객체프로그래밍 정리노트

202204103 ict융합공학부 이승훈
202204023 ict융합공학부 김보민

[바이러니 파일]

문자로 표현되지 않는 바이러니 데이터가 기록된 파일

[바이러니 파일의 종류]
-jpeg, bmp등의 이미지 파일
-mp3 등의 오디오 파일
-hwp, doc, ppt 등의 확장자를 가진 멀티미디어 문서 파일
-obj, exe 등의 확장자를 가진 컴파일된 코드나 실행 파일

[파일 입출력 모드: 텍스트 I/O와 바이너리 I/O]

[파일 입출력 방식]
-텍스트 I/O와 바이너리 I/O의 두 방식

[텍스트 I/O]
-문자 단위로 파일에 쓰기, 파일에서 읽기, 텍스트 파일에만 적용

[바이너리 I/O]
-바이트 단위의 파일에 쓰기, 파일에서 읽기, 텍스트 파일과 바이너리 파일 모두 가능

//보민 , ->승훈
[예제 12-1] 키보드로부터 입력 받아 텍스트 파일 저장하기

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

int main() {
	char name[10], dept[20];
	int sid;
	cout << "이름>>";
	cin >> name;
```

```
        cout << "학번>>";
        cin >> sid;
        cout << "학과>>";
        cin >> dept;

        ofstream fout("c:\\temp\\student.txt");
        if (!fout) {
                cout << “c:\\temp\\student.txt 파일을 열 수 없다";
                        return 0;
        }

        fout << name << endl;
        fout << sid << endl;
        fout << dept << endl;

        fout.close();
}
```

-> 메인함수에는 name 이라는 이름을 가진 10개의 문자로 이루어진 배열과, dept라는 이름을 가진 20개의 문자로 이루어진 배열, 그리고 sid 정수형 변수를 만들어주고 사용자에게 이름, 학번, 학과를 입력하라는 메시지를 먼저 출력한 뒤 cin을 통해 사용자에게 입력을 받고 있는 것을 볼 수 있어.

// 맞아 그리고 나서 fout 이라는 ofstream 클래스의 객체를 선언하고 파일을 쓰기 모드로 연 것을 볼 수 있어. 여기서 if 문은 파일이 성공적으로 열리지 않았을 경우 오류 메지를 출력하고 프로그램을 종료하게 하는 코드로 이해할 수 있어.

-> 처음보는 코드라 낯설긴 하다 다음 예제들에서 더 연습해봐야겠어

[예제 12-3] get()을 이용한 텍스트 파일 읽기

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

int main() {
        const char* file = "c:\\windows\\system.ini";
        ifstream fin(file);
        if (!fin) {
                cout << file << " 열기 오류" << endl;
                return 0;
        }
        // 여기서 const char* file 은 파일 경로를 ”“안에 있는 값에 지정하고, 해당 경로를 file 상수 포인터에 할당해준다고 알 수 있어. 그리고 ifstream fin(file);는 ifstream 클래스를 사용하여 파일을 읽기 모드로 열어줘. 여기서 fin은 파일 입력 스트림 객체로 이를 통해 파일에서 읽기 작업을 수행할 수 있어
```

-> 파일을 읽기 모드로 사용할 땐 ifstream을 사용하는구나. 그럼 다음 줄의 if문은 파일이 열리지 않았을 경우이거나 파일이 존재하지 않는 경우를 구별하는 거 맞지 ?

```
// 응 맞아. 파일이 열리지 않거나 존재하지 않으면 오류 메시지를 출력하고 프로그램은 종료돼.
        int count = 0;
        int c;
        while ((c = fin.get()) != EOF) {
                cout << (char)c;
                count++;
        }
        cout << "읽은 바이트 수는 " << count << endl;
        fin.close();
}
// 아래 while문은 fin.get을 사용해서 파일에서 한 바이트씩 읽어오도록 작성되어있는 것을 볼 수 있어. EOF의 실행에 대해 기억나 ?
```

-> 응. 이 코드를 보면 EOF에 도달할 때까지 반복문을 실행하면서 읽어온 바이트를 출력하고 count를 증가시킨다고 볼 수 있지.

```
// 맞아. 더 자세히 설명하면 읽은 바이트 변수를 c에 저장했잖아 그걸 다시 반환했고, 이 동작이 파일의 끝에 도달하지 않았다면, 도달할 때까지 파일에서 한 바이트씩을 읽어와서 그 값을 화면에 출력하고 읽은 바이트 수를 증가시켜줘.
```

[예제 12-4] 텍스트 파일 연결

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

int main() {
        const char* firstFile = "c:\\temp\\student.txt";
        const char* secondFile = "c:\\windows\\system.ini";
        fstream fout(firstFile, ios::out | ios::app);
        -> 다른 부분들은 이해가 되는데 이 문장이 잘 이해가 안돼
        // 방금 배웠듯이 fstream 클래스를 사용하여 파일을 쓰기 모드로 열어주는 역할을 하고
        ios::out는 출력모드, ios::app는 기존 내용 끝에 이어서 쓰기 모드를 뜻해
        이 코드에서는 이 부분을 가장 잘 살펴봐야 할 것 같아.
        if (!fout) {
                cout << firstFile << " 열기 오류";
                return 0;
        }
        fstream fin(secondFile, ios::in);
        if (!fin) {
                cout << secondFile << " 열기 오류";
                return 0;
```

```
        }
        int c;
        while ((c = fin.get()) != EOF) {
                fout.put(c);
        }
        fin.close();
        fout.close();
}
```

[read()/write()로 블록 단위 파일 입출력]

-get()/put()

ㄴ 문자 혹은 바이트 단위로 파일 입출력

-read()/wirte()

ㄴ 블록 단위로 파일 입출력

istream& read(char*s, int n)

-파일에서 최대 n개의 바이트를 배열 s에 읽어 들임, 파일의 끝을 만나면 읽기 중단

ostream& write(char*s, int n)

-배열 s에 있는 처음 n개의 바이트를 파일에 저장

int gcount()

-최근에 파일에서 읽은 바이트 수 리턴

[예제 12-8] read()를 이용하여 블록 단위로 텍스트 파일 읽기

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

int main() {
        const char* file = "c:\\windows\\system.ini";
        ifstream fin;
        fin.open(file, ios::in | ios::binary);
// ios::binary 이거는 이진 모드를 나타내
-> 이진모드가 뭐야 ?
// 이진 모드란 파일을 텍스트가 아닌 이진 데이터로 처리하는 모드를 나타내. 파일을 이진 모드로 열게 되면 파일의 내용을 문자로 해석하지 않고 이진 데이터 그 자체로 처리해. 즉, 텍스트 파일과 달리 줄바꿈 문자나 특정 문자의 해석을 하지 않고 파일의 모든 데이터를 그대로 다루는 것을 뜻해
-> 그렇구나. 나머지 내용은 윗내용이랑 비슷해서 스스로 이해할 수 있을 것 같아.
```

```
	if (!fin) {
		cout << "파일 열기 오류";
		return 0;
	}
	int count = 0;
	char s[32];
	while (!fin.eof()) {
		fin.read(s, 32);
		int n = fin.gcount();
		cout.write(s, n);
		count += n;
	}
	cout << "읽은 바이트 수는 " << count << endl;
	fin.close();
}
```

[예제 13-4]

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

int main() {
	int n, sum, average;
	while (true) {
		cout << "합을 입력하세요>>";
		cin >> sum;
		cout << "인원수를 입력하세요>>";
		cin >> n;
		try {
			if (n <= 0)
				throw n;
			else
				average = sum / n;
		}
		catch (int x) {
			cout << "예외 발생!! " << x << "으로 나눌 수 없음" << endl;
			average = 0;
			cout << endl;
			continue;
		}
		cout << "평균 = " << average << endl << endl;
	}
}
```

// 이 코드는 무한 루프 안을 돌면서 사용자가 원할 때까지 계속해서 합과 인원수를 입력받고 평균을 계산하고 있어. 입력된 수가 0 이하일 경우에는 예외 처리를 통해 오류를 출력하고 다시 입력받고 있어. 이해가 잘 안 되는 부분이 있었어 ?

-> 예외 처리를 다루는 부분이 잘 이해가 안돼.

// try 블록은 예외가 발생할 수 있는 코드를 감싸주고 try 내에서 예외가 발생하게 되면 catch 블록으로 제어가 전달돼. 이 코드에 적용시키면 사용자가 입력한 값이 0 이하이면 오류로 간주하게 되지 이 코드에서는 n을 예외로 지정했어 이 예외는 catch 블록으로 전달하게 되지.

-> 그럼 catch 블록에서는 n이라는 예외를 받아서 처리한다고 볼 수 있겠네 ?

// 맞아. 잘 이해했어 n이라는 예외를 x로 받아서 처리해줘.

-> 그렇구나.