2021-1 C++프로그래밍 실습과제 06

(1) 각 문제에 대한 분석과 및 해결 방법

6.1. 실습과제 3에서 구현한 두 가지 게임 중에서 하나를 골라 클래스 버전으로 수정하라.

(1) 게임 클래스를 만든다. 이를 위해 6.6절의 내용을 참고하라.

(2) main() 함수를 수정하여 실습과제 3에서와 같이 게임을 할 수 있도록 하라. 이를 위해 게임 객체를 만들고 play() 등의 메소드를 호출하여야 한다.

[문제분석 및 해결방법] : 기존의 함수와 변수를 기반으로 SpeedGugu 클래스의 함수와 멤버 변수등을 만들고 알맞은 접근 지정자를 사용하여 은닉성을 고려 하여 SpeedGugu 게임을 객체지향적인 형태로 알맞게 수정하였다.

6.2. 교재 291~294쪽의 코드를 참고하여 자신만의 행맨 게임을 구현하라.

(1) 자신만의 영어 단어를 100개 이상 준비하고, 이를 파일에 저장하라. 예를 들어, 인터넷에서 가장 많이 사용되는 영어 단어 1000개를 찾아 파일에 저장할 수 있다. (예: google에서 "top english word list" 등의 키워드로 검색)

(2) 6.8절의 행맨 프로그램을 수정하여 파일에서 모든 단어를 읽어들이고, 무작위로 임의의 단어가 선택되고 이를 맞히는 게임으로 수정한다.

(3) 게임이 진행될 때 마다 출력되는 그림을 자신만의 그림으로 수정하라. 이를 위해 HangmanProgress.txt 파일을 수정하면 된다. 하나씩 틀릴 때 마다 어떤 그림이 나타날 것인가를 먼저 설계하고, 이를 파일에 저장하라. 예를 들어, 아스키 아트를 사용할 수도 있다.

[문제분석 및 해결방법] : 영어 단어 파일을 구비하여 줄의 끝까지 읽고 난수 함수를 이용하여 랜덤한 단어를 추출하는 방식으로 행맨 게임을 수정하였고, HangmanProgress.txt 파일을 적절히 수정하여 나만의 행맨 아스키 아트를 화면에 표시하였다. 또한 playByWordList 함수를 만들어 랜덤한 단어를 고르고, 기존의 play 함수에 인자로 넘겨주는 형태로 구현하여 기존 구현 함수를 유지하면서 기능을 확장하였다.

(2) 자신이 구현한 주요 코드

6.1. 실습과제 3에서 구현한 두 가지 게임 중에서 하나를 골라 클래스 버전으로 수정하라.

<SpeedGugu.h>

```
class SpeedGugu {
private:
    int NumGames =0;     // 전체 시도 횟수
    int NumWins =0;      // 맞힌 횟수
    double Score =0;     // 점수
    double tElapsed =0;          // 게임 소요시간
    clock_t t0;
    clock_t t1;
    int nPlay =10;
    bool playGuguOnce() {
                                ...
    }
    bool playSpeedGugu2Once() {
                                ...
    }
    bool playSpeedSumOnce(int game) {
                                ...
    }

    void startClock() {
                                ...
    }

    double getScoreWithEndClock() {
                                ...
    }
    double playSpeedGugu(int nPlay) {
                                ...
    }
    double playSpeedGugu2(int nPlay) {
                                ...
    }
    double playSpeedSum(int nPlay, int game) {
                                ...
    }

public:
    void play() {
        srand((unsigned)time(NULL));
        int game;
        printf("게임을 선택하세요: ₩n");
        printf("      1: 스피드 구구단₩n");
        printf("      2: 두자리수 곱셈₩n");
        printf("      3~9 : n자리수 덧셈₩n");
        printf("       선택 ---> ");
        scanf("%d", &game);
        if (game ==1) printf("[스피드 구구단 게임]₩n₩n");
        else if (game ==2) printf("[두자리수 곱셈 게임]₩n₩n");
        else if (3 <= game && game <=9) printf("[%d자리수 덧셈 게임]₩n₩n", game);
        else return;
        getchar();
        printf(" %d번 테스트 하겠습니다.₩n", nPlay);
        printf(" 크게 심호흡을 하시고...₩n 준비되면 앤터를 누르세요...");
        getchar();
        system("cls");
        double score =0;
        if (game ==1) score = playSpeedGugu(nPlay);
        else if (game ==2) score = playSpeedGugu2(nPlay);
        else score = playSpeedSum(nPlay, game);
        printf("₩n점수 = %4.1f점(총 %4.1f초)₩n", score, tElapsed);
    }
};
```

```
#include "SpeedGugu.h"
int main() {
  SpeedGugu game;
  game.play();
  std::cout <<std::endl <<"Press ENTER to exit..."; fflush(stdin); getchar(); getchar();
}
```

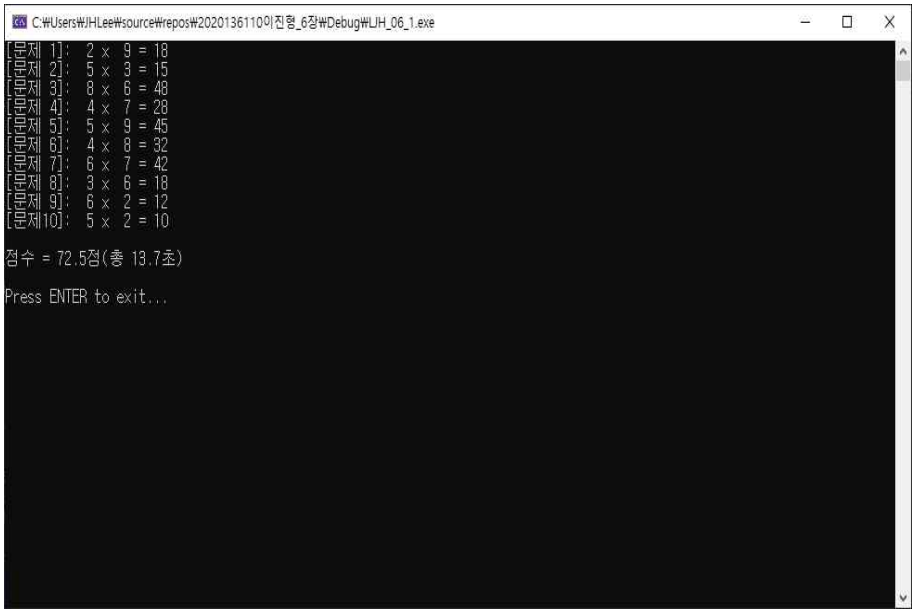
6.2. 교재 291~294쪽의 코드를 참고하여 자신만의 행맨 게임을 구현하라.

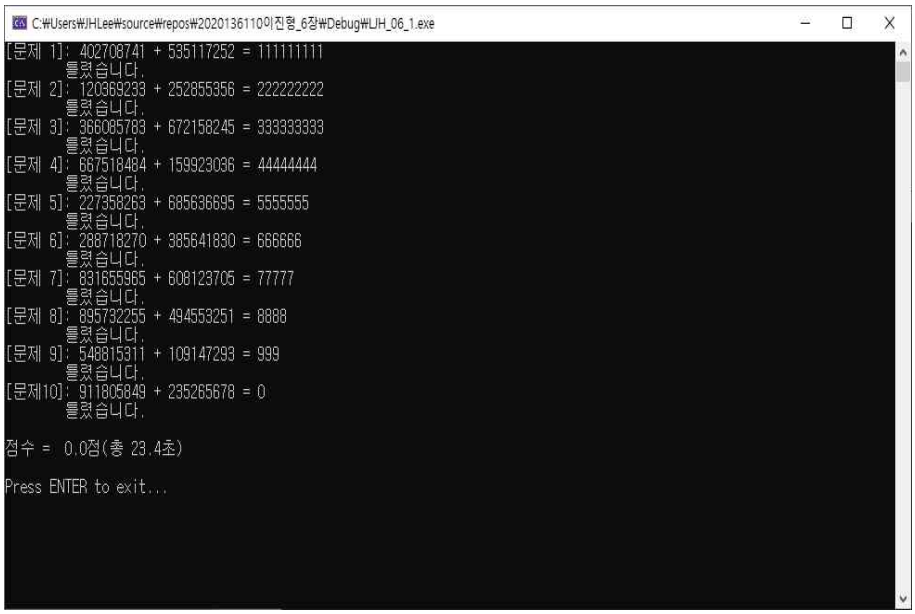
<Hangman.h>

```
class Hangman {
  string  progress[88];
  string  problem;
  string  answer;
  string  guessed;
  int   nTries;
  const int maxTries =7;
  void load(const char * progName ="HangmanProgress.txt") {
    ifstream fs(progName);
    if (fs) {
      getline(fs, progress[0]);
      for (int i =0; i <88; i ++)
        getline(fs, progress[i]);
    }
  }
  void print() {
    system("cls");
    cout <<"       <Hangman Game>₩n";
    for (int i =0; i <11; i ++)
      cout <<'₩t'<< progress[nTries *11 + i] <<endl;
    cout <<"₩n₩t"<< answer;
    cout <<"₩n₩n "<< guessed;
  }
  int countMatched(char ch) {
    int nMatched =0;
    for (int pos =-1; ; ) {
      pos = problem.find(ch, pos +1);
      if (pos <0) break;
      answer[pos] = ch;
      nMatched++;
    }
    return nMatched;
  }
  void guess() {
    char ch = getch();
    if (ch >='a'&& ch <='z') {
      int pos = guessed.find(ch);
      if (pos <0) {
        guessed[ch -'a'] = ch;
        if (countMatched(ch) ==0) nTries ++;
      }
    }
  }
public:
  void play(string prob) {
    load();
    problem = prob;
    answer =string(problem.length(), '-');
    guessed =string(26, '.');
    nTries =0;
    while (nTries < maxTries && answer != problem) {
      print();
      guess();
    }
    print();
    cout <<"₩n₩t"<< ((nTries == maxTries) ? "실패" : "정답") <<endl;
  }
  void playByWordList() {
    srand((unsigned int)time(NULL));
    ifstream fs("WordList.txt");
    if (fs) {
      string line;
      long maxLine;
      for (maxLine =0; getline(fs, line); ++maxLine);
      long targetLine = BIG_RAND() % maxLine;
      fs.clear();
      fs.seekg(0);
      for (long i =0; i < targetLine; i ++) {
        getline(fs, line);
      }
      play(line);
    }
  }
};
```

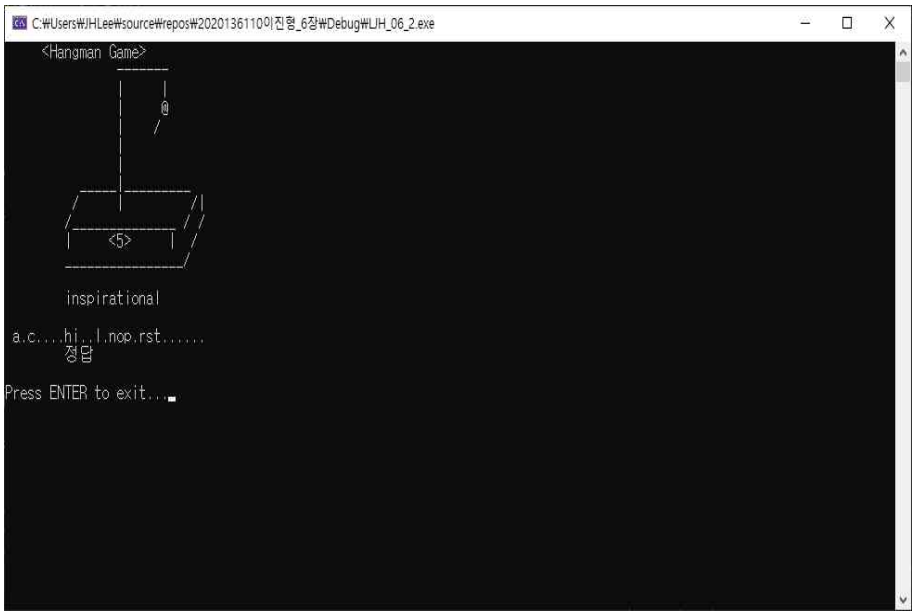
(3) 다양한 입력에 대한 테스트 결과

6.1. 실습과제 3에서 구현한 두 가지 게임 중에서 하나를 골라 클래스 버전으로 수정하라.





6.2. 교재 291~294쪽의 코드를 참고하여 자신만의 행맨 게임을 구현하라.



(4) 코드에 대한 설명 및 해당 문제에 대한 고찰

6.1. 실습과제 3에서 구현한 두 가지 게임 중에서 하나를 골라 클래스 버전으로 수정하라.

(1) 게임 클래스를 만든다. 이를 위해 6.6절의 내용을 참고하라.

<SpeedGugu.h>

```
class SpeedGugu {
private:
    int NumGames =0;     // 전체 시도 횟수
    int NumWins =0;      // 맞힌 횟수
    double Score =0;     // 점수
    double tElapsed =0;          // 게임 소요시간
    clock_t t0;
    clock_t t1;
    int nPlay =10;
    bool playGuguOnce() {
                                ...
    }
    bool playSpeedGugu2Once() {
                                ...
    }
    bool playSpeedSumOnce(int game) {
                                ...
    }

    void startClock() {
                                ...
    }

    double getScoreWithEndClock() {
                                ...
    }
    double playSpeedGugu(int nPlay) {
                                ...
    }
    double playSpeedGugu2(int nPlay) {
                                ...
    }
    double playSpeedSum(int nPlay, int game) {
                                ...
    }

public:
    void play() {
        srand((unsigned)time(NULL));
        int game;
        printf("게임을 선택하세요: ₩n");
        printf("     1: 스피드 구구단₩n");
        printf("     2: 두자리수 곱셈₩n");
        printf("     3~9 : n자리수 덧셈₩n");
        printf("      선택 ---> ");
        scanf("%d", &game);
        if (game ==1) printf("[스피드 구구단 게임]₩n₩n");
        else if (game ==2) printf("[두자리수 곱셈 게임]₩n₩n");
        else if (3 <= game && game <=9) printf("[%d자리수 덧셈 게임]₩n₩n", game);
        else return;
        getchar();
        printf(" %d번 테스트 하겠습니다.₩n", nPlay);
        printf(" 크게 심호흡을 하시고...₩n 준비되면 앤터를 누르세요...");
        getchar();
        system("cls");
        double score =0;
        if (game ==1) score = playSpeedGugu(nPlay);
        else if (game ==2) score = playSpeedGugu2(nPlay);
        else score = playSpeedSum(nPlay, game);
        printf("₩n점수 = %4.1f점(총 %4.1f초)₩n", score, tElapsed);
    }
};
```

기존의 코드를 클래스 형태로 수정하여 객체지향적인 형태로 수정하였다. 외부 사용이 불필요한 내부 처리용 멤버 변수, 함수 등은 private 접근 지정자를 사용하여 은닉하고, play() 함수로 게임을 시작할 수 있도록 구성하였다.

(2) main() 함수를 수정하여 실습과제 3에서와 같이 게임을 할 수 있도록 하라. 이를 위해 게임 객체를 만들고 play() 등의 메소드를 호출하여야 한다.

```
#include "SpeedGugu.h"
int main() {
  SpeedGugu game;
  game.play();
  std::cout <<std::endl <<"Press ENTER to exit..."; fflush(stdin); getchar(); getchar();
}
```

SpeedGugu 객체를 생성하고 play() 함수를 호출하여 게임을 시작하도록 구성하였다.

6.2. 교재 291~294쪽의 코드를 참고하여 자신만의 행맨 게임을 구현하라.

(1) 자신만의 영어 단어를 100개 이상 준비하고, 이를 파일에 저장하라. 예를 들어, 인터넷에서 가장 많이 사용되는 영어 단어 1000개를 찾아 파일에 저장할 수 있다. (예: google에서 "top english word list" 등의 키워드로 검색)

자연어 처리, 문장 감정 분석등에 사용되는 감정 사전인 VADER (Valence Aware Dictionary and sEntiment Reasoner)의 단어를 이용하여 단어 파일을 구성하였다.

본 프로그램에서는 VADER의 이모티콘 형태의 항목, 채팅 축약어, 단어에 대한 정량값(숫자) 등의 데이터를 사용할 필요가 없었고 따라서 자바스크립트와 Node.js File API를 이용하여 해당 단어 파일을 읽은 뒤 구분자를 기준으로

단어만을 분리하고, 정규 표현식을 사용하여 알파벳으로만 구성된 6글자 이상의 단어만을 추출하였다.

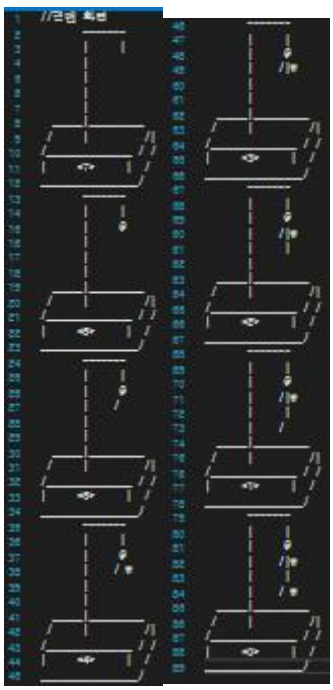
```
const fs = require('fs');
let extracted = fs.readFileSync('vader_lexicon.txt', 'utf8').split('₩n').map(e =>
e.split('₩t').shift()).filter(e => e.match(/[a -z]{6,}/)).join('₩r₩n');
fs.writeFileSync('WordList.txt', extracted, 'utf8');
```

(2) 6.8절의 행맨 프로그램을 수정하여 파일에서 모든 단어를 읽어들이고, 무작위로 임의의 단어가 선택되고 이를 맞히는 게임으로 수정한다.

```
<Hangman.h>
class Hangman {
 ...
public:
  void play(string prob) {
    ...
  }
  void playByWordList() {
    srand((unsigned int)time(NULL));
    ifstream fs("WordList.txt");
    if (fs) {
      string line;
      long maxLine;
      for (maxLine =0; getline(fs, line); ++maxLine);
      long targetLine = BIG_RAND() % maxLine;
      fs.clear();
      fs.seekg(0);
      for (long i =0; i < targetLine; i ++) {
        getline(fs, line);
      }
      play(line);
    }
  }
};
```

ifstream을 이용하여 파일의 끝 라인 수를 구하고, rand() 함수를 이용하여 무작위의 단어가 선택되어 게임이 시작될 수 있도록 구성하였다. 기존의 play 함수에 인자로 넘겨주는 형태로 구현하여 기존 구현 함수를 유지하면서 기능을 확장하였다.

(3) 게임이 진행될 때 마다 출력되는 그림을 자신만의 그림으로 수정하라. 이를 위해 HangmanProgress.txt 파일을 수정하면 된다. 하나씩 틀릴 때 마다 어떤 그림이 나타날 것인가를 먼저 설계하고, 이를 파일에 저장하라. 예를 들어, 아스키 아트를 사용할 수도 있다.



```
<Hangman.h>
class Hangman {
  string  progress[88];
  string  problem;
  string  answer;
  string  guessed;
  int   nTries;
  const int maxTries =7;
  void load(const char * progName ="HangmanProgress.txt") {
    ifstream fs(progName);
    if (fs) {
      getline(fs, progress[0]);
      for (int i =0; i <88; i ++)
        getline(fs, progress[i]);
    }
  }
  void print() {
    system("cls");
    cout <<"       <Hangman Game>₩n";
    for (int i =0; i <11; i ++)
      cout <<'₩t'<< progress[nTries *11 + i] <<endl;
    cout <<"₩n₩t"<< answer;
    cout <<"₩n₩n  "<< guessed;
  }
  …
};
```

행맨 아스키 아트를 아래에 조금 디테일하게 아래에 받침대가 있는 버전으로 수정하였고 기존 8줄에서 11줄이 된 아스키 아트를 지원하기 위해 적절히 코드를 수정하였다.

(5) 이번 과제에 대한 느낀점

이번 과제를 해결하기 위해 교재의 코드를 분석하고 조사하면서 모르고 있었던 세부적인 내용을 많이 알 수 있어서 좋았다.