C++프로그래밍및실습

단어스크램블게임

진척 보고서 #2

제출일자:2024.11.17
제출자명:윤동해
제출자학번:234129

1. 프로젝트 목표 (16 pt)

1) 배경 및 필요성 (14 pt)

단어 스크램블 게임은 단어의 철자를 무작위로 섞어 제시해 플레이어가 원래 단어를 맞추는 퍼즐 게임이다. 수업 시간에 학습한 내용과 그간 구현해보았던 '틱택토', '야구 게임', '머드 게임'을 참고해 간단한 단어 스크램블 게임을 구현할 수 있을 것 같아 주제로 선정하게 되었다. 이 게임은 언어 능력과 논리적 사고력을 향상시킬 수 있는 교육적 목적을 가진 게임으로, 이를 통해 영어 단어 어휘력 향상을 기대할 수 있다는 점에서 필요성을 가진다.

2) 프로젝트 목표

간단한 단어의 순서를 랜덤으로 제시해 플레이어의 언어 능력과 논리적 사고를 향상시키는 게임 프로그램을 만드는 것을 목표로 한다.

3) 차별점

기존 프로그램은 단어를 직접 입력 받아 프로그램이 뒤섞는데 이는 본인이 작성한 단어만 알면 쉽게 맞출 수 있기 때문에 랜덤하게 게임에서 다양한 단어 리스트를 통해 문제를 제시하며, 난이도 조절을 통해 기본 난이도 외에도, 쉬운 단어에서 어려운 단어로 점진적으로 상승하는 난이도 조절 기능을 제공하도록 한다. 점수 시스템을 추가해 직관적으로 점수를 확인할 수 있게 하며, 여유가 된다면 제한 시간 기능을 넣어 게임의 긴장감을 추가할 수 있도록 한다.

2. 기능 계획

1) 기능 1 기본 게임 플레이 (기본적인 게임 플레이에 필요한 기능들)

- 무작위 단어를 선택하고 이를 섞어 플레이어에게 제시한다.

(1) 세부기능 1 다양한 단어 테마 선택

- 플레이어가 게임 시작 시 원하는 주제를 선택하여 해당 주제의 단어들로만 게임을 진행할 수 있게 한다.

- 플레이어가 단어를 맞추면 점수를 획득하며, 틀리면 정답을 알려준다.

(2) 세부 기능 2 점수 시스템

- 맞출 때마다 점수를 획득하고, 틀릴 경우 감점하여 최종 점수를 누적한다.

2) 기능 2 단어 난이도 조절

- 짧은 단어에서 긴 단어로 점차 난이도가 상승하도록 설정한다.

3) 기능 3 제한 시간 추가

- 타이머를 사용하여 단어를 맞출 수 있는 제한 시간을 설정하고, 시간이 초과되면 점수가 차감되거나 게임이 종료된다.

4) 기능 4 최고 기록 저장

- 최고 점수 기록을 파일에 저장하여, 다음 게임 시작 시 플레이어에게 최고 점수를 보여준다.

3. 진척사항

1) 기능 구현

1. 기본 게임 플레이 (기본적인 게임 플레이에 필요한 기능들)

- 무작위 단어를 선택하고 이를 섞어 플레이어에게 제시한다.

(1) 세부기능 1 다양한 단어 테마 선택

- 플레이어가 게임 시작 시 원하는 주제를 선택하여 해당 주제의 단어들로만 게임을 진행할 수 있게 한다.

- 입출력

  - 입력: 주제를 선택할 때 1, 2, 3번을 입력 받는다.

    섞인 단어가 출력되었을 때 정답을 입력 받는다.

  - 출력: 주제를 선택하세요: 1. 과일 2. 동물 3. 직업 선택:

    잘못된 입력입니다. 기본 주제(과일)로 진행합니다.

    순서가 섞인 단어가 출력된다.

    정답일 경우에 "정답입니다!"가 출력된다.

    오답일 경우에 "틀렸습니다. 정답은 '<원래 단어>' 입니다."가 출력된다.

- 설명

  - 단어를 섞는 함수 shuffleWord

  • random_device rd: 난수 생성을 위한 장치를 초기화한다.

  • mt19937 generator(rd()): Mersenne Twister 난수 생성기를 초기화한다.

  • shuffle(word.begin(), word.end(), generator): 주어진 단어의 문자들을 무작위로 섞는다.

  - 주제를 선택하고 단어를 반환하는 함수 chooseTheme

  • 사용자가 주제를 선택하고 이에 맞는 단어 목록을 반환한다.

  • 잘못된 입력이 들어오면 기본 주제인 "과일" 목록을 반환한다.

- main 함수

  - chooseTheme 함수를 호출하여 사용자가 선택한 주제의 단어 목록을 불러온다.
  - 난수 생성기를 사용하여 단어 목록 중 하나를 무작위로 선택한다.
  - 선택된 단어를 suffleWord 함수를 통해 섞는다.
  - 섞인 단어를 출력하고, 사용자가 원래 단어를 맞추도록 한다.
  - 사용자의 입력과 원래 단어를 비교하여 정답 여부를 출력한다..

- 적용된 배운 내용 (예: 반복문, 조건문, 클래스, 함수, 포인터 등)

  - 벡터, 랜덤 난수, switch, string, cout, 조건문 if, void

- 새롭게 공부한 내용

  - #include <algorithm>, shuffleWord 함수는 따로 검색하여 알아보았다.

```
void shuffleWord(string &word) {
    random_device rd;               // 난수 생성기
    mt19937 generator(rd());        // Mersenne Twister 엔진
    shuffle(word.begin(), word.end(), generator); // std::shuffle 사용
}
```

random_device rd는 난수 생성기를 초기화한다.

mt19937 generator(rd())는 Mersenne Twister 알고리즘을 사용하는 난수 생성기다.

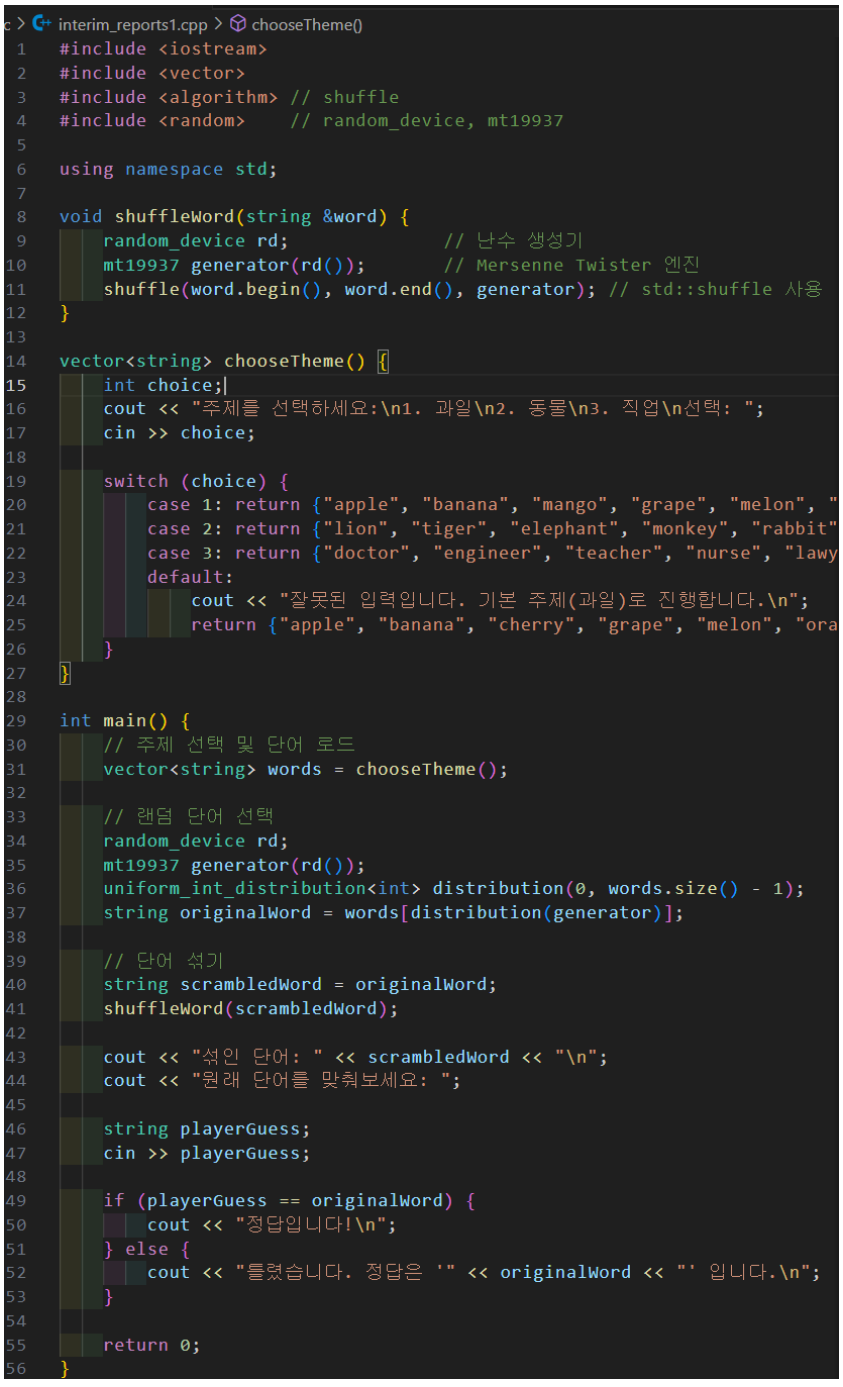
shuffle(word.begin(), word.end(), generator)은 shuffle 함수를 사용하여 word 문자열의 문자를 무작위로 섞는다. shuffle(시작, 끝, 난수 생성 엔진)

```
    // 랜덤 단어 선택
    random_device rd;
    mt19937 generator(rd());
    uniform_int_distribution<int> distribution(0, words.size() - 1);
    string originalWord = words[distribution(generator)];
```

uniform_int_distribution 객체를 생성한다. 이 객체는 0부터 words.size() - 1 (단어 목록의 마지막 인덱스) 사이의 정수 중에서 균일하게 난수를 생성한다.(모든 단어 동일 확률)

distribution(generator)을 호출하면 generator를 사용해 지정된 범위 내에서 난수를 생성한다. 이 난수는 words 벡터의 인덱스로 사용되고, 그 인덱스에 있는 단어가 originalWord에 저장된다.

- 코드 스크린샷



```cpp
#include <iostream>
#include <vector>
#include <algorithm> // shuffle
#include <random>    // random_device, mt19937

using namespace std;

void shuffleWord(string &word) {
    random_device rd;               // 난수 생성기
    mt19937 generator(rd());        // Mersenne Twister 엔진
    shuffle(word.begin(), word.end(), generator); // std::shuffle 사용
}

vector<string> chooseTheme() {
    int choice;
    cout << "주제를 선택하세요:\n1. 과일\n2. 동물\n3. 직업\n선택: ";
    cin >> choice;

    switch (choice) {
        case 1: return {"apple", "banana", "mango", "grape", "melon", "
        case 2: return {"lion", "tiger", "elephant", "monkey", "rabbit"
        case 3: return {"doctor", "engineer", "teacher", "nurse", "lawy
        default:
            cout << "잘못된 입력입니다. 기본 주제(과일)로 진행합니다.\n";
            return {"apple", "banana", "cherry", "grape", "melon", "ora
    }
}

int main() {
    // 주제 선택 및 단어 로드
    vector<string> words = chooseTheme();

    // 랜덤 단어 선택
    random_device rd;
    mt19937 generator(rd());
    uniform_int_distribution<int> distribution(0, words.size() - 1);
    string originalWord = words[distribution(generator)];

    // 단어 섞기
    string scrambledWord = originalWord;
    shuffleWord(scrambledWord);

    cout << "섞인 단어: " << scrambledWord << "\n";
    cout << "원래 단어를 맞춰보세요: ";

    string playerGuess;
    cin >> playerGuess;

    if (playerGuess == originalWord) {
        cout << "정답입니다!\n";
    } else {
        cout << "틀렸습니다. 정답은 '" << originalWord << "' 입니다.\n";
    }

    return 0;
}
```

(2) 세부기능 2 점수 시스템

- 재시작 기능을 통해 맞출 때마다 점수를 획득하고, 틀릴 경우 감점하여 최종 점수를 누적한다.

- 입출력

  - 입력: 재시작 여부를 입력 받는다. (y/n)

  - 출력: 다시 하시겠습니까? (y/n):

    현재 점수:

    최종 점수는 (점수)점 입니다!

- 설명

  - 점수 변수 추가

  • int score = 0;으로 점수 초기값을 설정한다.

  - 점수 시스템

  • 정답: 플레이어가 단어를 맞추면 score += 10;으로 점수를 10점 추가한다.

  • 오답: 틀릴 경우엔 score -= 5;으로 점수를 5점 감점한다.

  - 재시작 루프

  • char playAgain 변수를 사용하여 게임을 여러 번 플레이할 수 있도록 한다.

  • 플레이어가 y 또는 Y를 입력할 시 게임이 다시 시작된다.

  - 최종 점수 표시

  • 게임 종료 후 최종 점수를 표시한다.

- 적용된 배운 내용 (예: 반복문, 조건문, 클래스, 함수, 포인터 등)

  While 반복문을 사용해 재시작 여부를 판별하여 점수를 추가하고 있다.

- 코드 스크린샷

```cpp
#include <iostream>
#include <vector>
#include <algorithm> // shuffle
#include <random>    // random_device, mt19937

using namespace std;

void shuffleWord(string &word)
{
    random_device rd;                               // 난수 생성기
    mt19937 generator(rd());                        // Mersenne Twister 엔진
    shuffle(word.begin(), word.end(), generator); // std::shuffle 사용
}

vector<string> chooseTheme()
{
    int choice;
    cout << "주제를 선택하세요:\n1. 과일\n2. 동물\n3. 직업\n선택: ";
    cin >> choice;

    switch (choice)
    {
    case 1:
        return {"apple", "banana", "mango", "grape", "melon", "orange"};
    case 2:
        return {"lion", "tiger", "elephant", "monkey", "rabbit", "mouse"};
    case 3:
        return {"doctor", "engineer", "teacher", "nurse", "lawyer"};
    default:
        cout << "잘못된 입력입니다. 기본 주제(과일)로 진행합니다.\n";
        return {"apple", "banana", "cherry", "grape", "melon", "orange"};
    }
}

int main()
{
    int score = 0;         // 초기 점수 설정
    char playAgain = 'y'; // 재시작 여부

    while (playAgain == 'y' || playAgain == 'Y') // 대소문자
    {
        // 주제 선택 및 단어 로드
        vector<string> words = chooseTheme();

        // 랜덤 단어 선택
        random_device rd;
        mt19937 generator(rd());
        uniform_int_distribution<int> distribution(0, words.size() - 1);
        string originalWord = words[distribution(generator)];

        // 단어 섞기
        string scrambledWord = originalWord;
        shuffleWord(scrambledWord);

        // 게임 시작
        cout << "\n섞인 단어: " << scrambledWord << "\n";
        cout << "원래 단어를 맞춰보세요: ";
```

```cpp
        string playerGuess;
        cin >> playerGuess;

        if (playerGuess == originalWord)
        {
            cout << "정답입니다!\n";
            score += 10; // 정답 시 점수 증가
        }
        else
        {
            cout << "틀렸습니다. 정답은 '" << originalWord << "' 입니다.\n";
            score -= 5; // 오답 시 점수 감소
        }

        cout << "현재 점수: " << score << "\n";

        // 재시작 여부
        cout << "다시 하시겠습니까? (y/n): ";
        cin >> playAgain;
    }

    cout << "\n최종 점수는 " << score << "점 입니다!\n";
    return 0;
}
```

2. 단어 난이도 조절

- 짧은 단어에서 긴 단어로 점차 난이도가 상승하도록 설정한다.

- 입출력

  입출력에서는 추가된 부분이 없다.

- 설명

  - adjustDifficulty 함수를 통한 난이도 조절

  • 단어 길이를 기준으로 짧은 단어 -> 긴 단어 오름차순으로 정렬한다.

  • Sort와 람다 함수를 사용하여 단어의 길이를 비교했다.

  • 초반에는 짧은 단어로 시작해 후반으로 갈수록 긴 단어가 제시되도록 한다.

  - 랜덤 단어 선택 수정

  • 난이도 조절 기능을 추가하였기 때문에 단어 길이에 따라서

  단어가 제시되도록 한다. 랜덤 단어 선택 기능은 제거한다.

- 적용된 배운 내용 (예: 반복문, 조건문, 클래스, 함수, 포인터 등)

  void함수

- 새로 배운 내용

  - Sort는 정렬 알고리즘이다.

  지정된 범위의 요소를 정렬하는데, 이 코드에서는 words.begin()부터

  words.end()까지의 단어를 정렬한다.

  - 람다 함수는 이름이 없는 함수이다.

  코드의 특정 위치에서 즉석으로 정의하여 사용할 수 있다.

  a가 b보다 짧으면 true, 아니면 false를 반환해 단어 길이에 따라 오름차순으로 정렬된다.

- 코드 스크린샷

```cpp
#include <iostream>
#include <vector>
#include <algorithm> // shuffle
#include <random>    // random_device, mt19937

using namespace std;

void shuffleWord(string &word)
{
    random_device rd;                           // 난수 생성기
    mt19937 generator(rd());                    // Mersenne Twister 엔진
    shuffle(word.begin(), word.end(), generator); // std::shuffle 사용
}

vector<string> chooseTheme()
{
    int choice;
    cout << "주제를 선택하세요:\n1. 과일\n2. 동물\n3. 직업\n선택: ";
    cin >> choice;

    switch (choice)
    {
    case 1:
        return {"apple", "banana", "mango", "grape", "melon", "orange"};
    case 2:
        return {"lion", "tiger", "elephant", "monkey", "rabbit", "mouse"};
    case 3:
        return {"doctor", "engineer", "teacher", "nurse", "lawyer"};
    default:
        cout << "잘못된 입력입니다. 기본 주제(과일)로 진행합니다.\n";
        return {"apple", "banana", "cherry", "grape", "melon", "orange"};
    }
}

// 단어 길이에 따라 난이도 조절
void adjustDifficulty(vector<string> &words)
{
    sort(words.begin(), words.end(), [](const string &a, const string &b)
        { return a.size() < b.size(); });
}

int main()
{
    int score = 0;        // 초기 점수 설정
    char playAgain = 'y'; // 재시작 여부

    while (playAgain == 'y' || playAgain == 'Y') // 대소문자
    {
        // 주제 선택 및 단어 로드
        vector<string> words = chooseTheme();

        // 단어 난이도 조절
        adjustDifficulty(words);

        // 게임 시작
        for (const string &originalWord : words)
        {
```

```
58              // 단어 섞기
59              string scrambledWord = originalWord;
60              shuffleWord(scrambledWord);
61
62              cout << "\n섞인 단어: " << scrambledWord << "\n";
63              cout << "원래 단어를 맞춰보세요: ";
64
65              string playerGuess;
66              cin >> playerGuess;
67
68              if (playerGuess == originalWord)
69              {
70                  cout << "정답입니다!\n";
71                  score += 10; // 정답 시 점수 증가
72              }
73              else
74              {
75                  cout << "틀렸습니다. 정답은 '" << originalWord << "' 입니다.\n";
76                  score -= 5; // 오답 시 점수 감소
77              }
78
79              cout << "현재 점수: " << score << "\n";
80
81              // 플레이어가 중간에 종료를 원하면 루프를 빠져나갑니다.
82              cout << "다음 단어로 진행하시겠습니까? (y/n): ";
83              char next;
84              cin >> next;
85              if (next != 'y' && next != 'Y')
86                  break;
87          }
88
89          // 재시작 여부 확인
90          cout << "다시 하시겠습니까? (y/n): ";
91          cin >> playAgain;
92      }
93      cout << "\n최종 점수는 " << score << "점 입니다!\n";
94      return 0;
95  }
```

3. 제한 시간 추가

- 타이머를 사용하여 단어를 맞출 수 있는 제한 시간을 설정하고, 시간이 초과되면 점수가 차감되거나 게임이 종료된다.

- 입출력

    - 출력: 원래의 단어를 맞춰보세요 (제한 시간: 10초):

- 설명

    - waitForInput 함수

    • cin >> playerGuess를 통해 입력 값을 받아와 사용자의 입력을 기다린다.

    • lock_guard<mutex>는 mtx를 잠그고 헤제해 userInputReceived에

    여러 스레드가 동시 접근 못 하도록 한다,

    - startTimer 함수

    • 제한 시간 동안 입력 완료를 기다린다.

    • 제한 시간 초과 시 false를 반환하고, 입력이 완료되면 true를 반환한다.

    • cv.wait_for은 지정된 시간 동안 상태 변화를 기다린다.

    - 시간 초과 처리

    • 제한 시간이 초과되면 inputThread.join()으로 스레드를 종료하고

    입력 값과 정답을 비교하여 정답일 경우엔 점수 증가,

    오답일 경우엔 점수를 감소한다.

    - 추가된 헤더

    • <thread>: 스레드 생성 및 실행.

    • <condition_variable>: 입력 완료 여부를 동기화.

    • <chrono>: 제한 시간 설정.

- 적용된 배운 내용 (예: 반복문, 조건문, 클래스, 함수, 포인터 등)

  전역 변수, void 함수, if 조건문,

- 코드 스크린샷

```cpp
#include <iostream>
#include <vector>
#include <algorithm>          // shuffle
#include <random>             // random_device, mt19937
#include <thread>             // thread
#include <condition_variable> // condition_variable
#include <chrono>             // chrono

using namespace std;

void shuffleWord(string &word)
{
    random_device rd;                              // 난수 생성기
    mt19937 generator(rd());                       // Mersenne Twister 엔진
    shuffle(word.begin(), word.end(), generator); // std::shuffle 사용
}

vector<string> chooseTheme()
{
    int choice;
    cout << "주제를 선택하세요:\n1. 과일\n2. 동물\n3. 직업\n선택: ";
    cin >> choice;

    switch (choice)
    {
    case 1:
        return {"apple", "banana", "mango", "grape", "melon", "orange"};
    case 2:
        return {"lion", "tiger", "elephant", "monkey", "rabbit", "mouse"};
    case 3:
        return {"doctor", "engineer", "teacher", "nurse", "lawyer"};
    default:
        cout << "잘못된 입력입니다. 기본 주제(과일)로 진행합니다.\n";
        return {"apple", "banana", "cherry", "grape", "melon", "orange"};
    }
}

// 단어 길이에 따라 난이도 조절
void adjustDifficulty(vector<string> &words)
{
    sort(words.begin(), words.end(), [](const string &a, const string &b)
         { return a.size() < b.size(); });
}

// 제한 시간 기능 추가
bool userInputReceived = false;
condition_variable cv;
mutex mtx;

void waitForInput(string &playerGuess)
{
    cin >> playerGuess;
    lock_guard<mutex> lock(mtx);
    userInputReceived = true;
    cv.notify_one();
}

```

```cpp
bool startTimer(int timeLimit, string &playerGuess)
{
    unique_lock<mutex> lock(mtx);
    if (cv.wait_for(lock, chrono::seconds(timeLimit), []
                    { return userInputReceived; }))
    {
        return true; // 시간 내 입력 완료
    }
    else
    {
        return false; // 시간 초과
    }
}

int main()
{
    int score = 0;         // 초기 점수 설정
    char playAgain = 'y'; // 재시작 여부

    while (playAgain == 'y' || playAgain == 'Y') // 대소문자
    {
        // 주제 선택 및 단어 로드
        vector<string> words = chooseTheme();

        // 단어 난이도 조절
        adjustDifficulty(words);

        // 게임 시작
        for (const string &originalWord : words)
        {
            // 단어 섞기
            string scrambledWord = originalWord;
            shuffleWord(scrambledWord);

            cout << "\n섞인 단어: " << scrambledWord << "\n";
            cout << "원래 단어를 맞춰보세요 (제한 시간: 10초): ";

            string playerGuess;
            userInputReceived = false;                              // 초기화
            thread inputThread(waitForInput, ref(playerGuess)); // 입력 대기 스레드 시작

            if (startTimer(10, playerGuess))
            {
                inputThread.join(); // 입력이 완료되면 스레드 종료
                if (playerGuess == originalWord)
                {
                    cout << "정답입니다!\n";
                    score += 10; // 정답 시 점수 증가
                }
                else
                {
                    cout << "틀렸습니다. 정답은 '" << originalWord << "' 입니다.\n";
                    score -= 5; // 오답 시 점수 감소
                }
            }
            else
            {
                inputThread.detach(); // 시간 초과 시 스레드 분리
                cout << "\n시간 초과! 정답은 '" << originalWord << "' 입니다.\n";
                score -= 5; // 시간 초과 시 점수 감소
            }

            cout << "현재 점수: " << score << "\n";

            // 플레이어가 중간에 종료를 원하면 루프를 빠져나갑니다.
            cout << "다음 단어로 진행하시겠습니까? (y/n): ";
            char next;
            cin >> next;
            if (next != 'y' && next != 'Y')
                break;
        }

        // 재시작 여부 확인
        cout << "다시 하시겠습니까? (y/n): ";
        cin >> playAgain;
    }
    cout << "\n최종 점수는 " << score << "점 입니다!\n";
    return 0;
}
```

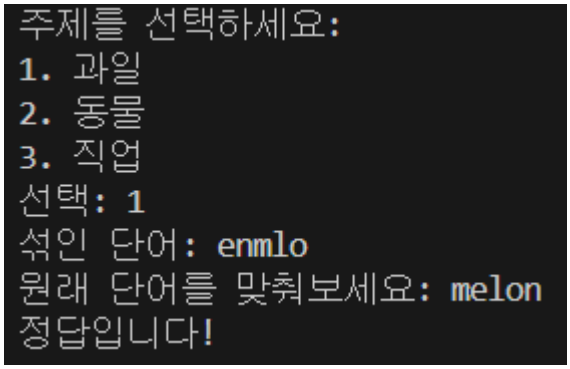
2) 테스트 결과

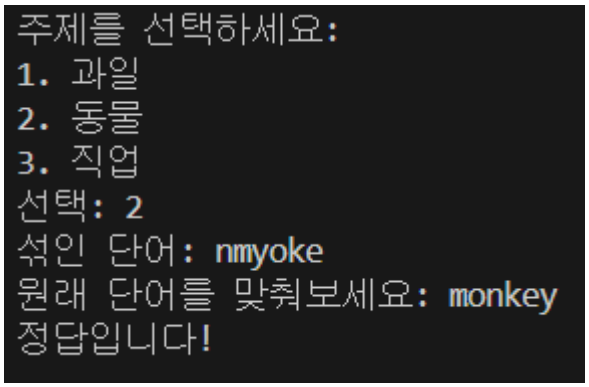
(1) 주제를 택하고 단어를 맞추었을 때와 오답인 경우

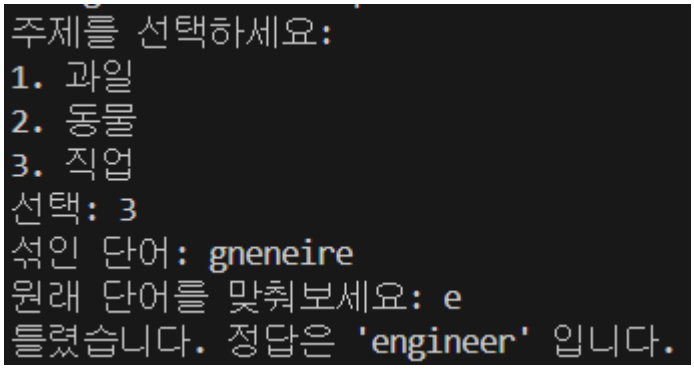
- 설명

세 가지 테마를 사용자가 선택하도록 해 선택할 시 리스트에 있던 단어의 철자가 랜덤으로 출력하게 하고, 잘못 선택할 시 1번 주제로 게임을 플레이하도록 한다. 사용자가 정답을 입력하면 정답인지 오답인지 출력한다. 다음 스크린샷은 모든 경우의 수를 캡쳐한 것이다.

- 테스트 결과 스크린샷







```
주제를 선택하세요:
1. 과일
2. 동물
3. 직업
선택: 4
잘못된 입력입니다. 기본 주제(과일)로 진행합니다.
섞인 단어: nnabaa
원래 단어를 맞춰보세요:
```

(2) 세부기능 2 점수 시스템

- 정답 시 점수 증가, 오답 시 점수 차감을 하고 있다. 재시작 기능을 통해 그만둘 시 최종 점수가 나오도록 한다.

- 테스트 결과 스크린샷

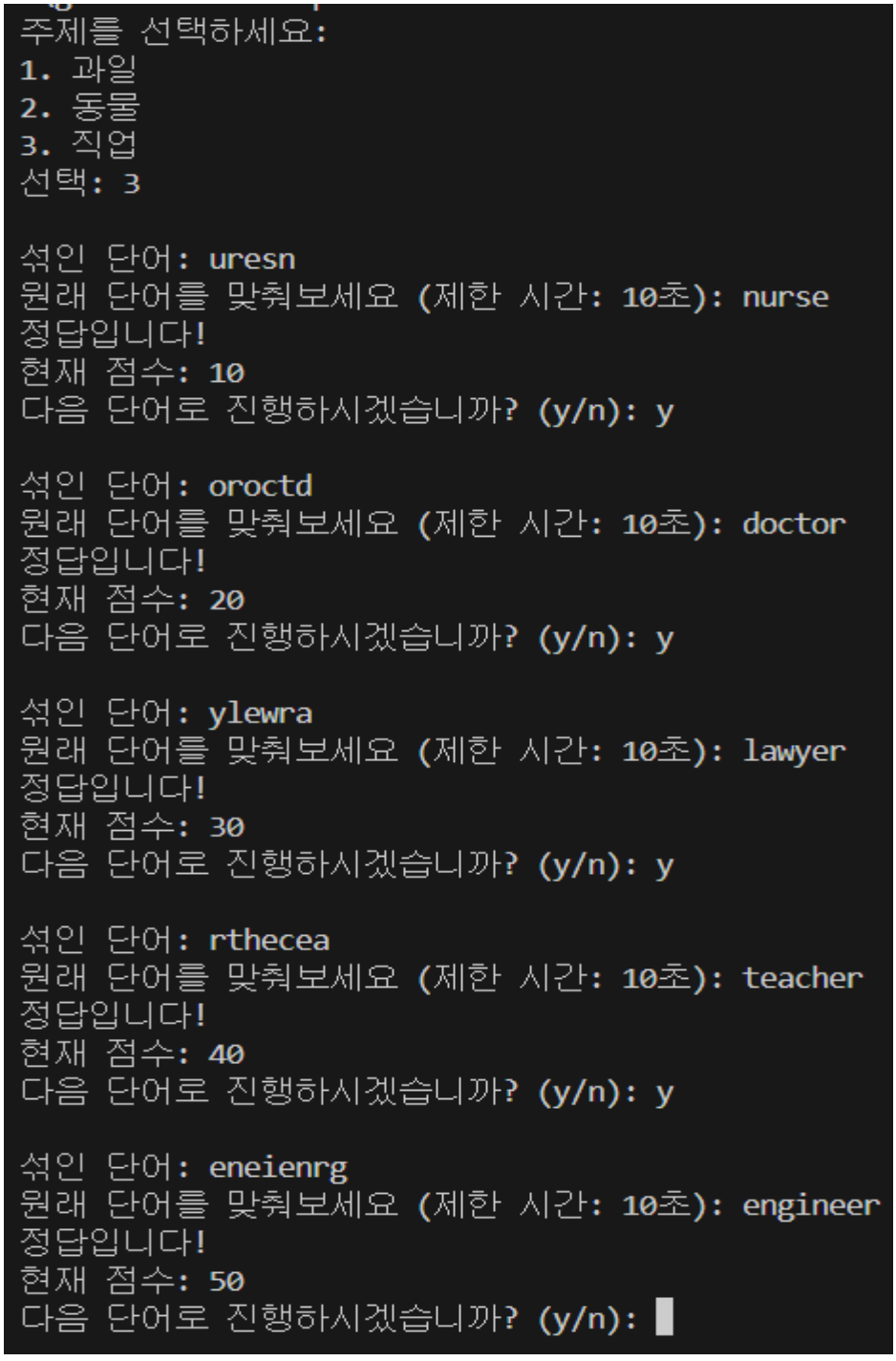
```
주제를 선택하세요:
1. 과일
2. 동물
3. 직업
선택: 1

섞인 단어: peapl
원래 단어를 맞춰보세요 (제한 시간: 10초): apple
정답입니다!
현재 점수: 10
다음 단어로 진행하시겠습니까? (y/n): y

섞인 단어: gnoma
원래 단어를 맞춰보세요 (제한 시간: 10초): mango
정답입니다!
현재 점수: 20
다음 단어로 진행하시겠습니까? (y/n): y

섞인 단어: erapg
원래 단어를 맞춰보세요 (제한 시간: 10초): grap
틀렸습니다. 정답은 'grape' 입니다.
현재 점수: 15
다음 단어로 진행하시겠습니까? (y/n): n
다시 하시겠습니까? (y/n): n

최종 점수는 15점 입니다!
```

(3) 게임의 난이도

- 설명

  게임을 진행할수록 단어의 길이가 길어져 난이도가 상승하도록 한다.

- 테스트 결과 스크린샷

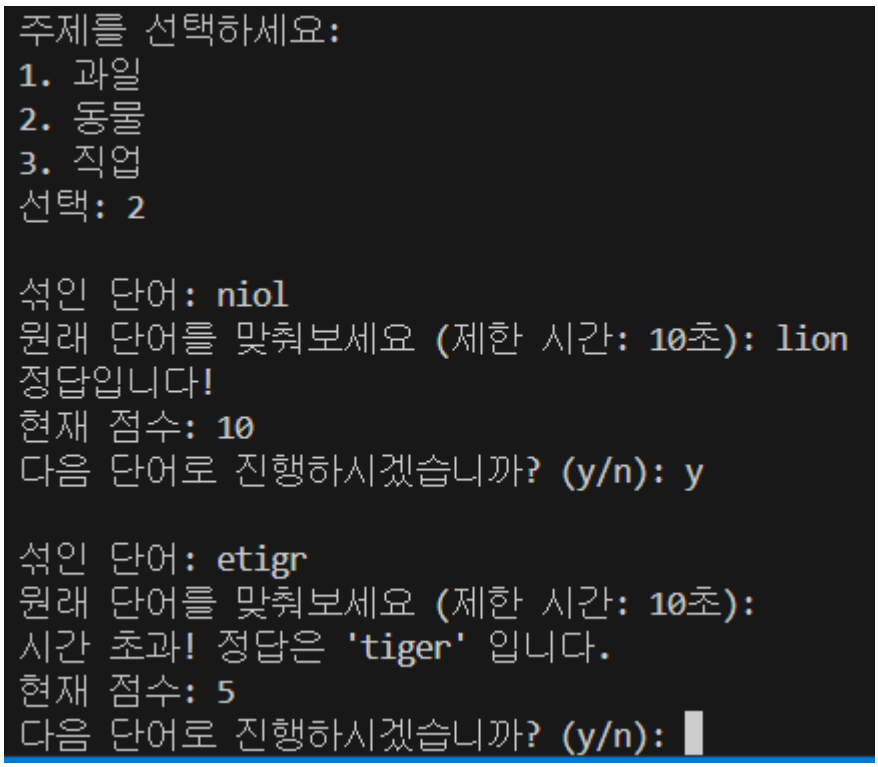


(4) 제한 시간

- 설명

  시간이 초과되면 점수 차감과 함께 정답을 제공한다.

- 테스트 결과 스크린샷



4. 계획 대비 변경 사항

1) 세부기능1

전에는 없던 기능인 잘못된 입력 처리에 관한 내용을 수정했다. Switch 문을 사용하면서 default 일 경우에 1번 주제인 과일로 넘어가도록 만들었다.

5. 프로젝트 일정

업무		11/3	11/10	11/17	11/24	12/1	12/8	12/15	12/22
제안서 작성		완료							
헤더파일공부			완료						
기능1	세부기능1			완료					
	세부기능2			완료					
기능2					완료				
기능3						완료			
기능4							----->		
테스트								----->	
최종보고서									----->