It is possible to pass some values from the command line to your C programs when they are executed. These values are called command line arguments and many times they are important for your program specially when you want to control your program from outside instead of hard coding those values inside the code.

Command Line Arguments

The command line arguments are handled using `main()` function arguments. main receives two arguments from the Operating System (Shell); traditionally they are called `argc` and `argv`, though any name can be given. `argc` is an integer and refers to the number of arguments passed, and `argv[]` is a pointer array which points to each argument passed to the program as a string.

```
int main(int argc, char* argv[]) {
```

Now argc and argv are like any variables in the program.

1. The *main()* can and should check `argc` to see how many arguments the user specified.
2. The minimum count for `argc` is 1; in this case the *argument* is just the name of the program itself.
3. Which means, a program can find out its own name as it was invoked! It is the string `argv[0]`. This is OS dependent.
4. The arguments are always strings. If you expect to receive an integer as the argument, you should convert it inside the code.
5. Command line arguments are separated by a space, but if an argument itself has a space then you can pass such arguments by putting them inside quotes.

Example

Let us start with a simple example:

```
1 #include <stdio.h>
2 int main(int argc, char* argv[]) {
3     for (int i = 0; i < argc; i++) {
4         printf("Argument #%d is <%s>\n", i, argv[i]);
```



```
5       }
6       return 0;
7   }
```

When the above code is compiled and executed with a single argument, it produces the following result.

```
$
$ c99 Program-12-1.c
$ ./a.out
Argument #0 is <./a.out>
$ ./a.out Hello World! This is Asokan Pichai
Argument #0 is <./a.out>
Argument #1 is <Hello>
Argument #2 is <World!>
Argument #3 is <This>
Argument #4 is <is>
Argument #5 is <Asokan>
Argument #6 is <Pichai>
$ ./a.out "Hello World" 'This is Asokan Pichai'
Argument #0 is <./a.out>
Argument #1 is <Hello World>
Argument #2 is <This is Asokan Pichai>
$
```

Figure 1: Echoing Command line arguments

A more complicated example: let us build a command line calculator for binary expressions.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int main(int argc, char* argv[]) {
4      if (argc != 4) {
5          printf("Usage: %s Number operator Number\n", argv[0]);
6          printf("Remember to quote * as it has special meaning\n");
7          exit(0);
8      }
9      int a = atoi(argv[1]);
10     int b = atoi(argv[3]);
11     char op = argv[2][0];
12     int result;
```



```
13
14      if (op == '+')
15          result = a + b;
16      else if (op == '-')
17          result = a - b;
18      else if (op == '*')
19          result = a * b;
20      else if (op == '/')
21          result = a / b;
22      else {
23          printf("Invalid operator <%c>\n", op);
24          exit(1);
25      }
26      printf("%d %c %d = %d\n", a, op, b, result);
27      return 0;
28  }
```

```
$
$ c99 Program-12-2.c
$ ./a.out
Usage: ./a.out Number operator Number
Remember to quote * as it has special meaning
$ ./a.out 13 + 13
13 + 13 = 26
$ ./a.out 29 / 12
29 / 12 = 2
$ ./a.out 9 * 12
Usage: ./a.out Number operator Number
Remember to quote * as it has special meaning
$ ./a.out 9 '*' 12
9 * 12 = 108
$ ./a.out 9 % 12
Invalid operator <%>
$ ./a.out 23 - 76
23 - 76 = -53
$
```

Figure 2: Command line calculator

Please note that we have not validated that only integers have been entered.

