Loops

Recap of Decision Making

Decision making helps the computer decide what to do based on certain conditions.

Yayyy, It's Your Birthday

Condition:

Give chocolate to friend,
if available in Plate

Introducing Loops as Repeated Decisions:

Now there are 100 Friends.....

Condition: Until chocolates are available in the plate, Distribute them

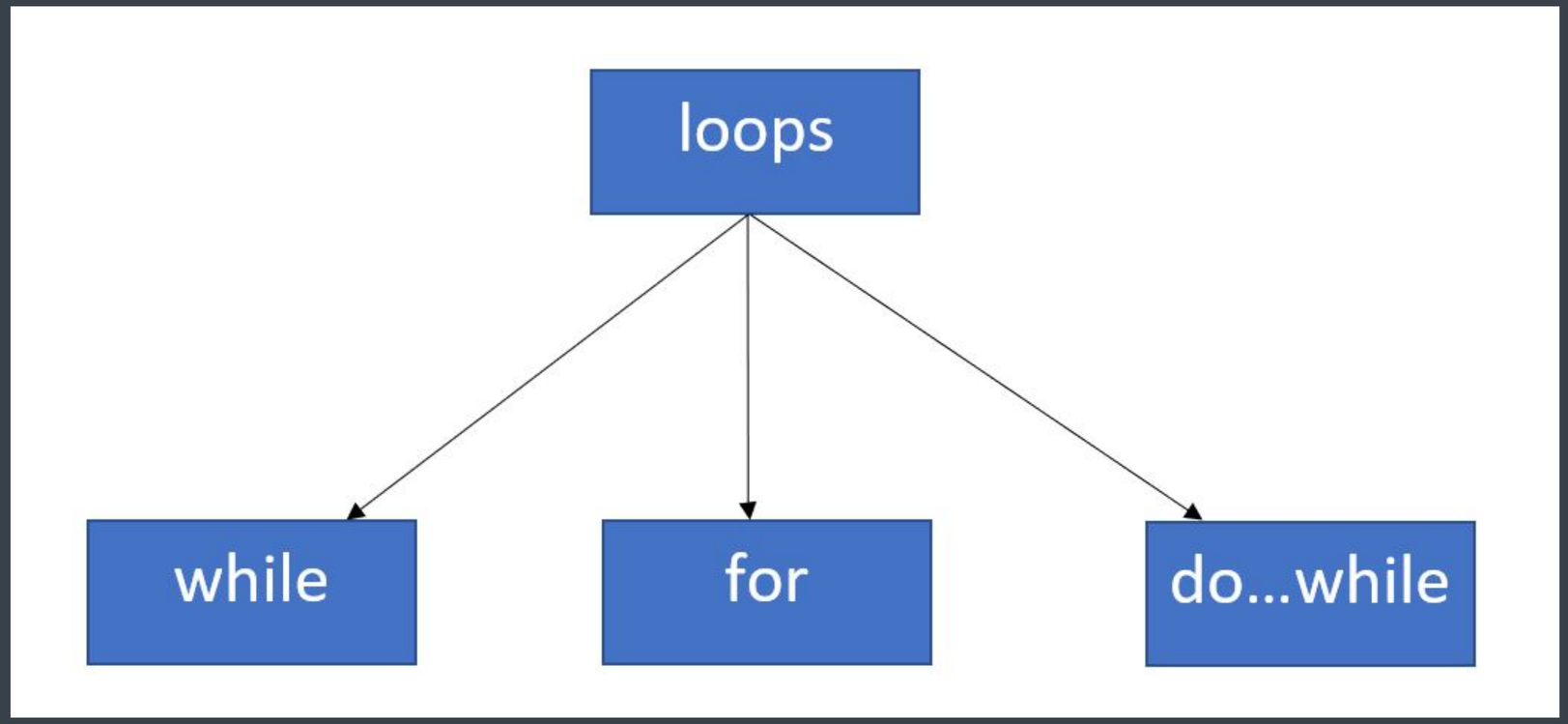
Loops

Loops are used to execute a block of code repeatedly as long as a certain condition is true or for a specific number of iterations

Types

- while
- do-while
- for

Loops



While loop

The while loop executes a block of code as long as a specified condition is true. It continuously checks the condition before each iteration and stops when the condition becomes false.

Syntax:

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

While loop

```c
#include <stdio.h>
int main() {
    int i = 1;
    while (i <= 5) {
        printf("%d\n", i);
        i++;
    }
    return 0;
}
```

```
1
2
3
4
5
```

do-while loop

The do-while loop is used when you want to execute a block of code at least once and then repeatedly as long as a condition is true.

Syntax:

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

do-while loop

```c
#include <stdio.h>

int main() {
    int i = 1;
    do {
        printf("%d\n", i);
        i++;
    } while (i <= 5);

    return 0;
}
```

```
1
2
3
4
5
```

for loop

The for loop is commonly used when you know the number of iterations in advance. It consists of three parts: initialization, condition, and update.

Syntax:

```
for (initialization; condition; update) {
// Code to be executed repeatedly
 }
```

for loop

```c
#include <stdio.h>
int main() {
    for (int i = 1; i <= 5; i++) {
        printf("%d\n", i);
    }
    return 0;
}
```

```
1
2
3
4
5
```

Infinite Loops

An infinite loop is a loop that continues to execute indefinitely without ever terminating. In programming, this usually occurs when the loop's termination condition is not met or when the loop's control flow prevents it from reaching a termination point.

Infinite Loops

```
for (;;) {
    // This loop will run forever
}
```

```
while (1) {
    // This loop will run forever
}
```

Nested loops

Nested loops refer to the situation where one loop is placed inside another loop. This allows you to execute a set of instructions repeatedly

Syntax:

```
for (initialization1; condition1; update1) {
    for (initialization2; condition2; update2) {
        // Inner loop statements
    }
}
```

Nested loops

```c
#include <stdio.h>
int main() {
    for (int i = 1; i <= 5; i++) {
        for (int j = 1; j <= 5; j++) {
            printf("%d x %d = %d\n", i, j, i * j);
        }
    }
    return 0;
}
```

Break

If during the execution of the loop C interpreter encounters break, it immediately stops the loop execution and exits out of it.

Syntax:

```
break;
```

Break

```c
for (int i = 1; i <= 10; i++) {
    if (i == 5) {
        break; // Exit the loop when i is 5
    }
    printf("%d\n", i);
}
```

Continue

Continue statement is used to skip the rest of the current iteration in a loop and move to the next iteration immediately.

Syntax:

```
continue;
```

Continue

```c
for (int i = 1; i <= 5; i++) {
    if (i == 3) {
        continue; // Skip iteration when i is 3
    }
    printf("%d\n", i);
}
```

Problems On

Loops +
Strings +
Numbers +
Decision Making

Print numbers from 1 to N

Take a positive integer N as input and print all the numbers from 1 to N.

Sample Input: **N = 5**

Sample Output:

1
2
3
4
5

Calculate the sum of N natural numbers

Take a positive integer N as input and calculate the sum of the first N natural numbers.

Sample Input: **N = 5**

Sample Output: **Sum of first 5 natural numbers: 15**

Print even numbers from 1 to N

Take a positive integer N as input and print all the even numbers from 1 to N.

Sample Input: N = 10

Sample Output:

2
4
6
8
10

Print odd numbers from 1 t num

Take a positive integer N as input and print all the odd numbers from 1 to N.

Sample Input: N = 10

Sample Output:

1
3
5
7
9

Multiplication table of a number

Take a positive integer N as input and print the multiplication table of N from 1 to 10.

Sample Input:

N = 3

Sample Output:

Multiplication table of 3:
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30

Calculate the factorial of a number

Take a positive integer N as input and calculate its factorial (N!).

Sample Input: **N = 5**

Sample Output: **Factorial of 5: 120**