Int main(int argc, *char argv[])

How you will retrieve the value fed in command prompt

$>

Gedit a.c

gcc

Command line arguments in C

Arguments passed to main function is called command line arguments.

Command-line arguments are given after the name of the program in command-line shell of Operating Systems.

To pass command line arguments, we typically define main() with two arguments : first argument is the number of command line arguments and second is list of command-line arguments.

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

or

```
int main(int argc, char **argv) { /* ... */ }
```

- **argc (ARGument Count) is int and stores number of command-line arguments passed by the user including the name of the program. So if we pass a value to a program, value of argc would be 2 (one for argument and one for program name)**
- **The value of argc should be non-negative.**
- **argv(ARGument Vector) is array of character pointers listing all the arguments.**

- **If argc is greater than zero,the array elements from argv[0] to argv[argc-1] will contain pointers to strings.**
- **Argv[0] is the name of the program , After that till argv[argc-1] every element is command -line arguments.**

gedit cla.c

```
// Name of program cla.c
#include <stdio.h>
int main(int argc, char *argv[])
{
   printf("\n no of arguments are %d", argc);
     for (int i = 0; i < argc; ++i)
       printf("%s\n", argv[i]);
    return 0;
}
```

Input:

```
$ gcc cla.c -o a  //-o is redirection to an output file   - a   .obj   bug free bits
$ ./a I love computers
```

Output:

```
no of arguments are 4
./a
I
Love
Computers
```

Properties of Command Line Arguments:

1. **They are passed to main() function.**
2. **They are parameters/arguments supplied to the program when it is invoked.**
3. **They are used to control program from outside instead of hard coding those values inside the code.**
4. **argv[argc] is a NULL pointer.**
5. **argv[0] holds the name of the program.**
6. **argv[1] points to the first command line argument and argv[n] points last argument.**

Note : You pass all the command line arguments separated by a space, but if argument itself has a space then you can pass such arguments by putting them inside double quotes “” or single quotes ”.

```
#include<stdio.h>
int main(int argc,char* argv[])
{
   int counter;
   printf("Program Name Is: %s",argv[0]);
   if(argc==1)
      printf("\nNo Extra Command Line Argument Passed Other Than
Program Name");
   if(argc>=2)
   {
      printf("\nNumber Of Arguments Passed: %d",argc);
      printf("\n----Following Are The Command Line Arguments Passed----
");
      for(counter=0;counter<argc;counter++)
         printf("\nargv[%d]: %s",counter,argv[counter]);
   }
   return 0;
}
```

Output in different scenarios:

Without argument: When the above code is compiled and executed without passing any argument, it produces following output.

```
$ ./a.out

Program Name Is: ./a.out

No Extra Command Line Argument Passed Other Than Program Name
```

Three arguments : When the above code is compiled and executed with a three arguments, it produces the following output.

```
$ ./a.out First Second Third

Program Name Is: ./a.out

Number Of Arguments Passed: 4

----Following Are The Command Line Arguments Passed----

argv[0]: ./a.out

argv[1]: First

argv[2]: Second

argv[3]: Third
```

Single Argument : When the above code is compiled and executed with a single argument separated by space but inside double quotes, it produces the following output.

```
$ ./a.out "First Second Third"

Program Name Is: ./a.out
```

```
Number Of Arguments Passed: 2

----Following Are The Command Line Arguments Passed----

argv[0]: ./a.out

argv[1]: First Second Third
```

Single argument in quotes separated by space : When the above code is compiled and executed with a single argument separated by space but inside single quotes, it produces the following output.

```
$ ./a.out 'First Second Third'

Program Name Is: ./a.out

Number Of Arguments Passed: 2

----Following Are The Command Line Arguments Passed----

argv[0]: ./a.out

argv[1]: First Second Third
```

Find the sum of two integer numbers using command line arguments

```c
#include <stdio.h>
int main(int argc, char *argv[])
{
        int a,b,sum;
        if(argc!=3)
        {
                printf("please use \"prg_name value1 value2 \"\n");
                return -1;
        }
        a = atoi(argv[1]);
        b = atoi(argv[2]);
        sum = a+b;
        printf("Sum of %d, %d is: %d\n",a,b,sum);
        return 0;
}
```

First run:

```
$ ./main 10 20
Sum of 10, 20 is: 30
```

Second run:

```
$ ./main 10 20 30 40
```

What is atoi()?

atoi() is a library function that converts string to integer, when program gets the input from command line, string values transfer in the program, we have to convert them to integers. atoi() is used to return the integer of the string arguments.