C++ Exception Handling – Error

Exception Handling in C++ is a process to handle runtime errors. We perform exception handling so the normal flow of the application can be maintained even after runtime errors.

In C++, exception is an event or object which is thrown at runtime. All exceptions are derived from std::exception class. It is a runtime error which can be handled. If we don't handle the exception, it prints exception message and terminates the program.

|  |  |
| --- | --- |
| **Exception** | **Description** |
| std::exception | It is an exception and parent class of all standard C++ exceptions. |
| std::logic\_failure | It is an exception that can be detected by reading a code. |
| std::runtime\_error | It is an exception that cannot be detected by reading a code. |
| std::bad\_exception | It is used to handle the unexpected exceptions in a c++ program. |
| std::bad\_cast | This exception is generally be thrown by **dynamic\_cast.** |
| std::bad\_typeid | This exception is generally be thrown by **typeid.** |
| std::bad\_alloc | This exception is generally be thrown by **new.** |

C++ Exception Handling Keywords

In C++, we use 3 keywords to perform exception handling:

* try
* catch, and
* throw

#include <iostream>

**using** **namespace** std;

**float** division(**int** x, **int** y) {

**if**( y == 0 ) {

**throw** "Attempted to divide by zero!";

   }

**return** (x/y);

}

**int** main () {

**int** i = 25;

**int** j = 0;

**float** k = 0;

**try** {

      k = division(i, j);

      cout << k << endl;

   }**catch** (**const** **char**\* e) {

      cerr << e << endl;

   }

**return** 0;

}

Output ->

Attempted to divide by zero!

C++ User-Defined Exceptions

The new exception can be defined by overriding and inheriting **exception** class functionality.

#include <iostream>

#include <exception>

using namespace std;

class MyException : public exception{

public:

const char \* what() const throw()

{

return "Attempted to divide by zero!\n";

}

};

int main()

{

try

{

int x, y;

cout << "Enter the two numbers : \n";

cin >> x >> y;

if (y == 0)

{

MyException z;

throw z;

}

else

{

cout << "x / y = " << x/y << endl;

}

}

catch(exception& e)

{

cout << e.what();

}

}

Output ->

Enter the two numbers :

10

2

x / y = 5

Enter the two numbers :

10

0

Attempted to divide by zero!

C++ Files

The fstream library allows us to work with files.

To use the fstream library, include both the standard <iostream> **AND** the <fstream> header file:

#include <iostream>

#include <fstream>

ofstream Creates and writes to files

ifstream Reads from files

fstream A combination of ofstream and ifstream: creates, reads, and writes to files

#include <iostream>

#include <fstream>

using namespace std;

int main() {

// Create and open a text file

ofstream MyFile(".txt");

// Write to the file

MyFile << "My name is Rajeev and I am working in Indra Academy";

// Close the file

MyFile.close();

}

// Create a text string, which is used to output the text file

string myText;

// Read from the text file

ifstream MyReadFile("filename.txt");

while (getline (MyReadFile, myText)) {

// Output the text from the file

cout << myText;

}

// Close the file

MyReadFile.close();

#include <iostream>

using namespace std;

int main () {

int var1;

char var2[10];

cout << "Address of var1 variable: ";

cout << &var1 << endl;

cout << "Address of var2 variable: ";

cout << &var2 << endl;

return 0;

}

C++ Pointers vs Arrays

#include <iostream>

using namespace std;

const int MAX = 3;

int main () {

int var[MAX] = {10, 100, 200};

int \*ptr;

ptr = var;

for (int i = 0; i < MAX; i++) {

cout << "Address of var[" << i << "] = ";

cout << ptr << endl;

cout << "Value of var[" << i << "] = ";

cout << \*ptr << endl;

// point to the next location

ptr++;

}

Return 0;

}

Passing Pointers to Functions in C++

#include <iostream>

using namespace std;

void test(int\*, int\*);

int main() {

int a = 5, b = 5;

cout << "Before changing:" << endl;

cout << "a = " << a << endl;

cout << "b = " << b << endl;

test(&a, &b);

cout << "\nAfter changing" << endl;

cout << "a = " << a << endl;

cout << "b = " << b << endl;

return 0;

}

void test(int\* n1, int\* n2) {

\*n1 = 10;

\*n2 = 11;

}

Pass by value, from Main to function only – value parameter

Pass by reference (address), Main to function, Function to main both – variable parameter

Swap 2 variables using pointer and functions

#include<iostream>

using namespace std;

void swap (int &num1, int &num2) //&num1 and &bnum2 are Reference variables

{

int temp;

temp=num1;

num1=num2;

num2=temp;

}

int main()

{

int a=5,b=10;

cout<<"\n Before swapping"<<"\n A = "<<a<<"\n B = "<<b<<endl;

swap(a,b);

cout<<"\n After swapping"<<"\n A = "<<a<<"\n B = "<<b<<endl;

return 0;

}

|  |
| --- |
|  |

Then make graphics project

Copy paste program with output

Make one MS-word file

Top – Write your name, class

Paste all pictures

Only 1 file

File name must be your name.

Byamba CPP.doc