컴퓨터학부 20162448 김병준

1. **개요**

수많은 세기를 거쳐오며 사람들은 복잡한 작업에 대해서 사람이 직접 해오던 일을 기계에게 맡기기 시작하였다. 그리고 정보화를 거치면서 자동화되었다. 그 중에는 학생들이 본 시험의 답안에 대해서 채점을 하는 작업도 포함할 수 있다. 대표적인 예시가 바로 광학마크인식(OMR)이다. 학생들이 답안을 용지에 컴퓨터가 인식할 수 있는 펜으로 작성을 하면, 그 작성된 답안을 컴퓨터가 인식하여 답안 테이블을 구성하고 채점을 한다. 본 과제는 학생들이 제출한 텍스트 파일과 프로그램 파일 답안에 대하여 텍스트 비교 및 프로그램 컴파일 및 실행 결과 비교를 통하여 학생들의 답안을 자동화하여 채점하는 프로그램이다.

이 프로그램은 프로그램 실행 전, 문제별로 점수를 할당할 수 있으며, 문제 별 점수를 테이블화 하여 파일에 기록한다. 그리고 프로그램 문제의 경우 스레드 옵션이 주어져야 할 경우에 대한 처리도 제공한다. 또한 프로그램 실행 후 학생들의 채점 결과를 각 문제별로 볼 수 있는 파일도 제공한다. 그리고 테이블이 생성되면 필요한 경우 해당 학생들에 대한 오답 목록을 출력한다.

1. **분석**

이 프로그램이 채점이란 동작을 진행하기 위해서는 표본이 필요하다. 그것은 학생들이 제출한 답안과 그것이 정답과 오답의 기준이 될 수 있는 답안이다. 우리는 이러한 답안들을 묶을 수 있는 특성이 필요한데 이것을 디렉토리로 묶어보고자 한다.

먼저 표준이 되는 답안의 경우 문제의 번호, 개수, 파일 유형 등 특성 세 개가 반드시 필요하다. 이것은 채점을 올바른 문제에 올바른 횟수로 빈칸문제인지 프로그램 문제인지를 구분할 수 있는 지표가 되기 때문이다. 그리고 이러한 답안 파일들을 가지고 있는 디렉토리의 경로가 반드시 필요하다. 이 경로를 우리는 명령을 실행할 때 인자로 학생, 정답 답안 디렉토리 명을 넘겨주기 때문에 해당 디렉토리 경로를 알 수 있다. 그리고 이러한 인자로 받아온 정답 답안 경로를 반드시 기억해 둘 필요가 있다.

반대로 이 프로그램의 궁극적인 목표라고 할 수 있는 학생의 답안의 경우이다. 표준 답안과 마찬가지로 문제의 번호, 개수, 파일 유형 총 세개의 특성이 반드시 필요하다. 그러나 이러한 특성들을 다시 학생들을 구분 짓는 학번으로 우리는 묶을 필요가 있다. 그렇지 않으면 우리는 답안이 누구의 답안인지 알 수 없기 때문이다. 그러므로 학생 답안의 경로 중에는 반드시 학번이 들어간 중간 경로가 필요하다.

**2-1. ssu\_score.h**

해당 헤더파일에는 main.c, ssu\_score.c에서 사용하는 함수들의 프로토타입과 전역변수들이 제시되어 있다.

1. **설계**

**3-1. ssu\_score.c**

**3-1-1. void ssu\_score(int argc, char \*argv[])**

서브 프로그램(채점) 메인 함수

**3-1-2. int check\_option(int argc, char \*argv[])**

옵션 및 인자 검사, 성공:1, 실패:0

**3-1-3. void print\_usage()**

사용법 출력 함수

**3-1-4. void score\_students()**

서브 프로그램(채점)의 모든 학생들의 채점 결과를 저장 및 출력, score.csv파일의 존재 여부 판단 및 생성해주는 함수

**3-1-5. double score\_student(int fd, char \*id)**

서브 프로그램(채점)의 각 학생의 채점 결과 판단 및 출력, score.csv에 기록해주는 함수.

**3-1-6. void write\_first\_row(int fd)**

score.csv의 제목 행 데이터를 기록해주는 함수

**3-1-7. char \*get\_answer(int fd, char \*result)**

빈칸 문제 답안에 대하여 다수의 답안이 존재할 경우, 구분하여 배열로 문자열을 저장하여 반환해주는 함수

**3-1-8. int score\_blank(char \*id, char \*filename)**

서브 프로그램(채점)의 기능 중 빈칸 문제를 채점하는 함수, 반환:점수

**3-1-9. double score\_program(char \*id, char \*filename)**

서브 프로그램(채점)의 기능 중 프로그램 문제를 채점하는 함수, 반환:점수

**3-1-10. double compile\_program(char \*id, char \*filename)**

프로그램 문제 답안 파일에 대하여 컴파일 및 에러 내용 저장 함수, 반환:점수

**3-1-11. int execute\_program(char \*id, char \*filname)**

프로그램 문제 답안 파일의 실행 결과 비교 및 채점, 반환:점수

**3-1-12. pid\_t inBackground(char \*name)**

프로그램 문제 답안 파일의 스레드 실행 확인, 성공:PID, 실패:0

**3-1-13. double check\_error\_warning(char \*filename)**

프로그램 컴파일 및 실행 결과 확인, 에러:0, 경고:원점수-0.1 실패:0

**3-1-14. int compare\_resultfile(char \*file1, char \*file2)**

학생 및 정답 프로그램 실행 결과 내용 비교, 같음:1, 다름:0

**3-1-15. void do\_iOption(char (\*ids)[FILELEN])**

-i 옵션이 존재할 경우 서브프로그램(학생들의 틀린 문제 목록 출력)의 메인 함수

**3-1-16. int get\_header\_idx(char \*header, int comma\_cnt)**

인자로 주어진 header문자열에서 comma\_cnt의 개수만큼의 쉼표 인덱스를 카운팅, 바로 다음 인덱스를 반환

**3-1-17. char \*get\_header\_char(char \*header, int idx)**

인자로 주어진 header문자열에서 idx의 위치부터 쉼표가 나오기 전까지의 문자열을 반환

**3-1-18. int is\_exist(char (\*src)[FILELEN], char \*target)**

인자로 주어진 target 학번이 src에 존재하는지 확인, 유:1, 무:0

**3-1-19. int is\_thread(char \*qname)**

-t 옵션이 존재할 경우 같이 주어진 문제 목록에 qname이 존재하는지 확인, 유:1, 무:0

**3-1-20. void redirection(char \*command, int newfd, int oldfd)**

인자로 주어진 old 디스크럽터를 new로 복사한 후, command 문자열을 터미널에서 실행, 실행 후 디스크럽터를 다시 원래대로 복구

**3-1-21. int get\_file\_type(char \*filename)**

주어진 filename의 파일 유형을 반환, txt:3, c:4, 그외:-1

**3-1-22. void rmdirs(const char \*path)**

주어진 디렉토리 경로(path)를 재귀적으로 제거

**3-1-23. void to\_lower\_case(char \*c)**

주어진 문자를 대문자라면 소문자로 변환

**3-1-24. void set\_scoreTable(char \*ansDir)**

서브 프로그램(채점)에 필요한 점수 테이블을 설정하는 메인 함수

**3-1-25. void read\_scoreTable(char \*path)**

서브 프로그램(채점)에 필요한 점수 테이블을 읽어들여서 score\_table구조체에 할당하는 함수

**3-1-26. void make\_scoreTable(char \*ansDir)**

서브 프로그램(채점)에 필요한 점수 테이블이 존재하지 않을 경우, score\_table.csv를 생성하는 함수

**3-1-27. void write\_scoreTable(char \*filename)**

score\_table.csv에 입력한 문제별 점수 정보를 파일에 작성하는 함수

**3-1-28. void set\_idTable(char \*stuDir)**

서브 프로그램(채점)에서 학생들의 학번 목록을 디렉토리에서 읽어들여 id\_table 배열에 저장하는 함수

**3-1-29. int get\_create\_type()**

score\_table.csv 파일이 존재하지 않을 경우, 점수 입력 방식을 결정하는 함수, 반환:1 | 2

**3-1-30. void sort\_idTable(int size)**

id\_table의 학생들 학번 목록을 번호순 정렬해주는 함수

**3-1-31. void sort\_scoreTable(int size)**

score\_table구조체의 문제 번호 목록을 번호순 정렬해주는 함수

**3-1-32. void get\_qname\_number(char \*qname, int \*num1, int \*num2)**

답안 파일 명(qname)을 문제번호로 구분해주는 함수, num1-num2.txt or num1.c

**3-1-33. void do\_mOption()**

-m 옵션이 존재할 경우 서브 프로그램(점수 테이블 수정)의 메인 함수

**3-1-34. int rescore(int line, double new\_score)**

인자로 주어지는 라인 카운트(line)과 수정 할 점수(new\_score)로 score\_table.csv 를 다시 작성 하는 함수, 성공:0, 실패:1

**3-1-35. void file\_remove(const char \*file, \_Bool isDir)**

인자로 주어지는 경로(file)에 해당하는 파일 혹은 디렉토리를 제거하는 함수, 디렉토리일 경우 isDir = 1

**3-1-36. void file\_copy(const char \*from, const char \*to)**

인자로 주어지는 경로 from을 경로 to로 복사

**3-2. blank.c**

**3-2-1. void compare\_tree(node \*root1, node \*root2, int \*result)**

정답, 학생답안 트리 비교, 같음:1, 다름:0

**3-2-2. node \*make\_tree(node \*root, char (\*tokens)[MINLEN], int \*idx, int parentheses)**

주어진 토큰들로 이루어진 우선순위 기반 트리 생성, 반환:최상위 노드포인터

**3-2-3. node \*change\_sibling(node \*parent)**

주어진 노드의 직계 자식의 첫째노드와 둘째노드의 위치 변경, 반환:parent

**3-2-4. node \*create\_node(char \*name, int parentheses)**

인자로 주어진 정보를 담은 노드 생성, 반환:새로운 노드포인터

**3-2-5. int get\_precedence(char \*op)**

주어진 문자에 해당하는 우선순위 반환

**3-2-6. int is\_operator(char \*op)**

주어진 문자가 연산자인지 확인, 연산자:1, 그외:0

**3-2-7. void print(node \*cur)**

주어진 노드가 부모가 되는 트리 출력

**3-2-8. node \*get\_operator(node \*cur)**

형제들의 부모 노드를 반환, 반환:첫째노드의 부모포인터

**3-2-9. node \*get\_root(node \*cur)**

트리의 최상위 노드를 반환, 반환:최상위 노드포인터

**3-2-10. node \*get\_high\_precedence\_node(node \*cur, node \*new)**

두 노드 중 우선순위가 가장 높은 연산자 노드를 반환(트리에 존재하는 연산자 노드만 리턴), 반환:트리에 존재하는 최우선 연산자 노드포인터

**3-2-11. node \*get\_most\_high\_precedence\_node(node \*cur, node \*new)**

새 연산자와 기존 연산자 노드, 형제, 부모와 우선순위 비교, 새연산자도 리턴 가능, 반환:최우선 연산자 노드포인터

**3-2-12. node \*insert\_node(node \*old, node \*new)**

노드를 삽입, 삽입한 위치 이후 노드는 자식 관계로 전환, 반환:new

**3-2-13. node \*get\_last\_child(node \*cur)**

인자로 주어진 노드의 직계 자식 중 막내 노드 반환

**3-2-14. void free\_node(node \*cur)**

주어진 노드 연결 해제

**3-2-15. int get\_sibling\_cnt(node \*cur)**

인자로 주어진 노드의 형제 개수 반환, 반환:형제 수(자기자신 미포함)

**3-2-16. int make\_tokens(char \*str, char tokens[TOKEN\_CNT][MINLEN])**

주어진 문자열 토큰화 정상: 1, 오류:0

**3-2-17. int is\_typeStatement(char \*str)**

타입정의문 유무 확인, 단순자료형이 포함된 문자열:0, 캐스팅(의미없음):1, 변수 선언(자료형으로 시작하는 문장):2

**3-2-18. int find\_typeSpecifier(char tokens[TOKEN\_CNT][MINLEN])**

형 변환을 확인하고 그에 해당하는 토큰 넘버 반환, 오류:-1

**3-2-19. int find\_typeSpecifier2(char tokens[TOKEN\_CNT][MINLEN])**

struct구조체 사용시 토큰 넘버 반환, 오류:-1

**3-2-20. int is\_character(char c)**

주어진 문자가 알파벳, 숫자인지 확인, 알파벳|숫자:1, 그 외:0

**3-2-21. int all\_star(char \*str)**

주어진 문자열이 모두 ‘\*’로 이루어졌는지 확인, 모두:1, 그 외:0

**3-2-22. int all\_character(char \*str)**

주어진 문자열 중에 알파벳 혹은 숫자가 존재할 경우 확인, 존재:1, 없음:0

**3-2-23. int reset\_tokens(int start, char tokens[TOKEN\_CNT][MINLEN])**

캐스팅 및 struct와 같은 토큰을 읽었을 경우 문법 확인 후 결과 반환, 맞음:1, 아님:0

**3-2-24. void clear\_tokens(char tokens[TOKEN\_CNT][MINLEN])**

토큰 배열 초기화

**3-2-25. int get\_token\_cnt(char tokens[TOKEN\_CNT][MINLEN])**

토큰 배열에 존재하는 토큰의 개수 반환

**3-2-26. char \*rtrim(char \*\_str)**

문자열 오른쪽 공백 제거, 반환:오른쪽 공백 제거된 문자열

**3-2-27. char \*ltrim(char \*\_str)**

문자열 왼쪽 공백 제거, 반환:왼쪽 공백 제거된 문자열

**3-2-28. void remove\_space(char \*str)**

문자열 공백 제거

**3-2-29. int check\_brackets(char \*str)**

괄호 짝 확인, 정상:1, 오류:0

**3-2-30. char\* remove\_extraspace(char \*str)**

문자열 잔여 공백 제거, 반환:풀필요한 공백이 제거된 문자열

**3-3. main.c**

**3-3-1. void ssu\_runtime(struct timeval \*begin\_t, struct timeval \*end\_t)**

서브 프로그램(채점)의 실행 시간 출력 함수

**3-3-2. int main(int argc, char \*argv[])**

메인 프로그램(실행 시간 기록)의 메인 함수, 반환:0

1. **구현**

**4-1. 수정**

**4-1-1. 전역변수**

기존에 ssu\_score( )함수 내부에서 존재하던 지역변수 saved\_path[]를 전역변수로 선언하였고, 옵션 변수들 중 pOption과 cOption을 iOptiond, mOption으로 변경하였음.

**4-1-2. void ssu\_score(int argc, char \*argv[])**

초반 -c 옵션을 체크하는 조건문을 -i옵션을 체크하는 조건문으로 변경하였음. 기존의 채점 테이블 파일이 존재할 경우 바로 틀린 문제 목록을 출력함. 그리고 마지막으로 다시 -i옵션을 체크하는 지점을 생성하여, 초반에 테이블 생성 전일 경우, 옵션을 실행하지 않고, 채점 완료 후 테이블이 생성된 후 옵션을 실행함. 또한 후반부에 채점 직전 -m 옵션을 체크하는 조건문을 반영하여 점수를 변경하는 지점을 새로 추가하였음.

**4-1-3. int check\_option(int argc, char \*argv[])**

getopt( )함수가 탐색하는 기존의 -c, -p 옵션을 제거하고, -m, -i옵션을 추가하였음.

**4-1-4. void set\_scoreTable(char \*ansDir)**

score\_table.csv파일의 위치를 $(PWD)/ANS\_DIR/score\_table.csv에서 $(PWD)/score\_table.csv로 변경

**4-1-5. viud make\_scoreTable(char \*ansDir)**

변경된 정답 디렉토리 하위에 존재하는 문제들의 위치에 맞게 정답 디렉토리 파싱 방법 변경

**4-1-6. void score\_student() … 2-3-17**

기존에 존재하던 -p 옵션 확인 조건문만 제거

**4-1-7. void score\_student(int fd, char \*id) … 2-3-18**

-p옵션 확인 조건문을 제거하고, 최종 출력문을 -p 옵션이 켜져 있을 경우 출력문으로 고정

**4-1-8. int score\_blank(char \*id, char \*filename)**

정답 빈칸 문제 채점을 위한 경로 변수 tmp를 ansDir/qname/filename에서 ansDir/filename으로 문자열 변경

**4-1-9. double compile\_program(char \*id, char \*filename)**

tmp\_f, tmp\_e의 경로를 ansDir/qname/filename, ansDir/qname/qname에서 ansDir/filename, ansDir/qname으로 변경

**4-1-10. int execute\_program(char \*id, char \*filename)**

정답 프로그램 실행을 위한 ans\_fname, tmp의 문자열을 ansDir/qname/qname에서 ansDir/qname으로 변경

**4-1-11. Makefile**

기존 Makefile에서 변경점은 make clean 명령어 시 기존에는 오브젝트 파일들과 실행 파일만 제거하였지만, 답안 프로그램 실행 후 생성되는 텍스트 파일들과 테이블 파일, 실행파일, 실행결과 파일들을 제거한다.

**4-2. 신규**

**4-2-1. void do\_iOption(char (\*ids)[FILELEN])**

학생 별 틀린 문제에 대한 목록을 출력해주는 옵션으로, score.csv파일을 참조한다. 만약 score.csv파일이 존재하지 않을 경우, iOption변수를 그대로 놔두고 리턴을 한다. 이후 학생 답안의 채점이 모두 완료되고 score.csv파일이 생성 되면그 때 다시 해당 변수를 확인 한 뒤 1일 경우 다시 옵션을 실행한다. 만약 학생, 정답 디렉토리를 인자로 주지 않고 옵션만 실행 할 경우, score.csv파일이 존재 할 경우에는 인자로 주어진 학번이 틀린 문제 목록을 리스트로 출력하고, 해당 옵션 변수를 0으로 바꾼 뒤 반환한다. 반대로 파일이 존재하지 않을 경우

**4-2-2. int get\_header\_idx(char \*header, int comma\_cnt)**

이 함수는 score.csv파일에서 첫번째 행을 콤마(,)를 기준으로 문자열 인덱스를 결정하며, 인자로 주어지는 첫번째 행 문자열 header와 쉼표 개수 comma\_cnt를 통해 해당하는 문자열 시작 인덱스를 반환한다. 쉼표 개수 만큼 카운트 하지 못했거나, 쉼표가 존재하지 않을 경우 문자열의 시작 인덱스를 반환한다.

**4-2-3. char \*get\_header\_char(char \*header, int idx)**

이 함수는 score.csv파일에서 첫번째 행을 콤마(,)를 기준으로 문자열 인덱스를 결정하며, 인자로 주어지는 idx를 기준으로 해당하는 문제 파일명 문자열을 반환한다.

**4-2-4. void do\_mOption()**

기존에 등록한 문제들에 대한 점수 수정기능을 제공해주는 옵션으로, score\_table.csv파일을 참조한다. 해당 옵션은 score\_table.csv파일이 생성되고 난 뒤 옵션 확인을 하며, 해당하는 문제 번호의 점수를 수정 할 경우, 기존의 점수를 출력해주고 새로운 점수를 입력 받은 뒤, score\_table.csv파일을 새로 작성한다. 문제 수정은 개수 제한 없이 이루어 질 수 있으며, “no”를 입력하면 종료한다.

**4-2-5. int rescore(int line, double new\_score)**

이 함수는 score\_table.csv파일을 참조하여 인자로 주어진 line 변수를 통해 해당하는 라인 전까지 score\_table.csv파일을 복사하고, 해당되는 라인에서는 새로운 점수 값을 가지고 있는 new\_score을 쓴다. 이후 다시 파일의 끝까지 내용을 복사한다.

**4-2-6. void file\_remove(const char \*file, \_Bool isDir)**

인자로 주어진 file변수로 rm file\_path명령어 문자열을 만들어 실행하며, \_isDir 변수를 통해 디렉토리일 경우 -r옵션을 준다.

**4-2-7. void file\_copy(const char \*from, const char \*to)**

인자로 주어진 from과 to변수로 cp from to 명령어 문자열을 만들어 실행한다.

1. **테스트 및 결과**

**5-1. 점수 테이블 파일 생성 (상:1번, 하:2번)**

1. **소스코드**

**6-1. Makefile**