프로그래밍 역량 강화 전문기관, 민코딩

C++ Exception

C++ Exception 소개

C언어 스타일 에러처리

고전적인 에러처리 방법

```c
int main(int argc, char* argv[])
{
    int t = 1;
    int ret = run(t);

    if (ret == ERROR_CODE_TYPE1)
    {
        //에러처리1
    }
    else if (ret == ERROR_CODE_TYPE2)
    {
        //에러처리2
    }
}
```

```c
#include <stdio.h>
#include <cstdlib>

#define ERROR_CODE_TYPE1 0xFF1D
#define ERROR_CODE_TYPE2 0xFF4A

int run(int x)
{
    int ret = EXIT_SUCCESS;

    printf("메모리 할당 malloc!\n");

    if (x == 0)
    {
        printf("ERROR :: 0일수 없습니다.\n");
        ret = ERROR_CODE_TYPE1;
        goto EXIT_FAIL;
    }

    if (x == 1)
    {
        printf("ERROR :: 1일수 없습니다.\n");
        ret = ERROR_CODE_TYPE2;
        goto EXIT_FAIL;
    }

    printf("잘 동작됩니다\n");

EXIT_FAIL:
    printf("메모리 해제 delete!\n");
    return ret;
}
```

C++ 스타일로 변경

exception에 사용하는 키워드

- try ~ catch
- throw

```
int main(int argc, char* argv[])
{
    try {
        int t = 1;
        run(t);
    }
    catch (std::invalid_argument& e) {
        std::cout << "에러메세지1 : " << e.what() << std::endl;
    }
    catch (std::out_of_range& e) {
        std::cout << "에러메세지2 : " << e.what() << std::endl;
    }
}
```

```
#include <iostream>
#include <stdexcept>

void run(int x)
{
    std::cout << "스마트 포인터를 사용하여, 메모리 할당!";

    if (x == 0)
    {
        throw std::invalid_argument("0이면 안돼요.");
    }

    if (x == 1)
    {
        throw std::out_of_range("1이면 안돼요.");
    }

    std::cout << "잘 동작됩니다\n";
}
```

C언어 vs C++ Exception 비교 1

main 코드는 두 언어 모두 가독성이 좋다.

- C언어 스타일 : if 문이 에러코드인지 다른 목적의 코드인지 구분이 잘 안간다.
- C++ 스타일 : 에러처리 코드가 if가 아닌, catch 를 사용함으로 목적 구분이 잘간다.

```c
int main(int argc, char* argv[])
{
    int t = 1;
    int ret = run(t);

    if (ret == ERROR_CODE_TYPE1)
    {
        //에러처리1
    }
    else if (ret == ERROR_CODE_TYPE2)
    {
        //에러처리2
    }
}
```

C언어 호출 후 에러처리 코드

```c
int main(int argc, char* argv[])
{
    try {
        int t = 1;
        run(t);
    }
    catch (std::invalid_argument& e) {
        std::cout << "에러메세지1 : " << e.what() << std::endl;
    }
    catch (std::out_of_range& e) {
        std::cout << "에러메세지2 : " << e.what() << std::endl;
    }
}
```

C++ 호출 후 에러처리 코드

C언어 vs C++ Exception 비교 2

에러코드를 리턴하는 방법에서 가독성 차이가 있다.

- C언어 : 변수의 값을 넣고 return, 에러코드인지 리턴값인지 코드를 들여보아야 한다.
- C++ : throw 키워드로, Exception 발생하여 함수를 즉시 끝낸다는 의미로,구분 명확

```c
#define ERROR_CODE_TYPE1 0xFF1D
#define ERROR_CODE_TYPE2 0xFF4A

int run(int x)
{
    int ret = EXIT_SUCCESS;

    printf("메모리 할당 malloc!\n");

    if (x == 0)
    {
        printf("ERROR :: 0일수 없습니다.\n");
        ret = ERROR_CODE_TYPE1;
        goto EXIT_FAIL;
    }

    if (x == 1)
    {
        printf("ERROR :: 1일수 없습니다.\n");
        ret = ERROR_CODE_TYPE2;
        goto EXIT_FAIL;
    }

    printf("잘 동작됩니다\n");

EXIT_FAIL:
    printf("메모리 해제 delete!\n");
    return ret;
}
```

C언어 에러코드 발생 코드

```cpp
void run(int x)
{
    std::cout << "스마트 포인터를 사용하여, 메모리 할당!";

    if (x == 0)
    {
        throw std::invalid_argument("0이면 안돼요.");
    }

    if (x == 1)
    {
        throw std::out_of_range("1이면 안돼요.");
    }

    std::cout << "잘 동작됩니다\n";
}
```

C++ 에러코드 발생 코드
(C / C++은 final 문법이 없다.)

[도전] go 함수 제작

go(int a, int b)

- 합을 출력하는 함수
- a와 b 중 음수가 있다면 exception 발생 (invalid argument)
- a와 b 중 10이 넘는 수가 있다면, exception 발생 (out of range)

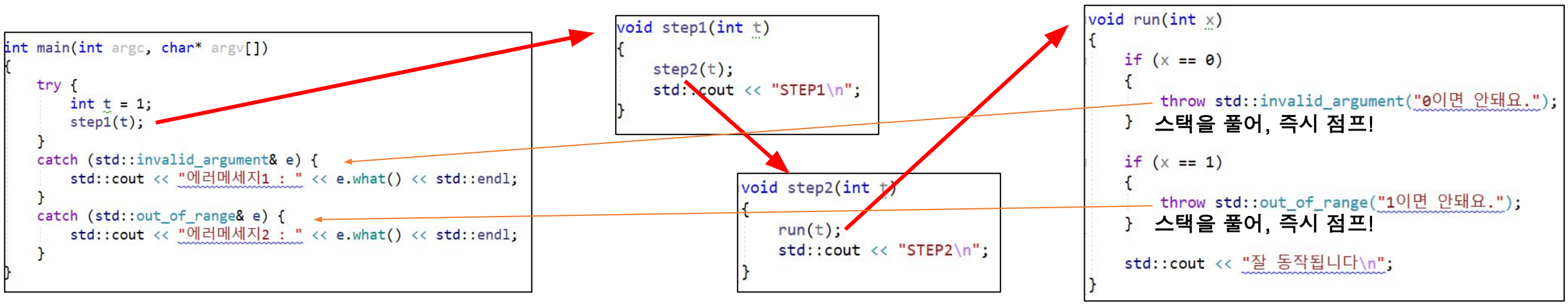
main함수

- go를 호출하고, exception 메세지를 출력한다.

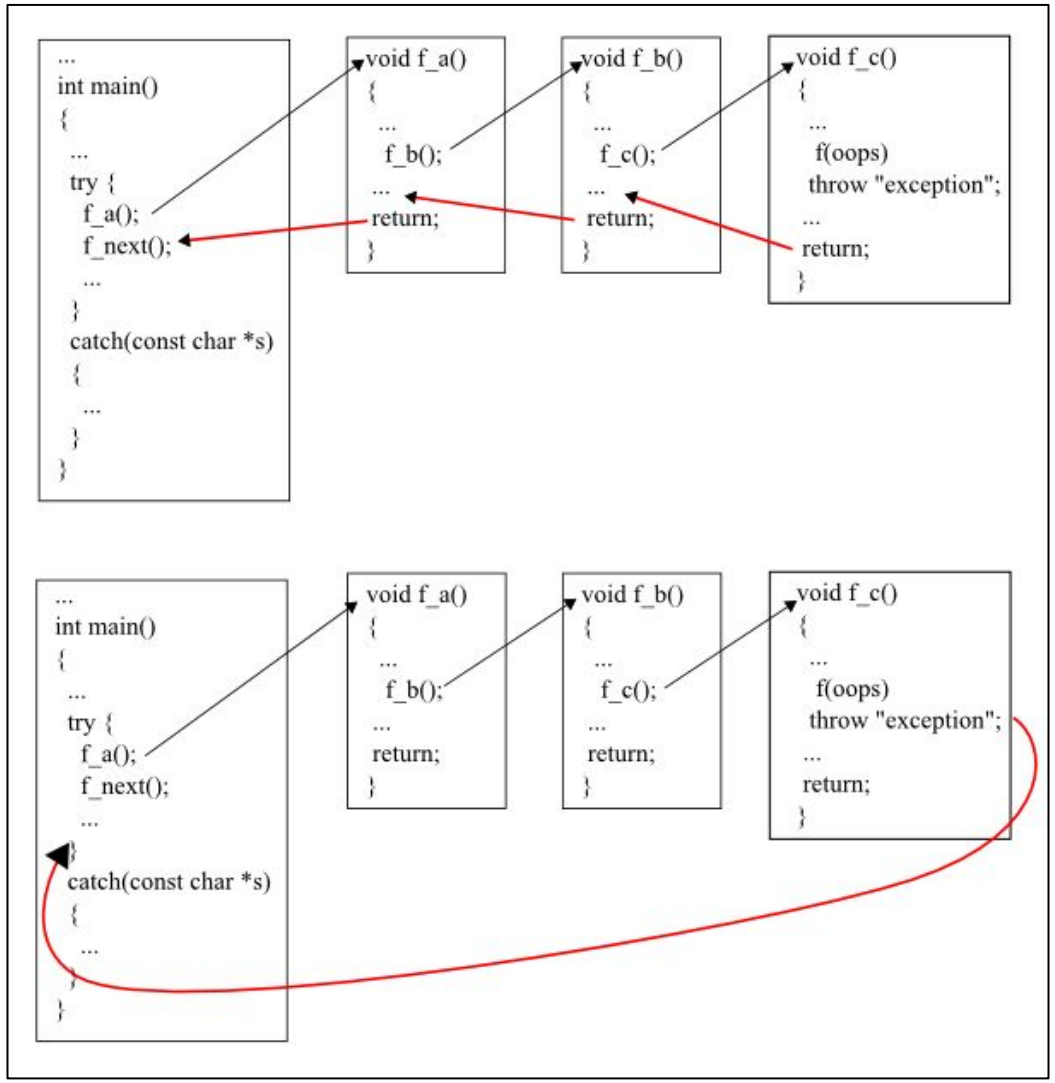
Stack Unwinding

stack unwinding (스택 풀기)

run 함수에서 throw가 발생하면
하위 함수에서 exception을 처리해줘야한다.



stack unwinding **VS** 함수 return



C언어 스타일의 메모리 해제

메모리 해제하는 코드를 모두
EXIT_FAIL에 넣어둔다.

메모리 할당 후,
예외가 발생해도 안전하게 해제할 수 있다.

```c
#define ERROR_CODE_TYPE1 0xFF1D
#define ERROR_CODE_TYPE2 0xFF4A

int run(int x)
{
    int ret = EXIT_SUCCESS;

    printf("메모리 할당 malloc!\n");

    if (x == 0)
    {
        printf("ERROR :: 0일수 없습니다.\n");
        ret = ERROR_CODE_TYPE1;
        goto EXIT_FAIL;
    }

    if (x == 1)
    {
        printf("ERROR :: 1일수 없습니다.\n");
        ret = ERROR_CODE_TYPE2;
        goto EXIT_FAIL;
    }

    printf("잘 동작됩니다\n");

EXIT_FAIL:
    printf("메모리 해제 delete!\n");
    return ret;
}
```

C++ 스타일의 메모리 해제

throw를 사용하면,

즉시 catch로 점프한다.

그리고 자동으로 메모리 해제가 이뤄진다.

```cpp
#include <iostream>
#include <stdexcept>

void run(int x)
{
    std::cout << "스마트 포인터를 사용하여, 메모리 할당!";

    if (x == 0)
    {
        throw std::invalid_argument("0이면 안돼요.");
    }

    std::cout << "잘 동작됩니다\n";
}

int main(int argc, char* argv[])
{
    try {
        int t = 1;
        run(t);
    }
    catch (std::invalid_argument& e) {
        std::cout << "에러메세지1 : " << e.what() << std::endl;
    }

    std::cout << "여기서 스마트 포인터는 벌써 해제 완료되었어요\n";
}
```

Custom Exceptions

나만의 Exception 만들기

✔ std::exception 상속만 하면 끝

```
class OhMyGodException : public std::exception
{
};

void run(int x)
{
    if (x == 2)
    {
        throw OhMyGodException();
    }

    std::cout << "잘 동작됩니다\n";
}
```

```
class OhMyGodException : public std::exception
{
public:
    char const* what() const override
    {
        "OK 메세지!"
    }
};
```

what() 호출시 에러메세지 설정

```
class OhMyGodException : public std::exception
{
public:
    explicit OhMyGodException(char const* _Message)
        : exception(_Message)
    {
    }
};
```

생성자를 추가하여,
사용자 정의 에러메세지 설정 가능

[도전] Exception 만들기

run(string name)

- 이름이 세 단어가 아니라면 : NoThreeException 발생
- 이름에 A 단어가 있는 경우 : NoAException 발생

main 함수

- Exception 처리 – 메세지 출력하기