**심화프로그래밍**

나만의 타자연습 프로그램 최종 보고서



**과 목 명** 심화프로그래밍

**담당교수** 윤성림 교수님

**학 과**  컴퓨터공학과

**학 번** 2022111923

** 이 름** 서하은

**제출일자** 2023-05-30

**목 차**

**Ⅰ. 프로젝트 개요**

1. 프로젝트 목표

2. 프로젝트 내용

**Ⅱ. 프로그램 구성**

1. 프로그램 설계 및 알고리즘

2. 소스코드 및 주석

**Ⅲ. 프로그램 결과**

1. 프로그램 실행 결과

2. 한계점 및 개선방안

**Ⅰ. 프로젝트 개요**

1. 프로젝트 목표

|  |
| --- |
| C++을 사용하여 나만의 타자연습 프로그램을 만든다. 관리자는 타자연습을 위한 단어/문장 데이터를 추가, 수정, 삭제할 수 있으며, 사용자는 타자연습과 게임을 진행하고 자신의 기록과 순위를 볼 수 있다. |

2. 프로젝트 내용

|  |
| --- |
| 프로그램이 실행되면 관리자 또는 사용자 권한을 선택할 수 있다. 관리자 권한을 선택할 경우 비밀번호를 입력해야 하며, 틀릴 경우 들어갈 수 없다. 단어/문장 데이터를 추가, 수정, 삭제, 그리고 회원 정보 삭제를 할 수 있다. 사용자는 낱말, 짧은글, 긴글 연습, 영타 게임1, 영타 게임2를 할 수 있다. 연습 결과는 파일에 기록되지 않으며 게임 결과만 파일에 저장된다. 기존 회원이라면 회원 이름으로 된 파일에 게임 결과가 저장되고, 신규 회원이라면 새로운 파일이 생성되어 그곳에 결과가 저장된다. 게임 결과와 함께 순위 또한 알 수 있다. |
| 관리자로 접근한 경우 |
| 사용자로 접근한 경우 |

**Ⅱ. 프로그램 구성**

1. 프로그램 설계 및 알고리즘

|  |
| --- |
| 총 3개의 파일 Typing.h, TypingMain.cpp, Windows.h 을 사용했다. Typing.h 에는 Typing 클래스와 멤버 변수, 함수들이 들어있고 TypingMain.cpp 에는 main 함수가 들어있다. Windows.h 에는 글자색 변경을 위해 필요한 소스코드가 들어있다.  Typing 클래스 안에 회원 정보와 파일에 있는 단어, 문장 정보, 그리고 연습 또는 게임 결과 정보를 저장할 멤버 변수들을 선언하고 생성자, 15개의 멤버 함수를 만든다. 단어나 문장을 추가, 수정, 삭제하거나 연습, 게임을 하기 위한 함수들이다. 세부적인 내용은 다음과 같다. |

|  |
| --- |
| setId() – 회원 정보 저장  사용자가 입력한 닉네임을 id 변수에 저장한다. |
| addData(), modData(), delData() – 단어/문장 데이터 추가, 수정, 삭제  여러 한글 단어, 문장들이 저장되어 있는 파일 word.txt와 sentence.txt가 존재하는데, 관리자는 원하는 파일을 열어서 데이터 추가, 수정, 삭제를 수행할 수 있다. 단어 추가를 원한다면 먼저 새로운 단어를 입력 받는다. word.txt 파일을 열고 기존의 내용 뒤에 이 단어를 덧붙여 쓴다.(ios::app 사용) 단어를 수정하려면 먼저 기존 단어와 수정할 단어를 입력 받는다. 그리고 word.txt 파일을 열어 그 안에 있는 단어를 하나씩 가져오는데, 이 단어들을 새로운 string에 저장하다가 해당 단어를 수정하고 이것을 다시 파일에 작성하는 방식으로 진행한다. 단어를 읽어올 때 수정할 단어가 아니라면 파일에 있는 단어 그대로 string에 추가하고 수정할 단어가 발견되면 기존 단어 대신 입력 받은 단어를 string에 추가한다. 단어를 삭제하려면 수정하는 방식과 동일하게 단어들을 새로운 string에 저장하여 파일에 다시 작성하는 방식을 사용한다. 관리자에게 삭제할 단어를 입력 받고 word.txt 파일을 열어 그 단어만 제외한 내용을 따로 저장하고 다시 파일에 작성한다. 문장 데이터를 추가, 수정, 삭제를 할 때도 마찬가지로 sentence.txt 파일을 열어 동일하게 수행한다. |
| wordTest(), shortTest(), longTest() – 낱말 연습, 짧은글 연습, 긴글 연습  낱말 연습을 위해 word.txt에서 단어를 하나씩 가져온다. 연습이 시작되면 그때부터 시간이 측정된다. 단어는 공백으로 구분되어 있으므로 공백을 기준으로 단어를 읽어오고 출력한다. 사용자는 출력된 단어를 보고 그대로 따라치는데, 입력한 단어와 맞은 단어를 카운팅한다. q가 입력되면 연습이 끝나고 시간 측정이 종료된다. 짧은글 연습의 경우 sentence.txt 에서 한 문장씩 가져오는데, 줄바꿈을 기준으로 문장들을 가져온다. 그리고 낱말 연습과 동일하게 시간 측정과 단어 카운팅이 된다. 긴글 연습은 longscript.txt 에서 줄바꿈을 기준으로 한 문장씩 가져오고 짧은글 연습과 동일하게 진행된다. |
| process, wrong, accuracy, duration – 진행도, 오타수, 정확도, 경과 시간  연습이 시작되고 낱말이나 문장이 출력되면 사용자는 그대로 입력한다. 시작과 동시에 시간이 측정되며, 정해진 횟수만큼 연습이 진행된 후 종료되면 경과 시간이 나온다. 이때 오타수를 체크하고 정확도를 계산하여 회원 파일에 저장한다. 단어 기준으로, 진행도는 (입력한 단어 개수) /(전체 단어 개수) \* 100, 오타수는 입력한 단어 개수 – 맞은 단어 개수, 정확도는 (맞은 단어 개수) / (입력한 단어 개수) \* 100으로 계산한다. 중간에 q가 입력되거나 정해진 횟수만큼 반복이 진행되면 연습이 종료되며, 그만큼 진행도에 반영된다. |
| countWordLine() – 전체 단어/문장 수 카운팅  진행도를 계산하기 위해 전체 단어/문장 수를 카운팅한다. |
| removeFile(), removeScore() – 관리자 버전  관리자는 회원 정보 삭제(filesystem::remove 사용)와 점수 기록 삭제(clear 사용)를 할 수 있다. |
| showRecord() – 사용자 버전  사용자는 회원 정보가 기록된 파일을 통해 그동안 했던 게임 결과를 조회할 수 있다. |
| printResult() – 게임 결과 출력 및 닉네임으로 회원 정보(게임 결과 기록) 저장  사용자가 연습이나 게임을 할 때마다 결과가 출력된다. 게임을 할 경우 게임 결과는 회원의 파일에도 저장되는데, 기존에 닉네임으로 로그인하여 게임을 한 적이 있는 회원이라면 닉네임으로 저장된 파일을 열고 결과를 저장한다. 그리고 처음 로그인을 하는 새로운 회원이라면, 입력한 닉네임을 파일 이름으로 하는 txt 파일이 생성되고 그 파일에 결과가 저장된다. 또한 score.txt 파일에 저장된 다른 회원들의 게임 점수와 비교하여 회원의 순위가 출력된다. 그리고 관리자 권한으로 들어오면, 저장된 회원 정보 파일들을 삭제할 수 있다.(filesystem::remove 사용) |
| printResult() – 점수 계산, 순위 출력 및 저장  연습이나 게임이 끝난 후 진행도, 오타수, 정확도, 경과 시간을 고려하여 점수를 계산한다. 이때 진행도 20%, 정확도 50%, 경과 시간 40%를 반영하여 점수를 계산하고 출력한다. 그리고 게임의 경우 회원 파일에 결과를 저장한다. score.txt에 저장된 모든 회원들의 게임 점수들과 해당 회원의 점수를 비교하여 순위를 매긴 후 출력한다. 게임을 할 때마다 순위는 score.txt에 저장된다. |
| reverseGame() – 영타 게임1 : 거꾸로 읽기 게임(난이도 하)  문장에 있는 단어들이 거꾸로 뒤집혀서 출력된다. 그 단어를 보고 다시 원본 단어로 바꾸어 타자를 치는 방식으로 진행된다. 단어나 문장을 그대로 따라치는 연습과 달리 거꾸로 뒤집어 나온 영어를 보고 다시 뒤집어 원본으로 만든 후 타이핑해야 하기 때문에 난이도가 있다.  ex) 출력 : olleH ym eman si nueah  입력 : Hello my name is haeun |
| guessWord() – 영타 게임2 : 단어 맞추기 게임(난이도 상)  단어에 있는 문자의 순서가 섞인 채로 출력된다. 랜덤으로 섞이며, 섞여있는 문자를 다시 조합해서 원래의 단어를 만든다. 단어의 첫글자는 그대로 나오며 이를 통해 유추해서 풀 수 있다.  ex) 출력 : alppe  입력 : apple |
| textcolor() – 글자색 변경  Windows.h 를 include 하여 터미널에서 강조하고자 하는 부분의 글자색을 변경한다. |

2. 소스코드 및 주석

|  |
| --- |
| Typing.h |
| #include <fstream>  #include <string>  #include <sstream>  #include <ctime>  #include <filesystem>  #include "Windows.h"  using namespace std;  class Typing {  string id; // 회원 닉네임  double totalWord, totalLine, totalLong, totalGame1, totalGame2; // 파일에 있는 총 단어(문장) 개수  double completed, correct, duration; // 입력한 단어(문장) 개수, 맞은 단어(문장) 개수, 경과 시간  double score, lank; // 점수, 순위  double process, wrong, accuracy, trans\_duration; // 진행도, 오타수, 정확도, 비율에 맞게 조정된 경과 시간  public:  Typing(); // 생성자  void setId(string id); // 닉네임 저장  void addData(string data, string filename); // 단어/문장 데이터 추가  void modData(string origin, string data, string filename); // 단어/문장 데이터 수정  void delData(string data, string filename); // 단어/문장 데이터 삭제  void removeFile(string id); // 회원 정보 삭제  void removeScore(); // 점수 기록 삭제  void countWordLine(string word, string line, string longline,  string game1, string game2); // 파일에 있는 전체 단어/문장 개수 카운팅  void wordTest(); // 낱말 연습  void shortTest(); // 짧은글 연습  void longTest(); // 긴글 연습  void showRecord(string idtxt); // 회원 정보 조회  void reverseGame(); // 거꾸로 읽기 게임  void guessWord(); // 단어 만들기 게임  void printResult(string wordORline); // 연습/게임 기록 출력, 순위 계산, 파일에 게임 기록 저장  void textcolor(int foreground); // 글자색 변경  };  Typing::Typing() {  // 멤버 변수 초기화  this->totalWord = 0;  this->totalLine = 0;  this->totalLong = 0;  this->totalGame1 = 0;  this->totalGame2 = 0;  this->completed = 0;  this->correct = 0;  this->duration = 0;  this->score = 0;  this->lank = 0;  this->process = 0;  this->wrong = 0;  this->accuracy = 0;  this->trans\_duration = 0;  }  void Typing::setId(string id) {  this->id = id;  }  // 단어/문장 데이터 추가  void Typing::addData(string data, string filename) {  ofstream ioFile(filename, ios::app); // 기존의 파일 뒤에 새로운 내용 작성 모드로 파일 열기  if (filename == "word.txt") { // 단어 파일이라면  ioFile << data << " "; // 파일에 단어와 공백 작성  }  else if (filename == "sentence.txt") { // 문장 파일이라면  ioFile << data << "\n"; // 파일에 문장과 줄바꿈 작성  }  ioFile.close(); // 파일 닫기  }  // 단어/문장 데이터 수정  void Typing::modData(string origin, string data, string filename) {  ifstream inputFile(filename); // 파일 읽기 모드  if (!inputFile.is\_open()) { // 파일이 열리지 않는다면  cout << "Failed to open the file." << endl; // 오류메시지 출력  }    string word, line;  string result; // 단어를 수정하여 파일에 새로 작성할 내용  if (filename == "word.txt") { // 단어 파일이라면  while(inputFile >> word) { // 한 단어씩 읽어오기  if (word != origin) result += word + " "; // 수정할 단어가 아니라면 result에 덧붙임  else result += data + " "; // 수정할 단어가 발견되면 수정 후 result에 덧붙임  }  }  else if (filename == "sentence.txt") { // 문장 파일이라면  while(getline(inputFile, line)) { // 한 줄씩 읽어오기  if (line != origin) result += line + "\n"; // 수정할 문장이 아니라면 result에 덧붙임  else result += data + "\n"; // 수정할 문장이 발견되면 수정 후 result에 덧붙임  }  }  inputFile.close(); // 파일 닫기  ofstream outputFile(filename); // 파일 쓰기 모드  outputFile << result; // 수정하고 재작성한 내용을 파일에 작성  outputFile.close(); // 파일 닫기  }  // 단어/문장 데이터 삭제  void Typing::delData(string data, string filename) {  ifstream inputFile(filename); // 파일 읽기 모드  if (!inputFile.is\_open()) { // 파일이 열리지 않는다면  cout << "Failed to open the file." << endl; // 오류메시지 출력  }    string word, line;  string result;  if (filename == "word.txt") { // 단어 파일이라면  while(inputFile >> word) { // 한 단어씩 읽어오기  if (word != data) result += word + " "; // 삭제할 단어가 아니라면 result에 덧붙임  }  }  else if (filename == "sentence.txt") { // 문장 파일이라면  while(getline(inputFile, line)) { // 한 줄씩 읽어오기  if (line != data) result += line + "\n"; // 삭제할 문장이 아니라면 result에 덧붙임  }  }  inputFile.close(); // 파일 닫기  ofstream outputFile(filename); // 파일 쓰기 모드  outputFile << result; // 삭제하고 재작성한 내용을 파일에 작성  outputFile.close(); // 파일 닫기  }  // 회원 정보 삭제  void Typing::removeFile(string id) {  string file = id + ".txt"; // 회원 이름으로 된 파일  ifstream inputFile(file); // 파일 열기  // 해당 회원이 존재하지 않는다면  if (!inputFile.is\_open()) cout << "삭제할 회원 정보가 없습니다!\n";  else auto exist = filesystem::remove(file); // 회원의 게임 기록이 저장된 파일 삭제  }  // 점수 기록 삭제  void Typing::removeScore() {  ofstream scorefile("score.txt"); // 파일 쓰기 모드  scorefile.clear(); // 파일에 있는 내용 지우기  scorefile.close(); // 파일 닫기  }  // 파일에 있는 전체 단어/문장 개수 카운팅  void Typing::countWordLine(string word, string line, string longline, string game1, string game2) {  ifstream wordfile(word), linefile(line), longfile(longline), game1file(game1), game2file(game2); // 파일 읽기 모드  string data;  while (wordfile >> data) totalWord++; // 한 단어씩 읽으면서 카운팅  while (getline(linefile, data)) totalLine++; // 한 줄씩 읽으면서 카운팅  while (getline(longfile, data)) totalLong++; // 한 줄씩 읽으면서 카운팅  while (getline(game1file, data)) totalGame1++; // 한 줄씩 읽으면서 카운팅  while (game2file >> data) totalGame2++; // 한 단어씩 읽으면서 카운팅  }  // 낱말 연습  void Typing::wordTest() {  ifstream inputFile("word.txt"); // 파일 읽기 모드  if (!inputFile.is\_open()) { // 파일이 열리지 않는다면  cout << "Failed to open the file." << endl; // 오류 메시지 출력  }  // 시간 측정  time\_t start, finish;  start = time(NULL);  // 새로운 연습이 시작되었으므로 입력한 단어 개수, 맞은 단어 개수 초기화  completed = 0;  correct = 0;  string word;  string myAnswer;  while (inputFile >> word) { // 한 단어씩 읽어오기  cout << "\n" << word << endl;  cout << "-> ";  cin >> myAnswer; // 단어 따라 입력  completed++; // 입력한 단어 개수 1증가  if (word == myAnswer) correct++; // 단어가 일치하다면 맞은 단어 개수 1증가  else if (myAnswer == "q") { // 'q'를 입력하면  completed--; // 단어를 입력했다고 보지 않음  break; // 연습 종료  }  }  inputFile.close(); // 파일 닫기  finish = time(NULL);  duration = (double)(finish - start); // 경과 시간 게산  printResult("word"); // 연습 결과 출력  }  // 짧은글 연습  void Typing::shortTest() {  ifstream inputFile("sentence.txt"); // 파일 읽기 모드  if (!inputFile.is\_open()) { // 파일이 열리지 않는다면  cout << "Failed to open the file." << endl; // 오류 메시지 출력  }  // 시간 측정  time\_t start, finish;  start = time(NULL);  // 새로운 연습이 시작되었으므로 입력한 문장 개수, 맞은 문장 개수 초기화  completed = 0;  correct = 0;  string line;  string myAnswer;  while (getline(inputFile, line)) { // 한 줄씩 읽어오기  textcolor(YELLOW);  cout << "\n" << line << endl;  textcolor(WHITE);  cout << "-> ";  if (completed == 0) cin.ignore(); // 처음 한번만 입력 버퍼 비우기  getline(cin, myAnswer); // 문장 따라 입력  completed++; // 입력한 문장 개수 1증가  if (line == myAnswer) correct++; // 문장이 일치하다면 맞은 문장 개수 1증가  else if (myAnswer == "q") { // 'q'를 입력하면  completed--; // 문장을 입력했다고 보지 않음  break; // 연습 종료  }  }  inputFile.close(); // 파일 닫기  finish = time(NULL);  duration = (double)(finish - start); // 경과 시간 계산  printResult("line"); // 연습 결과 출력  }  // 긴글 연습  void Typing::longTest() {  ifstream inputFile("longscript.txt"); // 파일 읽기 모드  if (!inputFile.is\_open()) { // 파일이 열리지 않는다면  cout << "Failed to open the file." << endl; // 오류 메시지 출력  }  // 시간 측정  time\_t start, finish;  start = time(NULL);  // 새로운 연습이 시작되었으므로 입력한 문장 개수, 맞은 문장 개수 초기화  completed = 0;  correct = 0;  string line;  string myAnswer;  while (getline(inputFile, line)) { // 한 줄씩 읽어오기  textcolor(YELLOW);  cout << "\n" << line << endl;  textcolor(WHITE);  cout << "-> ";  if (completed == 0) cin.ignore(); // 처음 한번만 입력 버퍼 비우기  getline(cin, myAnswer); // 문장 따라 입력  completed++; // 입력한 문장 개수 1증가  if (line == myAnswer) correct++; // 문장이 일치하다면 맞은 문장 개수 1증가  else if (myAnswer == "q") { // 'q'를 입력하면  completed--; // 문장을 입력했다고 보지 않음  break; // 연습 종료  }  }  inputFile.close(); // 파일 닫기  finish = time(NULL);  duration = (double)(finish - start); // 경과 시간 계산    printResult("long"); // 연습 결과 출력  }  // 회원 정보 조회  void Typing::showRecord(string idtxt) {  ifstream inputFile(idtxt); // 파일 읽기 모드  if (!inputFile.is\_open()) { // 회원 이름으로 된 파일이 없다면 (회원이 게임을 한 기록이 없다면)  textcolor(RED);  cout << "No game record!!" << endl; // 오류 메시지 출력  textcolor(WHITE);  }  cout << endl;  string line;  textcolor(RED);  while (getline(inputFile, line)) { // 한 줄씩 읽어오기  cout << line << endl;  }  textcolor(WHITE);  inputFile.close(); // 파일 닫기  }  // 거꾸로 읽기 게임  void Typing::reverseGame() {  ifstream inputFile("English\_sentence.txt"); // 파일 읽기 모드  if (!inputFile.is\_open()) { // 파일이 열리지 않는다면  cout << "Failed to open the file." << endl; // 오류 메시지 출력  }  // 시간 측정  time\_t start, finish;  start = time(NULL);  // 새로운 게임이 시작되었으므로 입력한 문장 개수, 맞은 문장 개수 초기화  completed = 0;  correct = 0;  string line;  string myAnswer;  while (getline(inputFile, line)) { // 한 줄씩 읽어오기  string origin = line; // 원본 문장을 origin에 저장  istringstream iss(line); // 문자열 읽기 모드  string word;  cout << endl;  while (iss >> word) { // 문장에 있는 단어 읽어오기  reverse(word.begin(), word.end()); // 단어를 거꾸로 뒤집기  textcolor(YELLOW);  cout << word << " ";  }    textcolor(WHITE);  cout << "\n-> ";  if (completed == 0) cin.ignore(); // 처음 한번만 입력 버퍼 비우기  getline(cin, myAnswer); // 거꾸로 뒤집어진 문장을 원래 문장으로 바꾸어 입력  completed++; // 입력한 문장 개수 1증가  if (origin == myAnswer) correct++; // 원래 문장과 일치하다면 맞은 문장 개수 1증가  else if (myAnswer == "q") { // 'q'를 입력하면  completed--; // 문장을 입력했다고 보지 않음  break; // 연습 종료  }  }  inputFile.close(); // 파일 닫기    finish = time(NULL);  duration = (double)(finish - start); // 경과 시간 계산  printResult("game1"); // 게임 결과 출력  }  // 단어 만들기 게임  void Typing::guessWord() {  ifstream inputFile("English\_word.txt"); // 파일 읽기 모드  if (!inputFile.is\_open()) { // 파일이 열리지 않는다면  cout << "Failed to open the file." << endl; // 오류 메시지 출력  }  // 시간 측정  time\_t start, finish;  start = time(NULL);  // 새로운 게임이 시작되었으므로 입력한 단어 개수, 맞은 단어 개수 초기화  completed = 0;  correct = 0;  string myAnswer;  string word;  srand(time(NULL));  while (inputFile >> word) { // 한 단어씩 읽어오기  string origin = word; // 원본 단어를 origin에 저장  // 단어에 있는 문자들을 랜덤으로 섞는 작업을 30번 반복하기 위한 for문  // 난이도를 고려하여 단어의 첫 알파벳은 원본과 동일하게 두고 나머지를 섞음  for(int i = 0; i < 3; i++) {  int len = word.length(); // 단어의 길이  int randNum1 = rand() % (len-1) + 1; //0~len-1까지의 랜덤한 수 생성  int randNum2 = rand() % (len-1) + 1; //0~len-1까지의 랜덤한 수 생성  char temp;  // 두 랜덤 정수 인덱스에 있는 원소들끼리 서로 바꿈  temp = word[randNum1];  word[randNum1] = word[randNum2];  word[randNum2] = temp;  }  textcolor(YELLOW);  cout << "\n" << word << endl;  textcolor(WHITE);  cout << "-> ";  cin >> myAnswer; // 순서가 랜덤으로 섞인 단어를 원래 단어로 바꾸어 입력  completed++; // 입력한 단어 개수 1증가    if (origin == myAnswer) correct++; // 원래 단어와 일치하다면 맞은 단어 개수 1증가  else if (myAnswer == "q") { // 'q'를 입력하면  completed--; // 단어를 입력했다고 보지 않음  break; // 연습 종료  }  }  inputFile.close(); // 파일 닫기  finish = time(NULL);  duration = (double)(finish - start); // 경과 시간 게산  printResult("game2"); // 게임 결과 출력  }  // 결과 출력  void Typing::printResult(string wordORline) {  // 오타수 = 입력한 단어 개수 - 맞은 단어 개수  // 진행도 = (입력한 단어 개수) / (전체 단어 개수) \* 100  // 정확도 = (맞은 단어 개수) / (입력한 단어 개수) \* 100  wrong = completed - correct; // 오타수 계산  accuracy = (correct / completed) \* 100; // 정확도 계산  // 진행도 계산  if (wordORline == "word") process = (completed / totalWord) \* 100;  else if (wordORline == "line") process = (completed / totalLine) \* 100;  else if (wordORline == "long") process = (completed / totalLong) \* 100;  else if (wordORline == "game1") process = (completed / totalGame1) \* 100;  else if (wordORline == "game2") process = (completed / totalGame2) \* 100;  // 경과시간 비율에 맞게 조정  if (duration <= 60) trans\_duration = 100;  else if (duration > 60 && duration <= 70) trans\_duration = 80;  else if (duration > 70 && duration <= 80) trans\_duration = 60;  else trans\_duration = 90;  // 진행도 20%, 정확도 50%, 경과시간 30%를 반영하여 점수 계산  score = (process \* 0.2) + (accuracy \* 0.5) + (trans\_duration \* 0.3);  // 게임 결과 출력  textcolor(RED);  cout << "\n진행도 : " << process << "%" << endl;  cout << "오타수 : " << wrong << "개"<< endl;  cout << "정확도 : " << accuracy << "%" << endl;  cout << "경과 시간 : " << duration << "초" << endl;  cout << "점수 : " << score << "점" << endl;  textcolor(WHITE);  // 게임 수행 시 순위 출력, 파일에 결과 기록  if (wordORline == "game1" || wordORline == "game2") {  ifstream inputFile("score.txt"); // 게임 결과 작성을 위해 "score.txt" 파일 열기  lank = 1; // 순위  string word;  while (inputFile >> word) { // 한 단어씩 읽어오기  if (score < stod(word)) lank++; // 회원의 점수가 다른 회원의 점수보다 작다면 lank가 1씩 밀림  }  textcolor(RED);  cout << "순위 : " << lank << "위" << endl; // 순위 출력  textcolor(WHITE);  ofstream outputFile(id, ios::app); // 기존의 파일 뒤에 새로운 내용 작성 모드로 파일 열기  // 파일에 게임 결과 작성  outputFile << "진행도 " << process << "\n";  outputFile << "오타수 " << wrong << "\n";  outputFile << "정확도 " << accuracy << "\n";  outputFile << "경과 시간 " << duration << "\n";  outputFile << "점수 " << score << "\n\n";  outputFile.close();  // lank는 계속 바뀌므로 파일에 작성 X  ofstream scoreFile("score.txt", ios::app); // 기존의 파일 뒤에 새로운 내용 작성 모드로 파일 열기  scoreFile << score << " "; // 게임 점수 작성  scoreFile.close(); // 파일 닫기  }  }  void Typing::textcolor(int color) {  SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), color);  } |

|  |
| --- |
| TypingMain.cpp |
| #include <iostream>  #include "Typing.h"  using namespace std;  int main(void) {  Typing typing;  typing.textcolor(RED);  cout << "\n\*\*\*\*\*\*\*\* 타자연습 프로그램을 시작합니다 \*\*\*\*\*\*\*\*\n" << endl;  typing.textcolor(WHITE);  cout << "관리자(1) / 사용자(2) 선택하세요 : ";  int usermanager;  cin >> usermanager; // 관리자 or 사용자 선택  // 관리자 권한으로 들어올 경우  if (usermanager == 1) {  cout << "비밀번호를 입력하세요 : ";  int password; // 비밀번호  cin >> password;  typing.textcolor(LIGHTMAGENTA);  while (true) {  if (password == 1014) { // 입력한 값이 비밀번호와 일치한다면  cout << "\n로그인 성공!" << endl; // 로그인 성공 메시지 출력  break;  }  else { // 비밀번호가 틀리다면  cout << "비밀번호가 틀렸습니다. 다시 입력해주세요 : "; // 오류 메시지 출력  cin >> password; // 다시 입력 받음  }  }    typing.textcolor(RED);  // 기능 메뉴  cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;  cout << "1. 단어 데이터 추가" << endl;  cout << "2. 단어 데이터 수정" << endl;  cout << "3. 단어 데이터 삭제" << endl;  cout << "4. 문장 데이터 추가" << endl;  cout << "5. 문장 데이터 수정" << endl;  cout << "6. 문장 데이터 삭제" << endl;  cout << "7. 회원 정보 삭제" << endl;  cout << "8. 점수 기록 삭제" << endl;  cout << "0. 종료" << endl;  cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;  typing.textcolor(WHITE);  while(1) {  cout << "\n원하는 기능을 선택하세요 : ";  int func;  cin >> func; // 번호 입력  if (func == 0) break; // 0 입력 시 프로그램 종료  cin.ignore(); // 입력 버퍼 비우기    string data; // 추가, 삭제, 수정 후 단어/문장  string origin; // 수정 전 단어/문장  switch (func) {  case 1:  cout << "추가할 단어를 입력하세요 : ";  getline(cin, data); // 추가할 단어 입력  typing.addData(data, "word.txt"); // "word.txt" 파일에 data 추가  break;  case 2:  cout << "기존 단어와 수정할 단어를 입력하세요." << endl;  cout << "기존 단어 : ";  getline(cin, origin); // 수정 전 단어  cout << "수정할 단어 : ";  getline(cin, data); // 수정 후 단어  typing.modData(origin, data, "word.txt"); // "word.txt" 파일에 있는 단어 수정  break;  case 3:  cout << "삭제할 단어를 입력하세요 : ";  getline(cin, data); // 삭제할 단어 입력  typing.delData(data, "word.txt"); // "word.txt" 파일에 있는 단어 삭제  break;  case 4:  cout << "추가할 문장을 입력하세요 : ";  getline(cin, data); // 추가할 문장 입력  typing.addData(data, "sentence.txt"); // "sentence.txt" 파일에 data 추가  break;  case 5:  cout << "기존 문장과 수정할 문장를 입력하세요." << endl;  cout << "기존 문장 : ";  getline(cin, origin); // 수정 전 문장  cout << "수정할 문장 : ";  getline(cin, data); // 수정 후 문장  typing.modData(origin, data, "sentence.txt"); // "sentence.txt" 파일에 있는 문장 수정  break;  case 6:  cout << "삭제할 문장을 입력하세요 : ";  getline(cin, data); // 삭제할 문장 입력  typing.delData(data, "sentence.txt"); // "sentence.txt" 파일에 있는 문장 삭제  break;  case 7:  cout << "삭제할 회원의 닉네임을 입력하세요 : ";  getline(cin, data); // 삭제할 회원 닉네임 입력  typing.removeFile(data); // 회원 이름으로 된 파일 삭제  break;  case 8:  cout << "score.txt에 저장된 점수 기록들을 삭제합니다." << endl;  typing.removeScore(); // 점수 기록 삭제  break;  default:  break;  }  }  }  // 사용자 권한으로 들어올 경우  else if (usermanager == 2) {  cout << "닉네임을 입력하세요 : ";  string id;  cin >> id; // 닉네임 입력  string idtxt = id + ".txt"; // 회원 이름으로 된 파일 이름  typing.setId(idtxt);  ifstream inputFile(idtxt); // 파일 열기  cout << endl;  typing.textcolor(LIGHTMAGENTA);  if (!inputFile.is\_open()) cout << "신규 회원이시네요. 반갑습니다!\n"; // 파일이 없다면 신규 회원으로 등록  else cout << id << "회원님, 반갑습니다!\n"; // 파일이 열린다면 기존 회원  typing.textcolor(RED);  cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;  cout << "1. 낱말 연습" << endl;  cout << "2. 짧은글 연습" << endl;  cout << "3. 긴글 연습" << endl;  cout << "4. 게임 결과 확인" << endl;  cout << "5. 거꾸로 읽기 게임 (난이도 하)" << endl;  cout << "6. 단어 맞추기 게임 (난이도 상)" << endl;  cout << "0. 종료" << endl;  cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;  // 파일에 있는 총 단어/문자 개수 카운팅  typing.countWordLine("word.txt", "sentence.txt", "longscript.txt",  "English\_sentence.txt", "English\_word.txt");  while(1) {  typing.textcolor(WHITE);  cout << "\n원하는 기능을 선택하세요 : ";  int func;  cin >> func; // 번호 입력  if (func == 0) break; // 0 입력 시 프로그램 종료  switch (func) {  case 1:  cout << "\n\*\*\*\*\* 낱말 연습을 시작합니다! \*\*\*\*\*" << endl;  typing.wordTest();  break;  case 2:  cout << "\n\*\*\*\* 짧은글 연습을 시작합니다! \*\*\*\*\*" << endl;  typing.shortTest();  break;  case 3:  cout << "\n\*\*\*\*\* 긴글 연습을 시작합니다! \*\*\*\*\*" << endl;  typing.longTest();  break;  case 4:  cout << "\n\*\*\*\*\* 게임 결과를 조회합니다! \*\*\*\*\*" << endl;  typing.showRecord(idtxt); // 회원 닉네임으로 저장된 파일 열고 결과 출력  break;  case 5:  cout << "\n\*\*\*\*\* 거꾸로 읽기 게임을 시작합니다! \*\*\*\*\*" << endl;  typing.reverseGame(); // 게임 후 회원 닉네임으로 저장된 파일에 결과 작성  break;  case 6:  cout << "\n\*\*\*\*\* 단어 맞추기 게임을 시작합니다! \*\*\*\*\*" << endl;  typing.guessWord(); // 게임 후 회원 닉네임으로 저장된 파일에 결과 작성  break;  default:  break;  }  }  }  else cout << "잘못 입력하셨습니다." << endl;  } |

|  |
| --- |
| Windows.h |
| #ifndef Windows\_h  #define Windows\_h  #include <unistd.h>  #include <sys/select.h>  #include <sys/ioctl.h>  #include <termios.h>  using namespace std;  #define BLACK 0  #define BLUE 1  #define GREEN 2  #define CYAN 3  #define RED 4  #define MAGENTA 5  #define BROWN 6  #define LIGHTGRAY 7  #define DARKGRAY 8  #define LIGHTBLUE 9  #define LIGHTGREEN 10  #define LIGHTCYAN 11  #define LIGHTRED 12  #define LIGHTMAGENTA 13  #define YELLOW 14  #define WHITE 15  #define FALSE 0  #define TRUE 1  #define STD\_OUTPUT\_HANDLE 1  #define FOREGROUND\_INTENSITY 0x0008 // text color is intensified.  class COORD {  public:  int x;  int y;  };  class CONSOLE\_CURSOR\_INFO {  public:  int bVisible;  int dwSize;  };  int GetStdHandle(int a) {  return 1;  }  void printColorString(int color) {  cout << "\033[" << color << "m";  }  const int dcolor[] = { 0, 4, 2, 6, 1, 5, 3, 7, 0, 4, 2, 6, 1, 5, 3, 7 };  void ColorChange(unsigned short color) {  int FG = (color & 0x000F);  int FG\_isIntensity = (color & FOREGROUND\_INTENSITY) ? 60 : 0;  int result\_foreground = 30 + FG\_isIntensity;  result\_foreground += dcolor[FG];  printColorString(result\_foreground);  }  void SetConsoleTextAttribute(int handle, int color) {  ColorChange(color);  }  #endif /\* Windows\_h \*/ |

**Ⅲ. 프로그램 결과**

1. 프로그램 실행 결과

|  |  |
| --- | --- |
| 관리자 | |
| 로그인 | 단어 데이터 추가 |
| 단어 데이터 수정 | 단어 데이터 삭제 |
| 회원 정보 삭제 | 점수 기록 삭제 |

|  |  |
| --- | --- |
| 사용자 | |
| 기존 회원 로그인 | 게임 후 기존 파일에 결과 저장 |
| 신규 회원 로그인 | 게임 후 새로운 파일에 결과 저장 |
| 게임 결과 조회 | |

2. 한계점 및 개선 방안

|  |
| --- |
| 사용자가 연습이나 게임을 실행하면 단어와 문장이 매번 파일에 저장된 순서로 출력된다. 게임을 할 때마다 같은 순서로 출력된다는 점에서 한계가 있었고, 단어나 문장을 읽어온 후 랜덤으로 나올 수 있도록 개선하면 좋을 것 같다.  그리고 단어 맞추기 게임에서 힌트 기능이 추가되면 좋을 것 같다. 맨 앞 문자를 제외하고 모두 랜덤으로 섞여있는데, 사용자가 계속해서 맞추지 못하고 시간이 경과되면 게임이 진행되는데 어려움이 있을 것이다. 사용자에게 다음 문자를 알려주는 힌트 기능이 있다면 게임 진행에 효과적일 것이다.  함수가 많고 코드가 길어서 가독성이 떨어지는 부분이 있었다. 클래스를 나누어서 각 기능을 정의하고 상속을 사용하면 이를 개선할 수 있을 것이다. 오버라이딩을 통해 같은 기능을 하는 함수들이 객체의 특성에 맞게 동작할 수 있도록 나눠보면 좋을 것 같다.  이 프로그램을 구현하면서 파일에 있는 데이터를 한 줄씩 읽어오고 한 줄씩 따라 입력할 때 버퍼에 문제가 생기는 경우가 많았다. 이 오류를 해결하는데 많은 시간을 투자했는데, 이 시간을 디버깅에 투자했다면 더 완성도 있는 코드가 나왔을 것 같다. |