exit statement in C++ with example

exit non-returning function in C++ used to terminate the program. where we terminate a program in two ways, normal termination and abnormal termination.

normally, `exit()` function reports the operating system that program performs a successful execution or not. `exit()` function terminates the program with error code where 0 indicates normal termination and non-zero indicates abnormal termination. abnormal termination is related to data overflow, size exceeded, incomplete file operation etc.

before terminating the Program `exit()` does the following clean-up,

- Terminate the calling process.
- close all files.
- calls registered function (if available).

`exit()` unction is defined in header file `stdlib` and process in the program. Therefore, we have to include their header-files in the program.

Here is the syntax,

```
exit(error-code);
```

where **error-code** indicates normal or abnormal Program termination, normally 0 indicates normal and other value indicates abnormal termination.

However, we can also use the following two constant, defined in header files `stdlib` and `process`

- *`EXIT_SUCCESS`*, represent by zero

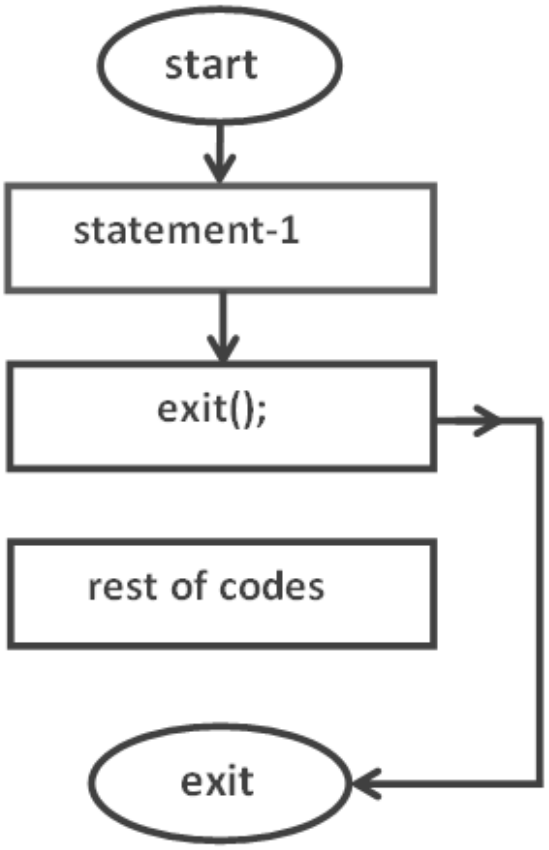
- *EXIT_FAILURE*, represent by none-zero

such as,

exit(0) or *exit(EXIT_SUCCESS)*	indicates normal program termination
exit(1) or *exit(EXIT_FAILURE)*	indicates abnormal program termination

here we will terminate the program in a normal and abnormal. The following first Two examples are abnormal termination,

Flow-diagram of exit statement in C++



as you can see in the above diagram when the execution goes to the **exit block** program will immediately exit or terminate and **rest of codes block** (extra statements in the program)will skip.

Let's try with an example here,

Example of exit in C++

In the following Program, we are trying to open a file in the reading mode (**ifstream** class),

```
#include<process>

#include<fstream>

#include<iostream>

using namespace std;

int main()

 {

     ifstream fin;

     fin.open("file.txt");

      if(!fin)  // test for an error on the stream

        {

            cout<<"File can't open...\n"; // display an error message

            getch();

            exit(EXIT_FAILURE);

        }

      else {
```

```
            cout<<"statement never executed...";

      }

    fin.close();

  }
```

OUTPUT

```
File can't open...
```

Explanation:

The program trying to open a file for reading but there is no file exist for reading. so here an execution failure error arises.

Here is another example,

where we working with graphics, before performing the task we will check graphic is responding or not,

```
#include <graphics>

#include <stdlib>

#include <stdio>

#include<iostream>

using namespace std;



 int main()
```

```
  {

     int gdriver = DETECT, gmode, error_code;

     initgraph(&gdriver, &gmode, "");

        error_code = graphresult();

      if(error_code != grOk)// check graphic driver responding or not

        {

           cout<<"Graphics error:\n"<<grapherrormsg(error_code);

           getch();

           exit(EXIT_FAILURE); // report Execution failure

        }

      // draw a line

     line(0, 0, getmaxx(), getmaxy());



     closegraph();

  }
```

OUTPUT

```
Graphic Error:

Device driver file not found (EGAVGA.BGI)
```

now let's perform normal termination in the following programs,

In the following Program, we perform a normal program termination which will depend on user input.

```
#include<iostream>

#include<stdlib>

using namespace std;

int main()

  {

      int num1,num2;

       for(int i=0; i>=0; i++) //infinite loop

         {

            cout<<"Enter Two number: ";

            cin>>num1>>num2;

            if(!num1||!num2)  {

                 cout<<"Program Exiting...";

                 exit(0);    // EXIT_SUCCESS

               }

            else

                 cout<<"sum: "<<num1+num2;
```

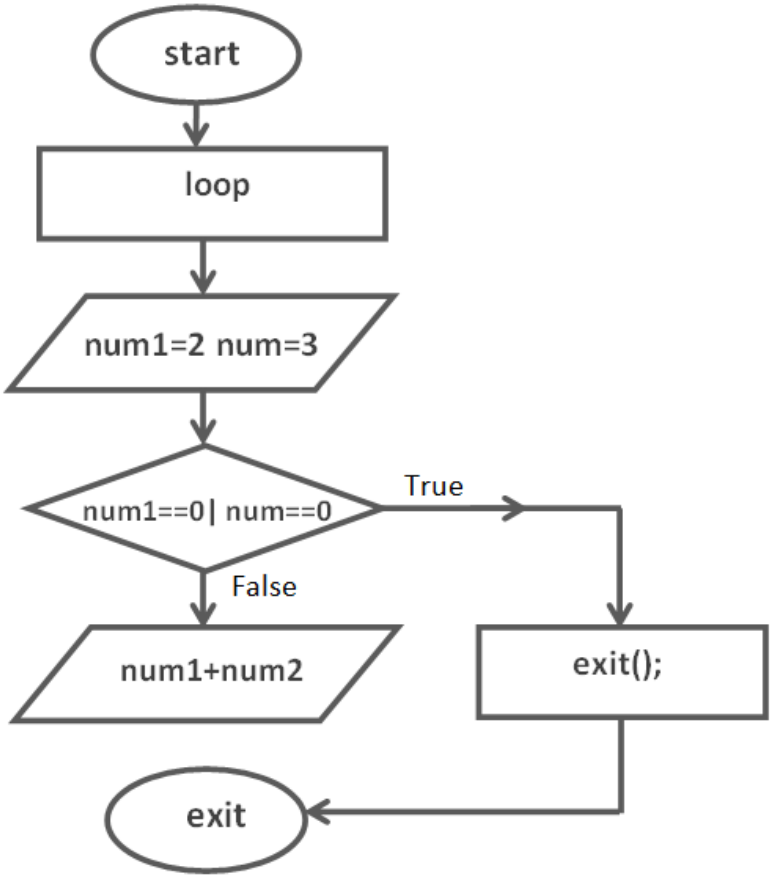
```
    return 0;

        }

  }
```

OUTPUT

```
Enter number: 3 6

sum 9

Enter number: 2 5

sum 7

Enter number: 0 3

Program Exiting...
```

Explanation

as soon as the value of one of the two variables is 0, the program performs normal exit. below flow-diagram is the demonstration of the above codes.



Example of exit() in a menu-driven program

In the below program we exiting from a menu-driven,

```
#include<iostream>

#include<stdlib>

using namespace std;

int input();

int input()

    {
```

```
        int value;

        cout<<"Enter Degree: ";

        cin>>value;

        return value;

    }

int main()

 {

   int select,result;

 do{

    clrscr();

    cout<<"1> celsius To Fahrenheit\n";

    cout<<"2> Fahrenheit To Celsius\n";

    cout<<"3> Celsius To Kelvin\n";

    cout<<"4> Kelvin To Celsius\n";

    cout<<"5> Fahrenheit To Kelvin\n";

    cout<<"0> Exit\n\n";



    cin>>select;
```

```
switch(select)  {
case 1:
        result=(input()*9/5)+32;
        cout<<result<<"f";
        break;
case 2:
        result=(input()-32)*5/9;
        cout<<result<<"c";
        break;
case 3:
        result=input()+273;
        cout<<result<<"k";
        break;
case 4:
        result=input()-273;
        cout<<result<<"c";
         break;
```

```
      case 5:

              result=(input()+459)*5/9;

              cout<<result<<"k";

              break;

      case 0: exit(0);

      default:

              cout<<"Not available..";

    }

  }while(select!=0);

}
```

OUTPUT

```
Enter degree: 212

100c
```

difference between break and exit statement in C++

in simple terms, an `exit()` function is always used to terminate an entire program while the `break` terminates the single statement only.

break statement	exit statement
syntax: `break;`	`exit(int error_code):`
`break` is a keyword.	the `exit` is a standard library function
it is used to exiting from other control statements. (terminate control statement, not the program)	it is used to exiting from the program. (terminates the entire program).
terminating a control statement is always a successful execution.	it has two, success termination and failure termination of a program.
after break execution programs remain in the execution.	after exit execution program terminated.
we do not need any header file before using break.	before using `exit()` we have to include its header file, `stdlib` or `process` in the program

---

here is another example function with switch statement in C++ where we give a menu an exit option.