**Exceptions in Constructors and Destructors**

It is possible that exceptions might raise in a constructor or destructors. If an exception is raised in a constructor, memory might be allocated to some data members and might not be allocated for others. This might lead to memory leakage problem as the program stops and the memory for data members stays alive in the RAM.

Similarly, when an exception is raised in a destructor, memory might not be deallocated which may again lead to memory leakage problem. So, it is better to provide exception handling within the constructor and destructor to avoid such problems. Following program demonstrates handling exceptions in a constructor and destructor:

#include <iostream>

using namespace std;

class Divide

{

private:

int \*x;

int \*y;

public:

Divide()

{

x = new int();

y = new int();

cout<<"Enter two numbers: ";

cin>>\*x>>\*y;

try

{

if(\*y == 0)

{

throw \*x;

}

}

catch(int)

{

delete x;

delete y;

cout<<"Second number cannot be zero!"<<endl;

throw;

}

}

~Divide()

{

try

{

delete x;

delete y;

}

catch(...)

{

cout<<"Error while deallocating memory"<<endl;

}

}

float division()

{

return (float)\*x / \*y;

}

};

int main()

{

try

{

Divide d;

float res = d.division();

cout<<"Result of division is: "<<res;

}

catch(...)

{

cout<<"Unkown exception!"<<endl;

}

return 0;

}

**Un-caught Exceptions**

#include <exception>

#include <iostream>

using namespace std;

//defining custom terminator

void myhandler(){

   cout << "Inside new terminate handler\n";

   abort();

}

int main(){

   set\_terminate(myhandler);

   try {

      cout << "Inside try block\n";

      throw 100;

   }

   catch (char a){

      cout << "Inside catch block\n";

   }

   return 0;

}

**Link for pre-defined exceptions**

https://www.tutorialspoint.com/cplusplus/cpp\_exceptions\_handling.htm