atexit() function in C/C++

The function pointed by *atexit()* is automatically called without arguments when the program terminates normally. In case more than one *atexit* function has been specified by different calls to this function, all are executed in the order of a [stack](http://www.geeksforgeeks.org/stack-data-structure/) (i.e. the last function specified is the first to be executed at exit). A single function can be registered to be executed at exit more than once.

**Syntax :**

extern "C" int atexit (void (\*func)(void)) noexcept;

extern "C++" int atexit (void (\*func)(void)) noexcept

**Note:** **extern** refers that the name will refer, to the same object in the entire program.

**Parameters:** The function accepts a single mandatory parameter **fun** which specifies the pointer to the function to be called on normal program termination (Function to be called).

**Return Value:** The function returns two values:

* Zero, if the function registration is successful
* Non zero, if the function registration failed

Below programs illustrates the above-mentioned function:

**Program 1:**

|  |
| --- |
| // C++ program to illusttrate  // atexit() function  #include <bits/stdc++.h>  using namespace std;    // Returns no value, and takes nothing as a parameter  void done()  {      cout << "Exiting Successfully"           << "\n"; // Executed second  }  // Driver Code  int main()  {      int value;      value = atexit(done);        if (value != 0) {          cout << "atexit () function registration failed";          exit(1);      }      cout << " Registration successful"           << "\n"; // Executed First      return 0;  } |

**Output:**

Registration successful

Exiting Successfully

If more than one *atexit* function have been called in the main, then all the specified function will be executed in a reverse manner, same as of the functioning of the stack.**Program 2:**

|  |
| --- |
| // C++ program to illustrate  // more than one atexit function  #include <bits/stdc++.h>  using namespace std;    // Executed last, in a Reverse manner  void first()  {      cout << "Exit first" << endl;  }    // Executed third  void second()  {      cout << "Exit Second" << endl;  }    // Executed Second  void third()  {      cout << "Exit Third" << endl;  }    // Executed first  void fourth()  {      cout << "Exit Fourth" << endl;  }  // Driver Code  int main()  {      int value1, value2, value3, value4;      value1 = atexit(first);      value2 = atexit(second);      value3 = atexit(third);      value4 = atexit(fourth);      if ((value1 != 0) or (value2 != 0) or          (value3 != 0) or (value4 != 0)) {          cout << "atexit() function registration Failed" << endl;          exit(1);      }      // Executed at the starting      cout << "Registration successful" << endl;      return 0;  } |

**Output:**

Registration successful

Exit Fourth

Exit Third

Exit Second

Exit first

**Program 3:**

|  |
| --- |
| // C++ program to illustrate  // atexit() function when it throws an exception.  #include <bits/stdc++.h>  using namespace std;    void shows\_Exception()  {      int y = 50, z = 0;      // Program will terminate here      int x = y / z;        // Cannot get printed as the program      // has terminated      cout << "Divided by zero";  }  // Driver Code  int main()  {      int value;      value = atexit(shows\_Exception);      if (value != 0) {          cout << "atexit() function registration failed";          exit(1);      }        // Executed at the starting      cout << "Registration successful" << endl;      return 0;  } |

**Note:** If a registered function throws an exception which cannot be handled, then the terminate function is called.