Exception in C++

⇒ Exception Handling provides a simpler and reliable way to:

1) Report errors as exceptional events
2) Transfer Control to an exception handler.
   ↳ Along with information about the event

⇒ It employs three keywords:

① throw
② catch
③ try

```
try {
  - - -
}
catch ( ) {
  - - -
}
```

★ Throwing

⇒ A throw-expression generally has the form:

```
throw expression;
```

⇒ The expression's value conveys information about the nature of the exception.
   ↳ You can throw objects of primitive types, but it's generally preferable to throw object of class types.

★ Exception Classes

⇒ Various standard C++ language and library components report errors by throwing an exception.

⇒ They throw only objects of types derived from a base class called exception, defined in std header <exception>

⇒ The Stadard header <stdexcept> defines classes such as invalid_argument, out_of_range, and overflow_error.

↳ All are derived from exception

Stack unwinding

→ objects on the stack are distroyed

↳ Destruction happens in reverse order.

⇒ Unhandled exceptions result in a call to
std::terminate()

→ No stack unwinding

→ No destructor are called

↳ Resources are potentially leaked

So you should at least have:

```
try{
}
Catch(...){ //Catch-all handler
   /*...*/
}
```

★ When to Use Exceptions

① Use exceptions for errors that are expected to occur rarely.

{As there is big performance overhead}

② Exceptional Cases that Cannot be dealt with locally.

③ Dont use exception for things that should never happen.

★ How to use Exceptions

1) Build on the std::exception hierarchy
2) throw by value
3) Catch by reference