CPT106

C++ Programming and Software Engineering II

Lecture 11 Exception Handling

Dr. Xiaohui Zhu

Xiaohui.zhu@xjtlu.edu.cn

Office: SD535

Exception Definition

- An exception is an error condition, possibly outside the program's control, that prevents the program from continuing along its regular execution path.
- Errors are often divided into two categories: logic errors and runtime errors.

Logic and runtime errors

- Logic errors are caused by programming mistakes.
  - Index out of range
- Runtime errors that are beyond the control of programmer
  - Network service unavailable
  - Disk error

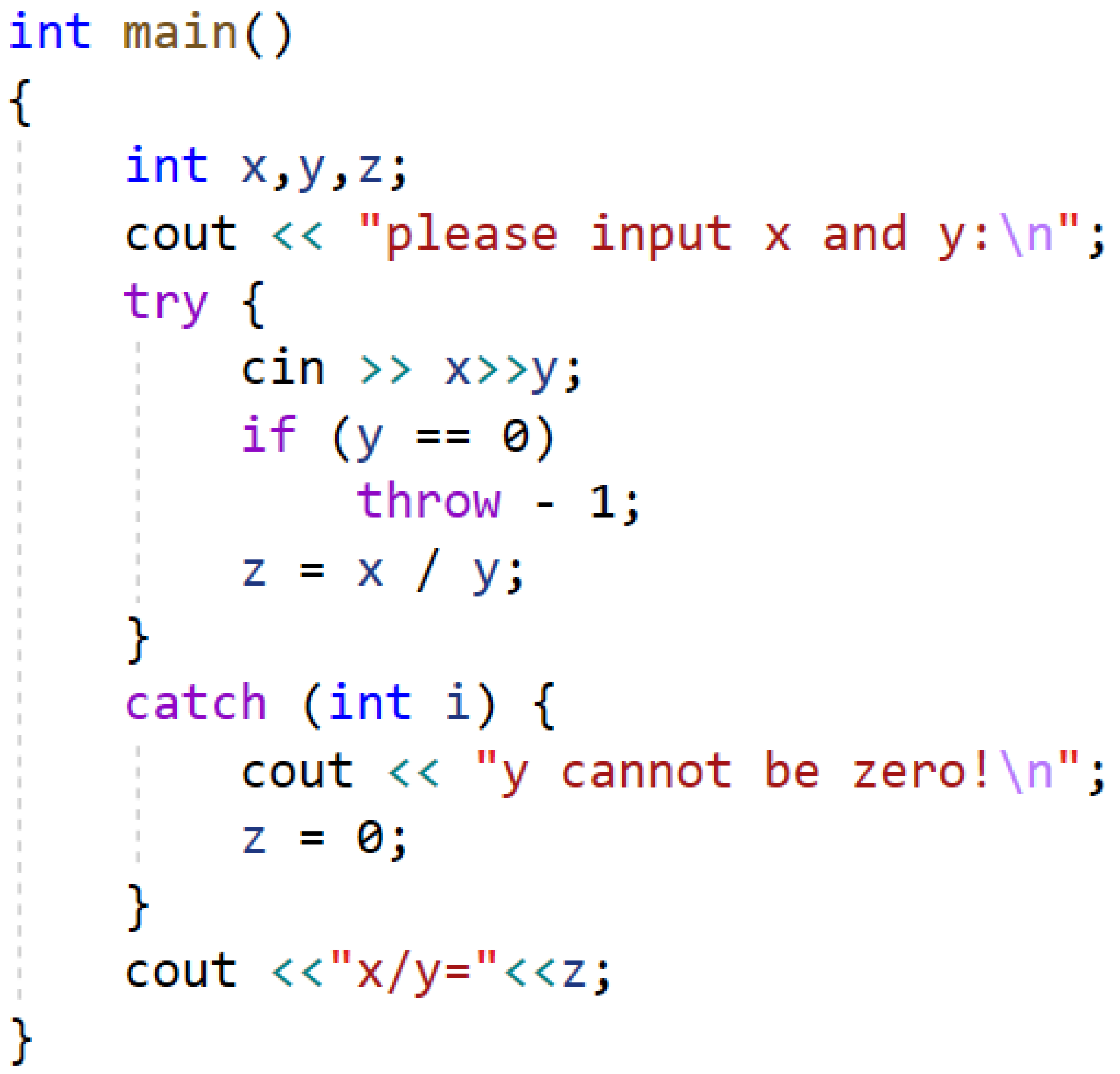
Exception and keywords

- Errors hinder normal execution of program. Exception handling is the process of handling errors and exceptions in such a way that they do not hinder normal execution of the system.
- In C++, Error handling is done using three keywords:
  - try
  - catch
  - throw

Exception syntax

```
try{
    … //codes that may throw exceptions.
     throw exceptions //throw exceptions explicitly
}
catch(exceptionName ex){
   … //codes that handle exceptions
}
```

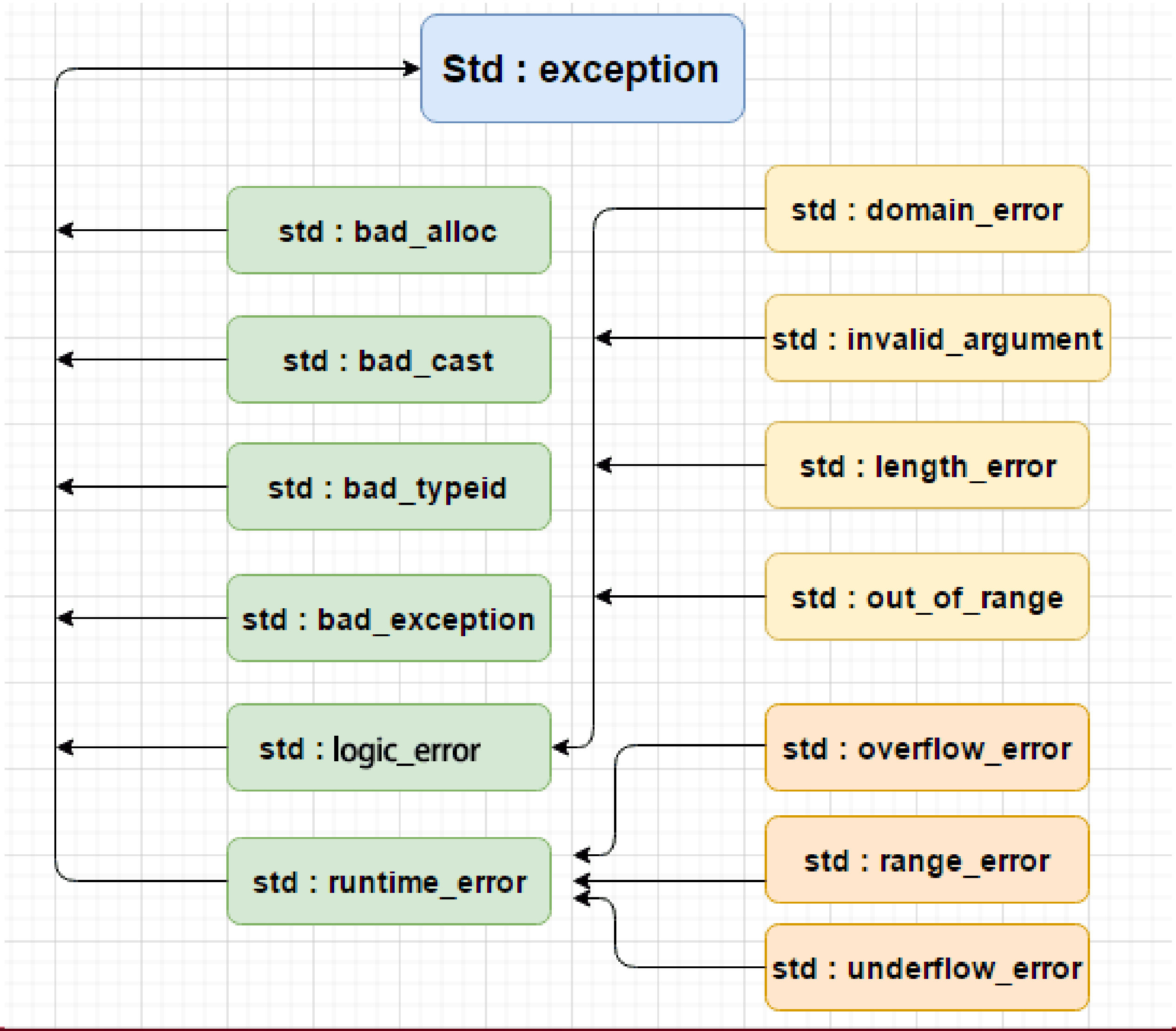
A simple exception example



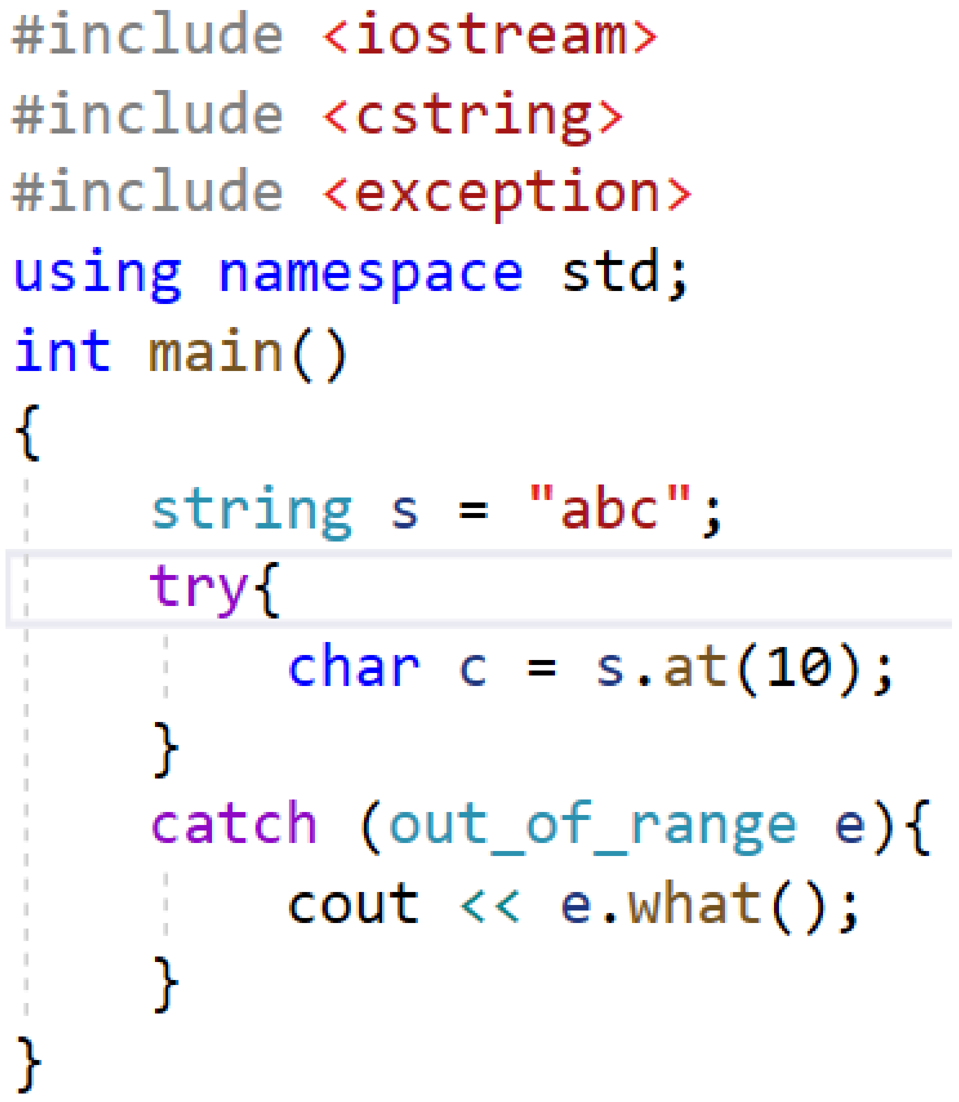
```
int main()
{
    int x,y,z;
    cout << "please input x and y:\n";
    try {
        cin >> x>>y;
        if (y == 0)
            throw - 1;
        z = x / y;
    }
    catch (int i) {
        cout << "y cannot be zero!\n";
        z = 0;
    }
    cout <<"x/y="<<z;
}
```

Denominator can not be zero

Standard exceptions

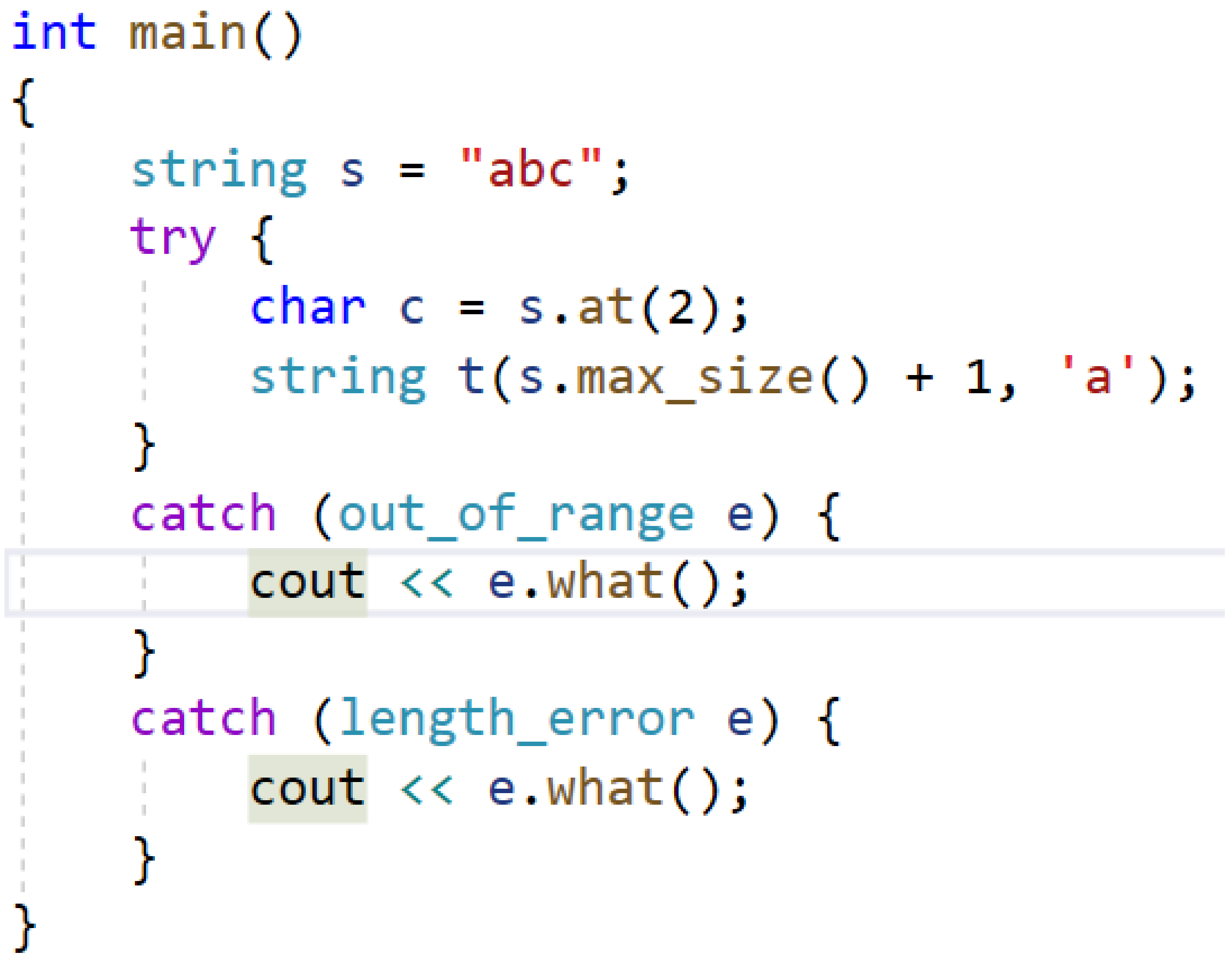


Example for standard exceptions



```
#include <iostream>
#include <cstring>
#include <exception>
using namespace std;
int main()
{
    string s = "abc";
    try{
        char c = s.at(10);
    }
    catch (out_of_range e){
        cout << e.what();
    }
}
```

Multiple try catch blocks



```
int main()
{
    string s = "abc";
    try {
        char c = s.at(2);
        string t(s.max_size() + 1, 'a');
    }
    catch (out_of_range e) {
        cout << e.what();
    }
    catch (length_error e) {
        cout << e.what();
    }
}
```

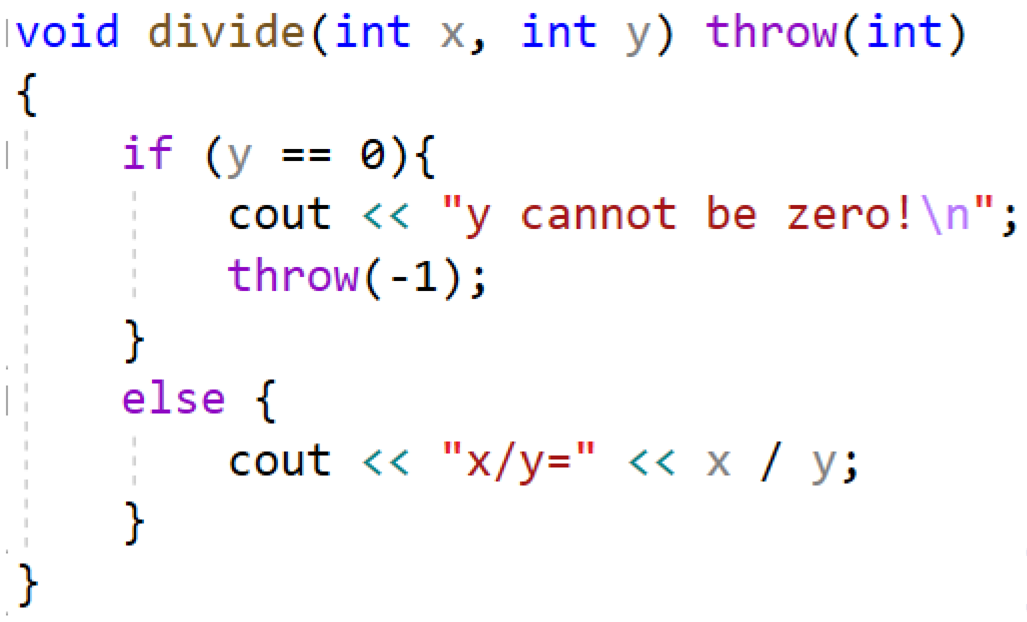
Exception specification in function

- Specify that a function may or may not exit by an exception by using an *exception specification*

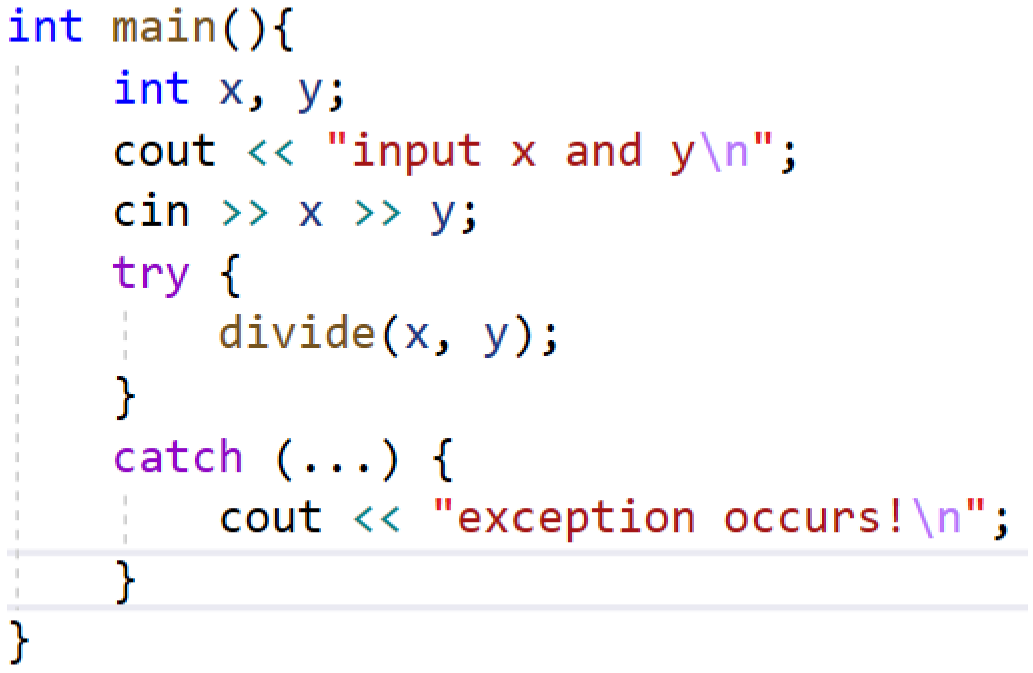
  Syntax:

  – *functionName (parameterList) throw()*

Exception specification in function



```cpp
void divide(int x, int y) throw(int)
{
    if (y == 0){
        cout << "y cannot be zero!\n";
        throw(-1);
    }
    else {
        cout << "x/y=" << x / y;
    }
}
```



```cpp
int main(){
    int x, y;
    cout << "input x and y\n";
    cin >> x >> y;
    try {
        divide(x, y);
    }
    catch (...) {
        cout << "exception occurs!\n";
    }
}
```

Define own exceptions

- We can define our own exceptions by inheriting and overriding **exception** class.

```
class MyException : public exception {
public:
    const char* what() const throw () {
        return "C++ Exception";
    }
};
```

About Assessment 4

- Group a team flexibly
- Each team has 4 members
- Email me if you cannot find a team to join
- Hold a team meeting, discuss and allocate tasks to each member, make a detailed development plan
- Hold regular meetings to confirm the development progress, discuss issues raised during the development.
- Use SDP tools, such as Visio, Rational Rose, Gitee and Github