Exception Handling

Humayun Kabir
Professor, CS, Vancouver Island University, BC, Canada

Exception Handling

Outline

- No Error Handling
- Error Handling
- Throwing Exception
- Catching Exception
- Non-Standard Exception
- Standard Exception

No Error Handling

```
int quotient(int dividend, int divisor) {
        //Will throw Floating point exception if divisor is 0
        //Potential error condition is not checked before doing the operation
        return dividend/divisor;
}
```

No Error Handling

```
int main(int argc, char** argv) {
            int dividend;
            int divisor;

            while (true) {
                        cout<<"Enter dividend and divisor: ";
                        cin>>dividend;
                        cin>>divisor;

                        //Potential error condition is not checked before
                        //calling 'quotient' function.
                        //Program will crash at this point when error condition, divide by zero,
                        //arises and the function throws an exception.
                        cout<<"quotient("<<dividend<<", "<<divisor<<"): "<<
                                                quotient(dividend, divisor)<<
                                                endl;

                        cout<<"Continue? (yes/no): ";
                        string response;
                        cin>>response;
                        if(response != "yes" && response != "YES" &&
                                    response != "y" && response != "Y" ){
                                    break;
                        }
            }


            return 0;
}
```

Error Handling

```
int quotient(int dividend, int divisor) {
        //Will throw Floating point exception if divisor is 0
        //Potential error condition is being checked before doing the
operation
        if(divisor == 0 ){
                //Function must return an integer value, i.e.
                //returning an incorrect quotient
                return -1;
        }
        return dividend/divisor;
}
```

Error Handling

```
int main(int argc, char** argv) {
            int dividend;
            int divisor;

            while (true) {
                        cout<<"Enter dividend and divisor: ";
                        cin>>dividend;
                        cin>>divisor;

                        //Outputs incorrect quotient in an error condition
                        cout<<"quotient("<<dividend<<", "<<divisor<<"): "<<
                                    quotient(dividend, divisor)<<
                                    endl;

                        cout<<"Continue? (yes/no): ";
                        string response;
                        cin>>response;
                        if(response != "yes" && response != "YES" &&
                                    response != "y" && response != "Y" ){
                                    break;
                        }
            }


            return 0;
}
```

Throw Exception

```
int quotient(int dividend, int divisor) {
        //Will throw Floating point exception if divisor is 0
        //Potential error condition is being checked before doing the
operation
        if(divisor == 0 ){
                //By throwing a 'const char*' exception like
                //an error message to the user function does not need
                //to return an incorrect quotient
                throw "Error! divisor can't be zero";
        }
        return dividend/divisor;
}
```

Throw Exception

```
int main(int argc, char** argv) {
            int dividend;
            int divisor;

            while (true) {
                        cout<<"Enter dividend and divisor: ";
                        cin>>dividend;
                        cin>>divisor;

                        //Gets an exception in an error condition and program terminates abnormally
                        cout<<"quotient("<<dividend<<", "<<divisor<<"): "<<
                                    quotient(dividend, divisor)<<
                                    endl;

                        cout<<"Continue? (yes/no): ";
                        string response;
                        cin>>response;
                        if(response != "yes" && response != "YES" &&
                                    response != "y" && response != "Y" ){
                                    break;
                        }
            }


            return 0;
}
```

Catching Exception

```
int main(int argc, char** argv) {

        ……….

        //Function that has potential to throw exception are
        //put inside 'try' block.
        //If no error condition exists, i.e. no exception is thrown by the statements
        //inside 'try' block,
        //these statements will be exceuted in usual manner.
        //All the statements inside 'catch' block will be skipped.
        //If an exception is thrown on an error condition by any
        //function call or statement inside 'try' block rest of the
        //statements inside 'try' block will be skipped and the
        //statements inside 'catch' block will be excuted..
        try {
             cout<<"quotient("<<dividend<<", "<<divisor<<"): "<<
                                        quotient(dividend, divisor)<<endl;

        }
        catch (const char* ex) {
             //Skipped in normal conditions
             //Executed only if exception is thrown while executing any statement
             //inside associated try block
             cout<<ex<<endl;
        }

        ……..

        return 0;
}
```

Non-Standard Exception

```
int quotient(int dividend, int divisor) {
        //Will throw Floating point exception if divisor is 0
        //Potential error condition is being checked before doing the
operation
        if(divisor == 0 ){
                //By throwing a 'const char*' exception like
                //an error message to the user function does not need
                //to return an incorrect quotient
                throw string("Error! divisor can't be zero");
        }
        return dividend/divisor;
}
```

Non-Standard Exception

```
int main(int argc, char** argv) {

        ……….

        //Function that has potential to throw exception are
        //put inside 'try' block.
        //If no error condition exists, i.e. no exception is thrown by the statements
        //inside 'try' block,
        //these statements will be exceuted in usual manner.
        //All the statements inside 'catch' block will be skipped.
        //If an exception is thrown on an error condition by any
        //function call or statement inside 'try' block rest of the
        //statements inside 'try' block will be skipped and the
        //statements inside 'catch' block will be excuted..
        try {
             cout<<"quotient("<<dividend<<", "<<divisor<<"): "<<
                                  quotient(dividend, divisor)<<endl;

        }
        catch (string ex) {
             //Skipped in normal conditions
             //Executed only if exception is thrown while executing any statement
             //inside associated try block
             cout<<ex<<endl;
        }

        ……..

        return 0;
}
```

Standard Exception: Base Class

- All objects thrown by components of the standard library are derived from this class.
- All standard exceptions can be caught by catching this type by reference.

```
class exception {
    public:
        exception () noexcept;                          //default constructor
        exception (const exception&) noexcept;          //copy constructor
        exception& operator= (const exception&) noexcept;   //copy assignment
        virtual ~exception();                                //destructor
        virtual const char* what() const noexcept;
        //returns cstring to identify the exception

};
```

Standard Exception: Custom Class

```
 class divide_by_zero_exception: public exception {
    public:
         const char* what() const noexcept override {
            return "divide_by_zero_exception: divisor can't be zero";
         }
};
```

Standard Exception

```
int quotient(int dividend, int divisor) {
        //Will throw Floating point exception if divisor is 0
        //Potential error condition is being checked before doing the
operation
        if(divisor == 0 ){
            //By throwing an standard but customized exception function does
            //not need to return an incorrect quotient
            throw divide_by_zero_exception();
        }
        return dividend/divisor;
}
```

Standard Exception

```
int main(int argc, char** argv) {

        ……….

        //Function that has potential to throw exception are
        //put inside 'try' block.
        //If no error condition exists, i.e. no exception is thrown by the statements
        //inside 'try' block,
        //these statements will be exceuted in usual manner.
        //All the statements inside 'catch' block will be skipped.
        //If an exception is thrown on an error condition by any
        //function call or statement inside 'try' block rest of the
        //statements inside 'try' block will be skipped and the
        //statements inside 'catch' block will be excuted..
        try {
             cout<<"quotient("<<dividend<<", "<<divisor<<"): "<<
                                    quotient(dividend, divisor)<<endl;

        }
        //catch clause can use standard exception reference to
        //catch any exception that has been derived from
        //the standard exception, simplifies coding
        catch (const exception& ex) {
                 cout<<ex.what()<<endl;
        }

        ……..

        return 0;
}
```