

**Ex.No:5.1**

## **Exception Handling**

**Aim:**

Develop a C++ program for handling Exceptions.

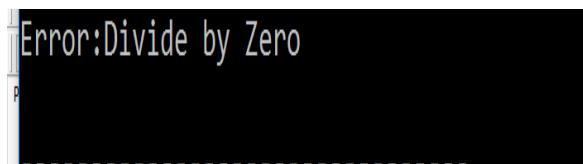
**Description:**

An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero. C++ exception handling is built upon three keywords: try, catch, and throw.

**Program:**

```
#include<iostream>
using namespace std;
int main ()
{
int x = 50;
int y = 0;
try
{
if( y == 0 )
throw "Divide by Zero";
else
cout<< (x/y);
}
catch (const char* msg)
{
cout <<"Error:"<<msg << endl;
}
return 0;
}
```

**Output:**



**Ex.No:5.2**

## **Exception Handling**

**Aim:**

Develop a C++ program to illustrate the use of multiple catch statements.

**Description:**

Multiple catch blocks are used when we have to catch a specific type of exception out of many possible type of exceptions i.e. an exception of type char or int or short or long etc

**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) {
        cerr << msg << endl;
    }

    return 0;
}
```

**Output:**

```
Division by zero condition!
```

**Ex.No:5.3**

**Aim:**

## **List and Vector**

Develop a C++ program to implement List, Vector and its Operations.

**Description:**

Both vector and list are sequential containers of C++ Standard Template Library. List stores elements at non contiguous memory location i.e. it internally uses a doubly linked list. Whereas, vector stores elements at contiguous memory locations like an array.

**Program:**

```
#include <iostream>
#include <list>
using namespace std;
int main()
{
    list<int> L;
    for(int i=1;i<=5;i++)
    {
        L.push_back(i);           // Insert a new element at the end
        L.push_front(i*5);
    }      // Insert a new element at the beginning
    L.insert(++L.begin(),2);   // Insert "2" after first node
    list<int>::iterator i;
    for(i=L.begin(); i != L.end(); ++i) cout << *i << " ";
    cout << endl;
    cout<<"After DELETION elements in list"<<endl;
    L.pop_front();
    L.pop_back();
    for(i=L.begin(); i != L.end(); ++i) cout << *i << " ";
    cout << endl;
    return 0;
}
```

**Output:**

```
25 20 15 10 5 1 2 3 4 5
After DELETION elements in list
2 20 15 10 5 1 2 3 4
```

## ii Vector and Vector Operations

```
#include <iostream>
using namespace std;
#include <vector>
#include <algorithm>
int main()
{
    int i;
    vector<int> ivect;
    for(i=1;i<=5;i++)
    {
        ivect.push_back(i); // input
    }
    vector<int>::iterator it;
    for (it=ivect.begin();it != ivect.end();++it )
    {
        cout << *it << " ";
    }
    cout << endl;
    ivect.insert(ivect.begin()+1,2,100);//insert two copies of 100 at second position
    cout<<"After Inserting element in list"<<endl;
    for ( it=ivect.begin();it != ivect.end();++it )
        cout << *it << " ";
    return 0;
}
```

### Output:

```
1 2 3 4 5
After Inserting element in list
1 100 100 2 3 4 5
```

**Ex.No:5.4**

## **Deque**

**Aim:**

Develop a C++ program to implement Deque and Deque Operations

**Description:**

Double ended queues are sequence containers with the feature of expansion and contraction on both the ends. They are similar to vectors, but are more efficient in case of insertion and deletion of elements at the end, and also the beginning. Unlike vectors, contiguous storage allocation may not be guaranteed

**Program:**

```
#include <iostream>
using namespace std;
#include <deque>
int main()
{
    deque<int> deq;
    cout<<"push_front() and push_back\n";
    for(int i=1; i<=5; ++i)
    {
        deq.push_front(i);// insert the elements each at the front
        deq.push_back(i*5);// insert the elements each at the end
    }
    deque <int>::iterator d;
    d=deq.begin();
    ++d;
    deq.insert (d,1,34);
    for(d=deq.begin(); d!=deq.end();d++)
        cout<<*d<<" ";
    deq.pop_front();// delete the elements from the front
    deq.pop_back();// delete the elements from the end
    for(d=deq.begin(); d!=deq.end();d++)
        cout<<*d<<" ";
    cout<<endl;
}
```

**Output:**

```
push_front()
5 34 4 3 2 1 5 10 15 20 25
34 4 3 2 1 5 10 15 20
```

**Ex.No:5.5**

## **Map**

**Aim:**

Develop a C++ program to implement Map and Map Operations.

**Description:**

Maps are associative containers that store elements in a mapped fashion. Each element has a key value and a mapped value. No two mapped values can have same key values.

**Functions associated with Map:**

begin() – Returns an iterator to the first element in the map  
end() – Returns an iterator to the theoretical element that follows last element in the map  
size() – Returns the number of elements in the map  
max\_size() – Returns the maximum number of elements that the map can hold  
empty() – Returns whether the map is empty  
pair insert(keyvalue,mapvalue) – Adds a new element to the map  
erase(iterator position) – Removes the element at the position pointed by the iterator  
erase(const g)- Removes the key value ‘g’ from the map  
clear() – Removes all the elements from the map

**Program:**

```
#include <iostream>
#include <map>
#include <string>
using namespace std;
int main()
{
    // type of the collection
    map<int, string> mp;
    // set container for int/string values insert some elements in arbitrary order
    mp.insert(make_pair(2805,"JAMES"));
    mp.insert(make_pair(2802,"SMITH"));
    mp.insert(make_pair(2801,"SCOTT"));
    mp.insert(make_pair(2807,"MILLER"));
    mp.insert(make_pair(2804,"JOY"));
    mp.insert(make_pair(2806,"JOHN"));
    mp.insert(make_pair(2801,"SAM"));
    mp.insert(make_pair(2803,"BOB"));
    // iterate over all elements and print, element member second is the value
    map<int, string>::iterator pos;
    cout<<"EMPID"<<"\t"<<"NAME"<<endl;
    for(pos = mp.begin(); pos != mp.end(); ++pos)
```

```
cout<<pos->first<<" "<<pos->second<<endl;
pos=mp.find(2803);      //finding element at key 2803
cout<<"Value at key 2803 is"<<pos->second<<endl;
mp.erase(2805);          //deleting value at key position 2805
cout<<"map after deletion"<<endl;
for(pos = mp.begin(); pos != mp.end(); ++pos)
cout<<pos->first<<" "<<pos->second<<endl;
return 0;
}
```

```
2801 SCOTT
2802 SMITH
2803 BOB
2804 JOY
2805 JAMES
2806 JOHN
2807 MILLER
Value at key 2803 isBOB
map after deletion
2801 SCOTT
2802 SMITH
2803 BOB
2804 JOY
2806 JOHN
2807 MILLER
```

**Additional Practice:****Exp.No:1****Develop a C++ program for flight booking system**

#include &lt;iostream&gt;

using namespace std;

```
void setRoute(char *path[], float *fare, float *tfare){  
    int i=0;  
    for(i=0;i<5;i++){  
        cout<<"Enter flight route:"<<endl;  
        cout<<"Route ["<<i+1<<"] : ";  
        path[i]=new char[100];  
        cin.ignore(1);  
        cin.getline(path[i],100);  
        cout<<"Enter fare: ";  
        cin>>fare[i];  
        tfare[i]=fare[i]+(fare[i]*19/100);  
    }  
}
```

```
void displayPath(char *path[],float *fare, float *tfare){  
    int i=0;  
    cout<<"Available flight routes are:"<<endl;  
    for(i=0;i<5;i++){  
        cout<<"Route ["<<i+1<<"] : "<<path[i]  
        <<" - Fare: "<<fare[i]<<,Total Fare: "<<tfare[i]<<endl;  
    }  
}
```

```
int main(){  
    //variable to store flight route  
    char *route[5];  
    float fare[5],totalFare[5];  
    char name[30],path[100];  
    int d,m,y;  
    int routeId;  
  
    setRoute(route,fare,totalFare);  
  
    cout<<endl<<endl;  
    cout<<"Enter name: ";  
    cin.ignore(1);  
    cin.getline(name,30);  
  
    cout<<"Enter date of travel (d m y): ";  
    cin>>d>>m>>y;
```

```

        displayPath(route,fare,totalFare);
        cout<<"Choose flight route (1 to 5): ";
        cin>>routeId;

        if(routeId<1 || routeId>5){
            cout<<"Invalid flight route!!!"<<endl;
            return 0;
        }

        cout<<endl<<endl;
        cout<<"Congratulations... "<<name<<" your ticket has been booked."<<endl;
        cout<<"Travel date is: "<<d<<"/"<<m<<"/"<<y<<endl;
        cout<<"Flight route: "<<route[routeId-1]<<endl;
        cout<<"Total fare is: "<<totalFare[routeId-1]<<endl;

        return 0;
    }
}

```

Output:

```

Enter flight route:
Route [2] : delhi
Enter fare: 2000
Enter flight route:
Route [3] : banglore
Enter fare: 1500
Enter flight route:
Route [4] : kochi
Enter fare: 2500
Enter flight route:
Route [5] : hyd
Enter fare: 1500

Enter name: Rinky
Enter date of travel (d m y): 22 10 2025
Available flight routes are:
Route [1] : oa - Fare: 3000,Total Fare: 3570
Route [2] : delhi - Fare: 2000,Total Fare: 2380
Route [3] : banglore - Fare: 1500,Total Fare: 1785
Route [4] : kochi - Fare: 2500,Total Fare: 2975
Route [5] : hyd - Fare: 1500,Total Fare: 1785
Choose flight route (1 to 5): 4

Congratulations... Rinky your ticket has been booked.
Travel date is: 22/10/2025
Flight route: kochi
Total fare is: 2975

-----
Process exited after 138.4 seconds with return value 0
Press any key to continue . . .

```

**Ex.No:4**

## **GUESSING WORD**

**Aim:**

Develop a C++ program with maximum of 20 characters, that your user will be guessed and will show only asterisks (\*) on the screen. The user will input or enter one character at a time. And for every correct character, the asterisk will be replaced by that character until all the characters of the mystery word/s will reveal. Your program will accept a maximum of three (3) errors or mistakes in entering/inputting character otherwise the mystery word/s will be viewed.

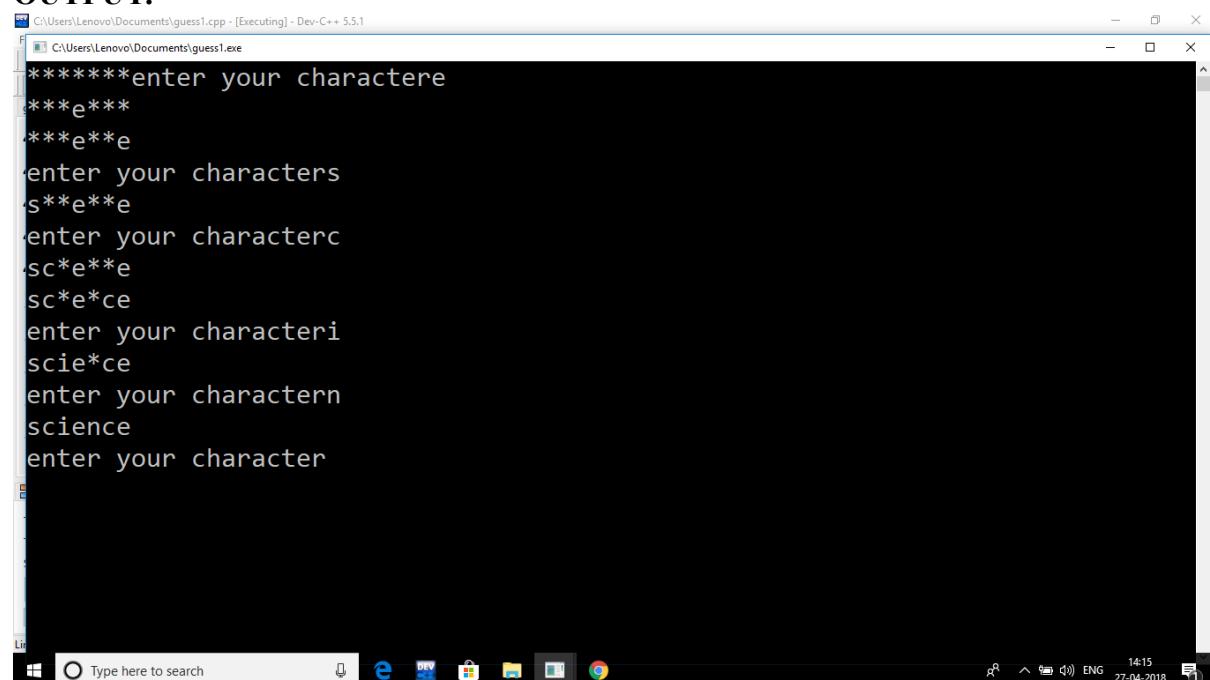
**Program:**

```
#include<iostream>
using namespace std;
class word
{
public:
    void guess()
    {
        int i=0, attempts=1, flag=0;
        char ch;
        char str[20]="science";
        char str1[20];
        for(i=0;str[i]!='\0';i++)
        {
            //str1[i]=str[i];
            str1[i]='*';
        }
        cout<<str1;
        while(attempts<=4)
        {
            cout<<"enter your character";
            cin>>ch;
            for(int i=0;str[i]!='\0';i++)
            {
                if(ch==str[i])
                {
                    str1[i]=ch;
                    cout<<str1<<"\n";
                    flag=1;
                }
            }
            if(flag==0)
            {
                cout<<"\nSorry! the character is not existing"<<endl;
                attempts++;
            }
        }
        //cout<<"you still have"<<attempts<<" chances"<<endl;
    }
}
```

```
if(attempts==4)
cout<<"sorry";
else
cout<<str1;
}
};

int main()
{
    word w;
    w.guess();
    return 0;
}
```

## OUTPUT:



```
*****enter your charactere
***e***
***e**e
enter your characters
s**e**e
enter your characterc
sc*e**e
sc*e*ce
enter your characteri
scie*ce
enter your charactern
science
enter your character
```