

Программирование на C++



| Минцифры
РОССИИ

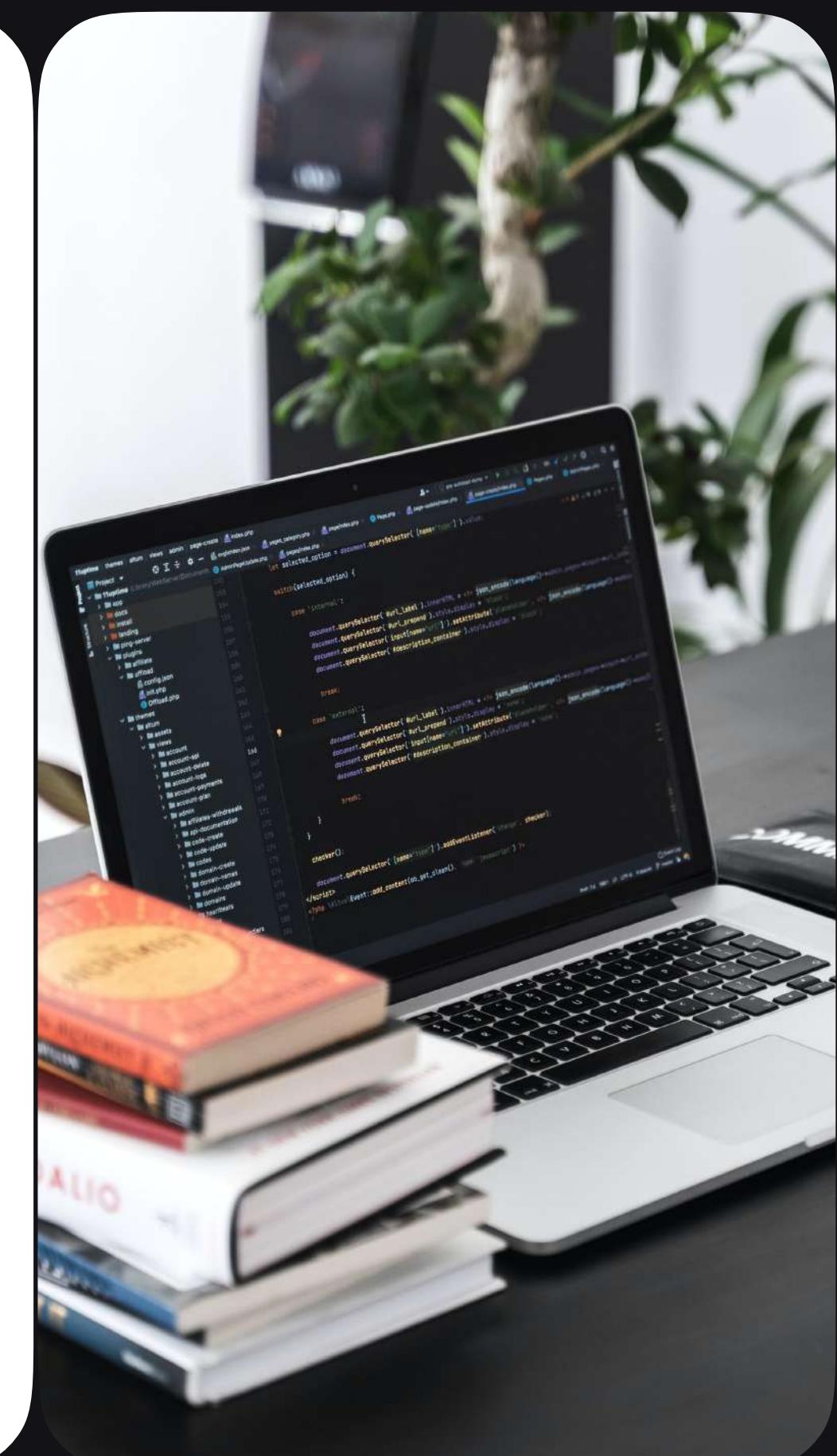
UCHI DOMA

20.35
УНИВЕРСИТЕТ

Урок 8 Модуль 3

Работа со строками

Полезные материалы



Цели урока



отработать на практике
работу со строками класса
`string` на C++



Класс string



В языке C++ для удобной работы со строками есть класс `string`, для использования которого необходимо подключить заголовочный файл `string`.

```
#include <string>  
using namespace std;
```

Строки

Строки можно объявлять и одновременно присваивать им значения:

```
string S1, S2 = "Hello";
```



Строка S1 будет пустой, строка S2 будет состоять из 5 символов.

К отдельным символам строки можно обращаться по индексу, как к элементам массива или С-строк. Например **S[0]** – это первый символ строки.

Для того, чтобы узнать длину строки можно использовать метод `size()` строки. Например, последний символ строки S это **S[S.size() - 1]**.

Конструкторы строк

Строки можно создавать с использованием следующих конструкторов:

string()

конструктор по умолчанию (без параметров) создает пустую строку

string(string & S)

копия строки S

string(size_t c)

строка из одного символа с

string(size_t n, char c)

повторение символа с заданное число n раз

string(string & S, size_t start, size_t len)

строка, содержащая не более, чем len символов данной строки S, начиная с символа номер start

Конструкторы строк

Конструкторы можно вызывать явно, например, так:

```
string s;  
s += string(10, 'z');
```



Явно вызывается конструктор `string` для создания строки, состоящей из 10 символов 'z'.

Конструкторы строк



Неявно конструктор вызывается при объявлении строки с указанием дополнительных параметров.

Например, так:

```
string S(10, 'z');
```

Ввод-вывод строк

Строка выводится точно так же, как и числовые значения:

```
cout << S;
```

Для считывания строки можно использовать операцию ">>" для объекта `cin`:

```
cin >> S;
```



В этом случае считывается строка из непробельных символов, пропуская пробелы и концы строк. Это удобно для того, чтобы разбивать текст на слова, или чтобы читать данные до конца файла при помощи `while (cin >> S)`.

Ввод-вывод строк

Можно считывать строки до появления символа конца строки при помощи функции `getline`. Сам символ конца строки считывается из входного потока, но к строке не добавляется:

```
getline(cin, S);
```

Арифметические операторы

Со строками можно выполнять следующие арифметические операции:

= присваивание значения

+= добавление в конец строки другой строки или символа

+ конкатенация двух строк, конкатенация строки и символа

==, != посимвольное сравнение

<, >, <=, >= лексикографическое сравнение



То есть можно скопировать содержимое одной строки в другую при помощи операции $S1 = S2$, сравнить две строки на равенство при помощи $S1 == S2$, сравнить строки в лексикографическом порядке при помощи $S1 < S2$, или сделать сложение (конкатенацию) двух строк в виде $S = S1 + S2$.

Методы строк

Метод `size()` возвращает длину строки. Возвращаемое значение является беззнаковым типом (как и во всех случаях, когда функция возвращает значение, равное длине строке или индексу элемента — эти значения беззнаковые). Поэтому нужно аккуратно выполнять операцию вычитания из значения, которое возвращает `size()`.



Например, ошибочным будет запись цикла, перебирающего все символы строки, кроме последнего, в виде `for (int i = 0; i < S.size() - 1; ++i)`.



Кроме того, у строк есть метод `length()`, который также возвращает длину строки.

Методы строк

resize

S.resize(n) – Изменяет длину строки, новая длина строки становится равна n. При этом строка может как уменьшиться, так и увеличиться. Если вызвать в виде **S.resize(n, c)**, где с – символ, то при увеличении длины строки добавляемые символы будут равны с.

Методы строк

clear

S.clear() – очищает строчку, строка становится пустой

Методы строк

empty

`S.empty()` — возвращает `true`, если строка пуста, `false` — если непуста

Методы строк

push_back

`S.push_back(c)` — добавляет в конец строки символ `c`, вызывается с одним параметром типа `char`

Методы строк

append

Добавляет в конец строки несколько символов, другую строку или фрагмент другой строки. Имеет много способов вызова.

S.append(n, c) – добавляет в конец строки n одинаковых символов, равных c. n имеет целочисленный тип, c – char

S.append(T) – добавляет в конец строки S содержимое строки T. T может быть объектом класса string или С-строкой

S.append(T, pos, count) – добавляет в конец строки S символы строки T начиная с символа с индексом pos количеством count

Итераторы

Функция `begin()` возвращает итератор, который указывает на первый элемент контейнера (при наличии в контейнере элементов).

Функция `end()` возвращает итератор, который указывает на следующую позицию после последнего элемента, то есть по сути на конец контейнера. Если контейнер пуст, то итераторы, возвращаемые обоими методами `begin` и `end` совпадают.

Итераторы обеспечивают доступ к элементам контейнера. С помощью итераторов очень удобно перебирать элементы, например элементы строки. Итератор описывается типом `iterator`. Но для каждого контейнера конкретный тип итератора будет отличаться.

Операции с итераторами:

`*iter:`

получение элемента, на который указывает итератор

`++iter:`

перемещение итератора вперед для обращения к следующему элементу

`--iter:`

перемещение итератора назад для обращения к предыдущему элементу

Пример



Посчитать количество символов «о» в строке

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int main()
6 {
7     string str = "Hello, World!";
8     cout << count(str.begin(), str.end(), 'o') << endl;
9     return 0;
10 }
```

Результат работы программы

2