");
      }
      printf("The world is becoming smaller\n");
}
```

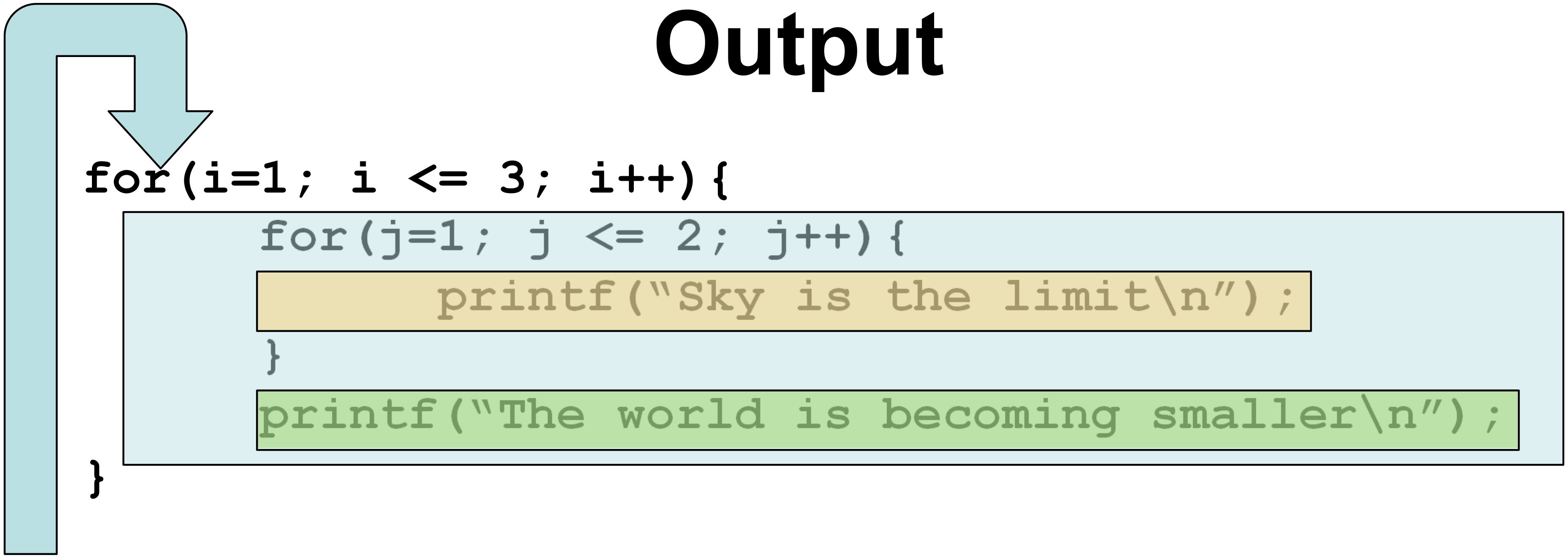
Output



```
for(i=1; i <= 3; i++){
        for(j=1; j <= 2; j++){
                printf(“Sky is the limit\n”);
        }
        printf(“The world is becoming smaller\n”);
}
```

```
Sky is the limit
Sky is the limit
The world is becoming smaller
```

Output



```
for(i=1; i <= 3; i++){
        for(j=1; j <= 2; j++){
              printf("Sky is the limit\n");
        }
        printf("The world is becoming smaller\n");
}
```

```
Sky is the limit
Sky is the limit
The world is becoming smaller
Sky is the limit
Sky is the limit
The world is becoming smaller
```

Output

```
for(i=1; i <= 3; i++){
     for(j=1; j <= 2; j++){
          printf(“Sky is the limit\n”);
     }
     printf(“The world is becoming smaller\n”);
}
```

```
Sky is the limit
Sky is the limit
The world is becoming smaller
Sky is the limit
Sky is the limit
The world is becoming smaller
Sky is the limit
Sky is the limit
The world is becoming smaller
```

Output

```
for(i=1; i <= 3; i++){
      for(j=1; j <= 2; j++){
            printf(“%d %d\n”,i,j);
      }
}
```

```
1 1
1 2
2 1
2 2
3 1
3 2
```

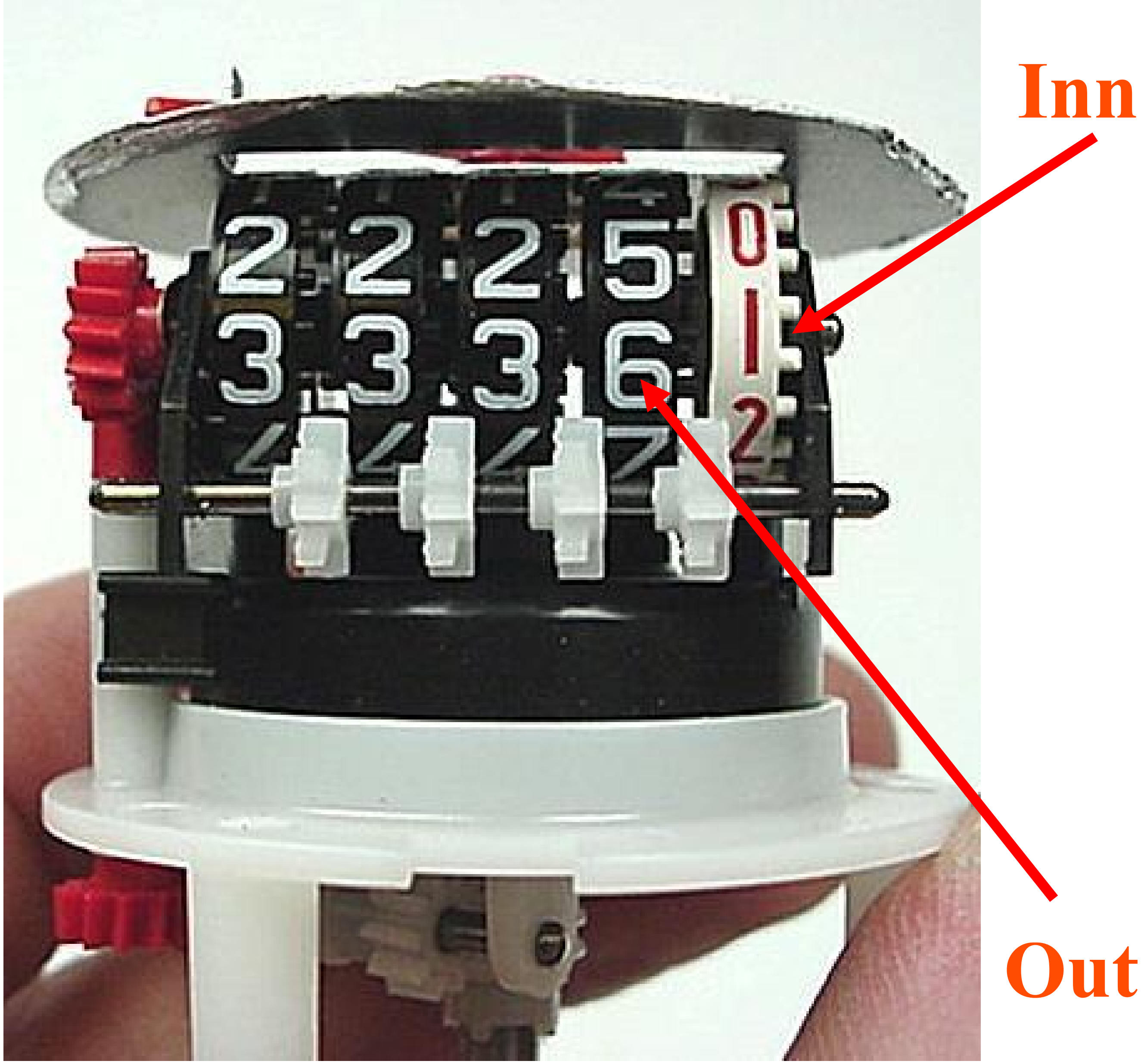
Nested Loops

- How many times will the string `"Here"` be printed?

```
count1 = 1;
while (count1 <= 10)
{
   count2 = 1;
   while (count2 <= 20)
   {
      printf ("Here \n");
      count2++;
   }
   count1++;
}
```

10 * 20 = 200

Analogy for Nested Loops



Inner Loop

Outer Loop

Some more Examples

- Write a program that prints all prime numbers up to *x*. The integer x will be input to your program.

- Write down a program that will take an integer x as input and will count and print the number of prime numbers up to x.

- Write a program that prints all perfect numbers up to *x*. The integer x will be input to your program.

- Write a program that prints all prime factors of a number x given as input.

Example: Stars

- Write a program that prints the following. The total number of lines will be input to your program.

```
*
**
***
****
*****
******
*******
********
*********
**********
```

Example: Stars

- Write a program that prints the following. The total number of lines will be input to your program.

```
    *
   **
  ***
 ****
*****
```