C++ Templates and Exceptions

Templates

Function Templates

Two almost identical functions: same code, different data types

```
int          integerMinimum(int a, int b)
{
      return(a < b ? a : b);
}
double          doubleMinimum(double a, double b)
{
      return(a < b ? a : b);
}
```

or one generic function with a data type parameter.

```
template<typename T>
T            genericMinimum(T a, T b)
{
      return(a < b ? a : b);
}
cout << genericMinimum(3, 4) << ' ' << genericMinimum(1.1, 3.1) << ' '
      << genericMinimum('t', 'g') << ' ';
```

Class Templates

```
template<typename T>
classBasicVector
{
    public:
        BasicVector(int capacity = 10);
        T&   operator[](int i) { return(a[i]); }
    private:
        T*          a;
        int         capacity_;
};
template<typename T>
BasicVector<T>::BasicVector(int capacity )
{
    capacity_ = capacity ;
    a = new T[capacity_];
}
```

```
BasicVector<int>        iv(5);
BasicVector<double>     dv(20);
BasicVector<string>     sv(10);

iv[3] = 8;
dv[14] = 2.5;
sv[7] = “hello”;
```

Templated Arguments

BasicVector<BasicVector<int>> xv(5);

array of 10 elements (default size), each an array with 5 elements.

5 rows, 10 columns.

xv[2][8] = 15;

Exceptions

An Alternative to Returning Error Status

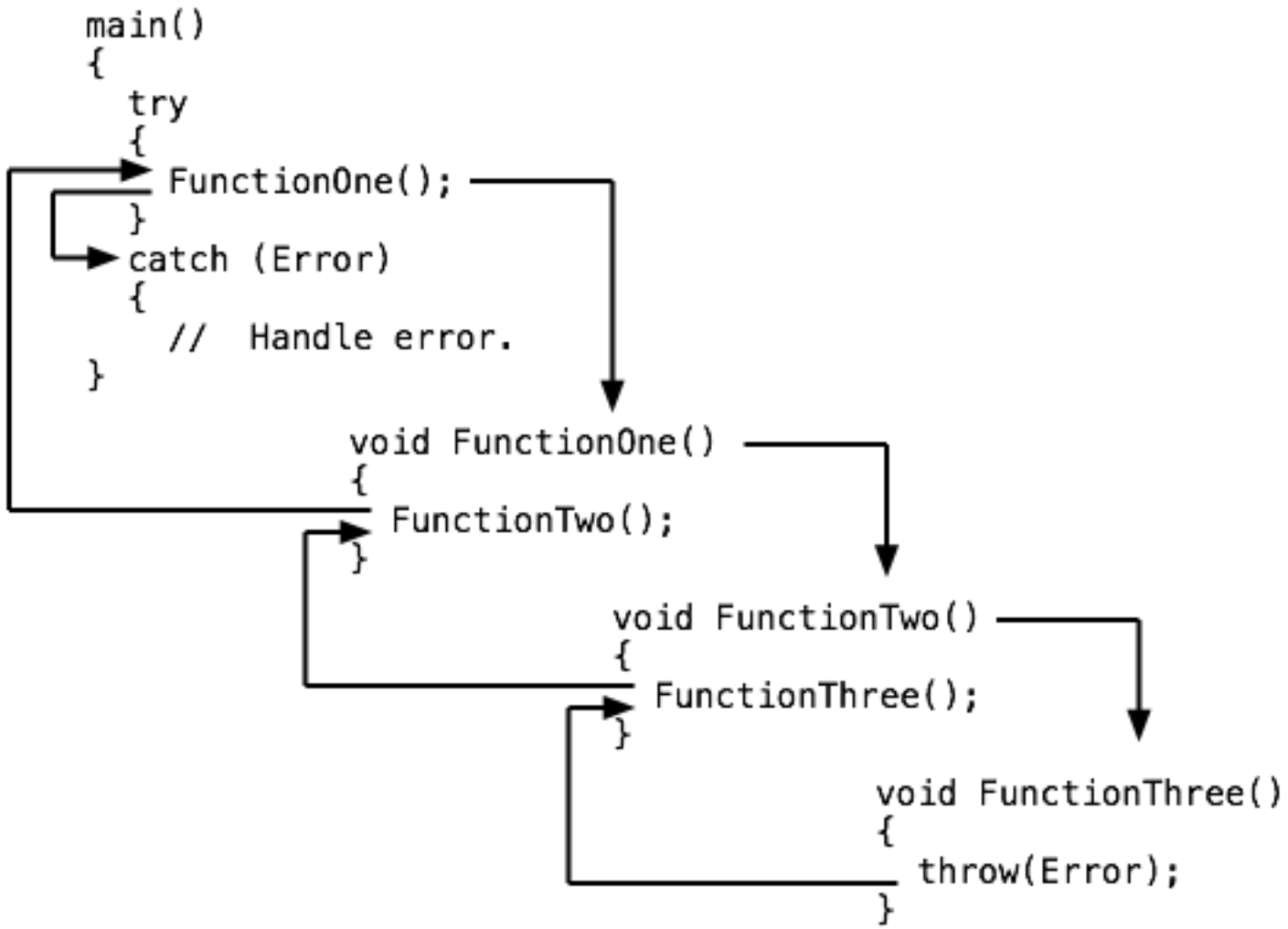
Caller:

```
try
{
        SomeFunction();
}
catch (exceptionClass)
{
        //          Handle exception
}
```

SomeFunction:

```
throw (exceptionClass);
```

Unwinding the Call Chain



Exceptions Can Include Relevant Data

```
classMathException
{
    public:
        MathException(const string& error)
        : errorMessage(error) { };

        string      getErrorMessage() { return(errorMessage); }
    private:
        string              errorMessage;
}
```

```
try
{
	if (divisor == 0)
	{
		throw MathException(“Divide by zero.”);
	}
}
catch (MathException exception)
{
	cout << exception.getErrorMessage() << endl;
}

void calculator() throw(MathException)		//	Can’t throw anything else.
void calculator()				//	Can throw anything.
void calculator() throw()			//	Can’t throw anything at all.
```