Sizeof is a compile time, unary operator which can be used to compute the size of its operands. The result of the sizeof is an unsigned integer.

Usage:

a) When operand is a data type, it simply returns the amount of the memory allocated to the data type. We have seen this before while discussing data types:

```
#include <stdio.h>
int main()
{
        printf ("%d ", sizeof(char));
        printf ("%d ", sizeof(int));
        printf ("%d ", sizeof(float));
        printf ("%d ", sizeof(double));
        return 0;
}
```

Output: 1   4   4   8

b) When the operand is an expression, it returns the size of the result of that expression.

```
#include <stdio.h>
int main()
{
        int a = 10; double a = 10.21;
        printf ("%d ", sizeof(a+d));
        return 0;
}
```

Output: 8

Facts on sizeof:

a) Typename must be specified in parentheses whereas expression can be specified with or without the parentheses.

```
i.e.    sizeof (typename);
        sizeof (expression);        or      sizeof  expression;
```

b) Statements written inside sizeof () are not executed. Here are following 3 examples to illustrate this:

- Example 1:

```
#include <stdio.h>

int main()

{

    int i = 5;

    int size = sizeof (i++);

    printf("%d %d\n", size, i)
    return 0;
}
```

Output: 4 5

- Example 2:

```
#include <stdio.h>
int main()
{
    int i =  sizeof (printf("Hello!"));
    printf("%d\n" i)
    return 0;
}
```

Output: 4

From this example, it is clear that printf() returns an integer value.

- Example 3:

```
#include <stdio.h>
int main()
{
    int a = 5;
    int b = sizeof (a = 6);
    printf("%d %d\n", a, b);
    return 0;
}
```

Output: 5, 4

A statement like this – int b = sizeof (int a = 6); keeping rest of the code same will give an error, and the error won't be of redeclaration of integer a, instead it will be –

[Error]missing ')' after 'int'

Because the compiler thinks this is our attempt of type casting (yeah, our next topic).