`**break**` Keyword

The `break` keyword is used to immediately terminate the execution of a loop or switch statement. It is primarily used in two contexts:

1. **Loops**: To exit the loop prematurely when a certain condition is met.

2. **Switch** Statements: To exit the switch block once a specific case has been executed, preventing the execution of subsequent cases.

### **break Keyword Behavior in Different Loops**

The break keyword behaves consistently across different types of loops and switch statements by immediately terminating the loop or switch block. Here is how it works in various contexts:

1. **while Loop**: When break is encountered, the loop terminates immediately, and the control exits the loop block and proceeds with the next statement after the loop.
2. **for Loop**: When break is encountered, the loop terminates immediately, and the control exits the loop block and proceeds with the next statement after the loop.
3. **do-while Loop**: When break is encountered, the loop terminates immediately, and the control exits the loop block and proceeds with the next statement after the loop.
4. **switch Statement**: When break is encountered within a switch case, the control exits the switch block, preventing the execution of subsequent cases.

In summary, break causes an immediate exit from the loop or switch statement, transferring control to the next statement following the loop or switch block.

Objective:

Enter a number and print is prime or not.

What is prime number?

All those numbers which are grather than 1 and only divisible by 1 and itself and should be grater than 1.

#include<stdio.h>

int main(){

int x, y, z;

// Prompt user to enter a number

printf("Enter a number: ");

scanf("%d", &x);

if(x > 1) {

z = 0; // Assume number is prime

y = 2; // Start checking from 2

// Check divisibility from 2 to x-1

while(y < x) {

printf("Dividing %d by %d\n", x, y);

if(x % y == 0) {

z = 1; // Number is not prime

}

y++;

}

} else {

z = 1; // Numbers less than or equal to 1 are not prime

}

// Output result

if(z == 0) {

printf("Number is prime");

} else {

printf("Number is not prime");

}

return 0;

}

### **Explanation:**

1. **Prompt User for Input**: The user is prompted to enter a number.
2. **Initial Check**: If the number is greater than 1, we assume it might be prime and proceed with the checks. If it is less than or equal to 1, it is not prime.
3. **Divisibility Check**:
   * Start checking divisibility from 2.
   * For each divisor, print the operation being performed.
   * If the number is divisible by any number (i.e., remainder is 0), set z to 1 (indicating it's not prime) and break out of the loop.
4. **Prime Check**:
   * If z remains 0, the number is prime.
   * Otherwise, the number is not prime.
5. **Output**: Print whether the number is prime or not.

### **Logical Issue in Original Code:**

In the original code, the loop continues even after finding a divisor, which is unnecessary. Once we determine a number is not prime (find a divisor), we should stop further checks using the break keyword.

#include<stdio.h>

int main(){

int x, y, z;

// Prompt user to enter a number

printf("Enter a number: ");

scanf("%d", &x);

if(x > 1) {

z = 0; // Assume number is prime

y = 2; // Start checking from 2

// Check divisibility from 2 to x-1

while(y < x) {

printf("Dividing %d by %d\n", x, y);

if(x % y == 0) {

z = 1; // Number is not prime

break; // Exit loop as we found a divisor // CHANGE HERE

}

y++;

}

} else {

z = 1; // Numbers less than or equal to 1 are not prime

}

// Output result

if(z == 0) {

printf("Number is prime");

} else {

printf("Number is not prime");

}

return 0;

}

In prime number concept there is 1 more point

If you take any number to find prime number

We start compare it from 2 to x-1

But after x/2 there is no possibility of division so we should only compare upto x/2

Find prime number between two numbers;

#include<stdio.h>

int main()

{

int x,y,z,ep,sp,j;

printf("Enter starting points: ");

scanf("%d", &sp);

printf("Enter ending points: ");

scanf("%d", &ep);

j = sp;

while(j<=ep){

x = j;

if(x>1){

y = 2;

z = 0;

while(y<=x-1){

if(x%y == 0){

z = 1;

break;

}

y++;

}

if(z == 0){

printf("%d\n", x);

}

}

j++;

}

return 0;

}

Q. Break terminate which loop?

It will only break the current loop in which we are iterating.

It will not break multiple loop.