delay in C

In this post, we will see how to give a time delay in C code. Basic idea is to get current clock and add the required delay to that clock, till current clock is less than required clock run an empty loop.

Here is implementation with a delay function.

```
// C function showing how to do time delay
#include <stdio.h>
// To use time library of C
#include <time.h>

void delay(int number_of_seconds)
{
    // Converting time into milli_seconds
    int milli_seconds = 1000 * number_of_seconds;

    // Stroing start time
    clock_t start_time = clock();

    // looping till required time is not acheived
    while (clock() < start_time + milli_seconds)
        ;
}


// Driver code to test above function
int main()
{
    int i;
    for (i = 0; i < 10; i++) {
        // delay of one second
        delay(1);
        printf("%d seconds have passed\n", i + 1);
    }
    return 0;
}
```

Output:

```
1 seconds have passed
…
2 seconds have passed
….
….
```

In C, the **`toupper()`** function is used to convert lowercase alphabets to uppercase letters.

When a **lowercase** alphabet is passed to the toupper() function it converts it to uppercase.

Note: A `ctype.h` header file needs to be included in order to use this function.

Syntax

The function takes in an alphabet as a **parameter** and **returns** that character in uppercase.

See the function definition below:

```
int toupper(int ch);
```

```
#include <ctype.h>

int main()
{
    char ch;

    // letter to convert to uppercase
    ch = 'a';

    printf("%c in uppercase is represented as %c",
        ch, toupper(ch));

    return 0;
}
```

C tolower()

The tolower() function takes an uppercase alphabet and convert it to a lowercase character.

If the arguments passed to the tolower() function is other than an uppercase alphabet, it returns the same character that is passed to the function.

It is defined in ctype.h header file.

Function Prototype of tolower()

```
int tolower(int argument);
```

```
#include <stdio.h>
#include <ctype.h>
int main()
{
    char c, result;

    c = 'M';
    result = tolower(c);
    printf("tolower(%c) = %c\n", c, result);
```

```
    c = 'm';
    result = tolower(c);
    printf("tolower(%c) = %c\n", c, result);

    c = '+';
    result = tolower(c);
    printf("tolower(%c) = %c\n", c, result);

    return 0;
}
```

---

isalpha()

Return Value	Remarks
Zero (0)	If the parameter isn't an alphabet.
Non zero number	If the parameter is an alphabet.

Example: C isalpha() function

```
#include <stdio.h>
#include <ctype.h>
int main()
{
    char c;
    c = 'Q';
    printf("\nResult when uppercase alphabet is passed: %d", isalpha(c));

    c = 'q';
    printf("\nResult when lowercase alphabet is passed: %d", isalpha(c));

    c='+';
    printf("\nResult when non-alphabetic character is passed: %d",
isalpha(c));

    return 0;
}
```

Output

```
Result when uppercase alphabet is passed: 1
Result when lowercase alphabet is passed: 2
Result when non-alphabetic character is passed: 0
```

C iscntrl()

The iscntrl() function checks whether a character is a control character or not. Characters that cannot be printed on the screen are known as control characters. For example, backspace, Escape, newline etc.

```
int iscntrl(int argument);
```

EXAMPLE

```
#include <stdio.h>
#include <ctype.h>

int main()
{
    char c;
    int result;

    c = 'Q';
    result = iscntrl(c);
    printf("When %c is passed to iscntrl() = %d\n", c, result);

    c = '\n';
    result = iscntrl(c);
    printf("When %c is passed to iscntrl() = %d", c, result);

    return 0;
}
```