

객체지향프로그래밍 LAB #14

<기초문제>

1. 아래의 프로그램을 작성하시오. (*구현* 부분을 채울 것, 표의 상단: 소스코드, 하단: 실행결과)

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

int sum(int x, int y) { return x + y; }
int mult(int x, int y) { return x * y; }
int evaluate(int(*f)(int, int), int x, int y) {
    return f(x, y);
}

int main() {
    cout << evaluate(&sum, 2, 3) << endl;
    cout << evaluate(&mult, 2, 3) << endl;

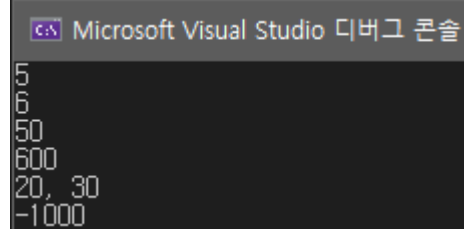
    // lambda 함수: [](입력변수)->리턴타입 {본문}
    // sum(): [](int x, int y)->int { return x + y; }
    cout << evaluate(/* 구현 */, 20, 30) << endl;

    // simplified lambda함수 표현: [](입력변수) {본문}
    // mult(): [](int x, int y) { return x * y; }
    cout << evaluate(/* 구현 */, 20, 30) << endl;

    //생성과 호출을 동시에: 람다함수(입력값)
    [](/* 구현 */) { /* 구현 */ }(20, 30);

    auto f = [](int x, int y) { return x - y; };
    cout << f(1000, 2000) << endl;

    return 0;
}
```



Microsoft Visual Studio 디버그 콘솔

```
5
6
50
600
20, 30
-1000
```

2. 아래의 프로그램을 작성하시오. (*구현* 부분을 채울 것, 표의 상단: 소스코드, 하단: 실행결과)

```
#include <iostream>
#include <functional> // function object
using namespace std;

//[closure]: 외부 변수를 lambda함수 내부로 전달
//[a]: 변수 a를 call by value로 lambda함수에 전달
//[&a]: 변수 a를 call by reference로 전달
//[=]: 모든 외부 변수를 call by value로 전달
//[&]: 모든 외부 변수를 call by ref.로 전달
// - 사용시 주의할 점: closure를 사용할 경우 function객체로 assign 받을 것
```

```

int evaluate2(function<int(int,int)> f, int x, int y) {
    return f(x, y);
}

int main() {
    int a = 10, b = 20;

    //[a]: 변수 a를 call by value로 lambda함수에 전달
    cout << evaluate2(/* a + x + y 람다 함수 구현 */, 2, 3) << endl;

    //[&]: 모든 외부 변수를 call by ref.로 전달
    /* a = 20; a * x 람다 함수 구현 */
    cout << "a: " << a << endl;
    return 0;
}

```

Microsoft Visual Studio 디버그 콘솔

```

15
200
a: 20

```

3. 아래의 프로그램을 작성하시오. (/*구현*/ 부분을 채울 것, 표의 상단: 소스코드, 하단: 실행결과)

```

#include <iostream>
#include <functional>
#include <algorithm> // for_each, copy, transform
#include <vector>
using namespace std;

int main() {
    vector<int> v1 = { 1,2,3,4 };
    for (int& elem : v1) {
        cout << elem << 'Wt';
    }
    cout << endl;

    // for_each(시작위치(iter), 끝위치(iter), 람다함수)
    for_each(/* 구현 */); // v1의 시작부터 끝까지 출력, (띄어쓰기는 탭으로)
    cout << endl;

    for_each(/* 구현 */); // v1의 시작부터 끝까지 모든 elem++
    for_each(/* 구현 */); // v1의 시작부터 끝까지 출력, (띄어쓰기는 탭으로)
    cout << endl;

    int a = 10;
    for_each(/* 구현 */); // v1의 시작부터 끝까지 모든 elem+=a
    for_each(/* 구현 */); // v1의 시작부터 끝까지 출력, (띄어쓰기는 탭으로)
    cout << endl;

    vector<int> v2(v1.size());
    // copy: container1 (source)의 element를 container 2(destinstion)로 복사
    // copy(src시작위치, src끝위치, dst시작위치)
    // v1: {1, 2, 3, 4}
    // v2: {0, 0, 2, 3}
}

```

```

copy(/* 구현 */); // v1의 (시작+1) ~ (끝-1)을 v2의 (시작+2)위치부터 하나씩 복사
for_each(/* 구현 */); // v2의 시작부터 끝까지 출력, (찍어쓰기는 탭으로)
cout << endl;

// transform: cont1의 element를 변형한다음(람다함수) cont2에 복사
// transform(src시작위치, src끝위치, dst시작위치, 람다함수)
transform(/* 구현 */); // v1의 시작부터 끝까지 제공해서 v2에 복사
for_each(/* 구현 */); // v2의 시작부터 끝까지 출력, (찍어쓰기는 탭으로)
cout << endl;

return 0;
}

```

Microsoft Visual Studio 디버그 콘솔

1	2	3	4
1	2	3	4
2	3	4	5
12	13	14	15
0	0	13	14
144	169	196	225

4. 아래의 프로그램을 작성하시오. (/*구현*/ 부분을 채울 것, 표의 상단: 소스코드, 하단: 실행결과)

```

#include <iostream>
#include <vector>
using namespace std;
int main() {
    vector<int> v{ 1, 2, 3, 4 };

    int index;
    cin >> index;
    try { cout << /* 구현 */ << endl; } // v.at(index) VS v[index]
    catch (exception& e) {
        cout << /* 구현 */ << endl;
        cout << "인덱스 에러" << endl;
    }

    cout << "[Program is running]" << endl;
    return 0;
}

```

Microsoft Visual Studio 디버그 콘솔

```

0
1
[Program is running]

```

Microsoft Visual Studio 디버그 콘솔

```

5
invalid vector<T> subscript
인덱스 에러
[Program is running]

```

5. 아래의 프로그램을 작성하시오. (/구현/ 부분을 채울 것, 표의 상단: 소스코드, 하단: 실행결과)

```
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
using namespace std;

//예외처리: try/catch구문, throw문
//1. 예외가 발생하는 부분을 try에 넣는다
//2. 예외가 발생하면 catch 구문이 실행된다
class FileNotFoundException : public exception {
    string message;
public:
    FileNotFoundException(string _m) :
        message("File not found: " + _m) {}
    virtual const char* what() const throw() {
        return message.c_str();
    }
};

vector<int> load_vector(string filename) {
    ifstream fin(filename);

    // 파일이 열리지 않으면(파일이 존재하지x)
    if (!fin) {
        // 예외처리 (throw)
        /* 구현 */
    }

    vector<int> result;
    int num, value;
    // 파일로부터 값을 result에 저장
    // 파일의 form: size, elements (5 1 2 3 4 4)
    fin >> num;
    for (int i = 0; i < num; i++) { /* 구현 */ }
    return result;
}

int main() {
    try {
        /* 구현 */ // values.dat 파일에서 vector 로드
        for (int elem : v)
            cout << elem << ' ';
        cout << endl;
    }
    catch (exception& e) {
        cout << e.what() << endl;
    }
    return 0;
}
```

Microsoft Visual Studio 디버그 콘솔

1 2 3 4 4

Microsoft Visual Studio 디버그 콘솔

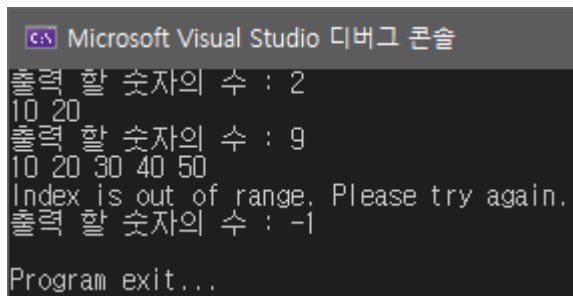
File not found: values.dat

<응용문제>

1. 아래의 코드를 기반으로 정수를 입력 받았을 때, 그 정수만큼 list를 출력하는 프로그램을 작성하시오. 이때, 만약 list의 크기보다 입력 받은 정수가 크다면 try/catch문을 이용하여 예외처리한다. 또한, 0 이하의 숫자를 입력 받으면 프로그램을 종료하도록 한다.

```
int main() {
    vector<int> list{ 10, 20, 30, 40, 50 };
    int num; // 출력할 list의 수
    while (1) { /* 구현 */ }
    cout << "Program exit..." << endl;
    return 0;
}
```

1-출력화면:



```
Microsoft Visual Studio 디버그 콘솔
출력 할 숫자의 수 : 2
10 20
출력 할 숫자의 수 : 9
10 20 30 40 50
Index is out of range. Please try again.
출력 할 숫자의 수 : -1
Program exit...
```

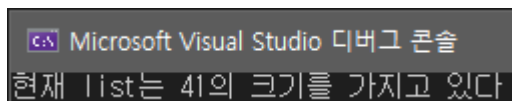
2. 아래의 조건을 만족하는 프로그램을 작성하시오.

- 1~100 사이의 랜덤한 크기를 가진 vector를 만들고 0, 1, 2, 3, ... 순서대로 채운다.
 - 예) 크기가 45인 vector: (0, 1, 2, 3, ..., 43, 44)
- 반복문을 이용하여 vector의 처음부터 시작하여 하나씩 접근한다.
- 주어진 vector의 크기를 벗어나 접근한다면 try/catch문을 이용하여 현재 vector의 크기를 출력한 후 프로그램을 종료한다.

```
int main() {
    vector<int> list;
    // vector를 1~100 사이의 random한 크기로 만들고 채우는 코드 구현

    int cnt = -1;
    while (1) {
        cnt++;
        try { /* 구현 */ }
        catch (exception& e) { /* 구현 */ }
    }
    return 0;
}
```

2-출력화면:



```
Microsoft Visual Studio 디버그 콘솔
현재 list는 41의 크기를 가지고 있다
```

3. 아래의 조건을 만족하는 프로그램을 작성하시오. 단, data.txt파일은 메모장을 이용해 직접 만들.

➤ 아래 코드를 기반으로 학생 정보를 파일입출력으로 읽어와 vector에 대입.

➤ 파일을 제대로 읽어왔는지 여부는 try/catch문으로 판단.

■ 제대로 읽어왔다면 3-출력화면 (a)와 같이 출력

■ 제대로 읽어오지 못했다면 3-출력화면 (b)와 같이 출력

```
class FileNotFoundException : public exception {
    string message; // Identifies the exception and filename
public:
    FileNotFoundException(const string& fname) :
        message("File W" + fname + "W not found") {}

    virtual const char* what() const throw () {
        return message.c_str();
    }
};

class CStudent
{
private:
    string m_Name;
    int m_Number;
    string m_Major;
public:
    CStudent() {}
    ~CStudent() {}
    void setAll(string _name, int _num, string _maj) {
        m_Name = _name;
        m_Number = _num;
        m_Major = _maj;
    }
    void Display() {
        cout << "이름: " << m_Name << endl;
        cout << "학번: " << m_Number << endl;
        cout << "전공: " << m_Major << endl << endl;
    }
};

vector<CStudent> read_file(string& filename) { /* 구현 */ }

int main() {
    string str;

    cout << "파일 이름 : ";
    cin >> str;

    try {
        vector<CStudent> numbers = read_file(str);
        for (CStudent value : numbers)
            value.Display();
    }
    catch (std::exception& e) {
        cout << e.what() << "Wn";
    }

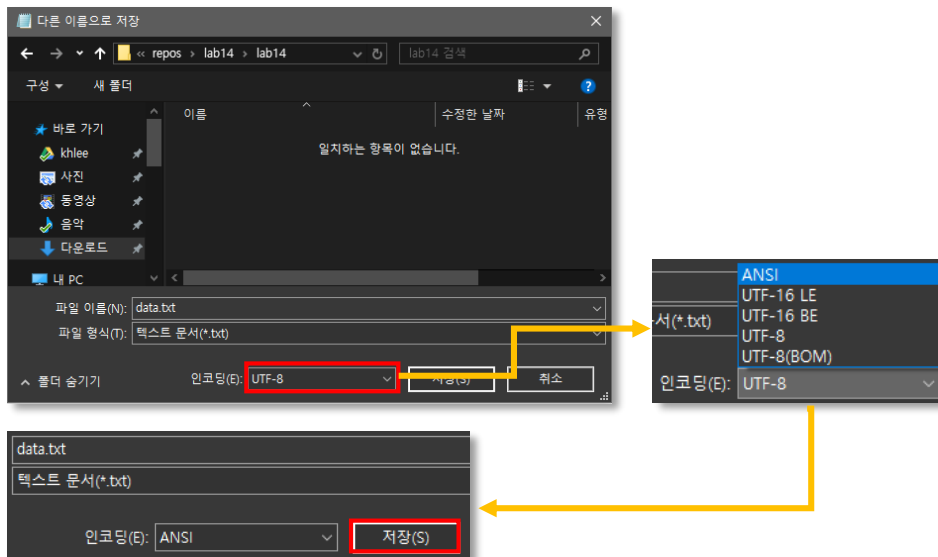
    return 0;
}
```

[data.txt]

5	
문재인 1972100100 법률학과	
한가인 2001111222 관광경영학과	
박효신 2003123789 포스트모던음악학과	
김태리 2008135135 언론정보학과	
장윤호 2014112358 컴퓨터공학과	

(주의사항)

저장할 때 아래와 같이 인코딩을 UTF-8이 아닌 ANSI로 선택해야 파일 입출력 시, 한글이 깨지지 않음. UTF-8로 저장했다면 다른 이름으로 저장을 통해 ANSI로 다시 선택하여 저장하거나, [링크](#)를 참고하여 학생 이름과 학과 정보를 받아올 때 코드 내에서 UTF-8을 ANSI로 바꿔주어야 함.



3-출력화면:

<p>(a) 파일을 제대로 읽어왔을 때</p> <pre> Microsoft Visual Studio 디버그 콘솔 파일 이름 : data.txt 이름 : 문재인 학번 : 1972100100 전공 : 법률학과 이름 : 한가인 학번 : 2001111222 전공 : 관광경영학과 이름 : 박효신 학번 : 2003123789 전공 : 포스트모던음악학과 이름 : 김태리 학번 : 2008135135 전공 : 언론정보학과 이름 : 장윤호 학번 : 2014112358 전공 : 컴퓨터공학과 </pre>	<p>(b) 파일을 제대로 읽어오지 못했을 때</p> <pre> Microsoft Visual Studio 디버그 콘솔 파일 이름 : data File "data" not found </pre>
--	--

4. 아래의 조건을 만족하는 프로그램을 작성하시오.

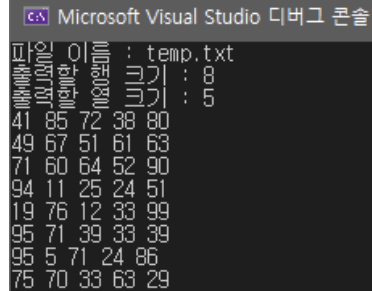
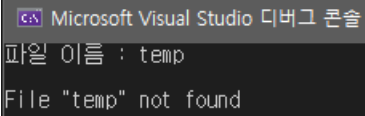
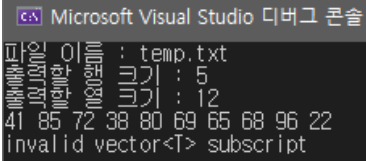
- 0~100 사이의 임의의 정수로 구성된 10x10 행렬을 **temp.txt** 파일로 저장.
- 위에서 생성한 파일을 읽어와 2차원 vector에 입력 후, 사용자가 원하는 크기만큼 부분적으로 출력.
- 이때, 읽어올 파일 이름을 string으로 입력 받아 이를 try/catch 문으로 예외처리함.
- vector를 출력할 때 행렬의 크기를 int로 입력 받고 이를 try/catch문으로 예외처리함.

```
int main()
{
    ofstream ofs;
    ofs.open("temp.txt");
    // 임의의 10x10 행렬 저장 구현
    ofs.close();

    // 파일이름 입력
    // 입력받은 파일이름에 맞는 파일을 읽어와 vector로 입력 후, 출력 구현

    return 0;
}
```

4-출력화면:

<정상출력>	<파일을 읽어오지 못한 경우>	<출력범위를 초과한 경우>
 <pre>Microsoft Visual Studio 디버그 콘솔 파일 이름 : temp.txt 출력행의 크기 : 8 출력열의 크기 : 5 41 85 72 38 80 49 67 51 61 63 71 60 64 52 90 94 11 25 24 51 19 76 12 33 99 95 71 39 33 39 95 5 71 24 86 75 70 33 63 29</pre>	 <pre>Microsoft Visual Studio 디버그 콘솔 파일 이름 : temp File "temp" not found</pre>	 <pre>Microsoft Visual Studio 디버그 콘솔 파일 이름 : temp.txt 출력행의 크기 : 5 출력열의 크기 : 12 41 85 72 38 80 69 65 68 96 22 invalid vector<T> subscript</pre>