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 topics:

- C Input Output (I/O)

Program to Display "Hello, World!"

```c
#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 topics:

- C Variables, Constants and Literals
- C Data Types
- C Input Output (I/O)

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 topics:

- C Data Types
- C Variables, Constants and Literals
- C Input Output (I/O)
- C Programming Operators

Program to Add Two Integers

```c
#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

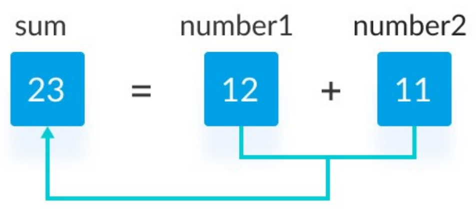
```
Enter two integers: 12
11
12 + 11 = 23
```

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

```c
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.

```c
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 topics:

- C Variables, Constants and Literals
- C Data Types
- C Input Output (I/O)
- C Programming 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 topics:

- C Data Types
- C Variables, Constants and Literals
- C Input Output (I/O)

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 topics:

- C Data Types
- C Variables, Constants and Literals
- C Input Output (I/O)
- C Programming Operators

Program to Compute Quotient and Remainder

```c
#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.

```c
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 topics:

- C Data Types
- C Variables, Constants and Literals
- C Input Output (I/O)

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 topics:

- C Data Types
- C Variables, Constants and Literals
- C Input Output (I/O)

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 topics:

- C Data Types
- C Programming Operators
- C Input Output (I/O)

Swap Numbers Using Temporary Variable

```c
#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

```c
#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
```