Строки в С++ и функции для работы с ними подключаются с помощью библиотеки <string> (пишем в самом начале #include <string>).

Мы можем объявлять строки несколькими способами:

1. Объявление пустой строки:

* string s1; //по умолчанию создаётся пустая строка

1. Объявление с инициализацией:

* Объявление с инициализацией-присваиванием одной фразы:  
  string s2 = "abcd";//присвоили фразу abcd строке s2
* Объявление с инициализацией (вносим фразу):  
  string s3("abcd"); //внесли abcd в строку s3
* Объявление с инициализацией (вносим n одинаковых символов):  
  string s4(4, 'p');//внесли 4 символа p в строку s4
* Объявление с присваиванием:  
  string s5 = s3; //объявили пустую строку s5 и присвоили ей значение s3 => в s5 содержится теперь abcd

Операции над строками:

1. Конкатенация (объединение). По сути сложение нескольких строк в одну:

* cout << s4 + s3; //результат вывода: ppppabcd
* string s6 = s2 + s5; //s6 содержит abcdabcd

1. Сравнение строк:

* bool t = s2 > s3; //сравниваем строки s2 и s3. Т.к. эти строки, как мы знаем, у нас содержат одинаковые наборы символов, t = false
* t = s5 == s3; //сравниваем строки s5 и s3. Т.к. у нас они содержат одинаковые значения, t = true
* Остальные операторы сравнения (<=, >=, !=) действуют по тому же принципу

**Важно.** Как понять, какая из двух строк больше? Строки сравниваются посимвольно (т.е. как в массивах - каждый элемент с одним индексом одной строки сравнивается с элементом с таким же индексом другой строки).

1. Строки считаются равными, если все символы обеих строк совпадают. Если хотя бы один символ будет отличаться, то какая-то из строк будет считаться большей, а какая-то - меньшей.
2. Когда мы находим первую пару отличающихся символов, мы открываем таблицу ASCII и смотрим, какой из двух символов, которые мы сравниваем, идёт раньше по таблице ASCII. Этот символ будет считаться меньшим => эта строка меньше.

**Вопрос**. Строка а1 содержит последовательность abcd. Строка а2 содержит последовательность bcde. Какая из этих строк больше?

**Решение**. Здесь мы можем логически догадаться, что в ASCII алфавитный порядок тоже учитывается. Поэтому чем дальше идёт буква по алфавиту, тем она считается больше. Следовательно, b > a. Ответ: a2 > a1.

**Вопрос.** Строка а1 содержит последовательность 14dfg. Строка а2 содержит последовательность 13dfg. Какая из двух строк меньше?

**Решение**. Первый символ в этих двух строках одинаковый, поэтому переходим к проверке следующей пары символов. По ASCII 4 > 3. Ответ: a2 < a1.

**Вопрос**. Строка а1 содержит последовательность 7kj\_i. Строка a2 содержит последовательность 7kj!u. Какая из двух строк больше?

**Решение**. Дай ответ самостоятельно, используя таблицу ASCII :)

**Методы строк**.

**Метод size().** Используется для того, чтобы узнать количество символов в строке. Возвращает целое число. Запись: s1.size();

**Метод empty().** Проверяет пустая наша строка или нет. Если пустая, то возвращает true. В ином случае - false. Запись: s1.empty();

**Метод erase().** Используется для того, чтобы удалить часть строки.

* s3.erase(2); // удаляется всё после 2 символа
* s6.erase(2, 4); // удаляется всё от 2 до 4 символа

**Метод insert().** Используется для того, чтобы вставить некоторые символы в строку.

* s2.insert(1, 3, '!'); // вставляем 3 восклицательных знака, начиная от позиции 1
* s3.insert(4, s2); // вставляем то, что у строке s2 в строку s3, начиная от позиции 4
* s5.insert(0, s3, 3, 2); // вставляем в строку s5 два символа из строки s3 от символа 3 строки s3. Начальная позиция вставки - 0

**Метод find().** Используется, чтобы найти от какого индекса в строке начинается, нужная последовательность символов. Возвращает целое число, равное индексу.

* s5.find(s4); //ищем, где в первый раз появляется строка s4 полностью в строке s5. Поиск начинаем от индекса 0.
* s6.find("d", 4); // ищем, где в первый раз появляется символ d в строке s6. Поиск начинаем от индекса 4.
* s3.find(s2, 2, 3); //ищем первые 3 символа строки s2 в строке s3, начиная от индекса 2.

**Метод getline()**. Используется, чтобы получить предложение (строку с пробелами).

Пример:

string s7; //объявляем две строки

string s8;

cin >> s7; //вводим s7 через cin. Если поставить пробел, то выведутся только символы до пробела

getline(cin, s8); //вводим s8 через поток cin с помощью getline. Если поставить пробел, то всё равно введутся все символы.