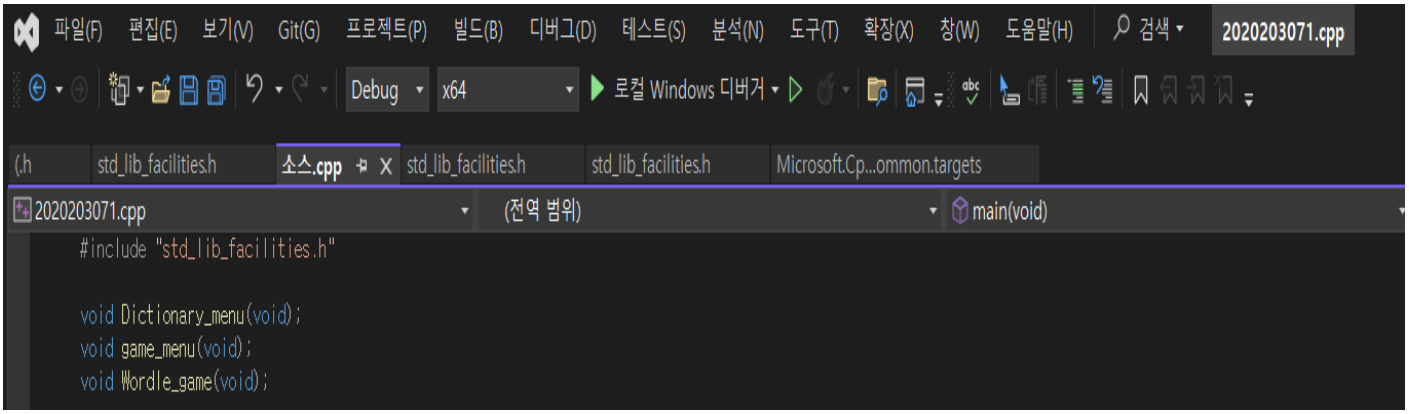
Wordle game by C++

2023/04/05

소프트웨어학부
2020203071
이강우

목 차

1.요구사항에 대한 충족



비주얼 스튜디오를 사용해서 헤더파일 #include "std_lib_facilities.h" 만을 사용하여 코드를 작성하였다. 그리고 main()에 들어갈 함수는 void Dictionary_menu(void), void game_menu(void), void Wordle_game(void)로 총 3개의 함수를 정의하였다.

```
int main(void){
    int put_num;
    cout << "<<< WORDLE >>> ₩n1.Dictionary₩n2.Game₩n3.Exit ₩n₩n";
    do {
        cout << "Choose an option(1 - 3) : ";
        cin >> put_num;
```

cin을 사용하여 메뉴 실행 번호, 추측하는 단어 등을 받았고 cout을 사용해서 메뉴 이름 등 출력값을 받아 출력하였다.

```
fstream file{ "words.txt" };
if (!file) error("can't open output file", "words.txt");

string s;
vector<string> words;

while (file >> s){
    words.push_back(s);
}
sort(words);
cout << "₩n";
for (string w :words ){
    cout << w << "₩n";
}
cout << "₩nA total of " << words.size() << " words are available.₩n";
file.close();
```

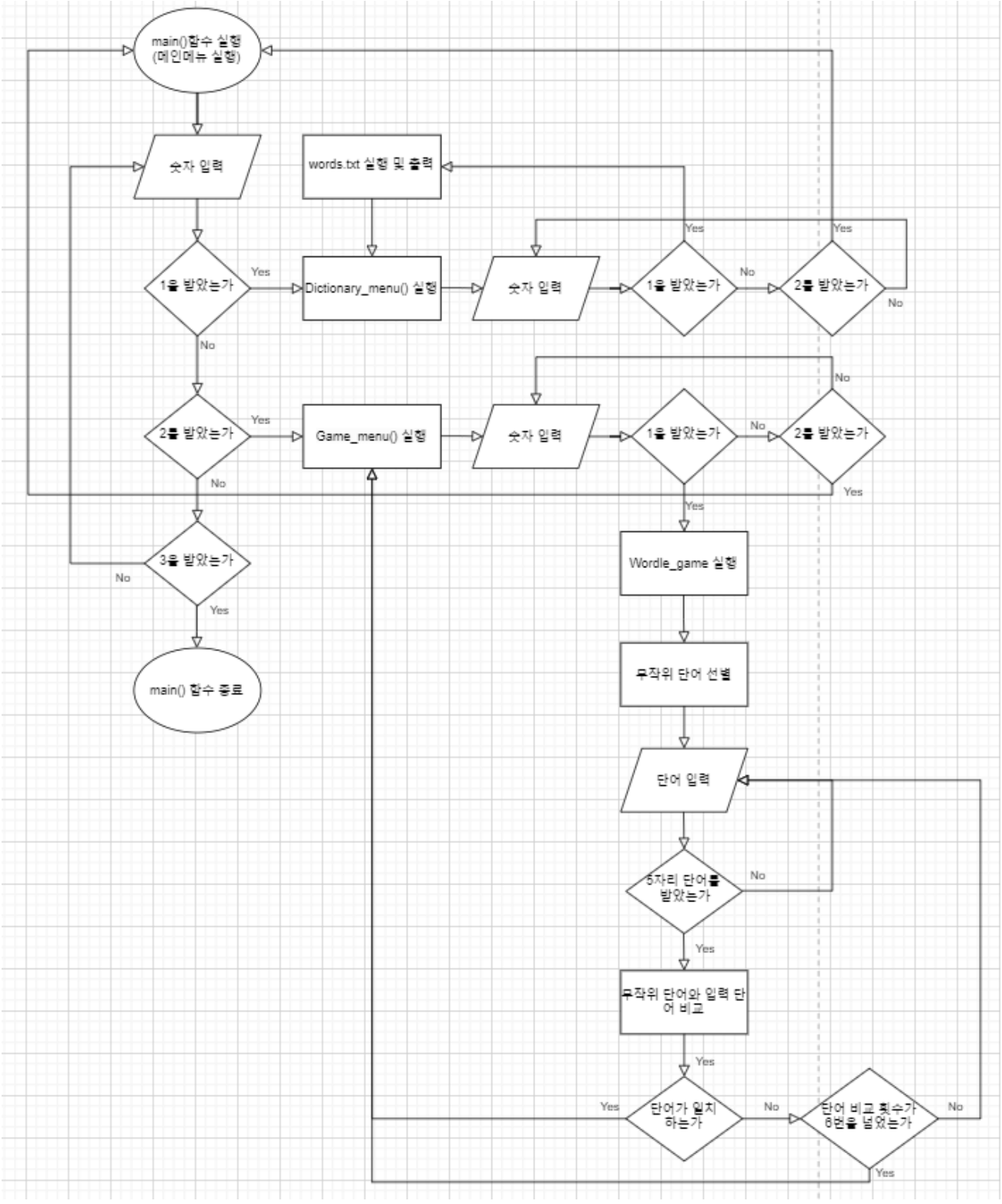
File* 대신 fstream을 사용해서 문서 "words.txt"를 받아서 열어주었고 배열 대신 vector<string>words를 string 타입으로 선언하여 "words.txt"에 있는 단어를 옮겨 받았다. 그리하여 벡터를 탐색하는 범위 기반 반복문 for (string w :words) 를 사용하여 벡터 안에 있는 string 타입 문자를 출력하였다. 그리고 오름차순으로 배열하는 함수 sort()를 사용하여 벡터 words를 오름차순으로 정리하였다.

```
fstream file{ "words.txt" };
if (!file) error("can't open output file", "words.txt");

string file_word;
vector<string> words;

while (file >> file_word){
    words.push_back(file_word);
}

string random_word;
random_word = words[randint((words.size()) - 1)];
```

또한 game을 시작할 때 필요한 벡터 words에 있는 무작위 단어 하나를 추출하기 위해서 randint() 함수를 사용하여 벡터 words에 있는 무작위 단어 하나를 추출하였고 string 타입 random_word에 대입하였다.

2.코드 설명

Overall control flow



메뉴 입력과 출력(범위를 넘을 때)

```
cout << "<<< WORDLE >>> ₩n1.Dictionary₩n2.Game₩n3.Exit ₩n₩n";
do {
    cout << "Choose an option(1 - 3) : ";
    cin >> put_num;

    if (put_num == 1){
        Dictionary_menu();
    }
    else if (put_num == 2) {
        game_menu();
    }
    else if (put_num == 3) {
        cout << "₩nGood bye!!!₩n";
        return 0;
    }
} while (put_num < 1 || put_num>3);
```

main() 함수를 예를 들면, 메인 메뉴를 출력할 때는 위와 같이 1번을 입력할 때 사전 메뉴를 실행하고, 2번을 입력할 때는 게임 메뉴를 실행하고 3번을 입력할 때는 "Good bye"를 출력하고 실행을 종료하였다. 그리고 do while 구문을 사용하여 1보다 작거나 3보다 큰 숫자를 입력하면 다시 숫자를 받도록 만들었다.

단어 목록 출력(정렬된)

```
fstream file{ "words.txt" };
if (!file) error("can't open output file", "words.txt");

string s;
vector<string> words;

while (file >> s){
    words.push_back(s);
}
sort(words);
cout << "₩n";
for (string w :words){
    cout << w << "₩n";
}
cout << "₩nA total of " << words.size() << " words are available.₩n";
file.close();
```

Dictionary_menu() 함수에서 word.txt에 있는 단어를 출력할 때를 보면, word.txt에 있는 단어를 string s에 옮겨 words 벡터에 push_back()함수로 단어를 추가시켰다. 또한 다음에 sort()함수를 사용하여 벡터에 담긴 단어를 오름차순으로 배열하였고, range-based for을 사용하여 words 벡터에 있는 단어를 string형식으로 탐색하여 출력하였다.

단어 선택(무작위로)

```
fstream file{ "words.txt" };
if (!file) error("can't open output file", "words.txt");

string file_word;
vector<string> words;

while (file >> file_word){
    words.push_back(file_word);
}

string random_word;
random_word = words[randint((words.size()) - 1)];
```

아까와 마찬가지로 word.txt에 있는 단어를 string file_word에 옮겨 words 벡터에 push_back()함수로 단어를 추가시켰다. 그리고 randint() 함수를 사용하여 벡터 words[]에 있는 무작위 번째에 있는 단어를 출력하기 위해 words[randint((words.size()) - 1)]을 만들어서 string random_word을 선언하고 여기에 초기화하였다.

추측 단어의 정확성을 정확히 표시했는지

```
for (int i = 0;i < 5;i++){
    for (int j = 0;j < 5;j++){
        if (guess_word[i] == random_word[i]){
            symbol[i] = "*";
            break;
        }
        else if (guess_word[i] == random_word[j] && i != j){
            symbol[i] = "#";
            break;
        }
        else
            symbol[i] = "@";
    }
}
cout << "==> ";
for(int i=0;i<5;i++)
 cout<<symbol[i];
```

나는 반복문을 사용하여 무작위 단어와 추측 단어의 각 자리의 문자를 서로 비교하여 추측 단어의 특정 자릿수의 문자가 무작위 단어의 문자와 같으면'*', 그렇지 않고 무작위 단어의 다른 자릿수의 문자와 같으면 '#', 그것도 아니면 '@'을 출력하게 했다.

예를 들어, 반복문을 사용하여 i+1번째 자리에 있는 추측단어의 문자와 무작위단어의 문자를 비교하고 맞으면 symbol[i]에 '*'를 초기화 시켜주고 i에 대하여 j에 관한 반복문은 의미가 없으므로 j에 관한 반복문을 탈출시켰다. 그러나 i+1번째 자리의 문자가 서로 시켜줬다. 또한 여기서도 특정 i에 관하여 i+1 자릿수에 관한 기호는 정해졌으므로 j에 관한 반복문을 탈출시켰다. 그리하여 j는 0부터 4까지 반복하는 동안 위 두 조건을 만족시키지 못하면 symbol[i]에 '@'를 초기화시키도록 하였다. 그리하여 모든 반복문이 끝나면 symbol[]에 초기화된 값을 출력하도록 하였다.

게임 마무리 조건을 정확히 확인했는지

```
for (int Number = 1;Number < 7;Number++){
    string guess_word;
    string symbol[5];
    do { ... } while (guess_word.size() != 5);

    for (int i = 0;i < 5;i++) { ... }
    cout << "==> ";
    for(int i=0;i<5;i++)
     cout<<symbol[i];
    cout << "₩n";
    if (guess_word == random_word){
        file.close();
        cout << "₩nCongratulation!!! Your guess is correct.₩n";
        game_menu();
    }
}
file.close();
cout << "₩nOoops!!! You lost all your chances. The correct answer is ₩""<<random_word<<"₩".₩n";
game_menu();
```

아까 말한 추측 단어와 무작위 단어를 비교 과정 거친 후, 추측 단어와 무작위 단어가 같으면 파일을 닫고 'Congratulation!!! Your guess is correct.'를 출력하고 game_menu()로 돌아가게 하여서 게임을 종료하였다. 그리고 추측 횟수를 6번을 넘어서는 것은 파일을 닫고 'Ooops!!! You lost all your chances. The correct answer is '무작위 단어'.'를 출력하고 game_menu()로 돌아가게 하여서 게임을 종료하였다.

3.마무리

이번 과제를 해결하는데 있어서 조건문의 탈출과 반복문의 활용이 가장 핵심사항이지 않았나 생각이 든다. 반복문 do while 구문을 이용하여 다양한 메뉴에서 숫자를 입력 받아서 실행할 때 다른 범위의 숫자를 입력 받으면 다시 숫자를 입력받게 할 수 있었다. 또한 If 조건문을 사용하여 Wordle_game을 실행할 때 무작위 단어와 추측 단어 사이의 연관성을 추측하는 코드를 짤 때 큰 도움이 되었다. 그리고 강의에서 배운 vector 타입, string 타입 그리고 파일 입출력 스트림을 배운것이 코드를 만드는 근간이 되었다.

이번 wordle game 코드를 작성하면서 wordle game을 작동 원리를 제대로 구현 못 한 것이 아쉬웠다. 예를 들어, 원래 wordle game은 저장된 정해진 문자 안에서 추측 문자를 선택해야 추측 조건에 성립하고 다시 정해진 문자를 받아내고 영어 글자만 받고 숫자나 한글과 같은 다른 언어는 받지 않는다. 다음 기회에는 그러한 기능까지 구현하는 코드를 짜보고 싶다.