# **Command line arguments example in C**

Prerequisite: [Command\_line\_argument](https://www.geeksforgeeks.org/command-line-arguments-in-c-cpp/). The problem is to find the largest integer among the three using command line arguments. **Notes:**

* **Command-line arguments** are given after the name of the program in the command-line shell of Operating Systems. To pass command line arguments, we typically define main() with two arguments: the first argument is the number of command-line arguments and the second is a list of command-line arguments.

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

* **atoi** – Used to convert string numbers to integers

Examples:

Input : filename 8 9 45

Output : 45 is largest

Input : filename 8 9 9

Output : Two equal number entered

Input : filename 8 -9 9

Output : negative number entered

[Recommended: Please try your approach on ***{IDE}*** first, before moving on to the solution.](https://ide.geeksforgeeks.org/)

When calling the program, we pass three integers along with its filename, and then the program prints out the largest of the three numbers. **Approach:**

1. The program “return 1” if one of the two following conditions is satisfied:

* If any two numbers are the same, print the statement “two equal numbers entered”.
* If any of the numbers is negative, print “negative number entered”.

1. Else “return 0” if three different integers are entered.

For better understanding, run this code for yourself.

|  |
| --- |
| // C program for finding the largest integer  // among three numbers using command line arguments  #include <stdio.h>  #include <stdlib.h>    // Taking argument as command line  intmain(intargc, char\*argv[])  {      inta, b, c;        // Checking if number of argument is      // equal to 4 or not.      if(argc < 4 || argc > 5)      {          printf("enter 4 arguments only eg.\"filename arg1 arg2 arg3!!\"");          return0;      }        // Converting string type to integer type      // using function "atoi( argument)"      a = atoi(argv[1]);      b = atoi(argv[2]);      c = atoi(argv[3]);        // Checking if all the numbers are positive of not      if(a < 0 || b < 0 || c < 0)      {          printf("enter only positive values in arguments !!");          return1;      }        // Checking if all the numbers are different or not      if(!(a != b && b != c && a != c))      {          printf("please enter three different value ");          return1;      }      else      {          // Checking condition for "a" to be largest          if(a > b && a > c)              printf("%d is largest", a);            // Checking condition for "b" to be largest          elseif(b > c && b > a)              printf("%d is largest", b);            // Checking condition for "c" to be largest..          elseif(c > a && c > b)              printf("%d is largest ",c);      }      return0;  } |

**Output :**



# **Command line arguments in C/C++**

The most important function of C/C++ is main() function. It is mostly defined with a return type of int and without parameters :

int main() { /\* ... \*/ }

We can also give command-line arguments in C and C++. 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.

For better understanding run this code on your linux machine.

|  |
| --- |
| // Name of program mainreturn.cpp  #include <iostream>  usingnamespacestd;    intmain(intargc, char\*\* argv)  {      cout << "You have entered "<< argc           << " arguments:"<< "\n";        for(inti = 0; i < argc; ++i)          cout << argv[i] << "\n";        return0;  } |

Input:

$ g++ mainreturn.cpp -o main

$ ./main geeks for geeks

Output:

You have entered 4 arguments:

./main

geeks

for

geeks

**Note :** Other platform-dependent formats are also allowed by the C and C++ standards; for example, Unix (though not POSIX.1) and Microsoft Visual C++ have a third argument giving the program’s environment, otherwise accessible through getenv in stdlib.h: Refer [C program to print environment variables](https://www.geeksforgeeks.org/c-program-print-environment-variables/) for details.

**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 ”.

|  |
| --- |
| // C program to illustrate  // command line arguments  #include<stdio.h>    intmain(intargc,char\* argv[])  {      intcounter;      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]);      }      return0;  } |

**Output in different scenarios:**

1. **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

1. **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

1. **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

1. **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

**References:**<http://www.cprogramming.com/tutorial/lesson14.html>  
<http://c0x.coding-guidelines.com/5.1.2.2.1.html>

This article is contributed by **Kartik Ahuja** and [**Avadhut Patade**](https://in.linkedin.com/in/avadhut-patade-4b5a5069). If you like GeeksforGeeks and would like to contribute, you can also write an article using [write.geeksforgeeks.org](http://www.write.geeksforgeeks.org/) or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.