**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №2

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-33Б |  | преподаватель каф. ИУ5 |
| Кузьмин Денис |  | Нардид А.Н. |
| Подпись и дата: |  | Подпись и дата: |

Текст программы

Файл game.py

import random  
  
  
def get\_dict(filename):  
 words\_dict = {}  
 with open(filename, 'r', encoding='utf-8') as f:  
 for line in f:  
 word, description = line.strip().split(':')  
 words\_dict[word] = description  
 return words\_dict  
  
  
def read\_file(result):  
 filename = 'txt\_files\\win.txt' if result == 'win' else 'txt\_files\\fails3.txt'  
 with open(filename, 'r', encoding='utf-8') as f:  
 msg\_result = f.read()  
 return f'{msg\_result}\n'  
  
  
def get\_word(words\_list):  
 words\_dict = get\_dict('txt\_files\\wnd.txt')  
 word = random.choice(words\_list)  
 word\_description = words\_dict.get(word)  
 return word, word\_description  
  
  
def get\_words\_list(): return list(get\_dict('txt\_files\\wnd.txt').keys())  
  
  
def get\_lives(): return 3  
  
  
def return\_len(item): return len(item)  
  
  
def word\_list\_is\_not\_empty(word\_list): return len(word\_list) - 1 != 0  
  
  
def delete\_word(current\_word, word\_list): word\_list.remove(current\_word)  
  
  
def get\_guessed(): return 0  
  
  
def words\_in\_list(words\_list): return len(words\_list) != 0  
  
  
def get\_unique\_letters(current\_word): return set([letter for letter in current\_word])  
  
  
def get\_guessed\_letters(): return set('')  
  
  
def create\_table(current\_word):  
 table = ''  
 for i in range(len(current\_word)):  
 table += '\u25A0 '  
 return table  
  
  
def check\_new\_game(continue\_game): return 'да' in continue\_game or 'нет' in continue\_game  
  
  
def continue\_game(continue\_game): return continue\_game == 'да'  
  
  
def add\_guessed\_letter(guessed\_letters, answer): guessed\_letters.add(answer)  
  
  
def is\_letter\_guessed(guessed\_letters, answer): return answer in guessed\_letters  
  
  
def word\_guessed(guessed, unique\_letters): return guessed == len(unique\_letters)  
  
  
def check\_word(current\_word, answer): return len(current\_word) == len(answer)  
  
  
def lose(): return 0  
  
  
def lives\_min(lives): return lives - 1  
  
  
def guessed\_plus(guessed): return guessed + 1  
  
  
def is\_alive(lives): return lives > 0  
  
  
def prompt(msg): return input(msg).lower()  
  
  
def show\_message(\*msg): print(\*msg)  
  
  
def letter\_in\_word(current\_word, answer): return answer in current\_word  
  
  
def is\_word\_correct(current\_word, answer): return answer == current\_word  
  
  
def check\_input(answer, current\_word): return len(answer) == 1 or len(answer) == len(current\_word)  
  
  
def msg\_lives(lives):  
 hearts = f'|❤️x{lives} '  
 filename = 'txt\_files\\'  
 if lives == 3:  
 filename += 'fails0.txt'  
 elif lives == 2:  
 filename += 'fails1.txt'  
 elif lives == 1:  
 filename += 'fails2.txt'  
 with open(filename, 'r', encoding='utf-8') as f:  
 pic = f.read()  
 return f'{pic}\n{hearts}'  
  
  
def new\_table(letter, current\_word, oldtable):  
 letter\_list = [lt for lt in current\_word]  
 count = current\_word.count(letter)  
 oldtable = oldtable.split(' ')  
  
 if count != 1:  
 while count != 0:  
 index = letter\_list.index(letter)  
 oldtable[index] = letter  
 letter\_list[index] = '-'  
 count -= 1  
 else:  
 index = letter\_list.index(letter)  
 oldtable[index] = letter  
 oldtable = ' '.join(oldtable)  
 return oldtable

Файл game\_logic.py

import game\_logic  
  
words\_list = game\_logic.get\_words\_list()  
  
while game\_logic.words\_in\_list(words\_list):  
 current\_word, word\_description = game\_logic.get\_word(words\_list)  
 lives = game\_logic.get\_lives()  
 guessed = game\_logic.get\_guessed()  
 table = game\_logic.create\_table(current\_word)  
 unique\_letters = game\_logic.get\_unique\_letters(current\_word)  
 guessed\_letters = game\_logic.get\_guessed\_letters()  
 count\_of\_words = game\_logic.return\_len(words\_list)  
 msg\_letter\_or\_word = 'Введите букву или слово целиком: '  
 msg\_wrong = '\nНеправильно. Вы теряете жизнь :( \n'  
 msg\_viselica\_win = game\_logic.read\_file('win')  
 msg\_win = '\nВы угадали слово ' + current\_word + '! Приз в студию!'  
 msg\_viselica\_lose = game\_logic.read\_file('lose')  
 msg\_lose = 'Вы проиграли'  
 msg\_letter\_guessed = 'Эта буква уже была угадана\n'  
 msg\_continue\_game = '\nХотите поиграть еще? Да | Нет \n'  
 msg\_start\_new\_game = f'\nПоехали! Осталось угадать {count\_of\_words - 1} слов'  
 msg\_no\_words = 'Вы угадали все слова'  
 msg\_exit = 'До встречи!'  
 msg\_ = ''  
  
 while game\_logic.is\_alive(lives):  
 if game\_logic.word\_guessed(guessed, unique\_letters):  
 break  
 game\_logic.show\_message(game\_logic.msg\_lives(lives) + table + '\n' + word\_description)  
 answer = game\_logic.prompt(msg\_letter\_or\_word)  
 while True:  
 if game\_logic.check\_input(answer, current\_word):  
 break  
 else:  
 answer = game\_logic.prompt(msg\_letter\_or\_word)  
 if game\_logic.check\_word(current\_word, answer):  
 if game\_logic.is\_word\_correct(current\_word, answer):  
 break  
 else:  
 lives = game\_logic.lose()  
 elif game\_logic.letter\_in\_word(current\_word, answer):  
 if game\_logic.is\_letter\_guessed(guessed\_letters, answer):  
 game\_logic.show\_message(msg\_letter\_guessed)  
 else:  
 game\_logic.add\_guessed\_letter(guessed\_letters, answer)  
 table = game\_logic.new\_table(answer, current\_word, table)  
 guessed = game\_logic.guessed\_plus(guessed)  
 game\_logic.show\_message(msg\_)  
 else:  
 lives = game\_logic.lives\_min(lives)  
 game\_logic.show\_message(msg\_wrong, msg\_)  
  
 if game\_logic.is\_alive(lives):  
 game\_logic.show\_message(msg\_viselica\_win, msg\_win)  
 while True:  
 continue\_game = game\_logic.prompt(msg\_continue\_game)  
 if game\_logic.check\_new\_game(continue\_game):  
 break  
 else:  
 continue\_game = game\_logic.prompt(msg\_continue\_game)  
 if game\_logic.continue\_game(continue\_game):  
 if game\_logic.word\_list\_is\_not\_empty(words\_list):  
 game\_logic.delete\_word(current\_word, words\_list)  
 game\_logic.show\_message(msg\_start\_new\_game)  
 continue  
 else:  
 game\_logic.show\_message(msg\_no\_words)  
 break  
 else:  
 game\_logic.show\_message(msg\_exit)  
 break  
 else:  
 game\_logic.show\_message(msg\_viselica\_lose, msg\_lose)  
 break  
else:  
 game\_logic.show\_message(msg\_no\_words)