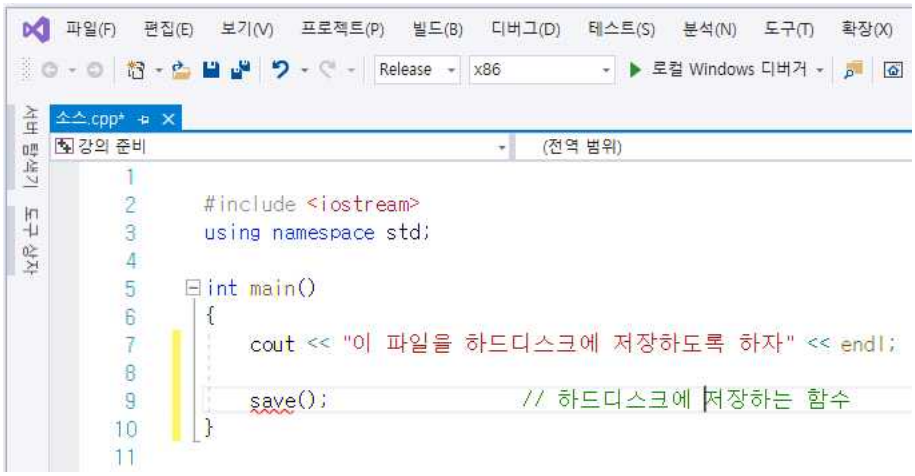
STL - 1주 2일

Wulong

파일 입출력

한 학기 강의를 하드디스크에 파일로 저장할 수 있는 기능을 코딩하겠습니다. 강의하면서 프로그램을 실행할 때마다 소스 코드를 저장하여 왔기 때문에 강의할 때 다시 설명할 기회가 있을 것입니다. 그때 파일입출력과 관련된 새로운 기능도 같이 소개하도록 하고 이 자료에서는 간단하게 저장하는 코드만 제시하겠습니다.



파란색으로 강조되어 있는 탭이 현재 편집 중인 파일이 “소스.cpp"임을 나타냅니다. main 함수에서 save 함수를 호출하면 “소스.cpp"을 파일로 저장하도록 하겠습니다. 제일 먼저 할 일은 이 함수를 선언해서 빨강 줄을 없애는 것입니다.

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

void save();                              // save 선언

int main()
{
    cout << "이 파일을 하드디스크에 저장하도록 하자" << endl;

    save();                               // 하드디스크에 저장하는 함수
}
```

이제 save 함수를 정의합니다. 바로 완성할 수는 없겠지만 무엇을 할 것인지는 표현해 볼 수 있습니다.

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

void save();                          // save 선언

int main()
{
    cout << "이 파일을 하드디스크에 저장하도록 하자" << endl;

    save();                           // 하드디스크에 저장하는 함수
}

void save()                           // save 정의
{
    // 내용을 읽을 파일을 연다 -> "소스.cpp"

    // 읽은 내용을 저장할 파일을 연다 -> 저장할 파일이름을 "강의저장.txt" 라고 하자

    // 소스.cpp의 모든 내용을 읽어 강의저장.txt 파일에 기록한다.
}
```

하드디스크의 파일을 사용하려면 파일입출력 라이브러리를 사용하여야 합니다. C++ 언어에서 이 파일입출력 기능은 fstream 헤더에 선언되어 있습니다. 입력은 ifstream, 출력은 ofstream 객체를 사용합니다.

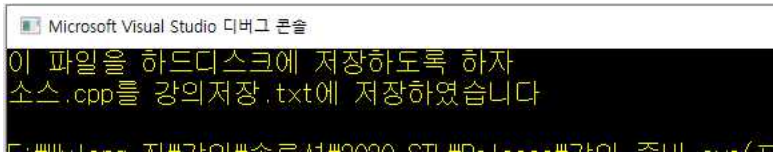
```
#include <iostream>
#include <fstream>
using namespace std;

void save();                          // save 선언

int main()
{
    cout << "이 파일을 하드디스크에 저장하도록 하자" << endl;

    save();                           // 하드디스크에 저장하는 함수
}

void save()                           // save 정의
```

```
{
    // 내용을 읽을 파일을 연다 -> "소스.cpp"
    ifstream in( "소스.cpp" );

    // 읽은 내용을 저장할 파일을 연다 -> 저장할 파일이름을 "강의저장.txt" 라고 하자
    ofstream out( "강의저장.txt" );

    // 소스.cpp의 모든 내용을 읽어 강의저장.txt 파일에 기록한다.
}
```

파일을 읽어 저장하는 방법은 여러가지가 있고 다른 것에 비하여 더 빠르고 더 효율적인 방식이 있습니다. 파일입출력은 간단한 주제가 아니므로 시간을 들여 찾아보고 공부한 후 실습까지 해서 알아둬야 합니다. 프로그래머가 반드시 알아둬야 할 내용입니다. 지금은 간단하게 작성하겠습니다.

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

void save();                            // save 선언

int main()
{
    cout << "이 파일을 하드디스크에 저장하도록 하자" << endl;

    save();                             // 하드디스크에 저장하는 함수
}

void save()                             // save 정의
{
    // 내용을 읽을 파일을 연다 -> "소스.cpp"
    ifstream in( "소스.cpp" );

    // 읽은 내용을 저장할 파일을 연다 -> 저장할 파일이름을 "강의저장.txt" 라고 하자
    ofstream out( "강의저장.txt" );

    // 소스.cpp의 모든 내용을 읽어 강의저장.txt 파일에 기록한다.
    int c;

    while ((c = in.get()) != EOF)
            out << (char)c;

    cout << "소스.cpp를 강의저장.txt에 저장하였습니다" << endl;
}
```

프로그램을 실행해 봅시다.



파일이 제대로 저장되었는지 확인해 봅니다. 어디에 저장되어 있나요? Ctrl+O를 누르면 저장된 폴더가 열립니다. “강의저장.trxt" 파일을 찾을 수 있을 것입니다.

[실습] 프로그램을 2번 연속 실행한 후 “강의저장.txt" 파일을 열어 내용을 확인해 보자.

파일을 살펴보니 어떤가요? 제대로 저장되어 있죠? 그런데 이와 같은 프로그램을 만든 목적은 매번 진행하며 변화하는 소스.cpp을 빼놓지 않고 저장하는 것인데요. 그렇게 저장되어 있지는 않죠?

하드디스크의 파일을

```
void save()                              // save 정의
{

    // 읽은 내용을 저장할 파일을 연다 -> 저장할 파일이름을 "강의저장.txt" 라고 하자
    ofstream out( "강의저장.txt" );

}
```

이와 같이 out 객체를 생성하여 접근하면 언제나 새로운 파일을 만들게 됩니다. 즉 디폴트 모드는 새로 만듦으로 설정되어 있다는 것입니다. 우리는 강의내용을 계속 덧붙일것이기 때문에

```
void save()                              // save 정의
{

    // 읽은 내용을 저장할 파일을 연다 -> 저장할 파일이름을 "강의저장.txt" 라고 하자
    ofstream out( "강의저장.txt", ios::app );
}
```

out 생성 시 덧붙여쓰기(ios::app) 모드로 생성하겠습니다

[실습] 프로그램을 2번 연속 실행한 후 “강의저장.txt" 파일을 열어 내용을 확인해 보자.

어떻습니까? 내용이 덧붙여져 있죠? 이제 save를 호출할 때마다 소스.cpp가 계속 강의.txt 파일에 덧붙여 기록될 것입니다. 만족스러운가요?

여기에 저장한 날짜와 시간을 덧붙이고 칸을 조정하고 장식을 덧붙인다면 좋은 한 학기 수업 자료가 될 것입니다. save 함수로 파일 분리하여 매번 저장되지 않도록 해야합니다. 말로 설명해야 할 내용이 많으니 다시 시간을 들여 save를 만들 겁니다. 기다려 주세요.

[실습] save 함수가 항상 “소스.cpp"를 저장하는 데 파일 이름을 인자로 준 다면 더 유연한 프로그램이 될 것입니다. 다음 main이 문제없이 실행되도록 프로그램을 수정하세요.

```
int main()
{
    cout << "이 파일을 하드디스크에 저장하도록 하자" << endl;

    save( "소스.cpp" );                    // 하드디스크에 저장하는 함수
}
```

정수 데이터를 파일에 기록하고 다시 읽어 오는 문제

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

int main()
{
    default_random_engine dre;
    uniform_int_distribution uid;

    for (int i = 0; i < 1'000; ++i)
            cout << uid( dre ) << endl;
}
```

임의의 정수값 1000개를 화면에 기록하는 프로그램입니다. 값이 상당히 크죠? 제일 큰 값을 찾을 수 있나요? 물론 컴퓨터 시켜 찾으란 말입니다. 사람이 찾기는 매우 힘듭니다. 불가능하지는 않습니다. 갯수가 늘어나면 훨씬 더 찾기 힘들 겁니다. 이런 일에 딱 맞는 것이 프로그램입니다. 작은 것부터 큰 것까지 순서대로 늘어놓을 수도 있겠죠? 정렬과 탐색 작업은 프로그램의 기본입니다.

지금은 정수 1000개를 바로 만들어 화면 출력하고 있지만 대개 우리가 처리할 자료는 다른 곳에 저장되어 있습니다. 저장되어 있는 장소는 내 컴퓨터일 수도 있지만 네트워크로 연결되어 있는 다른 컴퓨터일 수도 있습니다. 그럼 다른 곳에 있는 자료를 읽어와서 처리 할 줄 알아야 하겠습니다.

파일 입출력을 배웠으니 바로 응용해서 정수 1000개를 파일로 저장해 보겠습니다.

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

int main()
{
    default_random_engine dre;
    uniform_int_distribution uid;

    ofstream out( "정수1000개.txt" );

    for (int i = 0; i < 1'000; ++i)
            out << uid( dre ) << endl;
}
```

프로그램을 실행한 후 저장된 정수 파일을 찾아 내용을 확인해 보세요.

처리할 데이터는 하드디스크에 정수1000개.txt라는 이름으로 저장되어 있습니다. 만일 여기에서 가장 큰 값을 찾아야 한다면 어떻게 하겠습니까? 파일을 읽을 수 있어야 하는 점이 가장 우선되어야 할 것입니다.

파일을 읽어 그 값을 화면에 출력해 보겠습니다.

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

int main()
```

```
{
    ifstream in( "정수1000개.txt" );

    int n {};
    int i { 0 };

    while (in >> n)
            cout << ++i << " - " << n << endl;
}
```

정확하게 1000개를 읽을 수 있습니다.

[실습] 정수1000개.txt에 저장된 값 중 가장 큰 값을 찾아 화면에 출력하세요.

다음 시간에는 처리할 데이터의 정렬도 해 보겠습니다. 그 전에 꼭 짚어 볼 것이 있습니다.

[질문] 정수 1000개를 저장하려면 얼마만큼 메모리가 필요합니까?

[실습] 정수1000개.txt 파일의 크기를 확인하세요.

예상한 크기만큼 메모리를 차지하고 있나요? 그렇지 않을 것입니다. 필요한 크기보다 더 많은 메모리를 차지하고 있을 겁니다. 다음 과제를 해결한 후 1일차 과제와 같이 묶어 제출하세요.

[1주 2일 - 과제]

[문제 1] 정수1000개.txt가 차지하는 하드디스크 파일의 크기를 적어라.
[문제 2] 정수 1000개를 최소한의 하드디스크 크기를 사용하도록 하려면 무엇을 어떻게 해야 하는 지 설명하라.

1주 1일과 1주 2일 과제를
문서 파일 하나로 만들어 (형식은 윈도우 텍스트 문서 형식으로)
2주 1일 강의 시작 전까지 e-class로 제출해 주세요.
과제의 내용과 해결방법은 친구들이나 구글 검색을 적극 활용하여 해결하면 됩니다.
과제를 평가하여 앞으로 얼마나 열심히 공부해야 할 것인지 알려주겠습니다.