**Описание решения**.

Решение представляет собой модификацию итогового проекта по блоку “Основные конструкции C++”. Подробные описания функциональности и реализации итогового проекта можно найти в <https://github.com/dmitryfedotov-56/chat> в папке DOCS.

Функциональность решения не изменилась совершенно. В папке DEMO содержится приложение file\_chat.exe, демонстрирующее вариант с хранением базы данных в файле chat\_data. В базе данных уже содержатся данные трех пользователей и их демонстрационная переписка. Пользователи такие же, как и прежде.

|  |  |
| --- | --- |
| **Login** | **Password** |
| user1 | pass1 |
| user2 | pass2 |
| user3 | pass3 |

Реализация изменилась следующим образом.

Oбъект, содержащий данные пользователя.

Объект БД User (описываемый классом User\_Class – наследником абстрактного класса User\_Access) теперь хранит не пароль пользователя, а дайжест, вычисленный при помощи sha1.

Объект, содержащий список пользователей.

Список пользователей теперь реализован не как двусвязный список объектов, а как хеш-таблица. Объект БД User\_List (описываемый классом User\_List - наследником абстрактного класса User\_List\_Access) хранит текущее состояние хеш-таблицы.

Хеш-таблица.

Хеш-таблица реализуется классом Hash\_Table. Элементы таблицы хранят только значения link типа size\_t. Это могут быть как указатели на какие-либо объекты, так и некоторые идентификаторы объектов. В данном случае в таблице хранятся идентификаторы объектов БД класса User\_Class.

Элементы таблицы не хранят значения ключей, ключ вычисляется по значению link при помощи функци allocate\_Key. Такая функция возвращает указатель на ключ, размещенный в куче (heap). В данном случае это необходимо, потому что ключ (как, впрочем, и все данные) прочитывается из энергонезависимой памяти (в даном случае из файла).

Методы Hash\_Table::init\_Link и Hash\_Table::next\_Link позволяют обойти все значения link, содержащиеся в таблице. В данном случае это необходимо для отправки широковещательных сообщений всем пользователям.

Если значение link не найдено, то методы find\_Link, init\_Link, next\_Link возвращают значение SIZE\_MAX.

Хеш-функция вычисляется методом умножения, в качестве коэффициента используется золотое сечение. При опредении последовательности пробирования используются “треугольные числа” (суммы арифметической прогрессии), т.к. известно, что для любого m = 0,…M-1 для различных n = 0, …M-1 значения (m + (n + n\*n)/2)%M различны, т.е. пробная последовательность покрывает весь массив.

Архитектура решения представлена на рис.1. Стрелки соответствуют директивам include. Заголовочные файлы показаны белым цветом. Пунктирными линиями показаны альтернативные реализации интерфейса.

Interface

chat\_interface.cpp

sha1.cpp

Hash\_table.h

chat\_access.h

Logic

chat\_class.h

chat\_logic.cpp

sha1.cpp

sha1.h

chat\_object.h

DBCS

chat\_object.cpp

storage\_access.h

Storage

ram\_storage.cpp

file\_storage.cpp

Рис. 1. Схема зависимостей модулей.

Решение может быть представлено в виде нескольких слоев: Interface, Logic, DBCS, Storage. Функциональность каждогог слоя представлена заголовочным файлом. Реализция каждого слоя независима от реализации других слоев. Интерфейс реализован в файле chat\_interface.cpp.

Функциональность слоя Logic представлена файлом chat\_access.h, содержащим описания абстрактных классов. Файл Файл chat\_class.h содержит описание классов – наследников абстрактных классов, описанных в chat\_access.h. Файл chat\_logic.cpp содержит реализацию классов, описаных в chat\_class.h.

Функциональность слоя DBCS представлена файлом chat\_object.h, содержащим описание абстрактных классов. Реализация классов – наследников классов, описанных в chat\_object.h, содержится в chat\_object.cpp, при этом используется функциональность sha1 и hash\_table.

Функциональность слоя Storage описывается файлом storage\_access.h, описывающим доступ к среде размещения данных. Имеются две альтернативных реализации этой функциональности - ram\_storage.cpp и file\_storage.cpp.

**Список исходных файлов.**

|  |  |
| --- | --- |
| **Файл** | **Пояснения** |
| sha1.h | интерфейс sha1 |
| sha1.cpp | реализация sha1 |
| hash\_table.h | интерфейс hash\_table |
| hash\_table.cpp | реализация hash\_table |
| chat\_interface.cpp | Реализация интерфейса чата |
| chat\_access.h | Интерфейс чата |
| chat\_class.h | Описание классов чата |
| сhat\_logic.cpp | Реализация логики чата |
| сhat\_object.h | Интерфейс БД чата |
| сhat\_object.cpp | Реализация БД чата |
| storage\_access.h | Интерфейс хранилища данных |
| ram\_storage.cpp | Реализация хранения данных в RAM. |
| file\_storage.cpp | Реализация хранения данных в файле |