

## 13장 예외처리

1. 전통적인 예외 처리 방식
2. C++의 예외 처리 방식
3. 예외로 객체 사용하기

# 학습목표

- 예기치 않은 오동작을 방지하기 위한 예외 처리를 설명할 수 있다.
- try, throw, catch를 사용하여 예외 처리를 할 수 있다.
- 객체를 사용하여 예외 처리를 할 수 있다.

# 예외처리

- 예외(exception)
  - 프로그램은 실행되는 동안에 비정상적으로 종료되지 않도록 만들어져야 한다.
  - 개발자의 실수가 아닌, 컴퓨터의 리소스가 부족하거나 사용자의 잘못된 입력이나 범위를 벗어난 값의 입력과 같은 프로그램 외적인 원인에 의해 발생하여 예기치 않은 오류를 예외라고 한다.
- 예외 처리(exception handling)
  - 프로그램은 이러한 예외가 발생했을 때에도 예외가 발생했다는 사실을 사용자에게 알려 문제 해결에 적절하게 대처할 수 있게 만들어져야 한다.
  - 예외 발생에 올바르게 처리하지 못해 오류가 발생하였다면 프로그래머의 책임이다.
  - 예외 상황에서 오동작을 막고 정상적인 종료를 위한 안전장치를 예외 처리라고 한다.

# 전통적인 예외 처리 방식

- 예외 발생의 예
  - 다음 프로그램은 사용자가 5 글자 이하의 문자열을 제대로 입력하였다면 아무런 문제가 없다.
  - 그러나 5 글자를 초과하는 문자열을 입력하면 실행 시간 오류가 발생하며 프로그램이 비정상적으로 종료하게 된다.
- ex13\_1.cpp (1) (예외 발생)

```
#include <iostream>
using namespace std;

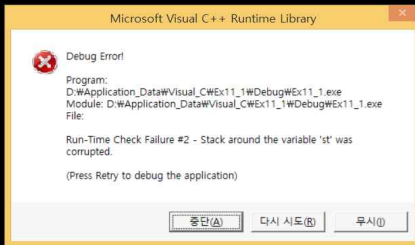
void main()
{
    char st[6];
    cout << "5 글자 이하의 문자열을 입력 하시오: ";
    cin >> st;
    cout << "입력한 문자열은 " << st << " 입니다." << endl;
}
```

# 전통적인 예외 처리 방식

- ex13\_1.cpp (2) (예외 발생)

5 글자 이하의 문자열을 입력 하시오: Korea  
입력한 문자열은 Korea 입니다.  
계속하려면 아무 키나 누르십시오 . . .

5 글자 이하의 문자열을 입력 하시오: Korean  
입력한 문자열은 Korean 입니다.



# 전통적인 예외 처리 방식

- C 프로그램에서는
  - 함수가 정상적으로 처리될 수 없을 때 특정 값을 반환하여 정상적으로 처리되지 못했음을 알려준다.
- C++에서는
  - 반환 자료형으로 bool 형을 사용하여 true를 반환하면 성공적인 수행을 나타내고 false를 반환하면 문제가 발생하였음을 나타낼 수 있다.
  - 다음 예제는 Dynamic Array 템플릿 클래스에서 bool 형을 반환하여 예외 처리를 하는 예이다.

# 전통적인 예외 처리 방식

- stack2.h (1) (예제 13.2 : 반환 값을 사용한 예외 처리)

```
// stack2.h
```

```
template <typename T> class Stack
{
protected:
    T* pstack;      // 할당된 메모리 주소
    int stsize;     // Stack의 크기
    int top;        // Stack의 top
public:
    Stack(int size = 10);
    ~Stack();
    bool push(T value, int &pt);
    bool pop(T& value, int &pt);
};
```

# 전통적인 예외 처리 방식

- stack2.h (2) (예제 13.2 : 반환 값을 사용한 예외 처리)

```
template <typename T> Stack<T>::Stack(int size)
{
    stsize = size;
    pstack = new T[stsize];
    top = 0;
}
```

```
template <typename T> Stack<T>::~~Stack()
{
    delete [] pstack;
    stsize = 0;
}
```



# 전통적인 예외 처리 방식

- stack2.h (3) (예제 13.2 : 반환 값을 사용한 예외 처리)

```
template <typename T> bool Stack<T>::push(T value, int &pt)
{
    if (top < stsize) {
        pt = top;
        pstack[top++] = value;
        return true;
    }
    else
        return false;
}

template <typename T> bool Stack<T>::pop(T& value, int &pt)
{
    if (top > 0) {
        value = pstack[--top];
        pt = top;
        return true;
    }
    else
        return false;
}
```

# 전통적인 예외 처리 방식

- ex13\_2.cpp (1) (예제 13.2 : 반환 값을 사용한 예외 처리)

```
// ex13_2.cpp
#include <iostream>
#include "stack2.h"
using namespace std;

int main()
{
    Stack<int> st(5);
    int pt, value;
    char code;

    cout << "원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): ";
    cin >> code;
    while (code != 'Q' && code != 'q') {
```

# 전통적인 예외 처리 방식

- ex13\_2.cpp (2) (예제 13.2 : 반환 값을 사용한 예외 처리)

```
while (cin.get() != '\n')
    continue;
switch (code) {
case 'A':
case 'a':
    cout << "스택에 저장할 정수를 입력하세요: ";
    cin >> value;
    if (st.push(value, pt))
        cout << "스택 " << pt << "에 " << value &
            << "을 저장했습니다." << endl;
    else
        cout << "스택이 가득 차 있습니다." << endl;
    break;
```

# 전통적인 예외 처리 방식

- ex13\_2.cpp (3) (예제 13.2 : 반환 값을 사용한 예외 처리)

```
case 'D':
case 'd':
    if (st.pop(value, pt))
        cout << "스택 " << pt << "에서 " << value ₩
            << "을 읽어왔습니다." << endl;
    else
        cout << "스택이 비어 있습니다." << endl;
    break;
}
cout << "원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): ";
cin >> code;
}
cout << "프로그램을 종료합니다." << endl;
return 0;
}
```

# 전통적인 예외 처리 방식

- ex13\_2.cpp (3) (예제 13.2 : 반환 값을 사용한 예외 처리)

```
원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): a
스택에 저장할 정수를 입력하세요: 10
원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): a
스택에 저장할 정수를 입력하세요: 20
원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): d
스택에서 20을 읽어왔습니다.
원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): d
스택에서 10을 읽어왔습니다.
원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): d
스택이 비어 있습니다.
원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): q
프로그램을 종료합니다.
계속하려면 아무 키나 누르십시오 . . .
```

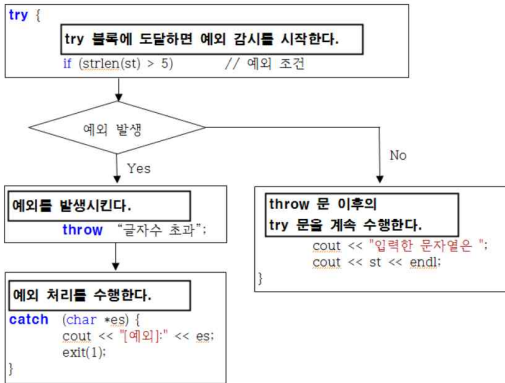
# C++의 예외 처리 방식

- 반환 값을 이용하여 예외 처리를 하면
  - 함수의 처리 결과를 반환할 수가 없어 처리 결과는 함수의 인수를 통해 참조로 처리하여야 한다.
  - 처리 결과를 함수의 반환 값으로 직접 받는 것보다 불편하다.
- C++에서는 이러한 문제점을 해결하기 위하여 예외처리 구문으로 try, throw, 그리고 catch 문을 지원한다.

```
try {  
    // 예외가 발생할 가능성이 있는 문장  
    if (예외 조건)  
        throw 예외의 형태 // 예외 발생  
}  
catch (예외의 형태) {  
    // 예외를 처리하는 문장  
}
```

# C++의 예외 처리 방식

- 예외 처리(try, throw, catch)의 동작



# C++의 예외 처리 방식

- ex13\_3.cpp (예제 13.1을 try, throw, catch 문 사용)

```
#include <iostream>
using namespace std;

void main()
{
    char st[6];
    cout << "5 글자 이하의 문자열을 입력 하시오: ";

    try {
        cin >> st;
        if (strlen(st) > 5)
            throw "글자 수 초과";
        cout << "입력한 문자열은 " << st << " 입니다." << endl;
    }
    catch (char *estr){
        cout << "[예외 발생]: " << estr << endl;
        exit(1);
    }
}
```

5 글자 이하의 문자열을 입력 하시오: Korean  
[예외 발생]: 글자 수 초과  
계속하려면 아무 키나 누르십시오 . . .



# C++의 예외 처리 방식

- stack4.h (1) (예제 13.4 : try, throw, catch 문 사용)

```
// stack4.h
```

```
template <typename T> class Stack
{
protected:
    T* pstack;           // 할당된 메모리 주소
    int stsize;          // Stack의 크기
    int top;             // Stack의 top
public:
    Stack(int size = 10);
    ~Stack();
    void push(T value, int &pt);
    T pop(int &pt);
};
```

# C++의 예외 처리 방식

- stack4.h (2) (예제 13.4 : try, throw, catch 문 사용)

```
template <typename T> Stack<T>::Stack(int size)
{
    stsize = size;
    pstack = new T[stsize];
    if (pstack == NULL)
        throw "메모리 할당 실패";
    top = 0;
}

template <typename T> Stack<T>::~~Stack()
{
    delete [] pstack;
    stsize = 0;
}
```

# C++의 예외 처리 방식

- stack4.h (3) (예제 13.4 : try, throw, catch 문 사용)

```
template <typename T> void Stack<T>::push(T value, int &pt)
{
    if (top < stsize) {
        pt = top;
        pstack[top++] = value;
    }
    else
        throw "잘못된 메모리 참조";
}
```

```
template <typename T> T Stack<T>::pop(int &pt)
{
    if (top > 0) {
        T value = pstack[--top];
        pt = top;
        return value;
    }
    else
        throw "잘못된 메모리 참조";
}
```

# C++의 예외 처리 방식

- ex13\_4.cpp (1) (예제 13.4 : try, throw, catch 문 사용)

```
// ex13_4.cpp
```

```
#include <iostream>
```

```
#include "stack4.h"
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    Stack<int> st(5);
```

```
    int pt, value;
```

```
    char code;
```

```
    try {
```

```
        cout << "원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): ";
```

```
        cin >> code;
```

```
        while (code != 'Q' && code != 'q') {
```

```
            while (cin.get() != '\n')
```

```
                continue;
```

# C++의 예외 처리 방식

- ex13\_4.cpp (2) (예제 13.4 : try, throw, catch 문 사용)

```
switch (code) {  
    case 'A':  
    case 'a':  
        cout << "스택에 저장할 정수를 입력하세요: ";  
        cin >> value;  
        st.push(value, pt);  
        cout << "스택 " << pt << "에 " << value %  
            << "을 저장했습니다." << endl;  
        break;  
    case 'D':  
    case 'd':  
        value = st.pop(pt);  
        cout << "스택 " << pt << "에서 " << value %  
            << "을 읽어왔습니다." << endl;  
        break;  
}  
cout << "원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): ";  
cin >> code;  
}
```

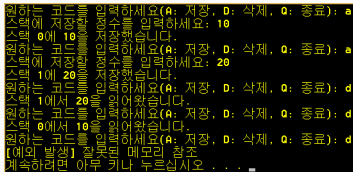
# C++의 예외 처리 방식

- ex13\_4.cpp (3) (예제 13.4 : try, throw, catch 문 사용)

```
        cout << "원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): ";
        cin >> code;
    }
    cout << "프로그램을 종료합니다." << endl;
}

catch (const char* estr) {
    cout << "[예외 발생] " << estr << endl;
}

return 0;
}
```



```
원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): a
저장수 입력하세요: 10
a에 10을 저장했습니다.
원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): a
저장수 입력하세요: 20
a에 20을 저장했습니다.
원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): d
1에서 20을 입력하였습니다.
원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): d
0에서 10을 입력하였습니다.
원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): d
[예외 발생] 잘못된 메모리 참조
계속하려면 아무 키나 누르십시오 . . .
```

# C++의 예외 처리 방식

- 다중 catch 문
  - 하나의 try 블록에서 예외의 유형에 따라 여러 개의 throw문을 던질 수 있다.
  - 여러 예외 유형에 따라 예외를 처리하려면 여러 개의 catch 블록을 두어야 한다.
  - throw문에서 던지는 예외의 자료형과 일치하는 첫 번째 catch 블록만이 수행되고, 만약 예외의 자료형과 일치하는 catch 블록이 없으면 예외 처리가 되지 않는다.

# C++의 예외 처리 방식

- ex13\_5.cpp (1) (다중 catch 문 사용)

```
#include <iostream>
using namespace std;

class Student
{
private:
    int id;
    double score;
public:
    Student():id(0), score(0) { };
    void getStud();
    void showStud();
};
```



# C++의 예외 처리 방식

- ex13\_5.cpp (2) (다중 catch 문 사용)

```
void Student::getStud()
{
    Student s;
    cout << "ID(1000 ~ 9999)와 성적(0 ~ 100)을 입력하시오 : " ;
    try {
        cin >> s.id >> s.score;
        if (s.id > 9999 || s.id < 1000)
            throw s.id;
        if (s.score < 0 || s.score > 100)
            throw s.score;
        this->id = s.id;
        this->score = s.score;
    }
```

# C++의 예외 처리 방식

- ex13\_5.cpp (3) (다중 catch 문 사용)

```
catch (int id) {  
    cout << "ID = " << id << " : ID가 틀렸습니다." << endl;  
    this->id = 0;  
}  
catch (double score) {  
    cout << "Score = " << score << " : 성적이 틀렸습니다." << endl;  
    this->score = 0;  
}  
}  
  
void Student::showStud()  
{  
    cout << "ID : " << id << ", 성적 : " << score << endl;  
}
```

# C++의 예외 처리 방식

- ex13\_5.cpp (4) (다중 catch 문 사용)

```
int main()
{
    Student st, stu[3];

    for (int i=0; i<3; i++)
        stu[i].getStud();
    cout << endl;
    for (int i=0; i<3; i++)
        stu[i].showStud();
    return 0;
}
```

```
ID(1000 ~ 9999)와 성적(0 ~ 100)을 입력하십시오 : 1001 92
ID(1000 ~ 9999)와 성적(0 ~ 100)을 입력하십시오 : 1002 105
Score = 105 : 성적이 틀렸습니다.
ID(1000 ~ 9999)와 성적(0 ~ 100)을 입력하십시오 : 10005 85
ID = 10005 : ID가 틀렸습니다.

ID : 1001, 성적 : 92
ID : 0, 성적 : 0
ID : 0, 성적 : 0
계속하려면 아무 키나 누르십시오 . . .
```

# 예외로 객체 사용하기

- 예외가 발생했을 때 throw는 클래스 객체도 던질 수 있다.
  - 프로그램이 비정상적으로 동작했을 때 예외 정보를 담은 클래스를 이용하여 예외 코드와 예외 메시지 등을 함께 보낼 수 있으며 catch 블록에서 참조로 받아 처리한다.
  - 예외 클래스를 정의하여 사용하면 예외를 각 유형별로 정리할 수 있어 프로그램의 비정상적인 동작을 막는 외에 문제의 보고에도 사용될 수 있다.

# 예외로 객체 사용하기

- exception.h (예제 13.6 : 예외 클래스 사용)

```
// exception.h

#include <string>
using namespace std;

class Exception
{
protected:
    int errCode;
    string errMessage;
    const void* errPtr;
public:
    Exception(int code, string msg, const void* src)
        :errCode(code), errMessage(msg), errPtr(src)
    {
    }
    ~Exception() { };
    int getCode() { return errCode; }
    const string getMessage() { return errMessage; }
    const void* getSource() const { return errPtr; }
};
```

# 예외로 객체 사용하기

- stack6.h (1) (예제 13.6 : 예외 클래스 사용)

```
// stack6.h

#include "exception.h"

template <typename T> class Stack
{
protected:
    T* pstack;          // 할당된 메모리 주소
    int stsize;         // Stack의 크기
    int top;            // Stack의 top
public:
    Stack(int size = 10);
    ~Stack();
    void push(T value, int &pt);
    T pop(int &pt);
};
```

# 예외로 객체 사용하기

- stack6.h (2) (예제 13.6 : 예외 클래스 사용)

```
template <typename T> Stack<T>::Stack(int size)
{
    stsize = size;
    pstack = new T[stsize];
    if (pstack == NULL)
        throw Exception(101, "메모리 할당 실패", this);
    top = 0;
}

template <typename T> Stack<T>::~~Stack()
{
    delete [] pstack;
    stsize = 0;
}
```

# 예외로 객체 사용하기

- stack6.h (3) (예제 13.6 : 예외 클래스 사용)

```
template <typename T> void Stack<T>::push(T value, int &pt)
{
    if (top < stsize) {
        pt = top;
        pstack[top++] = value;
    }
    else
        throw Exception(201, "잘못된 메모리 참조", this);
}
```

```
template <typename T> T Stack<T>::pop(int &pt)
{
    if (top > 0) {
        T value = pstack[--top];
        pt = top;
        return value;
    }
    else
        throw Exception(201, "잘못된 메모리 참조", this);
}
```



# 예외로 객체 사용하기

- ex13\_6.cpp (1) (예제 13.6 : 예외 클래스 사용)

```
// ex13_6.cpp

#include <iostream>
#include "stack6.h"
using namespace std;

int main()
{
    Stack<int> st(5);
    int pt, value;
    char code;
    try {
        cout << "원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): ";
        cin >> code;
        while (code != 'Q' && code != 'q') {
            while (cin.get() != '\n')
                continue;
        }
    }
```

# 예외로 객체 사용하기

- ex13\_6.cpp (2) (예제 13.6 : 예외 클래스 사용)

```
switch (code) {
case 'A':
case 'a':
    cout << "스택에 저장할 정수를 입력하세요: ";
    cin >> value;
    st.push(value, pt);
    cout << "스택 " << pt << "에 " << value &W
        << "을 저장했습니다." << endl;
    break;
case 'D':
case 'd':
    value = st.pop(pt);
    cout << "스택 " << pt << "에서 " << value &W
        << "을 읽어왔습니다." << endl;
    break;
}
cout << "원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): ";
cin >> code;
}
```

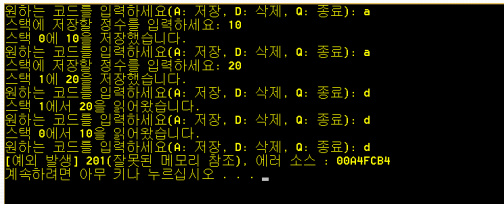
# 예외로 객체 사용하기

- ex13\_6.cpp (3) (예제 13.6 : 예외 클래스 사용)

```
        cout << "프로그램을 종료합니다." << endl;
    }

    catch (Exception& ex) {
        cout << "[예외 발생] " << ex.getCode() << "(" << ex.getMessage() <<
            << ", 에러 소스 : " << ex.getSource() << endl;
    }

    return 0;
}
```



```
원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): a
스택에 저장할 정수를 입력하세요: 10
0에 10 저장했습니다.
원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): a
스택에 저장할 정수를 입력하세요: 20
1에 20 저장했습니다.
원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): d
스택 1에서 20을 읽어왔습니다.
원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): d
스택 0에서 10을 읽어왔습니다.
원하는 코드를 입력하세요(A: 저장, D: 삭제, Q: 종료): d
[예외 발생] 201(잘못된 메모리 참조), 에러 소스 : 00A4FCB4
계속하려면 아무 키나 누르십시오 . . .
```

# Question & Answer

Any Question?

Please.