Chapter - 22
Exceptions

Exceptions

Exceptions are emergency procedures

We've already seen exceptions caused by "divide by 0" or "segmentation violation." The way our program handled these exceptions is to die.

We can generate our own exceptions and our own handling of them.

Exception Description

- A description of a possible problem.
- A section of code in which the exception may occur, which is enclosed in a **try** statement.
- Something that causes an exception and triggers the emergency procedures through a **throw** statement.
- Exception handling code inside a **catch** block.

Exception Description

- A description of a possible problem.

```
class fire_emergency {
        public:
  // Which engine is on fire
        int engine;

    // Other information
      // about the fire
        };
```

Exception Description

- A section of code in which the exception may occur, which is enclosed in a **try** statement.

```
try {
      fly_from_point_a_to_point_b();
}
```

Exception Description

- Something that causes an exception and triggers the emergency procedures through a **throw** statement.

```
// Watch for fire in engine #2
void sensor_2(void) {
    while (engine_running()) {
        if (engine_on_fire()) {
            fire_emergency fire_info;

            fire_info.engine = 2;
            throw(fire_info);
        }
    }
}
```

Exception Description

- Exception handling code inside a **catch** block.

```
catch (fire_emergency &fire_info) {
    active_extinguisher(fire_info.engine);
    turn_off(fire_info.engine)
    land_at_next_airport();
}
```

Stack Exceptions

The class describing the error:

```
class bound_err {
    public:
        std::string what;     // What caused the error


        // Initialize the bound error with a message
        bound_err(const std::string &i_what):what(i_what) {}


        // bound_err(bound_err) -- default copy constructor
        // ~ bound_err -- default destructor
};
```

The “try/catch” block

```
try {
    do_big_stack_operation();
};
catch (bound_err &what_happened) {
    std::cerr << "Error: Bounds exceeded\n";
    std::cerr << "Reason: " << what_happened.what << '\n';
}
```

Checking the bounds

```
inline void b_stack::push(const int item)
{
   if (count >= STACK_SIZE) {
      bound_err overflow("Push overflows stack");
      throw overflow;
   }
   stack::push(item);
}
```

This can be shortened:

```
inline void b_stack::push(const int item)
{
   if (count >= STACK_SIZE) {
      throw bound_err("Push overflows stack");
   }
   stack::push(item);
}
```

Run time library exceptions

The draft ANSI C++ standard defines many exceptions generated by the run time library.

Because the standard is so new, no one uses the standards.

Watch Out!