Lecture 5

Program Looping

Core Concept

Looping allows a program to repeat a block of statements multiple times.

This is essential for efficiency and reducing code redundancy in any programming language, including C.

Classification of Loops

There are two primary categories of loops:

*** Unconditional Loops:**

Executed for a predefined/set number of times.

The count is known before the loop finishes.

تعديل من خلال WPS Office

*** Conditional Loops:**

The iterations depend on a specific logical condition.

The loop halts (stops) once the condition becomes true.

The number of iterations can vary each time the program runs.

Key Takeaway

The main difference lies in predictability: Unconditional loops have a fixed count, while conditional loops are dynamic and depend on real-time data or states.

تعديل من خلال WPS Office

for Loop

The for loop is C's form of an unconditional loop.

The basic syntax of the for statement is,

for (initialization expression;test expr;increment expr)

program statement;

an example:

sum=10;

for(i=0;i<6;++i)

sum =sum+i;

The operation for the loop is as follows:

1) The initialization expression is evaluated.

2) The test expression is evaluated.

If it is TRUE,body of the loop is executed.

If it is FALSE,exit the for loop.

تعديل من خلال WPS Office

3) Assume test expression is TRUE.

Execute the program statements making up the body of the loop.

4) Evaluate the increment expression and return to step 2.

5) When test expression is FALSE,exit loop and move on to next line of code.

Sample Loop

sum = 10;

for (i=0; i<6; ++i)

sum=sum+i;

We can trace the execution of the sample loop as follows

تعديل من خلال WPS Office

Iteration	*i*	*i<6*	*sum*
1st	0	TRUE	10
2nd	1	TRUE	11
3rd	2	TRUE	13
4th	3	TRUE	16
5th	4	TRUE	20
6th	5	TRUE	25
7th	6	FALSE	25

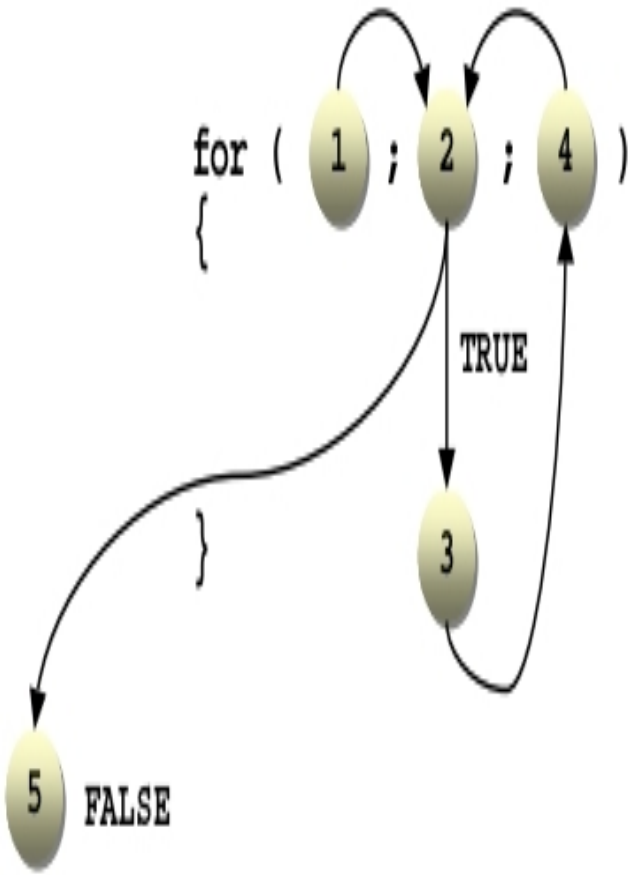
تعديل من خلال WPS Office

For loop diagram

for Loop Diagram

The following diagram illustrates the operation of a for loop

تعديل من خلال WPS Office



تعديل من خلال WPS Office

General Comments about for Loop

Some general comments regarding the use of the for statement:

-Control expressions are separated by;not,

If there are multiple C statements that make up the loop body,enclose them in

brackets.

```
for (x=100;x!=65;x-=5)
{
z=sqrt(x);
printf("The square root of %d is %f\n",x,z);
}
```


تعديل من خلال WPS Office

The output

-The square root of 100 is 10.000000

The square root of 95 is 9.746794

The square root of 90 is 9.486833

The square root of 85 is 9.219544

The square root of 80 is 8.944272

The square root of 75 is 8.660254

The square root of 70 is 8.366600

تعديل من خلال WPS Office

Control expressions can be any valid expression.

Don't necessarily have to

perform initialization,testing,and incrementation.

-Any of the control expressions can be omitted (but always need the two

semicolons for syntax sake).

```
product=1 ;

for(i=1;i<=6;)

product*=i++;
```


تعديل من خلال WPS Office

General Comments about for Loop Continued

Some general comments regarding the use of the for statement:

-Since test performed at beginning of loop,body may never get executed

```
x=10;

for(y=10;y!=x;++y)

printf("%d",y);
```

-Can string together multiple expressions in the for statement by separating

them by commas.

```
for(x=1,y=5;x+y<100;++x)

z=x%y;
```


تعديل من خلال WPS Office

Basic Increment (Counting Up)

This is the most common use of a for loop. We initialize a counter and increase it until a condition is met.

```
#include <stdio.h>

int main() {

    int i;

    // (Start at 1; stay as long as i <= 5; add 1 each time)

    for (i = 1; i <= 5; i++) {

        printf("Current value: %d\n", i);

    }

    return 0;

}
```


تعديل من خلال WPS Office

Initialization: i = 1 sets the starting point.

Condition: i <= 5 is checked before every turn.

Increment: i++ happens after the printf runs

Summation Logic (Accumulating Data)

We use the loop to perform a repeated calculation.

In this case, adding numbers from 1 to 10.

```
#include <stdio.h>
int main() {
    int sum = 0; // Starts at zero
     for (int j = 1; j <= 10; j++) {
        sum = sum + j; // Adds the current j to the total
    }
    printf("The final sum is: %d\n", sum);
    return 0;
}
```


تعديل من خلال WPS Office

Variable Scope: Note that int j is declared inside the loop; it only exists while the loop is running.

Logic: The loop runs 10 times, updating the sum variable in each iteration.

Decrementing (Counting Down)

Loops don't always have to go up.

You can use them to count backwards by using the -- operator.

```
#include <stdio.h>

int main() {

    // (Start at 10; stay as long as k > 0; subtract 1 each time)

    for (int k = 10; k > 0; k--) {

        printf("%d seconds remaining...\n", k);

    }

    printf("Liftoff! 🚀\n");

    return 0;
```


تعديل من خلال WPS Office

}

Condition Change: Since we are going down, the condition uses "Greater Than" (>) instead of "Less Than."

Step: k-- reduces the value by 1 every cycle.

Code: Multiplication Table of 7

```
#include <stdio.h>

int main() {

    int i;

    int number = 7; // The table we want to print

    printf("Multiplication Table of %d:\n", number);

    printf("-------------------------\n");

    // Loop from 1 to 10

    for (i = 1; i <= 10; i++) {

        // Calculation: 7 multiplied by the current loop index (i)

        printf("%d x %d = %d\n", number, i,
```


تعديل من خلال WPS Office

```
number * i);
    }
    return 0;
}
```

Homework (HW)

Homework 1: The "Even Numbers" Challenge

Goal: Write a C program that prints all even numbers between 1 and 20.

* Hint: Start your loop at i = 2. Instead of i++, use i += 2 to jump by two.

* Expected Output: 2, 4, 6, 8, ... 20

تعديل من خلال WPS Office

Homework 2: The "Sum of Squares" Calculator

Goal: Create a program that calculates the sum of the squares of the first 5 natural numbers (1^2 + 2^2 + 3^2 + 4^2 + 5^2).

* Hint: Initialize a totalSum = 0 outside the loop.

Inside the loop, calculate i * i and add it to totalSum.

* Expected Result: 1 + 4 + 9 + 16 + 25 = 55.

تعديل من خلال WPS Office

Homework 3: The User-Defined Table (Dynamic)

Goal: Modify the "Table of 7" code you just saw.

Ask the user to input a number (using scanf), and then print the multiplication table for that specific number up to 12.

* Hint: Use int userNum; and scanf("%d", &userNum); before the loop starts.

* Expected Output: If the user enters 5, it prints 5 x 1 = 5 up to 5 x 12 = 60.

تعديل من خلال WPS Office