TEXT NOVEL PARSER

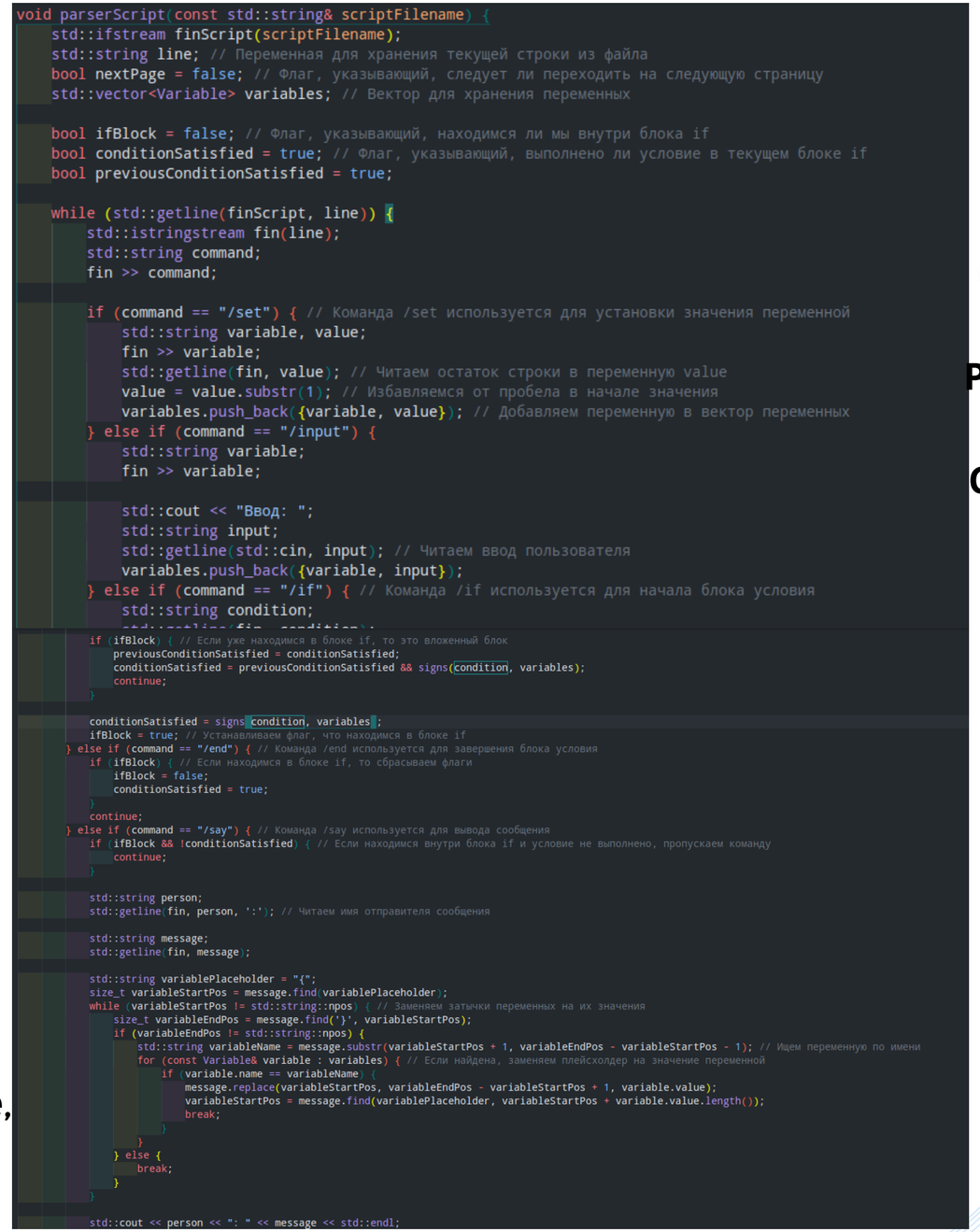
Xdd

ЭТОТ КОД ПРЕДСТАВЛЯЕТ ФУНКЦИЮ SIGNS КОТОРАЯ ПРИНИМАЕТ УСЛОВИЕ В ВИДЕ СТРОКИ РАЗБИРАЕТ ЕГО НА ОПЕРАНДЫ И ОПЕРАТОР СРАВНЕНИЯ ЗАТЕМ ИЩЕТ СОВПАДЕНИЯ СРЕДИ ПЕРЕМЕННЫХ И ВЫПОЛНЯЕТ СРАВНЕНИЕ В СООТВЕТСТВИИ С ОПЕРАТОРОМ. ФУНКЦИЯ ВОЗВРАЩАЕТ TRUE ЕСЛИ УСЛОВИЕ ВЫПОЛНЯЕТСЯ И FALSE В ПРОТИВНОМ СЛУЧАЕ.

```cpp
bool signs(const std::string& condition, const std::vector<Variable>& variables) {
    // Разбиваем условие на операнды и оператор сравнения
    std::istringstream fin(condition);
    std::string leftOperand, signOperand, rightOperand;
    fin >> leftOperand >> signOperand >> rightOperand;

    // Проходимся по списку переменных и ищем совпадение с левым операндом
    for (const Variable& variable : variables) {
        if (variable.name == leftOperand) {
            std::string leftValue = variable.value;
            // Внутренний цикл для поиска совпадения с правым операндом
            for (const Variable& variable : variables) {
                if (variable.name == rightOperand) {
                    std::string rightValue = variable.value;

                    // Проверяем оператор сравнения и возвращаем результат сравнения
                    if (signOperand == "==") {
                        return leftValue == rightValue;
                    } else if (signOperand == "!=") {
                        return leftValue != rightValue;
                    } else if (signOperand == ">") {
                        return std::stoi(leftValue) > std::stoi(rightValue);
                    } else if (signOperand == ">=") {
                        return std::stoi(leftValue) >= std::stoi(rightValue);
                    } else if (signOperand == "<") {
                        return std::stoi(leftValue) < std::stoi(rightValue);
                    } else if (signOperand == "<=") {
                        return std::stoi(leftValue) <= std::stoi(rightValue);
                    }
                }
            }
        }
    }

    // Если не найдено совпадение или неправильный оператор сравнения, возвращаем fa
    return false;
}
```

Основные элементы кода включают:

- Чтение команд из файла и выполнение соответствующих действий, таких как установка переменных, ввод значений пользователем, выполнение условных блоков и вывод сообщений.
- Обработка блоков условий, где команды между **/if** и **/end** выполняются только в случае выполнения условия.
- Замена плейсхолдеров переменных (**{variable}**) в сообщениях на значения переменных из вектора **variables**.
- Вывод сообщений в формате **person: message**.
- Обработка пользовательского ввода и установка флага **nextPage**, который указывает на необходимость перехода на следующую страницу.



```
void parserScript(const std::string& scriptFilename) {
    std::ifstream finScript(scriptFilename);
    std::string line; // Переменная для хранения текущей строки из файла
    bool nextPage = false; // Флаг, указывающий, следует ли переходить на следующую страницу
    std::vector<Variable> variables; // Вектор для хранения переменных

    bool ifBlock = false; // Флаг, указывающий, находимся ли мы внутри блока if
    bool conditionSatisfied = true; // Флаг, указывающий, выполнено ли условие в текущем блоке if
    bool previousConditionSatisfied = true;

    while (std::getline(finScript, line)) {
        std::istringstream fin(line);
        std::string command;
        fin >> command;

        if (command == "/set") { // Команда /set используется для установки значения переменной
            std::string variable, value;
            fin >> variable;
            std::getline(fin, value); // Читаем остаток строки в переменную value
            value = value.substr(1); // Избавляемся от пробела в начале значения
            variables.push_back({variable, value}); // Добавляем переменную в вектор переменных
        } else if (command == "/input") {
            std::string variable;
            fin >> variable;

            std::cout << "Ввод: ";
            std::string input;
            std::getline(std::cin, input); // Читаем ввод пользователя
            variables.push_back({variable, input});
        } else if (command == "/if") { // Команда /if используется для начала блока условия
            std::string condition;
            std::getline(fin, condition);
            if (ifBlock) { // Если уже находимся в блоке if, то это вложенный блок
                previousConditionSatisfied = conditionSatisfied;
                conditionSatisfied = previousConditionSatisfied && signs(condition, variables);
                continue;
            }

            conditionSatisfied = signs(condition, variables);
            ifBlock = true; // Устанавливаем флаг, что находимся в блоке if
        } else if (command == "/end") { // Команда /end используется для завершения блока условия
            if (ifBlock) { // Если находимся в блоке if, то сбрасываем флаги
                ifBlock = false;
                conditionSatisfied = true;
            }
            continue;
        } else if (command == "/say") { // Команда /say используется для вывода сообщения
            if (ifBlock && !conditionSatisfied) { // Если находимся внутри блока if и условие не выполнено, пропускаем команду
                continue;
            }

            std::string person;
            std::getline(fin, person, ':'); // Читаем имя отправителя сообщения

            std::string message;
            std::getline(fin, message);

            std::string variablePlaceholder = "{";
            size_t variableStartPos = message.find(variablePlaceholder);
            while (variableStartPos != std::string::npos) { // Заменяем затычки переменных на их значения
                size_t variableEndPos = message.find('}', variableStartPos);
                if (variableEndPos != std::string::npos) {
                    std::string variableName = message.substr(variableStartPos + 1, variableEndPos - variableStartPos - 1); // Ищем переменную по имени
                    for (const Variable& variable : variables) { // Если найдена, заменяем плейсхолдер на значение переменной
                        if (variable.name == variableName) {
                            message.replace(variableStartPos, variableEndPos - variableStartPos + 1, variable.value);
                            variableStartPos = message.find(variablePlaceholder, variableStartPos + variable.value.length());
                            break;
                        }
                    }
                } else {
                    break;
                }
            }

            std::cout << person << ": " << message << std::endl;
```

ЭТОТ КОД ПРЕДСТАВЛЯЕТ ФУНКЦИЮ PARSERSCRIPT, КОТОРАЯ ВЫПОЛНЯЕТ РАЗБОР СКРИПТА И ВЫПОЛНЕНИЕ КОМАНД В СООТВЕТСТВИИ С ИХ СЕМАНТИКОЙ. ОН ЧИТАЕТ ФАЙЛ SCRIPTFILENAME ПОСТРОЧНО, РАЗДЕЛЯЕТ СТРОКИ НА КОМАНДЫ И ВЫПОЛНЯЕТ СООТВЕТСТВУЮЩИЕ ДЕЙСТВИЯ В ЗАВИСИМОСТИ ОТ КОМАНДЫ.

```cpp
void parserScript(const std::string& scriptFilename) {
    std::ifstream finScript(scriptFilename);
    std::string line; // Переменная для хранения текущей строки из файла
    bool nextPage = false; // Флаг, указывающий, следует ли переходить на следующую страницу
    std::vector<Variable> variables; // Вектор для хранения переменных

    bool ifBlock = false; // Флаг, указывающий, находимся ли мы внутри блока if
    bool conditionSatisfied = true; // Флаг, указывающий, выполнено ли условие в текущем блоке if
    bool previousConditionSatisfied = true;

    while (std::getline(finScript, line)) {
        std::istringstream fin(line);
        std::string command;
        fin >> command;

        if (command == "/set") { // Команда /set используется для установки значения переменной
            std::string variable, value;
            fin >> variable;
            std::getline(fin, value); // Читаем остаток строки в переменную value
            value = value.substr(1); // Избавляемся от пробела в начале значения
            variables.push_back({variable, value}); // Добавляем переменную в вектор переменных
        } else if (command == "/input") {
            std::string variable;
            fin >> variable;

            std::cout << "Ввод: ";
            std::string input;
            std::getline(std::cin, input); // Читаем ввод пользователя
            variables.push_back({variable, input});
        } else if (command == "/if") { // Команда /if используется для начала блока условия
            std::string condition;
            std::getline(fin, condition);
            condition = condition.substr(1);
```

```
    if (ifBlock) { // Если уже находимся в блоке if, то это вложенный блок
        previousConditionSatisfied = conditionSatisfied;
        conditionSatisfied = previousConditionSatisfied && signs(condition, variables);
        continue;
    }

    conditionSatisfied = signs(condition, variables);
    ifBlock = true; // Устанавливаем флаг, что находимся в блоке if
} else if (command == "/end") { // Команда /end используется для завершения блока условия
    if (ifBlock) { // Если находимся в блоке if, то сбрасываем флаги
        ifBlock = false;
        conditionSatisfied = true;
    }
    continue;
} else if (command == "/say") { // Команда /say используется для вывода сообщения
    if (ifBlock && !conditionSatisfied) { // Если находимся внутри блока if и условие не выполнено, пропускаем команду
        continue;
    }

    std::string person;
    std::getline(fin, person, ':'); // Читаем имя отправителя сообщения

    std::string message;
    std::getline(fin, message);

    std::string variablePlaceholder = "{";
    size_t variableStartPos = message.find(variablePlaceholder);
    while (variableStartPos != std::string::npos) { // Заменяем затычки переменных на их значения
        size_t variableEndPos = message.find('}', variableStartPos);
        if (variableEndPos != std::string::npos) {
            std::string variableName = message.substr(variableStartPos + 1, variableEndPos - variableStartPos - 1); // Ищем переменную по имени
            for (const Variable& variable : variables) { // Если найдена, заменяем плейсхолдер на значение переменной
                if (variable.name == variableName) {
                    message.replace(variableStartPos, variableEndPos - variableStartPos + 1, variable.value);
                    variableStartPos = message.find(variablePlaceholder, variableStartPos + variable.value.length());
                    break;
                }
            }
        } else {
            break;
        }
    }

    std::cout << person << ": " << message << std::endl;
```

```
/say user: Привет тут мини история типо.
/say user: Введи выбор 1 или 2.
/input guess
/set chose 1
/set chosetwo 2

/if guess == chose
/say user: 1 - вы выбрали 1.
/end

/if guess == chosetwo
/say user: 2 - не круто. Давай еще раз.
/input guess2

/if guess2 == chose
/say user: Ошибка! Ты выбрал {guess2}, а не 2.
/end

/if guess2 == chosetwo
/say user: Отлично, ты  {guess2}! Это круто!

/end

/end

/say user: Это конец...
```

THANK YOU

FOR WATCHING