МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВВГУ»)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАЛИЗА ДАННЫХ
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

ОТЧЕТ
ПО ТЕКСТОВОЙ RPG-ИГРЕ

по дисциплине
«Информатика и основы программирования»

Студент
гр. БИН-25-3 _____ В.Е. Соловьева

Ассистент
преподавателя _____ М.В. Водяницкий

Владивосток 2026

Задание

Вы работаете программистом в небольшой японской компании на заре игровой индустрии. Компания разрабатывает свою первую экспериментальную игру - текстовую RPG, которая должна запускаться прямо в консоли и погружать игрока в атмосферу подземелий, опасностей и развития персонажа

Ваша задача - реализовать прототип игры, который демонстрирует основные игровые механики: характеристики персонажа, бои, прокачку, инвентарь и случайные события

1. Общая идея программы

Программа представляет собой консольную текстовую RPG, в которой игрок:

– создает персонажа (выбор расы)

– получает случайные характеристики в рамках выбранной расы

– исследует подземелье, состоящее из случайных комнат

– сражается с врагами, находит предметы и улучшает персонажа

– повышает уровень и распределяет очки характеристик

– принимает решения, влияющие на дальнейший путь

Игра работает в пошаговом режиме и управляется вводом команд с клавиатуры

2. Создание персонажа

2.1 Выбор расы

В начале игры пользователь выбирает расу персонажа (например):

– Человек

– Эльф

– Дворф

Каждая раса задает диапазоны генерации характеристик

2.2 Характеристики персонажа

Характеристики генерируются случайным образом при создании персонажа, но в допустимых пределах для выбранной расы

Пример набора характеристик (можно расширять):

– HP - здоровье

– Attack - сила атаки

– Defense - защита

– Agility - ловкость (влияет на уклонение)

– Height - рост

– Weight - вес

Допускается, что некоторые характеристики влияют друг на друга (например, рост и вес влияют на уклонение или скорость)

3. Опыт и уровни

– Персонаж получает опыт за победу над врагами
– При накоплении нужного количества опыта повышается уровень
– Каждый новый уровень дает очки прокачки

3.1 Прокачка характеристик

Игрок может распределять очки вручную между характеристиками

Пример:

– +1 к атаке
– +2 к HP
– +1 к ловкости

Распределение очков выполняется в комнатах отдыха

4. Инвентарь и экипировка

4.1 Инвентарь

Инвентарь хранит предметы:

– зелья (лечение и др.)
– монеты
– оружие
– прочие предметы

Игрок может:

– просматривать инвентарь
– использовать предметы
– выбрасывать любые предметы

4.2 Экипировка

В инвентаре должны быть отдельные слоты:

– оружие
– броня

Экипированные предметы влияют на характеристики персонажа

5. Подземелье и комнаты

5.1 Структура подземелья

– Игра начинается в подземелье
– Подземелье состоит из комнат

– После каждой комнаты игрок выбирает путь:

1) налево

2) направо

Развилка есть после каждой комнаты

5.2 Типы комнат

Комнаты генерируются случайно:

– Боевая комната - бой с врагом

– Комната отдыха - без событий

– Комната с сундуком - предметы или золото

Возможны комбинации:

– слева враг, справа сундук

– оба врага

– обе комнаты отдыха

5.3 Видимость комнат

Перед выбором направления игрок:

– иногда знает, что находится дальше

– иногда не знает (темно, неизвестно)

Информация о видимости определяется случайно.

6. Враги и сложность

– Враги генерируются случайно

– У врагов есть характеристики (HP, атака, защита и т.д.)

– С каждым этажом подземелья сложность возрастает

– Каждые N комнат или действий происходит переход на новый этаж

7. Боевая система

Бой происходит в пошаговом режиме:

Пример действий игрока:

– атаковать

– использовать предмет

– попытаться уклониться

Учитываются:

– характеристики игрока

– экипировка

– случайные факторы (уклонение, критический удар)

8. Предметы и добыча

– Враги и сундуки могут давать:

1) зелья

2) оружие

3) другие предметы

– Полученные предметы добавляются в инвентарь

– При нехватке места игрок решает, что выбросить

9. Хранение данных

Допускается (но не обязательно):

– сохранение состояния игры в файл

– использование формата JSON для хранения:

1) характеристик персонажа

2) инвентаря

3) текущего этажа

10. Пример работы программы (фрагмент)

Выберите расу:

1 - Человек

2 - Эльф

3 - Дворф

> 2

Ваш персонаж создан!

HP: 85

ATK: 12

DEF: 6

AFI: 14

Вы входите в подземелье...

Перед вами развилка.

(1) Слева: ???

(2) Справа: Комната отдыха

Куда пойти?

> 1

11. Ограничения и требования

– Программа консольная

– Управление через текстовое меню и ввод команд

– Язык программирования - не ограничен (в том числе можно Python)

– Код должен быть читаемым и логически структурированным, можно делить на разные файлы

Содержание

Введение

Работая программистом в небольшой японской компании на заре игровой индустрии, нам предстоит разработать нашу первую экспериментальную игру — текстовую RPG, которая должна запускаться прямо в консоли и погружать игрока в атмосферу подземелий, опасностей и развития персонажа.

Мы стоим на пороге новой эры. В то время как крупные игроки экспериментируют с графикой, наша маленькая команда в Токио верит в силу воображения. Наша первая игра не будет сверкать пикселями — она будет сверкать словами. Мы создаем текстовое подземелье, мир, который рождается в диалоге между программой и игроком. Ограничения консоли — наш главный вызов и наша творческая сила.

Наша задача — реализовать прототип игры, который демонстрирует основные игровые механики: характеристики персонажа, бои, прокачку, инвентарь и случайные события.

Планируется:

– Создать несколько играбельных классов

– Создать несколько классов противников

– Реализовать систему базовых характеристик персонажа (сила, ловкость, выносливость) и их влияние на геймплей

– Реализовать различные комнаты

– Внедрить систему прокачки персонажа через получение опыта и повышение уровня

– Реализовать упрощенную систему инвентаря для управления снаряжением и предметами

– Создать пошаговую боевую систему против разнотипных противников

1 Гейм-дизайн

Название разработанного прототипа пошаговой RPG-игры — «FEMBOY ALFA ORC DUNGEON».

Игрок внезапно оказывается в подземелье и в течение его прохождения поднимается.

Существуют четыре класса или же расы:

– Фембойчик инкуб
– Эльфийка
– Альфа оборотень
– Кошкодевочка

Каждый класс имеет собственный диапазон возможных значений характеристик и одну уникальную способность.

Характеристики персонажей:

– Максимальное здоровье
– Атака
– Защита
– Ловкость
– Рост
– Вес

Характеристики персонажей, за исключением роста и веса, могут улучшаться с помощью очков прокачки, которые могут быть получены при повышении уровня.

Существует некоторое количество предметов, которые можно найти в сундуках или получить при победе над противниками:

– Золотые монеты
– Зелье здоровья
– Зелье молодости

Существует три возможные комнаты:

– Комната отдыха
– Боевая комната
– Комната с сундуком

Через каждые десять комнат игрок поднимается на этаж выше.

Бои пошаговые. Защита восстанавливается после каждой битвы, здоровье — нет.

Максимальное число слотов в инвентаре — пять.

2 Выполнение работы

2.1 Приветствие

В начале присутствует приветствие.

На рисунке 1 представлен код этого приветствия.

```
print('='*100)
print('Добро пожаловать в игру «FEMBOY ALFA ORK DUNGEON»!')
print('='*100)
```

Рисунок 1 – Приветствие

Пояснение работы программы:

1) Первая и треться строчки отвечают за оформление и представляют собой разделители;

2) Вторая строчка выводит приветствие.

Таким образом приветствие завершено.

2.2 Проверка введенных данных

Введенные данные проверяются по одному критерию корректности, а именно, являются ли они числом. Для этого была создана функция.

На рисунке 2 представлен код этой функции.

```
# check if int
import sys
def is_int(x):
    try:
        int(x)
    except ValueError:
        print('Должно быть введено ЦЕЛОЕ ЧИСЛО!!!!!!')
        sys.exit()
```

Рисунок 2 – Функция для проверки введенных данных

Пояснение работы программы:

1) Импортируется библиотека sys;

2) Создается функция, в которой будеут прверяться введенные данные;

3) Пробуется перевести введенные данные, строку, в целочисленные;

4) Если введенные данные переводятся в целочисленный тип данных, то проверка пройдена;

5) Если получается ошибка, то программа завершается с помощью функции exit из модуля sys.

Таким образом получается функция для проверки введенных данных.

Но так как это не единственный критерий корректности данных, данная функция будет только частью этой проверки.

Полная проверка корректности данных представлена на рисунке 3.

```
# check if the entered data is valid
is_int(class_choice)
if (int(class_choice) > len(classes)) or (int(class_choice)
   < 1):
    print(f'ОТ 1 ДО {len(classes)}!!!!!')
    sys.exit()
```

Рисунок 3 – Полная проверка введенных данных

Пояснение работы программы:

1) Используется функция, которая разбиралась выше;

2) Если введенные данные не проходят проверку, то программа завершается;

3) Если введенные данные проходят проверку, то они продолжают проверяться;

4) С помощью условного оператора if, проверяется соответсвует ли введенное значение диапазону представленных вариантов ответа;

5) Если введенные данные не проходят проверку, то программа завершается с помощью функции exit из модуля sys.

Данная проверка появляется после каждого ввода данных, изменяются только переменные.

2.3 Выбор класса

Вывод выбора класса представлен на рисунке 4.

```
# class selection
classes = ['Фембойчик инкуб', 'Эльфийка', 'Альфа оборотень', '
   Кошкодевочка']
print('Выбор класса')
print('='*100)
for i in range(len(classes)):
    print(f'{i+1}. {classes[i]}')
print('='*100)
class_choice = input(f'Выберите кем вы хотите быть введите(
   целое число от 1 до {len(classes)}): ')

# check if the entered data is valid
is_int(class_choice)
if (int(class_choice) > len(classes)) or (int(class_choice)
   < 1):
    print(f'ОТ 1 ДО {len(classes)}!!!!!')
    sys.exit()
```

Рисунок 4 – Выбор класса

Пояснение работы программы:

1) Создается список, содержащий все возможные классы: Фембойчик инкуб, Эльфийка, Альфа оборотень и Кошкодевочка;

2) Вывод текста «Выбор класса»;

3) С помощью цикла for название каждого класса из списка classes выводится на каждой строчке с соответствующим номером;

4) Осуществляется ввод данных пользователем, который сохраняется в переменную class_choice;

5) Далее осуществляется проверка введенных данных class_choice, представленная ранее в пункте 2.2.

В зависимости от выбранного класса, с помощью модуля random осуществляется присвоение характеристик в заданном диапазоне.

Код присвоения характеристик представлен на рисунке 5.

```
# class choice
your_class = classes[int(class_choice)-1]

import random

if your_class == 'Фембойчик инкуб':
    stats = {'MaxHP': random.randint(10,13),
             'ATK': random.randint(2,3),
             'DEF': random.randint(1,3),
             'Agility': random.randint(7,8),
             'Height': random.randint(150,155),
             'Weight': random.randint(35,40),
             'Weapon': 'Отсутствует',
             'Armor': 'Отсутствует'}
if your_class == 'Эльфийка':
    stats = {'MaxHP': random.randint(10,14),
             'ATK': random.randint(1,3),
             'DEF': random.randint(3,4),
             'Agility': random.randint(5,7),
             'Height': random.randint(160,165),
             'Weight': random.randint(45,50),
             'Weapon': 'Лук',
             'Armor': 'Легкая броня'}
if your_class == 'Альфа оборотень':
    stats = {'MaxHP': random.randint(15,20),
             'ATK': random.randint(5,8),
             'DEF': random.randint(8,10),
             'Agility': random.randint(2,3),
             'Height': random.randint(185,190),
             'Weight': random.randint(90,100),
             'Weapon': 'Клыки',
             'Armor': 'Средняя броня'}
if your_class == 'Кошкодевочка':
    stats = {'MaxHP': random.randint(10,15),
             'ATK': random.randint(4,5),
             'DEF': random.randint(1,2),
             'Agility': random.randint(7,9),
             'Height': random.randint(155,160),
             'Weight': random.randint(40,45),
             'Weapon': 'Когти',
             'Armor': 'Отсутствует'}
```

Рисунок 5 – Присвоение характеристик

Пояснение работы программы:

1) Создается список your_class, в котором содержатся названия всех возможных классов;

2) С помощью оператора import мпортируется модуль random;

3) С помощью условного оператора if, в зависимости от выбранного класса, создается словарь stats, содержащий случайные значения характеристик из заданного диапазона;

4) Для генерации случайных характеристик в заданном диапазоне используется функция randint из модуля random;

5) В созданном словаре также присутствуют значения оружия и брони, которые заданы вручную.

Таким образом осуществляется присвоение характеристик выбранному персонажу.

Далее заданные характеристики изменяются в зависимости от заданной экипировки (Оружия, брони).

Код представлен на рисунке 6.

```
if stats['Armor'] == 'Легкая броня':
    stats['DEF'] += 2
if stats['Armor'] == 'Средняя броня':
    stats['DEF'] += 4
if stats['Weapon'] != 'Отсутствует':
    stats['ATK'] += 2
```

Рисунок 6 – Экипировка

Пояснение работы программы:

1) С помощью оператора if проверятся значение в словаре stats под ключом Armor;

2) Если значением является «Легкая броня», то в словаре stats к значению под ключом DEF прибавляется две единицы;

3) Если значением является «Средняя броня», то в словаре stats к значению под ключом DEF прибавляется четыре единицы;

4) С помощью оператора if проверятся значение в словаре stats под ключом Weapon;

5) Если значением не является «Отсутствует», то в словаре stats к значению под ключом ATK прибавляется две единицы.

Таким образом получаются полные характеристики выбранного персонажа.

Далее осуществляется вывод полученных характеристик, представленный на рисунке 7.

```
print('='*100)
print(f'Вы - {your_class}!')
print('='*100)
print('Ваши характеристики: ')
print(f'HP - {stats['MaxHP']}')
print(f'ATK - {stats['ATK']}')
print(f'DEF - {stats['DEF']}')
print(f'Agility - {stats['Agility']}')
print(f'Height - {stats['Height']}')
print(f'Weight - {stats['Weight']}')
print(f'Weapon - {stats['Weapon']}')
print(f'Armor - {stats['Armor']}')
```

Рисунок 7 – Вывод характеристик

Пояснение работы программы:

1) Выводится название выбранного класса;

2) На разных строках выводятся полученные характеристики.

Так как каждый персонаж имеют особую способность, далее, в зависимости от класса, выводится информация о его особой способности.

Код представлен на рисунке 8.

```
if your_class == 'Фембойчик инкуб':
    print('Особая способность Фембойчика инкуба - Соблазнить!')
    print('Фембойчик инкуб может соблазнить противника!')
    print('Шанс успеха - 50%!')
    print('Когда Фембойчик инкуб встречается с Орком или Бесом, шанс успеха возрастает до 100%!')
    print('Когда Фембойчик инкуб встречается с Бобром, шанс успеха снижается до 0%!')
if your_class == 'Эльфийка':
    print('Особая способность Эльфийки - Запугать!')
    print('Эльфийка может запугать противника!')
    print('Шанс успеха - 50%!')
    print('Когда Эльфийка встречается с Бобром, шанс успеха возрастает до 100%!')
    print('Когда Эльфийка встречается с Ведьмой, шанс успеха снижается до 0%!')
if your_class == 'Альфа оборотень':
    print('Особая способность Альфа оборотня - Выть!')
    print('Альфа оборотень может издать вой, который полностью уничтожает врага!')
    print('Шанс успеха - 50%!')
    print('Когда Альфа оборотень встречается с Бобром, шанс успеха возрастает до 100%!')
    print('Когда Альфа оборотень встречается с Ведьмой, шанс успеха снижается до 0%!')
if your_class == 'Кошкодевочка':
    print('Особая способность Кошкодевочки - Умилить противника!')
    print('Кошкодевочка может сбежать с поля боя!')
    print('Шанс успеха - 50%!')
    print('Когда Кошкодевочка встречается с Орком или Бобром, шанс успеха возрастает до 100%!')
    print('Когда Кошкодевочка встречается с Ведьмой, шанс успеха снижается до 0%!')
```

Рисунок 8 – Особая способность

Пояснение работы программы:

1) С помощью условного оператора if проверяется содержимое переменной your_-class, в которой содержится информация о выбранном классе;

2) Если выбранный класс соответсвует «Фембойчик инкуб», то выводится информация о его особой способности;

3) Если выбранный класс соответсвует «Эльфийка», то выводится информация о ее особой способности;

4) Если выбранный класс соответсвует «Альфа оборотень», то выводится информация о его особой способности;

5) Если выбранный класс соответсвует «Кошкодевочка», то выводится информация о его особой способности.

Таким образом, выбор персонажа завершается.

2.4 Противники

Создаются противники и их характеристики.

На рисунке 9 представлен код.

```
# opponents
opponents = ['Орк', 'Ведьма', 'Бобр', 'Бес']

orc_stats = {'MaxHP': random.randint(15,20),
             'ATK': random.randint(1,2),
             'DEF': random.randint(4,5),
             'Agility': random.randint(1,2),
             'Height': random.randint(200,250),
             'Weight': random.randint(250,300)}

witch_stats = {'MaxHP': random.randint(10,15),
               'ATK': random.randint(4,5),
               'DEF': random.randint(1,2),
               'Agility': random.randint(4,5),
               'Height': random.randint(160,165),
               'Weight': random.randint(45,50)}

beaver_stats = {'MaxHP': random.randint(2,3),
                'ATK': random.randint(1,2),
                'DEF': random.randint(1,2),
                'Agility': random.randint(5,6),
                'Height': 35,
                'Weight': 30}

imp_stats = {'MaxHP': random.randint(5,6),
             'ATK': random.randint(4,5),
             'DEF': random.randint(2,3),
             'Agility': random.randint(5,6),
             'Height': random.randint(140,145),
             'Weight': random.randint(30,35)}
```

Рисунок 9 – Противники

Пояснение работы программы:

1) Создается список opponents, в котором содержатся названия всех возможных противников: Орк, Ведьма, Бобр и Бес;

2) Создается словарь orc_stats, содержащий случайные значения характеристик Орка из заданного диапазона;

3) Для генерации случайных характеристик в заданном диапазоне используется функция randint из модуля random;

4) Создаются точно такие же словари witch_stats для Ведьмы, beaver_stats для Бобра и imp_stats для Беса, но с разными диапазонами значений характеристик.

Таким образом созданы противники и заданы их характеристики.

2.5 Инвентарь

Создание инвентаря.

На рисунке 10 представлен код, задающий инвентарь.

```
# inventory
inventory = [0] # first value - number of gold coins
# max number of items in inventory - 5
weapon = [stats['Weapon']]
armor = [stats['Armor']]
MaxHP = stats['MaxHP']
hp = stats['MaxHP'] # can only be regenerated via health
    potions
# defence refreshes after every battle
```

Рисунок 10 – Инвентарь

Пояснение работы программы:

1) Создается переменная inventory, в которой и будут храниться предметы;

2) Изначально в данной переменной содержится одно значение равное нулю, оно представляет собой количество золотых монет и существует всегда;

3) Максимальное количество предметов в инвентаре - 5;

4) Создается пременная weapon, которой присваивается значение из словаря stats по ключу Weapon;

5) Создается пременная aeapon, которой присваивается значение из словаря stats по ключу Aeapon;

6) Создается пременная MaxHP, которой присваивается значение из словаря stats по ключу MaxHP;

7) Создается пременная hp, которой присваивается значение из словаря stats по ключу MaxHp;

8) Отдельные переменные для максимального здоровья и здоровья создаются, потому что после потери какого-либо количества здоровья в бою, его можно будет восстановить только с помощью Зелья Здоровья, автоматически оно не восстанавливается.

Таким образом создается инвентарь для хранения предметов.

Но это еще не все, потому что необходимо создать меню для инвентаря.

Для этого была создана функция, которая представлена на рисунке 11.

```
def Inventory():
    global inventory
    global weapon
    global armor
    global MaxHP
    global hp

    print('='*100)

    # show inventory
    print('Просмотр инвентаря:')
    print(f'Инвентарь - {inventory}')
    print(f'Оружие - {weapon[0]}')
    print(f'Броня - {armor[0]}')
    print('='*100)
    print('1. Закрыть инвентарь')
    print('2. Использовать предмет')
    print('3. Выбросить предмет')
    inventory_choice = input('Что будете делать? ')

    is_int(inventory_choice)
    if (int(inventory_choice) > 3) or (int(inventory_choice)
    < 1):
        print('ОТ 1 ДО 3!!!!!')
        sys.exit()

    if inventory_choice == '1':
        return
```

Рисунок 11 – Функция инвентарь

Пояснение работы программы:

1) Создается функция Inventory;

2) Внутри этой функции значения inventory, weapon, armor, MaxHp и hp, заданные на предыдущем шаге, объявляются глобальными;

3) Выводится текст «Просмотр инвентаря:»;

4) На разных строках выводится инвентарь, который соответсвует переменной inventory, оружие, которое соответсвует переменной weapon и броня, которая соответствут переменной armor;

5) На разных строках выводятся пронумерованные опции: Закрыть инвентарь, Использовать предмет, Выбросить предмет;

6) Производится ввод данных пользователем, который сохраняются в переменную inventory_choice;

7) Далее осуществляется проверка введенных данных inventory_choice, представленная ранее в пункте 2.2;

8) С помощью условно оператора if проверяется, что пользователь ввел «1», что соответсвует закрытию инвентаря;

9) С помощью return функция завершается.

Соответственно, в зависимости от выбранного пользователем пункта, будут открываться разные меню.

При выборе пользователем пункта 1 (Закрыть инвентарь), инвентарь закрывается.

2.5.1 Использовать предмет

Выбор пункта 2 (Использовать предмет) представлен на рисунке 12.

```
# use an item
if inventory_choice == '2':
    print('='*100)

    for i in range(len(inventory)):
        print(f'{i+1}. {inventory[i]}')
    item_choice = input('Выберите предмет для использования
    индекс(): ')

    #check is the entered data is valid
    is_int(item_choice)
    if (int(item_choice) > len(inventory)) or (int(
    item_choice) < 1):
        print(f'ОТ 1 ДО {len(inventory)}!!!!!')
        sys.exit()
```

Рисунок 12 – Использовать предмет

Пояснение работы программы:

1) С помощью условно оператора if проверяется, что пользователь ввел «2», что соответсвует использованию предмета;

2) С помощью цикла for на разных строках выводятся пронумерованные элементы инвентаря, списка inventory;

3) Производится ввод данных пользователем, который сохраняются в переменную item_choice;

4) Далее осуществляется проверка введенных данных item_choice, представленная ранее в пункте 2.2.

На разных слотах могут быть разные предметы, поэтому для каждой позиции, кроме первой, код одинаковый.

Код представлен на рисунке 13.

```
if item_choice == '1':
    print('='*100)

    print('Вы не можете использовать золото в данный момент!')
    Inventory()
if item_choice == '2':
    print('='*100)

    if inventory[1] == {'Зелье здоровья'}:
        if (hp < MaxHP) and ((hp + 2)<=MaxHP):
            hp += 2
            print('Зелье здоровья успешно использовано!')
            print(f'Ваше текущее здоровье: {hp}')
        elif (hp < MaxHP) and ((hp + 2)>=MaxHP):
            hp = MaxHP
            print('Зелье здоровья успешно использовано!')
            print(f'Ваше текущее здоровье: {hp}')
        else:
            print('Невозможно использовать Зелье здоровья!')
            print('Ваше здоровье полное!')
            Inventory()
    if inventory[1] == {'Зелье молодости'}:
        print('Вам нет необходимости использовать Зелье
   молодости! Вы и так молоды!')
        Inventory()
```

Рисунок 13 – Использование предмета

Пояснение работы программы:

1) С помощью условно оператора if проверяется, что пользователь ввел «1», что соответсвует слоту с золотыми монетами;

2) Выводится сообщение «Вы не можете использовать золото в данный момент!»;

3) Вызывается функция Inventory, открывающая меню инвентаря;

4) С помощью условно оператора if проверяется, что пользователь ввел «2», что соответсвует одному из четырех слотов с предметами;

5) С помощью условно оператора if проверяется, что элемент под соответствующим индексом соответствует «Зелье здоровья»;

6) С помощью условно оператора if проверяется, что текущее здоровье меньше максимального как минимум на две единицы;

7) Если это так, то в текущему здоровью прибавляются две единицы, выводится сообщение о том, что Зелье здоровья было успешно использовано, и текущее здоровье выводится в консоль;

8) С помощью условно оператора if проверяется, что текущее здоровье меньше максимального меньше чем на две единицы;

9) Если это так, то текущее здоровье восстанавливается до значения максимального здоровья;

10) С помощью условно оператора if проверяется, что текущее и максимальное здоровье равны;

11) Выводится сообщение о том, что использовать Зелье здоровья невозможно и что текущее здоровье уже полное;

12) Вызывается функция Inventory, открывающая меню инвентаря;

13) С помощью условно оператора if проверяется, что элемент под соответствующим индексом соответствует «Зелье молодости»;

14) Выводится сообщение «Вам нет необходимости использовать Зелье молодости! Вы и так молоды!»;

15) Вызывается функция Inventory, открывающая меню инвентаря.

На оставшихся трех позициях данный блок кода, начиная с четвертой строки, повторяется.

2.5.2 Выбросить предмет

Выбор пункта 3 (Выбросить предмет) представлен на рисунке 14.

```
# throw an item away
if inventory_choice == '3':
    print('='*100)
    for i in range(len(inventory)):
        print(f'{i+1}. {inventory[i]}')
    item_choice = input('Какой предмет хотите выбросить? ')

    is_int(item_choice)
    if (int(item_choice) > len(inventory)) or (int(
   item_choice) < 1):
        print(f'ОТ 1 ДО {len(inventory)}!!!!!!')
        sys.exit()

    if int(item_choice) >= 2:
        del inventory[int(item_choice)-1]
        print('Предмет был успешно выброшен!')
        print(f'Ваш текущий инвентарь: {inventory}')
        Inventory()
    else:
        inventory[0] = 0
        print('Предмет был успешно выброшен!')
        print(f'Ваш текущий инвентарь: {inventory}')
        Inventory()
```

Рисунок 14 – Выбрасывание предмета

Пояснение работы программы:

1) С помощью условно оператора if проверяется, что пользователь ввел «1», что соответсвует выбрасыванию предмета;

2) С помощью цикла for на разных строках выводятся пронумерованные элементы инвентаря, списка inventory;

3) Производится ввод данных пользователем, который сохраняются в переменную item_choice;

4) Далее осуществляется проверка введенных данных item_choice, представленная ранее в пункте 2.2;

5) С помощью условно оператора if проверяется, что пользователь ввел значение большее или равное двум, что соответсвует любому из четырех слотов с предметами;

6) С помощью del производится удаление элемента из инвентаря, списка inventory, по индексу;

7) Выводится сообщение о том, что предмет был успешно выброшен;

8) Выводится текущий инвентарь;

9) Вызывается функция Inventory, открывающая меню инвентаря;

10) С помощью else проверяется, что пользователь ввел единственное оставшееся значение равное единице, что соответсвует слоту с золотыми монетами;

11) В таком случае количество золотых монет в инвентаре, переменной inventory, по индексу приравнивается к нулю;

12) Выводится сообщение о том, что предмет был успешно выброшен;

13) Выводится текущий инвентарь;

14) Вызывается функция Inventory, открывающая меню инвентаря.

Таким образом осуществляется выбрасывание предмета из инвентаря.

На этом функция Inventory, отвечающая за вывод меню инвентаря, заканчивается.

2.6 Уровень и прокачка

Была создана возожность прокачки характеристик, использую очки опыта, получаемые за каждый новый уровень.

Код представлен на рисунке 15.

```
#experience
exp = 0
# lvl increases every 5 exp
lvl = 1
points = 0

# experience reset
def exp_reset():
    global exp
    global lvl
    global points
    if exp == 5:
        lvl += 1
        points += 1
        exp = 0
```

Рисунок 15 – Уровень

Пояснение работы программы:

1) Создается переменная exp, равная нулю, которая хранит количество опыта;

2) Создается переменная lvl, равная единице, которая хранит значение текущего уровня игрока;

3) Создается переменная points, равная нулю, которая хранит количество очков прокачки;

4) Создается функция exp_reset, которая отвечает за сброс опыта, повышение уровня и получение очков прокачки;

5) Переменные exp, lvl и point объявляются глобальными;

6) С помощью условно оператора if проверяется, если количество опыта равно пяти, то есть если переменная exp равна пяти;

7) Если это так, то уровень повышается на один, то есть значение переменной lvl увеличивается на один;

8) Количество очков прокачки увеличивается на один, то есть переменная points увеличивается на один;

9) Опыт сбрасывается, то есть переменная exp приравнивается к нулю.

Таким образом осуществляется повышение уровня игрока.

Далее необходимо сделать функцию, которая будет позволять тратить очки прокачки на прокачку характеристик.

Код, выводящий меню такой функции, представлен на рисунке 16.

```
def Upgrades():
    global stats
    global points
    global lvl
    global exp
    print('='*100)
    print(f'На данный момент у вас {points} очков прокачки')
    print('1. Выйти')
    print('2. Перейти к прокачке')
    upgrades_choice = input('Что будете делать? ')

    # check if the entered data is valid
    is_int(upgrades_choice)
    if (int(upgrades_choice) > 2) or (int(upgrades_choice) <
    1):
        print(f'ОТ 1 ДО 2!!!!!')
        sys.exit()

    # exit
    if upgrades_choice == '1':
        return
```

Рисунок 16 – Прокачка

Пояснение работы программы:

1) Создается функция Upgrades, которая будет отвечать за меню прокачку;

2) Значения stats, points, lvl и exp объявляются глобальными;

3) Выводится сообщение о текущем количестве очков прокачки игрока;

4) На разных строках с соответствующей нумерацией выводятся опции: Выйти, Перейти к прокачке;

5) Осуществляется ввод данных пользователем, который сохраняется в переменную upgrades_choice;

6) Далее осуществляется проверка введенных данных upgrades_choice, представленная ранее в пункте 2.2;

7) С помощью условно оператора if проверяется, что пользователь ввел «1», что соответсвует пункту «Выйти»;

8) С помощью return функция завершается.

Таким образом выводится меню прокачки и в соответствии от выбора пользователя, открываюся разные меню, и при выборе пункта «1», который соответсвует выходу из меню прокачки, функция завершается.

2.6.1 Перейти к прокачке

Если пользователь выбирает пункт для перехода к прокачке, то открывается новое меню.

Код представлен на рисунке 17.

```
# upgrade
elif upgrades_choice == '2':
    print('='*100)

    upgrades = ['+1MaxHP', '+1ATK', '+1DEF', '+1Agility']

    # insufficient points
    if points == 0:
        print('Недостаточно очков прокачки!')
        print(f'Текущее количество - {points}')
        Upgrades()

    # sufficient points
    else:
        print('Выберите характеристику для прокачки: ')
        for i in range(4):
            print(f'{i+1}. {upgrades[i]}')
        upgrade_choice = input('Прокачать: ')

        # check if the entered data is valid
        is_int(upgrade_choice)
        if (int(upgrade_choice) > 4) or (int(upgrade_choice) < 1):
            print(f'ОТ 1 ДО 4!!!!!')
            sys.exit()
```

Рисунок 17 – Выбор характеристики для прокачки

Пояснение работы программы:

1) С помощью условно оператора elif проверяется, что пользователь ввел «2», что соответствует переходу к прокачке;

2) Создается список upgrades, который содержит все возможные характеристики для прокачки;

3) С помощью условного оператора if происходит проверка количества очков характеристик, то есть значения переменной points;

4) Если очков прокачки нет, то есть переменная points равна нулю, то выводится сообщение о недостаточном количестве очков прокачки и вызывается функция Upgrades, которая возвращает пользователя в меню прокачки;

5) В противном случае, то есть когда очки прокачки есть, выводится меню с выбором характеристик для прокачки;

6) С помощью цикла for переменные с названиями характеристик выводятся на разных строках с соответствующей нумерацией;

7) Пользователь вводит номер характеристики, которую хотел бы прокачать, и данное значение сохраняется в переменную udgrade_choice;

8) Далее осуществляется проверка введенных данных upgrade_choice, представленная ранее в пункте 2.2.

Таким образом осуществляется выбор характеристики для прокачки.

Далее необходимо осуществить саму прокачку, код для этого представлен на рисунке 18.

```
if upgrade_choice == '1':
    print('='*100)
    stats['MaxHP'] += 1
    points -= 1
    print('Ваше максимальное здоровье было увеличено на 1!')
    print(f'Текущее максимальное здоровье - {stats['MaxHP']}')
    Upgrades()
if upgrade_choice == '2':
    print('='*100)
    stats['ATK'] += 1
    points -= 1
    print('Ваша атака была увеличена на 1!')
    print(f'Текущая атака - {stats['ATK']}')
    Upgrades()
if upgrade_choice == '3':
    print('='*100)
    stats['DEF'] += 1
    points -= 1
    print('Ваша защита была увеличена на 1!')
    print(f'Текущая защита - {stats['DEF']}')
    Upgrades()
if upgrade_choice == '4':
    print('='*100)

    # not max agility
    if stats['Agility'] < 10:
        stats['Agility'] += 1
        points -= 1
        print('Ваша ловкость была увеличена на 1!')
        print(f'Текущее ловкость - {stats['Agility']}')
        Upgrades()

    # max agility
    else:
        print('Ваша ловкость достигла максимального значения - 10!')
        print('Дальнейшая прокачка невозможна!')
        Upgrades()
```

Рисунок 18 – Прокачка выбранной характеристики

Пояснение работы программы:

1) Если пользователь выбрал пункт 1, что соответсвует максимальному здоровью, то максимальное здоровье увеличивается на одну единицу и текущее значение максимального здоровья выводится в консоль;

2) Отнимается одно очко прокачки, то есть от переменной points отнимается единица;

3) Если пользователь выбрал пункт 2, что соответсвует матаке, то атака увеличивается на одну единицу и текущее значение атаки выводится в консоль;

4) Отнимается одно очко прокачки, то есть от переменной points отнимается единица;

5) Если пользователь выбрал пункт 3, что соответсвует защите, то защита увеличивается на одну единицу и текущее значение защиты выводится в консоль;

6) Отнимается одно очко прокачки, то есть от переменной points отнимается единица;

7) Если пользователь выбрал пункт 4, что соответсвует ловкости, то с помощью оператора if проверяется достигла ли она своего максимального значения;

8) Если ловкость не достигла своего максимального значения, десяти, то ловкость увеличивается на одну единицу и текущее значение ловкости выводится в консоль;

9) Отнимается одно очко прокачки, то есть от переменной points отнимается единица;

10) Если ловкость достигла своегое максимального значения, десяти, то выводится сообщение о том, что дальнейшая ее прокачка не возможна и вызывается функция Upgrades, которая возвращает пользователя в меню прокачки.

Таким образом осуществляется прокачка характеристик.

2.7 Комнаты

В подземелье создаются комнаты.

Код представлен на рисунке 19.

```
# rooms
floor = 1
room_count = 0
total_rooms = 0
rooms = ['Боевая комната', 'Комната отдыха', 'Комната с сундуком']

# next floor
def next_floor():
    global floor
    global room_count

    if room_count == 10:
        floor += 1

        orc_stats['MaxHP'] += 2
        orc_stats['ATK'] += 2
        orc_stats['DEF'] += 2

        witch_stats['MaxHP'] += 2
        witch_stats['ATK'] += 2
        witch_stats['DEF'] += 2

        beaver_stats['MaxHP'] += 2
        beaver_stats['ATK'] += 2
        beaver_stats['DEF'] += 2

        imp_stats['MaxHP'] += 2
        imp_stats['ATK'] += 2
        imp_stats['DEF'] += 2

        room_count = 0
```

Рисунок 19 – Комнаты

Пояснение работы программы:

1) Создается переменная floor, равная единице, в которой хранится номер этажа;

2) Создается переменная room_count, равная нулю, в которой хранится номер комнаты на этаже;

3) Создается переменная total_rooms, равная нулю, в которой хранится количество пройденных комнат;

4) Создается список rooms, в котором хранятся названия всех существующий комнат: Боевая комната, Комната отдыха, Комната с сундуком;

5) Создается функция next_floor, которая отвечает за переход на следующий этаж и поднятие сложности;

6) Переменные floor и room_count объявляются глобальными;

7) С помощью условного оператора if проверяется, были ли пройдены десять комнат на этаже, то есть равна ли переменная room_count десяти;

8) Если было пройдено десять комнат на этаже, то номер этажа повышается на единицу и максимальное здоровье, атака и защита противников повышается на две единицы;

9) Переменная room_count сбрасывается, то есть приравнивается к нулю.

Таким образом отслеживаются пройденные комнаты и осуществляется повышение сложности при переходе на следующий этаж.

2.7.1 Развилка

Была создана функция, которая отвечает за вывод меню Развилки. Код представлен на рисунке 20.

```
def directions():
    left = random.choice(rooms)
    right = random.choice(rooms)

    known_or_not_left = [left, 'Тьма темная']
    known_or_not_right = [right, 'Тьма темная']

    print('='*100)
    print('Перед вами развилка')
    print(f'Слева - {random.choice(known_or_not_left)}')
    print(f'Справа - {random.choice(known_or_not_right)}')
    print('1. Налево')
    print('2. Направо')
    direction_choice = input('Куда пойдете? ')

    is_int(direction_choice)
    if (int(direction_choice) > 2) or (int(direction_choice)
    < 1):
        print(f'ОТ 1 ДО 2!!!!!')
        sys.exit()

    if direction_choice == '1':
        print('='*100)
        print('Вы свернули налево')
        print(f'Слева - {left}')
        if left == rooms[0]:
            battle_room()
        elif left == rooms[1]:
            rest_room()
        elif left == rooms[2]:
            reward_room()

    if direction_choice == '2':
        print('='*100)
        print('Вы свернули направо')
        print(f'Справа - {right}')
        if right == rooms[0]:
            battle_room()
        if right == rooms[1]:
            rest_room()
        elif right == rooms[2]:
            reward_room()
```

Рисунок 20 – Развилка

Пояснение работы программы:

1) Создается функция directions, которая отвечает за Развилку;

2) Создается переменная left, которой присваивается случайное значение из списка rooms;

3) Создается переменная right, которой присваивается случайное значение из списка rooms;

4) Случайные значения присваиваются с помощью функции choice из модуля random;

5) Создается список known_or_not_left, в котором содержится переменная left и «Тьма темная» (То есть неизвестно);

6) Создается список known_or_not_right, в котором содержится переменная right и «Тьма темная» (То есть неизвестно);

7) Выводится сообщение «Перед вами развилка»;

8) Выводится сообщение о том, что находится слева, для этого снова используется функция choice из модуля random, которая достает случайное значение из списка known_or_not_left;

9) Выводится сообщение о том, что находится справа, для этого снова используется функция choice из модуля random, которая достает случайное значение из списка known_or_not_right;

10) Выводятся варианты «Налево» и «Направо» с соответствующей нумерацией;

11) Пользователь вводит номер выбранного варианта и это значение сохраняется в переменную direction_choice;

12) Далее осуществляется проверка введенных данных direction_choice, представленная ранее в пункте 2.2;

13) Если пользователь выбрал пункт 1, что соответсвует «Налево», выводится сообщение о том, какая комната находилась слева и вызывается соответствующая функция: battle_room для Боевой комнаты, rest_room для Комнаты отдыха и reward_room для Комнаты с сундуком;

14) Если пользователь выбрал пункт 2, что соответсвует «Направо», выводится сообщение о том, какая комната находилась справа и вызывается соответствующая функция: battle_room для Боевой комнаты, rest_room для Комнаты отдыха и reward_room для Комнаты с сундуком;

Таким образом осуществляется Развилка.

2.7.2 Комната отдыха

Была создана функция, которая отвечает за вывод меню Комнаты отдыха. Код представлен на рисунке 21.

```
def rest_room():
    print('='*100)

    # room counter
    global floor
    global room_count
    global total_rooms
    room_count+=1
    print(f'ЭТАЖ {floor}')
    print(f'Комната {room_count}')
    print(f'Всего пройдено комнат: {total_rooms}')
    total_rooms += 1
    next_floor()

    print('='*100)

    # rest room
    print('Вы попали в комнату отдыха!')
    print('1. Покинуть комнату отдыха')
    print('2. Прокачка')
    print('3. Открыть инвентарь')
    print('ВНИМАНИЕ!!! Если вы открываете инвентарь или
    прокачку в комнате отдыха, после их закрытия автоматически
    следует развилка!')
    rest_room_choice = input('Что будете делать? ')

    # check if the entered data is valid
    is_int(rest_room_choice)
    if (int(rest_room_choice) > 3) or (int(rest_room_choice)
    < 1):
        print(f'ОТ 1 ДО 3!!!!!')
        sys.exit()

    if rest_room_choice == '1':
        directions()
    if rest_room_choice == '2':
        Upgrades()
        directions()
    if rest_room_choice == '3':
        Inventory()
        directions()
```

Рисунок 21 – Комната отдыха

Пояснение работы программы:

1) Создается функция rest_room, отвечающая за Комнату отдыха;

2) Переменные floor, room_count и total_rooms объявляются глобальными;

3) К счетчику комнат на текущем этаже, то есть переменной room_count, прибавляется одна единица;

4) Выводится номер этажа, номер текущей комнаты и общее количество пройденных комнат (Не считая текущую);

5) К общему счетчику комнат, то есть переменной total_rooms, прибавляется одна единица;

6) Вызывается функция next_floor, которая отвечает за переход на следующий этаж и поднятие сложности;

7) Выводится сообщение о том, что пользователь попал в Комнату отдыха;

8) На разных сроках выводятся варианты действий с соответствующей нумерацией: Покинуть комнату отдыха, Прокачка, Инвентарь;

9) Пользователь вводит данные, которые сохраняются в переменную rest_room_choice;

10) Далее осуществляется проверка введенных данных rest_room_choice, представленная ранее в пункте 2.2;

11) В зависимости от выбранного пункта вызывается соответствующая функция или функции: directions для «Покинуть комнату отдыха», Upgrades и потом directions для «Прокачка», Inventory и потом directions для «Открыть инвентарь».

Таким образом осуществляется Комната отдыха.

2.7.3 Боевая комната

Была создана функция, которая отвечает за вывод меню Боевой комнату.

Код представлен на рисунке 22.

```
def battle_room():
    print('='*100)

    # room counter
    global floor
    global room_count
    global total_rooms
    room_count+=1
    print(f'ЭТАЖ {floor}')
    print(f'Комната {room_count}')
    print(f'Всего пройдено {total_rooms} комнат')
    total_rooms += 1
    next_floor()

    # global values
    global stats
    global hp
    defence = stats['DEF']
    agility = stats['Agility']
    atk = stats['ATK']
    global inventory
    global exp

    print('='*100)

    # battle room
    print('Вы попали в боевую комнату!')
    global opponents
    opponent = random.choice(opponents)
    print(f'Ваш противник - {opponent}!')
```

Рисунок 22 – Боевая комната

Пояснение работы программы:

1) Создается функция battle_room, которая отвечает за Боевую комнату;

2) Переменные floor, room_count и total_rooms объявляются глобальными;

3) К счетчику комнат на текущем этаже, то есть переменной room_count, прибавляется одна единица;

4) Выводится номер этажа, номер текущей комнаты и общее количество пройденных комнат (Не считая текущую);

5) К общему счетчику комнат, то есть переменной total_rooms, прибавляется одна единица;

6) Вызывается функция next_floor, которая отвечает за переход на следующий этаж и поднятие сложности;

7) Значения stats и hp объявляются глобальными;

8) Создается переменная defence, которой присваивается значение из словаря stats по ключу «DEF»;

9) Создается переменная agility, которой присваивается значение из словаря stats по ключу «Agility»;

10) Создается переменная atk, которой присваивается значение из словаря stats по ключу «ATK»;

11) Значения inventory и exp объявляются глобальными;

12) Выводится сообщение о том, что пользователь попал в Боевую комнату;

13) Значение opponents объявляется глобальным;

14) Создается переменная opponent, которой присваивается случайное значение из списка opponents;

15) Генерация случайного оппонента в переменной opponent из списка opponents осуществляется через функицию choice из модуля random;

16) Название случайного оппонента выводится в консоль.

Таким образом осуществляется меню Боевой комнаты.

Далее начинается битва с противниками. Первым будет представлен Орк против Фембойчика инкуба.

Код меню начала битвы с Орком представлен на рисунке 23.

```
# orc
if opponent == opponents[0]:
    global orc_stats
    orc_hp = orc_stats['MaxHP']
    orc_def = orc_stats['DEF']
    orc_agility = orc_stats['Agility']
    orc_atk = orc_stats['ATK']

    # orc / femboy incubus
    if your_class == 'Фембойчик инкуб':
        while orc_hp > 0 and hp > 0:
            print('='*100)

            print(f'HP Орка - {orc_hp}')
            print(f'DEF Орка - {orc_def}')

            print('='*100)

            print(f'HP Фембочика инкуба - {hp}')
            print(f'DEF Фембойчика инкуба - {defence}')

            print('='*100)
            print('1. Атаковать')
            print('2. Соблазнить')
            print('3. Открыть инвентарь')
            battle_choice = input('Что будете делать? ')

            # check if the entered data is valid
            is_int(battle_choice)
            if (int(battle_choice) > 3) or (int(
   battle_choice) < 1):
                print(f'ОТ 1 ДО 3!!!!!')
                sys.exit()
```

Рисунок 23 – Меню битвы с Орком

Пояснение работы программы:

1) Если случайным образом выпадает Орк, по индексу 0 в списке opponents, то начинается битва с Орком;

2) Значение orc_stats объявляется глобальной;

3) Создается переменная orc_hp, который присваивается значение из словаря orc_stats по ключу «MaxHP»;

4) Создается переменная orc_def, который присваивается значение из словаря orc_stats по ключу «DEF»;

5) Создается переменная orc_agility, который присваивается значение из словаря orc_stats по ключу «Agility»;

6) Создается переменная orc_atk, который присваивается значение из словаря orc_stats по ключу «ATK»;

7) Если выбранный пользователем класс, your_class, соответствует «Фембойчик инкуб», то начинается битва Орка против Фембойчика инкуба;

8) С помощью цикла while мы делаем так, чтобы битва длилась пока здоровье Орка, orc_hp, было больше нуля и здоровье Фембойчика инкуба, hp, было больше нуля;

9) На разных строках выводится здоровье Орка, защита Орка, здоровье Фембойчика инкуба и защита Фембойчика инкуба;

10) На разных строках выводятся варианты действий с соответствующей нумерацией: «Атаковать», «Соблазнить» и «Открыть инвентарь»;

11) Осуществляется ввод данных пользователем, который сохраняется в переменную battle_choice;

12) Далее осуществляется проверка введенных данных battle_choice, представленная ранее в пункте 2.2.

Таким образом выглядит начало битвы Орка и Фембойчика инкуба.

Выбор пункта 1 (Атаковать) представлен на рисунке 24.

```
# attack
if battle_choice == '1':
        print('='*100)

        print('Фембойчик инкуб атакует Орка!')

        # orc evasion success
        orc_evasion_success = 0
        if orc_agility < random.randint(1,10):
            orc_evasion_success = 0
        else:
            orc_evasion_success = 1

        # orc doesn't evade
        if orc_evasion_success == 0:
            print('Фембойчик инкуб наносит удар Орку!')
            print(f'Нанесенный урон - {atk}!')
            if (orc_def > 0) and (orc_def - atk >= 0):
                orc_def -= atk
            elif (orc_def > 0) and (orc_def - atk < 0):
                orc_def = 0
            elif orc_def == 0:
                orc_hp -= atk

            print('='*100)


            print('Орк атакует Фембойчика инкуба!')

            # femboy incubus evasion success
            evasion_success = 0
            if agility < random.randint(1,10):
                evasion_success = 0
            else:
                evasion_success = 1

            # femboy incubus doesn't evade
            if evasion_success == 0:
                print('='*100)

                print('Орк наносит удар Фембойчику инкубу!')
                print(f'Нанесенный урон - {orc_atk}!')
                if (defence > 0) and (defence - orc_atk >= 0):
                    defence -= orc_atk
                elif (defence > 0) and (defence - orc_atk < 0):
                    defence = 0
                elif defence == 0:
                    hp -= orc_atk

            # femboy incubus evades
            elif evasion_success == 1:
                print('='*100)

                print('Фембойчик инкуб уклонился от удара Орка!')

        # orc evades
        elif orc_evasion_success == 1:
            print('Орк уклонился от удара Фембойчика инкуба и атакует его!')

            # femboy incubus evasion success
            evasion_success = 0
            if agility < random.randint(1,10):
                evasion_success = 0
            else:
                evasion_success = 1

            # femboy incubus doesn't evade
            if evasion_success == 0:
                print('='*100)

                print('Орк наносит удар Фембойчику инкубу!')
                print(f'Нанесенный урон - {orc_atk}!')
                if (defence > 0) and (defence - orc_atk >= 0):
                    defence -= orc_atk
                elif (defence > 0) and (defence - orc_atk < 0):
                    defence = 0
                elif defence == 0:
                    hp -= orc_atk

            # femboy incubus evades
            elif evasion_success == 1:
                print('='*100)

                print('Фембойчик инкуб уклонился от удара Орка!')
```

Рисунок 24 – Атаковать Орка

Пояснение работы программы:

1) Если пользователь выбирает пункт 1, что соответствует «Атаковать», то выводится сообщение о том, что Фембойчик инкуб атакует Орка;

2) Создается переменная orc_evasion_success, равная нулю, в которой будет содержаться значение успеха уворота Орка;

3) Используется условный оператор if, если orc_agility, то есть ловкость Орка, меньше случайного значения от 1 до 10, то переменная orc_evasion_success приравнивается к нулю;

4) Случайное значение от 1 до 10 генерируется с помощью функции randint из модуля random;

5) В противном случае, переменная orc_evasion_success равна единице;

6) При помощи условно оператора if проверяется если orc_evasion_success равна нулю, то есть Орк не уворачивается;

7) В таком случае выводится сообщение о том, что Фембойчик инкуб наносит удар Орку;

8) Выводится сообщение о значении нанесенного урона;

9) Используется условный оператор if, если защита Орка, ork_def, больше нуля и защита Орка, orc_def, минус атака Фембойчика инкуба, atk, больше или равно нулю, то от защиты орка, orc_def, отнимается значение атаки Фембойчика инкуба, atk;

10) Если предыдущее условие не выполняется, то применяется elif, если защита Орка, ork_def, больше нуля и защита Орка, orc_def, минус атака Фембойчика инкуба, atk, меньше нуля, то защита Орка, orc_def, приравнивается к нулю;

11) Выводится сообщение о том, что Орк атакует Фембойчика инкуба;

12) Создается переменная evasion_success, равная нулю, в которой будет содержаться значение успеха уворота Фембойчика инкуба;

13) Используется условный оператор if, если agility, то есть ловкость Фембойчика инкуба, меньше случайного значения от 1 до 10, то переменная evasion_success приравнивается к нулю;

14) Случайное значение от 1 до 10 генерируется с помощью функции randint из модуля random;

15) В противном случае, переменная evasion_success равна единице;

16) При помощи условно оператора if проверяется если evasion_success равна нулю, то есть Фембойчик инкуб не уворачивается;

17) В таком случае выводится сообщение о том, что Орк наносит удар Фембойчику инкубу;

18) Выводится сообщение о значении нанесенного урона;

19) Используется условный оператор if, если защита Фембойчика инкуба, defence, больше нуля и его защита, defence, минус атака Орка, orc_atk, больше или равно нулю, то от защиты Фембойчика инкуба, defence, отнимается значение атаки Орка, orc_atk;

20) Если предыдущее условие не выполняется, то применяется elif, если защита Фембойчика инкуба, defence, больше нуля и его защита, defence, минус атака Орка, orc_atk, меньше нуля, то защита Фембочика инкуба, defence, приравнивается к нулю;

21) Если предыдущие условия не выполняются, то применяется elif, если защита Фембойчика инкуба, defence, равна нулю, то от здоровья Фембойчика инкуба, hp, отнимается значение атаки Орка, orc_atk;

22) В случае если Фембойчик инкуб уворачивается от удара Орка, то есть переменная evasion_success равна единице, то выводится соответствующее сообщение;

23) Если же Орк уклонился от предыдущей атаки Фембойчика инкуба, то есть переменная orc_evasion_success равна единице, то выводится соответствующее сообщение и Орк атакует Фембойчика инкуба;

24) Создается переменная evasion_success, равная нулю, в которой будет содержаться значение успеха уворота Фембойчика инкуба;

25) Используется условный оператор if, если agility, то есть ловкость Фембойчика инкуба, меньше случайного значения от 1 до 10, то переменная evasion_success приравнивается к нулю;

26) Случайное значение от 1 до 10 генерируется с помощью функции randint из модуля random;

27) В противном случае, переменная evasion_success равна единице;

28) При помощи условно оператора if проверяется если evasion_success равна нулю, то есть Фембойчик инкуб не уворачивается;

29) В таком случае выводится сообщение о том, что Орк наносит удар Фембойчику инкубу;

30) Выводится сообщение о значении нанесенного урона;

31) Используется условный оператор if, если защита Фембойчика инкуба, defence, больше нуля и его защита, defence, минус атака Орка, orc_atk, больше или равно нулю, то от защиты Фембойчика инкуба, defence, отнимается значение атаки Орка, orc_atk;

32) Если предыдущее условие не выполняется, то применяется elif, если защита Фембойчика инкуба, defence, больше нуля и его защита, defence, минус атака Орка, orc_atk, меньше нуля, то защита Фембочика инкуба, defence, приравнивается к нулю;

33) Если предыдущие условия не выполняются, то применяется elif, если защита Фембойчика инкуба, defence, равна нулю, то от здоровья Фембойчика инкуба, hp, отнимается значение атаки Орка, orc_atk;

34) В случае если Фембойчик инкуб уворачивается от удара Орка, то есть переменная evasion_success равна единице, то выводится соответствующее сообщение.

Таким образом выглядит выбор пункта 1 (Атаковать) на Орке.

Выборы пункта 2 (Соблазнить) и пункта 3 (Открыть инвентарь) представлены на рисунке 25.

```
# seduce
if battle_choice == '2':
    print('='*100)

    print('Фембойчик инкуб успешно соблазняет Орка и тем самым покидает поле битвы!')
    print('Фембойчик инкуб получает +1 к опыту и 15 золотых монет за победу над Орком!')
    inventory[0] += 15
    print(f'Текущий инвентарь: {inventory}')
    exp += 1
    exp_reset()
    print(f'Текущий уровень: {lvl}')
    print(f'Текущий опыт: {exp}/5')
    directions()

# open inventory
if battle_choice == '3':
    Inventory()
```

Рисунок 25 – Соблазнить Орка/Открыть инвентарь

Пояснение работы программы:

1) Если пользователь выбирает пункт 2, что соответствует «Соблазнить», то выводится сообщение о том, что Фембойчик инкуб успешно соблазняет Орка, так как эта способность абсолютно всегда успешно срабатывает на Орке;

2) Выводится сообщение о том, что за победу над Орком Фембойчик инкуб получает одну единицу опыта и пятнадцать золотых монет;

3) В первый слот инвентаря, inventory, который хранит золотые монеты, прибавляется пятнадцать золотых монет;

4) Выводится текущий инвентарь;

5) Добавляется одна единица опыта, то есть к переменной exp прибавляется единица;

6) Вызывается функция exp_reset, которая отвечает за сброс опыта, повышение уровня и получание очков прокачки;

7) На разных строчках выводятся текущие уровень и опыт;

8) Вызывается функция direction, то есть далее следует Развилка;

9) Если пользователь выбирает пункт 3, что соответствует «Открыть инвентарь», то вызывается функция Inventory, которая отвечает за инвентарь.

Таким образом осуществляются действия «Соблазнить» и «Открыть инвентарь».

Победа или поражение, то есть когда либо здоровье Орка, либо здоровье Фембойчика инкуба соответственно заканчиваются представлены на рисунке 26.

```
else:
    # orc defeated
    if orc_hp <= 0:
        print('='*100)
        print('Фембойчик инкуб побеждает Орка!')
        print('Фембойчик инкуб получает +1 к опыту и 15
   золотых монет за победу над Орком!')
        inventory[0] += 15
        print(f'Текущий инвентарь: {inventory}')
        exp += 1
        exp_reset()
        print(f'Текущий уровень: {lvl}')
        print(f'Текущий опыт: {exp}/5')
        directions()

    # femboy incubus defeated
    elif hp <= 0:
        print('='*100)
        print('Орк побеждает Фембойчика инкуба и утаскивает к
   себе в пещеру!')
        print('Игра завершена!')
        sys.exit()
```

Рисунок 26 – Победа/Поражение в битве с Орком

Пояснение работы программы:

1) Данный else является частью после цикла while, то есть при этом условии ни либо здоровье Фембойчика инкуба, hp, либо здоровье Орка, orc_hp, меньше или равно нулю;

2) Если здоровье Орка, orc_hp, меньше или равно нулю, то Фембойчик инкуб побеждает Орка;

3) Выводится соответсвующее сообщение о победе;

4) Выводится сообщение о том, что за победу над Орком Фембойчик инкуб получает одну единицу опыта и пятнадцать золотых монет;

5) В первый слот инвентаря, inventory, который хранит золотые монеты, прибавляется пятнадцать золотых монет;

6) Выводится текущий инвентарь;

7) Добавляется одна единица опыта, то есть к переменной exp прибавляется единица;

8) Вызывается функция exp_reset, которая отвечает за сброс опыта, повышение уровня и получание очков прокачки;

9) На разных строчках выводятся текущие уровень и опыт;

10) Вызывается функция direction, то есть далее следует Развилка;

11) Если же здоровье Фембойчика инкуба, hp, меньше или равно нулю, то Фембойчик инкуб терпит поражение;

12) Выводится сообщение о том, что Орк побеждает Фембойчика инкуба и утаскивает к себе в пещеру;

13) Выводится сообщение о том, что игра завершена;

14) Осуществляется завершение программы с помощью функции exit из модуля sys.

Таким образом происходит победа или поражение в битве с Орком.

Для остальных классов битва с Орком аналогична, за исключением особых способностей. Как было сказано ранее, особая способность Кошкодевочки работает на Орка абсолютно всегда, как и способность Фембойчика инкуба, однако особые способности Эльфийки и Альфы оборотня работают на Орка с вероятностью в пятьдесят процентов, некоторые особые способности так же никогда не работают на некоторых врагов.

Код для работы способностей с вероятностью в пятьдесят процентов, как и пару других уникальных вещей, будет рассмотрен в битве с Ведьмой.

Код меню начала битвы с Ведьмой представлен на рисунке 27. Выбранным классом все так же является Фембойчик инкуб.

```
# witch
if opponent == opponents[1]:
    global witch_stats
    witch_hp = witch_stats['MaxHP']
    witch_def = witch_stats['DEF']
    witch_agility = witch_stats['Agility']
    witch_atk = witch_stats['ATK']
    defence = stats['DEF']
    agility = stats['Agility']
    atk = stats['ATK']

    # witch / femboy incubus
    if your_class == 'Фембойчик инкуб':
        while witch_hp > 0 and hp > 0:
            print('='*100)

            print(f'HP Ведьмы - {witch_hp}')
            print(f'DEF Ведьмы - {witch_def}')

            print('='*100)

            print(f'HP Фембочика инкуба - {hp}')
            print(f'DEF Фембойчика инкуба - {defence}')

            print('='*100)
            print('1. Атаковать')
            print('2. Соблазнить')
            print('3. Открыть инвентарь')
            print('4. Подкупить (100 золотых монет)')
            print('5. Подкупить Зелье( молодости)')
            battle_choice = input('Что будете делать? ')

            # check if the entered data is valid
            is_int(battle_choice)
            if (int(battle_choice) > 5) or (int(
   battle_choice) < 1):
                print(f'ОТ 1 ДО 5!!!!!')
                sys.exit()
```

Рисунок 27 – Меню битвы с Ведьмой

Пояснение работы программы:

1) Если случайным образом выпадает Ведьма, по индексу 1 в списке opponents, то начинается битва с Ведьмой;

2) Значение witch_stats объявляется глобальной;

3) Создается переменная witch_hp, который присваивается значение из словаря witch_stats по ключу «MaxHP»;

4) Создается переменная witch_def, который присваивается значение из словаря witch_stats по ключу «DEF»;

5) Создается переменная witch_agility, который присваивается значение из словаря witch_stats по ключу «Agility»;

6) Создается переменная witch_atk, который присваивается значение из словаря witch_stats по ключу «ATK»;

7) Если выбранный пользователем класс, your_class, соответствует «Фембойчик инкуб», то начинается битва Ведьмы против Фембойчика инкуба;

8) С помощью цикла while мы делаем так, чтобы битва длилась пока здоровье Ведьмы, witch_hp, было больше нуля и здоровье Фембойчика инкуба, hp, было больше нуля;

9) На разных строках выводится здоровье Ведьмы, защита Ведьмы, здоровье Фембойчика инкуба и защита Фембойчика инкуба;

10) На разных строках выводятся варианты действий с соответствующей нумерацией: «Атаковать», «Соблазнить», «Открыть инвентарь», «Подкупить (100 золотых монет)» и «Подкупить (Зелье молодости)»;

11) Осуществляется ввод данных пользователем, который сохраняется в переменную battle_choice;

12) Далее осуществляется проверка введенных данных battle_choice, представленная ранее в пункте 2.2.

Так как атака Ведьмы аналогичка атаке Орка, рассматривать мы ее не будем, а сразу перейдем к особой способности Фембойчика инкуба, которая работает на Ведьме с вероятностью в пятьдесят процентов.

Код соблазнения Ведьмы представлен на рисунке 28.

```
# seduce
if battle_choice == '2':
    print('='*100)

    # seduction success
    seduction_success = random.randint(0,1)

    # failed seduction
    if seduction_success == 0:
        print('Фембойчику инкубу не удалось соблазнить Ведьму!')
        print('Ведьма кидает в Фембойчика инкуба картами таро!')

        # femboy incubus evasion success
        evasion_success = 0
        if agility < random.randint(1,10):
            evasion_success = 0
        else:
            evasion_success = 1

        # femboy incubus doesn't evade
        if evasion_success == 0:
            print('Карты таро Ведьмы попадают в Фембойчика инкуба!')
            print(f'Нанесенный урон - {witch_atk}!')
            if (defence > 0) and (defence - witch_atk >= 0):
                defence -= witch_atk
            elif (defence > 0) and (defence - witch_atk < 0):
                defence = 0
            elif defence == 0:
                hp -= witch_atk

        # femboy incubus evades
        elif evasion_success == 1:
            print('Фембойчик инкуб уклонился от карт таро Ведьмы!')

    # succeeded seduction
    if seduction_success == 1:
        print('Фембойчик инкуб успешно соблазняет Ведьму и тем самым покидает поле битвы!')
        print('Фембойчик инкуб получает +1 к опыту и 20 золотых монет за победу над Ведьмой!')
        inventory[0] += 20
        print(f'Текущий инвентарь: {inventory}')
        exp += 1
        exp_reset()
        print(f'Текущий уровень: {lvl}')
        print(f'Текущий опыт: {exp}/5')
        directions()
```

Рисунок 28 – Соблазнить Ведьму

Пояснение работы программы:

1) Если пользователь выбрал пункт 2, что соответствует «Соблазнить» то создается переменная seduction_success, которой присваивается случайное значение от 0 до 1, она содержит значение успеха соблазнения;

2) Случайное значение переменной seduction_success генерируется с помощью функции randint модуля random;

3) Если Фембойчику инкубу не удалось соблазнить Ведьму, то есть переменная seduction_success равна нулю, то выводится соответствующее сообщение и Ведьма атакует Фембойчика инкуба;

4) Создается переменная evasion_success, равная нулю, в которой будет содержаться значение успеха уворота Фембойчика инкуба;

5) Используется условный оператор if, если agility, то есть ловкость Фембойчика инкуба, меньше случайного значения от 1 до 10, то переменная evasion_success приравнивается к нулю;

6) Случайное значение от 1 до 10 генерируется с помощью функции randint из модуля random;

7) В противном случае, переменная evasion_success равна единице;

8) При помощи условно оператора if проверяется если evasion_success равна нулю, то есть Фембойчик инкуб не уворачивается;

9) В таком случае выводится сообщение о том, что Ведьма наносит удар Фембойчику инкубу;

10) Выводится сообщение о значении нанесенного урона;

11) Используется условный оператор if, если защита Фембойчика инкуба, defence, больше нуля и его защита, defence, минус атака Ведьмы, witch_atk, больше или равно нулю, то от защиты Фембойчика инкуба, defence, отнимается значение атаки Ведьмы, witch_atk;

12) Если предыдущее условие не выполняется, то применяется elif, если защита Фембойчика инкуба, defence, больше нуля и его защита, defence, минус атака Ведьмы, witch_atk, меньше нуля, то защита Фембочика инкуба, defence, приравнивается к нулю;

13) Если предыдущие условия не выполняются, то применяется elif, если защита Фембойчика инкуба, defence, равна нулю, то от здоровья Фембойчика инкуба, hp, отнимается значение атаки Ведьмы, witch_atk;

14) В случае если Фембойчик инкуб уворачивается от удара Ведьмы, то есть переменная evasion_success равна единице, то выводится соответвующее сообщение;

15) Если же Фембойчику инкубу удалось соблазнить Ведьму, то есть переменная seduction_success равна единице, то выводится соответствующее сообщение и битва завершается;

16) Выводится сообщение об успешном соблазнении Ведьмы;

17) Выводится сообщение о том, что Фембойчик инкуб получает одно очко опыта и двадцать золотых монет за победу над Ведьмой;

18) В первый слот инвентаря, inventory, который хранит золотые монеты, прибавляется двадцать золотых монет;

19) Выводится текущий инвентарь;

20) Добавляется одна единица опыта, то есть к переменной exp прибавляется единица;

21) Вызывается функция exp_reset, которая отвечает за сброс опыта, повышение уровня и получание очков прокачки;

22) На разных строчках выводятся текущие уровень и опыт;

23) Вызывается функция direction, то есть далее следует Развилка.

Помимо этого, при битве с Ведьмой также появляются две новые опции: «Подкупить (100 золотых монет)» и «Подкупить (Зелье молодости)». Они рассмотрены на рисунке 29. Выбранным классом все еще является Фембойчик инкуб.

```
# bribe with money
if battle_choice == '4':
    print('='*100)

    # insufficient funds
    if inventory[0] < 100:
        print('Недостаточно средств!')

    # sufficient funds
    else:
        print('Вы успешно подкупаете Ведьму и покидаете поле битвы!')
        print('Фембойчик инкуб получает +1 к опыту за подкуп Ведьмы!')
        exp += 1
        exp_reset()
        print(f'Текущий уровень: {lvl}')
        print(f'Текущий опыт: {exp}/5')
        directions()

# bribe with a potion of youth
if battle_choice == '5':
    print('='*100)

    # no potion of youth
    if {'Зелье молодости'} not in inventory:
        print('У вас нет Зелья молодости и нечем подкупить Ведьму!')

    # potion of youth
    else:
        print('Вы успешно подкупаете Ведьму и покидаете поле битвы!')
        print('Фембойчик инкуб получает +1 к опыту за подкуп Ведьмы!')
        exp += 1
        exp_reset()
        print(f'Текущий уровень: {lvl}')
        print(f'Текущий опыт: {exp}/5')
        directions()
```

Рисунок 29 – Подкупить Ведьму

Пояснение работы программы:

1) Если пользователь выбрал пункт 4, что соответствует «Подкупить (100 золотых монет)», то происходит проверка необходимых средств;

2) С помощью условно оператора if определяется, если у пользователя нет 100 золотых монет, и в таком случае выводится сообщение о недостатке средств;

3) В противном случае, выводится сообщение о том, что Фембойчик инкуб успешно подкупает Ведьму;

4) Выводится сообщение о том, что Фембойчик инкуб получает одну единицу опыта за подкуп Ведьмы;

5) Добавляется одна единица опыта, то есть к переменной exp прибавляется единица;

6) Вызывается функция exp_reset, которая отвечает за сброс опыта, повышение уровня и получание очков прокачки;

7) На разных строчках выводятся текущие уровень и опыт;

8) Вызывается функция direction, то есть далее следует Развилка.

9) Если же пользователь выбрал пункт 5, что соответствует «Подкупить (Зелье молодости)», то происходит проверка необходимого предмета;

10) С помощью условно оператора if определяется, если у пользователя в инвентаре нет Зелья молодости, и в таком случае выводится сообщение о его отсутствии;

11) В противном случае, выводится сообщение о том, что Фембойчик инкуб успешно подкупает Ведьму;

12) Выводится сообщение о том, что Фембойчик инкуб получает одну единицу опыта за подкуп Ведьмы;

13) Добавляется одна единица опыта, то есть к переменной exp прибавляется единица;

14) Вызывается функция exp_reset, которая отвечает за сброс опыта, повышение уровня и получание очков прокачки;

15) На разных строчках выводятся текущие уровень и опыт;

16) Вызывается функция direction, то есть далее следует Развилка.

Таким образом происходит подкуп Ведьмы.

Возвращаясь к особым функциям, собая функция также может и не срабатывать вообще. Пример этого можно рассмотреть при битве между Фембойчиком инкубом и Бобром.

Код соблазнения Бобра представлен на рисунке 30.

```
# seduce
if battle_choice == '2':
    print('='*100)

    print('Фембойчику инкубу не удалось соблазнить Бобра!')
    print('Бобер бьет Фембойчка инкуба хвостом!')

    # femboy incubus evasion success
    evasion_success = 0
    if agility < random.randint(1,10):
        evasion_success = 0
    else:
        evasion_success = 1

    # femboy incubus doesn't evade
    if evasion_success == 0:
        print('Бобер наносит Фембойчику инкубу удар хвостом!')
        print(f'Нанесенный урон - {beaver_atk}!')
        if (defence > 0) and (defence - beaver_atk >= 0):
            defence -= beaver_atk
        elif (defence > 0) and (defence - beaver_atk < 0):
            defence = 0
        elif defence == 0:
            hp -= beaver_atk

    # femboy incubus evades
    elif evasion_success == 1:
        print('Фембойчик инкуб уклонился от хвоста Бобра!')
```

Рисунок 30 – Соблазнить Бобра

Пояснение работы программы:

1) Если пользователь выбрал пункт 4 в битве с Бобром, что соответствует «Соблазнить», то выводится сообщение о неудачной попытке соблазнения, так как особая способность Фембойчика инкуба ни при каких условиях не работает на Бобра;

2) Выводится сообщение о том, что Бобер атакует Фембойчика инкуба;

3) Создается переменная evasion_success, равная нулю, в которой будет содержаться значение успеха уворота Фембойчика инкуба;

4) Используется условный оператор if, если agility, то есть ловкость Фембойчика инкуба, меньше случайного значения от 1 до 10, то переменная evasion_success приравнивается к нулю;

5) Случайное значение от 1 до 10 генерируется с помощью функции randint из модуля random;

6) В противном случае, переменная evasion_success равна единице;

7) При помощи условно оператора if проверяется если evasion_success равна нулю (То есть Фембойчик инкуб не уворачивается);

8) В таком случае выводится сообщение о том, что Бобр наносит удар Фембойчику инкубу;

9) Выводится сообщение о значении нанесенного урона;

10) Используется условный оператор if, если защита Фембойчика инкуба, defence, больше нуля и его защита, defence, минус атака Бобра, beaver_atk, больше или равно нулю, то от защиты Фембойчика инкуба, defence, отнимается значение атаки Бобра, beaver_atk;

11) Если предыдущее условие не выполняется, то применяется elif, если защита Фембойчика инкуба, defence, больше нуля и его защита, defence, минус атака Бобра, beaver_atk, меньше нуля, то защита Фембочика инкуба, defence, приравнивается к нулю;

12) Если предыущие условия не выполняются, то применяется elif, если защита Фембойчика инкуба, defence, равна нулю, то от здоровья Фембойчика инкуба, hp, отнимается значение атаки Бобра, beaver_atk;

13) В случае если Фембойчик инкуб уворачивается от удара Бобра, то есть переменная evasion_success равна единице, то выводится соответвующее сообщение;

Таким образом работает особая способность «Соблазнить» без шанса успеха.

Абсолютно все особые способности персонажей работают аналогичным образом.

2.7.4 Комната с сундуком

Для Комнаты с сундуком была создана функция, которая представлена на рисунке 31.

```
def reward_room():
    print('='*100)

    # room counter
    global floor
    global room_count
    global total_rooms
    room_count+=1
    print(f'ЭТАЖ {floor}')
    print(f'Комната {room_count}')
    print(f'Всего пройдено {total_rooms} комнат')
    total_rooms += 1
    next_floor()

    print('='*100)

    print('Вы попали в Комнату с сундуком!')

    print('='*100)

    print('1. Уйти')
    print('2. Заглянуть в сундук')
    chest_choice = input('Что будете делать? ')

    # check if the entered data is valid
    is_int(chest_choice)
    if (int(chest_choice) > 2) or (int(chest_choice) < 1):
        print(f'ОТ 1 ДО 2!!!!!')
        sys.exit()

    # leave
    if chest_choice == '1':
        directions()
```

Рисунок 31 – Меню комнаты с сундуком

Пояснение работы программы:

1) Создается функция reward_room, которая отвечает за Комнату с сундуком;

2) Переменные floor, room_count и total_rooms объявляются глобальными;

3) К счетчику комнат на текущем этаже, то есть переменной room_count, прибавляется одна единица;

4) Выводится номер этажа, номер текущей комнаты и общее количество пройденных комнат (Не считая текущую);

5) К общему счетчику комнат, то есть переменной total_rooms, прибавляется одна единица;

6) Вызывается функция next_floor, которая отвечает за переход на следующий этаж и поднятие сложности;

7) Выводится сообщение о том, что пользователь попал в Комнату с сундуком;

8) Выводятся варианты «Уйти» и «Заглянуть в сундук» с соответствующей нумерацией;

9) Пользователь вводит номер выбранного варианта и это значение сохраняется в переменную direction_choice;

10) Далее осуществляется проверка введенных данных direction_choice, представленная ранее в пункте 2.2;

11) Если пользователь выбрал пункт 1, то вызывается функция directions, которая отправляет его к следующей развилке.

Код выбора пункта 2, то есть «Заглянуть в сундук», представлен на рисунке 32.

```
# take a look
if chest_choice == '2':
    print('='*100)

    available_items = [5, 10, 25, 30, 50, 'Зелье здоровья', '
   Зелье молодости', 'Пыль']
    chest = random.choice(available_items)
    print('Вы заглядывате в сундук, а там...')
    if chest == 'Пыль':
        print('Пыль')
        print('Кажется, вы не первый, кто сюда пришел')
        print('Увы, вы уходите без добычи')
        directions()
```

Рисунок 32 – Заглянуть в сундук

Пояснение работы программы:

1) Если пользователь выбирает пункт 2, то есть «Заглянуть в сундук», то создается список available_items, которая хранит все возможные предметы, которые можно найти в сундуке: 5 золотых монет, 10 золотых монет, 25 золотых монет, 30 золотых монет, 50 золотых монет, Зелье здоровья, Зелье молодости, Пыль;

2) Создается переменная chest, которой присваивается случайное значение из списка available_items;

3) Случайное значение переменной chest генерируется с помощью функции choice из модуля random;

4) Если в сундуке находится Пыль, то есть значение переменной chest соответствует «Пыль», то выводится соответствующее сообщение;

5) Вызывается функция directions, которая отправляет пользователя к следующей Развилке.

Еще одним, который можно найти в сундуке являются золотые монеты в разных количествах.

Код данной программы представлен на рисунке 33.

```
# gold coins
if chest == 5 or chest == 10 or chest == 25 or chest == 30 or chest
     == 50:
    print(f'{chest} золотых монет!')

    print('='*100)

    print('1. Нет')
    print('2. Да')
    take_choice = input('Забрать? ')

    # check if the entered data is valid
    is_int(take_choice)
    if (int(take_choice) > 2) or (int(take_choice) < 1):
        print(f'ОТ 1 ДО 2!!!!!')
        sys.exit()

    # no not take
    if take_choice == '1':
        print('='*100)

        print(f'Вы решаете не поддаваться капитализму и оставляете {
    chest} золотых монет там, где нашли!')
        print(f'Ваш текущий инвентарь: {inventory}')
        directions()

    # yes take
    if take_choice == '2':
        print('='*100)

        print(f'Вы забираете {chest} золотых монет!')
        inventory[0] += chest
        print(f'Ваш текущий инвентарь: {inventory}')
        directions()
```

Рисунок 33 – Золотые монеты в сундуке

Пояснение работы программы:

1) Если в сундуке находятся золотые монеты, то есть значение переменной chest соответствует пяти, десяти, двадцати пяти, тридцати или пятидесяти, то выводится соответствующее сообщение;

2) На разных строках выводятся опции с соответсвующей нумерацией: «Нет», «Да»;

3) Производится ввод данных пользователем, который сохраняется в переменную take_choice;

4) Далее осуществляется проверка введенных данных take_choice, представленная ранее в пункте 2.2;

5) Используется условный оператор if, если пользователь выбирает пункт 1, то есть «Нет», то выводится соответствующее сообщение, выводится текущий инвентарь и вызывается функция directions, которая отправляет пользователя на следующую Развилку;

6) Используется условный оператор if, если пользователь выбирает пункт 2, то есть «Да», то выводится соответствующее сообщение о том, какой предмет вы забрали, выводится текущий инвентарь и вызывается функция directions, которая отправляет пользователя на следующую Развилку.

Еще один предметом, который можно найти в сундуке является Зелье здоровья.

Код данной программы представлен на рисунке 34.

```
# health potion
if chest == 'Зелье здоровья':
    print(f'{chest}!')

    print('='*100)

    print('1. Нет')
    print('1. Да')
    take_choice = input('Забрать? ')

    # check if the entered data is valid
    is_int(take_choice)
    if (int(take_choice) > 2) or (int(take_choice) < 1):
        print(f'ОТ 1 ДО 2!!!!!')
        sys.exit()

    # no not take
    if take_choice == '1':
        print('='*100)

        print(f'Вы решаете, что хил для слабаков и оставляете Зелье здоровья там, где нашли!')
        print('Ваш текущий инвентарь:')
        print(inventory)
        directions()

    # take if enough space
    if take_choice == '2' and len(inventory) < 5:
        print('='*100)

        print(f'Вы забираете Зелье здоровья!')
        inventory.append({chest})
        print(f'Ваш текущий инвентарь: {inventory}')
        directions()
```

Рисунок 34 – Зелье здоровья в сундуке

Пояснение работы программы:

1) Если в сундуке находится Зелье здоровья, то есть значение переменной chest соответствует «Зелье здоровья», то выводится соответствующее сообщение;

2) На разных строках выводятся опции с соответсвующей нумерацией: «Нет», «Да»;

3) Производится ввод данных пользователем, который сохраняется в переменную take_choice;

4) Далее осуществляется проверка введенных данных take_choice, представленная ранее в пункте 2.2;

5) Используется условный оператор if, если пользователь выбирает пункт 1, то есть «Нет», то выводится соответствующее сообщение, выводится текущий инвентарь и вызывается функция directions, которая отправляет пользователя на следующую Развилку;

6) Используется условный оператор if, если пользователь выбирает пункт 2, то есть «Да», то проверяется есть ли свободные слоты в инвентаре, то есть меньше ли длина списка inventory, чем пять;

7) Если проверка проходит успешно, то есть длина списка inventory составляет менее пяти элиментов, то выводится соответствующее сообщение о том, какой предмет вы забрали, выводится текущий инвентарь и вызывается функция directions, которая отправляет пользователя на следующую Развилку.

В инвентаре может и не быть свободных слотов. Такой случай рассмотрен на рисунке 35.

```
# take if not enough space
if take_choice == '2' and len(inventory) == 5:
    print('='*100)

    print(f'Увы, вам не хватает слотов в инвентаре')
    print('Вы можете выбросить какойто- предмет из инвентаря или смириться')

    print('='*100)

    print('1. Уйти')
    print('2. Выбросить предмет')
    slots_choice = input('Что будете делать? ')

    # check if the entered data is valid
    is_int(take_choice)
    if (int(take_choice) > 2) or (int(take_choice) < 1):
        print(f'ОТ 1 ДО 2!!!!!')
        sys.exit()

    # leave
    if slots_choice == '1':
        print('='*100)

        print('Вы принимаете свою судьбу такой, какая она есть и идете дальше')
        print(f'Ваш текущий инвентарь: {inventory}')
        directions()

    #trow an item away
    if slots_choice == '2':
        print('='*100)

        for i in range(len(inventory)):
            print(f'{i+1}. {inventory[i]}')
        item_choice = input('Какой предмет хотите выбросить? ')

        # check if the entered data is valid
        is_int(item_choice)
        if (int(item_choice) > len(inventory)) or (int(item_choice) < 1):
            print(f'ОТ 1 ДО {len(inventory)}!!!!!')
            sys.exit()

        del inventory[int(item_choice)-1]
        print('Предмет был успешно выброшен!')
        print(f'Ваш текущий инвентарь: {inventory}')

        print('='*100)

        print(f'Вы забираете Зелье здоровья!')
        inventory.append({chest})
        print(f'Ваш текущий инвентарь: {inventory}')
        directions()
```

Рисунок 35 – Нет свободных слотов

Пояснение работы программы:

1) Если пользователь выбрал пункт 2, то есть «Да», но у него заняты все пять слотов инвентаря, то есть длина списка inventory равняется пяти, то выводится соответсвующее сообщение;

2) На разных строках выводятся две опции с соответсвующей нумерацией: «Уйти» и «Выбросить предмет»;

3) Производится ввод данных пользователем, который сохраняется в переменную slots_choice;

4) Далее осуществляется проверка введенных данных slots_choice, представленная ранее в пункте 2.2;

5) Используется условный оператор if, если пользователь выбирает пункт 1, то есть «Уйти», то выводится соответствующее сообщение, выводится текущий инвентарь и вызывается функция directions, которая отправляет пользователя на следующую Развилку;

6) Используется условный оператор if, если пользователь выбирает пункт 2, то есть «Выбросить предмет», то на разных строках выводятся предметы из инвентаря с соответствующей нумерацией;

7) Производится ввод данных пользователем, который сохраняется в переменную item_choice;

8) Далее осуществляется проверка введенных данных item_choice, представленная ранее в пункте 2.2;

9) Выбранный элемент инвентаря выбрасывается, то есть удаляется по индексу из списка inventory;

10) Выводится соответствующее сообщение о выброшенной предмете и выводится текущий инвентарь;

11) Выводится сообщение о том, что пользователь забрал Зелье здоровье и оно добавляется в инвентарь, то есть список inventory;

12) Выводится текущий инвентарь;

13) Вызывается функция directions, которая отправляет пользователя на следующую развилку.

Код для нахождения Зелья молодости в сундуке аналогичен.

2.8 Начало подземелья

Начало подземелье находится в самом конце кода.

Данный блок кода представлен на рисунке 36.

```
print('Вы попали в подземелье!')
print(f'ЭТАЖ {floor}')
print(f'Всего пройдено комнат: {total_rooms}')
print(f'Текущий инвентарь: {inventory}')
directions()
```

Рисунок 36 – Начало подземелья

Пояснение работы программы:

1) Выводится сообщение о том, что пользователь попал в подземелье;

2) Выводится номер этажа, то есть переменная floor;

3) Выводится количество пройденных комнат на этаже, то есть переменная room_count;

4) Выводится количество пройденных комнат в общем, то есть переменная total_rooms;

5) Выводится текущий инвентарь, то есть список inventory;

6) Вызывается функция directions, которая отправляет пользователя к первой развилке.

Таким образом начинается подземелье.

3 Тестирование

Было проведено тестирование созданной RPG-игры.

Выбор класса представлен на рисунке 37.

```
==========================================================================================
Добро пожаловать в игру «FEMBOY ALFA ORK DUNGEON»!
==========================================================================================
Выбор класса
==========================================================================================
1. Фембойчик инкуб
2. Эльфийка
3. Альфа оборотень
4. Кошкодевочка
==========================================================================================
Выберите кем вы хотите быть (введите целое число от 1 до 4): 1
```

Рисунок 37 – Выбор класса

Выбран Фембойчик инкуб. Продолжение представлено на рисунке 38.

```
==========================================================================================
Вы - Фембойчик инкуб!
==========================================================================================
Ваши характеристики:
HP - 13
ATK - 3
DEF - 1
Agility - 8
Height - 150
Weight - 40
Weapon - Отсутствует
Armor - Отсутствует
==========================================================================================
Особая способность Фембойчика инкуба - Соблазнить!
Фембойчик инкуб может соблазнить противника!
Шанс успеха - 50%!
Когда Фембойчик инкуб встречается с Орком или Бесом, шанс успеха возрастает до 100%!
Когда Фембойчик инкуб встречается с Бобром, шанс успеха снижается до 0%!
==========================================================================================
Вы попали в подземелье!
ЭТАЖ 1
Всего пройдено комнат: 0
Текущий инвентарь: [0]
==========================================================================================
Перед вами развилка
Слева - Тьма темная
Справа - Тьма темная
1. Налево
2. Направо
Куда пойдете? 1
```

Рисунок 38 – Первая Развилка

Выводятся характеристики выбранного класса и нас направляют на Развилку, где мы выбираем пойти налево. Продолжение представлено на рисунке 39.

```
==========================================================================================
Вы свернули налево
Слева - Комната с сундуком
==========================================================================================
ЭТАЖ 1
Комната 1
Всего пройдено 0 комнат
==========================================================================================
Вы попали в Комнату с сундуком!
==========================================================================================
1. Уйти
2. Заглянуть в сундук
Что будете делать? 2
```

Рисунок 39 – Первая Комната с сундуком

Мы попадаем в комнату с сундуком, где выбираем заглянуть в сундук. Продолжение представлено на рисунке 40.

Рисунок 40 – Заглядываем в сундук

Мы заглядывает в сундук и находим там пять золотых монет, которые забираем. Продолжение представлено на рисунке 41.

Рисунок 41 – Вторая Развилка

Мы попадаем на вторую Развилку, где выбираем пойти направо. Продолжение представлено на рисунке 42.

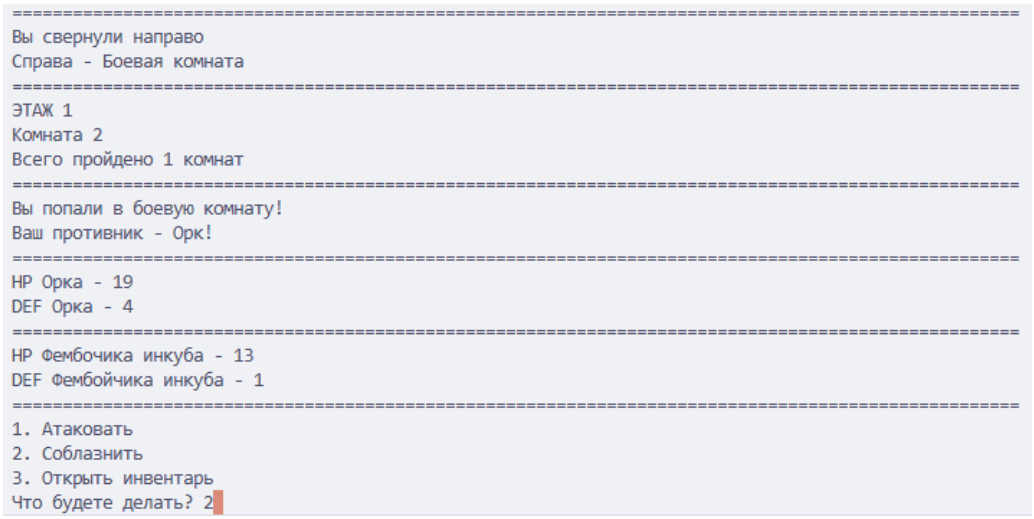

Рисунок 42 – Первая Боевая комната

Мы попадаем в Боевую комнату, где нас встречает Орк. Мы выбираем соблазнить его. Продолжение представлено на рисунке 43.

```
==========================================================================================
Фембойчик инкуб успешно соблазняет Орка и тем самым покидает поле битвы!
Фембойчик инкуб получает +1 к опыту и 15 золотых монет за победу над Орком!
Текущий инвентарь: [20]
Текущий уровень: 1
Текущий опыт: 1/5
==========================================================================================
Перед вами развилка
Слева - Боевая комната
Справа - Тьма темная
1. Налево
2. Направо
Куда пойдете? 1
```

Рисунок 43 – Третья Развилка

Фембойчик инкуб успешно соблазняет Орка и мы попадаем на третью Развилку.

Так как игра бесконечная и заканчивается только тогда, когда выбранный персонаж теряет все свое здоровье или при неверном вводе, то смысла линейно продолжать тестирование нет.

На рисунке 44 представлено меню прокачки персонажа.

```
==========================================================================================
На данный момент у вас 0 очков прокачки
1. Выйти
2. Перейти к прокачке
Что будете делать? 2
==========================================================================================
Недостаточно очков прокачки!
Текущее количество - 0
==========================================================================================
На данный момент у вас 0 очков прокачки
1. Выйти
2. Перейти к прокачке
Что будете делать?
```

Рисунок 44 – Прокачка персонажа

На рисунке 45 представлено меню Комнаты отдыха.

```
==========================================================================================
ЭТАЖ 1
Комната 4
Всего пройдено комнат: 3
==========================================================================================
Вы попали в комнату отдыха!
1. Покинуть комнату отдыха
2. Прокачка
3. Открыть инвентарь
ВНИМАНИЕ!!! Если вы открываете инвентарь или прокачку в комнате отдыха, после их закрытия автоматически следует развилка!
Что будете делать?
```

Рисунок 45 – Меню Комнаты отдыха

Таким образом созданная игра была протестирована множество раз, все ошибки исправлены.

Заключение

Текстовая RPG-игра «FEMBOY ALFA ORC DUNGEON» на Python успешно реализована и демонстрирует работоспособность всех запланированных базовых механик. Созданный каркас игры подтверждает, что даже в текстовом формате возможно создать атмосферу подземелий, ощущение прогресса и тактические решения в бою.

Реализованы:

– Четыре класса персонажей;

– Четыре класса противников;

– Система базовых характеристик персонажа (сила, ловкость, выносливость) и их влияние на геймплей;

– Особые способности персонажей;

– Пошаговую боевая система против разнотипных противников;

– Три комнаты;

– Упрощенная система инвентаря для управления снаряжением и предметами;

– Система прокачки персонажа через получение опыта и повышение уровня;

– Случайные события для повышения реиграбельности и создания непредсказуемости.

Код устойчив к ошибкам ввода, структурирован и готов к расширению новыми играбельными персонажами, противниками и предметами.

Прототип успешно выполнил свою главную задачу: он доказал, что даже в рамках текстового интерфейса можно создать увлекательное игровое ядро. Базовые циклы «исследование-бой-прокачка» работают и вызывают желание продолжать. Механики, хотя и минималистичные, демонстрируют достаточную глубину для принятия тактических решений.