HW2. class with file IO

부산대학교 정보컴퓨터공학부
2020-55645
신세환
제출일: 2024-03-26

1. 문제정의

- 작성자의 표현으로 문제가 요구하는 사항을 재정리 한다

student클래스에서 <<연산자와 >>연산자를 오버로딩해서 객체 단위로 cout, cin이 가능하게 만들고 driver함수를 정의해서 점검하기

CourceRegistration클래스에 <<연산자, >>연산자를 파일 스트림을 사용할 수 있도록 오버로딩하고 이를 통해 driver함수에서 파일 입출력이 가능한지 점검하기

2. 클래스 정의 설명

Const reference형태로 들어가는 인자가 많기 때문에 getter에 일반함수 말고도 const요소를 위한 함수를 별도 정의해준다.

또한 외부 스트림에서 함수 내부 요소에 접근가능하도록 <<, >>오버로딩 함수들은 모두 각 클래스의 friend함수로 정의하고 구현은 cpp파일에서 진행했다.

Ex>

```
// << 오버로딩
friend ostream& operator<<(ostream& ostr, const Student& s);
// >> 오버로딩
friend istream& operator>>(istream& istr, Student& s);
```

3. 클래스의 주요 함수 설명

-main함수의 가독성을 높이기 위해서 최대한 함수로 분리하고 main에서는 최소한의 함수 실행만 제어하게 했다.

-이를 위해 파일 스트림은 driver함수 내에서만 실행되고 close하도록 만들었다.

-연산자 오버로딩을 진행해서 사용하는 측에서는 >>, <<한 번만 사용해서 바로 활용 가능하도록 만들었다.

1) Main 함수

함수 부분

```
// student data
    vector<Student> students;
```

```
// course data
vector<CourseRegistration> courses;

set_circum(students, courses);

vector<CourseRegistration> coursesTest(10);
for(int i = 0; i < 10; i++)
{
	cr_write_driver(courses[i]);
	cr_get_driver(coursesTest[i]);
}

print_data(students, coursesTest);


student_driver();
```

Student, courseRegistration객체 만들고 적절한 변수를 set_circum으로 선언한 후 cr_write_driver와 cr_get_driver를 통해 파일 스트림을 내부적으로 다뤄서 새 객체 coursesTest에 courses내용을 넣어주고 테스트한다.
Student_diver()를 통해 객체단위로 cin, cout이 가능한지 테스트한다.

2) cr_write_driver

```
함수 부분
// CourseRegistration객체를 파일 스트림에 쓰는 driver
void cr_write_driver(const CourseRegistration& ori)
{

	ofstream ofst("courseTest.bin", std::ios::out | std::ios::binary);

	if(ofst.is_open())
	{
		ofst << ori;
	}
	else
	{
```

```
        cerr << "ofstream error";
    }
}
```

객체를 const reference형으로 받아서 courseTest.bin파일에 한 줄 단위로 구분해서 바이너리로 저장한다.

3) cr_get_driver

```
함수 부분

// 파일 스트림의 binary정보로 CourseRegistration객체를 업데이트 하는 driver
void cr_get_driver(CourseRegistration& target){
    ifstream ifst("courseTest.bin", std::ios::in | std::ios::binary);

    if(ifst.is_open())
    {
        ifst >> target;
    }
    else
    {
        cerr << "ifstream error";
    }
}
```

파일 스트림 객체 상태를 점검하면서 courseTest.bin파일 내용을 target객체에 넣어준다

4) student_driver

```
함수 부분

void student_driver()
{
    Student s;
    cout << "student_driver test₩n";
    cout << "-----------------------";
    cout << "입력 값 : ₩n";
    cin >> s;
```

```
    cout << "------------------------";
    cout << "입력받은 값 : ₩n";
    cout << s;
}
```

Student 클래스에서 cin, cout이 가능한지 체크한다.

5) CourseRegistration의 <<, >> 오버로딩

```
함수 부분

ostream& operator<<(ostream& ostr, const CourseRegistration& c)
{
    ostr << c.getCourseID() << endl;
    ostr << c.getStudentID().size() << endl;
    for(int i = 0; i < c.getStudentID().size(); i++)
    {
        ostr << c.getStudentID()[i] << endl;
        ostr << c.getCourseGrade()[i] << endl;
    }

    ostr << c.getCreditHours() << endl;
    return ostr;
}

istream& operator>>(istream& istr, CourseRegistration& c)
{
    int id;
    int studentNum = 0;
    int creditHours = 0;

    // 입력 받을때마다 스트림 상태 체크
    // 과목 id
    if(!(istr >> id)){
        return istr;
    }
    c.setCourseID(id);

    // 학생 수
    if(!(istr >> studentNum)){
        return istr;
    }
```

```
    c.getStudentID().clear();
    c.getCourseGrade().clear();

    // 학생 정보
    for(int i = 0; i < studentNum; i++)
    {
        int sID;
        string grade;

        if(!(istr >> id)){
            return istr;
        }

        if(!(istr >> grade)){
            return istr;
        }

        c.addStudent(id, grade);
    }

    // 수업 시수
    if(!(istr >> creditHours))
    {
        return istr;
    }
    c.setCreditHours(creditHours);

    return istr;
}
```

<<연산자는 Ostr에 부가 설명 없이 한 라인씩 출력하게 구현했다.
vector요소는 요소 길이를 받아서 기존 값을 지우고 새로 등록하게 구현했다.
>>의 경우 스트림 상태를 체크하면서 한 줄씩 입력받게 구현했다.

6) Student의 <<, >> 오버로딩

```
함수 부분

ostream& operator<<(ostream& ostr, const Student& s)
{
        cout << "ID : " << s.getID() << endl;
        cout << "이름 : " << s.getName() << endl;
        cout << "주소 : " << s.getAddress() << endl;
        // Date클래스도 <<연산자 오버로딩 완료
        cout << "첫 수강 날짜 : " << s.getFirstDate() << endl;
        cout << "이수학점 : " << s.getCch() << endl;
        return ostr;
```

```
}


istream& operator>>(istream& istr, Student& s)
{
        printf("id : ");
        istr >> s.id;
        printf("이름 : ");
        istr >> s.name;
        printf("주소 : ");
        istr >> s.address;
        printf("날짜 : ");
        istr >> s.firstDate;
        printf("이수학점 : ");
        istr >> s.completedCreditHours;
        return istr;
}
```

Cout, cin을 사용할 것이기에 부가 설명도 같이 적어서 구현했다.

s.firstDate도 입력받을 수 있도록 Date클래스도 이와 비슷하게 <<, >> 연산자 오버로딩을 인라인함수로 정의해주었다.

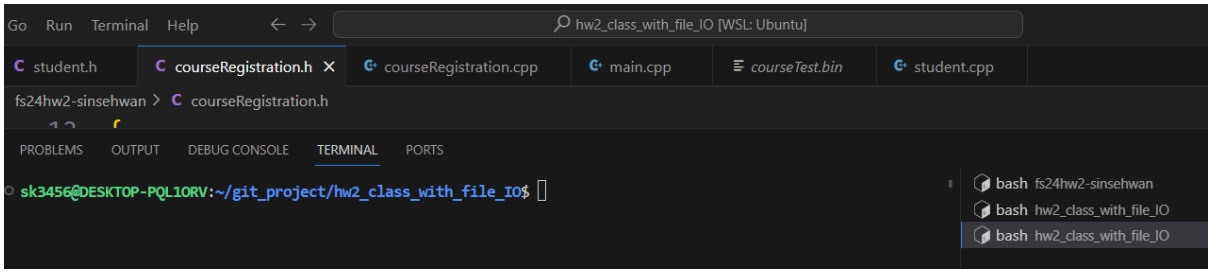
4. 프로그램 실행

1) 프로그램 실행 환경

우분투 WSL을 윈도우에 설치한 후 window powerShell로 실행한 WSL 우분투 환경에서 code .. 명령어를 입력해서 현재 디렉토리에서 VSCODE를 실행시킴

```
sk3456@DESKTOP-PQL1ORV:~/git_project/hw2_class_with_file_IO/fs24hw2-sinsehwan$ code ..
sk3456@DESKTOP-PQL1ORV:~/git_project/hw2_class_with_file_IO/fs24hw2-sinsehwan$
```

VScode에디터에서 터미널을 열고 WSL에 간접적으로 접근함



G++ 버전은 다음과 같음

```
sk3456@DESKTOP-PQL1ORV:~/git_project/hw2_class_with_file_IO/fs24hw2-sinsehwan$ g++ --version
g++ (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

sk3456@DESKTOP-PQL1ORV:~/git_project/hw2_class_with_file_IO/fs24hw2-sinsehwan$
```

2) 프로그램 실행 방법

- hw1때 사용했던 makefile을 그대로 사용하고 make명령어를 통해 main 실행파일을 생성해서 실행함.

실행화면

```
student ID : 3, Grade : B0
student ID : 5, Grade : A+
student ID : 7, Grade : A0
student ID : 13, Grade : B0
-------------------------------------
course 8
course ID : 108
course Credit hours : 1
students :
student ID : 4, Grade : A0
student ID : 6, Grade : B0
student ID : 8, Grade : F
student ID : 14, Grade : A0
-------------------------------------
course 9
course ID : 109
course Credit hours : 2
students :
student ID : 5, Grade : A+
student ID : 7, Grade : F
student ID : 9, Grade : B0
-------------------------------------
student_driver test
----------------------입력 값 :
id : 23
이름 : tls
주소 : asdf
날짜 : 3 26
이수학점 : 70
----------------------입력받은 값 :
ID : 23
이름 : tls
주소 : asdf
첫 수강 날짜 : 3월 26일
이수학점 : 70
```

5. Github 화면

(1) Git clone, add, commit, push 수행

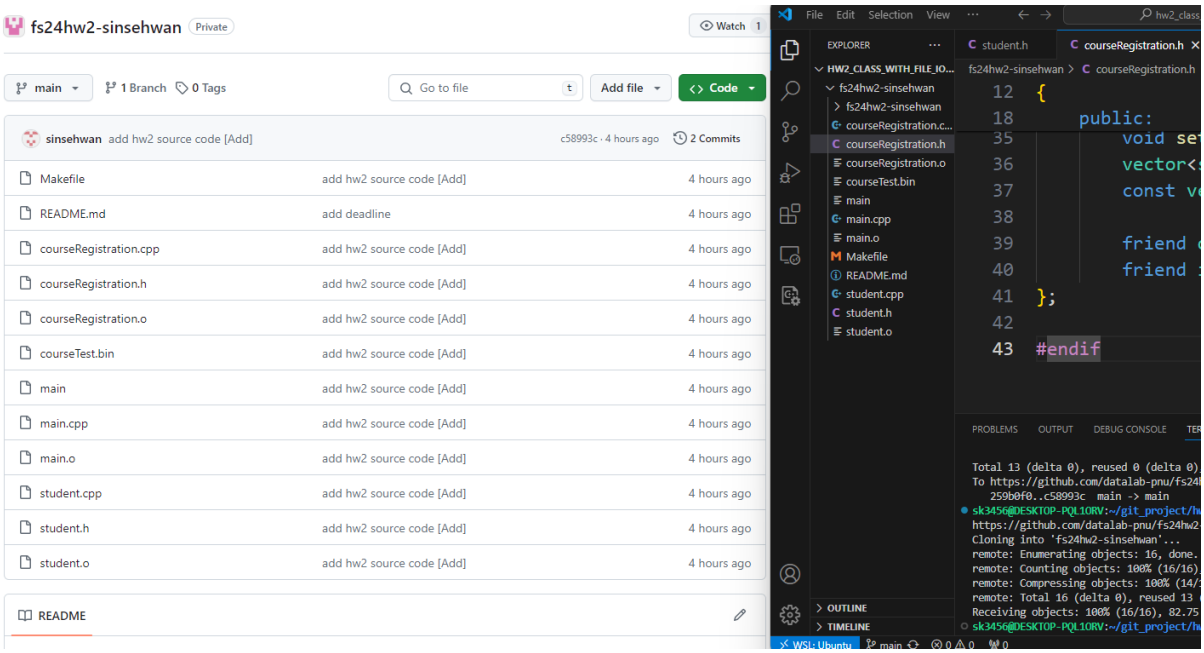
Git clone 실행

```
sk3456@DESKTOP-PQL1ORV:~/git_project$ mkdir hw1_simple_class
sk3456@DESKTOP-PQL1ORV:~/git_project$ cd hw1_simple_class
sk3456@DESKTOP-PQL1ORV:~/git_project/hw1_simple_class$ git clone https://github.com/datalab-pnu/fs24hw1-sinsehwan.git
Cloning into 'fs24hw1-sinsehwan'...
Username for 'https://github.com': sinsehwan
Password for 'https://sinsehwan@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/get-started/getting-started-with-git/about-remote-repositories#cloning-with-ht
n on currently recommended modes of authentication.
fatal: Authentication failed for 'https://github.com/datalab-pnu/fs24hw1-sinsehwan.git/'
sk3456@DESKTOP-PQL1ORV:~/git_project/hw1_simple_class$ git clone https://github.com/datalab-pnu/fs24hw1-sinsehwan.git
Cloning into 'fs24hw1-sinsehwan'...
Username for 'https://github.com': sinsehwan
Password for 'https://sinsehwan@github.com':
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
sk3456@DESKTOP-PQL1ORV:~/git_project/hw1_simple_class$ ls
fs24hw1-sinsehwan
sk3456@DESKTOP-PQL1ORV:~/git_project/hw1_simple_class$ cd fs24hw1-sinsehwan/
sk3456@DESKTOP-PQL1ORV:~/git_project/hw1_simple_class/fs24hw1-sinsehwan$ ls
README.md
sk3456@DESKTOP-PQL1ORV:~/git_project/hw1_simple_class/fs24hw1-sinsehwan$
```

Git add, commit, push 수행

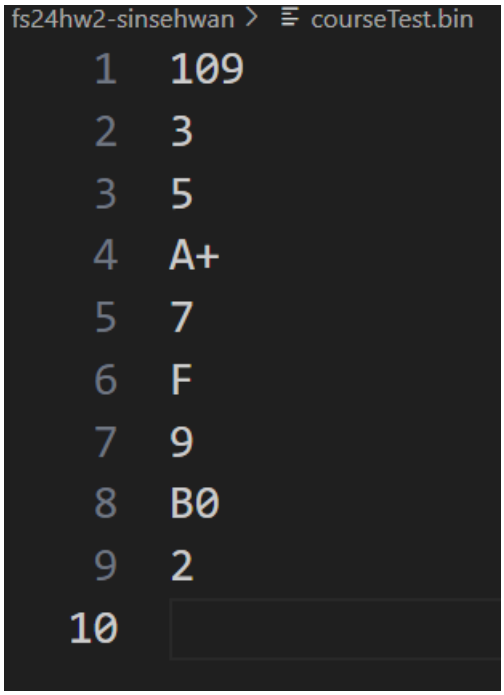
```
sk3456@DESKTOP-PQL1ORV:~/git_project/hw2_class_with_file_IO/fs24hw2-sinsehwan$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Makefile
        courseRegistration.cpp
        courseRegistration.h
        courseRegistration.o
        courseTest.bin
        main
        main.cpp
        main.o
        student.cpp
        student.h
        student.o

nothing added to commit but untracked files present (use "git add" to track)
sk3456@DESKTOP-PQL1ORV:~/git_project/hw2_class_with_file_IO/fs24hw2-sinsehwan$ git add .
sk3456@DESKTOP-PQL1ORV:~/git_project/hw2_class_with_file_IO/fs24hw2-sinsehwan$ git commit -m "add hw2 source code [Add]"
[main c58993c] add hw2 source code [Add]
 11 files changed, 422 insertions(+)
 create mode 100644 Makefile
 create mode 100644 courseRegistration.cpp
 create mode 100644 courseRegistration.h
 create mode 100644 courseRegistration.o
 create mode 100644 courseTest.bin
 create mode 100755 main
 create mode 100644 main.cpp
 create mode 100644 main.o
 create mode 100644 student.cpp
 create mode 100644 student.h
 create mode 100644 student.o
sk3456@DESKTOP-PQL1ORV:~/git_project/hw2_class_with_file_IO/fs24hw2-sinsehwan$ git push
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 16 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (13/13), 81.77 KiB | 9.09 MiB/s, done.
Total 13 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/datalab-pnu/fs24hw2-sinsehwan.git
   259b0f0..c58993c  main -> main
sk3456@DESKTOP-PQL1ORV:~/git_project/hw2_class_with_file_IO/fs24hw2-sinsehwan$
```

(2) Github repository



6. 논의 사항

- 더 나은 프로그램 작성을 위한 개선 사항,



CourseRegister클래스 객체를 파일 입출력을 통해 다룰 때 ofstream, ifstream에서 다루는 파일의 데이터 형식이 직관적이지 못하고 한 줄에 하나씩 정의된 순서대로 데이터를 단순 나열하기만 했다.

이걸 개선해서 각 데이터가 어떤 의미를 가지는 것인지 (설명) : 자료 이런식으로 json처럼 나타내는 게 더 나을 것 같은데 그렇게 작성했을 때 ifstream을 다루는 driver 함수를 어떻게 깔끔하게

디자인할 수 있을지 고민이 필요하다. 왜 json이나 xml처럼 포맷을 정해서 통신하는지 이유를 알 것 같다.
이렇게 단순한 정보의 나열로 파일에 저장하는 게 driver 함수의 가독성은 높일 수 있지만 파일 자체의 가독성을 낮추는 것이기에 코드의 가독성, 자료의 가독성 모두 높일 수 있는 방법을 생각해봐야 할 것 같다.