#include<stdio.h>  
  
int main() {  
 int x;  
 x = 50;  
 printf("%d", x);  
 return 0;  
}

## **Explanation**

### **1. Variable Declaration and Memory Allocation:**

When you write int x;, it means that in RAM, 4 bytes of memory are allocated for the variable x. 4 bytes = 4 \* 8 bits = 32 bits of memory.

### **2. Assigning a Value:**

When you write x = 50;, the value 50 is converted into its binary equivalent and stored in the allocated memory space for x.

### **3. Printing the Value:**

The statement printf("%d", x);:  
%d is a format specifier used in printf to print an integer value.  
printf will convert the binary value stored in x back to a decimal value and print it.

### **4. Garbage Value:**

If you declare a variable and do not initialize it, it will hold a random value known as a "garbage value".

Example:

#include<stdio.h>  
  
int main() {  
 int x;  
 printf("%d", x);  
 return 0;  
}

The output will be a random value, e.g., 22412.

## **Looping Constructs**

To avoid repetitive code, use looping constructs. Available keywords are while, for, and do-while.

### **Example without Loop:**

int main() {  
 int x;  
 x = 1;  
 printf("%d", x);  
 x++;  
 printf("%d", x);  
 x++;  
 printf("%d", x);  
 x++;  
 printf("%d", x);  
 x++;  
 printf("%d", x);  
 return 0;  
}

### **Example with while Loop:**

int main() {  
 int x; // Memory allocation for x in RAM  
 x = 1; // Value assigned to x is 1  
 while (x <= 5) {  
 printf("%d", x); // Prints value of x: 1, 2, 3, 4, 5  
 x++;  
 }  
 return 0;  
}

### **Syntax of while loop:**

while (condition) {  
 // code block to be executed  
}

### **Working:**

If the condition evaluates to true, the control enters the loop. If false, the control bypasses the loop. After each loop execution, the condition is re-evaluated.

### **Output of the Loop Example:**

12345