**1) В чем заключается метод закрытого хеширования?**

Первый метод назывался открытым, потому что он позволял хранить сколь угодно много элементов, а при закрытом хешировании их количество ограниченно размером хеш-таблицы. В отличие от открытого хеширования закрытое не требует каких-либо дополнительных структур данных. В ячейках таблицы хранятся не указатели, а элементы исходного массива, доступ к каждому из которых осуществляется по хеш-коду ключа, при этом одна ячейка может содержать только один элемент.

**2) Что такое коллизия?**

Это когда у двух разных входных элементов хэш-таблицы будет одинаковым индекс.

**3) Что такое "линейная последовательность проб"?**

Линейное последовательность проб сводится к последовательному перебору сегментов таблицы с некоторым фиксированным шагом.

index = хэш-функция + c\*i

i – номер попытки разрешить коллизию

с-константа

**4) Что такое "квадратичная последовательность проб"?**

Квадратичная последовательность проб основана

на нелинейной адресации, достигаемой за счет

суммирования значений основной и дополнительной хеш-функций.

Выбираем шаг q. При попытке добавить элемент в занятую ячейку. начинаем последовательно просматривать ячейки

q=1,4,9,16…

**5) В чем заключается метод двойного хеширования?**

Метод борьбы с коллизиями, возникающими при открытой адресации, основанный на использовании двух хеш-функций для построения различных последовательностей исследования хеш-таблицы.

index= h1(x) + i\*h2(x)

**6)В чем заключается алгоритм добавления данных в хеш-таблицу?**

while (i != hash\_size)

{

Index = (number + i) % hash\_size;

if (hash\_array[Index].data == 0 || hash\_array[Index].data == -1)

{

hash\_array[Index].data = number;

hash\_array[Index].tries = i;

found = true;

break;

}

else

{

i++;

}

}

**7)В чем заключается алгоритм поиска данных в хеш-таблице?**

while (i != hash\_size)

{

int index = (number + i) % hash\_size;

if (hash\_array[index].data == number)

{

cout << "Число " << number << " найдено" << endl;

cout << "Кол-во попыток при поиске "<<i << endl;

found = true;

break;

}

else if (hash\_array[index].data == 0)

{

break;

}

i++;

}

**8)Удаление данных из хеш-таблицы.**

while (i != hash\_size)

{

int index = (number + i) % hash\_size;

if (hash\_array[index].data == number)

{

hash\_array[index].data = -1;

hash\_array[index].tries = -1;

cout << "Число<< number << удалено" << endl;

found = true;

break;

}

i++;

}

**9)Что такое хеш-функция, основанной на методе деления с остатком?**

Метод хеширования посредством деления заключается во взятии остатка от деления ключа на максимально возможное число хеш-кодов.

**10)Что такое хеш-функция, основанной на методе умножения?**

Здесь производится умножение ключа на некую константу C, лежащую в интервале [0..1]. После этого берется дробная часть этого выражения и умножается на максимально возможное число хеш-кодов.

**11)Как преобразовать символьную информацию в хеш-значение?**

Вычитание из символов самый минимальный код

**12)Что такое "открытое хеширование"?**

Принцип организации хеш-таблицы методом открытого хеширования заключается в реализации логически связанных цепочек, начинающихся в ячейках хеш-таблицы. Под цепочками подразумеваются связанные списки, указатели на которые хранятся в ячейках хеш-таблицы. Каждый элемент связанного списка – блок данных, и если с некоторым указателем, хранящимся по адресу **i**, связаны **n** таких блоков (n>1), то это свидетельствует о том, что **n** ключей получили один и тот же хеш-код **i**, т. е. имеет место коллизия. Но метод открытого хеширования устраняет конфликт, поскольку данные хранятся не в самой таблице, а в связных списках, которые увеличиваются при возникновении конфликта.

**13)Эффективность метода закрытого хеширования**

В отличие от открытого хеширования закрытое не требует каких-либо дополнительных структур данных.

**14)Эффективность метода открытого хеширования**

Он позволял хранить сколько угодно много элементов, а при закрытом хешировании их количество ограниченно размером хеш-таблицы.