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

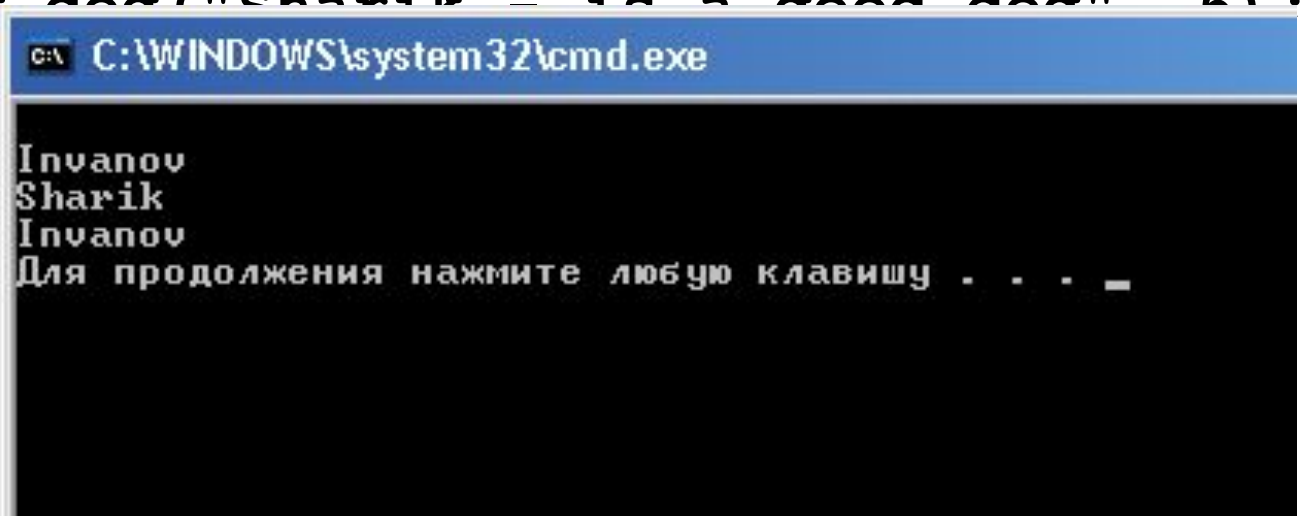
  `string(const char* s)`
- Создание строки на основе первых `n` символов нуль-терминальной строки

  `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", 6);
string
cout <
cout <
cout <
cout <
```



Присвоение строк

- Присвоение строки типа string

```
string operator=(const string& s);
```

- Присвоение нуль-терминальной строки

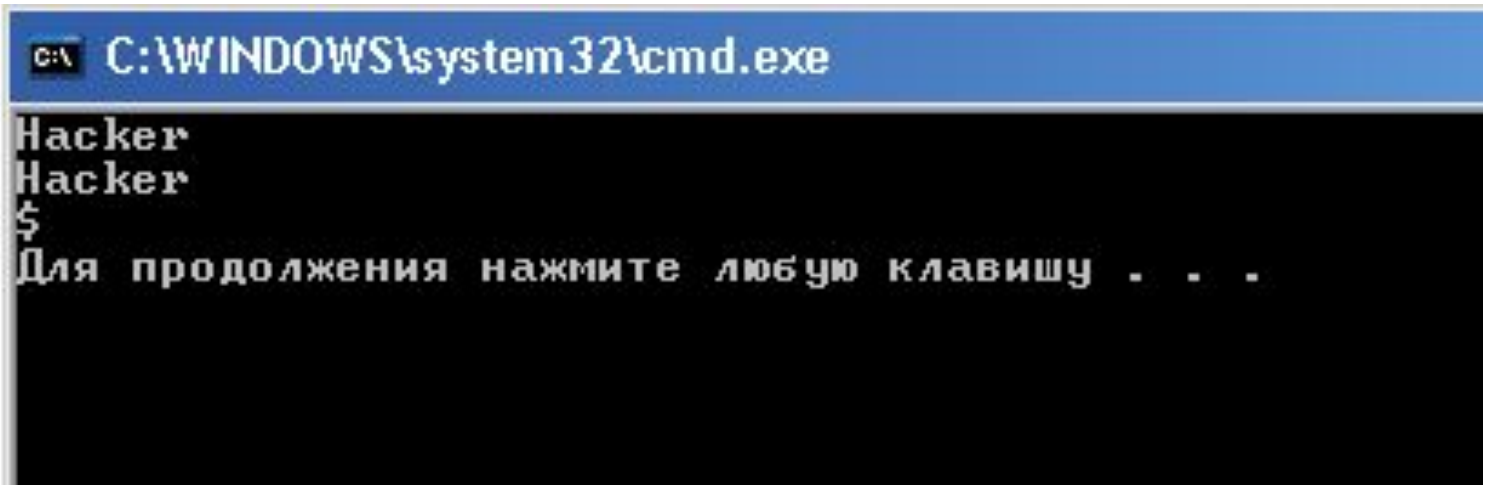
```
string operator=(const char* s);
```

- Присвоение символа

```
string operator=(char c);
```

Пример использования конструкторов и присвоения

```
string s1, s2, s3;
s1 =
s2 =
s3 =
cou
cou
cou
```



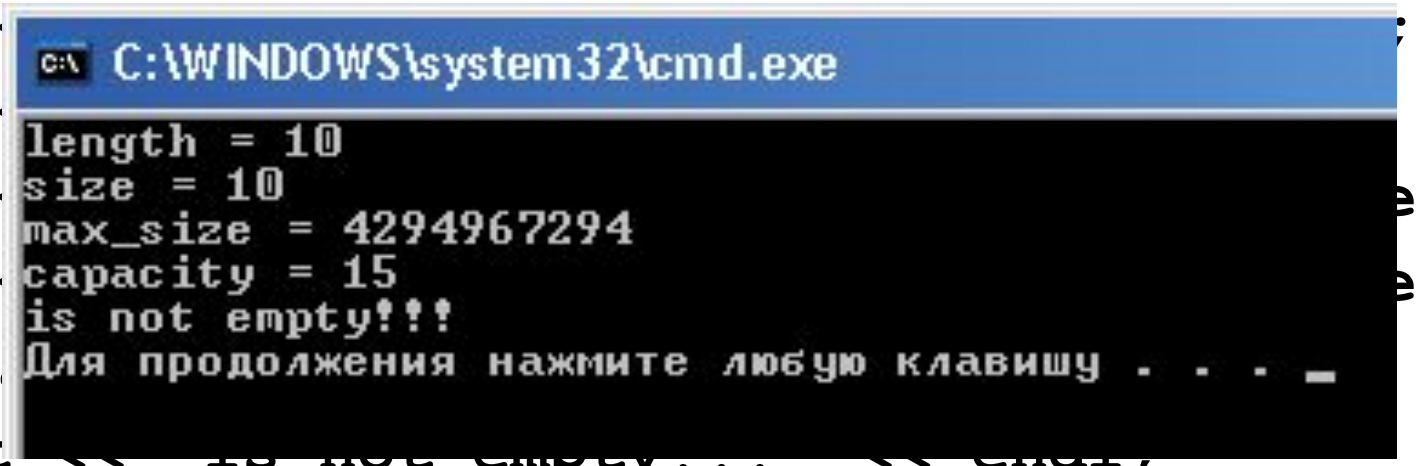
Методы получения характеристик строк

Метод	Назначение
`size_type length() const;`	получение длины строки
`size_type size() const;`	получение длины строки
`size_type max_size() const;`	максимальная допустимая длина строки
`size_type capacity() const;`	объем памяти, занимаемый строкой
`bool empty() const;`	возвращает истину ⇔ строка пустая

`size_type` – тип размера, как правило **`unsigned long`**

Пример

```
string s = "Pink Floyd";
cout <
cout <
cout <                                                    endl;
cout <                                                    endl;
if (!
  cout <<  is not empty!!! << endl;
```



Перегруженные операции

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

Замечания об индексировании

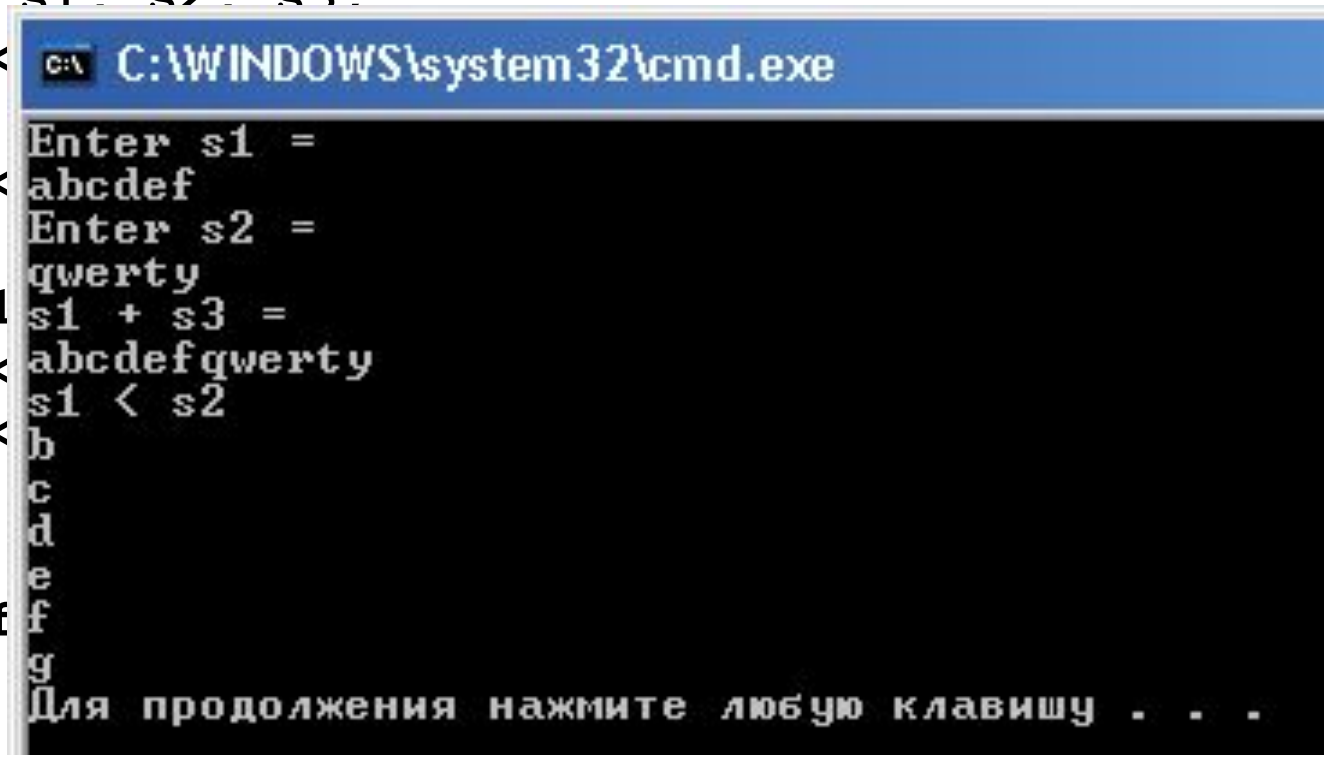
- Символы строки индексируются от `0` до `length()-1`
- Для индексирования строк также существует метод

  `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 <<
cin >>
s3 = s1
cout <<
cout <<
if (s1
  cout
else if
  cout
else
  cout << "s1 > s2" << endl;
for (int i = 0; i < s1.length(); i++) {
   s1[i]++;
```



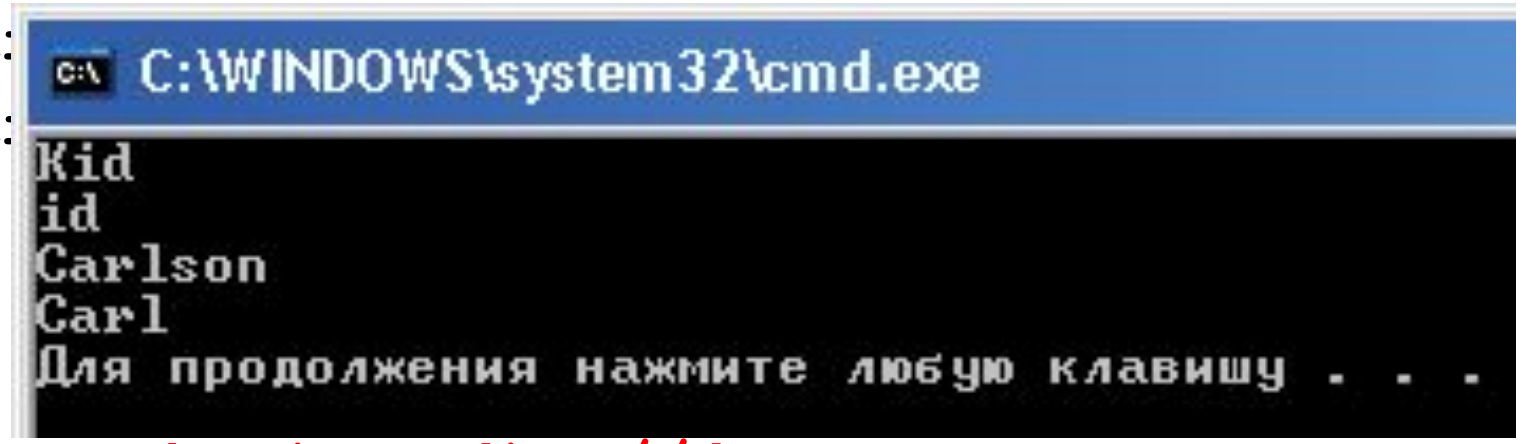
Методы присвоения строк

Метод	Назначение
`assign(const string& s)`	присвоение вызывающей строке строки `s`
`assign(const string& s, size_type pos, size_type n)`	присвоение вызывающей строке `n` символов строки `s` начиная с позиции `pos`
`assign(const char* s)`	присвоение вызывающей строке нуль-терминальной строки `s`
`assign(const char* s, size_type n)`	присвоение вызывающей строке первых `n` символов нуль-терминальной строки `s`

Первый и третий метод эквивалентны использованию операции =

Пример

```
char os[] = "Carlson";
st
st
s1
s2
s3
s4.assign(os, 4); //4 - n
cout << s1 << endl << s2 << endl << s3 <<
  endl << s4 << endl;
```

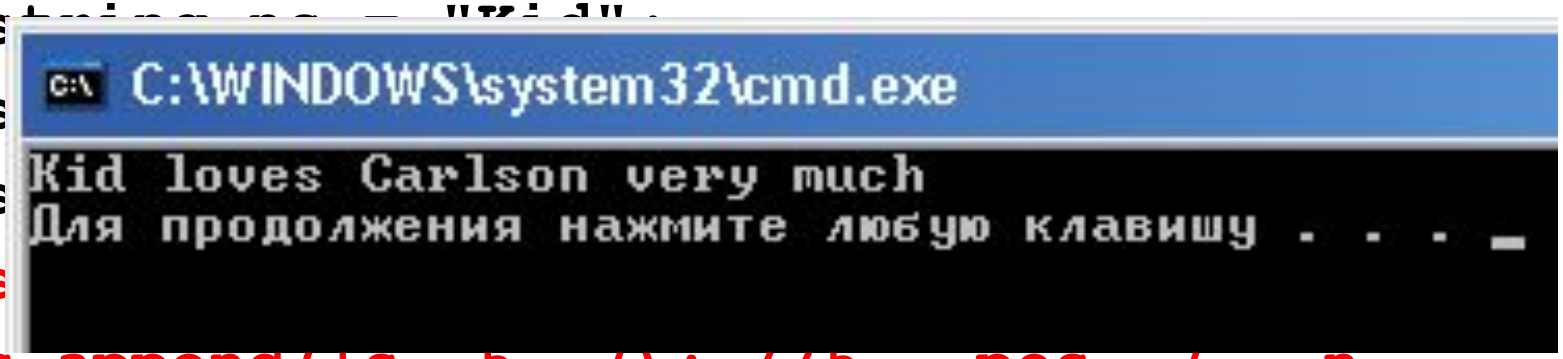


Методы добавления строк

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

Пример

```
char os[] = "Carlson";
s
s
s
s
s.append(is, 5, 7); //5 - pos, 7 - n
s.append(os);
s.append(" very much thank you (c)Master
  Yoda", 10); //10 - n
cout << s << endl;
```

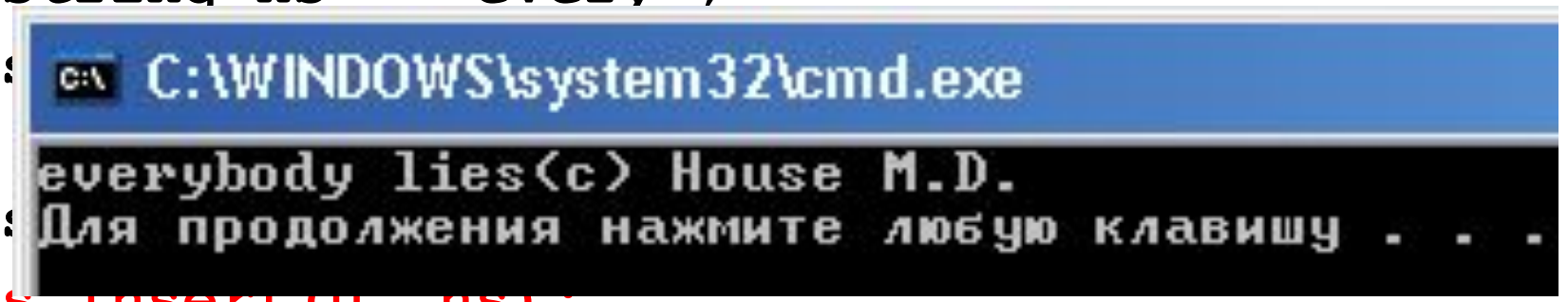


Методы вставки в строку

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

Пример

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




```
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)
```

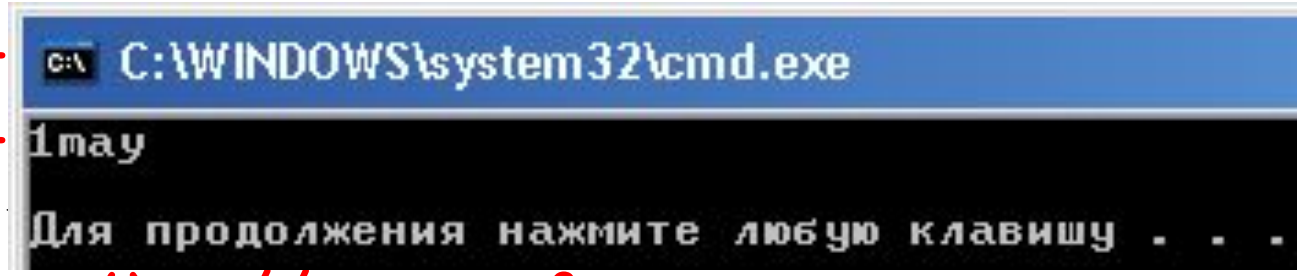
если `n` не указано – удаляются символы до конца строки

- 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

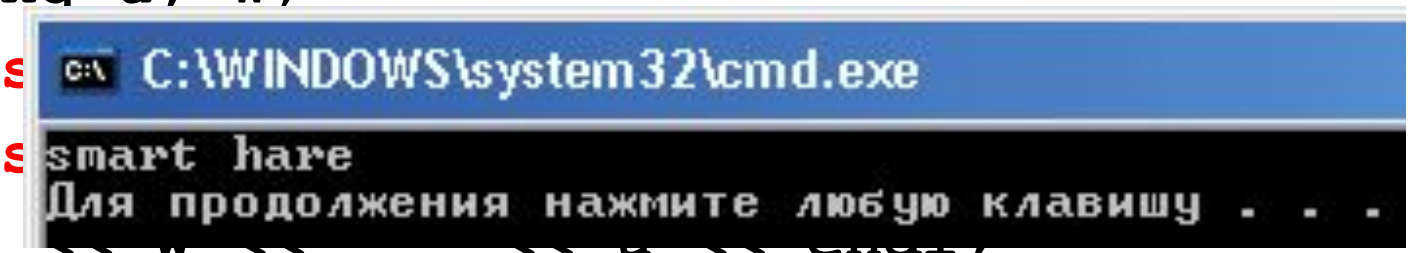
- Получение подстроки длины `n`, начиная с позиции `pos`

```
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
`replace(size_type pos1, size_type n1, const string& s)`	pos1 – позиция в вызывающей строке, начиная с которой происходит замена n1 – количество заменяемых символов s – заменяющая строка
`replace(size_type pos1, size_type n1, const string& s, size_type pos2, size_type n2)`	дополнительно: pos2 – позиция в строке **s**, начиная с которой идут заменяющие символы n2 – количество заменяющих символов

Пример

```
string u = "slonik";
str
str
s.r
s.replace(3, 3, w, 4, 4);
cout << s << endl;
```

Методы поиска подстрок

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

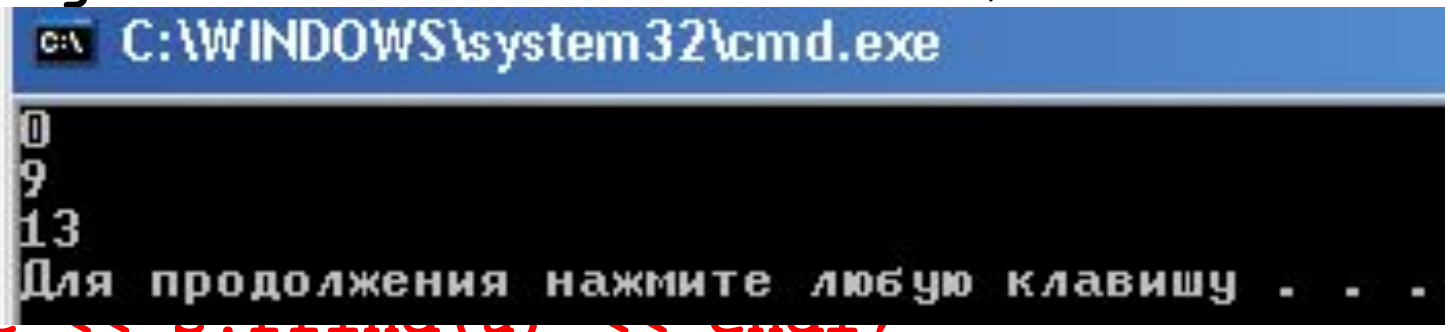
```
size_type find(const string& s,
               size_type pos = 0)const;
```

  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";
stri
cout
cout
cout << s.find(a) << endl;
```



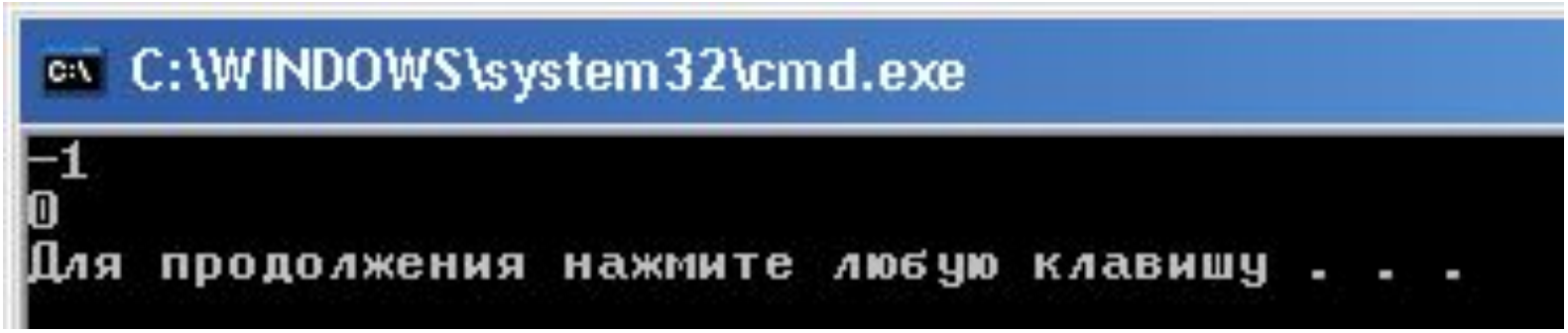
Методы сравнения

```
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 `stcmp`

Пример

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



Дополнительные методы

- Получение нуль-терминальной строки

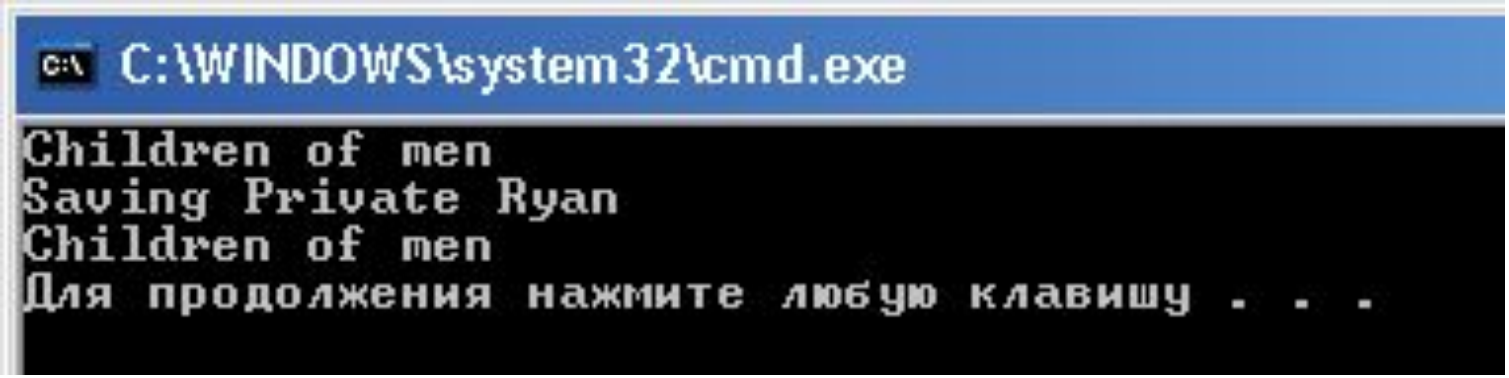
```
const char* c_str() const;
```

- Обмен строк

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

Пример

```
string s1 = "Children of men";
st
cha
st
cou
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