C PROGRAMMING

DANIEL HOUSARD

ONLINE COMPUTER TRAINING CENTER

C Data Types

In C programming, data types are declarations for variables. This determines the type and size of data associated with variables.
For example, int myVar;
Here, myVar is a variable of int (integer) type. The size of int is 4 bytes.

Basic types

Here's a table containing commonly used types in C programming for quick access.

Type	Size (bytes)	Format Specifier
int	at least 2, usually 4	%d, %i
char	1	%c
float	4	%f
double	8	%lf
short int	2 usually	%hd
unsigned int	at least 2, usually 4	%u
long int	at least 4, usually 8	%ld, %li
long long int	at least 8	%lld, %lli
unsigned long int	at least 4	%lu
unsigned long long int	at least 8	%llu
signed char	1	%c
unsigned char	1	%c
long double	at least 10, usually 12 or 16	%Lf

int

Integers are whole numbers that can have both zero, positive and negative values but no decimal values. For example, 0, -5, 10
We can use int for declaring an integer variable.

int id;
Here, id is a variable of type integer.
You can declare multiple variables at once in C programming. For example, int id, age;
The size of int is usually 4 bytes (32 bits). And, it can take 232 distinct states from -2147483648 to 2147483647.

float and double

float and double are used to hold real numbers.
float salary;
double price;
In C, floating-point numbers can also be represented in exponential. For example,
float normalizationFactor = 22.442e2;
What's the difference between float and double?
The size of float (single precision float data type) is 4 bytes. And the size of double (double precision float data type) is 8 bytes.

char

Keyword char is used for declaring character type variables. For example, char test = 'h';
The size of the character variable is 1 byte.

void
void is an incomplete type. It means "nothing" or "no type". You can think of void as **absent**. For example, if a function is not returning anything, its return type should be void.
Note that, you cannot create variables of void type.

short and long
If you need to use a large number, you can use a type specifier long. Here's how:
long a;
long long b;
long double c;

Here variables a and b can store integer values. And, c can store a floating-point number.
If you are sure, only a small integer ([−32,767, +32,767] range) will be used, you can use short.
short d;

You can always check the size of a variable using the sizeof() operator.

```
#include <stdio.h>
int main() {
  short a;
  long b;
  long long c;
  long double d;

  printf("size of short = %d bytes\n", sizeof(a));
  printf("size of long = %d bytes\n", sizeof(b));
  printf("size of long long = %d bytes\n", sizeof(c));
  printf("size of long double= %d bytes\n", sizeof(d));
  return 0;
}
```

signed and unsigned
In C, signed and unsigned are type modifiers. You can alter the data storage of a data type by using them. For example, unsigned int x; int y;

Here, the variable x can hold only zero and positive values because we have used the unsigned modifier.

Considering the size of int is 4 bytes, variable y can hold values from -231 to 231-1, whereas variable x can hold values from 0 to 232-1.

Other data types defined in C programming are:
bool Type
Enumerated type
Complex types

Derived Data Types
Data types that are derived from fundamental data types are derived types. For example: arrays, pointers, function types, structures, etc.

C Output

In C programming, printf() is one of the main output function. The function sends formatted output to the screen. For example,

Example 1: C Output

```
#include <stdio.h>
int main()
{
   // Displays the string inside quotations
   printf("C Programming");
   return 0;
}
```

Output
C Programming

How does this program work?
All valid C programs must contain the main() function. The code execution begins from the start of the main() function.
The printf() is a library function to send formatted output to the screen. The function prints the string inside quotations.
To use printf() in our program, we need to include stdio.h header file using the #include <stdio.h> statement.
The return 0; statement inside the main() function is the "Exit status" of the program. It's optional.

Example 2: Integer Output

```
#include <stdio.h>
int main()
{
   int testInteger = 5;
   printf("Number = %d", testInteger);
   return 0;
}
```

Output
Number = 5

We use %d format specifier to print int types. Here, the %d inside the quotations will be replaced by the value of testInteger.

Example 3: float and double Output

```
#include <stdio.h>
int main()
{
   float number1 = 13.5;
   double number2 = 12.4;
   printf("number1 = %f\n", number1);
   printf("number2 = %lf", number2);
   return 0;
}
```

Output
number1 = 13.500000
number2 = 12.400000

To print float, we use %f format specifier. Similarly, we use %lf to print double values.

Example 4: Print Characters

```
#include <stdio.h>
int main()
{
   char chr = 'a';
   printf("character = %c", chr);
   return 0;
}
```

Output
character = a

To print char, we use %c format specifier.

C Input

In C programming, scanf() is one of the commonly used function to take input from the user. The scanf() function reads formatted input from the standard input such as keyboards.

Example 5: Integer Input/Output

```
#include <stdio.h>
int main()
{
   int testInteger;
   printf("Enter an integer: ");
   scanf("%d", &testInteger);
   printf("Number = %d",testInteger);
   return 0;
}
```

Output
Enter an integer: 4
Number = 4

Here, we have used %d format specifier inside the scanf() function to take int input from the user. When the user enters an integer, it is stored in the testInteger variable.

Notice, that we have used &testInteger inside scanf(). It is because &testInteger gets the address of testInteger, and the value entered by the user is stored in that address.

Example 6: Float and Double Input/Output

```
#include <stdio.h>
int main()
{
   float num1;
   double num2;

   printf("Enter a number: ");
   scanf("%f", &num1);
   printf("Enter another number: ");
   scanf("%lf", &num2);

   printf("num1 = %f\n", num1);
   printf("num2 = %lf", num2);

   return 0;
}
```

Output
Enter a number: 12.523
Enter another number: 10.2
num1 = 12.523000
num2 = 10.200000

We use %f and %lf format specifier for float and double respectively.

Example 7: C Character I/O

```
#include <stdio.h>
int main()
{
   char chr;
   printf("Enter a character: ");
   scanf("%c",&chr);
   printf("You entered %c.", chr);
   return 0;
}
```

Output
Enter a character: g
You entered g

When a character is entered by the user in the above program, the character itself is not stored. Instead, an integer value (ASCII value) is stored.

And when we display that value using %c text format, the entered character is displayed. If we use %d to display the character, it's ASCII value is printed.

Example 8: ASCII Value

```
#include <stdio.h>
int main()
{
   char chr;
   printf("Enter a character: ");
   scanf("%c", &chr);

   // When %c is used, a character is displayed
   printf("You entered %c.\n",chr);

   // When %d is used, ASCII value is displayed
   printf("ASCII value is %d.", chr);
   return 0;
}
```

Output
Enter a character: g
You entered g.
ASCII value is 103.

I/O Multiple Values

Here's how you can take multiple inputs from the user and display them.

```
#include <stdio.h>
int main()
{
   int a;
   float b;

   printf("Enter integer and then a float: ");

   // Taking multiple inputs
   scanf("%d%f", &a, &b);

   printf("You entered %d and %f", a, b);
   return 0;
}
```

Output
Enter integer and then a float: -3
3.4
You entered -3 and 3.400000

Format Specifiers for I/O

As you can see from the above examples, we use

- %d for int
- %f for float
- %lf for double
- %c for char

Here's a list of commonly used C data types and their format specifiers.

Data Type	Format Specifier
int	%d
char	%c
float	%f
double	%lf
short int	%hd
unsigned int	%u
long int	%li
long long int	%lli
unsigned long int	%lu
unsigned long long int	%llu
signed char	%c
unsigned char	%c
long double	%Lf

1 C "Hello, World!" Program

In this example, you will learn to print "Hello, World!" on the screen in C programming.

```
Program to Display "Hello, World!"
#include <stdio.h>
int main() {
  // printf() displays the string inside quotation
  printf("Hello, World!");
  return 0;
}
```

Output:
Hello, World!

How "Hello, World!" program works?
The **#include** is a preprocessor command that tells the compiler to include the contents of **stdio.h** (standard input and output) file in the program.

The **stdio.h** file contains functions such as **scanf()** and **printf()** to take input and display output respectively.

If you use the **printf()** function without writing **#include <stdio.h>,** the program will not compile.

The execution of a C program starts from the **main()** function.
printf() is a library function to send formatted output to the screen. In this program, **printf()** displays **Hello, World!** text on the screen.

The **return 0;** statement is the "Exit status" of the program. In simple terms, the program ends with this statement.

2 How to Print an Integer (Entered by the User)

In this example, the integer entered by the user is stored in a variable and printed on the screen.

```
#include <stdio.h>
int main() {
   int number;

   printf("Enter an integer: ");

   // reads and stores input
   scanf("%d", &number);

   // displays output
   printf("You entered: %d", number);

   return 0;
}
```

Output:
Enter an integer: 25
You entered: 25

In this program, an integer variable number is declared.
int number;

Then, the user is asked to enter an integer number. This number is stored in the number variable.
printf("Enter an integer: ");
scanf("%d", &number);

Finally, the value stored in number is displayed on the screen using printf().
printf("You entered: %d", number);

3 How to Add Two Integers

In this example, the user is asked to enter two integers. Then, the sum of these two integers is calculated and displayed on the screen.

```
#include <stdio.h>
int main() {

   int number1, number2, sum;

   printf("Enter two integers: ");
   scanf("%d %d", &number1, &number2);

   // calculating sum
   sum = number1 + number2;

   printf("%d + %d = %d", number1, number2, sum);
   return 0;
}
```

Output:
Enter two integers:
12
11
12 + 11 = 23

In this program, the user is asked to enter two integers. These two integers are stored in variables number1 and number2 respectively.
printf("Enter two integers: ");
scanf("%d %d", &number1, &number2);

Then, these two numbers are added using the + operator, and the result is stored in the sum variable.
sum = number1 + number2;

4 How to Multiply Two Floating-Point Numbers

In this example, the product of two floating-point numbers entered by the user is calculated and printed on the screen.

```
#include <stdio.h>
int main() {
   double a, b, product;
   printf("Enter two numbers: ");
   scanf("%lf %lf", &a, &b);

   // Calculating product
   product = a * b;

   // %.2lf displays number up to 2 decimal point
   printf("Product = %.2lf", product);

   return 0;
}
```

Output
Enter two numbers: 2.4
1.12
Product = 2.69

In this program, the user is asked to enter two numbers which are stored in variables a and b respectively.
printf("Enter two numbers: ");
scanf("%lf %lf", &a, &b);

Then, the product of a and b is evaluated and the result is stored in product.
product = a * b;

Finally, product is displayed on the screen using printf().
printf("Product = %.2lf", product);

Notice that, the result is rounded off to the second decimal place using %.2lf conversion character.

5 How to Find ASCII Value of a Character

In this example, you will learn how to find the ASCII value of a character.

In C programming, a character variable holds ASCII value (an integer number between 0 and 127) rather than that character itself. This integer value is the ASCII code of the character. For example, the ASCII value of 'A' is 65.

What this means is that, if you assign 'A' to a character variable, 65 is stored in the variable rather than 'A' itself.

```
#include <stdio.h>
int main() {
   char c;
   printf("Enter a character: ");
   scanf("%c", &c);

   // %d displays the integer value of a character
   // %c displays the actual character
   printf("ASCII value of %c = %d", c, c);

   return 0;
}
```

Output

Enter a character: G
ASCII value of G = 71

In this program, the user is asked to enter a character. The character is stored in variable c.
When %d format string is used, **71** (the ASCII value of G) is displayed.
When %c format string is used, 'G' itself is displayed.

6 How to Compute Quotient and Remainder

In this example, you will learn to find the quotient and remainder when an integer is divided by another integer.

```
#include <stdio.h>
int main() {
   int dividend, divisor, quotient, remainder;
   printf("Enter dividend: ");
   scanf("%d", &dividend);
   printf("Enter divisor: ");
   scanf("%d", &divisor);

   // Computes quotient
   quotient = dividend / divisor;

   // Computes remainder
   remainder = dividend % divisor;

   printf("Quotient = %d\n", quotient);
   printf("Remainder = %d", remainder);
   return 0;
}
```

Output
Enter dividend: 25
Enter divisor: 4
Quotient = 6
Remainder = 1

In this program, the user is asked to enter two integers (dividend and divisor). They are stored in variables dividend and divisor respectively.
printf("Enter dividend: ");
scanf("%d", &dividend);
printf("Enter divisor: ");
scanf("%d", &divisor);

Then the quotient is evaluated using / (the division operator), and stored in quotient.
quotient = dividend / divisor;

Similarly, the remainder is evaluated using % (the modulo operator) and stored in remainder.
remainder = dividend % divisor;

Finally, the quotient and remainder are displayed using printf().
printf("Quotient = %d\n", quotient);
printf("Remainder = %d", remainder);

7 How to Find the Size of int, float, double and char

In this example, you will learn to evaluate the size of each variable using sizeof operator.

The sizeof(variable)operator computes the size of a variable. And, to print the result returned by sizeof, we use either %lu or %zu format specifier.

```
#include<stdio.h>
int main() {
   int intType;
   float floatType;
   double doubleType;
   char charType;

   // sizeof evaluates the size of a variable
   printf("Size of int: %zu bytes\n", sizeof(intType));
   printf("Size of float: %zu bytes\n", sizeof(floatType));
   printf("Size of double: %zu bytes\n", sizeof(doubleType));
   printf("Size of char: %zu byte\n", sizeof(charType));

   return 0;
}
```

Output

Size of int: 4 bytes
Size of float: 4 bytes
Size of double: 8 bytes
Size of char: 1 byte

In this program, 4 variables intType, floatType, doubleType and charType are declared. Then, the size of each variable is computed using the sizeof operator.

8 How to Demonstrate the Working of Keyword long

In this example, you will learn to demonstrate the working of the long keyword.

```
#include <stdio.h>
int main() {
   int a;
   long b;   // equivalent to long int b;
   long long c;  // equivalent to long long int c;
   double e;
   long double f;

   printf("Size of int = %zu bytes \n", sizeof(a));
   printf("Size of long int = %zu bytes\n", sizeof(b));
   printf("Size of long long int = %zu bytes\n", sizeof(c));
   printf("Size of double = %zu bytes\n", sizeof(e));
   printf("Size of long double = %zu bytes\n", sizeof(f));

   return 0;
}
```

Output

Size of int = 4 bytes
Size of long int = 8 bytes
Size of long long int = 8 bytes
Size of double = 8 bytes
Size of long double = 16 bytes

In this program, the sizeof operator is used to find the size of int, long, long long, double and long double variables.

As you can see, the size of long int and long double variables are larger than int and double variables, respectively.

By the way, the sizeof operator returns size_t (unsigned integral type).

The size_t data type is used to represent the size of an object. The format specifier used for size_t is %zu.

Note: The long keyword cannot be used with float and char types.

9 How to Swap Two Numbers

In this example, you will learn to swap two numbers in C programming using two different techniques.

Swap Numbers Using Temporary Variable

```
#include<stdio.h>
int main() {
    double first, second, temp;
    printf("Enter first number: ");
    scanf("%lf", &first);
    printf("Enter second number: ");
    scanf("%lf", &second);

    // Value of first is assigned to temp
    temp = first;

    // Value of second is assigned to first
    first = second;

    // Value of temp (initial value of first) is assigned to second
    second = temp;

    // %.2lf displays number up to 2 decimal points
    printf("\nAfter swapping, firstNumber = %.2lf\n", first);
    printf("After swapping, secondNumber = %.2lf", second);
    return 0;
}
```

Output
Enter first number: 1.20
Enter second number: 2.45

After swapping, firstNumber = 2.45
After swapping, secondNumber = 1.20

In the above program, the temp variable is assigned the value of the first variable.
Then, the value of the first variable is assigned to the second variable.

Finally, the temp (which holds the initial value of first) is assigned to second. This completes the swapping process.

Swap Numbers Without Using Temporary Variable

```
#include <stdio.h>
int main() {
   double a, b;
   printf("Enter a: ");
   scanf("%lf", &a);
   printf("Enter b: ");
   scanf("%lf", &b);

   // Swapping

   // a = (initial_a - initial_b)
   a = a - b;

   // b = (initial_a - initial_b) + initial_b = initial_a
   b = a + b;

   // a = initial_a - (initial_a - initial_b) = initial_b
   a = b - a;

   // %.2lf displays number up to 2 decimal points
   printf("After swapping, a = %.2lf\n", a);
   printf("After swapping, b = %.2lf", b);
   return 0;
}
```

Output
Enter a: 10.25
Enter b: -12.5
After swapping, a = -12.50
After swapping, b = 10.25

10 How to Check Whether a Number is Even or Odd

In this example, you will learn to check whether a number entered by the user is even or odd.
An even number is an integer that is exactly divisible by 2. For example: 0, 8, -24
An odd number is an integer that is not exactly divisible by 2. For example: 1, 7, -11, 15

```
#include <stdio.h>
int main() {
   int num;
   printf("Enter an integer: ");
   scanf("%d", &num);

   // true if num is perfectly divisible by 2
   if(num % 2 == 0)
      printf("%d is even.", num);
   else
      printf("%d is odd.", num);

   return 0;
}
```

Output
Enter an integer: -7
-7 is odd.

In the program, the integer entered by the user is stored in the variable num.

Then, whether num is perfectly divisible by 2 or not is checked using the modulus % operator.

If the number is perfectly divisible by 2, test expression number%2 == 0 evaluates to 1 (true). This means the number is even.

However, if the test expression evaluates to 0 (false), the number is odd.

11 How to Check Whether a Character is a Vowel or Consonant

In this example, you will learn to check whether an alphabet entered by the user is a vowel or a consonant.

The five letters A, E, I, O and U are called vowels. All other alphabets except these 5 vowels are called consonants.
This program assumes that the user will always enter an alphabet character.

```
#include <stdio.h>
int main() {
   char c;
   int lowercase_vowel, uppercase_vowel;
   printf("Enter an alphabet: ");
   scanf("%c", &c);

   // evaluates to 1 if variable c is a lowercase vowel
   lowercase_vowel = (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u');

   // evaluates to 1 if variable c is a uppercase vowel
   uppercase_vowel = (c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U');

   // evaluates to 1 (true) if c is a vowel
   if (lowercase_vowel || uppercase_vowel)
      printf("%c is a vowel.", c);
   else
      printf("%c is a consonant.", c);
   return 0;
}
```

Output
Enter an alphabet: G
G is a consonant.

The character entered by the user is stored in variable c.

The lowercase_vowel variable evaluates to 1 (true) if c is a lowercase vowel and 0 (false) for any other characters.

Similarly, the uppercase_vowel variable evaluates to 1 (true) if c is an uppercase vowel and 0 (false) for any other character.

If either lowercase_vowel or uppercase_vowel variable is 1 (true), the entered character is a vowel. However, if both lowercase_vowel and uppercase_vowel variables are 0, the entered character is a consonant.

12 How to Find the Largest Number Among Three Numbers

In this example, you will learn to find the largest number among the three numbers entered by the user.

Example 1: Using if Statement

```
#include <stdio.h>
int main() {
   double n1, n2, n3;
   printf("Enter three different numbers: ");
   scanf("%lf %lf %lf", &n1, &n2, &n3);

   // if n1 is greater than both n2 and n3, n1 is the largest
   if (n1 >= n2 && n1 >= n3)
      printf("%.2f is the largest number.", n1);

   // if n2 is greater than both n1 and n3, n2 is the largest
   if (n2 >= n1 && n2 >= n3)
      printf("%.2f is the largest number.", n2);

   // if n3 is greater than both n1 and n2, n3 is the largest
   if (n3 >= n1 && n3 >= n2)
      printf("%.2f is the largest number.", n3);

   return 0;
}
```

Example 2: Using if...else Ladder

```
#include <stdio.h>
int main() {
   double n1, n2, n3;
   printf("Enter three numbers: ");
   scanf("%lf %lf %lf", &n1, &n2, &n3);

   // if n1 is greater than both n2 and n3, n1 is the largest
   if (n1 >= n2 && n1 >= n3)
      printf("%.2lf is the largest number.", n1);

   // if n2 is greater than both n1 and n3, n2 is the largest
   else if (n2 >= n1 && n2 >= n3)
      printf("%.2lf is the largest number.", n2);

   // if both above conditions are false, n3 is the largest
   else
      printf("%.2lf is the largest number.", n3);

   return 0;
}
```

Example 3: Using Nested if...else

```
#include <stdio.h>
int main() {
   double n1, n2, n3;
   printf("Enter three numbers: ");
   scanf("%lf %lf %lf", &n1, &n2, &n3);

   if (n1 >= n2) {
      if (n1 >= n3)
         printf("%.2lf is the largest number.", n1);
      else
         printf("%.2lf is the largest number.", n3);
   } else {
      if (n2 >= n3)
         printf("%.2lf is the largest number.", n2);
      else
         printf("%.2lf is the largest number.", n3);
   }

   return 0;
}
```

The output of all these programs above will be the same.
Enter three numbers: -4.5
3.9
5.6
5.60 is the largest number.

13 How to Check Leap Year

In this example, you will learn to check whether the year entered by the user is a leap year or not.

A leap year is exactly divisible by 4 except for century years (years ending with 00). The century year is a leap year only if it is perfectly divisible by 400.

For example,
1999 is not a leap year
2000 is a leap year
2004 is a leap year

```
#include <stdio.h>
int main() {
  int year;
  printf("Enter a year: ");
  scanf("%d", &year);

  // leap year if perfectly divisible by 400
  if (year % 400 == 0) {
    printf("%d is a leap year.", year);
  }
  // not a leap year if divisible by 100
  // but not divisible by 400
  else if (year % 100 == 0) {
    printf("%d is not a leap year.", year);
  }
  // leap year if not divisible by 100
  // but divisible by 4
  else if (year % 4 == 0) {
    printf("%d is a leap year.", year);
  }
  // all other years are not leap years
  else {
    printf("%d is not a leap year.", year);
  }
  return 0;
}
```

Output 1
Enter a year: 1900
1900 is not a leap year.

Output 2
Enter a year: 2012
2012 is a leap year.

14 How to Check Whether a Number is Positive or Negative

In this example, you will learn to check whether a number (entered by the user) is negative or positive.

This program takes a number from the user and checks whether that number is either positive or negative or zero.

```
#include <stdio.h>
int main() {
   double num;
   printf("Enter a number: ");
   scanf("%lf", &num);
   if (num <= 0.0) {
      if (num == 0.0)
         printf("You entered 0.");
      else
         printf("You entered a negative number.");
   } else
      printf("You entered a positive number.");
   return 0;
}
```

You can also solve this problem using nested if else statement.

```
#include <stdio.h>
int main() {
   double num;
   printf("Enter a number: ");
   scanf("%lf", &num);

   if (num < 0.0)
      printf("You entered a negative number.");
   else if (num > 0.0)
      printf("You entered a positive number.");
   else
      printf("You entered 0.");

   return 0;
}
```

Output 1
Enter a number: 12.3
You entered a positive number.

Output 2
Enter a number: 0
You entered 0.

15 How to Check Whether a Character is an Alphabet or not

In this example, you will learn to check whether a character entered by the user is an alphabet or not.

In C programming, a character variable holds an ASCII value (an integer number between 0 and 127) rather than that character itself.

The ASCII value of the lowercase alphabet is from 97 to 122. And, the ASCII value of the uppercase alphabet is from 65 to 90.

If the ASCII value of the character entered by the user lies in the range of 97 to 122 or from 65 to 90, that number is an alphabet.

```
#include <stdio.h>
int main() {
   char c;
   printf("Enter a character: ");
   scanf("%c", &c);

   if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z'))
      printf("%c is an alphabet.", c);
   else
      printf("%c is not an alphabet.", c);

   return 0;
}
```

Output

Enter a character: *

***** is not an alphabet**

16 How to Calculate the Sum of Natural Numbers

In this example, you will learn to calculate the sum of natural numbers entered by the user.

The positive numbers **1, 2, 3...** are known as natural numbers. The sum of natural numbers up to 10 is:
sum = 1 + 2 + 3 + ... + 10

```
#include <stdio.h>
int main() {
   int n, i, sum = 0;

   printf("Enter a positive integer: ");
   scanf("%d", &n);

   for (i = 1; i <= n; ++i) {
      sum += i;
   }

   printf("Sum = %d", sum);
   return 0;
}
```

The above program takes input from the user and stores it in the variable n. Then, for loop is used to calculate the sum up to n.

Sum of Natural Numbers Using while Loop

```
#include <stdio.h>
int main() {
   int n, i, sum = 0;
   printf("Enter a positive integer: ");
   scanf("%d", &n);
   i = 1;

   while (i <= n) {
      sum += i;
      ++i;
   }

   printf("Sum = %d", sum);
   return 0;
}
```

Output
Enter a positive integer: 100
Sum = 5050

In both programs, the loop is iterated n number of times. And, in each iteration, the value of i is added to sum and i is incremented by 1.

Though both programs are technically correct, it is better to use for loop in this case. It's because the number of iterations is known.

The above programs don't work properly if the user enters a negative integer. Here is a little modification to the above program where we keep taking input from the user until a positive integer is entered.

Read Input Until a Positive Integer is Entered

```
#include <stdio.h>
int main() {
   int n, i, sum = 0;

   do {
      printf("Enter a positive integer: ");
      scanf("%d", &n);
   } while (n <= 0);

   for (i = 1; i <= n; ++i) {
      sum += i;
   }

   printf("Sum = %d", sum);
   return 0;
}
```

17 How to Find Factorial of a Number

In this example, you will learn to calculate the factorial of a number entered by the user.

The factorial of a positive number n is given by:
factorial of n (n!) = 1 * 2 * 3 * 4....n
The factorial of a negative number doesn't exist. And, the factorial of 0 is 1.

Factorial of a Number

```
#include <stdio.h>
int main() {
   int n, i;
   unsigned long long fact = 1;
   printf("Enter an integer: ");
   scanf("%d", &n);

   // shows error if the user enters a negative integer
   if (n < 0)
      printf("Error! Factorial of a negative number doesn't exist.");
   else {
      for (i = 1; i <= n; ++i) {
         fact *= i;
      }
      printf("Factorial of %d = %llu", n, fact);
   }

   return 0;
}
```

Output

```
Enter an integer: 10
Factorial of 10 = 3628800
```

This program takes a positive integer from the user and computes the factorial using for loop. Since the factorial of a number may be very large, the type of factorial variable is declared as unsigned long long.

If the user enters a negative number, the program displays a custom error message.

18 How to Generate Multiplication Table

In this example, you will learn to generate the multiplication table of a number entered by the user.

The program below takes an integer input from the user and generates the multiplication tables up to 10.

```
#include <stdio.h>
int main() {
   int n, i;
   printf("Enter an integer: ");
   scanf("%d", &n);
   for (i = 1; i <= 10; ++i) {
      printf("%d * %d = %d \n", n, i, n * i);
   }
   return 0;
}
```

Output

Enter an integer: 9
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
9 * 10 = 90

19 How to Display Fibonacci Sequence

In this example, you will learn to display the Fibonacci sequence of first n numbers (entered by the user).

The Fibonacci sequence is a sequence where the next term is the sum of the previous two terms. The first two terms of the Fibonacci sequence are 0 followed by 1.
The Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21

```
#include <stdio.h>
int main() {
   int i, n, t1 = 0, t2 = 1;
   int nextTerm = t1 + t2;
   printf("Enter the number of terms: ");
   scanf("%d", &n);
   printf("Fibonacci Series: %d, %d, ", t1, t2);

   for (i = 1; i <= n; ++i) {
      printf("%d, ", nextTerm);
      t1 = t2;
      t2 = nextTerm;
      nextTerm = t1 + t2;
   }

   return 0;
}
```

Output
Enter the number of terms: 10
Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,

20 How to Find GCD of two Numbers

Examples on different ways to calculate GCD of two integers (for both positive and negative integers) using loops and decision making statements.

The HCF (Highest Common Factor) or GCD (Greater Common Divisor) of two integers is the largest integer that can exactly divide both numbers (without a remainder).

There are many ways to find the greatest common divisor in C programming.

Example #1: GCD Using for loop and if Statement

```
#include <stdio.h>
int main()
{
   int n1, n2, i, gcd;

   printf("Enter two integers: ");
   scanf("%d %d", &n1, &n2);

   for(i=1; i <= n1 && i <= n2; ++i)
   {
      // Checks if i is factor of both integers
      if(n1%i==0 && n2%i==0)
         gcd = i;
   }

   printf("G.C.D of %d and %d is %d", n1, n2, gcd);

   return 0;
}
```

In this program, two integers entered by the user are stored in variable n1 and n2.Then, for loop is iterated until i is less than n1 and n2.

In each iteration, if both n1 and n2 are exactly divisible by i, the value of i is assigned to gcd. When the for loop is completed, the greatest common divisor of two numbers is stored in variable gcd.

Example #2: GCD Using while loop and if...else Statement

```
#include <stdio.h>
int main()
{
   int n1, n2;

   printf("Enter two positive integers: ");
   scanf("%d %d",&n1,&n2);

   while(n1!=n2)
   {
      if(n1 > n2)
         n1 -= n2;
      else
```

```
        n2 -= n1;
    }
    printf("GCD = %d",n1);

    return 0;
}
```

Output
Enter two positive integers: 81
153
GCD = 9

This is a better way to find the GCD. In this method, smaller integer is subtracted from the larger integer, and the result is assigned to the variable holding larger integer. This process is continued until n1 and n2 are equal.

The above two programs works as intended only if the user enters positive integers. Here's a little modification of the second example to find the GCD for both positive and negative integers.

Example #3: GCD for both positive and negative numbers

```
#include <stdio.h>
int main()
{
    int n1, n2;

    printf("Enter two integers: ");
    scanf("%d %d",&n1,&n2);

    // if user enters negative number, sign of the number is changed to positive
    n1 = ( n1 > 0) ? n1 : -n1;
    n2 = ( n2 > 0) ? n2 : -n2;

    while(n1!=n2)
    {
        if(n1 > n2)
            n1 -= n2;
        else
            n2 -= n1;
    }
    printf("GCD = %d",n1);

    return 0;
}
```

Output
Enter two integers: 81
-153
GCD = 9

21 How to Find LCM of two Numbers

In this example, you will learn to calculate the LCM (Lowest common multiple) of two numbers entered by the user.

The LCM (Least Common Multiple) of two integers n1 and n2 is the smallest positive integer that is perfectly divisible by both n1 and n2 (without a remainder). For example, the LCM of 72 and 120 is 360.

LCM using while and if

```
#include <stdio.h>
int main() {
   int n1, n2, max;
   printf("Enter two positive integers: ");
   scanf("%d %d", &n1, &n2);

   // maximum number between n1 and n2 is stored in max
   max = (n1 > n2) ? n1 : n2;

   while (1) {
      if (max % n1 == 0 && max % n2 == 0) {
         printf("The LCM of %d and %d is %d.", n1, n2, max);
         break;
      }
      ++max;
   }
   return 0;
}
```

Output
Enter two positive integers: 72
120
The LCM of 72 and 120 is 360.

In this program, the integers entered by the user are stored in variable n1 and n2 respectively. The largest number among n1 and n2 is stored in max. The LCM of two numbers cannot be less than max.

The test expression of while loop is always true.
In each iteration, whether max is perfectly divisible by n1 and n2 is checked.
if (min % n1 == 0 && max% n2 == 0) { ... }

If this test condition is not true, max is incremented by 1 and the iteration continues until the test expression of the if statement is true.

The LCM of two numbers can also be found using the formula:
LCM = (num1*num2)/GCD
Learn how to find the GCD of two numbers in C programming.

LCM Calculation Using GCD

```
#include <stdio.h>
int main() {
   int n1, n2, i, gcd, lcm;
   printf("Enter two positive integers: ");
   scanf("%d %d", &n1, &n2);

   for (i = 1; i <= n1 && i <= n2; ++i) {

      // check if i is a factor of both integers
      if (n1 % i == 0 && n2 % i == 0)
         gcd = i;
   }

   lcm = (n1 * n2) / gcd;

   printf("The LCM of two numbers %d and %d is %d.", n1, n2, lcm);
   return 0;
}
```

Output

Enter two positive integers: 72
120
The LCM of two numbers 72 and 120 is 360.

22 How to Display Characters from A to Z Using Loop

In this example, you will learn to print all the letters of the English alphabet.

Program to Print English Alphabets

```
#include <stdio.h>
int main() {
   char c;
   for (c = 'A'; c <= 'Z'; ++c)
      printf("%c ", c);
   return 0;
}
```

Output

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

In this program, the for loop is used to display the English alphabet in uppercase.
Here's a little modification of the above program. The program displays the English alphabet in either uppercase or lowercase depending upon the input given by the user.

Print Lowercase/Uppercase alphabets

```
#include <stdio.h>
int main() {
   char c;
   printf("Enter u to display uppercase alphabets.\n");
   printf("Enter l to display lowercase alphabets. \n");
   scanf("%c", &c);

   if (c == 'U' || c == 'u') {
      for (c = 'A'; c <= 'Z'; ++c)
         printf("%c ", c);
   } else if (c == 'L' || c == 'l') {
      for (c = 'a'; c <= 'z'; ++c)
         printf("%c ", c);
   } else {
      printf("Error! You entered an invalid character.");
   }

   return 0;
}
```

Output

Enter u to display uppercase alphabets.
Enter l to display lowercase alphabets. l
a b c d e f g h i j k l m n o p q r s t u v w x y z

23 How to Count Number of Digits in an Integer

In this example, you will learn to count the number of digits in an integer entered by the user.

This program takes an integer from the user and calculates the number of digits. For example: If the user enters 2319, the output of the program will be 4.

Program to Count the Number of Digits

```
#include <stdio.h>
int main() {
   long long n;
   int count = 0;
   printf("Enter an integer: ");
   scanf("%lld", &n);

   // iterate until n becomes 0
   // remove last digit from n in each iteration
   // increase count by 1 in each iteration
   while (n != 0) {
      n /= 10;    // n = n/10
      ++count;
   }
   printf("Number of digits: %d", count);
}
```

Output
Enter an integer: 3452
Number of digits: 4

The integer entered by the user is stored in variable n. Then the while loop is iterated until the test expression n! = 0 is evaluated to 0 (false).

- After the first iteration, the value of n will be 345 and the count is incremented to 1.
- After the second iteration, the value of n will be 34 and the count is incremented to 2.
- After the third iteration, the value of n will be 3 and the count is incremented to 3.
- After the fourth iteration, the value of n will be 0 and the count is incremented to 4.
- Then the test expression of the loop is evaluated to false and the loop terminates.

24 How to Reverse a Number

In this example, you will learn to reverse the number entered by the user.

Reverse an Integer

```
#include <stdio.h>
int main() {
   int n, rev = 0, remainder;
   printf("Enter an integer: ");
   scanf("%d", &n);
   while (n != 0) {
      remainder = n % 10;
      rev = rev * 10 + remainder;
      n /= 10;
   }
   printf("Reversed number = %d", rev);
   return 0;
}
```

Output
Enter an integer: 2345
Reversed number = 5432

This program takes an integer input from the user. Then the while loop is used until n != 0 is false (0).

In each iteration of the loop, the remainder when n is divided by 10 is calculated and the value of n is reduced by 10 times.

Inside the loop, the reversed number is computed using:
rev = rev*10 + remainder;

25 How to Calculate the Power of a Number

In this example, you will learn to calculate the power of a number.

The program below takes two integers from the user (a base number and an exponent) and calculates the power.

For example: In the case of 23
2 is the base number
3 is the exponent
And, the power is equal to 2*2*2

Power of a Number Using the while Loop

```
#include <stdio.h>
int main() {
   int base, exp;
   long double result = 1.0;
   printf("Enter a base number: ");
   scanf("%d", &base);
   printf("Enter an exponent: ");
   scanf("%d", &exp);

   while (exp != 0) {
      result *= base;
      --exp;
   }
   printf("Answer = %.0Lf", result);
   return 0;
}
```

Output
Enter a base number: 3
Enter an exponent: 4
Answer = 81