Do not make a separate cover, but fill in the box below with the assignment number and title, department, student ID, name and submission date. Delete this phrase when printing. If you make a separate cover, you will be deducted 10% from the total score. You should include class number (060).

.

HW3. RecordFile

Department Name: 정보컴퓨터공학부

201624548

Github ID: lejaeyun@naver.com

이재윤

Submission date: 2020-05-08

The report should include the following:

1. Class design

(1) UML class diagram

-Please include an UML class diagram for your source codes.

(2) Explanations for each class

- What are the member variables and member functions.

2. Environment

(1) Include Makefile

- Describe how to compile and execute you program

(2) Include a screen shot of your GitHub repository

3. Test Program

- Explain how your system works perfectly according to the specification.
- How can we know a fixed length record and a variable length record are well implemented?

4. Discussion (10 points)

- What you learned while doing your homework (contents other than class hours),
- Describe difficulties during homework

4. Discussion (결과에 대한 설명 이후 과제에 대한 설명이 이어질 예정입니다.)

이번 과제는 Template를 잘 사용할 수 있느냐에 대한 과제였다.
관련 내용을 홍봉희 교수님의 C++, 자료구조 수업을 통해 충분히 숙지하였기 때문에 그리 어렵지 않았다.

이전 과제에서는 iobuffer, buffile을 편한 입맛대로 수정하여 과제를 수행했다면,
이번에는 제공해 주신 iobuffer, buffile, fixfld, fixlen recfile, varlen을 그대로 사용하기로 결정하였다.

이전 과제에서 겪은 경험을 통해 관련 내용을 잘 숙지하였다고 생각하였고, 이를 통해
과제를 해 나가는데 큰 에러사항이 없었다.

클래스를 설계하는데에는 큰 어려움이 없었지만. Driver 함수를 제작하는데 가장 많은 시간이 소요되었다.

그 이유는 고정길이레코드에 아직 익숙하지 않았기 때문만은 아니고, InitBuffer를 Create 이전에 호출해
주어야 한다는 점을 몰랐고 수행해 나가면서 .dat 파일에 Header가 쓰여지지 않는 이유를 분석하느라
모든 파일을 다시 처음부터 분석하였기 때문이다. 또한 파일을 저장을 하지 않아 이유없는 출력 체크문이 자꾸 나타나 이를 알아채는데도 많은 시간이 걸렸다.

또 하나의 에러가 나타나 나를 괴롭혔는데, 그건 바로 Template를 사용하는 경우 undefined reference to ~~
에러문이다. 이는 컴파일러가 템플릿을 명확히 인식하지 못해 문제가 생기는데 이를 처리하기 위하여
다음과 같은 방법으로 명시적으로 선언해버리는 방법을 사용하였다.

```
#include "recfile.h"
#include "Student.h"
#include "CourseRegistration.h"
template class RecordFile<Student>;
template class RecordFile<CourseRegistration>;
```

이 방법을 사용한 결과 컴파일과 실행이 제대로 되었다. 아래는 컴파일 결과 스크린샷이다.

```
lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW3$ make
g++ -c -I./include -Wall   -o MySchoolTest.o MySchoolTest.cpp
make[1]: Entering directory '/home/lep/HW/FIle_Structure/HW3/buf'
g++ -fPIC -c -Wall   -I../include -o buffile.o buffile.cpp
g++ -fPIC -c -Wall   -I../include -o iobuffer.o iobuffer.cpp
g++ -fPIC -shared -Wl,-soname=libmybuffer.so.1 buffile.o iobuffer.o -o libmybuff
er.so.1.0 -lc
cp libmybuffer.so.1.0 ../lib
ln -s ../lib/libmybuffer.so.1.0 ../lib/libmybuffer.so
ln -s ../lib/libmybuffer.so.1.0 ../lib/libmybuffer.so.1
make[1]: Leaving directory '/home/lep/HW/FIle_Structure/HW3/buf'
make[1]: Entering directory '/home/lep/HW/FIle_Structure/HW3/fixed'
g++ -fPIC -c -Wall   -I../include -o fixfld.o fixfld.cpp
g++ -fPIC -c -Wall   -I../include -o fixlen.o fixlen.cpp
g++ -fPIC -shared -Wl,-soname=libmyfixed.so.1 fixfld.o fixlen.o -o libmyfixed.so
.1.0 -lc
cp libmyfixed.so.1.0 ../lib
ln -s ../lib/libmyfixed.so.1.0 ../lib/libmyfixed.so
ln -s ../lib/libmyfixed.so.1.0 ../lib/libmyfixed.so.1
make[1]: Leaving directory '/home/lep/HW/FIle_Structure/HW3/fixed'
make[1]: Entering directory '/home/lep/HW/FIle_Structure/HW3/school'
g++ -fPIC -c -Wall   -I../include -o Student.o Student.cpp
```

```
cp libmyschool.so.1.0 ../lib
ln -s ../lib/libmyschool.so.1.0 ../lib/libmyschool.so
ln -s ../lib/libmyschool.so.1.0 ../lib/libmyschool.so.1
make[1]: Leaving directory '/home/lep/HW/FIle_Structure/HW3/school'
make[1]: Entering directory '/home/lep/HW/FIle_Structure/HW3/var'
g++ -fPIC -c -Wall  -I../include -o delim.o delim.cpp
g++ -fPIC -c -Wall  -I../include -o varlen.o varlen.cpp
g++ -fPIC -shared -Wl,-soname=libmyvar.so.1 delim.o varlen.o -o libmyvar.so.1.0
-lc
cp libmyvar.so.1.0 ../lib
ln -s ../lib/libmyvar.so.1.0 ../lib/libmyvar.so
ln -s ../lib/libmyvar.so.1.0 ../lib/libmyvar.so.1
make[1]: Leaving directory '/home/lep/HW/FIle_Structure/HW3/var'
make[1]: Entering directory '/home/lep/HW/FIle_Structure/HW3/rec'
g++ -fPIC -c -Wall  -I../include -o recfile.o recfile.cpp
g++ -fPIC -shared -Wl,-soname=libmyrec.so.1 recfile.o -o libmyrec.so.1.0 -lc
cp libmyrec.so.1.0 ../lib
ln -s ../lib/libmyrec.so.1.0 ../lib/libmyrec.so
ln -s ../lib/libmyrec.so.1.0 ../lib/libmyrec.so.1
make[1]: Leaving directory '/home/lep/HW/FIle_Structure/HW3/rec'
g++ -o MySchoolTest MySchoolTest.o -L./lib -lmybuffer -lmyfixed -lmyrec -lmyscho
ol -lmyvar
export LD_LIBRARY_PATH=./lib
lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW3$
```

다음은 실행 화면이다.

```
lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW3$ ls
Makefile      MySchoolTest.cpp  buf    include  rec     var
MySchoolTest  MySchoolTest.o    fixed  lib      school
lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW3$ export LD_LIBRARY_PATH=$LD_LIBR
ARY_PATH:./lib
lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW3$
```

```
lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW3$ ./MySchoolTest
1
id : 201624548
name : lee jae yoon
address : Pusan
first_enrollment : 2034-10-15-Sun-07-21-00
credit_hours : 80

2
id : 907368179
name : KIM
address : Seoul
first_enrollment : 2031-10-06-Mon-02-13-52
credit_hours : 61
```

```
10
Class_Identifier : 527798215
Class_Credit_Hours : 3
 ##### Blow the Students #####
Student 1 : 966316183
Grade_Sum : 0

Student 2 : 1567573423
Grade_Sum : 0

Student 3 : 1000807384
Grade_Sum : 0

Student 4 : 1235632892
Grade_Sum : 0

Student 5 : 102090908
Grade_Sum : 0


lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW3$
```

```
lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW3$ ls
CourseRegistration.dat  MySchoolTest.cpp  buf      lib     var
Makefile                MySchoolTest.o    fixed    rec
MySchoolTest            Students.dat      include  school
lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW3$ cat CourseRegistration.dat
IOBuffer>240800246377947383401465642989032914490201158463948O?116983965713291449
020123789107601465642989039371689201
789107601660853233231158463948014656429890393716892033536303032914490201126747283
232016245480995652720039371689201401594171033536303OL17010004453146564298909956
527200115846394807794738340140159417103188935689112378910760140159417101158463948
0J14917820811158463948014015941710123789107607794738340335363030H19672749893329
144902099565272003353630302016245480779473834O>189632884627794738340115846394803
937168920329144902Olep@lep-Virtual-Machine:~/HW/FIle_Structure/HW3$
```

```
lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW3$ cat Students.dat
IOBufferFixed◆Field
 c

201624548lee jae yoon  Pusan  td  9 9 9  04447726080447726090736817 9KIMj
ae yoon  Seoul  td  9 9 9  94898683261898683215675734238ONGae yoon  Dae
jeon  td  9 9 9  6644048937544048931000807384Annae yoon  Jeonju  td  9
 9 9  5159148235459148231020909081ohnae yoon  Daegu  td  9 9 9  3678
763991787639912356328921lenae yoon  Pohang  td  9 9 9  27459522934459522
9966316183Yoonae yoon  Daegu  td  9 9 9  279500221695002221468588221hoia
e yoon  Ulsan  td  9 9 9  7868907044968907041895852821Moonae yoon  Busa
n  td  9 9 9  7291999226691999221723175850Sunae yoon  Dokdo  td  9 9
 9  2094864219424864219lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW3$
```

모두 이상없이 잘 실행되는것을 볼 수 있다.

1. Class Design

클래스 디자인은 이전과 달라진 부분만 설명하고자 한다.
Pack, Unpack 함수는 이전 HW#2에 구현했던 함수로 잘 구동되었다.
또한 CourseRegistration.h 파일은 여전히 가변길이 레코드를 사용하기 때문에, 추가될 내용이 없었다.
Student.h 파일은 고정길이 레코드를 사용하기 때문에 InitBuffer 함수 하나의 추가가 있었는데, 내용은 아래와 같다.

```
int Student::InitBuffer(FixedFieldBuffer & Buffer)
// initialize a FixedFieldBuffer to be used for Persons
{
        int result;
        result = Buffer.AddField (10); // int형 11자리
        result = result && Buffer.AddField (max_namelength-1); // name maxlength
        result = result && Buffer.AddField (max_addresslength-1); // Address max_addresslength
        result = result && Buffer.AddField (10); // first_enrollment int형 11자리
        result = result && Buffer.AddField (10);   // credit_hours int형 11자리
        return result;
}
```

가장 변화가 많았던 부분은 Driver 파일이고, 이에 대한 설명은 3)에서 이어질 예정이다.

2) Environment

Makefile

Makefile은 ch5.2_Buff_Recursive 를 참조하여 모든 파일을 들러 make하도록 하였고, shared library도 만들어 lib 파일에 넣도록 하였다.

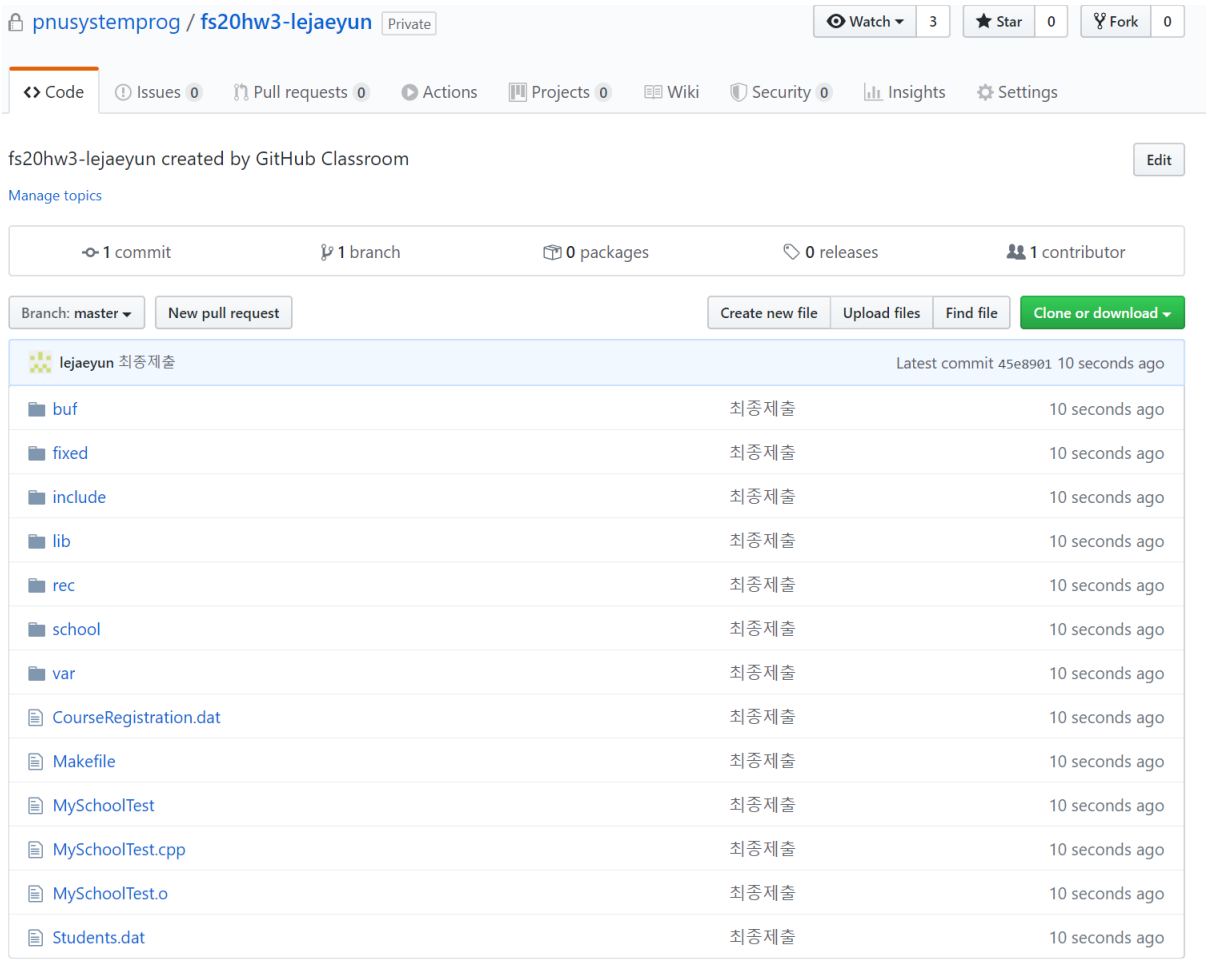
```
lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW3$ make
g++ -c -I./include -Wall  -o MySchoolTest.o MySchoolTest.cpp
make[1]: Entering directory '/home/lep/HW/FIle_Structure/HW3/buf'
g++ -fPIC -c -Wall  -I../include -o buffile.o buffile.cpp
g++ -fPIC -c -Wall  -I../include -o iobuffer.o iobuffer.cpp
g++ -fPIC -shared -Wl,-soname=libmybuffer.so.1 buffile.o iobuffer.o -o libmybuff
er.so.1.0 -lc
cp libmybuffer.so.1.0 ../lib
ln -s ../lib/libmybuffer.so.1.0 ../lib/libmybuffer.so
ln -s ../lib/libmybuffer.so.1.0 ../lib/libmybuffer.so.1
make[1]: Leaving directory '/home/lep/HW/FIle_Structure/HW3/buf'
make[1]: Entering directory '/home/lep/HW/FIle_Structure/HW3/fixed'
g++ -fPIC -c -Wall  -I../include -o fixfld.o fixfld.cpp
g++ -fPIC -c -Wall  -I../include -o fixlen.o fixlen.cpp
g++ -fPIC -shared -Wl,-soname=libmyfixed.so.1 fixfld.o fixlen.o -o libmyfixed.so
.1.0 -lc
cp libmyfixed.so.1.0 ../lib
ln -s ../lib/libmyfixed.so.1.0 ../lib/libmyfixed.so
ln -s ../lib/libmyfixed.so.1.0 ../lib/libmyfixed.so.1
make[1]: Leaving directory '/home/lep/HW/FIle_Structure/HW3/fixed'
make[1]: Entering directory '/home/lep/HW/FIle_Structure/HW3/school'
g++ -fPIC -c -Wall  -I../include -o Student.o Student.cpp
```

```
cp libmyschool.so.1.0 ../lib
ln -s ../lib/libmyschool.so.1.0 ../lib/libmyschool.so
ln -s ../lib/libmyschool.so.1.0 ../lib/libmyschool.so.1
make[1]: Leaving directory '/home/lep/HW/FIle_Structure/HW3/school'
make[1]: Entering directory '/home/lep/HW/FIle_Structure/HW3/var'
g++ -fPIC -c -Wall  -I../include -o delim.o delim.cpp
g++ -fPIC -c -Wall  -I../include -o varlen.o varlen.cpp
g++ -fPIC -shared -Wl,-soname=libmyvar.so.1 delim.o varlen.o -o libmyvar.so.1.0
-lc
cp libmyvar.so.1.0 ../lib
ln -s ../lib/libmyvar.so.1.0 ../lib/libmyvar.so
ln -s ../lib/libmyvar.so.1.0 ../lib/libmyvar.so.1
make[1]: Leaving directory '/home/lep/HW/FIle_Structure/HW3/var'
make[1]: Entering directory '/home/lep/HW/FIle_Structure/HW3/rec'
g++ -fPIC -c -Wall  -I../include -o recfile.o recfile.cpp
g++ -fPIC -shared -Wl,-soname=libmyrec.so.1 recfile.o -o libmyrec.so.1.0 -lc
cp libmyrec.so.1.0 ../lib
ln -s ../lib/libmyrec.so.1.0 ../lib/libmyrec.so
ln -s ../lib/libmyrec.so.1.0 ../lib/libmyrec.so.1
make[1]: Leaving directory '/home/lep/HW/FIle_Structure/HW3/rec'
g++ -o MySchoolTest MySchoolTest.o -L./lib -lmybuffer -lmyfixed -lmyrec -lmyscho
ol -lmyvar
export LD_LIBRARY_PATH=./lib
lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW3$
```

```
lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW3$ ls
Makefile      MySchoolTest.cpp  buf    include  rec     var
MySchoolTest  MySchoolTest.o    fixed  lib      school
lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW3$ export LD_LIBRARY_PATH=$LD_LIBR
ARY_PATH:./lib
lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW3$
```

4)에서 보인 스크린샷과 같이 Shared Library Path를 설정해준 이후 ./MySchoolTest로 실행한다.

Github Screenshot



3) Test Program (MySchoolTest.cpp)

MySchoolTest는 아예 조작이 필요없도록 제작되었다.

학생 데이터를 랜덤으로 생성하고, 수업 데이터에 랜덤으로 삽입하여 .dat 파일로 출력 및 화면에 출력한다. 아래는 학생을 랜덤으로 생성 및 저장하는 내용이다. RecordFile 템플릿을 사용하였다.

```
FixedFieldBuffer Student_Buffer(5); // Student는 5개의 클래스 변수가 존재한다.
Student S;
S.InitBuffer(Student_Buffer); // 이 부분을 처리하지 않아 고생하였다.

remove("Students.dat");
RecordFile<Student> St_Rec2file (Student_Buffer); // 템플릿 RecFile을 사용한다.
St_Rec2file.Create("Students.dat",ios::out);

srand(time(NULL));
// 데이터 생성, 저장 드라이버
S.setStudent(201624548, (char *)"lee jae yoon", (char *)"Pusan", rand(), rand() % 100 + 1);
St_Rec2file.Write(S);
S.setStudent(rand(), (char *)"KIM", (char *)"Seoul", rand(), rand() % 100 + 1);
St_Rec2file.Write(S);
S.setStudent(rand(), (char *)"BONG", (char *)"Daejeon", rand(), rand() % 100 + 1);
St_Rec2file.Write(S);
S.setStudent(rand(), (char *)"Ann", (char *)"Jeonju", rand(), rand() % 100 + 1);
St_Rec2file.Write(S);
S.setStudent(rand(), (char *)"John", (char *)"Daegu", rand(), rand() % 100 + 1);
St_Rec2file.Write(S);
S.setStudent(rand(), (char *)"Elen", (char *)"Pohang", rand(), rand() % 100 + 1);
St_Rec2file.Write(S);
S.setStudent(rand(), (char *)"Yoon", (char *)"Daegu", rand(), rand() % 100 + 1);
St_Rec2file.Write(S);
S.setStudent(rand(), (char *)"Choi", (char *)"Ulsan", rand(), rand() % 100 + 1);
St_Rec2file.Write(S);
S.setStudent(rand(), (char *)"Moon", (char *)"Busan", rand(), rand() % 100 + 1);
St_Rec2file.Write(S);
S.setStudent(rand(), (char *)"Sun", (char *)"Dokdo", rand(), rand() % 100 + 1);
St_Rec2file.Write(S);
St_Rec2file.Close();

// 저장된 데이터 불러오기 및 화면에 출력하기 (학생 10명)
Student St[10];
St_Rec2file.Open((char*)"Students.dat", ios::in);
i = 0;
while (i < 10) {
        St_Rec2file.Read(St[i]);
        cout << i+1 << endl;
        cout << St[i++] << endl;
}
St_Rec2file.Close();
```

또한 아래는 수업을 랜덤으로 생성 및 저장하는 내용이다. 이 또한 RecordFile 템플릿을 사용하였다. 위 사용했던 학생 생성 및 불러오기 이후 저장된 학생을 사용한다. HW#2와 크게 달라진점은 없다.

```
DelimFieldBuffer Courseregistration_Buffer;
remove("CourseRegistration.dat");
RecordFile<CourseRegistration> Cr_Rec2file (Courseregistration_Buffer);
Cr_Rec2file.Create("CourseRegistration.dat",ios::out);

CourseRegistration cr[10];
// 수업을 랜덤으로 생성
for (i = 0; i < 10; i++) {
        cr[i].setCourseRegistration(rand(), rand() % 3 + 1);
}
// 학생을 랜덤으로 삽입
for (i = 0; i < 10; i++) {
        for (int j = 0; j < 5; j++){
                cr[i].addstudent(St[rand()%10]);
        }
        cout << cr[i] << endl;
}
// 데이터 파일 저장 .dat 파일
for (int i = 0; i < 10 ; i++){
        Cr_Rec2file.Write(cr[i]);
}

// 저장된 데이터 불러오기
Cr_Rec2file.Open((char*)"Students.dat", ios::in);

i = 0;
while (i < 10) {
        Cr_Rec2file.Read(cr[i]);
        cout << i+1 << endl;
        cout << cr[i++] << endl;
}

Cr_Rec2file.Close();

// 데이터 누수 방지를 위해 포인터 해제
for (int i = 0; i < 10 ; i++){
        cr[i].delPointer();
}
```