**Exception Handling**

**Exceptions:**

When executing C++ code, different errors can occur like coding errors made by the programmer or errors due to wrong input.

When an error occurs, C++ will normally stop and generate an error message.

**Exception handling in C++ consists of three keywords: try, throw and catch:**

1. **Try**

The try statement allows you to define a block of code to be tested for errors while it is being executed.

1. **Throw**

The throw keyword throws an exception when a problem is detected, which lets us create a custom error.

1. **Catch**

The catch statement allows you to define a block of code to be executed if an error occurs in the try block.

Try and catch block comes in pair. For example: if a person age less than 18 then try block test the code and identify error. and With the help of throw it will send the error or exception to catch block. Catch block handle this error.

By using exception handling concept, we can execute the part of code which is not under exception or which is stop due to occurring of exception.

**Syntax**

try {

// Block of code to try

throw exception; // Throw an exception when a problem arises

}

catch () {

// Block of code to handle errors

}

**Example**

#include <iostream>

using namespace std;

int main() {

try {

int age = 15;

if (age >= 18) {

cout << "Access granted - you are eligible for vote.";

} else {

throw age;

}

}

catch (int myNum) {

cout << "Access denied - You must be at least 18 years old.\n";

}

return 0;

}

**Type of Exceptions**

1. **Range Exception:**

Type of exception that is thrown when a function tries to generate a value that is out of range, such as using an invalid index to access an array.

**Example**

#include <iostream>

using namespace std;

int main() {

try {

// Code that may throw a range error exception

int array[5] = {6, 9, 3, 40, 15};

int index = 7; // Index out of range

if (index < 0 || index >= 5) {

throw index; // Throw a range error exception

}

int value = array[index]; // Accessing an element with a valid index

cout << "Value at index " << index << " is: " << value <<endl;

} catch (int e) {

cout << "Index is out of range"<<endl;

}

return 0;

}

1. **Division By Zero Exception**

Division by zero exceptions are those exceptions that happens when denominator value becomes zero.

**Example**

#include <iostream>

using namespace std;

int main() {

try {

// Code that may throw a division by zero error

int denominator = 0;

if (denominator == 0) {

throw denominator; //Throw a division by zero error

}

int result = 42 / denominator;

cout << "Result: " << result << endl;

} catch (int e) {

cout<< "Division by zero exception: "<<endl;

}

return 0;

}

1. **Invalid Argument Exception**

Invalid argument exceptions are those exceptions when an invalid value occur that does not follow the condition.

**Example**

#include <iostream>

using namespace std;

int main() {

int value = -10;

try {

if (value<0)

{

throw value;

}

}

catch (int e) {

cout<< "Caught an invalid argument exception " << endl;

}

return 0;

}

1. **Data Type Exception**

Invalid argument exceptions are those exceptions which occurs due to data type.

**Example**

#include <iostream>

using namespace std;

int main() {

int x;

float y;

try

{

throw x;

}

catch(int ex)

{

cout<<"int"<<endl;

}

try

{

throw y;

}

catch(float ex1)

{

cout<<"float"<<endl;

}

return 0;

}