C++

행맨 게임

진척 보고서 #3

제출일자: 12월 15일

제출자명: 김강문

제출자학번: 233958

1. 프로젝트 목표

1) 배경 및 필요성

C++ 수업시간에 배운 내용을 바탕으로 게임을 구현하며 조건문, 반복문, 함수화 등 프로그래밍의 이해와 수준을 발전시키기 위함. 사람들이 많이 알고 하기 쉬운 게임을 구현해 혼자 혹은 여러 명이서 즐길 수 있는 게임인 행맨 게임을 구현하려 한다.

2) 프로젝트 목표

혼자 혹은 여러 명이서 행맨 게임을 플레이 할 수 있는 프로그램을 만드는 것

3) 차별점

기존의 행맨 게임과는 다르게 혼자서도 플레이 할 수 있도록 난이도 설정을 넣어 미리 적어놓은 단어 혹은 상대방이 입력한 단어를 맞추도록 한다.

2. 기능 계획

1) 기능 1 (난이도 선택)

- 난이도 선택에 따른 단어를 출력한다.

(1) 세부 기능 1 (상대방이 입력한 단어 맞추기)

- 난이도 설정을 통해 상대방이 입력한 단어를 맞출 수 있도록 한다.

(1) 세부 기능 2 (미리 입력한 단어 맞추기)

- 난이도 설정을 통해 미리 입력해 놓은 단어를 맞출 수 있도록 한다.

2) 기능 2 (남은 기회와 단어 맞추기)

- 남은 기회와 맞춘 문자를 보여준다. 남은 기회가 0이되면 실패 메시지 출력과 함께 프로그램이 종료되고, 단어를 다 맞추면 성공 메시지 출력과 함께 프로그램이 종료된다.

3. 진척사항

1) 기능 구현

(1) 게임 방법 설정 및 난이도 설정

- 입출력

vector<string> words	// 단어를 저장하기 위한 동적 배열 구조
vector<string> hints	// 힌트(단어의 뜻)을 저장하기 위한 배열 구조
words.txt	// 단어와 단어의 뜻을 저장해 놓은 txt파일
word	// 단어 저장과 입력을 위한 변수
string setting	// 게임 방법(단어를 불러오는)을 결정하기 위한 변수
string difficulty	// 난이도를 설정하기 위한 변수

- 설명

프로그램 시작 시 설정 안내가 출력되고 입력한 값(1, 2 또는 다른 글자)을 확인하고 그에 따른 동작 수행과 안내 메시지를 출력한다. 1을 입력한 경우, 함수를 통해 단어와 힌트를 랜덤으로 가져온다. 2를 입력한 경우, 사용자에게 단어를 입력하도록 한다. 그 외의 경우 잘못 입력했다는 메시지를 출력하고 다시 입력받는다.

설정에 1 또는 2를 입력했을 경우, 다음 단계인 난이도 설정 단계로 넘어가게 되는데 난이도 설정 안내가 출력되고 입력한 값(1, 2, 3 또는 다른 글자)에 따라 최대 시도 횟수를 저장한다. 그 외의 경우 잘못 입력했다는 메시지를 출력하고 다시 입력받는다.

- 적용된 배운 내용 (반복문, 조건문, 함수, 벡터, 클래스, 파일IO, 예외, 코딩 스타일 등)

: 반복문을 이용해 무한 루프를 만들고 일정 조건에 만족할 때 break를 통해 반복문을 탈출하도록 구성했다. 조건문을 통해 설정과 난이도를 선택하도록 하였다. 함수, 벡터, 클래스, 파일 IO 등을 통해 단어와 힌트를 파일에서 불러온 후 랜덤으로 반환받아 저장한다. 파일을 제대로 불러오지 못할 경우를 대비해 예외 처리를 하였고 수업 시간에 배운 코딩 스타일로 변수와 함수, 클래스를 작성하였다.

- 코드 스크린샷

```cpp
// 단어 데이터를 관리하는 클래스
class WordManager {
private:
    vector<string> words;
    vector<string> hints;

public:
    // 단어와 힌트를 파일에서 불러오는 함수
    void LoadWordsFromFile(const string& filename) {
        ifstream file(filename);
        if (!file.is_open()) {
            throw runtime_error("파일을 열 수 없습니다: " + filename);
        }

        string word, hint;
        while (file >> word >> hint) {
            words.push_back(word);
            hints.push_back(hint);
        }
        file.close();

        if (words.empty()) {
            throw runtime_error("파일에 단어가 없습니다.");
        }
    }

    // 랜덤으로 단어와 힌트를 반환하는 함수
    pair<string, string> GetRandomWord() const {
        srand(time(0));
        int index = rand() % words.size();
        return {words[index], hints[index]};
    }

    bool IsEmpty() const {
        return words.empty();
    }
};
```

```cpp
// 메인 함수
int main() {
    WordManager wordManager;

    // 파일에서 단어 불러오기
    try {
        wordManager.LoadWordsFromFile("words.txt");
    } catch (const exception& e) {
        cerr << e.what() << endl;
        return 1;
    }

    if (wordManager.IsEmpty()) {
        cerr << "단어 데이터가 비어있습니다." << endl;
        return 1;
    }
```

```cpp
    string setting;
    string difficulty;
    int max_attempts;

    // 설정 입력
    while (true) {
        cout << "설정 : 1. 혼자(저장된 단어가 랜덤으로 선택됩니다.) 2. 같이(상대방이 맞출 단어를 직접 입력합니다.) : ";
        cin >> setting;

        if (setting == "1" || setting == "2") {
            break;
        }
        cout << "잘못된 입력입니다. 1 또는 2를 입력해주세요.\n\n";
    }

    string word, hint;
    if (setting == "1") {
        auto selectedWord = wordManager.GetRandomWord();
        word = selectedWord.first;
        hint = selectedWord.second;
    } else {
        cout << "상대방이 맞출 단어를 입력해주세요: ";
        cin >> word;
        cout << "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n";
    }
```

```cpp
    // 난이도 설정
    while (true) {
        cout << "\n난이도 선택 : 1. 쉬움(기회: 8) 2. 보통(기회: 7) 3. 어려움(기회: 6) : ";
        cin >> difficulty;

        if (difficulty == "1") {
            max_attempts = 8;
            break;
        } else if (difficulty == "2") {
            max_attempts = 7;
            break;
        } else if (difficulty == "3") {
            max_attempts = 6;
            break;
        }
        cout << "잘못된 입력입니다. 1, 2 또는 3을 입력해주세요.\n";
    }
```

(2) 글자 맞추기

- 입출력

word	// 단어가 저장된 변수
max_attempts	// 최대 시도 횟수
IsGameOver	// 게임 종료 조건을 만족했는지 확인하는 함수
IsWordGuessed	// 글자를 모두 맞췄는지 확인하는 함수
Display	// 현재 상태를 출력하는 함수
input	// 글자를 입력받을 변수
GetRemainingAttempts	// 남은 기회를 반환하는 함수
guess	// 입력한 글자를 저장할 변수
ProcessInput	// 글자가 입력했던 것인지, 맞았는지 확인하는 함수
GetWord	// word에 저장된 단어를 반환하는 함수

- 설명

게임 시작 안내와 함께 루프를 돌며 게임 종료 조건(남은 기회가 있는지)을 확인하고 현재 상태(남은 기회, 맞춘 글자, 틀린 글자 등)를 출력한다. 입력한 글자가 1글자가 아니라면 안내 메시지와 함께 루프의 처음으로 돌아간다. 이미 입력한 글자라면 루프를 돌아 루프의 처음으로 가게 된다. 입력한 글자가 맞았으면 빈칸에 채워지고 아니면 틀렸다는 메시지 출력, 틀린 횟수 증가 그리고 틀린 글자에 추가된다. 루프를 돌며 단어를 다 맞췄을 경우 축하 메시지와 함께 게임을 종료하고 남은 기회를 모두 사용하면 실패 메시지 출력과 게임을 종료한다.

- 적용된 배운 내용 (반복문, 조건문, 클래스, 함수, 벡터 코딩 스타일 등)

: 반복문을 이용해서 틀린 횟수가 최대 횟수보다 작으면 단어를 맞출 수 있도록 구성했다. 조건문과 클래스, 함수, 벡터 등을 통해 현재 상태와 글자를 제대로 입력했는지 확인하고 그에 대한 결과를 출력한다. +

- 코드 스크린샷

```
50  // 게임을 관리하는 클래스
51  class HangmanGame {
52  private:
53      string word;
54      string hint;
55      int max_attempts;
56      int wrong_guesses;
57      vector<bool> guessed;
58      vector<char> wrong_letters;
59      set<char> used_letters; // 이미 입력된 글자
60
61  public:
62      HangmanGame(const string& word, const string& hint, int max_attempts)
63          : word(word), hint(hint), max_attempts(max_attempts), wrong_guesses(0), guessed(word.length(), false) {}
64
65      // 현재 상태를 출력하는 함수
66      void Display() const {
67          for (size_t i = 0; i < word.length(); ++i) {
68              if (guessed[i]) {
69                  cout << word[i];
70              } else {
71                  cout << "_";
72              }
73              cout << " ";
74          }
75          cout << endl;
76          cout << "남은 기회 : " << max_attempts - wrong_guesses << endl;
77          cout << "틀린 글자 : ";
78          for (char c : wrong_letters) {
79              cout << c << " ";
80          }
81          cout << endl;
82      }
```

```
84      // 플레이어의 입력을 처리하는 함수
85      bool ProcessInput(char guess) {
86          // 이미 입력한 글자인지 확인
87          if (used_letters.find(guess) != used_letters.end()) {
88              cout << "이미 입력한 글자입니다. 다시 입력해주세요.\n";
89              return true; // 입력을 다시 요구하므로 틀린 입력으로 처리하지 않음
90          }
91
92          used_letters.insert(guess); // 새로운 입력으로 추가
93
94          bool correct_guess = false;
95          for (size_t i = 0; i < word.length(); ++i) {
96              if (word[i] == guess && !guessed[i]) {
97                  guessed[i] = true;
98                  correct_guess = true;
99              }
100         }
101
102         if (!correct_guess) {
103             wrong_guesses++;
104             wrong_letters.push_back(guess);
105         }
106
107         return correct_guess;
108     }
```

```
    // 게임 종료 여부 확인
    bool IsGameOver() const {
        return wrong_guesses >= max_attempts;
    }

    bool IsWordGuessed() const {
        return all_of(guessed.begin(), guessed.end(), [](bool g) { return g; });
    }

    // 남은 시도 횟수 반환
    int GetRemainingAttempts() const {
        return max_attempts - wrong_guesses;
    }
    // 단어 반환
    string GetWord() const {
        return word;
    }
};
```

```
    HangmanGame game(word, hint, max_attempts);

    cout << "\n행맨 게임을 시작합니다!" << endl;

    while (!game.IsGameOver()) {
        game.Display();

        string input;
        cout << "글자를 입력하세요 (힌트를 보려면 'hint' 입력): ";
        cin >> input;
```

```
        if (input.length() != 1) {
            cout << "하나의 글자를 입력해 주세요.\n";
            continue;
        }

        char guess = input[0];
        if (!game.ProcessInput(guess)) {
            cout << "틀렸습니다!\n";
        }

        if (game.IsWordGuessed()) {
            cout << "\n축하합니다! 단어(" << game.GetWord() << ")를 맞췄습니다." << endl;
            return 0;
        }
    }

    cout << "기회를 모두 사용했습니다. 맞출 단어는 " << game.GetWord() << " 이었습니다." << endl;
    return 0;
}
```

(3) 힌트

- 입출력

hint // 힌트가 저장된 변수

ShowHint // 힌트를 보여주는 함수

- 설명

hint를 입력하면 단어의 뜻을 출력하도록 작성했다. 힌트는 설정을 1로 하고 기회가 1번 남았을 때만 쓸 수 있다. (설정이 2일 경우, 단어를 입력 후 힌트도 입력하도록 했지만 한글을 입력하는 것에 문제가 있어 설정이 1일때만 힌트를 사용할 수 있도록 수정했다.)

- 적용된 배운 내용 (조건문, 함수, 클래스 등)

: 조건문을 통해 설정을 먼저 확인하고 그 후 남은 횟수를 확인하도록 하였다.

- 코드 스크린샷

```
110     // 힌트를 보여주는 함수
111     void ShowHint() const {
112         cout << "힌트를 사용했습니다. 단어의 뜻: " << hint << endl;
113     }
```

```
206         if (input == "hint") {
207             if (setting == "2") {
208                 cout << "setting이 2일 때는 힌트를 사용할 수 없습니다. \n";
209             } else if (game.GetRemainingAttempts() == 1) {
210                 game.ShowHint();
211             } else {
212                 cout << "힌트는 남은 기회가 1일 때만 사용할 수 있습니다.\n";
213             }
214             continue;
215         }
```

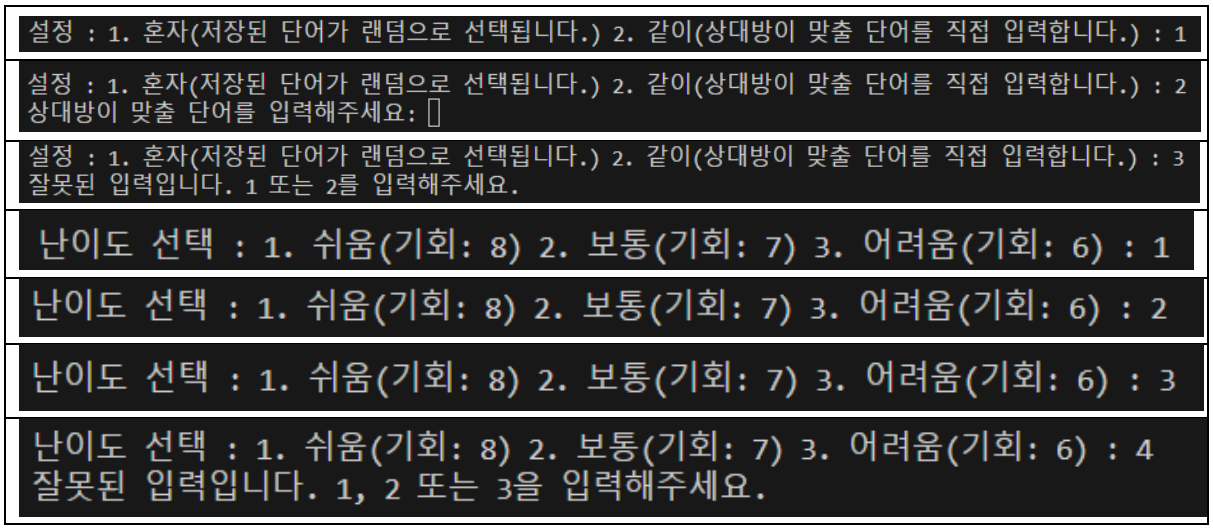
2) 테스트 결과

(1) 게임 방법 설정 및 난이도 설정

- 설명

: 1, 2 ,3 사진은 설정 안내와 그에 따른 출력이고, 4, 5 ,6 ,7 사진은 난이도 안내와 그에 따른 출력

- 테스트 결과 스크린샷



(2) 글자 맞추기

- 설명

: 사진4는 입력했던 글자를 다시 입력했을 경우, 사진5는 글자를 여러 개 입력했을 경우, 사진6은 단어를 맞췄을 경우, 사진7은 기회를 모두 사용했을 경우이다.

- 테스트 결과 스크린샷

행맨 게임을 시작합니다! _ _ _ _ _ _ _ _ 남은 기회 : 6 틀린 글자 : 글자를 입력하세요: p	행맨 게임 시작안내와 현재 상태와 남은 기회, 틀린 글자 출력 후 입력 받기

<table>
<tr><td>

```
p _ _ _ _ _ _
남은 기회 : 6
틀린 글자 :
글자를 입력하세요: o
```

</td><td>전에 입력에 따라 남은 횟수가 바뀌고 맞춘 글자 출력과 못 맞춘 글자 등 출력, 입력 받기</td></tr>
<tr><td>

```
_ _ a _ h
남은 기회 : 5
틀린 글자 : r
글자를 입력하세요 (힌트를 보려면 'hint' 입력): z
틀렸습니다!
```

</td><td>현재 상태 출력과 입력한 글자가 확인 후 틀렸다는 메시지 출력</td></tr>
<tr><td>

```
_ _ _ a _ _ _ _
남은 기회 : 6
틀린 글자 :
글자를 입력하세요: a
이미 입력한 글자입니다. 다시 입력하세요.
```

</td><td>

```
_ _ _ _ _ _ _ _
남은 기회 : 7
틀린 글자 :
글자를 입력하세요: 123
하나씩 입력해 주세요.
```

</td></tr>
<tr><td colspan="2">

```
축하합니다. 단어 (orange)를 맞췄습니다.
```

</td></tr>
<tr><td colspan="2">

```
기회를 모두 사용했습니다. 맞출 단어는 program 이었습니다.
```

</td></tr>
</table>

(2) 힌트

- 설명

: 사진1은 설정을 2로 했을 때 힌트를 입력한 경우, 사진2는 설정을 1로 했을 때 남은 기회가 1보다 많은 경우, 사진3은 설정을 1로 했을 때 남은 기회가 1인 경우이다.

- 테스트 결과 스크린샷

<table>
<tr><td>

```
_ r _ _ r _ _
남은 기회 : 1
틀린 글자 : q w e t y
글자를 입력하세요 (힌트를 보려면 'hint' 입력): hint

setting이 2일 때는 힌트를 사용할 수 없습니다.
```

</td><td></td></tr>
<tr><td>

```
_ _ _ _ _ _ _ _
남은 기회 : 6
틀린 글자 :
글자를 입력하세요 (힌트를 보려면 'hint' 입력): hint

힌트는 남은 기회가 1일 때만 사용할 수 있습니다.
```

</td><td></td></tr>
<tr><td>

```
_ _ e r r y
남은 기회 : 1
틀린 글자 : q w t u i
글자를 입력하세요 (힌트를 보려면 'hint' 입력): hint
힌트를 사용했습니다. 단어의 뜻: 체리
```

</td><td></td></tr>
</table>

4. 계획 대비 변경 사항

1) 난이도 선택

- 이전: 난이도 선택에 따라 맞출 단어를 다르게 설정

- 이후: 설정과 난이도 선택에 따라 게임 방법과 기회 설정

- 사유: 알고 있는 영어 단어가 많지 않았고, 게임 방법과 난이도를 나누어 설정하고 싶었다. 하지만 게임 방법을 먼저 설정하면 난이도에 따라 단어를 저장시키기가 애매했다. 그래서 기회를 다르게 하기로 결정했다.

3) 보완할 기능

- 이후: 힌트 기능 추가

- 사유: 단어의 주제를 고르지 못하므로 힌트 기능을 추가해 단어를 맞출 수 있도록 하기 위함

5. 프로젝트 일정

업무		11/3	11/10	11/17	12/1
제안서 작성		완료			
기능1	세부기능1		완료		
	세부기능2		완료		
기능2				완료	
보완할 기능					완료

업무		12/15	12/22
제안서 작성			
기능21	세부기능1		

	세부기능2		
기능2			
보완할 기능		**완료**	