Programming and Data Structures - I

Lecture 9

Kripabandhu Ghosh

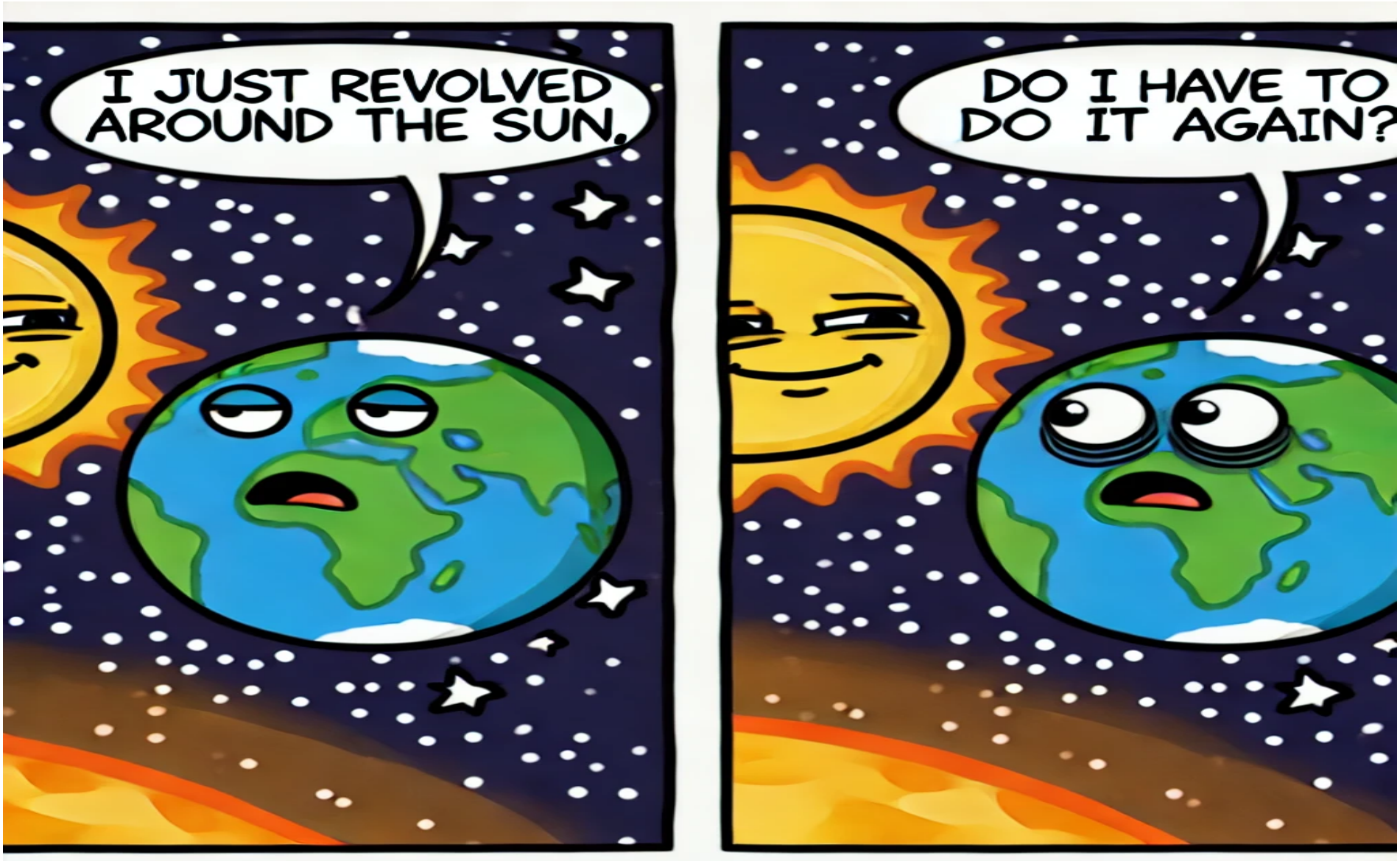
CDS, IISER Kolkata

DECISION MAKING AND LOOPING

Repeated execution of instructions[1]

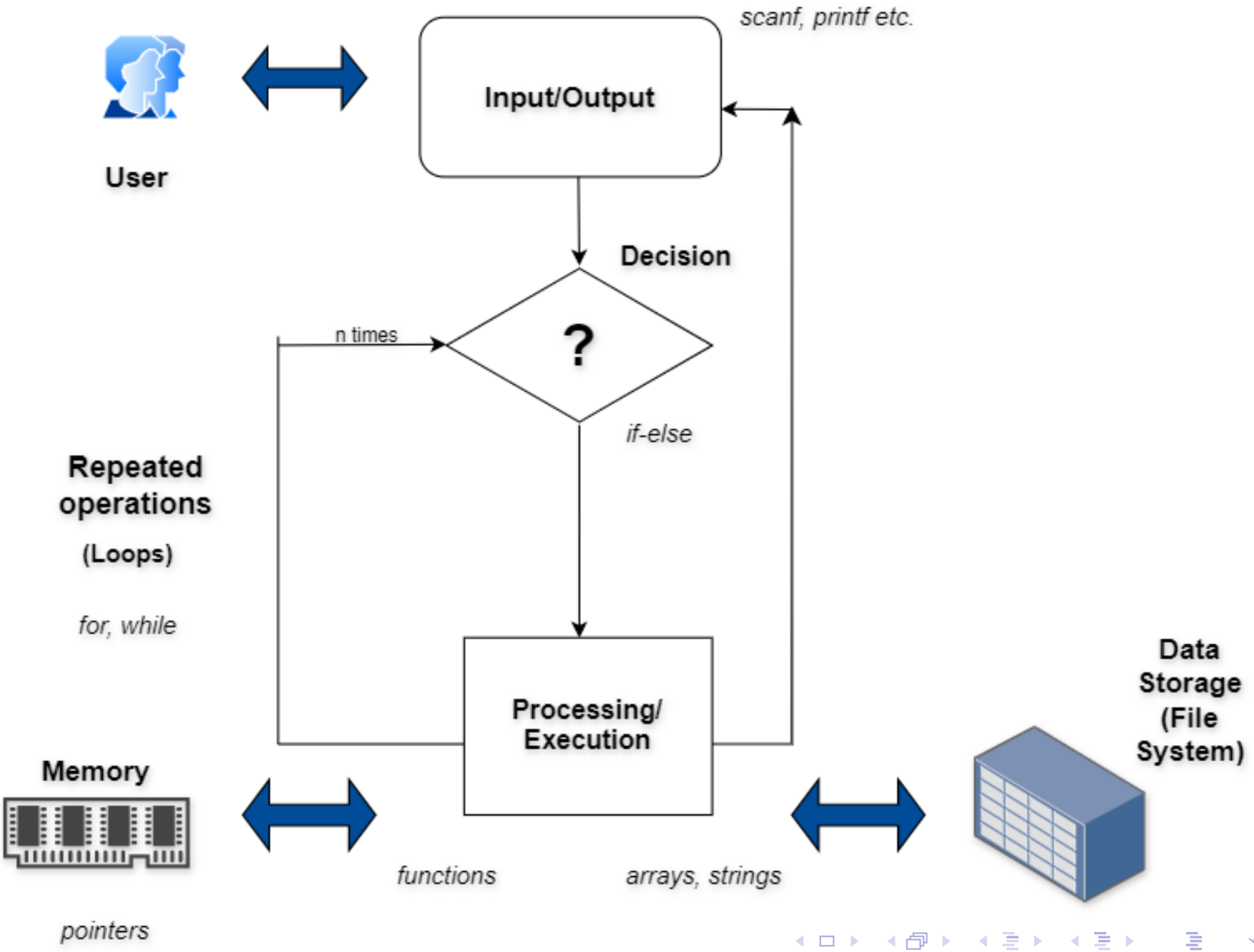
[1] Image generated by ChatGPT

Repeated execution of instructions[2]



[2]Image generated by ChatGPT

Bigger Picture



Loops

Features

- Execute statement(s) (*body*) multiple times while some *condition* is True
- *condition* : position can be at the start (entry control) or at the end (exit control) of the *body*

Condition

Entry control

1. *condition* is first checked
2. *body* is executed only if *condition* is True
3. *body* is executed as long as *condition* is True

Examples

for, while

Condition

Exit control

1. *body* is first executed
2. *condition* is then checked
3. *body* is executed as long as *condition* is True

Example

do-while

while loop

format

```
while(condition)
{
   body
}
```

- *condition* can be a relational (e.g. $a > 5$) or logical (e.g. $a > 5$ && $a <= 10$) expression (*body* executed only if this is True)
- *body* contains zero or more number of statements
- entry-controlled loop

while loop: example

Program

```
#include<stdio.h>

int main()
{
   int a = 5;
   while(a >= 5) //Condition
   {
      printf("Loop executed!\n");
      a = a - 1;
   }
   return 0;
}
```

Output

```
Loop executed!
```

do-while loop

format

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

- *condition* can be a relational (e.g. $a > 5$) or logical (e.g. $a > 5$ && $a <= 10$) expression (checked after *body* has executed once)
- *body* contains zero or more number of statements
- exit-controlled loop (*body* executed et least once)

do-while loop: example

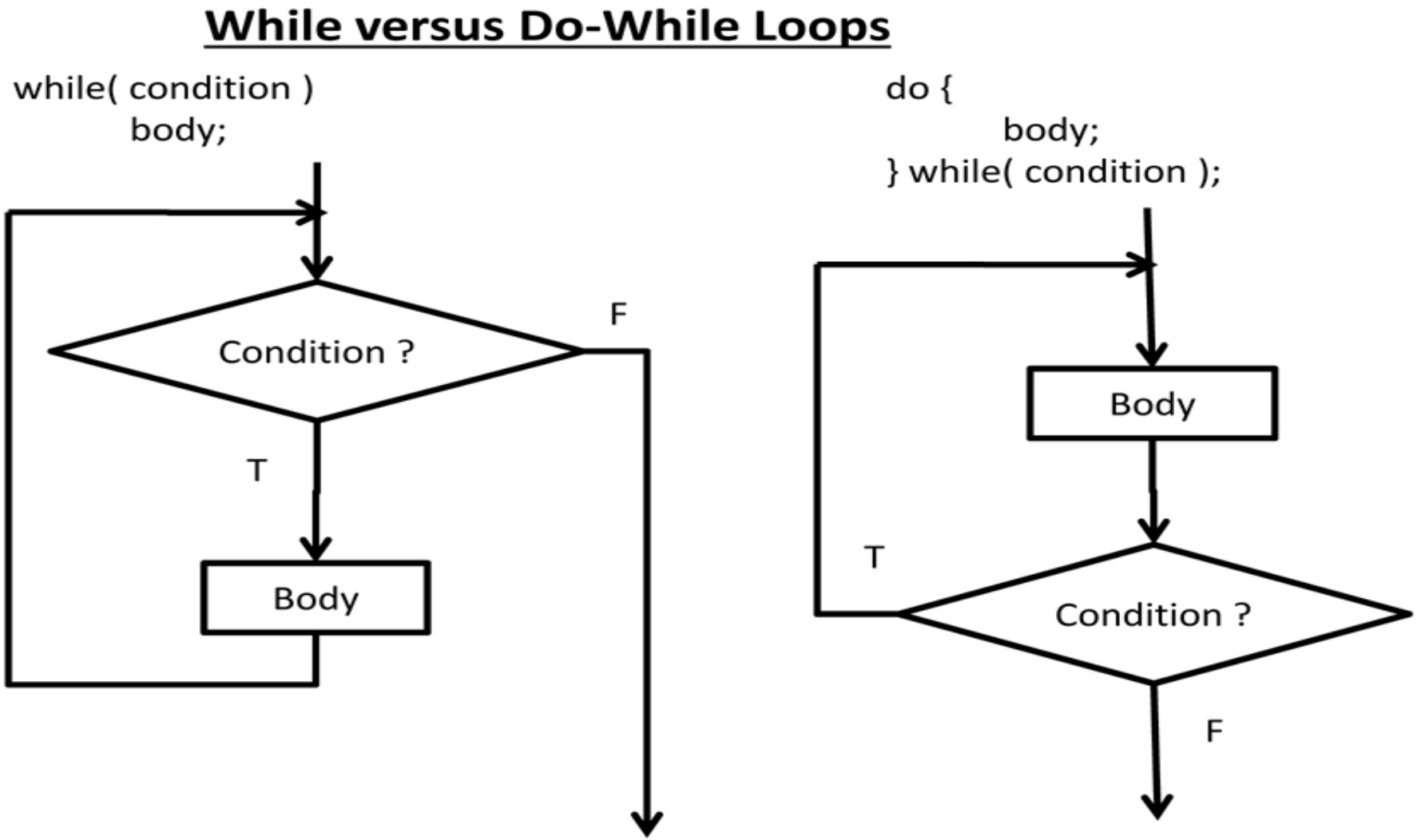
Program

```
#include<stdio.h>

int main()
{
   int a = 5;

   do
   {
      printf("Loop executed!\n");
      a = a - 1;
   }
   while(a >= 5); //Condition
   return 0;
}
```

Output

```
Loop executed!
```

While Vs Do-while[3]



[3] https://www.cs.uic.edu/~jbell/CourseNotes/C_Programming/Looping.html

for loop

format

for(initialization; condition; increment)
{
  body
}

- *initialization*: control variable(s) initialized (done only once) (e.g. $a = 5$)
- *condition* can be a relational (e.g. $a > 5$) or logical (e.g. $a > 5$ && $a <= 10$) expression (*body* executed only if this is True)
- *increment*: control variable(s) incremented (decremented) after *body* is executed (e.g. $a = a + 1$)
- *body* contains zero or more number of statements
- entry-controlled loop

Advantage

More organized loop control structure

for loop: example

Program

```
#include<stdio.h>

int main()
{
   int a;
   for(a = 5 ; a >= 5 ; a = a - 1)
   {
      printf("Loop executed!\n");
   }
   return 0;
}
```

Output

```
Loop executed!
```

for loop: multiple initializations and increment/decrement

Program

```
#include<stdio.h>

int main()
{
   int a;
   for(a = 5, b = 6 ; a <= b ; a = a + 1, b = b - 1)
   {
      printf("Loop executed!\n");
   }
   return 0;
}
```

Output

```
Loop executed!
```

for loop: multiple initializations, increment/decrement, complex condition

Program

```
#include<stdio.h>

int main()
{
   int a;
   for(a = 5, b = 6 ; a <= b && b > 0; a = a + 1, b = b - 1)
   {
      printf("Loop executed!\n");
   }
   return 0;
}
```

Output

Loop executed!

for loop: optional initialization and increment/decrement

Program

```
#include<stdio.h>

int main()
{
   int a;
   a = 5;
   for( ; a >= 5 ; )
   {
      printf("Loop executed!\n");
      a = a - 1;
   }
   return 0;
}
```

Output

```
Loop executed!
```

for loop: nested loops

Program

```
#include<stdio.h>

int main()
{
   int i, j;
   for(i = 1 ; i <= 3 ; i = i + 1)
   {
      for(j = 1 ; j <= 2 ; j = j + 1)
      {
         printf("i = %d j = %d\n", i, j);
      }
   }
return 0;
}
```

Output

```
i = 1 j = 1
i = 1 j = 2
i = 2 j = 1
i = 2 j = 2
i = 3 j = 1
i = 3 j = 2
```

for loop: nested loops (contd.)

Program

```
#include<stdio.h>

int main()
{
   int i, j;
   for(i = 1 ; i <= 3 ; i = i + 1)
   {
      for(j = 1 ; j <= i ; j = j + 1)
      {
         printf("%d ", j);
      }
      printf("\n");
   }
return 0;
}
```

Output

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

Jumps in loops

- **break**: Jump out of the current loop
  - The current loop immediately terminates
  - The control goes to the statement immediately following the loop
- **continue**: The following sentences in the current loop are ignored and the next iteration is started

break : example

Program

```
#include<stdio.h>

int main()
{
   int i;
   for(i = 1 ; i <= 3 ; i = i + 1)
   {
      if(i == 3)
      {
         break;
      }
      printf("i = %d\n", i);
   }
   return 0;
}
```

Output

```
i = 1
i = 2
```

break : linear search

Program

```
#include<stdio.h>

int main()
{
   int a[5] = {1, 4, 7 , 2, 7}, key = 7, found = 0, i;
   for(i = 0 ; i < 5 ; i = i + 1)
   {
      if(a[i] == key)
      {
         printf("Found %d at position %d!\n", key, i);
         found = 1;
         break;
      }
   }
   if(found == 0)
   {
      printf("Search failed!\n");
   }
   return 0;
}
```

Output

```
Found 7 at position 2!
```

continue : example

Program

```
#include<stdio.h>
#include<math.h>

int main()
{
float a[5] = {-1.0, -4.0, -7.0 , 2.0, 7.0};
   int i;
   for(i = 0 ; i < 5 ; i = i + 1)
   {
      if(a[i] < 0)
      {
         continue;
      }
      printf("SQRT(%f) = %f\n", a[i], sqrtf(a[i]));
   }
   return 0;
}
```

Output

```
SQRT(2.000000) = 1.414214
SQRT(7.000000) = 2.645751
```

continue : *while* anomaly

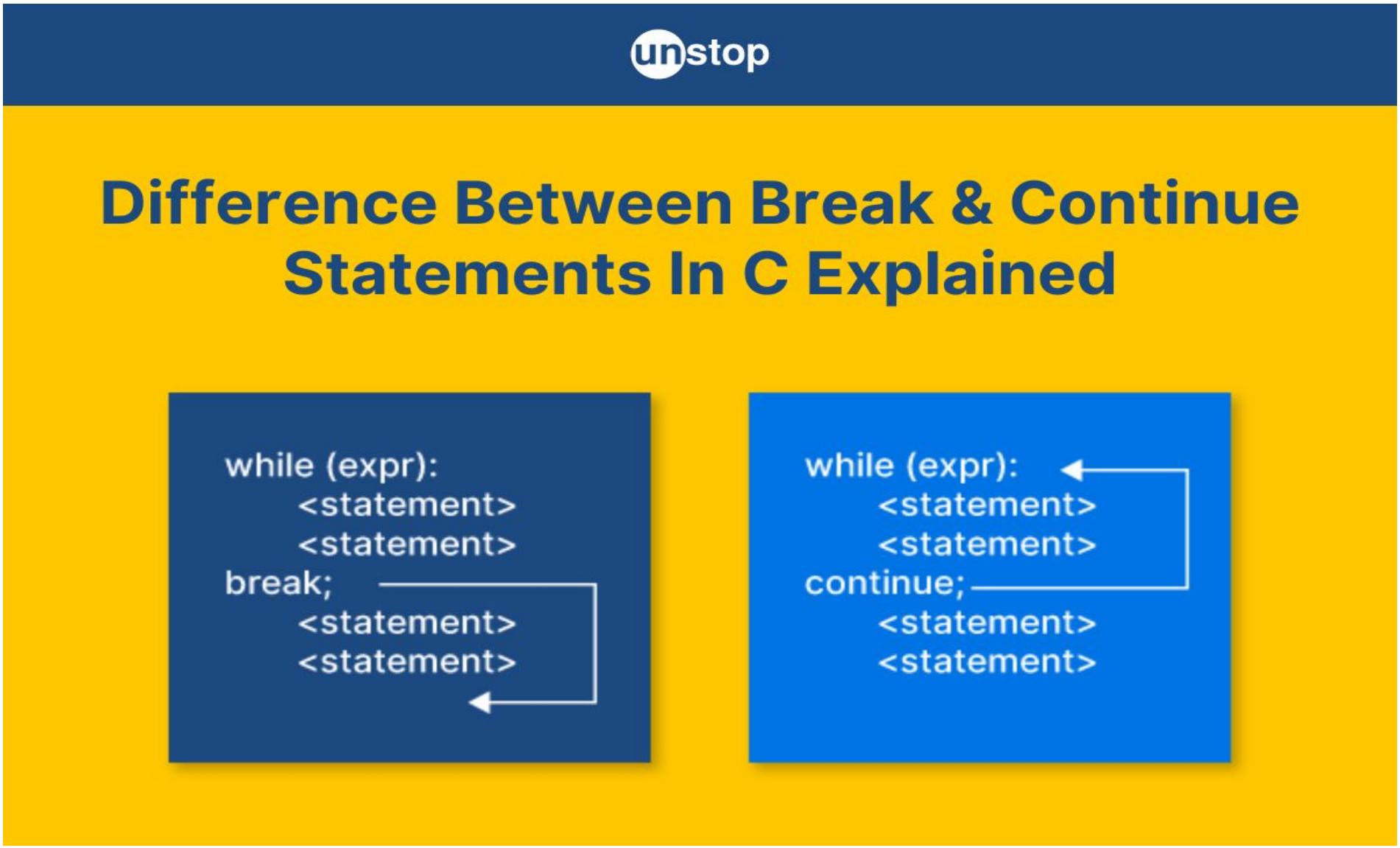
Program

```
#include<stdio.h>

int main()
{
   int i = 2;
   while(i < 5)
   //for(i = 2 ; i < 5 ; i++)
   {
      printf("i=%d\n", i);
      if(i == 3)
      {
         continue;
      }
      i = i + 1;
   }
   return 0;
}
```

Output

```
i = 2
i = 3
i = 3
i = 3 ...(infinite loop)
```

Break Vs Continue[4]



[4] https://unstop.com/blog/difference-between-break-and-continue

Pascal's Triangle

Coefficients of Binomial expansion

$$(a+b)^n = {}^nC_0 a^{n-0} b^0 + {}^nC_1 a^{n-1} b^1 + \ldots + {}^nC_n a^{n-n} b^n$$
$$= a^n + na^{n-1}b + \ldots + b^n$$

Coefficients of Binomial expansion (n = 2)

$$(a+b)^2 = a^2 + 2ab + b^2$$

Coefficients of Binomial expansion (n = 3)

$$(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$$

Pascal's Triangle (contd.)

$$
\begin{array}{ccccccccc}
 & & & & 1 & & & & (n = 0) \\
 & & & 1 & & 1 & & & (n = 1) \\
 & & 1 & & 2 & & 1 & & (n = 2) \\
 & 1 & & 3 & & 3 & & 1 & (n = 3) \\
1 & & 4 & & 6 & & 4 & & 1 \quad (n = 4)
\end{array}
$$

Useful formulae (ith row, jth column)

- ${}^0C_0 = 1$, ${}^iC_0 = 1$, ${}^iC_i = 1$ (i <= 1, j = 0, j = i)
- ${}^iC_j = {}^{i-1}C_{j-1} + {}^{i-1}C_j$ (i > 1, 0 < j < i)

Useful formulae (ith row, jth column)

- ${}^0C_0 = 1$, ${}^iC_0 = 1$ (i = 0, j = 0)
- ${}^iC_{j-1}$ x $\frac{i-j+1}{j} = {}^iC_j$ (i > 0, j > 0)

Pascal's Triangle (contd.)

Program

```
#include <stdio.h>

int main() {
   int no_of_rows = 5, coef = 1, s, i, j;
   for (i = 0; i < no_of_rows; i++) {
      for (s = 1; s <= no_of_rows - i; s++)
         printf(" ");
      for (j = 0; j <= i; j++) {
         if (j == 0 || i == 0)
            coef = 1;
         else
            coef = coef * (i - j + 1) / j;
         printf("%4d", coef);
      }
      printf("\n");
   }
   return 0;
}
```


THANK YOU