Conditional Loops

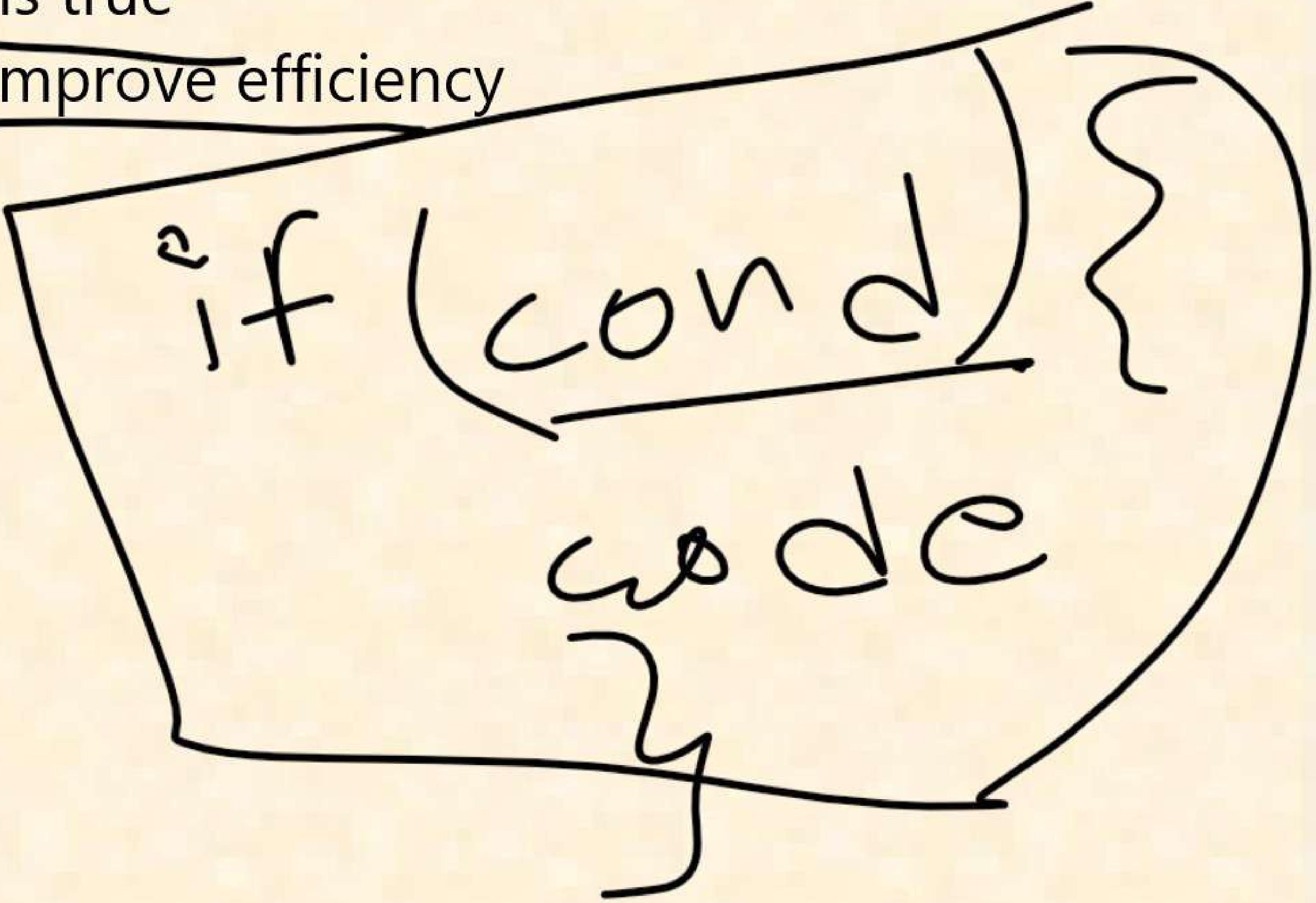
What are Conditional Loops?

Conditional loops repeat a block of code as long as a condition is true.

- A loop is a control structure that repeats a block of code
- Repetition continues until a condition is true
- Used to reduce code duplication and improve efficiency

Why Use Loops?

- Avoid writing repetitive code
- Save time and effort
- Make programs easier to maintain
- Handle large data efficiently



Types of Loops

1. For Loop
2. While Loop
3. do-While Loop
4. For-each Loop
5. Nested Loop
6. Infinite Loop

For Loop

- Used when the number of iterations is **known**
- Has initialization, condition, and update
- Condition is checked **before** each iteration

Example Use:

Printing numbers from $\boxed{\text{1 to 10}}$

for

10 times

Syntax of FOR Loop:-

```
for (initialization; condition; update)
{
    // statements
}
```

i = 0 → start

1 2 3 4

i <= 10 → true

i++;

While Loop

- Used when the number of iterations is **unknown**
- Condition is checked **before** execution
- Loop may run **zero times**

Example Use:

Reading input until user enters a valid value

Exit=0

while(i != 0){

int i = input

1 3 0

Syntax of While Loop:-

```
while (condition)
{
    // statements
}
```

code

i++ ;

Do-While Loop

- Condition is checked **after** execution
- Loop runs **at least once**
- Ensures minimum one execution

Example Use:

Menu-driven programs

Syntax of do-While Loop:-

```
do
{
   // statements
}
while (condition);
```

code

Difference between while and do-while loops

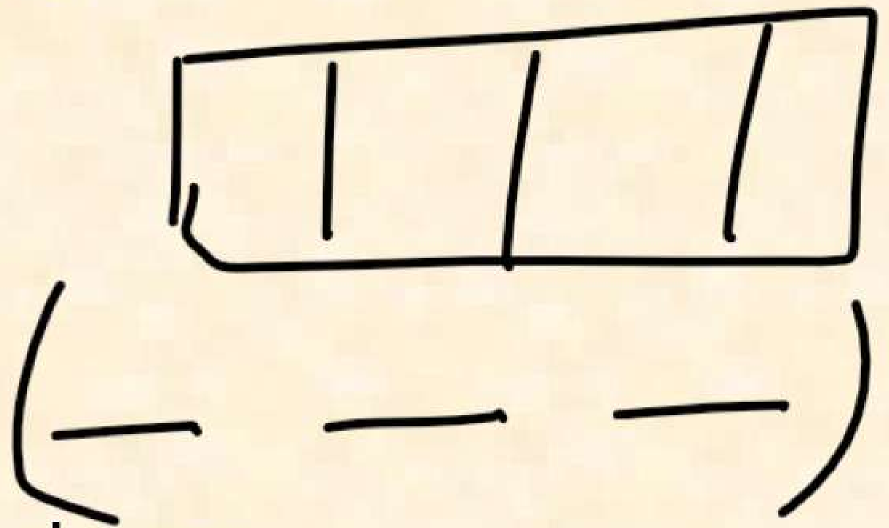
Feature	While Loop	do-while Loop
Condition checking	Before loop execution	After loop execution
Minimum execution	May execute **zero times**	Executes **at least once**
Loop type	Entry-controlled loop	Exit-controlled loop
Syntax structure	while(condition) { }	do { } while(condition);
Semicolon usage	No semicolon after condition	Semicolon required after condition
Best used when	Condition must be checked first	Code must run at least once
Example use	Input validation	Menu-driven programs
Control flow	Condition → Body	Body → Condition

For-Each Loop

- A loop used to iterate through **collections** or **arrays**
- Automatically accesses each element one by one
- No need for index or counter variable
-

When to Use For-Each Loop?

- When the number of elements is **known**
- When only **reading** elements, not modifying indexes
- Works best with arrays, lists, and collections



How For-Each Loop Works

- Takes one element at a time from the collection
- Repeats until all elements are processed
- Stops automatically at the last element

Syntax of While Loop:-

```
for (dataType variable : collection)
{
   // statements
}
```

Nested Loop

- A loop inside another loop
- Inner loop completes fully for each outer loop iteration

Example Use:

- Printing patterns
- Working with tables or matrices

```
for{
while(){
} }
```

Example of Nested Loop:-

```
for (initialization; condition; update)
{
    while (condition)
    {
        // statements
    }
}
```

Infinite Loop

- Loop that never ends
- Condition always remains true
- Usually caused by programming errors

Example:

- Missing condition update

Note: Can crash or freeze programs

Example of Infinite Loop

```
while (true)
{
   // statements
}
```

(;;)

```
for (;;)
{
   // statements
}
```

code

Transfer Statements

Transfer statements are used to **change the normal flow of program execution**. They transfer control from one part of the program to another.

1.break Statement

Used to **terminate a loop or switch statement immediately**.

Syntax:

```
break;
```

2. continue Statement

Used to **skip the current iteration** of a loop and move to the next iteration.

Syntax:

```
continue;
```

skip

3. return Statement

Used to **exit from a method** and optionally return a value to the calling method.

return method

Syntax:

```
return;
return value;
```