전자정보공학부 it융합학과 20180622 천범수

1.개요

기존 제공된 빈칸 채우기 문제와 프로그램 작성 문제를 채점하는 프로그램에 일부 옵션을 추가하여 기능을 추가한다. 실제로 구현한 옵션은 -n , -m , -c , -p , -t , -e, -h 옵션이다. 기본 채점 프로그램은 옵션 없이 실행되는 경우 score\_table이 없음을 출력하고 각 문제 유형 별 점수를 타입별로 입력받도록 지정한다. 그럼 STD 디렉토리 하위에 있는 학번 디렉토리순서대로 학생들의 정답과 ANS 디렉토리의 정답을 비교하고 실행시키며, ANS 디렉토리 밑의 문제번호의 실행파일과 그 결과를 저장한다. 저장된 파일들은 정답과 비교하여 유사 정답이 허용되는경우와 그렇지 않은 경우를 포함하여 채점하고 점수를 score.csv 파일로 저장한다. 위 파일은 ANS 디렉토리 하위에 저장되며 n옵션읠 경우 n옵션 인자로 받게되는 arg의 이름에 따라 저장된다. 학번 별로 채점이 끝나면 지정된 문자열을 출력하고 모든 학생들의 채점이 끝난 경우, \*.csv 파일이 저장된 위치, 런타임을 출력하고 프로그램을 종료한다. m옵션의 경우 일전에 지정되어있던 문제의 배점을 수정하는 옵션이다. 추가 arg 없이 단독 옵션으로 지정되며 문제 번호를 입력받아 해당번호의 현재 배점과 수정할 배점을 입력받는다. 입력이 ‘no’일때까지 입력을 받으며 그럴 경우 모든 학생들의 점수를 재채점한다. -c 옵션의 경우 인자로 입력받은 학생들의 점수를 출력하고 해당 학생들의 평균을 구하는 옵션이다. 이때 최대로 받을 수 있는 가변인자는 5개이고 그 이상을 받을 시, 상위 5개의 인자에 대해서만 프로그램을 진행한다. p옵션의 경우 인자로 받은 학생들의 틀린 문제를 링크드 리스트로 관리하여 출력한다. c옵션과 p옵션 모두가변인자 개수제한이 넘거나 인자로 받은 학생의 정보가 없다면 에러를 출력하고 프로그램을 종료한다. c옵션과 p옵션은 동시에 사용할 수 있다. 두 옵션이 동시에 사용되는 경우 학생의 점수를 먼저 출력하며, 틀린 문제에대한 링크드 리스트 정보는 그 후에 붙여 출력한다. -t옵션은 -lpthread에 대해 추가하는 옵션이고, -e옵션의 경우 에러가 발생한 문제에 대해 에러 내용을 저장하는 옵션이며, -h 옵션은 usage를 출력하는 옵션이다.

2.구현 기능

아래는 함수의 프로토 타입이다.

struct ssu\_scoreTable {

char qname[FILELEN];

double score;

};

struct wrong\_problem {

char \*qname;

struct wrong\_problem \*next;

};

void ssu\_score(int argc, char \*argv[]);

int check\_option(int argc, char \*argv[]);

void print\_usage();

void score\_students();

double score\_student(int fd, char \*id, int flag);

void write\_first\_row(int fd);

char \*get\_answer(int fd, char \*result);

int score\_blank(char \*id, char \*filename);

double score\_program(char \*id, char \*filename);

double compile\_program(char \*id, char \*filename);

int execute\_program(char \*id, char \*filname);

pid\_t inBackground(char \*name);

double check\_error\_warning(char \*filename);

int compare\_resultfile(char \*file1, char \*file2);

void do\_iOption(char (\*ids)[FILELEN]);

void do\_mOption();

int is\_exist(char (\*src)[FILELEN], char \*target);

int is\_thread(char \*qname);

void redirection(char \*command, int newfd, int oldfd);

int get\_file\_type(char \*filename);

void rmdirs(const char \*path);

void to\_lower\_case(char \*c);

void set\_scoreTable(char \*ansDir);

void read\_scoreTable(char \*path);

void make\_scoreTable(char \*ansDir);

void write\_scoreTable(char \*filename);

void set\_idTable(char \*stuDir);

int get\_create\_type();

void sort\_idTable(int size);

void sort\_scoreTable(int size);

void get\_qname\_number(char \*qname, int \*num1, int \*num2);

typedef struct node{

int parentheses;

char \*name;

struct node \*parent;

struct node \*child\_head;

struct node \*prev;

struct node \*next;

}node;

typedef struct operator\_precedence{

char \*operator;

int precedence;

}operator\_precedence;

void compare\_tree(node \*root1, node \*root2, int \*result);

node \*make\_tree(node \*root, char (\*tokens)[MINLEN], int \*idx, int parentheses);

node \*change\_sibling(node \*parent);

node \*create\_node(char \*name, int parentheses);

int get\_precedence(char \*op);

int is\_operator(char \*op);

void print(node \*cur);

node \*get\_operator(node \*cur);

node \*get\_root(node \*cur);

node \*get\_high\_precedence\_node(node \*cur, node \*new);

node \*get\_most\_high\_precedence\_node(node \*cur, node \*new);

node \*insert\_node(node \*old, node \*new);

node \*get\_last\_child(node \*cur);

void free\_node(node \*cur);

int get\_sibling\_cnt(node \*cur);

int make\_tokens(char \*str, char tokens[TOKEN\_CNT][MINLEN]);

int is\_typeStatement(char \*str);

int find\_typeSpecifier(char tokens[TOKEN\_CNT][MINLEN]);

int find\_typeSpecifier2(char tokens[TOKEN\_CNT][MINLEN]);

int is\_character(char c);

int all\_star(char \*str);

int all\_character(char \*str);

int reset\_tokens(int start, char tokens[TOKEN\_CNT][MINLEN]);

void clear\_tokens(char tokens[TOKEN\_CNT][MINLEN]);

int get\_token\_cnt(char tokens[TOKEN\_CNT][MINLEN]);

char \*rtrim(char \*\_str);

char \*ltrim(char \*\_str);

void remove\_space(char \*str);

int check\_brackets(char \*str);

char\* remove\_extraspace(char \*str);

프로젝트 상 수정이 필요하여 직접 구현, 수정한 함수는 크게 4가지였다. 첫 번째로 score\_student 함수, score\_students 함수, check\_option 함수 그리고 링크드 리스트 관리를 위한 struct wrong\_problem 구조체와 reverse\_list struct func이다.

1. check\_option()

함수에서 인자로 받은 값들 중 옵션의 여부를 체크하고 옵션별로 상이한 코드를 실행한다. 특히 옵션 이후 인자들을 활용하기 위해서는 지정값에 콜론(:)을 붙여서 활용하는데 c 옵션의 경우 인자가 없는 경우와 인자가 있는 경우를 모두 처리해야하므로 콜론을 지정하지 않고 argv 배열에 입력되는 인자들의 인덱스 값을 조정하여 설계하였다. 위 설명은 p옵션도 동일하다. c옵션과 p옵션에서 받는 가변인자의 값은 5개를 초과할 수 없으며 해당부분을 별도로 관리하기 위해 초과되는 부분은 별도의 배열에 저장하여 출력에만 관여한다. 또한 받은 인자의 개수를 저장하여 추후 score\_students() 함수에서 평균을 구하기 위한 값으로 사용한다.

2. score\_student()

이 함수는 각 학생들에 대해 문제 항목별로 정답을 채점하는 함수이다. 이 함수는 p옵션에서 사용되기 위한 틀린문제에 대한 정보를 링크드 리스트의 형태로 관리하고 있으며 새 노드를 동적으로 할당한다. 또한 출력이 완료된 경우 동적 할당된 메모리를 해제한다. 각 옵션 별로 출력해야하는 값들이 다르므로 옵션 별로 다르게 지정한다. 이 때 score\_students 함수에서 바뀐 flag 값과 check\_option 함수에서 변화된 arg\_flag 값들에 따라 출력 결과물이 달라지기도 한다. flag값은 사용자가 입력한 학번이 학생학번테이블에 존재하는 지 확인하는 플래그이고, arg\_flag는 c옵션과 p옵션을 인자 없이 실행 시 발생하는 플래그이다.

3.score\_students()

이 함수는 위 함수를 통해 채점한 학생들의 점수를 합하여 평균을 구하고 출력하는 프로그램이다. 이때 flag의 지정을 위해 입력받은 check\_option 함수에서 저장하였던 배열들과 id\_table의 학번 값들을 비교하여 같은 학번이 있는 경우에만 flag를 지정하고 아닌경우, 학번이 존재하지 않음을 출력하고 프로그램을 종료한다.

3.상세 설계

프로그램의 전체적인 흐름도는 다음과 같다.

도표이(가) 표시된 사진

자동 생성된 설명

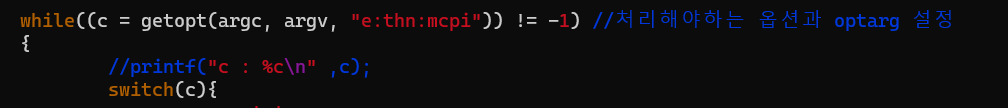
main함수가 실행되면 gettimeofday 함수로 실행시간을 출력한다. 이 코드는 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

가장 중요한 부분인 ssu\_score함수로 들어가면 우선적으로 check\_option 함수를 통해 옵션을 확인하고 옵션이 확인되지 않으면 프로그램을 종료한다. 그 후 각 옵션 별로 함수를 실행하거나 분기를 나누어 동작한다.

1. check\_option( )



getopt 함수를 사용하여 인자를 처리하고 switch case 절을 활용하여 각 옵션별 분기를 설정한다.

n옵션의 경우 optarg의 뒷 네자리 문자열이 .csv가 아니면 에러처리하고 프로그램을 종료한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

m옵션의 경우 따로 do\_mOption 함수를 활용하였다.



텍스트이(가) 표시된 사진

자동 생성된 설명

P 옵션의 경우 인자들을 순회하며 인자값들을 그 개수에 따라 p\_threadFiles와 p\_noThreadFiles에 저장한다. 개수가 5개가 넘어가는 경우 출력을 한후 5개 까지의 인자만을 출력하도록 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

c옵션의 경우도 유사하게 진행한다. 다른 점은 첫 인자로 -p가 들어왔을 경우 p옵션을 처리하는 부분이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

2.score\_students( )

위 함수는 각 학번별로 채점을 하고 조건별로 각자에 맞는 결과를 출력하는 함수이다. 실질적인 채점은 flag 별 score\_student( )함수를 호출하여 진행하고 위 함수는 전달받은 score값과 옵션별 평균, 결과등을 printf( ) 함수로 출력한다. n과 e옵션의 경우 상대경로 및 절대경로를 모두 처리할 수 있도록 하였고, c 와 p 옵션의 경우 인자 값에 따른 flag를 설정하여 flag 별로 score\_student( ) 함수를 호출하도록 하였다. 해당 함수에 대한 흐름도는 다음과 같다.

도표이(가) 표시된 사진

자동 생성된 설명

3. score\_student

Score\_student 함수는 각 학번 별로 전체 문제를 채점하는 함수이다. 이 함수는 get\_file\_type 함수로 문제유형을 받고 각 문제 유형별로 result값에 score\_blank을 호출받아 틀렸을 경우 링크드 리스트로 틀린 문제를 관리한다. Result가 true인 경우만 해당하여 score\_table의 score값을 score에 추가하고 결과를 리턴한다. 그 후 각 옵션별로, 하나의 학생에 대한 채점이 종료되면 score값을 출력할지, wrong problem을 출력할지 결정한다. 해당 함수의 흐름도는 다음과 같다.

도표이(가) 표시된 사진

자동 생성된 설명

4. 실행결과 및 분석

첫 번째로 프로그램을 실행시키면 아래 그림과 같이 점수 지정 방식을 선택하도록 입력을 대기한 후 지정 방식대로 선택을 한 후 점수를 입력받는다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1번 항목 선택 : blank, program 문제 한번에 배점 지정

텍스트이(가) 표시된 사진

자동 생성된 설명

2번 항목 선택 모든 문제 각자 배점 지정

이렇게 되면 ANS 디렉토리 하위에 score.csv 와 score\_table.csv파일이 생성된다.

텍스트, 벽, 그린, 은이(가) 표시된 사진

자동 생성된 설명

-n옵션으로 new\_score.csv 인자로 지정하면,

텍스트이(가) 표시된 사진

자동 생성된 설명

동일하게 채점을 진행하고 new\_score.csv파일로 저장되었음을 출력한다.

텍스트, 야외, 그린, 은이(가) 표시된 사진

자동 생성된 설명

최초 실행한 score.csv파일 외에 new\_score.csv파일이 생겼음을 볼 수 있다.

또한 csv파일이 아닌 다른 확장자로 생성할 경우 에러를 출력하고 프로그램을 종료한다..



다음으로 m 옵션의 경우 다음과 같이 지정된 문제 번호의 점수를 바꿀 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

c옵션의 경우

단독으로 사용될 시 모든 학생들의 점수를 출력하고 가변인자를 통해 지정할 경우 해당 학번만 출력한다. 이때 가변인자는 5개로 제한하고 가변인자와 일치하는 학번이 존재하지 않으면 에러를 출력 후 프로그램을 종료한다.

텍스트이(가) 표시된 사진

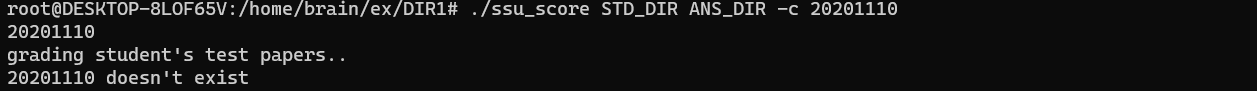
자동 생성된 설명

1. 단독 사용경우

텍스트이(가) 표시된 사진

자동 생성된 설명

5개 미만의 가변인자 경우



3. 가변인자에 해당하는 학번이 없는 경우

텍스트이(가) 표시된 사진

자동 생성된 설명

4. 가변인자가 5개를 넘는 경우

p옵션

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 단독 사용의 경우



2. 가변인자가 학번 내에 존재하지 않는 경우

텍스트이(가) 표시된 사진

자동 생성된 설명

3. 가변인자가 5개를 넘는 경우

텍스트이(가) 표시된 사진

자동 생성된 설명

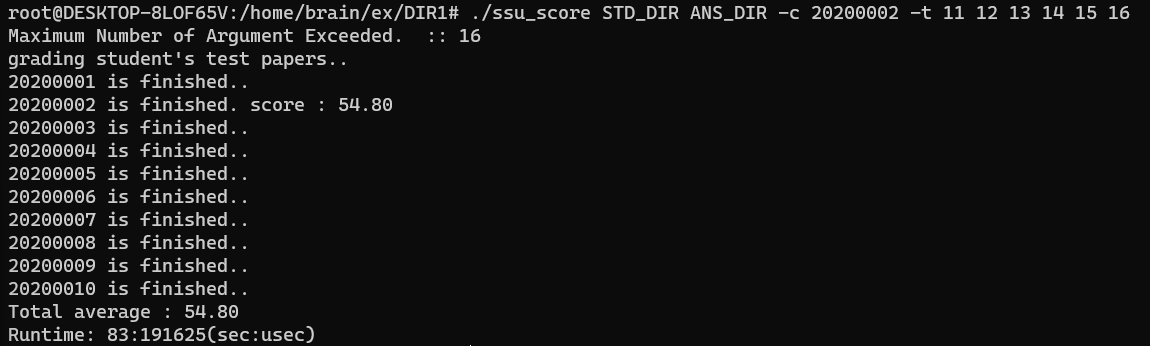
4. 가변인자가 5개 이하인 경우

c옵션과 p옵션이 동시에 사용된 경우 점수를 먼저 출력하고 틀린문항을 이어서 출력한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

t옵션



e옵션

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

E 옵션은 절대경로로 입력받은 경우에도 실행가능하다.

텍스트이(가) 표시된 사진

자동 생성된 설명

h옵션

텍스트이(가) 표시된 사진

자동 생성된 설명