**C "Hello, World!" Program**

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

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Input Output (I/O)](https://www.programiz.com/c-programming/c-input-output)

## 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.

# C Program 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.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Variables, Constants and Literals](https://www.programiz.com/c-programming/c-variables-constants)
* [C Data Types](https://www.programiz.com/c-programming/c-data-types)
* [C Input Output (I/O)](https://www.programiz.com/c-programming/c-input-output)

## Program to Print an Integer

#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);

# C Program 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.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Data Types](https://www.programiz.com/c-programming/c-data-types)
* [C Variables, Constants and Literals](https://www.programiz.com/c-programming/c-variables-constants)
* [C Input Output (I/O)](https://www.programiz.com/c-programming/c-input-output)
* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)

## Program to Add Two Integers

#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

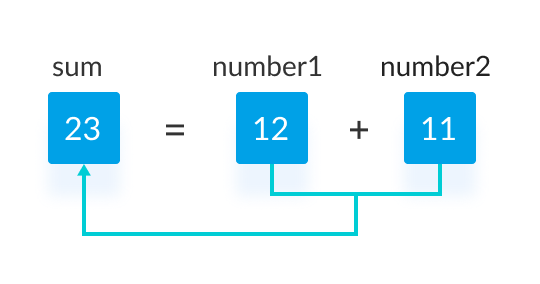
In this program, the user is asked to enter two integers. These two integers are stored in variables number1 and number2 respectively.

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;

Add Two Numbers

Finally, the printf() function is used to display the sum of numbers.

printf("%d + %d = %d", number1, number2, sum);

# C Program 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.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Variables, Constants and Literals](https://www.programiz.com/c-programming/c-variables-constants)
* [C Data Types](https://www.programiz.com/c-programming/c-data-types)
* [C Input Output (I/O)](https://www.programiz.com/c-programming/c-input-output)
* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)

## Program to Multiply Two Numbers

#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.

# C Program to Find ASCII Value of a Character

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

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Data Types](https://www.programiz.com/c-programming/c-data-types)
* [C Variables, Constants and Literals](https://www.programiz.com/c-programming/c-variables-constants)
* [C Input Output (I/O)](https://www.programiz.com/c-programming/c-input-output)

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.

Now, let's see how we can print the ASCII value of characters in C programming.

## Program to Print ASCII Value

#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.

# C Program 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.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Data Types](https://www.programiz.com/c-programming/c-data-types)
* [C Variables, Constants and Literals](https://www.programiz.com/c-programming/c-variables-constants)
* [C Input Output (I/O)](https://www.programiz.com/c-programming/c-input-output)
* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)

## Program to Compute Quotient and Remainder

#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);

# C Program 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.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Data Types](https://www.programiz.com/c-programming/c-data-types)
* [C Variables, Constants and Literals](https://www.programiz.com/c-programming/c-variables-constants)
* [C Input Output (I/O)](https://www.programiz.com/c-programming/c-input-output)
* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)

## Program to Compute Quotient and Remainder

#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);

# C Program 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.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Data Types](https://www.programiz.com/c-programming/c-data-types)
* [C Variables, Constants and Literals](https://www.programiz.com/c-programming/c-variables-constants)
* [C Input Output (I/O)](https://www.programiz.com/c-programming/c-input-output)

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.

## Program to Find the Size of Variables

#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.

# C Program to Demonstrate the Working of Keyword long

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

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Data Types](https://www.programiz.com/c-programming/c-data-types)
* [C Variables, Constants and Literals](https://www.programiz.com/c-programming/c-variables-constants)
* [C Input Output (I/O)](https://www.programiz.com/c-programming/c-input-output)

## Program Using 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.

# C Program to Swap Two Numbers

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

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Data Types](https://www.programiz.com/c-programming/c-data-types)
* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)
* [C Input Output (I/O)](https://www.programiz.com/c-programming/c-input-output)

## 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, first number = %.2lf\n", first);

printf("After swapping, second number = %.2lf", second);

return 0;

}

**Output**

Enter first number: 1.20

Enter second number: 2.45

After swapping, first number = 2.45

After swapping, second number = 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 Variables

#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 numbers up to 2 decimal places

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

# C Program 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.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)
* [C if...else Statement](https://www.programiz.com/c-programming/c-if-else-statement)

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

## Program to Check Even or Odd

#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.

## Program to Check Odd or Even Using the Ternary Operator

#include <stdio.h>

int main() {

int num;

printf("Enter an integer: ");

scanf("%d", &num);

(num % 2 == 0) ? printf("%d is even.", num) : printf("%d is odd.", num);

return 0;

}

**Output**

Enter an integer: 33

33 is odd.

In the above program, we have used the ternary operator ?: instead of the if...else statement.

# C Program 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.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)
* [C if...else Statement](https://www.programiz.com/c-programming/c-if-else-statement)
* [C while and do...while Loop](https://www.programiz.com/c-programming/c-do-while-loops)

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.

## Program to Check Vowel or consonant

#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.

**Note:** This program assumes that the user will enter an alphabet. If the user enters a non-alphabetic character, it displays the character is a consonant.

To fix this, we can use the [isalpha()](https://www.programiz.com/c-programming/library-function/ctype.h/isalpha" \o "C isalpha() Function) function. The islapha() function checks whether a character is an alphabet or not.

#include <ctype.h>

#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');

// Show error message if c is not an alphabet

if (!isalpha(c))

printf("Error! Non-alphabetic character.");

else if (lowercase\_vowel || uppercase\_vowel)

printf("%c is a vowel.", c);

else

printf("%c is a consonant.", c);

return 0;

}

Now, if the user enters a non-alphabetic character, you will see:

Enter an alphabet: 3

Error! Non-alphabetic character.

# C Program 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.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)
* [C if...else Statement](https://www.programiz.com/c-programming/c-if-else-statement)

## 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;

}

Here, we have used 3 different if statements. The first one checks whether n1 is the largest number.

The second and third if statements check if n2 and n3 are the largest, respectively.

The biggest drawback of this program is that all 3  if statements are executed, regardless of which number is the largest.

However, we want to execute only one if statement. We can do this by using an if...else ladder.

## 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;

}

In this program, only the if statement is executed when n1 is the largest.

Similarly, only the else if statement is executed when n2 is the largest, and so on.

## 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);

// outer if statement

if (n1 >= n2) {

// inner if...else

if (n1 >= n3)

printf("%.2lf is the largest number.", n1);

else

printf("%.2lf is the largest number.", n3);

}

// outer else statement

else {

// inner if...else

if (n2 >= n3)

printf("%.2lf is the largest number.", n2);

else

printf("%.2lf is the largest number.", n3);

}

return 0;

}

In this program, we have used nested if...else statements to find the largest number. Let's see how they work in greater detail.

**1. Outer if Statement**

First, notice the outer if statement and the inner if...else statement inside it:

// outer if statement

if (n1 >= n2) {

// inner if...else

if (n1 >= n3)

printf("%.2lf is the largest number.", n1);

else

printf("%.2lf is the largest number.", n3);

}

Here, we are checking if n1 is greater than or equal to n2. If it is, the program control goes to the inner if...else statement.

The inner if statement checks whether n1 is also greater than or equal to n3.

If it is, then n1 is either equal to both n2 and n3, or it is now greater than both n2 and n3 i.e. n1 >= n2 >= n3. Hence, n1 is the largest number.

Else, n1 is greater than or equal to n2 but it is less than n3 i.e. n3 > n1 >= n2. Hence, n3 is the largest number.

**2. Outer else Statement**

The outer else statement is executed when n2 > n1:

// outer else statement

else {

// inner if...else

if (n2 >= n3)

printf("%.2lf is the largest number.", n2);

else

printf("%.2lf is the largest number.", n3);

}

The inner if...else of this part of the program uses the same logic as the one before. The only difference here is that we're checking if n2 is greater than n3.

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.

# C Program to Find the Roots of a Quadratic Equation

In this example, you will learn to find the roots of a quadratic equation in C programming.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)
* [C if...else Statement](https://www.programiz.com/c-programming/c-if-else-statement)

The standard form of a quadratic equation is:

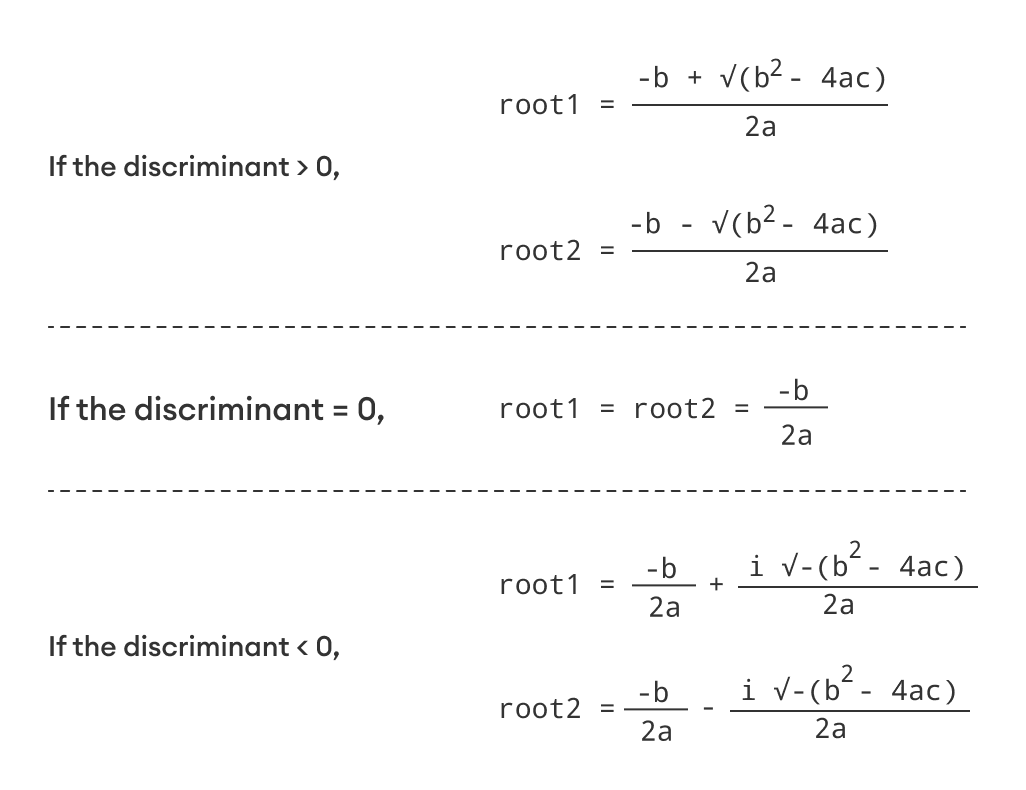
ax2 + bx + c = 0, where

a, b and c are real numbers and

a != 0

The term b2; - 4ac is known as the **discriminant** of a quadratic equation. It tells the nature of the roots.

* If the discriminant is greater than 0, the roots are real and different.
* If the discriminant is equal to 0, the roots are real and equal.
* If the discriminant is less than 0, the roots are complex and different.

Figure: Roots of a Quadratic Equation

## Program to Find Roots of a Quadratic Equation

#include <math.h>

#include <stdio.h>

int main() {

double a, b, c, discriminant, root1, root2, realPart, imagPart;

printf("Enter coefficients a, b and c: ");

scanf("%lf %lf %lf", &a, &b, &c);

discriminant = b \* b - 4 \* a \* c;

// condition for real and different roots

if (discriminant > 0) {

root1 = (-b + sqrt(discriminant)) / (2 \* a);

root2 = (-b - sqrt(discriminant)) / (2 \* a);

printf("root1 = %.2lf and root2 = %.2lf", root1, root2);

}

// condition for real and equal roots

else if (discriminant == 0) {

root1 = root2 = -b / (2 \* a);

printf("root1 = root2 = %.2lf;", root1);

}

// if roots are not real

else {

realPart = -b / (2 \* a);

imagPart = sqrt(-discriminant) / (2 \* a);

printf("root1 = %.2lf+%.2lfi and root2 = %.2f-%.2fi", realPart, imagPart, realPart, imagPart);

}

return 0;

}

**Output**

Enter coefficients a, b and c: 2.3

4

5.6

root1 = -0.87+1.30i and root2 = -0.87-1.30i

In this program, the sqrt() library function is used to find the square root of a number. To learn more, visit: [sqrt() function](https://www.programiz.com/c-programming/library-function/math.h/sqrt).

# C Program 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.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)
* [C if...else Statement](https://www.programiz.com/c-programming/c-if-else-statement)

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

## Program to Check 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.

# C Program 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.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)
* [C if...else Statement](https://www.programiz.com/c-programming/c-if-else-statement)

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

## Check Positive or Negative Using Nested if...else

#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](https://www.programiz.com/c-programming/c-if-else-statement).

## Check Positive or Negative Using if...else Ladder

#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.

# C Program 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.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)
* [C if...else Statement](https://www.programiz.com/c-programming/c-if-else-statement)

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.

## Program to Check 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

In the program, 'a' is used instead of 97 and 'z' is used instead of 122. Similarly, 'A' is used instead of 65 and 'Z' is used instead of 90.

**Note:** It is recommended we use the [isalpha()](https://www.programiz.com/c-programming/library-function/ctype.h/isalpha) function to check whether a character is an alphabet or not.

if (isalpha(c))

printf("%c is an alphabet.", c);

else

printf("%c is not an alphabet.", c);

# C Program to Calculate the Sum of Natural Numbers

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

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C for Loop](https://www.programiz.com/c-programming/c-for-loop)
* [C while and do...while Loop](https://www.programiz.com/c-programming/c-do-while-loops)

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

## Sum of Natural Numbers Using for Loop

#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;

}

Visit this page to learn [how to find the sum of natural numbers using recursion](https://www.programiz.com/c-programming/examples/natural-number-sum-recursion).

# C Program to Find Factorial of a Number

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

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Data Types](https://www.programiz.com/c-programming/c-data-types)
* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)
* [C if...else Statement](https://www.programiz.com/c-programming/c-if-else-statement)
* [C for Loop](https://www.programiz.com/c-programming/c-for-loop)

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.

You can also find the [factorial of a number using recursion](https://www.programiz.com/c-programming/examples/factorial-recursion).

# C Program to Generate Multiplication Table

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

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)
* [C for Loop](https://www.programiz.com/c-programming/c-for-loop)

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

## Multiplication Table Up to 10

#include <stdio.h>

int main() {

int n;

printf("Enter an integer: ");

scanf("%d", &n);

for (int 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

Here, the user input is stored in the int variable n. Then, we use a for loop to print the multiplication table up to 10.

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

printf("%d \* %d = %d \n", n, i, n \* i);

}

The loop runs from i = 1 to i = 10. In each iteration of the loop, n \* i is printed.

Here's a little modification of the above program to generate the multiplication table up to a range (where range is also a positive integer entered by the user).

## Multiplication Table Up to a range

#include <stdio.h>

int main() {

int n, i, range;

printf("Enter an integer: ");

scanf("%d", &n);

// prompt user for positive range

do {

printf("Enter the range (positive integer): ");

scanf("%d", &range);

} while (range <= 0);

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

printf("%d \* %d = %d \n", n, i, n \* i);

}

return 0;

}

**Output**

Enter an integer: 12

Enter the range (positive integer): -8

Enter the range (positive integer): 8

12 \* 1 = 12

12 \* 2 = 24

12 \* 3 = 36

12 \* 4 = 48

12 \* 5 = 60

12 \* 6 = 72

12 \* 7 = 84

12 \* 8 = 96

Here, we have used a do...while loop to prompt the user for a positive range.

// prompt user for positive range

do {

printf("Enter the range (positive integer): ");

scanf("%d", &range);

} while (range <= 0);

If the value of range is negative, the loop iterates again to ask the user to enter a positive number. Once a positive range has been entered, we print the multiplication table.

# C Program to Display Fibonacci Sequence

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

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)
* [C while and do...while Loop](https://www.programiz.com/c-programming/c-do-while-loops)
* [C for Loop](https://www.programiz.com/c-programming/c-for-loop)
* [C break and continue](https://www.programiz.com/c-programming/c-break-continue-statement)

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

Visit this page to learn about [the Fibonacci sequence](https://www.mathsisfun.com/numbers/fibonacci-sequence.html).

## Fibonacci Series up to n terms

#include <stdio.h>

int main() {

int i, n;

// initialize first and second terms

int t1 = 0, t2 = 1;

// initialize the next term (3rd term)

int nextTerm = t1 + t2;

// get no. of terms from user

printf("Enter the number of terms: ");

scanf("%d", &n);

// print the first two terms t1 and t2

printf("Fibonacci Series: %d, %d, ", t1, t2);

// print 3rd to nth terms

for (i = 3; 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,

Let us suppose n = 10. First, we have printed the first two terms of the Fibonacci sequence before using a for loop to print the next n terms.

Let us see how the for loop works:

|  |  |  |  |
| --- | --- | --- | --- |
| i | t1 | t2 | nextTerm |
| 3 | 0 | 1 | 1 |
| 4 | 1 | 1 | 2 |
| 5 | 1 | 2 | 3 |
| 6 | 2 | 3 | 5 |
| 7 | 3 | 5 | 8 |
| 8 | 5 | 8 | 13 |
| 9 | 8 | 13 | 21 |
| 10 | 13 | 21 | 34 |

## Fibonacci Sequence Up to a Certain Number

#include <stdio.h>

int main() {

int t1 = 0, t2 = 1, nextTerm = 0, n;

printf("Enter a positive number: ");

scanf("%d", &n);

// displays the first two terms which is always 0 and 1

printf("Fibonacci Series: %d, %d, ", t1, t2);

nextTerm = t1 + t2;

while (nextTerm <= n) {

printf("%d, ", nextTerm);

t1 = t2;

t2 = nextTerm;

nextTerm = t1 + t2;

}

return 0;

}

**Output**

Enter a positive integer: 100

Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89,

In this program, we have used a while loop to print all the Fibonacci numbers up to n.

If n is not part of the Fibonacci sequence, we print the sequence up to the number that is closest to (and lesser than) n.

Suppose n = 100. First, we print the first two terms t1 = 0 and t2 = 1.

Then the while loop prints the rest of the sequence using the nextTerm variable:

|  |  |  |  |
| --- | --- | --- | --- |
| t1 | t2 | nextTerm | nextTerm <= n |
| 0 | 1 | 1 | true. Print nextTerm. |
| 1 | 1 | 2 | true. Print nextTerm. |
| 1 | 2 | 3 | true. Print nextTerm. |
| ... | ... | ... | ... |
| 34 | 55 | 89 | true. Print nextTerm. |
| 55 | 89 | 144 | false. Terminate Loop. |

# C Program 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.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)
* [C if...else Statement](https://www.programiz.com/c-programming/c-if-else-statement)
* [C while and do...while Loop](https://www.programiz.com/c-programming/c-do-while-loops)

The LCM 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, we check whether max is perfectly divisible by n1 and n2.

if ((max % n1 == 0) && (max % n2 == 0)) {

// code

}

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.

## LCM Calculation Using GCD

We can also find the LCM of two numbers num1 and num2 using their GCD:

LCM = (num1 \* num2) / GCD

Learn how to find [the GCD of two numbers in C programming](https://www.programiz.com/c-programming/examples/hcf-gcd) before finding the LCM with this method.

#include <stdio.h>

int main() {

int n1, n2, i, gcd, lcm;

printf("Enter two positive integers: ");

scanf("%d %d", &n1, &n2);

// loop to find the GCD

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.

# C Program to Display Characters from A to Z Using Loop

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

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C if...else Statement](https://www.programiz.com/c-programming/c-if-else-statement)
* [C while and do...while Loop](https://www.programiz.com/c-programming/c-do-while-loops)

## 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

# C Program 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.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)
* [C while and do...while Loop](https://www.programiz.com/c-programming/c-do-while-loops)

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 at least once, then until n becomes 0

// remove last digit from n in each iteration

// increase count by 1 in each iteration

do {

n /= 10;

++count;

} while (n != 0);

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 do...while [loop](https://www.programiz.com/c-programming/c-do-while-loops) 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.

**Note:** We have used a do...while loop to ensure that we get the correct digit count when the user enters **0**.

# C Program to Reverse a Number

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

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)
* [C while and do...while Loop](https://www.programiz.com/c-programming/c-do-while-loops)

## Reverse an Integer

#include <stdio.h>

int main() {

int n, reverse = 0, remainder;

printf("Enter an integer: ");

scanf("%d", &n);

while (n != 0) {

remainder = n % 10;

reverse = reverse \* 10 + remainder;

n /= 10;

}

printf("Reversed number = %d", reverse);

return 0;

}

**Output**

Enter an integer: 2345

Reversed number = 5432

This program takes 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:

reverse = reverse \* 10 + remainder;

Let us see how the while loop works when n = 2345.

|  |  |  |  |
| --- | --- | --- | --- |
| n | n != 0 | remainder | reverse |
| 2345 | true | 5 | 0 \* 10 + 5 = 5 |
| 234 | true | 4 | 5 \* 10 + 4 = 54 |
| 23 | true | 3 | 54 \* 10 + 3 = 543 |
| 2 | true | 2 | 543 \* 10 + 2 = 5432 |
| 0 | false | - | Loop terminates. |

Finally, the reverse variable (which contains the reversed number) is printed on the screen.

# C Program to Calculate the Power of a Number

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

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)
* [C while and do...while Loop](https://www.programiz.com/c-programming/c-do-while-loops)

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

We can also use the pow() function to calculate the power of a number.

## Power Using pow() Function

#include <math.h>

#include <stdio.h>

int main() {

double base, exp, result;

printf("Enter a base number: ");

scanf("%lf", &base);

printf("Enter an exponent: ");

scanf("%lf", &exp);

// calculates the power

result = pow(base, exp);

printf("%.1lf^%.1lf = %.2lf", base, exp, result);

return 0;

}

**Output**

Enter a base number: 2.3

Enter an exponent: 4.5

2.3^4.5 = 42.44

The programs above can only calculate the power of the base number if the exponent is positive. For negative exponents, use the following mathematical logic:

base(-exponent) = 1 / (baseexponent)

For example,

2-3 = 1 / (23)

# C Program to Check Whether a Number is Palindrome or Not

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

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)
* [C if...else Statement](https://www.programiz.com/c-programming/c-if-else-statement)
* [C while and do...while Loop](https://www.programiz.com/c-programming/c-do-while-loops)

An integer is a palindrome if the reverse of that number is equal to the original number.

## Program to Check Palindrome

#include <stdio.h>

int main() {

int n, reversed = 0, remainder, original;

printf("Enter an integer: ");

scanf("%d", &n);

original = n;

// reversed integer is stored in reversed variable

while (n != 0) {

remainder = n % 10;

reversed = reversed \* 10 + remainder;

n /= 10;

}

// palindrome if orignal and reversed are equal

if (original == reversed)

printf("%d is a palindrome.", original);

else

printf("%d is not a palindrome.", original);

return 0;

}

**Output**

Enter an integer: 1001

1001 is a palindrome.

Here, the user is asked to enter an integer. The number is stored in variable n.

We then assigned this number to another variable orignal. Then, the reverse of n is found and stored in reversed.

If original is equal to reversed, the number entered by the user is a palindrome.

# C Program to Check Whether a Number is Prime or Not

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

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C if...else Statement](https://www.programiz.com/c-programming/c-if-else-statement)
* [C for Loop](https://www.programiz.com/c-programming/c-for-loop)
* [C break and continue](https://www.programiz.com/c-programming/c-break-continue-statement)

A prime number is a positive integer that is divisible only by **1** and itself. For example: 2, 3, 5, 7, 11, 13, 17.

## Program to Check Prime Number

#include <stdio.h>

int main() {

int n, i, flag = 0;

printf("Enter a positive integer: ");

scanf("%d", &n);

// 0 and 1 are not prime numbers

// change flag to 1 for non-prime number

if (n == 0 || n == 1)

flag = 1;

for (i = 2; i <= n / 2; ++i) {

// if n is divisible by i, then n is not prime

// change flag to 1 for non-prime number

if (n % i == 0) {

flag = 1;

break;

}

}

// flag is 0 for prime numbers

if (flag == 0)

printf("%d is a prime number.", n);

else

printf("%d is not a prime number.", n);

return 0;

}

**Output**

Enter a positive integer: 29

29 is a prime number.

In the program, a for loop is iterated from i = 2 to i < n/2.

In each iteration, whether n is perfectly divisible by i is checked using:

if (n % i == 0) {

flag = 1;

break;

}

If n is perfectly divisible by i, n is not a prime number. In this case, flag is set to **1**, and the loop is terminated using the break statement.

Notice that we have initialized flag as **0** during the start of our program.

So, if n is a prime number after the loop, flag will still be **0**. However, if n is a non-prime number, flag will be **1**.

Visit this page to learn how you can [print all the prime numbers between two intervals](https://www.programiz.com/c-programming/examples/prime-number-intervals).

# C Program to Display Prime Numbers Between Two Intervals

In this example, you will learn to print all prime numbers between two numbers entered by the user.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C if...else Statement](https://www.programiz.com/c-programming/c-if-else-statement)
* [C for Loop](https://www.programiz.com/c-programming/c-for-loop)
* [C break and continue](https://www.programiz.com/c-programming/c-break-continue-statement)

## Display Prime Numbers Between Two Intervals

#include <stdio.h>

int main() {

int low, high, i, flag;

printf("Enter two numbers(intervals): ");

scanf("%d %d", &low, &high);

printf("Prime numbers between %d and %d are: ", low, high);

// iteration until low is not equal to high

while (low < high) {

flag = 0;

// ignore numbers less than 2

if (low <= 1) {

++low;

continue;

}

// if low is a non-prime number, flag will be 1

for (i = 2; i <= low / 2; ++i) {

if (low % i == 0) {

flag = 1;

break;

}

}

if (flag == 0)

printf("%d ", low);

// to check prime for the next number

// increase low by 1

++low;

}

return 0;

}

**Output**

Enter two numbers(intervals): 20

50

Prime numbers between 20 and 50 are: 23 29 31 37 41 43 47

In this program, the while loop is iterated ( high-low-1) times.

In each iteration, whether low is a prime number or not is checked, and the value of low is incremented by 1 until low is equal to high.

Visit this page to learn more about how to [check whether a number is prime or not](https://www.programiz.com/c-programming/examples/prime-number).

If the user enters the larger number first, the above program doesn't work as intended. You can solve this issue by [swapping the numbers](https://www.programiz.com/c-programming/examples/swapping).

## Display Prime Numbers when Larger Number is Entered first

#include <stdio.h>

int main() {

int low, high, i, flag, temp;

printf("Enter two numbers(intervals): ");

scanf("%d %d", &low, &high);

// swap numbers if low is greather than high

if (low > high) {

temp = low;

low = high;

high = temp;

}

printf("Prime numbers between %d and %d are: ", low, high);

while (low < high) {

flag = 0;

// ignore numbers less than 2

if (low <= 1) {

++low;

continue;

}

for (i = 2; i <= low / 2; ++i) {

if (low % i == 0) {

flag = 1;

break;

}

}

if (flag == 0)

printf("%d ", low);

++low;

}

return 0;

}

Visit this page to learn how you can [display all the prime numbers between the two intervals by creating a user-defined function](https://www.programiz.com/c-programming/examples/prime-interval-function)

# C Program to Check Armstrong Number

In this example, you will learn to check whether an integer entered by the user is an Armstrong number or not.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C if...else Statement](https://www.programiz.com/c-programming/c-if-else-statement)
* [C while and do...while Loop](https://www.programiz.com/c-programming/c-do-while-loops)

A positive integer is called an Armstrong number (of order n) if

abcd... = an + bn + cn + dn +

In the case of an Armstrong number of 3 digits, the sum of cubes of each digit is equal to the number itself. For example, 153 is an Armstrong number because

153 = 1\*1\*1 + 5\*5\*5 + 3\*3\*3

## Check Armstrong Number of three digits

#include <stdio.h>

int main() {

int num, originalNum, remainder, result = 0;

printf("Enter a three-digit integer: ");

scanf("%d", &num);

originalNum = num;

while (originalNum != 0) {

// remainder contains the last digit

remainder = originalNum % 10;

result += remainder \* remainder \* remainder;

// removing last digit from the orignal number

originalNum /= 10;

}

if (result == num)

printf("%d is an Armstrong number.", num);

else

printf("%d is not an Armstrong number.", num);

return 0;

}

**Output**

Enter a three-digit integer: 371

371 is an Armstrong number.

## Check Armstrong Number of n digits

#include <math.h>

#include <stdio.h>

int main() {

int num, originalNum, remainder, n = 0;

float result = 0.0;

printf("Enter an integer: ");

scanf("%d", &num);

originalNum = num;

// store the number of digits of num in n

for (originalNum = num; originalNum != 0; ++n) {

originalNum /= 10;

}

for (originalNum = num; originalNum != 0; originalNum /= 10) {

remainder = originalNum % 10;

// store the sum of the power of individual digits in result

result += pow(remainder, n);

}

// if num is equal to result, the number is an Armstrong number

if ((int)result == num)

printf("%d is an Armstrong number.", num);

else

printf("%d is not an Armstrong number.", num);

return 0;

}

**Output**

Enter an integer: 1634

1634 is an Armstrong number.

In this program, the number of digits of an integer is calculated first and stored in n. And, the pow() function is used to compute the power of individual digits in each iteration of the second for loop.

# C Program to Display Armstrong Number Between Two Intervals

In this example, you will learn to find all Armstrong numbers between two integers entered by the user.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C if...else Statement](https://www.programiz.com/c-programming/c-if-else-statement)
* [C for Loop](https://www.programiz.com/c-programming/c-for-loop)

A positive integer is called an Armstrong number (of order n) if

abcd... = an + bn + cn + dn +

In the case of an Armstrong number of 3 digits, the sum of cubes of each digit is equal to the number itself. For example, 153 is an Armstrong number because

153 = 1\*1\*1 + 5\*5\*5 + 3\*3\*3

In this program, we will print all the Armstrong numbers **between** two integers. This means that the two integers will not be part of the range, but only those integers that are between them.

For example, suppose we want to print all Armstrong numbers between **153** and **371**. Both of these numbers are also Armstrong numbers.

Then, this program prints all Armstrong numbers that are greater than **153** and less than **371** i.e. **153** and **371** won't be printed even though they are Armstrong numbers.

**Tip:** Before trying this program, learn how to [check whether an integer is an Armstrong number or not](https://www.programiz.com/c-programming/examples/check-armstrong-number).

## Armstrong Numbers Between Two Integers

#include <math.h>

#include <stdio.h>

int main() {

int low, high, number, originalNumber, rem, count = 0;

double result = 0.0;

printf("Enter two numbers(intervals): ");

scanf("%d %d", &low, &high);

printf("Armstrong numbers between %d and %d are: ", low, high);

// swap numbers if high < low

if (high < low) {

high += low;

low = high - low;

high -= low;

}

// iterate number from (low + 1) to (high - 1)

// In each iteration, check if number is Armstrong

for (number = low + 1; number < high; ++number) {

originalNumber = number;

// number of digits calculation

while (originalNumber != 0) {

originalNumber /= 10;

++count;

}

originalNumber = number;

// result contains sum of nth power of individual digits

while (originalNumber != 0) {

rem = originalNumber % 10;

result += pow(rem, count);

originalNumber /= 10;

}

// check if number is equal to the sum of nth power of individual digits

if ((int)result == number) {

printf("%d ", number);

}

// resetting the values

count = 0;

result = 0;

}

return 0;

}

[Run Code](https://www.programiz.com/c-programming/online-compiler)

**Output**

Enter two numbers(intervals): 200

2000

Armstrong numbers between 200 and 2000 are: 370 371 407 1634

In the program, the outer loop is iterated from **(low+ 1)** to **(high - 1)**. In each iteration, it's checked whether number is an Armstrong number or not.

Inside the outer loop, the number of digits of an integer is calculated first and stored in count. And, the sum of the power of individual digits is stored in the result variable.

If number is equal to result, the number is an Armstrong number.

**Notes:**

* You need to swap low and high if the user input for high is less than that of low. To learn more, check our example on [swapping two numbers](https://www.programiz.com/c-programming/examples/swapping).
* You need to reset count and result to 0 in each iteration of the outer loop.

# C Program to Display Factors of a Number

In this example, you will learn to find all the factors of an integer entered by the user.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Programming Operators](https://www.programiz.com/c-programming/c-operators)
* [C if...else Statement](https://www.programiz.com/c-programming/c-if-else-statement)
* [C for Loop](https://www.programiz.com/c-programming/c-for-loop)

This program takes a positive integer from the user and displays all the positive factors of that number.

## Factors of a Positive Integer

#include <stdio.h>

int main() {

int num, i;

printf("Enter a positive integer: ");

scanf("%d", &num);

printf("Factors of %d are: ", num);

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

if (num % i == 0) {

printf("%d ", i);

}

}

return 0;

}

**Output**

Enter a positive integer: 60

Factors of 60 are: 1 2 3 4 5 6 10 12 15 20 30 60

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

The for loop is iterated until i is false.

In each iteration, whether num is exactly divisible by i is checked. It is the condition for i to be a factor of num.

if (num % i == 0) {

printf("%d ", i);

}

Then the value of i is incremented by 1.

# C Program to Make a Simple Calculator Using switch...case

In this example, you will learn to create a simple calculator in C programming using the switch statement.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C switch Statement](https://www.programiz.com/c-programming/c-switch-case-statement)
* [C break and continue](https://www.programiz.com/c-programming/c-break-continue-statement)

This program takes an arithmetic operator +, -, \*, / and two operands from the user. Then, it performs the calculation on the two operands depending upon the operator entered by the user.

## Simple Calculator using switch Statement

#include <stdio.h>

int main() {

char op;

double first, second;

printf("Enter an operator (+, -, \*, /): ");

scanf("%c", &op);

printf("Enter two operands: ");

scanf("%lf %lf", &first, &second);

switch (op) {

case '+':

printf("%.1lf + %.1lf = %.1lf", first, second, first + second);

break;

case '-':

printf("%.1lf - %.1lf = %.1lf", first, second, first - second);

break;

case '\*':

printf("%.1lf \* %.1lf = %.1lf", first, second, first \* second);

break;

case '/':

printf("%.1lf / %.1lf = %.1lf", first, second, first / second);

break;

// operator doesn't match any case constant

default:

printf("Error! operator is not correct");

}

return 0;

}

**Output**

Enter an operator (+, -, \*,): \*

Enter two operands: 1.5

4.5

1.5 \* 4.5 = 6.8

The \* operator entered by the user is stored in op. And, the two operands, 1.5 and 4.5 are stored in first and second respectively.

Since the operator \* matches case '\*':, the control of the program jumps to

printf("%.1lf \* %.1lf = %.1lf", first, second, first \* second);

This statement calculates the product and displays it on the screen.

To make our output look cleaner, we have simply limited the output to one decimal place using the code %.1lf.

Finally, the break; statement ends the switch statement.

# C Program to Display Prime Numbers Between Intervals Using Function

In this example, you will learn to print all prime numbers between two numbers (entered by the user).

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C for Loop](https://www.programiz.com/c-programming/c-for-loop)
* [C break and continue](https://www.programiz.com/c-programming/c-break-continue-statement)
* [C Functions](https://www.programiz.com/c-programming/c-functions)
* [C User-defined functions](https://www.programiz.com/c-programming/c-user-defined-functions)

Make sure you visit these tutorials before looking at this example:

* [Check Whether a Number is Prime or Not](https://www.programiz.com/c-programming/examples/prime-number)
* [Swap Two Numbers](https://www.programiz.com/c-programming/examples/swapping#example2)

## Prime Numbers Between Two Integers

#include <stdio.h>

int checkPrimeNumber(int n);

int main() {

int n1, n2, i, flag;

printf("Enter two positive integers: ");

scanf("%d %d", &n1, &n2);

// swap n1 and n2 if n1 > n2

if (n1 > n2) {

n1 = n1 + n2;

n2 = n1 - n2;

n1 = n1 - n2;

}

printf("Prime numbers between %d and %d are: ", n1, n2);

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

// flag will be equal to 1 if i is prime

flag = checkPrimeNumber(i);

if (flag == 1) {

printf("%d ", i);

}

}

return 0;

}

// user-defined function to check prime number

int checkPrimeNumber(int n) {

int j, flag = 1;

for (j = 2; j <= n / 2; ++j) {

if (n % j == 0) {

flag = 0;

break;

}

}

return flag;

}

**Output**

Enter two positive integers: 12

30

Prime numbers between 12 and 30 are: 13 17 19 23 29

### Explanation

1. In this program, we print all the prime numbers between n1 and n2. If n1 is greater than n2, we [swap their values](https://www.programiz.com/c-programming/examples/swapping#example2):

if (n1 > n2) {

n1 = n1 + n2;

n2 = n1 - n2;

n1 = n1 - n2;

}

2. Then, we run a for loop from i = n1 + 1 to i = n2 - 1.

In each iteration of the loop, we check if i is a prime number using the checkPrimeNumber() function.

If i is prime, we print it.

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

flag = checkPrimeNumber(i);

if (flag == 1)

printf("%d ", i);

}

}

3. The checkPrimeNumber() function contains the code to [check whether a number is prime or not](https://www.programiz.com/c-programming/examples/prime-number).

int checkPrimeNumber(int n) {

int j, flag = 1;

for (j = 2; j <= n / 2; ++j) {

if (n % j == 0) {

flag = 0;

break;

}

}

return flag;

}

# C Program to Check Prime or Armstrong Number Using User-defined Function

In this example, you will learn to check whether an integer is a prime number or an Armstrong or both by creating two separate functions.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C for Loop](https://www.programiz.com/c-programming/c-for-loop)
* [C while and do...while Loop](https://www.programiz.com/c-programming/c-do-while-loops)
* [C break and continue](https://www.programiz.com/c-programming/c-break-continue-statement)
* [C Functions](https://www.programiz.com/c-programming/c-functions)
* [Types of User-defined Functions in C Programming](https://www.programiz.com/c-programming/types-user-defined-functions)

Visit these pages to learn to check whether a number is

* [a prime number or not](https://www.programiz.com/c-programming/examples/prime-number)
* [an Armstrong number or not](https://www.programiz.com/c-programming/examples/check-armstrong-number)

## Example: Check Prime and Armstrong

#include <math.h>

#include <stdio.h>

int checkPrimeNumber(int n);

int checkArmstrongNumber(int n);

int main() {

int n, flag;

printf("Enter a positive integer: ");

scanf("%d", &n);

// check prime number

flag = checkPrimeNumber(n);

if (flag == 1)

printf("%d is a prime number.\n", n);

else

printf("%d is not a prime number.\n", n);

// check Armstrong number

flag = checkArmstrongNumber(n);

if (flag == 1)

printf("%d is an Armstrong number.", n);

else

printf("%d is not an Armstrong number.", n);

return 0;

}

// function to check prime number

int checkPrimeNumber(int n) {

int i, flag = 1, squareRoot;

// computing the square root

squareRoot = sqrt(n);

for (i = 2; i <= squareRoot; ++i) {

// condition for non-prime number

if (n % i == 0) {

flag = 0;

break;

}

}

return flag;

}

// function to check Armstrong number

int checkArmstrongNumber(int num) {

int originalNum, remainder, n = 0, flag;

double result = 0.0;

// store the number of digits of num in n

for (originalNum = num; originalNum != 0; ++n) {

originalNum /= 10;

}

for (originalNum = num; originalNum != 0; originalNum /= 10) {

remainder = originalNum % 10;

// store the sum of the power of individual digits in result

result += pow(remainder, n);

}

// condition for Armstrong number

if (round(result) == num)

flag = 1;

else

flag = 0;

return flag;

}

**Output**

Enter a positive integer: 407

407 is not a prime number.

407 is an Armstrong number.

In this program, two user-defined functions checkPrimeNumber() and checkArmstrongNumber() are created.

The checkPrimeNumber() function returns:

* 1 if the number entered by the user is a prime number.
* 0 if the number entered by the user is not a prime number.

Similarly, checkArmstrongNumber() function returns:

* 1 if the number entered by the user is an Armstrong number.
* 0 if the number entered by the user is not an Armstrong number.

**Note:** In checkPrimeNumber() and checkArmstrongNumber() functions, the flag variables are the return value of the functions.

In main(), the flag variable stores the values returned by checkPrimeNumber() and checkArmstrongNumber().

# C Program to Check Whether a Number can be Expressed as Sum of Two Prime Numbers

In this example, you will learn to check if an integer entered by the user can be expressed as the sum of two prime numbers of all possible combinations.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C if...else Statement](https://www.programiz.com/c-programming/c-if-else-statement)
* [C for Loop](https://www.programiz.com/c-programming/c-for-loop)
* [C Functions](https://www.programiz.com/c-programming/c-functions)
* [C User-defined functions](https://www.programiz.com/c-programming/c-user-defined-functions)

This program takes a positive integer from the user and checks whether that number can be expressed as the sum of two prime numbers.

If the number can be expressed as the sum of two prime numbers, the output shows the combination of the prime numbers.

To perform this task, a user-defined function is created to [check prime number](https://www.programiz.com/c-programming/examples/prime-number).

## Integer as a Sum of Two Prime Numbers

#include <stdio.h>

int checkPrime(int n);

int main() {

int n, i, flag = 0;

printf("Enter a positive integer: ");

scanf("%d", &n);

for (i = 2; i <= n / 2; ++i) {

// condition for i to be a prime number

if (checkPrime(i) == 1) {

// condition for n-i to be a prime number

if (checkPrime(n - i) == 1) {

printf("%d = %d + %d\n", n, i, n - i);

flag = 1;

}

}

}

if (flag == 0)

printf("%d cannot be expressed as the sum of two prime numbers.", n);

return 0;

}

// function to check prime number

int checkPrime(int n) {

int i, isPrime = 1;

// 0 and 1 are not prime numbers

if (n == 0 || n == 1) {

isPrime = 0;

}

else {

for(i = 2; i <= n/2; ++i) {

if(n % i == 0) {

isPrime = 0;

break;

}

}

}

return isPrime;

}

**Output**

Enter a positive integer: 34

34 = 3 + 31

34 = 5 + 29

34 = 11 + 23

34 = 17 + 17

In this program, we use the checkPrime() function to check whether a number is prime or not.

In main(), we take a number from the user and store it in the variable n.

We also initialize the int variable flag to 0. We use this variable to determine whether the input number can be expressed as the sum of two prime numbers.

We then iterate a loop from i = 2 to i = n/2. In each iteration, we check whether i is a prime number or not.

If i is a prime, we check whether n - i is prime or not.

If n - i is also a prime, then we know that n can be expressed as the sum of two prime numbers i and n - i.

So, we print the result on the screen and change the value of flag to 1. Otherwise, flag remains 0.

This process continues until the loop ends.

If flag is still 0, then we know that n can't be expressed as the sum of two primes, and we print that message on the screen.

# C Program to Find the Sum of Natural Numbers using Recursion

In this example, you will learn to find the sum of natural numbers using a recursive function.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C User-defined functions](https://www.programiz.com/c-programming/c-user-defined-functions)
* [C Recursion](https://www.programiz.com/c-programming/c-recursion)

The positive numbers 1, 2, 3... are known as natural numbers. The program below takes a positive integer from the user and calculates the sum up to the given number.

Visit this page to [find the sum of natural numbers using a loop](https://www.programiz.com/c-programming/examples/sum-natural-numbers).

## Sum of Natural Numbers Using Recursion

#include <stdio.h>

int addNumbers(int n);

int main() {

int num;

printf("Enter a positive integer: ");

scanf("%d", &num);

printf("Sum = %d", addNumbers(num));

return 0;

}

int addNumbers(int n) {

if (n != 0)

return n + addNumbers(n - 1);

else

return n;

}

**Output**

Enter a positive integer: 20

Sum = 210

Suppose the user entered **20**.

Initially, addNumbers() is called from main() with **20** passed as an argument.

The number **20** is added to the result of addNumbers(19).

In the next function call from addNumbers() to addNumbers(), **19** is passed which is added to the result of addNumbers(18). This process continues until n is equal to **0**.

When n is equal to **0**, there is no recursive call. This returns the sum of integers ultimately to the main() function.

# C Program to Find Factorial of a Number Using Recursion

In this example, you will learn to find the factorial of a non-negative integer entered by the user using recursion.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Functions](https://www.programiz.com/c-programming/c-functions)
* [C User-defined functions](https://www.programiz.com/c-programming/c-user-defined-functions)
* [C Recursion](https://www.programiz.com/c-programming/c-recursion)

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.

You will learn to find the factorial of a number using recursion in this example. Visit this page to learn how you can find the [factorial of a number using a loop](https://www.programiz.com/c-programming/examples/factorial).

## Factorial of a Number Using Recursion

#include<stdio.h>

long int multiplyNumbers(int n);

int main() {

int n;

printf("Enter a positive integer: ");

scanf("%d",&n);

printf("Factorial of %d = %ld", n, multiplyNumbers(n));

return 0;

}

long int multiplyNumbers(int n) {

if (n>=1)

return n\*multiplyNumbers(n-1);

else

return 1;

}

**Output**

Enter a positive integer: 6

Factorial of 6 = 720

Suppose the user entered 6.

Initially, multiplyNumbers() is called from main() with 6 passed as an argument.

Then, 5 is passed to multiplyNumbers() from the same function (recursive call). In each recursive call, the value of argument n is decreased by 1.

When the value of n is less than 1, there is no recursive call and the factorial is returned ultimately to the main() function.

# C Program to Find G.C.D Using Recursion

In this example, you will learn to find the GCD (Greatest Common Divisor) of two positive integers entered by the user using recursion.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Functions](https://www.programiz.com/c-programming/c-functions)
* [C User-defined functions](https://www.programiz.com/c-programming/c-user-defined-functions)
* [C Recursion](https://www.programiz.com/c-programming/c-recursion)

This program takes two positive integers as input from the user and calculates GCD using recursion.

Visit this page to learn how you can [calculate the GCD using loops](https://www.programiz.com/c-programming/examples/hcf-gcd).

## GCD of Two Numbers using Recursion

#include <stdio.h>

int hcf(int n1, int n2);

int main() {

int n1, n2;

printf("Enter two positive integers: ");

scanf("%d %d", &n1, &n2);

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

return 0;

}

int hcf(int n1, int n2) {

if (n2 != 0)

return hcf(n2, n1 % n2);

else

return n1;

}

**Output**

Enter two positive integers: 366

60

G.C.D of 366 and 60 is 6.

In this program, recursive calls are made until the value of n2 is equal to 0.

# C Program to Convert Binary Number to Decimal and vice-versa

In this example, you will learn to convert binary numbers to decimal and vice-versa manually by creating a user-defined function.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Functions](https://www.programiz.com/c-programming/c-functions)
* [C User-defined functions](https://www.programiz.com/c-programming/c-user-defined-functions)

## Example 1: C Program to Convert Binary Number to Decimal

// convert binary to decimal

#include <stdio.h>

#include <math.h>

// function prototype

int convert(long long);

int main() {

long long n;

printf("Enter a binary number: ");

scanf("%lld", &n);

printf("%lld in binary = %d in decimal", n, convert(n));

return 0;

}

// function definition

int convert(long long n) {

int dec = 0, i = 0, rem;

while (n!=0) {

rem = n % 10;

n /= 10;

dec += rem \* pow(2, i);

++i;

}

return dec;

}

**Output**

Enter a binary number: 1101

1101 in binary = 13 in decimal

In the program, we have included the header file math.h to perform mathematical operations in the program.

We ask the user to enter a binary number and pass it to the convert() function to convert it decimal.

Suppose n = 1101. Let's see how the while loop in the convert() function works.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n != 0 | rem = n % 10 | n /= 10 | i | dec += rem \* pow(2, i) |
| 1101 != 0 | 1101 % 10 = 1 | 1101 / 10 = 110 | 0 | 0 + 1 \* pow (2, 0) = 1 |
| 110 != 0 | 110 % 10 = 0 | 110 / 10 = 11 | 1 | 1 + 0 \* pow (2, 1) = 1 |
| 10 != 0 | 11 % 10 = 1 | 11 /10 = 1 | 2 | 1 + 1 \* pow (2, 2) = 5 |
| 1 != 0 | 1 % 10 = 1 | 1 / 10 = 0 | 3 | 5 + 1 \* pow (2, 3) = 13 |

So, 1101 in binary is 13 in decimal.

Now, let's see how we can change the decimal number into a binary number.

## Example 2: C Program to convert decimal number to binary

// convert decimal to binary

#include <stdio.h>

#include <math.h>

long long convert(int);

int main() {

int n, bin;

printf("Enter a decimal number: ");

scanf("%d", &n);

bin = convert(n);

printf("%d in decimal = %lld in binary", n, bin);

return 0;

}

long long convert(int n) {

long long bin = 0;

int rem, i = 1;

while (n!=0) {

rem = n % 2;

n /= 2;

bin += rem \* i;

i \*= 10;

}

return bin;

}

**Output**

Enter a decimal number: 13

13 in decimal = 1101 in binary

Suppose n = 13. Let's see how the while loop in the convert() function works.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n != 0 | rem = n % 2 | n /= 2 | i | bin += rem \* i | i \* = 10 |
| 13 != 0 | 13 % 2 = 1 | 13 / 2 = 6 | 1 | 0 + 1 \* 1 = 1 | 1 \* 10 = 10 |
| 6 != 0 | 6 % 2 = 0 | 6 / 2 = 3 | 10 | 1 + 0 \* 10 = 1 | 10 \* 10 = 100 |
| 3 != 0 | 3 % 2 = 1 | 3 / 2 = 1 | 100 | 1 + 1 \* 100 = 101 | 100 \* 10 = 1000 |
| 1 != 0 | 1 % 2 = 1 | 1 / 2 = 0 | 1000 | 101 + 1 \* 1000 = 1101 | 1000 \* 10 = 10000 |
| 0 != 0 | - | - |  | - | Loop terminates |

Thus, 13 in decimal is 1101 in binary.

# C Program to Convert Octal Number to Decimal and vice-versa

In this example, you will learn to convert octal numbers to decimal and vice-versa manually by creating a user-defined function.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Functions](https://www.programiz.com/c-programming/c-functions)
* [C User-defined functions](https://www.programiz.com/c-programming/c-user-defined-functions)

## Example 1: Program to Convert Decimal to Octal

#include <stdio.h>

#include <math.h>

// function prototype

int convertDecimalToOctal(int decimalNumber);

int main() {

int decimalNumber;

printf("Enter a decimal number: ");

scanf("%d", &decimalNumber);

printf("%d in decimal = %d in octal", decimalNumber, convertDecimalToOctal(decimalNumber));

return 0;

}

// function to convert decimalNumber to octal

int convertDecimalToOctal(int decimalNumber) {

int octalNumber = 0, i = 1;

while (decimalNumber != 0) {

octalNumber += (decimalNumber % 8) \* i;

decimalNumber /= 8;

i \*= 10;

}

return octalNumber;

}

**Output**

Enter a decimal number: 78

78 in decimal = 116 in octal

## Example 2: Program to Convert Octal to Decimal

#include <stdio.h>

#include <math.h>

// function prototype

long long convertOctalToDecimal(int octalNumber);

int main() {

int octalNumber;

printf("Enter an octal number: ");

scanf("%d", &octalNumber);

printf("%d in octal = %lld in decimal", octalNumber, convertOctalToDecimal(octalNumber));

return 0;

}

// function to convert octalNumber to decimal

long long convertOctalToDecimal(int octalNumber) {

int decimalNumber = 0, i = 0;

while(octalNumber != 0) {

decimalNumber += (octalNumber%10) \* pow(8,i);

++i;

octalNumber/=10;

}

i = 1;

return decimalNumber;

}

**Output**

Enter an octal number: 116

116 in octal = 78 in decimal

**Note:** This program doesn't work if we input non-octal numbers i.e., numbers that contain 8 or 9. For example, 187, 96, 985, etc., are not octal numbers.

## Example 3: Check for Octal Number and Convert it to Decimal

#include <stdio.h>

#include <math.h>

// function prototypes

int checkOctal(int);

long long convertOctalToDecimal(int);

int main() {

int octalNumber;

int condition;

// repeat loop as long as user

// gives a non-octal number

do {

printf("Enter an octal number: ");

scanf("%d", &octalNumber);

// check if number is octal

condition = checkOctal(octalNumber);

if (!condition) {

printf("%d is not an octal number!\n", octalNumber);

}

} while (condition == 0);

printf("%d in octal = %lld in decimal", octalNumber, convertOctalToDecimal(octalNumber));

return 0;

}

// function to check octal number

int checkOctal(int octalNumber) {

int remainder;

// check each digit of input number

while(octalNumber != 0) {

remainder = octalNumber % 10;

// return 0 if a digit is 8 or 9

if (remainder >= 8) {

return 0;

}

octalNumber/= 10;

}

// return 1 if number is octal

return 1;

}

// function to convert octalNumber to decimal

long long convertOctalToDecimal(int octalNumber) {

int decimalNumber = 0, i = 0;

while(octalNumber != 0) {

decimalNumber += (octalNumber%10) \* pow(8,i);

++i;

octalNumber/=10;

}

i = 1;

return decimalNumber;

}

Enter an octal number: 96

96 is not an octal number!

Enter an octal number: 25

25 in octal = 21 in decimal

# C Program to Convert Binary Number to Octal and vice-versa

In this example, you will learn to convert binary numbers to octal and vice versa manually by creating a user-defined function.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Functions](https://www.programiz.com/c-programming/c-functions)
* [C User-defined functions](https://www.programiz.com/c-programming/c-user-defined-functions)

## Program to Convert Binary to Octal

In this program, we will first convert a binary number to decimal. Then, the decimal number is converted to octal.

#include <math.h>

#include <stdio.h>

int convert(long long bin);

int main() {

long long bin;

printf("Enter a binary number: ");

scanf("%lld", &bin);

printf("%lld in binary = %d in octal", bin, convert(bin));

return 0;

}

int convert(long long bin) {

int oct = 0, dec = 0, i = 0;

// converting binary to decimal

while (bin != 0) {

dec += (bin % 10) \* pow(2, i);

++i;

bin /= 10;

}

i = 1;

// converting to decimal to octal

while (dec != 0) {

oct += (dec % 8) \* i;

dec /= 8;

i \*= 10;

}

return oct;

}

**Output**

Enter a binary number: 101001

101001 in binary = 51 in octal

## Program to Convert Octal to Binary

In this program, an octal number is converted to decimal at first. Then, the decimal number is converted to binary number.

#include <math.h>

#include <stdio.h>

long long convert(int oct);

int main() {

int oct;

printf("Enter an octal number: ");

scanf("%d", &oct);

printf("%d in octal = %lld in binary", oct, convert(oct));

return 0;

}

long long convert(int oct) {

int dec = 0, i = 0;

long long bin = 0;

// converting octal to decimal

while (oct != 0) {

dec += (oct % 10) \* pow(8, i);

++i;

oct /= 10;

}

i = 1;

// converting decimal to binary

while (dec != 0) {

bin += (dec % 2) \* i;

dec /= 2;

i \*= 10;

}

return bin;

}

**Output**

Enter an octal number: 67

67 in octal = 110111 in binary

# C Program to Reverse a Sentence Using Recursion

In this example, you will learn to take a sentence from the user and reverse it using recursion.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Functions](https://www.programiz.com/c-programming/c-functions)
* [C User-defined functions](https://www.programiz.com/c-programming/c-user-defined-functions)
* [C Recursion](https://www.programiz.com/c-programming/c-recursion)

## Reverse a sentence using recursion

#include <stdio.h>

void reverseSentence();

int main() {

printf("Enter a sentence: ");

reverseSentence();

return 0;

}

void reverseSentence() {

char c;

scanf("%c", &c);

if (c != '\n') {

reverseSentence();

printf("%c", c);

}

}

**Output**

Enter a sentence: margorp emosewa

awesome program

This program first prints Enter a sentence: . Then, the reverseSentence() function is called.

This function stores the first letter entered by the user in c. If the variable is any character other than \n (newline), reverseSentence() is called again.

This process goes on until the user hits enter.

When the user hits enter, the reverseSentence() function starts printing characters from last.

# C program to calculate the power using recursion

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

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Functions](https://www.programiz.com/c-programming/c-functions)
* [C User-defined functions](https://www.programiz.com/c-programming/c-user-defined-functions)
* [C Recursion](https://www.programiz.com/c-programming/c-recursion)

## Program to calculate power using recursion.

#include <stdio.h>

int power(int n1, int n2);

int main() {

int base, a, result;

printf("Enter base number: ");

scanf("%d", &base);

printf("Enter power number(positive integer): ");

scanf("%d", &a);

result = power(base, a);

printf("%d^%d = %d", base, a, result);

return 0;

}

int power(int base, int a) {

if (a != 0)

return (base \* power(base, a - 1));

else

return 1;

}

**Output**

Enter base number: 3

Enter power number(positive integer): 4

3^4 = 81

You can also [compute the power of a number using a loop](https://www.programiz.com/c-programming/examples/power-number).

If you need to calculate the power of a number raised to a decimal value, you can use the [pow() library function](https://www.programiz.com/c-programming/library-function/math.h/pow).

# C Program to Calculate Average Using Arrays

In this example, you will learn to calculate the average of n number of elements entered by the user using arrays.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C while and do...while Loop](https://www.programiz.com/c-programming/c-do-while-loops)
* [C for Loop](https://www.programiz.com/c-programming/c-for-loop)
* [C Arrays](https://www.programiz.com/c-programming/c-arrays)

## Store Numbers and Calculate Average Using Arrays

#include <stdio.h>

int main() {

int n, i;

float num[100], sum = 0.0, avg;

printf("Enter the numbers of elements: ");

scanf("%d", &n);

while (n > 100 || n < 1) {

printf("Error! number should in range of (1 to 100).\n");

printf("Enter the number again: ");

scanf("%d", &n);

}

for (i = 0; i < n; ++i) {

printf("%d. Enter number: ", i + 1);

scanf("%f", &num[i]);

sum += num[i];

}

avg = sum / n;

printf("Average = %.2f", avg);

return 0;

}

**Output**

Enter the numbers of elements: 6

1. Enter number: 45.3

2. Enter number: 67.5

3. Enter number: -45.6

4. Enter number: 20.34

5. Enter number: 33

6. Enter number: 45.6

Average = 27.69

Here, the user is first asked to enter the number of elements. This number is assigned to n.

If the user entered integer is greater less than 1 or greater than 100, the user is asked to enter the number again. This is done using a while loop.

Then, we have iterated a for loop from i = 0 to i . In each iteration of the loop, the user is asked to enter numbers to calculate the average. These numbers are stored in the num[] array.

scanf("%f", &num[i]);

And, the sum of each entered element is computed.

sum += num[i];

Once the for loop is completed, the average is calculated and printed on the screen.

# C Program to Find Largest Element in an Array

In this example, you will learn to display the largest element entered by the user in an array.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C for Loop](https://www.programiz.com/c-programming/c-for-loop)
* [C Arrays](https://www.programiz.com/c-programming/c-arrays)

## Example: Largest Element in an array

#include <stdio.h>

int main() {

int n;

double arr[100];

printf("Enter the number of elements (1 to 100): ");

scanf("%d", &n);

for (int i = 0; i < n; ++i) {

printf("Enter number%d: ", i + 1);

scanf("%lf", &arr[i]);

}

// storing the largest number to arr[0]

for (int i = 1; i < n; ++i) {

if (arr[0] < arr[i]) {

arr[0] = arr[i];

}

}

printf("Largest element = %.2lf", arr[0]);

return 0;

}

**Output**

Enter the number of elements (1 to 100): 5

Enter number1: 34.5

Enter number2: 2.4

Enter number3: -35.5

Enter number4: 38.7

Enter number5: 24.5

Largest element = 38.70

This program takes n number of elements from the user and stores it in the arr array.

To find the largest element,

* the first two elements of array are checked and the largest of these two elements are placed in arr[0]
* the first and third elements are checked and largest of these two elements is placed in arr[0].
* this process continues until the first and last elements are checked
* the largest number will be stored in the arr[0] position

// storing the largest number at arr[0]

for (int i = 1; i < n; ++i) {

if (arr[0] < arr[i]) {

arr[0] = arr[i];

}

}

# C Program to Calculate Standard Deviation

In this example, you will learn to calculate the standard deviation of 10 numbers stored in an array.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Arrays](https://www.programiz.com/c-programming/c-arrays)
* [Pass arrays to a function in C](https://www.programiz.com/c-programming/c-arrays-functions)

This program calculates the standard deviation of an individual series using arrays. Visit this page to learn about [Standard Deviation](https://www.mathsisfun.com/data/standard-deviation.html).

To calculate the standard deviation, we have created a function named calculateSD().

## Example: Population Standard Deviation

// SD of a population

#include <math.h>

#include <stdio.h>

float calculateSD(float data[]);

int main() {

int i;

float data[10];

printf("Enter 10 elements: ");

for (i = 0; i < 10; ++i)

scanf("%f", &data[i]);

printf("\nStandard Deviation = %.6f", calculateSD(data));

return 0;

}

float calculateSD(float data[]) {

float sum = 0.0, mean, SD = 0.0;

int i;

for (i = 0; i < 10; ++i) {

sum += data[i];

}

mean = sum / 10;

for (i = 0; i < 10; ++i) {

SD += pow(data[i] - mean, 2);

}

return sqrt(SD / 10);

}

**Output**

Enter 10 elements: 1

2

3

4

5

6

7

8

9

10

Standard Deviation = 2.872281

Here, the array containing 10 elements is passed to the calculateSD() function. The function calculates the standard deviation using **mean** and returns it.

**Note:** The program calculates the standard deviation of a **population**. If you need to find the standard deviation of a **sample**, the formula is slightly different.

# C Program to Add Two Matrices Using Multi-dimensional Arrays

In this example, you will learn to add two matrices in C programming using two-dimensional arrays.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Arrays](https://www.programiz.com/c-programming/c-arrays)
* [C Multidimensional Arrays](https://www.programiz.com/c-programming/c-multi-dimensional-arrays)

## Program to Add Two Matrices

#include <stdio.h>

int main() {

int r, c, a[100][100], b[100][100], sum[100][100], i, j;

printf("Enter the number of rows (between 1 and 100): ");

scanf("%d", &r);

printf("Enter the number of columns (between 1 and 100): ");

scanf("%d", &c);

printf("\nEnter elements of 1st matrix:\n");

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

for (j = 0; j < c; ++j) {

printf("Enter element a%d%d: ", i + 1, j + 1);

scanf("%d", &a[i][j]);

}

printf("Enter elements of 2nd matrix:\n");

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

for (j = 0; j < c; ++j) {

printf("Enter element b%d%d: ", i + 1, j + 1);

scanf("%d", &b[i][j]);

}

// adding two matrices

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

for (j = 0; j < c; ++j) {

sum[i][j] = a[i][j] + b[i][j];

}

// printing the result

printf("\nSum of two matrices: \n");

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

for (j = 0; j < c; ++j) {

printf("%d ", sum[i][j]);

if (j == c - 1) {

printf("\n\n");

}

}

return 0;

}

**Output**

Enter the number of rows (between 1 and 100): 2

Enter the number of columns (between 1 and 100): 3

Enter elements of 1st matrix:

Enter element a11: 2

Enter element a12: 3

Enter element a13: 4

Enter element a21: 5

Enter element a22: 2

Enter element a23: 3

Enter elements of 2nd matrix:

Enter element b11: -4

Enter element b12: 5

Enter element b13: 3

Enter element b21: 5

Enter element b22: 6

Enter element b23: 3

Sum of two matrices:

-2 8 7

10 8 6

In this program, the user is asked to enter the number of rows r and columns c. Then, the user is asked to enter the elements of the two matrices (of order r**x**c).

We then added corresponding elements of two matrices and saved it in another matrix (two-dimensional array). Finally, the result is printed on the screen.

# C Program to Multiply Two Matrices Using Multi-dimensional Arrays

In this example, you will learn to multiply two matrices and display it using user-defined functions.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Arrays](https://www.programiz.com/c-programming/c-arrays)
* [C Multidimensional Arrays](https://www.programiz.com/c-programming/c-multi-dimensional-arrays)

This program asks the user to enter the size (rows and columns) of two matrices.

To multiply two matrices, **the number of columns of the first matrix should be equal to the number of rows of the second matrix**.

The program below asks for the number of rows and columns of two matrices until the above condition is satisfied.

Then, the multiplication of two matrices is performed, and the result is displayed on the screen.

To perform this, we have created three functions:

* getMatrixElements() - to take matrix elements input from the user.
* multiplyMatrices() - to multiply two matrices.
* display() - to display the resultant matrix after multiplication.

## Multiply Matrices by Passing it to a Function.

#include <stdio.h>

// function to get matrix elements entered by the user

void getMatrixElements(int matrix[][10], int row, int column) {

printf("\nEnter elements: \n");

for (int i = 0; i < row; ++i) {

for (int j = 0; j < column; ++j) {

printf("Enter a%d%d: ", i + 1, j + 1);

scanf("%d", &matrix[i][j]);

}

}

}

// function to multiply two matrices

void multiplyMatrices(int first[][10],

int second[][10],

int result[][10],

int r1, int c1, int r2, int c2) {

// Initializing elements of matrix mult to 0.

for (int i = 0; i < r1; ++i) {

for (int j = 0; j < c2; ++j) {

result[i][j] = 0;

}

}

// Multiplying first and second matrices and storing it in result

for (int i = 0; i < r1; ++i) {

for (int j = 0; j < c2; ++j) {

for (int k = 0; k < c1; ++k) {

result[i][j] += first[i][k] \* second[k][j];

}

}

}

}

// function to display the matrix

void display(int result[][10], int row, int column) {

printf("\nOutput Matrix:\n");

for (int i = 0; i < row; ++i) {

for (int j = 0; j < column; ++j) {

printf("%d ", result[i][j]);

if (j == column - 1)

printf("\n");

}

}

}

int main() {

int first[10][10], second[10][10], result[10][10], r1, c1, r2, c2;

printf("Enter rows and column for the first matrix: ");

scanf("%d %d", &r1, &c1);

printf("Enter rows and column for the second matrix: ");

scanf("%d %d", &r2, &c2);

// Taking input until

// 1st matrix columns is not equal to 2nd matrix row

while (c1 != r2) {

printf("Error! Enter rows and columns again.\n");

printf("Enter rows and columns for the first matrix: ");

scanf("%d%d", &r1, &c1);

printf("Enter rows and columns for the second matrix: ");

scanf("%d%d", &r2, &c2);

}

// get elements of the first matrix

getMatrixElements(first, r1, c1);

// get elements of the second matrix

getMatrixElements(second, r2, c2);

// multiply two matrices.

multiplyMatrices(first, second, result, r1, c1, r2, c2);

// display the result

display(result, r1, c2);

return 0;

}

**Output**

Enter rows and column for the first matrix: 2

3

Enter rows and column for the second matrix: 3

2

Enter elements:

Enter a11: 2

Enter a12: -3

Enter a13: 4

Enter a21: 53

Enter a22: 3

Enter a23: 5

Enter elements:

Enter a11: 3

Enter a12: 3

Enter a21: 5

Enter a22: 0

Enter a31: -3

Enter a32: 4

Output Matrix:

-21 22

159 179

# C Program to Find Transpose of a Matrix

In this example, you will learn to find the transpose of a matrix in C programming.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Arrays](https://www.programiz.com/c-programming/c-arrays)
* [C Multidimensional Arrays](https://www.programiz.com/c-programming/c-multi-dimensional-arrays)

The transpose of a matrix is a new matrix that is obtained by exchanging the rows and columns.

In this program, the user is asked to enter the number of rows r and columns c. Their values should be less than 10 in this program.

Then, the user is asked to enter the elements of the matrix (of order r\*c).

The program below then computes the transpose of the matrix and prints it on the screen.

## Program to Find the Transpose of a Matrix

#include <stdio.h>

int main() {

int a[10][10], transpose[10][10], r, c;

printf("Enter rows and columns: ");

scanf("%d %d", &r, &c);

// asssigning elements to the matrix

printf("\nEnter matrix elements:\n");

for (int i = 0; i < r; ++i)

for (int j = 0; j < c; ++j) {

printf("Enter element a%d%d: ", i + 1, j + 1);

scanf("%d", &a[i][j]);

}

// printing the matrix a[][]

printf("\nEntered matrix: \n");

for (int i = 0; i < r; ++i)

for (int j = 0; j < c; ++j) {

printf("%d ", a[i][j]);

if (j == c - 1)

printf("\n");

}

// computing the transpose

for (int i = 0; i < r; ++i)

for (int j = 0; j < c; ++j) {

transpose[j][i] = a[i][j];

}

// printing the transpose

printf("\nTranspose of the matrix:\n");

for (int i = 0; i < c; ++i)

for (int j = 0; j < r; ++j) {

printf("%d ", transpose[i][j]);

if (j == r - 1)

printf("\n");

}

return 0;

}

**Output**

Enter rows and columns: 2

3

Enter matrix elements:

Enter element a11: 1

Enter element a12: 4

Enter element a13: 0

Enter element a21: -5

Enter element a22: 2

Enter element a23: 7

Entered matrix:

1 4 0

-5 2 7

Transpose of the matrix:

1 -5

4 2

0 7

# C Program to Multiply two Matrices by Passing Matrix to a Function

In this example, you'll learn to multiply two matrices and display it using user defined function.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Arrays](https://www.programiz.com/c-programming/c-arrays)
* [C Multidimensional Arrays](https://www.programiz.com/c-programming/c-multi-dimensional-arrays)
* [Pass arrays to a function in C](https://www.programiz.com/c-programming/c-arrays-functions)

This program asks the user to enter the size of the matrix (rows and column).

Then, it asks the user to enter the elements of those matrices and finally adds and displays the result.

To perform this task three functions are made:

1. To takes matrix elements from user enterData()
2. To multiply two matrix multiplyMatrices()
3. To display the resultant matrix after multiplication display()

### Example: Multiply Matrices by Passing it to a Function

#include <stdio.h>

void enterData(int firstMatrix[][10], int secondMatrix[][10], int rowFirst, int columnFirst, int rowSecond, int columnSecond);

void multiplyMatrices(int firstMatrix[][10], int secondMatrix[][10], int multResult[][10], int rowFirst, int columnFirst, int rowSecond, int columnSecond);

void display(int mult[][10], int rowFirst, int columnSecond);

int main()

{

int firstMatrix[10][10], secondMatrix[10][10], mult[10][10], rowFirst, columnFirst, rowSecond, columnSecond, i, j, k;

printf("Enter rows and column for first matrix: ");

scanf("%d %d", &rowFirst, &columnFirst);

printf("Enter rows and column for second matrix: ");

scanf("%d %d", &rowSecond, &columnSecond);

// If colum of first matrix in not equal to row of second matrix, asking user to enter the size of matrix again.

while (columnFirst != rowSecond)

{

printf("Error! column of first matrix not equal to row of second.\n");

printf("Enter rows and column for first matrix: ");

scanf("%d%d", &rowFirst, &columnFirst);

printf("Enter rows and column for second matrix: ");

scanf("%d%d", &rowSecond, &columnSecond);

}

// Function to take matrices data

enterData(firstMatrix, secondMatrix, rowFirst, columnFirst, rowSecond, columnSecond);

// Function to multiply two matrices.

multiplyMatrices(firstMatrix, secondMatrix, mult, rowFirst, columnFirst, rowSecond, columnSecond);

// Function to display resultant matrix after multiplication.

display(mult, rowFirst, columnSecond);

return 0;

}

void enterData(int firstMatrix[][10], int secondMatrix[][10], int rowFirst, int columnFirst, int rowSecond, int columnSecond)

{

int i, j;

printf("\nEnter elements of matrix 1:\n");

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

{

for(j = 0; j < columnFirst; ++j)

{

printf("Enter elements a%d%d: ", i + 1, j + 1);

scanf("%d", &firstMatrix[i][j]);

}

}

printf("\nEnter elements of matrix 2:\n");

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

{

for(j = 0; j < columnSecond; ++j)

{

printf("Enter elements b%d%d: ", i + 1, j + 1);

scanf("%d", &secondMatrix[i][j]);

}

}

}

void multiplyMatrices(int firstMatrix[][10], int secondMatrix[][10], int mult[][10], int rowFirst, int columnFirst, int rowSecond, int columnSecond)

{

int i, j, k;

// Initializing elements of matrix mult to 0.

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

{

for(j = 0; j < columnSecond; ++j)

{

mult[i][j] = 0;

}

}

// Multiplying matrix firstMatrix and secondMatrix and storing in array mult.

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

{

for(j = 0; j < columnSecond; ++j)

{

for(k=0; k<columnFirst; ++k)

{

mult[i][j] += firstMatrix[i][k] \* secondMatrix[k][j];

}

}

}

}

void display(int mult[][10], int rowFirst, int columnSecond)

{

int i, j;

printf("\nOutput Matrix:\n");

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

{

for(j = 0; j < columnSecond; ++j)

{

printf("%d ", mult[i][j]);

if(j == columnSecond - 1)

printf("\n\n");

}

}

}

**Output**

Enter rows and column for first matrix: 3

2

Enter rows and column for second matrix: 3

2

Error! column of first matrix not equal to row of second.

Enter rows and column for first matrix: 2

3

Enter rows and column for second matrix: 3

2

Enter elements of matrix 1:

Enter elements a11: 3

Enter elements a12: -2

Enter elements a13: 5

Enter elements a21: 3

Enter elements a22: 0

Enter elements a23: 4

Enter elements of matrix 2:

Enter elements b11: 2

Enter elements b12: 3

Enter elements b21: -9

Enter elements b22: 0

Enter elements b31: 0

Enter elements b32: 4

Output Matrix:

24 29

6 25

# C Program to Access Array Elements Using Pointer

In this example, you will learn to access elements of an array using a pointer.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C for Loop](https://www.programiz.com/c-programming/c-for-loop)
* [C Arrays](https://www.programiz.com/c-programming/c-arrays)
* [C Pointers](https://www.programiz.com/c-programming/c-pointers)
* [Relationship Between Arrays and Pointers](https://www.programiz.com/c-programming/c-pointers-arrays)

## Access Array Elements Using Pointers

#include <stdio.h>

int main() {

int data[5];

printf("Enter elements: ");

for (int i = 0; i < 5; ++i)

scanf("%d", data + i);

printf("You entered: \n");

for (int i = 0; i < 5; ++i)

printf("%d\n", \*(data + i));

return 0;

}

**Output**

Enter elements: 1

2

3

5

4

You entered:

1

2

3

5

4

In this program, the elements are stored in the integer array data[].

Then, the elements of the array are accessed using the pointer notation. By the way,

* data[0] is equivalent to \*data and &data[0] is equivalent to data
* data[1] is equivalent to \*(data + 1) and &data[1] is equivalent to data + 1
* data[2] is equivalent to \*(data + 2) and &data[2] is equivalent to data + 2
* ...
* data[i] is equivalent to \*(data + i) and &data[i] is equivalent to data + i

# C Program Swap Numbers in Cyclic Order Using Call by Reference

In this example, the three numbers entered by the user are swapped in cyclic order using call by reference.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Pointers](https://www.programiz.com/c-programming/c-pointers)
* [C Pass Addresses and Pointers](https://www.programiz.com/c-programming/c-pointer-functions)

## Program to Swap Elements Using Call by Reference

#include <stdio.h>

void cyclicSwap(int \*a, int \*b, int \*c);

int main() {

int a, b, c;

printf("Enter a, b and c respectively: ");

scanf("%d %d %d", &a, &b, &c);

printf("Value before swapping:\n");

printf("a = %d \nb = %d \nc = %d\n", a, b, c);

cyclicSwap(&a, &b, &c);

printf("Value after swapping:\n");

printf("a = %d \nb = %d \nc = %d", a, b, c);

return 0;

}

void cyclicSwap(int \*n1, int \*n2, int \*n3) {

int temp;

// swapping in cyclic order

temp = \*n2;

\*n2 = \*n1;

\*n1 = \*n3;

\*n3 = temp;

}

**Output**

Enter a, b and c respectively: 1

2

3

Value before swapping:

a = 1

b = 2

c = 3

Value after swapping:

a = 3

b = 1

c = 2

Here, the three numbers entered by the user are stored in variables a, b and c respectively. The addresses of these numbers are passed to the cyclicSwap() function.

cyclicSwap(&a, &b, &c);

In the function definition of cyclicSwap(), we have assigned these addresses to pointers.

cyclicSwap(int \*n1, int \*n2, int \*n3) {

...

}

When n1, n2 and n3 inside cyclicSwap() are changed, the values of a, b and c inside main() are also changed.

**Note:** The cyclicSwap() function is not returning anything.

# C Program to Find Largest Number Using Dynamic Memory Allocation

In this example, you will learn to find the largest number entered by the user in a dynamically allocated memory.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Pointers](https://www.programiz.com/c-programming/c-pointers)
* [C Dynamic Memory Allocation](https://www.programiz.com/c-programming/c-dynamic-memory-allocation)
* [C for Loop](https://www.programiz.com/c-programming/c-for-loop)

## Example: Find the Largest Element

#include <stdio.h>

#include <stdlib.h>

int main() {

int n;

double \*data;

printf("Enter the total number of elements: ");

scanf("%d", &n);

// Allocating memory for n elements

data = (double \*)calloc(n, sizeof(double));

if (data == NULL) {

printf("Error!!! memory not allocated.");

exit(0);

}

// Storing numbers entered by the user.

for (int i = 0; i < n; ++i) {

printf("Enter number%d: ", i + 1);

scanf("%lf", data + i);

}

// Finding the largest number

for (int i = 1; i < n; ++i) {

if (\*data < \*(data + i)) {

\*data = \*(data + i);

}

}

printf("Largest number = %.2lf", \*data);

free(data);

return 0;

}

**Output**

Enter the total number of elements: 5

Enter number1: 3.4

Enter number2: 2.4

Enter number3: -5

Enter number4: 24.2

Enter number5: 6.7

Largest number = 24.20

### Explanation

In the program, we have asked the user to enter the total number of elements which is stored in the variable n. Then, we have allocated memory for n number of double values.

// Allocating memory for n double values

data = (double \*)calloc(n, sizeof(double));

Then, we used a for loop to take n number of data from the user.

// Storing elements

for (int i = 0; i < n; ++i) {

printf("Enter Number%d: ", i + 1);

scanf("%lf", data + i);

}

Finally, we used another for loop to compute the largest number.

// Computing the largest number

for (int i = 1; i < n; ++i) {

if (\*data < \*(data + i))

\*data = \*(data + i);

}

}

**Note:** Instead of calloc(), it's also possible to solve this problem using the [malloc()](https://www.programiz.com/c-programming/c-dynamic-memory-allocation) function.

# C Program to Find the Frequency of Characters in a String

In this example, you will learn to find the frequency of a character in a string.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Arrays](https://www.programiz.com/c-programming/c-arrays)
* [C Programming Strings](https://www.programiz.com/c-programming/c-strings)

## Find the Frequency of a Character

#include <stdio.h>

int main() {

char str[1000], ch;

int count = 0;

printf("Enter a string: ");

fgets(str, sizeof(str), stdin);

printf("Enter a character to find its frequency: ");

scanf("%c", &ch);

for (int i = 0; str[i] != '\0'; ++i) {

if (ch == str[i])

++count;

}

printf("Frequency of %c = %d", ch, count);

return 0;

}

**Output**

Enter a string: This website is awesome.

Enter a character to find its frequency: e

Frequency of e = 4

In this program, the string entered by the user is stored in str.

Then, the user is asked to enter the character whose frequency is to be found. This is stored in variable ch.

Then, a for loop is used to iterate over characters of the string. In each iteration, if the character in the string is equal to the ch, count is increased by 1.

Finally, the frequency stored in the count variable is printed.

**Note:** This program is case-sensitive i.e. it treats uppercase and lowercase versions of the same alphabet as different characters.

# C Program to Count the Number of Vowels, Consonants and so on

In this example, the number of vowels, consonants, digits, and white-spaces in a string entered by the user is counted.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Arrays](https://www.programiz.com/c-programming/c-arrays)
* [C Programming Strings](https://www.programiz.com/c-programming/c-strings)

## Program to count vowels, consonants, etc.

#include <stdio.h>

int main() {

char line[150];

int vowels, consonant, digit, space;

// initialize all variables to 0

vowels = consonant = digit = space = 0;

// get full line of string input

printf("Enter a line of string: ");

fgets(line, sizeof(line), stdin);

// loop through each character of the string

for (int i = 0; line[i] != '\0'; ++i) {

// convert character to lowercase

line[i] = tolower(line[i]);

// check if the character is a vowel

if (line[i] == 'a' || line[i] == 'e' || line[i] == 'i' ||

line[i] == 'o' || line[i] == 'u') {

// increment value of vowels by 1

++vowels;

}

// if it is not a vowel and if it is an alphabet, it is a consonant

else if ((line[i] >= 'a' && line[i] <= 'z')) {

++consonant;

}

// check if the character is a digit

else if (line[i] >= '0' && line[i] <= '9') {

++digit;

}

// check if the character is an empty space

else if (line[i] == ' ') {

++space;

}

}

printf("Vowels: %d", vowels);

printf("\nConsonants: %d", consonant);

printf("\nDigits: %d", digit);

printf("\nWhite spaces: %d", space);

return 0;

}

**Output**

Enter a line of string: C++ 20 is the latest version of C++ yet.

Vowels: 9

Consonants: 16

Digits: 2

White spaces: 8

Here, the string entered by the user is stored in the line variable.

Initially, the variables vowel, consonant, digit, and space are initialized to **0**.

Then, a for loop is used to iterate over the characters of the string. In each iteration, we:

* convert the character to lowercase using the tolower() function
* check whether the character is a vowel, a consonant, a digit, or an empty space. Suppose the character is a consonant. Then, the consonant variable is increased by **1**.

When the loop ends, the number of vowels, consonants, digits, and white spaces are stored in variables vowel, consonant, digit, and space respectively.

**Note:** We have used the [tolower() function](https://www.programiz.com/c-programming/library-function/ctype.h/tolower) to simplify our program. To use this function, we need to import the [ctype.h](https://www.programiz.com/c-programming/library-function/ctype.h) header file.

# C Program to Remove all Characters in a String Except Alphabets

In this example, you will learn to remove all the characters from a string entered by the user except the alphabets.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Arrays](https://www.programiz.com/c-programming/c-arrays)
* [C Programming Strings](https://www.programiz.com/c-programming/c-strings)
* [C for Loop](https://www.programiz.com/c-programming/c-for-loop)
* [C while and do...while Loop](https://www.programiz.com/c-programming/c-do-while-loops)

## Remove Characters in String Except Alphabets

#include <stdio.h>

int main() {

char line[150];

printf("Enter a string: ");

fgets(line, sizeof(line), stdin); // take input

for (int i = 0, j; line[i] != '\0'; ++i) {

// enter the loop if the character is not an alphabet

// and not the null character

while (!(line[i] >= 'a' && line[i] <= 'z') && !(line[i] >= 'A' && line[i] <= 'Z') && !(line[i] == '\0')) {

for (j = i; line[j] != '\0'; ++j) {

// if jth element of line is not an alphabet,

// assign the value of (j+1)th element to the jth element

line[j] = line[j + 1];

}

line[j] = '\0';

}

}

printf("Output String: ");

puts(line);

return 0;

}

**Output**

Enter a string: p2'r-o@gram84iz./

Output String: programiz

This program takes a string input from the user and stores in the line variable. Then, a for loop is used to iterate over characters of the string.

If the character in a string is not an alphabet, it is removed from the string and the position of the remaining characters are shifted to the left by 1 position.

# C Program to Find the Length of a String

In this example, you will learn to find the length of a string manually without using the strlen() function.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Programming Strings](https://www.programiz.com/c-programming/c-strings)
* [String Manipulations In C Programming Using Library Functions](https://www.programiz.com/c-programming/string-handling-functions)
* [C for Loop](https://www.programiz.com/c-programming/c-for-loop)

As you know, the best way to find the length of a string is by using the strlen() function. However, in this example, we will find the length of a string manually.

## Calculate Length of String without Using strlen() Function

#include <stdio.h>

int main() {

char s[] = "Programming is fun";

int i;

for (i = 0; s[i] != '\0'; ++i);

printf("Length of the string: %d", i);

return 0;

}

**Output**

Length of the string: 18

Here, using a for loop, we have iterated over characters of the string from i = 0 to until '\0' (null character) is encountered. In each iteration, the value of i is increased by 1.

When the loop ends, the length of the string will be stored in the i variable.

**Note:** Here, the array s[] has 19 elements. The last element s[18] is the null character '\0'. But our loop does not count this character as it terminates upon encountering it.

# C Program to Concatenate Two Strings

In this example, you will learn to concatenate two strings manually without using the strcat() function.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Arrays](https://www.programiz.com/c-programming/c-arrays)
* [C Programming Strings](https://www.programiz.com/c-programming/c-strings)
* [C for Loop](https://www.programiz.com/c-programming/c-for-loop)

As you know, the best way to concatenate two strings in C programming is by using the [strcat()](https://www.programiz.com/c-programming/library-function/string.h/strcat" \o "C strcat()) function. However, in this example, we will concatenate two strings manually.

## Concatenate Two Strings Without Using strcat()

#include <stdio.h>

int main() {

char s1[100] = "programming ", s2[] = "is awesome";

int length, j;

// store length of s1 in the length variable

length = 0;

while (s1[length] != '\0') {

++length;

}

// concatenate s2 to s1

for (j = 0; s2[j] != '\0'; ++j, ++length) {

s1[length] = s2[j];

}

// terminating the s1 string

s1[length] = '\0';

printf("After concatenation: ");

puts(s1);

return 0;

}

**Output**

After concatenation: programming is awesome

Here, two strings s1 and s2 and concatenated and the result is stored in s1.

It's important to note that the length of s1 should be sufficient to hold the string after concatenation. If not, you may get unexpected output.

# C Program to Copy String Without Using strcpy()

In this example, you will learn to copy strings without using the strcpy() function.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Arrays](https://www.programiz.com/c-programming/c-arrays)
* [C Programming Strings](https://www.programiz.com/c-programming/c-strings)
* [C for Loop](https://www.programiz.com/c-programming/c-for-loop)

As you know, the best way to copy a string is by using the strcpy() function. However, in this example, we will copy a string manually without using the strcpy() function.

## Copy String Without Using strcpy()

#include <stdio.h>

int main() {

char s1[100], s2[100], i;

printf("Enter string s1: ");

fgets(s1, sizeof(s1), stdin);

for (i = 0; s1[i] != '\0'; ++i) {

s2[i] = s1[i];

}

s2[i] = '\0';

printf("String s2: %s", s2);

return 0;

}

**Output**

Enter string s1: Hey fellow programmer.

String s2: Hey fellow programmer.

The above program copies the content of string s1 to string s2 manually.

# C Program to Sort Elements in Lexicographical Order (Dictionary Order)

In this example, you will learn to sort 5 strings entered by the user in the lexicographical order (dictionary order).

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Multidimensional Arrays](https://www.programiz.com/c-programming/c-multi-dimensional-arrays)
* [C Programming Strings](https://www.programiz.com/c-programming/c-strings)
* [String Manipulations In C Programming Using Library Functions](https://www.programiz.com/c-programming/string-handling-functions)

## Sort strings in the dictionary order

#include <stdio.h>

#include <string.h>

int main() {

char str[5][50], temp[50];

printf("Enter 5 words: ");

// Getting strings input

for (int i = 0; i < 5; ++i) {

fgets(str[i], sizeof(str[i]), stdin);

}

// storing strings in the lexicographical order

for (int i = 0; i < 5; ++i) {

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

// swapping strings if they are not in the lexicographical order

if (strcmp(str[i], str[j]) > 0) {

strcpy(temp, str[i]);

strcpy(str[i], str[j]);

strcpy(str[j], temp);

}

}

}

printf("\nIn the lexicographical order: \n");

for (int i = 0; i < 5; ++i) {

fputs(str[i], stdout);

}

return 0;

}

**Output**

Enter 5 words: R programming

JavaScript

Java

C programming

C++ programming

In the lexicographical order:

C programming

C++ programming

Java

JavaScript

R programming

To solve this program, a two-dimensional string named str is created. The string can hold a maximum of 5 strings and each string can have a maximum of 50 characters (including the null character).

In the program, we have used two library functions:

* [strcmp()](https://www.programiz.com/c-programming/library-function/string.h/strcmp) - to compare strings
* [strcpy()](https://www.programiz.com/c-programming/library-function/string.h/strcpy) - to copy strings

These functions are used to compare strings and sort them in the correct order.

# C Program to Store Information of a Student Using Structure

In this example, you will learn to store the information of a student in a structure and display them on the screen.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C struct](https://www.programiz.com/c-programming/c-structures)

## Store Information and Display it Using Structure

#include <stdio.h>

struct student {

char name[50];

int roll;

float marks;

} s;

int main() {

printf("Enter information:\n");

printf("Enter name: ");

fgets(s.name, sizeof(s.name), stdin);

printf("Enter roll number: ");

scanf("%d", &s.roll);

printf("Enter marks: ");

scanf("%f", &s.marks);

printf("Displaying Information:\n");

printf("Name: ");

printf("%s", s.name);

printf("Roll number: %d\n", s.roll);

printf("Marks: %.1f\n", s.marks);

return 0;

}

**Output**

Enter information:

Enter name: Jack

Enter roll number: 23

Enter marks: 34.5

Displaying Information:

Name: Jack

Roll number: 23

Marks: 34.5

In this program, a structure student is created. The structure has three members: name (string), roll (integer) and marks (float).

Then, a structure variable s is created to store information and display it on the screen.

# C Program to Add Two Distances (in inch-feet system) using Structures

In this example, you will learn to take two distances (in the inch-feet system), add them and display the result on the screen.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C struct](https://www.programiz.com/c-programming/c-structures)

If you do not know, 12 inches is 1 foot.

## Program to add two distances in the inch-feet system

#include <stdio.h>

struct Distance {

int feet;

float inch;

} d1, d2, result;

int main() {

// take first distance input

printf("Enter 1st distance\n");

printf("Enter feet: ");

scanf("%d", &d1.feet);

printf("Enter inch: ");

scanf("%f", &d1.inch);

// take second distance input

printf("\nEnter 2nd distance\n");

printf("Enter feet: ");

scanf("%d", &d2.feet);

printf("Enter inch: ");

scanf("%f", &d2.inch);

// adding distances

result.feet = d1.feet + d2.feet;

result.inch = d1.inch + d2.inch;

// convert inches to feet if greater than 12

while (result.inch >= 12.0) {

result.inch = result.inch - 12.0;

++result.feet;

}

printf("\nSum of distances = %d\'-%.1f\"", result.feet, result.inch);

return 0;

}

**Output**

Enter 1st distance

Enter feet: 23

Enter inch: 8.6

Enter 2nd distance

Enter feet: 34

Enter inch: 2.4

Sum of distances = 57'-11.0"

In this program, a structure Distance is defined. The structure has two members:

* **feet** - an integer
* **inch** - a float

Two variables d1 and d2 of type struct Distance are created. These variables store distances in the feet and inches.

Then, the sum of these two distances are computed and stored in the result variable. Finally, result is printed on the screen.

# C Program to Add Two Complex Numbers by Passing Structure to a Function

In this example, you will learn to take two complex numbers as structures and add them by creating a user-defined function.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C struct](https://www.programiz.com/c-programming/c-structures)
* [C Structure and Function](https://www.programiz.com/c-programming/c-structure-function)

## Add Two Complex Numbers

#include <stdio.h>

typedef struct complex {

float real;

float imag;

} complex;

complex add(complex n1, complex n2);

int main() {

complex n1, n2, result;

printf("For 1st complex number \n");

printf("Enter the real and imaginary parts: ");

scanf("%f %f", &n1.real, &n1.imag);

printf("\nFor 2nd complex number \n");

printf("Enter the real and imaginary parts: ");

scanf("%f %f", &n2.real, &n2.imag);

result = add(n1, n2);

printf("Sum = %.1f + %.1fi", result.real, result.imag);

return 0;

}

complex add(complex n1, complex n2) {

complex temp;

temp.real = n1.real + n2.real;

temp.imag = n1.imag + n2.imag;

return (temp);

}

**Output**

For 1st complex number

Enter the real and imaginary parts: 2.1

-2.3

For 2nd complex number

Enter the real and imaginary parts: 5.6

23.2

Sum = 7.7 + 20.9i

In this program, a structure namedcomplex is declared. It has two members: real and imag. We then created two variables n1 and n2 from this structure.

These two structure variables are passed to the add() function. The function computes the sum and returns the structure containing the sum.

Finally, the sum of complex numbers is printed from the main() function.

# C Program to Calculate Difference Between Two Time Periods

In this example, you will learn to calculate the difference between two time periods using a user-defined function.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C User-defined functions](https://www.programiz.com/c-programming/c-user-defined-functions)
* [C struct](https://www.programiz.com/c-programming/c-structures)
* [C Structure and Function](https://www.programiz.com/c-programming/c-structure-function)
* [C structs and Pointers](https://www.programiz.com/c-programming/c-structures-pointers)

## Calculate Difference Between Two Time Periods

#include <stdio.h>

struct TIME {

int seconds;

int minutes;

int hours;

};

void differenceBetweenTimePeriod(struct TIME t1,

struct TIME t2,

struct TIME \*diff);

int main() {

struct TIME startTime, stopTime, diff;

printf("Enter the start time. \n");

printf("Enter hours, minutes and seconds: ");

scanf("%d %d %d", &startTime.hours,

&startTime.minutes,

&startTime.seconds);

printf("Enter the stop time. \n");

printf("Enter hours, minutes and seconds: ");

scanf("%d %d %d", &stopTime.hours,

&stopTime.minutes,

&stopTime.seconds);

// Difference between start and stop time

differenceBetweenTimePeriod(startTime, stopTime, &diff);

printf("\nTime Difference: %d:%d:%d - ", startTime.hours,

startTime.minutes,

startTime.seconds);

printf("%d:%d:%d ", stopTime.hours,

stopTime.minutes,

stopTime.seconds);

printf("= %d:%d:%d\n", diff.hours,

diff.minutes,

diff.seconds);

return 0;

}

// Computes difference between time periods

void differenceBetweenTimePeriod(struct TIME start,

struct TIME stop,

struct TIME \*diff) {

while (stop.seconds > start.seconds) {

--start.minutes;

start.seconds += 60;

}

diff->seconds = start.seconds - stop.seconds;

while (stop.minutes > start.minutes) {

--start.hours;

start.minutes += 60;

}

diff->minutes = start.minutes - stop.minutes;

diff->hours = start.hours - stop.hours;

}

**Output**

Enter the start time.

Enter hours, minutes and seconds: 13

34

55

Enter the stop time.

Enter hours, minutes and seconds: 8

12

15

Time Difference: 13:34:55 - 8:12:15 = 5:22:40

In this program, the user is asked to enter two time periods and these two periods are stored in structure variables startTime and stopTime respectively.

Then, the function differenceBetweenTimePeriod() calculates the difference between the time periods. The result is displayed from the main() function without returning it (using **call by reference** technique).

# C Program to Store Information of Students Using Structure

In this example, you will learn to store the information of 5 students by using an array of structures.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Arrays](https://www.programiz.com/c-programming/c-arrays)
* [C struct](https://www.programiz.com/c-programming/c-structures)

## Store Information in Structure and Display it

#include <stdio.h>

struct student {

char firstName[50];

int roll;

float marks;

} s[5];

int main() {

int i;

printf("Enter information of students:\n");

// storing information

for (i = 0; i < 5; ++i) {

s[i].roll = i + 1;

printf("\nFor roll number%d,\n", s[i].roll);

printf("Enter first name: ");

scanf("%s", s[i].firstName);

printf("Enter marks: ");

scanf("%f", &s[i].marks);

}

printf("Displaying Information:\n\n");

// displaying information

for (i = 0; i < 5; ++i) {

printf("\nRoll number: %d\n", i + 1);

printf("First name: ");

puts(s[i].firstName);

printf("Marks: %.1f", s[i].marks);

printf("\n");

}

return 0;

}

**Output**

Enter information of students:

For roll number1,

Enter name: Tom

Enter marks: 98

For roll number2,

Enter name: Jerry

Enter marks: 89

.

.

.

Displaying Information:

Roll number: 1

Name: Tom

Marks: 98

.

.

.

In this program, a structure student is created. The structure has three members: name (string), roll (integer) and marks (float).

Then, we created an array of structures s having 5 elements to store information of 5 students.

Using a for loop, the program takes the information of 5 students from the user and stores it in the array of structure. Then using another for loop, the information entered by the user is displayed on the screen.

# C Program to Store Data in Structures Dynamically

In this example, you will learn to store the information entered by the user using dynamic memory allocation.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Pointers](https://www.programiz.com/c-programming/c-pointers)
* [C Dynamic Memory Allocation](https://www.programiz.com/c-programming/c-dynamic-memory-allocation)
* [C struct](https://www.programiz.com/c-programming/c-structures)

This program asks the user to store the value of noOfRecords and allocates the memory for the noOfRecords structure variables dynamically using the malloc() function.

## Demonstrate the Dynamic Memory Allocation for Structure

#include <stdio.h>

#include <stdlib.h>

struct course {

int marks;

char subject[30];

};

int main() {

struct course \*ptr;

int noOfRecords;

printf("Enter the number of records: ");

scanf("%d", &noOfRecords);

// Memory allocation for noOfRecords structures

ptr = (struct course \*)malloc(noOfRecords \* sizeof(struct course));

for (int i = 0; i < noOfRecords; ++i) {

printf("Enter subject and marks:\n");

scanf("%s %d", (ptr + i)->subject, &(ptr + i)->marks);

}

printf("Displaying Information:\n");

for (int i = 0; i < noOfRecords; ++i) {

printf("%s\t%d\n", (ptr + i)->subject, (ptr + i)->marks);

}

free(ptr);

return 0;

}

**Output**

Enter the number of records: 2

Enter subject and marks:

Science 82

Enter subject and marks:

DSA 73

Displaying Information:

Science 82

DSA 73

# C Program to Write a Sentence to a File

In this example, you will learn to write a sentence in a file using fprintf() statement.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C File Handling](https://www.programiz.com/c-programming/c-file-input-output)
* [C Standard Library Functions](https://www.programiz.com/c-programming/library-function)

This program stores a sentence entered by the user in a file.

#include <stdio.h>

#include <stdlib.h>

int main() {

char sentence[1000];

// creating file pointer to work with files

FILE \*fptr;

// opening file in writing mode

fptr = fopen("program.txt", "w");

// exiting program

if (fptr == NULL) {

printf("Error!");

exit(1);

}

printf("Enter a sentence:\n");

fgets(sentence, sizeof(sentence), stdin);

fprintf(fptr, "%s", sentence);

fclose(fptr);

return 0;

}

**Output**

Enter a sentence: C Programming is fun

Here, a file named program.txt is created. The file will contain C programming is fun text.

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

Then, a file named **program.txt** is opened in writing mode. If the file does not exist, it will be created.

Finally, the string entered by the user will be written to this file using the fprintf() function and the file is closed.

# C Program to Read the First Line From a File

In this example, you will learn to print the first line from a file in C programming.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C File Handling](https://www.programiz.com/c-programming/c-file-input-output)
* [C Programming Strings](https://www.programiz.com/c-programming/c-strings)

## Program to read the first line from a file.

#include <stdio.h>

#include <stdlib.h> // For exit() function

int main() {

char c[1000];

FILE \*fptr;

if ((fptr = fopen("program.txt", "r")) == NULL) {

printf("Error! File cannot be opened.");

// Program exits if the file pointer returns NULL.

exit(1);

}

// reads text until newline is encountered

fscanf(fptr, "%[^\n]", c);

printf("Data from the file:\n%s", c);

fclose(fptr);

return 0;

}

If the file is found, the program saves the content of the file to a string c until '\n' newline is encountered.

Suppose the program.txt file contains the following text in the current directory.

C programming is awesome.

I love C programming.

How are you doing?

The output of the program will be:

Data from the file:

C programming is awesome.

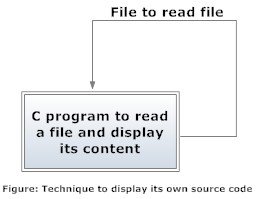
If the file program.txt is not found, the program prints the error message.

# C Program to Display its own Source Code as Output

In this example, you'll learn to display source of the program using \_\_FILE\_\_ macro.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C Preprocessor and Macros](https://www.programiz.com/c-programming/c-preprocessor-macros)
* [C File Handling](https://www.programiz.com/c-programming/c-file-input-output)

Though this problem seems complex, the concept behind this program is straightforward; display the content from the same file you are writing the source code.

In C programming, there is a predefined macro named \_\_FILE\_\_ that gives the name of the current input file.

#include <stdio.h>

int main() {

  // location the current input file.

printf("%s",\_\_FILE\_\_);

}

### C program to display its own source code

#include <stdio.h>

int main() {

FILE \*fp;

int c;

  // open the current input file

fp = fopen(\_\_FILE\_\_,"r");

do {

c = getc(fp); // read character

putchar(c); // display character

}

while(c != EOF); // loop until the end of file is reached

  fclose(fp);

return 0;

}

# C Program to Print Pyramids and Patterns

In this example, you will learn to print half pyramids, inverted pyramids, full pyramids, inverted full pyramids, Pascal's triangle, and Floyd's triangle in C Programming.

To understand this example, you should have the knowledge of the following [C programming](https://www.programiz.com/c-programming) topics:

* [C if...else Statement](https://www.programiz.com/c-programming/c-if-else-statement)
* [C for Loop](https://www.programiz.com/c-programming/c-for-loop)
* [C while and do...while Loop](https://www.programiz.com/c-programming/c-do-while-loops)
* [C break and continue](https://www.programiz.com/c-programming/c-break-continue-statement)

Here is a list of programs you will find in this page.

| C Examples |
| --- |
| Half pyramid of \* |
| Half pyramid of numbers |
| Half pyramid of alphabets |
| Inverted half pyramid of \* |
| Inverted half pyramid of numbers |
| Full pyramid of \* |
| Full pyramid of numbers |
| Inverted full pyramid of \* |
| Pascal's triangle |
| Floyd's triangle |

### Example 1: Half Pyramid of \*

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

**C Program**

#include <stdio.h>

int main() {

int i, j, rows;

printf("Enter the number of rows: ");

scanf("%d", &rows);

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

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

printf("\* ");

}

printf("\n");

}

return 0;

}

### Example 2: Half Pyramid of Numbers

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

**C Program**

#include <stdio.h>

int main() {

int i, j, rows;

printf("Enter the number of rows: ");

scanf("%d", &rows);

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

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

printf("%d ", j);

}

printf("\n");

}

return 0;

}

### Example 3: Half Pyramid of Alphabets

A

B B

C C C

D D D D

E E E E E

**C Program**

#include <stdio.h>

int main() {

int i, j;

char input, alphabet = 'A';

printf("Enter an uppercase character you want to print in the last row: ");

scanf("%c", &input);

for (i = 1; i <= (input - 'A' + 1); ++i) {

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

printf("%c ", alphabet);

}

++alphabet;

printf("\n");

}

return 0;

}

### Example 4: Inverted half pyramid of \*

\* \* \* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

**C Program**

#include <stdio.h>

int main() {

int i, j, rows;

printf("Enter the number of rows: ");

scanf("%d", &rows);

for (i = rows; i >= 1; --i) {

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

printf("\* ");

}

printf("\n");

}

return 0;

}

### Example 5: Inverted half pyramid of numbers

1 2 3 4 5

1 2 3 4

1 2 3

1 2

1

**C Program**

#include <stdio.h>

int main() {

int i, j, rows;

printf("Enter the number of rows: ");

scanf("%d", &rows);

for (i = rows; i >= 1; --i) {

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

printf("%d ", j);

}

printf("\n");

}

return 0;

}

### Example 6: Full Pyramid of \*

\*

\* \* \*

\* \* \* \* \*

\* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \*

**C Program**

#include <stdio.h>

int main() {

int i, space, rows, k = 0;

printf("Enter the number of rows: ");

scanf("%d", &rows);

for (i = 1; i <= rows; ++i, k = 0) {

for (space = 1; space <= rows - i; ++space) {

printf(" ");

}

while (k != 2 \* i - 1) {

printf("\* ");

++k;

}

printf("\n");

}

return 0;

}

### Example 7: Full Pyramid of Numbers

1

2 3 2

3 4 5 4 3

4 5 6 7 6 5 4

5 6 7 8 9 8 7 6 5

**C Program**

#include <stdio.h>

int main() {

int i, space, rows, k = 0, count = 0, count1 = 0;

printf("Enter the number of rows: ");

scanf("%d", &rows);

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

for (space = 1; space <= rows - i; ++space) {

printf(" ");

++count;

}

while (k != 2 \* i - 1) {

if (count <= rows - 1) {

printf("%d ", i + k);

++count;

} else {

++count1;

printf("%d ", (i + k - 2 \* count1));

}

++k;

}

count1 = count = k = 0;

printf("\n");

}

return 0;

}

### Example 8: Inverted full pyramid of \*

\* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \*

\* \* \* \* \*

\* \* \*

\*

**C Program**

#include <stdio.h>

int main() {

int rows, i, j, space;

printf("Enter the number of rows: ");

scanf("%d", &rows);

for (i = rows; i >= 1; --i) {

for (space = 0; space < rows - i; ++space)

printf(" ");

for (j = i; j <= 2 \* i - 1; ++j)

printf("\* ");

for (j = 0; j < i - 1; ++j)

printf("\* ");

printf("\n");

}

return 0;

}

### Example 9: Pascal's Triangle

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

**C Program**

#include <stdio.h>

int main() {

int rows, coef = 1, space, i, j;

printf("Enter the number of rows: ");

scanf("%d", &rows);

for (i = 0; i < rows; i++) {

for (space = 1; space <= rows - i; space++)

printf(" ");

for (j = 0; j <= i; j++) {

if (j == 0 || i == 0)

coef = 1;

else

coef = coef \* (i - j + 1) / j;

printf("%4d", coef);

}

printf("\n");

}

return 0;

}

### Example 10: Floyd's Triangle.

1

2 3

4 5 6

7 8 9 10

**C Program**

#include <stdio.h>

int main() {

int rows, i, j, number = 1;

printf("Enter the number of rows: ");

scanf("%d", &rows);

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

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

printf("%d ", number);

++number;

}

printf("\n");

}

return 0;

}