



# Семинар 9

## Строки в Си++



# Строки в Си++

Строки в Си – массивы символов с элементами типа `char`.

Строки в Си++ – экземпляры типа `string`.

Для работы с классом `string` нужно подключить заголовок `<string>`:  
`#include <string>`



# Строки в Си++

Основные отличия Си-строк от строк в стиле Си++:

1. Си-строка не может динамически изменять размеры
2. В функциях для работы с Си-строками нет контроля за границами



# Строки в Си++

В большинстве из методов класса `string` используются параметры целочисленного беззнакового типа `string::size_type`. При неудачном поиске эти методы возвращают значение `string::npos`.



# Строки в Си++

## 1. Конструкторы

// Формирует пустую строку в стиле Си++  
`string();`

`string(const char *, size_type);`

// Формирует объект-строку в стиле Си++.

Первый аргумент – строка в стиле Си. Второй аргумент – число, ограничивающее количество символов второго аргумента.



# Строки в Си++

## 1. Конструкторы

// Конструктор копирования  
`string(const string &);`

`string(const string &, size_type pos=0, size_type nPos=npos);`

// Конструктор копирования части строки,  
заданной первым параметром. `size_type pos` –  
начало копируемой подстроки. `size_type nPos` –  
количество копируемых символов.



# Строки в Си++

## 1. Конструкторы

// Конструктор, который в создаваемую строку  
n раз помещает символ ch.

```
string(size_type n, char ch);
```



# Строки в Си++

## 1. Конструкторы

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

```
void main()
{
    string str1("Hello, world!", 5); // Скопировать в str1 первые 5 символов
    cout << str1 << endl;
    string str2(str1); // Конструктор копирования
    cout << str2 << endl;
    string str3(string("Hello, world!"), 5); // Скопировать часть строки с 5-го
    // индекса
    cout << str3 << endl;
    string str4(3, '!'); // Создать строку из трёх символов '!'
    cout << str4 << endl;
}
```

Вывод:  
Hello  
Hello  
, world!  
!!!





# Строки в Си++

## 2. Операции над строками

Присваивание

=

строка = строка

строка = с\_строка

строка = символ



# Строки в Си++

## 2. Операции над строками

Конкатенация

+

строка + строка

строка + си\_строка

строка + символ

си\_строка + строка

символ + строка



# Строки в Си++

## 2. Операции над строками

Конкатенация с присваиванием  
+=

строка += строка

строка += си\_строка

строка += СИМВОЛ

```
string str1("Hello");
```

```
string str2("world");
```

```
cout << str1 + ", " + str2 + '!';
```



# Строки в Си++

## 2. Операции над строками

Операции сравнения

== сравнение на равенство

!= сравнение на неравенство

<

<=

>

>=

Каждая операция в качестве операндов принимает строки в стиле Си или строки в стиле Си++



# Строки в Си++

## 2. Операции над строками

```
#include <iostream>
#include <string>
using namespace std;
#define PRINT(x) cout<<#x" = "<<(x)<<endl;
void main()
{
    string str1("Hello");
    string str2("hello");
    string str3("HELLO");
    string str4("HELLOWORLD");
    PRINT(str1<str2);
    PRINT(int('H'));
    PRINT(int('h'));
    PRINT(str1<str3);
    PRINT(int('e'));
    PRINT(int('E'));
    PRINT(str3<str4);
}
```

Вывод:

```
str1<str2 = 1
int('H') = 72
int('h') = 104
str1<str3 = 0
int('e') = 101
int('E') = 69
str3<str4 = 1
```



# Строки в Си++

## 2. Операции над строками

Ввод и вывод из/в стандартный выходной  
ПОТОК

<<

>>



# Строки в Си++

## 3. Методы, заменяющие операции

- Доступ к символу по его индексу  
`char & at(size_type k);`

Применение этого метода аналогично применению операции `[]`. Метод обеспечивает контроль за диапазоном аргумента. При выходе за пределы строки генерируется исключение `out_of_range`.



# Строки в Си++

## 3. Методы, заменяющие операции

```
#define PRINT(x) cout<<#x" = "<<(x)<<endl;
```

```
int main()
```

```
{
```

```
    try{
```

```
        string str("Hello");
```

```
        PRINT(str[4]);
```

```
        PRINT(str.at(4));
```

```
        PRINT(str.at(5));
```

```
    }
```

```
    catch(...)
```

```
    {
```

```
        cout<<"Out of range";
```

```
    }
```

```
}
```





# Строки в Си++

## 3. Методы, заменяющие операции

Пример. Простейшая программа для подсчёта количества слов, разделённых пробелами.

```
string str("Lorem ipsum dolor sit amet, consectetur adipiscing elit... ");
unsigned count=0, i=0;
while(i<str.size()-1){
    if(str.at(i)!=' ' &&
        str.at(i+1)==' ')
        count++;
    i++;
}
if(str.at(str.size()-1)!=' ')
    cout<<"Words: "<<count+1;
else
    cout<<"Words: "<<count;
```



# Строки в Си++

## 3. Методы, заменяющие операции

- Конкатенация

`string & append(параметры);`

Метод конкатенации заменяет операцию `+`.  
При этом вызывающая метод строка  
изменяется.

Объявления функции `append`:

<http://www.cplusplus.com/reference/string/string/append/>



# Строки в Си++

## 3. Методы, заменяющие операции

```
string name, surname;  
cout<<"Enter name: ";  
cin>>name;  
cout<<"Enter surname: ";  
cin>>surname;  
cout<<"Hello, "+name.append(" ").append(surname);
```



# Строки в Си++

## 3. Методы, заменяющие операции

- Присваивание

`string & assign(параметры);`

Метод заменяет содержимое вызывающей строки на последовательность символов, определяемую параметрами.

Объявления функции `assign`:

<http://www.cplusplus.com/reference/string/string/assign/>



# Строки в Си++

## 3. Методы, заменяющие операции

```
string fullname("Aluminium");  
string str;  
str.assign(fullname, 3, 4);  
cout << str;
```



# Строки в Си++

## 4. Размеры строк (определить/задать)

Для определения размеров строк используется два метода:

```
size_type size()
```

```
size_type length()
```

```
string str("Aluminium");
```

```
PRINT(str.length());
```

```
// Вывод: str.length() = 9
```



# Строки в Си++

## 4. Размеры строк (определить/задать)

Для определения количества символов, которые можно поместить в строку без выделения дополнительной памяти:

```
size_type capacity()
```



# Строки в Си++

## 4. Размеры строк (определить/задать)

Если во время выполнения программы для некоторой строки выполняется соотношения `s.size()==s.capacity()`, то следующее увеличение строки `s` приведёт к автоматическому перераспределению памяти. Всегда выполняется соотношение `s.size()<=s.capacity()`





# Строки в Си++

## 4. Размеры строк (определить/задать)

Для определения предельной максимальной ёмкости строки:

```
size_type max_size()
```

Пример для онлайн-компилятора `cpp.sh`:

```
string str("Aluminium");  
PRINT(str.max_size());  
// Вывод: str.max_size() = 4611686018427387897
```



# Строки в Си++

## 4. Размеры строк (определить/задать)

Для изменения размеров строки:

`void resize(size_type n, char ch)`

`n < s.size()`: в строке остаются только первые `n`  
СИМВОЛОВ

`n > s.size()`: строка увеличивается до размера `n`, в  
её конец дописываются символы `ch`.



# Строки в Си++

## 4. Размеры строк (определить/задать)

Для изменения размеров строки существует перегруженный метод:

```
void resize(size_type n)
```

В этом случае для заполнения используется стандартный символ.



# Строки в Си++

## 4. Размеры строк (определить/задать)

Пример. Даны слова, перечисленные через пробел. Оставить в строке только первое слово.

```
string str("Lorem ipsum dolor sit amet");
int i=0;
bool flag=true;
while(i<str.length() && flag){
    if(str.at(i)==' ') flag=false;
    i++;
}
str.resize(--i);
cout<<str;
```



# Строки в Си++

## 5. Вставка символов

`string & insert(параметры);`

Объявления функции `insert`:

<http://www.cplusplus.com/reference/string/string/insert/>

Например,

`string& insert (size_t pos, const string& str, size_t subpos, size_t sublen);`

вставляет в вызывающую строку с позиции `pos` подстроку `str` длиной `sublen`, начинающуюся с позиции `subpos`



# Строки в Си++

## 5. Вставка символов

Пример.

```
string str("ABCDEF");  
string str2("123456");  
str.insert(3, str2, 1, 4);  
cout << str;
```

//Вывод: ABC2345DEF



# Строки в Си++

## 6. Удаление части строки

```
string & erase(int pos, int n=npos);
```

Удаляет с позиции pos количество символов n.

```
string str("ABCDEF");  
str.erase(2,2);  
cout<<str;  
// Вывод: ABEF
```



# Строки в Си++

## 7. Замена символов

`string & replace(параметры);`

Объявления функции `replace`:

<http://www.cplusplus.com/reference/string/string/replace/>

Например,

`string& replace (size_t pos, size_t len, const string& str, size_t subpos, size_t sublen);`

Заменяет в вызывающей строке `len` символов, начиная с позиции `pos`, на `sublen` символов строки `str`, начиная с позиции `subpos`.





# Строки в Си++

## 7. Замена символов

Пример.

```
string s1("0123456789");  
string s2("qwertyuiop");  
s1.replace(2, 3, s2, 1, 5);  
cout<<"s1="<<s1;  
//Вывод: 01werty56789
```



# Строки в Си++

## 8. Поиск строки или символов

`size_type find(параметры);`

Объявления функции `find`:

<http://www.cplusplus.com/reference/string/string/find/>

Например,

`string& replace (size_t pos, size_t len, const string& str, size_t subpos, size_t sublen);`

Заменяет в вызывающей строке `len` символов, начиная с позиции `pos`, на `sublen` символов строки `str`, начиная с позиции `subpos`.



# Строки в Си++

## 8. Поиск строки или символов

Пример. Простейшая замена слов в строке.

```
string str("one two three four one two ");
string::size_type i;
i=str.find("one",0);
while(i!=string::npos){
    str.replace(i, 3, string("odin"));
    i=str.find("one",0);
}
cout << str;
```



# Строки в Си++

## 8. Поиск строки или символов с конца

`size_type rfind(параметры);`

Объявления функции `rfind`:

<http://www.cplusplus.com/reference/string/string/rfind/>



# Строки в Си++

## 9. Выделение подстроки

```
string substr(size_type beg, size_type n=npos);
```

Начиная с позиции `beg` выделяет из вызывающей строки `n` символов. Вызывающая строка остаётся неизменной.

Эту функцию вызывают следующим образом:  
`newstr = oldstr.substr(...);`



# Строки в Си++

## 9. Поиск любого символа из заданных

```
size_type find_first_of(параметры);  
size_type find_last_of(параметры);  
size_type find_first_not_of(параметры);  
size_type find_last_not_of(параметры);
```

Пример:

```
string str("123456789123456789");  
string::size_type first = str.find_first_of("5463");  
cout<<"first="<<first<<endl; // first=2  
string::size_type last = str.find_last_of("5463");  
cout<<"last="<<last<<endl; // last=14
```



# Строки в Си++

## 10. Обмен значений строк

```
void swap(string & str2);
```

Вызов:

```
str1.swap(str2);
```

Строки `str1` и `str2` поменяются своими значениями.



# Строки в Си++

## 11. Получение строк в стиле Си

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

Получает строку в стиле Си и дописывает в качестве последнего символа '\0'.

```
const char * data();
```

Получает строку в стиле Си и не дописывает в конец '\0'.





# Строки в Си++

## 12. Копирование символов в Си-строку

```
size_type copy(char* str, size_type beg, size_type len=0);
```

Из вызывающей строки, начиная с позиции `beg`, выбираются `len` символов и копируются в символьный массив `str`. Этот метод не добавляет в массив символ `'\0'`, в случае необходимости его нужно добавить самостоятельно.



# Строки в Си++

## 12. Копирование символов в Си-строку

Пример.

```
string s("123456789123456789");  
char * ptr = new char[s.length()];  
s.copy(ptr, s.length());  
ptr[s.length()] = '\\0';  
cout<<ptr<<endl;
```

```
C:\Windows\system32\cmd.exe  
123456789123456789  
Для продолжения нажмите любую клавишу . . .
```

При неверном выделении памяти  
и отсутствии последнего символа:

```
C:\Windows\system32\cmd.exe  
5 ]♦♀@Δ+89123456789#####тФ?  
Для продолжения нажмите любую клавишу . . .
```



# Строки в Си++

## 13. Сравнение строк

`int compare(параметры);`

0 – строки совпадают,

>0 – последовательность из вызывающей строки больше последовательности, указанной в параметрах

<0 – последовательность из вызывающей строки меньше.

Объявления функции `compare`:

<http://www.cplusplus.com/reference/string/string/compare/>



# Строки в Си++

## 13. Сравнение строк

Например,

```
string s1("123");
```

```
string s2("6789");
```

```
cout << s1.compare(s2); // Напечатает -1.
```

# Задания



1. На входе имеется текст, содержащий слова, разделённые пробелами и символами табуляции, причём между двумя словами может быть один или несколько пробелов. Сформировать новую строку, в которой все слова будут отделены символами '\n'.
2. Из заданной строки сформировать новую строку таким образом, чтобы между любыми двумя символами исходной строки в новой строке были вставлены пробелы.
3. Доработать код на стр. 37 («простейшая замена слов в строке») таким образом, чтобы, например, слово «one» можно было заменить на «oneone».
4. В заданной строке содержатся слова, разделённые пробелами. Слова могут состоять из символов алфавита и цифр. Подсчитать количество слов, состоящих только из цифр.