**Класс string**

Для того чтобы использовать объекты класса string, необходимо включить заголовочный файл:

**#include <string>**

|  |  |  |
| --- | --- | --- |
| *Операция* | *Примеры* | *Результат* |
| Инициализация | string S1 = "sun", S2, S3;  S2 = "day";  S3 = '7'; //нельзя при объявлении  string A(S1), B("hello"); | S1 = "sun"  S2 = "day"  S3 = "7"  A = "sun"  B = "hello" |
| Прочитать строку до пробелов | cin >> A >> S;  //входной поток: "sun day" | A = "sun"  S = "day" |
| Прочитать строку с пробелами | getline(cin, S);  //входной поток: "sun day" | S = "sun day" |
| Копирование одной строки в другую | S2 = S1;  //находятся в разных областях памяти | S2 = "sun"  S1 = "sun" |
| Доступ к отдельным символам строки для чтения и записи | B[1] = 'E';  S3[0] = '5'; | B = "hEllo"  S3 = "5" |
| Проверка двух строк на равенство | if (A == S1)…  if (S1 != S2)… | 1  1 |
| Конкатенация двух строк | string S1 = "sun", S2 = "day";  string S3 = S1 + S2 + '!';  S1 += S2; | S3 = "sunday!"  S1 = "sunday" |
| Вычисление длины строки | int n1 = S1.size();  int n2 = S1.length(); | n1 = 6  n2 = 6 |
| Возможность узнать, пуста ли строка | string A;  if (!A.size())…  // метод empty() проверяет, пуста ли строка  if (A.empty())… | A = "" A.size()равно 0  истина |
| Преобразование string в char\* | string A = "day";  const char \*str = A.c\_str(); |  |
| Выделить из строки s подстроку, начинающуюся с позиции i, длиной k символов | string S = "sunday";  string S1 = S.substr(0, 3);  int i = 0, k = 3;  string S2 = S.substr(i, k);  string S3 = S1.substr(2);  /\*  По умолчанию k имеет значение **string::npos**, что означает «подстрока продолжается до конца строки». Начальный индекс подстроки тоже имеет значение по умолчанию, 0.  \*/ | S = "sun"  B = "sun"  S3 = "nday" |
| Убрать часть строки s, начинающуюся с позиции i, длиной k символов | string S = "sunday";  //s.erase(i, k)  S.erase(0,3); | S = "day" |
| Найти индекс первого вхождения символа c в строку s;  найти индекс первого вхождения в строку s любого из символов строки s1 ;  можно добавить дополнительный целый параметр, тогда вхождения будут искаться только после указанной позиции (включительно). | string S = "sunday!!!";  int p1 = S.find\_first\_of('!');  string digits = "0123456789";  int p = S.find\_first\_of(digits);  int p3 = S.find\_first\_of('!', 7);  /\*  Если заменить в имени метода first на last, будет выдан индекс последнего вхождения.  \*/ | p1 = 6  p = string::npos  // если  // не найдено  p3 = 7 |
| Найти индекс первого символа первого вхождения строки s1 в строку s2 | string s2 = "Hello, world!", s1 = "llo";  cout << s2.find(s1);  cout << s2.find(s1, 3); | 2  string::npos |
| Вставить строку s1 в строку s в позицию, заданную индексом i:  s.insert(i, s1) | string s = "All numbers are odd.";  string s1 = "prime ";  cout << s.insert(4, s1); | "All prime numbers are odd." |
| Заменить один символ на другой | string B = "hello";  replace( B.begin(), B.end(), 'l', '\*' );  /\*функция пробегает диапазон от begin() до end(), которые возвращают указатели на начало и конец строки, и заменяет элементы, равные третьему своему параметру, на четвертый\*/ | B = "he\*\*o" |
| В строке s заменяет l1 символов, начиная с индекса i, на l2 экземпляров символа c. | s.replace(i, l1, l2, c); |  |