

# Языки программирования



Строки  
стандартной  
библиотеки

# Строки

Стандартная библиотека C++ включает стандартный готовый класс `string` в заголовочном файле `string`.

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

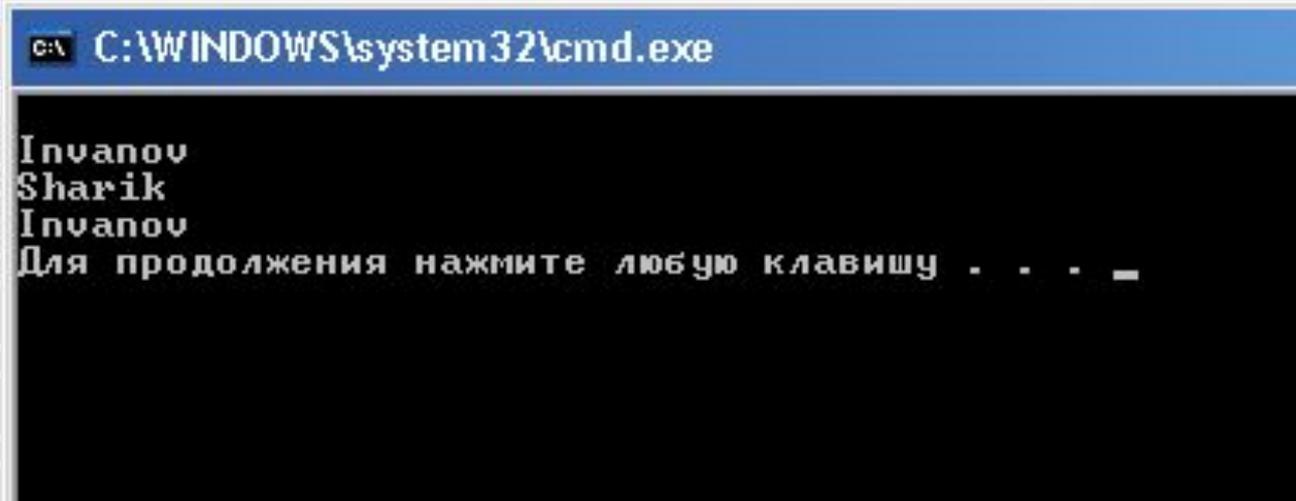
Класс `string` автоматически управляет памятью для строк, определяет большое количество удобных операций и методов.

# Конструкторы строк

- Создание пустой строки  
`string()`
- Создание строки на основе нуль-терминальной строки  
`string(const char* s)`
- Создание строки на основе первых *n* символов нуль-терминальной строки  
`string(const char* s, int n)`
- Конструктор копирования  
`string(string& s)`

# Пример использования конструкторов

```
string s1;  
string man("Ivanov");  
string dog("Sharik - is a good dog" );  
string  
cout << Ivanov  
cout << Sharik  
cout << Ivanov  
cout << Для продолжения нажмите любую клавишу . . .  
cout <<  
cout <<
```



# Присвоение строк

- Присвоение строки типа `string`  
`string operator=(const string& s);`
- Присвоение нуль-терминальной строки  
`string operator=(const char* s);`
- Присвоение символа  
`string operator=(char c);`

# Пример использования конструкторов и присвоения

```
string s1, s2, s3;
```

```
s1 =
```

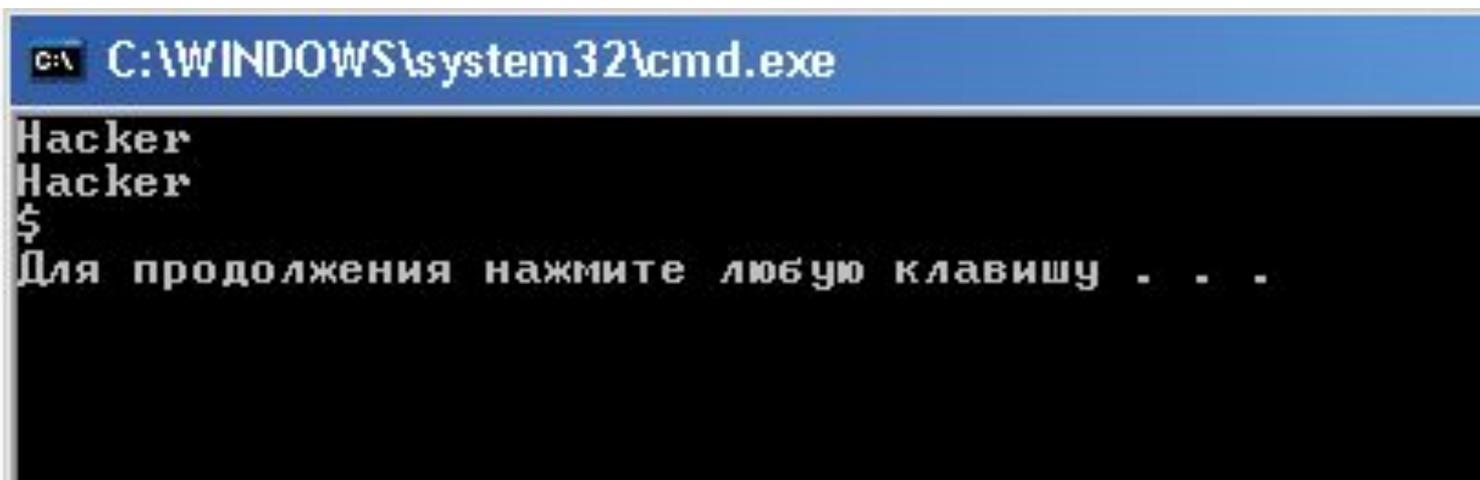
```
s2 =
```

```
s3 =
```

```
cout
```

```
cout
```

```
cout
```



```
C:\WINDOWS\system32\cmd.exe
Hacker
Hacker
$
Для продолжения нажмите любую клавишу . . .
```

# Методы получения характеристик строк

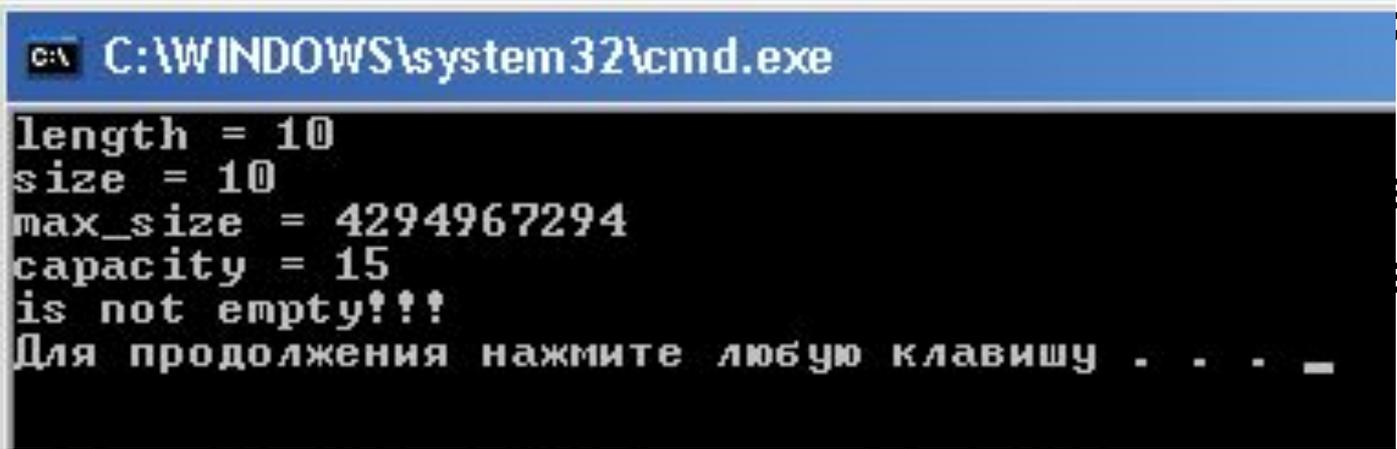
Метод	Назначение
<code>size_type length() const;</code>	получение длины строки
<code>size_type size() const;</code>	получение длины строки
<code>size_type max_size() const;</code>	максимальная допустимая длина строки
<code>size_type capacity() const;</code>	объем памяти, занимаемый строкой
<code>bool empty() const;</code>	возвращает истину $\Leftrightarrow$ строка пустая

`size_type` - тип размера, как правило `unsigned long`

# Пример

```
string s = "Pink Floyd";
```

```
cout << s.length() << endl;
cout << s.size() << endl;
cout << s.max_size() << endl;
cout << s.capacity() << endl;
if (!s.empty())
    cout << "is not empty..." << endl;
```



C:\WINDOWS\system32\cmd.exe

```
length = 10
size = 10
max_size = 4294967294
capacity = 15
is not empty!!!
Для продолжения нажмите любую клавишу . . . _
```

# Перегруженные операции

Операции	Назначение
=	присвоение
+, +=	сцепление (конкатенация) строк
==, !=, <, >, <=, >=	сравнение строк в лексикографическом порядке
>>, <<	ввод и вывод
[ ]	индексирование символов строки

# Замечания об индексировании

- Символы строки индексируются от 0 до `length() - 1`
- Для индексирования строк также существует метод

```
char& at(size_type index);
```

В отличие от `[]` он контролирует выход индекса за границу диапазона и при необходимости генерирует стандартное исключение `out_of_range`

# Пример

```
string s1, s2, s3;
```

```
cout <<
```

```
cin >>
```

```
cout << abcdef
```

```
cin >>
```

```
s3 = s1 + s2;
```

```
cout << abcdefg
```

```
cout <<
```

```
if (s1 < s2)
```

```
    cout <<
```

```
else if (s1 > s2)
```

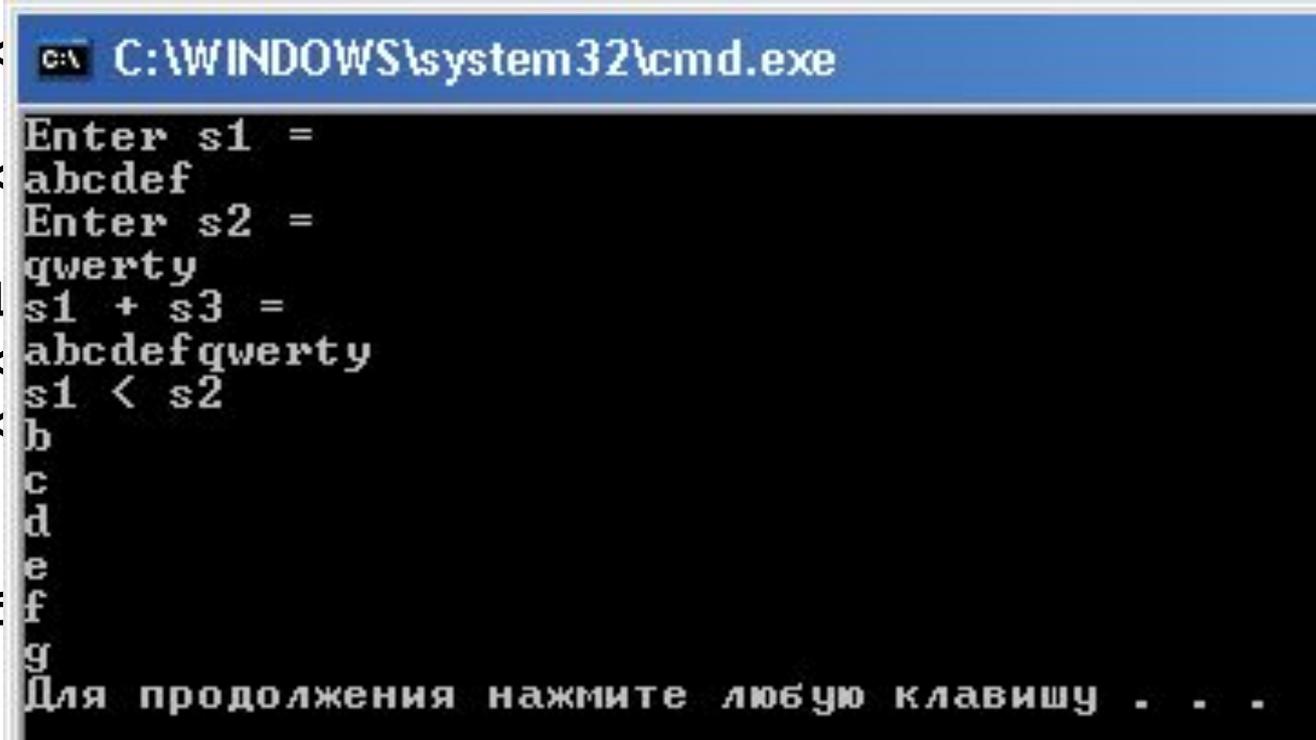
```
    cout <<
```

```
else
```

```
    cout << "s1 > s2" << endl;
```

```
for (int i = 0; i < s1.length(); i++) {
```

```
    s1[i]++;
```



```
C:\WINDOWS\system32\cmd.exe
Enter s1 =
abcdef
Enter s2 =
qwerty
s1 + s2 =
abcdefg
s1 < s2
b
Для продолжения нажмите любую клавишу . . .
```

# Методы присвоения строк

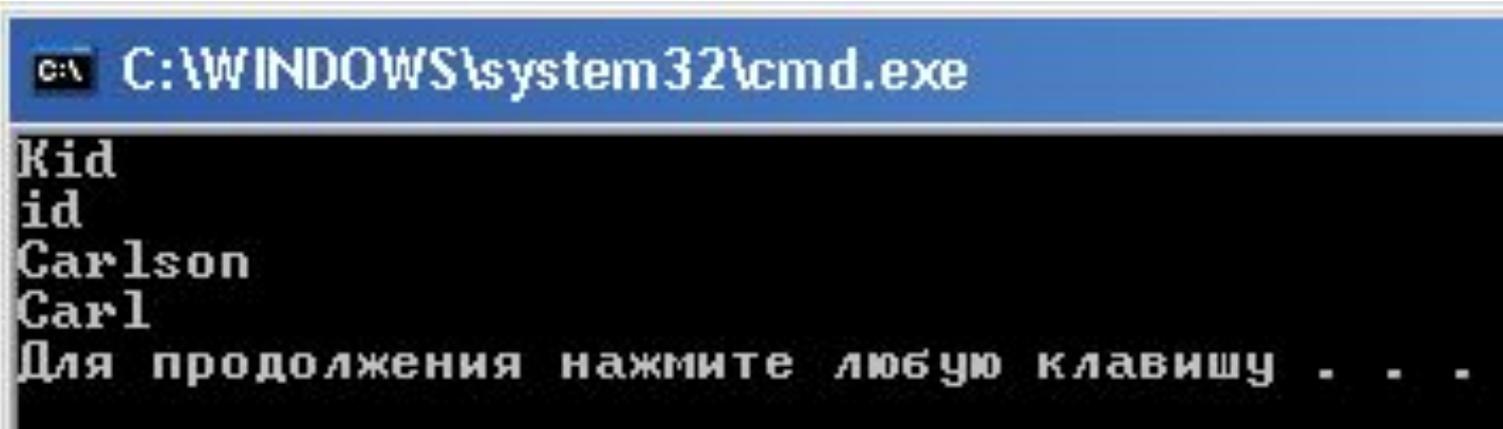
Метод	Назначение
<code>assign(const string&amp; s)</code>	присвоение вызывающей строке строки <code>s</code>
<code>assign(const string&amp; s, size_type pos, size_type n)</code>	присвоение вызывающей строке <code>n</code> символов строки <code>s</code> начиная с позиции <code>pos</code>
<code>assign(const char* s)</code>	присвоение вызывающей строке нуль-терминальной строки <code>s</code>
<code>assign(const char* s, size_type n)</code>	присвоение вызывающей строке первых <code>n</code> символов нуль-терминальной строки <code>s</code>

Первый и третий метод эквивалентны использованию операции =

# Пример

```
char os[] = "Carlson";
```

```
st:
```



```
C:\WINDOWS\system32\cmd.exe
Kid
id
Carlson
Carl
Для продолжения нажмите любую клавишу . . .
```

```
st:
```

```
s1
```

```
id
```

```
s2
```

```
Carlson
```

```
Carl
```

```
s3
```

```
Для продолжения нажмите любую клавишу . . .
```

```
s4.assign(os, 4); //4 - n
```

```
cout << s1 << endl << s2 << endl << s3 <<
    endl << s4 << endl;
```

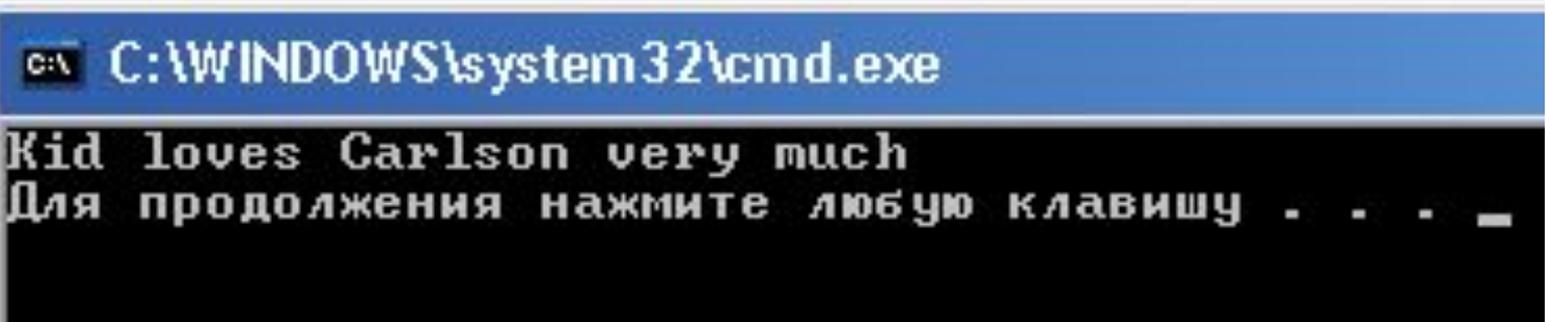
# Методы добавления строк

Метод	Назначение
<code>append(const string&amp; s)</code>	добавление <code>s</code> в конец вызывающей строки
<code>append(const string&amp; s, size_type pos, size_type n)</code>	добавление в конец вызывающей строки <code>n</code> символов строки <code>s</code> начиная с позиции <code>pos</code>
<code>append(const char* s)</code>	добавление в конец вызывающей строки нуль-терминальной строки <code>s</code>
<code>append(const char* s, size_type n)</code>	добавление в конец вызывающей строки первых <code>n</code> символов нуль-терминальной строки <code>s</code>

# Пример

```
char os[] = "Carlson";
```

```
string s = "Kid";
```



```
C:\WINDOWS\system32\cmd.exe
Kid loves Carlson very much
Для продолжения нажмите любую клавишу . . . _
```

```
s.append(1s, 5, 1); //5 - pos, 1 - n
```

```
s.append(os);
```

```
s.append(" very much thank you (c)Master  
Yoda", 10); //10 - n
```

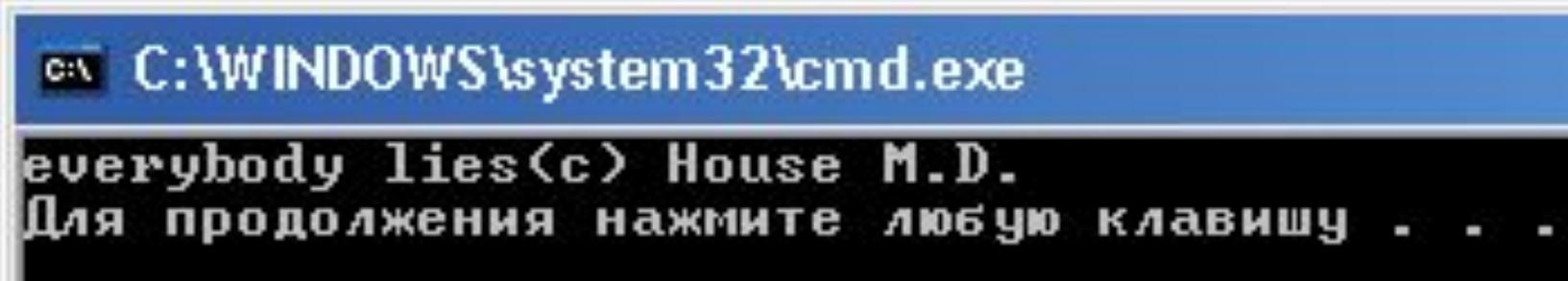
```
cout << s << endl;
```

# Методы вставки в строку

Метод	Назначение
<code>insert(size_type pos1, const string&amp; s)</code>	вставка строки <code>s</code> в позицию <code>pos1</code> вызывающей строки
<code>insert(size_type pos1, const string&amp; s, size_type pos2, size_type n)</code>	вставка <code>n</code> символов строки <code>s</code> , начиная с ее позиции <code>pos2</code> , в позицию <code>pos1</code> вызывающей строки
<code>insert(size_type pos1, const char* s)</code>	вставка нуль-терминальной строки <code>s</code> в позицию <code>pos1</code> вызывающей строки
<code>insert(size_type pos1, const char* s, size_type n)</code>	вставка первых <code>n</code> символов нуль-терминальной строки <code>s</code> в позицию <code>pos1</code> вызывающей строки

# Пример

```
char os[] = "lies";  
string ns = "every";
```



```
C:\WINDOWS\system32\cmd.exe
```

```
everybody lies(c) House M.D.
```

```
Для продолжения нажмите любую клавишу . . .
```

```
s.insert(0, ns);  
s.insert(5, ns1, 8, 4);  
s.insert(s.length(), os);  
s.insert(s.length(), "(c) House M.D. I'm too  
handsome to do paperwork.", 15);  
cout << s << endl;
```

# Методы удаления

- Метод удаления  $n$  символов строки начиная с позиции `pos`

```
erase(size_type pos = 0,  
      size_type n = npos)
```

если  $n$  не указано – удаляются символы до конца строки

- Метод очистки строки  
`clear()`

# Пример

```
string s = "1qwanza1234maydan";
```

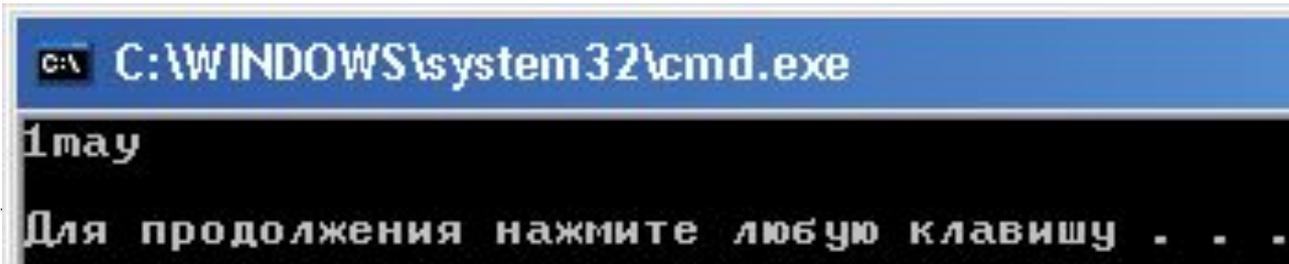
```
s.era
```

```
s.era
```

```
cout
```

```
s.erase(); //pos = 0, n = npos
```

```
cout << s << endl;
```



# Метод получения подстроки

- Получение подстроки длины `n`, начиная с позиции `pos`

```
string substr(size_type pos = 0,  
              size_type n = npos) const;
```

Если `n` не указывается, то возвращается подстрока из всех символов до конца вызывающей строки

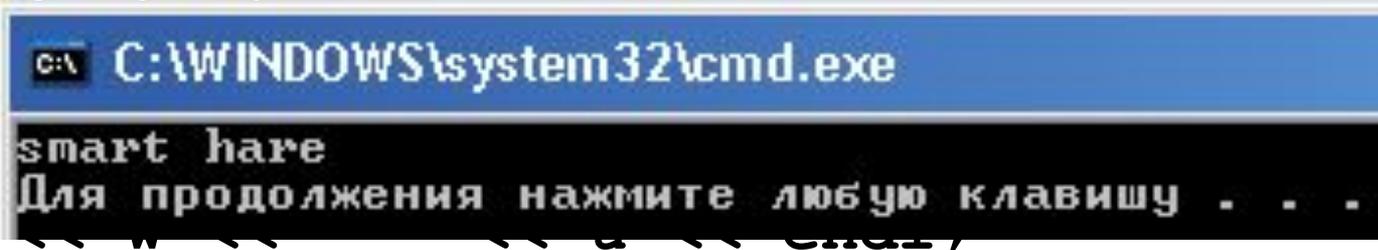
# Пример

```
string s = "foxy hare is very smart";  
string u, w;
```

```
u = s
```

```
w = s
```

```
cout
```



# Методы замены в строке

Метод	Параметры
<pre>replace(size_type pos1,         size_type n1,         const string&amp; s)</pre>	<p>pos1 – позиция в вызывающей строке, начиная с которой происходит замена</p> <p>n1 – количество заменяемых СИМВОЛОВ</p> <p>s – заменяющая строка</p>
<pre>replace(size_type pos1,         size_type n1,         const string&amp; s,         size_type pos2,         size_type n2)</pre>	<p>дополнительно:</p> <p>pos2 – позиция в строке s, начиная с которой идут заменяющие символы</p> <p>n2 – количество заменяющих СИМВОЛОВ</p>



# Методы поиска подстрок

- Метод нахождения позиции первого (левого) вхождения подстроки  $s$  в вызывающую строку. Поиск производится, начиная с позиции  $pos$ .

```
size_type find(const string& s,  
              size_type pos = 0) const;
```

Возвращаемое значение – индекс вхождения, если нет ни одного вхождения, то возвращается  $npos$ .

- Метод нахождения позиции последнего (правого) вхождения подстроки.  $pos$  определяет позицию, до которой происходит поиск

```
size_type rfind(const string& s,  
              size_type pos = npos) const;
```

# Пример

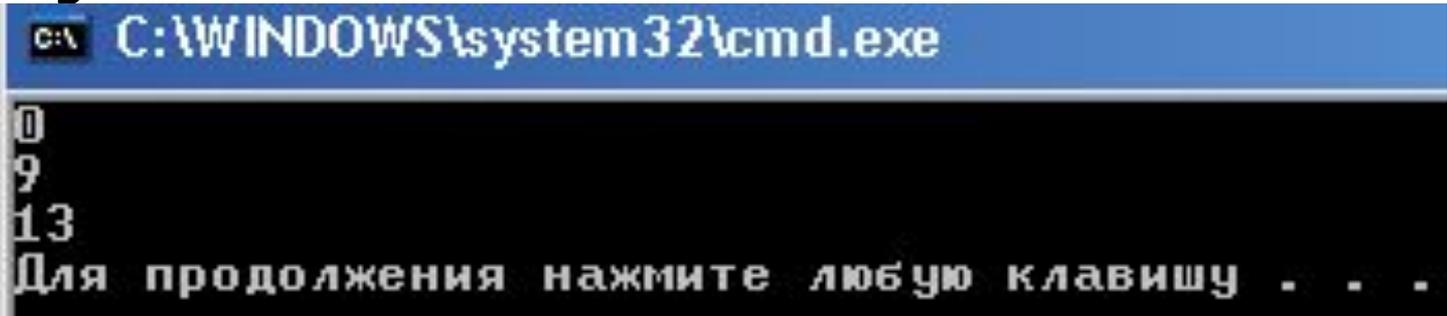
```
string s = "two tea. two two";
```

```
string str = s.substr(0, 10);
```

```
cout << str << endl;
```

```
cout << s << endl;
```

```
cout << s.substr(0, 10) << endl;
```



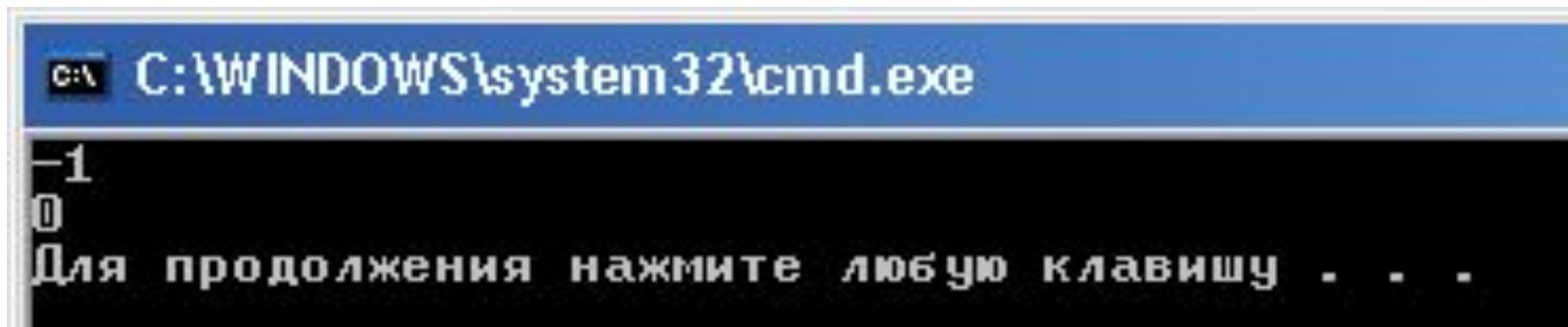
# Методы сравнения

```
int compare(const string& s) const;  
int compare(size_type pos1,  
            size_type n1,  
            const string& s,  
            size_type pos2,  
            size_type n2) const;
```

Возвращаемое значение интерпретируется также, как у функции `strcmp`

# Пример

```
string s = "aabb";  
string u = "aabc";  
cout << s.compare(u) << endl;  
cout << s.compare(0, 2, u, 0, 2) << endl;
```



The screenshot shows a Windows command prompt window with the title bar "C:\WINDOWS\system32\cmd.exe". The output of the program is displayed on the command line: "-1" on the first line and "0" on the second line. Below the output, there is a prompt "Для продолжения нажмите любую клавишу . . ." (Press any key to continue . . .).

# Дополнительные методы

- Получение нуль-терминальной строки

```
const char* c_str() const;
```

- Обмен строк

```
swap(string& s)
```

# Пример

```
string s1 = "Children of men";
```

```
st: c:\ C:\WINDOWS\system32\cmd.exe
```

```
ch: Children of men
```

```
Saving Private Ryan
```

```
st: Children of men
```

```
co: Для продолжения нажмите любую клавишу . . .
```

```
s1.swap(s2);
```

```
cout << s1 << endl << s2 << endl;
```

# Стандартные исключения при работе строк

- **out\_of\_range** – при неправильном указании индексов, позиций в операции [ ] и различных методах
- **length\_error** – при превышении максимально-допустимой длины строки