

# Некоторые функции обработки строк в языке программирования C++

Подготовил  
учащийся 11 Б  
класса

Мурко Кирилл

# Длина строки

- `size_t length() const;`
- `size_t size() const;`

```
#include <iostream>
#include <string>
using namespace std;
int main (){
    string str ("Test string");
    cout << "The length of str is " << str.length() << " characters.\n";
    cout << "The size of str is " << str.size() << " characters.\n";
    return 0;}
```

---

The length of str is 11 characters.

The size of str is 11 characters.

# c\_str – преобразование строки типа string в char\*

- const char\* c\_str ( ) const;

```
#include <iostream>
#include <cstring>
#include <string>
using namespace std;
int main (){
    char * cstr, *p;
    string str ("Please split this phrase into tokens");
    cstr = new char [str.size()+1];
    strcpy (cstr, str.c_str());
    p=strtok (cstr, " ");
    while (p!=NULL) {
        cout << p << endl;
        p=strtok(NULL, " "); }
    delete[] cstr;
    return 0;}
```

---

Please  
split  
this  
phrase  
into  
tokens

# empty – проверка, является ли строка пустой

- `bool empty ( ) const;` Возвращает true если строка пустая

```
#include <iostream>
#include <string>
using namespace std;
int main (){
    string content;
    string line;
    cout << "Please introduce a text. Enter an empty line to finish:\n";
    do {
        getline(cin,line);
        content += line + '\n';
    } while (!line.empty());
    cout << "The text you introduced was:\n" << content;
    return 0;}
```

---

Эта программа ожидает ввода данных пользователем строка за строкой и сохраняет его в содержимом строки, пока не введётся пустая строка.

# compare – функция сравнения строк

- `int compare ( const string& str )const;`
- `int compare ( const char* s ) const;`
- `int compare ( size_t pos1, size_t n1, const string& str ) const;`
- `int compare ( size_t pos1, size_t n1, const char* s);`
- `int compare ( size_t pos1, size_t n1, const string& str, size_t pos2, size_t n2 ) const;`
- `int compare ( size_t pos1, size_t n1, const char* s, size_t n2) const;`

Переменной `result` присваивается 0, если строки идентичны, положительное число, если `str1` лексикографически больше `str2`, и отрицательное число, если `str1` лексикографически меньше `str2`.

(Лексикон — это словарь. Когда мы говорим, что одна строка лексикографически меньше другой, мы имеем в виду, что первая строка идет по алфавиту раньше. Компьютер применяет тот же критерий, что применили бы вы, расставляя по алфавиту имена в списке.)

# Example

```
#include <iostream>
#include <string>
using namespace std;
int main (){
    string str1 ("green apple");
    string str2 ("red apple");
    if (str1.compare(str2) != 0)
        cout << str1 << " is not " << str2 << "\n";
    if (str1.compare(6,5,"apple") == 0)
        cout << "still, " << str1 << " is an apple\n";
    if (str2.compare(str2.size()-5,5,"apple") == 0)
        cout << "and " << str2 << " is also an apple\n";
    if (str1.compare(6,5,str2,4,5) == 0)
        cout << "therefore, both are apples\n";
    return 0;}
```

gre

---

en apple is not red apple  
still, green apple is an apple  
and red apple is also an apple  
therefore, both are apples

# erase - удаление символов из строки

- `string& erase ( size_t pos = 0, size_t n = npos );` Удаляет последовательность из `n` символов, начиная с позиции `pos`. Обратите внимание, что оба параметра могут быть опущены: только с одним аргументом, функция удаляет все, начиная с позиции `pos` вперед, и без аргументов, функция удаляет всю строку.

```
#include <iostream>
#include <string>
using namespace std;
int main (){
    string str ("This is an example phrase.");
    str.erase (10,8);
    cout << str << endl;    // "This is an phrase."
    return 0;}

```

# insert

- `string& insert ( size_t pos1, const string& str );` Вставляет копию всей подстроки в позицию символа `pos1`.
- `string& insert ( size_t pos1, const string& str, size_t pos2, size_t n );` Вставляет копию подстроки начиная с позиции символа `pos2` и длиной `n` символов в позицию символа `pos1`.
- `string& insert ( size_t pos1, const char* s, size_t n);` Вставляет в позицию символа `pos1`, копию строки, сформированной первыми `n` символами в массиве символов, указанных `s`.
- `string& insert ( size_t pos1, const char* s );` Вставляет в позицию символа `pos1`, копию строки, сформированной последовательностью символов (в строке), предоставленных `s`.
- `string& insert ( size_t pos1, size_t n, char c );` Вставляет строку, полученную путем повторения символа `c`, `n` раз, на позиции символа `pos1`.



# EXAMPLE

```
#include <iostream>
#include <string>
using namespace std;
int main (){
    string str="to be question";
    string str2="the ";
    string str3="or not to be";
    str.insert(6,str2);           // to be (the )question
    str.insert(6,str3,3,4);      // to be (not )the question
    str.insert(10,"that is cool",8); // to be not (that is )the question
    str.insert(10,"to be ");     // to be not (to be )that is the question
    str.insert(15,1,':');        // to be not to be(:) that is the question
    cout << str << endl;
    return 0;}
```

# replace - замена части строки

- `string& replace ( size_t pos1, size_t n1, const string& str );`  
Раздел строки от `pos1` длиной `n1` заменяется копией объекта `string str`.
- `string& replace ( size_t pos1, size_t n1, const string& str, size_t pos2, size_t n2 );`  
Раздел строки от `pos1` длиной `n1` заменяется копией подстроки `str` от `pos2` длиной `n2`.
- `string& replace ( size_t pos1, size_t n1, const char* s, size_t n2 );`  
Раздел строки от `pos1` длиной `n1` заменяется копией строки сформированной первыми `n2` символами в массиве символов, указанных `s`.
- `string& replace ( size_t pos1, size_t n1, const char* s );`  
Раздел строки от `pos1` длиной `n1` заменяется копией строки сформированной символами в массиве символов, указанных `s`.
- `string& replace ( size_t pos1, size_t n1, size_t n2, char c );`  
Раздел строки от `pos1` длиной `n1` заменяется повторением символа `c`, `n2` раз.

# EXAMPLE

```
#include <iostream>
#include <string>
using namespace std;
int main (){
    string base="this is a test string.";
    string str2="n example";
    string str3="sample phrase";
    string str4="useful.";
    string str=base;           // "this is a test string."
    str.replace(9,5,str2);     // "this is an example string."
    str.replace(19,6,str3,7,6); // "this is an example phrase."
    str.replace(8,10,"just all",6); // "this is just a phrase."
    str.replace(8,6,"a short"); // "this is a short phrase."
    str.replace(22,1,3,'!');   // "this is a short phrase!!!"
    cout << str << endl;
    return 0;}

```

# find – функция поиска первого вхождения подстроки в строку

- `size_t find ( const string& str, size_t pos = 0 ) const;`
- `size_t find ( const char* s, size_t pos, size_t n ) const;`
- `size_t find ( const char* s, size_t pos = 0 ) const;`
- `size_t find ( char c, size_t pos = 0 ) const;`

функция `find` осуществляет поиск подстроки слева направо(прямой поиск). Если подстрока найдена, возвращается индекс начальной позиции этой подстроки. Если строка не найдена, возвращается значение `string::npos` (открытая статическая константа, определенная в классе `string`). Это значение возвращают функции `string`, связанные с поиском, для указания того, что подстрока или символ в строке не найдены.

[Замечание. Остальные функции поиска, представленные в этом разделе, возвращают такие же значения, если не оговорено иначе.]

# EXAMPLE

```
#include <iostream>
#include <string>
using namespace std;
int main (){
    string str ("There are two needles in this haystack with needles.");
    string str2 ("needle");
    size_t found;
    found=str.find(str2);
    if (found!=string::npos)
        cout << "first 'needle' found at: " << int(found) << endl;
    found=str.find("needles are small",found+1,6);
    if (found!=string::npos)
        cout << "second 'needle' found at: " << int(found) << endl;
    found=str.find("haystack");
    if (found!=string::npos)
        cout << "'haystack' also found at: " << int(found) << endl;
    found=str.find('.');
    if (found!=string::npos)
        cout << "Period found at: " << int(found) << endl;
    return 0;}

```

---

first 'needle' found at: 14  
second 'needle' found at: 44  
'haystack' also found at: 30  
Period found at: 51

# Тема – функция поиска ПОСЛЕДНЕГО ВХОЖДЕНИЯ

## ПОДСТРОКИ В СТРОКУ

- `size_t rfind ( const string& str, size_t pos = npos ) const;`
- `size_t rfind ( const char* s, size_t pos, size_t n ) const;`
- `size_t rfind ( const char* s, size_t pos = npos ) const;`
- `size_t rfind ( char c, size_t pos = npos ) const;`

`rfind` производит в строке обратный поиск. Если подстрока найдена, возвращается индекс позиции.

```
#include <iostream>
#include <string>
using namespace std;
int main (){
    string str ("The sixth sick sheik's sixth sheep's sick.");
    string key ("sixth");
    size_t found;
    found=str.rfind(key);
    if (found!=string::npos)
        str.replace (found,key.length(),"seventh");
    cout << str << endl;
    return 0;}          ////The sixth sick sheik's seventh sheep's sick.
```

# find\_first\_of

- `size_t find_first_of ( const string& str, size_t pos = 0 ) const;`
- `size_t find_first_of ( const char* s, size_t pos, size_t n ) const;`
- `size_t find_first_of ( const char* s, size_t pos = 0 ) const;`
- `size_t find_first_of ( char c, size_t pos = 0 ) const;`

Функция поиска первого вхождения в строку любого символа из подстроки. Поиск производится с начала строки.

```
#include <iostream>
#include <string>
using namespace std;
int main (){
    string str ("Replace the vowels in this sentence by asterisks.");
    size_t found;
    found=str.find_first_of("aeiou");
    while (found!=string::npos){
        str[found]='*';
        found=str.find_first_of("aeiou",found+1);}
    cout << str << endl;
    return 0;}    /////R*pl*c* th* v*w*ls *n th*s s*nt*nc* by *st*r*sks.
```

# find\_last\_of

- `size_t find_last_of ( const string& str, size_t pos = npos ) const;`
- `size_t find_last_of ( const char* s, size_t pos, size_t n ) const;`
- `size_t find_last_of ( const char* s, size_t pos = npos ) const;`
- `size_t find_last_of ( char c, size_t pos = npos ) const;`

Функция поиска последнего вхождения в строку любого символа из подстроки. Поиск производится с конца строки.



# find\_first\_not\_of

- `size_t find_first_not_of ( const string& str, size_t pos = 0 ) const;`
- `size_t find_first_not_of ( const char* s, size_t pos, size_t n ) const;`
- `size_t find_first_not_of ( const char* s, size_t pos = 0 ) const;`
- `size_t find_first_not_of ( char c, size_t pos = 0 ) const;`

Функция поиска в строке первого символа не из подстроки. Поиск производится с начала строки.

```
#include <iostream>
#include <string>
using namespace std;
int main (){
    string str ("look for non-alphabetic characters...");
    size_t found;
    found=str.find_first_not_of("abcdefghijklmnopqrstuvwxy z");
    if (found!=string::npos){
        cout << "First non-alphabetic character is " << str[found];
        cout << " at position " << int(found) << endl;
    }
    return 0;}    //// First non-alphabetic character is - at position 12
```

# find\_last\_not\_of

- `size_t find_last_not_of ( const string& str, size_t pos = npos ) const;`
- `size_t find_last_not_of ( const char* s, size_t pos, size_t n ) const;`
- `size_t find_last_not_of ( const char* s, size_t pos = npos ) const;`
- `size_t find_last_not_of ( char c, size_t pos = npos ) const;`

Функция поиска в строке первого символа не из подстроки. Поиск производится с конца строки.

```
#include <iostream>
#include <string>
using namespace std;
int main (){
    string str ("erase trailing white-spaces \n");
    string whitespaces (" \t\n");
    size_t found;
    found=str.find_last_not_of(whitespaces);
    if (found!=string::npos)
        str.erase(found+1);
    cout << "" << str << "" << endl;
    return 0;}          ////"erase trailing white-spaces"
```

# substr – ВЫДЕЛЕНИЕ ПОДСТРОКИ

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

Возвращает строку содержащую подстроку текущего объекта. Эта подстрока последовательность символов, которая начинается в позиции символа *pos* и имеет длину *n* символов.

```
#include <iostream>
#include <string>
using namespace std;
int main (){
    string str="We think in generalities, but we live in details.";
    string str2, str3;
    size_t pos;
    str2 = str.substr (12,12);    // "generalities"
    pos = str.find("live");      //позиция "live" в str
    str3 = str.substr (pos);     // получить "live" в конце
    cout << str2 << ' ' << str3 << endl;
    return 0;}  //// generalities live in details.
```