**Exceptions And Error Handling**

**Writer: Sayak Haldar**

**1. To where does the program control transfers when exception arises?**

a) catch

b) handlers

c) throw

d) none of the mentioned

**Answer: b**

Explanation: When a exception arises that means the exception is caught by handlers and then it decides the type of exception.

**2. Which key word is used to check exception in the block of code?**

a) catch

b) throw

c) try

d) none of the mentioned

**Answer: c**

Explanation: The try() statement is used for exceptions in c++.

**3. What will happen when the exception is not caught in the program?**

a) error

b) program will execute

c) block of that code will not execute

d) none of the mentioned

**Answer: a**

Explanation: None.

**4. What is the output of this program?**

#include <iostream>

using namespace std;

int main()

{

int age = 0;

try

{

if (age < 0)

{

throw "Positive Number Required";

}

cout << age;

}

catch(const char \*Message)

{

cout << "Error: " << Message;

}

return 0;

}

a) 0

b) error:Positive Number Required

c) compile time error

d) none of the mentioned

View Answer

**Answer: a**

Since, age is 0. No exception will be thrown.

**5. What is the output of this program?**

#include <iostream>

using namespace std;

void PrintSequence(int StopNum)

{

int Num;

Num = 1;

while (true)

{

if (Num >= StopNum)

throw Num;

cout << Num;

Num++;

}

}

int main(void)

{

try

{

PrintSequence(20);

}

catch(int ExNum)

{

cout << "Caught an exception with value: " << ExNum;

}

return 0;

}

a) compile time error

b) prints first 19 numbers

c) prints first 19 numbers and throws exception at 20

d) none of the mentioned

**Answer: c**

Explanation: In this program, we are printing upto 19 numbers and when executing the 20, we are raising a exception.

**6. What is the output of this program?**

#include <iostream>

using namespace std;

double division(int a, int b)

{

if (b == 0)

{

throw "Division by zero condition!";

}

return (a / b);

}

int main ()

{

int x = 50;

int y = 2;

double z = 0;

try

{

z = division(x, y);

cout << z;

}

catch(const char \*msg)

{

cerr << msg;

}

return 0;

}

a) 25

b) 20

c) Division by zero condition!

d) None of the mentioned

**Answer: a**

We are dividing by 2.

**7. What is the output of this program?**

#include <iostream>

using namespace std;

int main()

{

char\* buff;

try

{

buff = new char[1024];

if (buff == 0)

throw "Memory allocation failure!";

else

cout << sizeof(buff) << "Byte successfully allocated!"<<endl;

}

catch(char \*strg)

{

cout<<"Exception raised: "<<strg<<endl;

}

return 0;

}

a) 4 Bytes allocated successfully

b) 8 Bytes allocated successfully

c) Memory allocation failure

d) Depends on the size of data type

**Answer: d**

Explanation: As we are allocating the memory to the variables and if there is not sufficient size means, it will throw an exception.

Now, since, this is dynamic memory allocation, there is a better way to know if the requested memory is allocated fully. That is catching std::bad\_alloc.

**8. What is the output of this program?**

#include <iostream>

using namespace std;

void Funct();

int main()

{

try

{

Funct();

}

catch(double)

{

cerr << "caught a double type..." << endl;

}

return 0;

}

void Funct()

{

throw 3;

}

a) caught a double type

b) compile time error

c) abnormal program termination

d) none of the mentioned

**Answer: c**

Explanation: As we are throwing integer to double it will raise as abnormal program after termination throw statement.

**In runtime, the following thing would happen:**

terminate called after throwing an instance of ‘int’

Aborted

**9. What is the output of this program?**

#include <iostream>

#include <exception>

using namespace std;

int main()

{

try

{

int \* array1 = new int[100000000];

int \* array2 = new int[100000000];

int \* array3 = new int[100000000];

int \* array4 = new int[100000000];

cout << "Allocated successfully";

}

catch(bad\_alloc&)

{

cout << "Error allocating the requested memory." << endl;

}

return 0;

}

a) Allocated successfully

b) Error allocating the requested memory

c) Depends on the memory of the computer

d) None of the mentioned

**Answer: c**

**10. What will happen when the handler is not found for exception?**

a) calls the standard library function terminate()

b) raise an error

c) executes the remaining block

d) none of the mentioned

**Answer: a**

**11. How many types of exception handling are there in c++?**

a) 1

b) 2

c) 3

d) 4

**Answer: b**

Explanation: There are two types of exception handling in c++. They are synchronous exception handling and asynchronous exception handling.

**12. How many runtime error messages associated with exception?**

a) 2

b) 4

c) 5

d) infinite

**Answer: b**

Explanation: There are four runtime error messages associated with exceptions.They are **overflow\_error, range\_error, system\_error** and **underflow\_error**.

**13. Which block should be placed after try block?**

a) catch

b) throw

c) either catch or throw

d) none of the mentioned

**Answer: a**

**14. What is the output of this program?**

#include <iostream>

using namespace std;

int main()

{

double a = 10, b = 5, res;

char Operator = '/';

try

{

if (b == 0)

throw "Division by zero not allowed";

res = a / b;

cout << a << " / " << b << " = " << res;

}

catch(const char\* Str)

{

cout << "\n Bad Operator: " << Str;

}

return 0;

}

a) 10

b) 2

c) Bad Operator

d) 10 / 5 = 2

**Answer: d.**

we are dividing by 5.

However, if we divide by 0, It would print:

**Bad Operator: Division by zero not allowed.**

**15. What is the output of this program?**

#include <iostream>

using namespace std;

int main()

{

try

{

throw 1;

}

catch (int a)

{

cout << "exception number: " << a << endl;

return 0;

}

cout << "No exception " << endl;

return 0;

}

a) No exception

b) exception number

c) exception number: 1

d) none of the mentioned

View Answer

Answer: c

**16. What is the output of this program?**

#include <iostream>

using namespace std;

int main()

{

int a = 10, b = 20, c = 30;

float d;

try

{

if ((a - b) != 0)

{

d = c / (a - b);

cout << d;

}

else

{

throw(a - b);

}

}

catch (int i)

{

cout<<"Answer is infinite "<<i;

}

}

a) 10

b) -3

c) 15

d) none of the mentioned

Answer: b

**17. What is the output of this program?**

#include <iostream>

using namespace std;

void test(int x)

{

try

{

if (x > 0)

throw x;

else

throw 'x';

}

catch(int x)

{

cout<<"integer:"<<x;

}

catch(char x)

{

cout << "character:" << x;

}

}

int main()

{

test(10);

test(0);

}

a) integer:10character:0

b) integer:10

c) character:0

d) none of the mentioned

**Answer: a**

Explanation: We are passing the integer and character and catching it by using multiple catch statement.

**18. What is the output of this program?**

#include <iostream>

using namespace std;

void PrintSequence(int StopNum)

{

int Num;

Num = 1;

while (true)

{

if (Num >= StopNum)

throw Num;

cout << Num << endl;

Num++;

}

}

int main(void)

{

try

{

PrintSequence(2);

}

catch(int ExNum)

{

cout << "exception: " << ExNum << endl;

}

return 0;

}

a) 1

b) exception: 2

c) 1

exception: 2

d) none of the mentioned

**exception: 2**

**19. Pick out the correct Answer.**

a) Exceptions are not suitable for critical points in code

b) Exception are suitable for critical points in code

c) All of the mentioned

d) None of the mentioned

**Answer) a)**

Explanation: If there is many number of exceptions in the program means, We have to use multiple catch statement and it is hard to keep track of the program.

**20. When exceptions are used?**

a) To preserve the program

b) Exceptions are used when postconditions of a function cannot be satisfied

c) Exceptions are used when postconditions of a function can be satisfied

d) None of the mentioned

**Answer: c**

**21. How many parameters does the throw expression can have?**

a) 1

b) 2

c) 3

d) 4

Answer: a

**22. Where exception are handled?**

a) inside the program

b) outside the regular code

c) both inside or outside

d) none of the mentioned

Answer: b

**23. Which is used to check the error in the block?**

a) try

b) throw

c) catch

d) none of the mentioned

**Answer: a**

Explanation: The try block is used to check for errors, if there is any error means, it can throw it to catch block.

**24. What is the output of this program?**

#include <iostream>

#include <exception>

using namespace std;

class myexception: public exception

{

virtual const char\* what() const throw()

{

return "exception arised";

}

} myex;

int main ()

{

try

{

throw myex;

}

catch (exception& e)

{

cout << e.what() << endl;

}

return 0;

}

a) exception arised

b) error

c) exception

d) runtime error

**Answer: a**

Explanation: In this program, We are arising a standard exception and catching that and returning a statement.

**25. What is the output of this program?**

#include <iostream>

using namespace std;

int main()

{

int age=5;

try

{

if (age < 0)

throw "Positive Number Required";

cout << age << "\n\n";

}

catch(const char\* Message)

{

cout << "Error: " << Message;

}

return 0;

}

a) 5

b) 10

c) 15

d) Positive Number Required

**Answer: a**

Explanation: In this program, We are checking the age of a person, If it is zero means, We will arise a exception. However, the age is 5 here.

**26. What is the output of this program?**

#include <iostream>

using namespace std;

double division(int a, int b)

{

if ( b == 0 )

{

throw "Division by zero condition!";

}

return (a / b);

}

int main ()

{

int x = 50;

int y = 0;

double z = 0;

try

{

z = division(x, y);

cout << z << endl;

}

catch (const char\* msg)

{

cout << msg << endl;

}

return 0;

}

a) 50

b) 0

c) Division by zero condition!

d) None of the mentioned

**Answer: c**

Explanation: We are dividing the values and if one of the values is zero means, We are arising an exception.

**27. What is the output of this program?**

#include <iostream>

#include <string>

using namespace std;

int main()

{

double Op1 = 10, Op2 = 5, Res;

char Op;

try

{

if (Op != '+' && Op != '-' && Op != '\*' && Op != '/')

throw Op;

switch(Op)

{

case '+':

Res = Op1 + Op2;

break;

case '-':

Res = Op1 - Op2;

break;

case '\*':

Res = Op1 \* Op2;

break;

case '/':

Res = Op1 / Op2;

break;

}

cout << "\n" << Op1 << " " << Op << " "<< Op2 << " = " << Res;

}

catch (const char n)

{

cout << n << " is not a valid operator";

}

return 0;

}

a) 15

b) 5

c) 2

d) is not a valid operator

**Answer: d**

Explanation: It will arise a exception because we missed a operator.

**28. What is the output of this program?**

#include<iostream>

#include "math.h"

using namespace std;

double MySqrt(double d)

{

if (d < 0.0)

throw "Cannot take sqrt of negative number";

return sqrt(d);

}

int main()

{

double d = 5;

cout << MySqrt(d) << endl;

}

a) 5

b) 2.236

c) Error

d) Cannot take sqrt of negative number

**Answer: b**

Explanation: We are finding the square root of the number, if it is a positive number, it can manipulate, Otherwise it will arise a exception.

**29. How to handle the exception in constructor?**

a) We have to throw an exception

b) We have to return the exception

c) We have to throw an exception & return the exception

d) none of the mentioned

**Answer: a**

Explanation: As a constructor don’t have a return type, We have to throw the exception.

**30. What should present when throwing a object?**

a) constructor

b) copy-constructor

c) destructor

d) none of the mentioned

**Answer) b)**

**31.How do define the user-defined exceptions?**

a) inheriting and overriding exception class functionality

b) overriding class functioality

c) inheriting class functionality

d) none of the mentioned

**Answer: a**

**32. Which exception is thrown by dynamic\_cast?**

a) bad\_cast

b) bad\_typeid

c) bad\_exception

d) bad\_alloc

**Answer: a**

Explanation: bad\_cast exception is thrown by dynamic\_cast.

**33.Which type of program is recommended to include in try block?**

a) static memory allocation

b) dynamic memory allocation

c) const reference

d) pointer

**Answer: b**

Explanation: While during dynamic memory allocation, Your system may not have sufficient resources to handle it, So it is better to use it inside the try block.

**34.Which statement is used to catch all types of exceptions?**

a) catch()

b) catch(Test t)

c) catch(…)

d) none of the mentioned

**Answer: c**

Explanation: This catch statement will catch all types of exceptions that arises in the program.

1. **What is the output of this program?**

#include <iostream>

#include <typeinfo>

using namespace std;

class Polymorphic {virtual void Member(){}};

int main ()

{

try

{

Polymorphic \* pb = 0;

typeid(\*pb);

}

catch (exception& e)

{

cerr << "exception caught: " << e.what() << endl;

}

return 0;

}

a) exception caught: std::bad\_typeid

b) exception caught: std::bad\_alloc

c) exception caught: std::bad\_cast

d) none of the mentioned

Now, typeid is used where the dynamic type of a polymorphic object must be known and for static type identification.

**36.How to handle error in the destructor?**

a) throwing

b) terminate

c) both throwing & terminate

d) none of the mentioned

Answer) b) terminate.

**Answer)**Now, throwing exceptions from destructor is illegal. (compilation error)

But, we can call **std::terminate()** from destructor.

**37. What is meant by exception specification?**

a) A function is limited to throwing only a specified list of exceptions

b) A catch can catch all types of exceptions

c) A function can throw any type of exceptions

d) None of the mentioned

**Answer: a**

**Explanation:** C++ provides a mechanism to ensure that a given function is limited to throwing only a specified list of exceptions. It is called as exception specification.

**Specifying Data Type Of An Exception:**

type function(arg-list) throw (type-list)

{

}

Type list will be comma separated.

Consider the following example:

void test(int x)throw (int,double)

{

If(x==0) throw ‘x’;

else

{

if(x==1) throw x;

else

{

if(x==-1) throw 1.0;

}

}

cout<<”End of function block\n”;

}

**38.What do you mean by “No exception specification”?**  
a) It throws nothing  
b) It can throw anything  
c) It can catch anything  
d) None of the mentioned

**Answer) b) It can throw anything.**

**39.Which operations don’t throw anything?**

a) Operations which are reversible

b) Operations which are irreversible

c) Operations which are static

d) Operations which are dynamic

**Answer) b) Operations which are irreversible**

**40.Which operator is used in catch-all handler?**

a) ellipses operator

b) ternary operator

c) string operator

d) unary operator

**Answer: a**

Explanation: The ellipses operator can be represented as (…).

catch(…)

**41.What is the output of this program?**

#include <iostream>

using namespace std;

class Base

{

protected:

int a;

public:

Base()

{

a = 34;

}

Base(int i)

{

a = i;

}

virtual ~Base()

{

if (a < 0) throw a;

}

virtual int getA()

{

if (a < 0)

{

throw a;

}

}

};

int main()

{

try

{

Base b(-25);

cout << endl << b.getA();

}

catch (int)

{

cout << endl << "Illegal initialization";

}

}

a) Illegal initialization

b) Terminate called after throwing an instance of ‘int’

c) Illegal initialization & terminate called after throwing an instance

d) None of the mentioned

**Answer: b**

Explanation: As we are throwing an negative number and we are using only integer, So it is arising an error.

Concept of calling terminate is based on the mismatch of exception thrown and exception caught. Because, that necessarily means one thing: that is one exception is thrown **which is not caught.**

**#include<cstdio>**

**using namespace std;**

**void func(double a)**

**{**

**if(a==10.000000)**

**{**

**throw a;**

**}**

**}**

**int main()**

**{**

**try**

**{**

**func(10);**

**}**

**catch(int a)**

**{**

**printf("The integer value thrown an exception is %d\n",a);**

**}**

**return 0;**

**}**

This will generate the same kind of error message.

**terminate called after throwing an instance of 'double'**

**Aborted**

**42.What is the output of this program?**

#include <iostream>

#include <exception>

using namespace std;

void terminator()

{

cout << "terminate" << endl;

}

void (\*old\_terminate)() = set\_terminate(terminator);

class Botch

{

public:

class Fruit {};

void f()

{

cout << "one" << endl;

throw Fruit();

}

~Botch()

{

throw 'c';

}

};

int main()

{

try

{

Botch b;

b.f();

}

catch(...)

{

cout << "inside catch(...)" << endl;

}

}

a) one

b) inside catch

c) one

terminate

d) one

terminate

Aborted

**Answer: d**

Explanation: This program will call terminate function as it is having an uncaught exception. Now, here, in this case, a user defined terminate function is called **(“terminator”) ,** That function is set as terminate function by using set\_terminate.

**43.What is the output of the following code?** #include <iostream>

#include <exception>

#include <cstdlib>

using namespace std;

void myterminate ()

{

cerr << "terminate handler called";

abort();

}

int main (void)

{

set\_terminate (myterminate);

throw 0;

return 0;

}

a) terminate handler called

b) aborted

c) both terminate handler & Aborted

d) none of the mentioned

**Answer) c)**

Because, main throws an exception which is uncaught.

**44.What function will be called when we have a uncaught exception?**a) catch  
b) throw  
c) terminate  
d) none of the mentioned

**Answer) c)**

**45.What will not be called when the terminate() arises in constructor?**

a) main()

b) class

c) destructor

d) none of the mentioned

Answer) c) destructor.

**46. What will happen when we move try block far away from catch block?**a) Reduces the amount of code in cache  
b) Increases the amount of code in cache  
c) Don’t alter anything  
d) None of the mentioned

**Answer: a**  
Explanation: compilers may try to move the catch-code far away from the try-code, which reduces the amount of code to keep in cache normally, thus enhancing performance.

**47.What will happen if an exception that is thrown may causes a whole load of objects to go out of scope?**

a) Terminate the program

b) Produce a runtime error

c) It will be added to the overhead

d) None of the mentioned

**Answer: c**

**48.What operation can be performed by destructor?**

a) Abort the program

b) Resource cleanup

c) Exit from the current block

d) None of the mentioned

**Answer: b**

Explanation: It will be used to free all the resources that are used by block of code during execution.

**49.What is the main purpose of the constructor?**

a) Begin the execution of the class

b) Include the macros for the program

c) Establish the class invariant

d) None of the mentioned

**Answer: c**

**Explanation:** The purpose of a constructor is to establish the class invariant. To do that, it often needs to acquire system resources or in general perform an operation that may fail.

(class invariant is a class state)

**50.Which alternative can replace the throw statement?**

a) for

b) break

c) return

d) exit

**Answer: c**

Explanation: throw and return does the same job like return a value. So it can be replaced.

**51. What are the disadvantages if use return keyword to return error codes?**

a) You have to handle all exceptional cases explicitly

b) Your code size increases dramatically

c) The code becomes more difficult to read

d) All of the mentioned

**Answer: d**

Explanation: As we are using return for each and every exception, It will definitely increase the code size.

**52.What is most suitable for returning the logical errors in the program?**

a) Use contructor and destructor

b) Set a global error indicator

c) Use break keyword

d) None of the mentioned

**Answer) b) Set a global error indicator.**

errno.h is present in c and c++

Also, the errno provided is thread safe. On Linux, the global errno variable is thread-specific. POSIX requires that errno be threadsafe. In POSIX.1, errno is defined as an external global variable.Nov 8, 2009

**53.How many levels are there in exception safety?**

a) 1

b) 2

c) 3

d) 4

**Answer: c**

Explanation: The three levels of exception safety are basic, strong and nothrow.

**Nothrow:**

This constant value is used as an argument for operator new and operator new[] to indicate that these functions shall not throw an exception on failure, but return a null pointer instead.

By default, when the new operator is used to attempt to allocate memory and the handling function is unable to do so, a bad\_alloc exception is thrown. But when nothrow is used as argument for new, it returns a null pointer instead.

**// nothrow example**

**#include <iostream> // std::cout**

**#include <new> // std::nothrow**

**int main () {**

**std::cout << "Attempting to allocate 1 MiB... ";**

**char\* p = new (std::nothrow) char [1048576];**

**if (!p) { // null pointers are implicitly converted to false**

**std::cout << "Failed!\n";**

**}**

**else {**

**std::cout << "Succeeded!\n";**

**delete[] p;**

**}**

**return 0;**

**}**

**54. Pick out the correct statement for error handling alternatives.**

a) Terminate the program

b) Use the stack

c) Exit from the block

d) None of the mentioned

View Answer

**Answer: b**

Explanation: When an error arises, it will be pushed into stack and it can be corrected later by the programmer.

**55. What will happen when an exception is not processed?**

a) It will eat up lot of memory and program size

b) Terminate the program

c) Crash the compiler

d) None of the mentioned

**Answer: a**

Explanation: As in the case of not using an exception, it will remain useless in the program and increase the code complexity.

**56.Which header file is used to declare the standard exception?**

a) #include<exception>

b) #include<except>

c) #include<error>

d) none of the mentioned

Answer: a

**57. Where are standard exception classes grouped?**

a) namespace std

b) error

c) catch

d) none of the mentioned

View Answer

**Explanation:** As these are standard exceptions, they need to be defined in the standard block, So it is defined under namespace std.

**58. How many types of standard exception are there in c++?**

a) 9

b) 5

c) 6

d) 7

**Answer: a**

Explanation: There are nine standard exceptions in c++. They are bad\_alloc, bad\_cast, bad\_exception, bad\_function\_call, bad\_typeid, bad\_weak\_ptr, ios\_base::failure, logic\_error and runtime\_error.