

РАЗРАБОТКА НА C++

Урок 6. Строки

Что такое строка в C++?

Строки - это последовательность элементов типа `char` (символ), обычно строки называют “массив символов”.

Для различия строк и символов в C++ используются одинарные и двойные кавычки (так как в `string` может храниться один символ, а в `char` специальные символы, например `\n`, `\t`, `\r` и т.д.)

```
char c = 'q';
```

```
std::string str = "qqq";
```

Использование строк в проекте

В C++, библиотеки имеют зависимости между собой. Например, `<iostream>` включает `<string>`, поэтому при включении `<iostream>` вы автоматически получаете доступ к функциональности `std::string`.

Однако лучшей практикой всегда является явное подключение необходимых заголовочных файлов, чтобы ваш код был более читаемым, понятным и надежным.

Реализация строк находится в `std`, поэтому без подключения пространства имен нужно писать `std::string`

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

int main()
{
    string s1 = "Hello";

    cout << s1 << endl;
}
```

Как можно создать строку?

Строки в C++ не просто тип данных, а полноценная структура, которая имеет большую функциональность.

Строки можно создавать различными способами, но избегайте типичных ошибок, например:

```
string h1 = Hello; //ошибка, нет кавычек
string h2 = 'Hello'; //ошибка, не те кавычки
string h3 = 15; //ошибка, число не строка
String h4; //ошибка, нужна маленькая s
```

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

int main()
{
    string s1; //пустая строка ""
    string s2 = "Hello";
    string s3("World");
    string s4(3, '!'); //строка "!!!"
    string s5 = s4;
    string s6 = {'X', 'Y', 'Z'}; // "XYZ"

    cout << s2 << s1 << s3 << s4;
    cout << '\n' << s5 << '\n' << s6;
}
```

Операции над строками

- сравнения: ==, !=, >, >=, <, <=
- присваивания: =
- конкатенация: +
- присваивания с конкатенацией: +=
- индексация: []
- ВВОД: >>
- ВЫВОД: <<

Конкатенация строк — так называется операция объединения двух строк (также можно объединить строку и символ)

Присваивание - установка значения для переменной

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

int main()
{
    string s1 = "Hello";
    string s2("World");
    string s3 = s1 + ' ' + s2;
    cout << s3;
}
```

Операции над строками

Операций вычитания, умножения и деления для строк не существует.

В будущем мы познакомимся с механизмом “перегрузка операторов” для того чтобы создавать такие операции в наших проектах.

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

int main()
{
    string s1 = "Hello";
    string s2("World");
    string s3 = s1 - s2; //ошибка!
    string s4 = s1 * 2; //ошибка!
    string s5 = s1 + 2; //ошибка! строка+число
}
```

Операции над строками

Строки могут быть очень большими.
Количество символов в строке
ограничено только количеством памяти,
которое выделяется компьютером при
выполнении программы.

В большинстве практических случаев вы
можете хранить очень большие строки,
не беспокоясь о фиксированных
ограничениях, как в других типах данных.

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

int main()
{
    string s1 = "Hello";
    string s2("World!");
    string s3;
    for (int i=1;i<1000000;i++)
    {
        s3 += s1 + ' ' + s2 + ' ';
    }
    cout << s3; //ооочень большая строка
}
```

Операции над строками

Строки можно сравнивать. Равенство достигается при полном совпадении всех символов

Очень важно! Большей считается та строка, что идёт позже при алфавитном порядке

“Ivan” > “Andrey” => true

“Ivan” == “Anna” => false

“Ivan” == “Ivak” => false

“Ivan” == “Ivana” => false

Каждый символ в строке это тип char, а как мы знаем char кодируется в таблицах ASCII

символ с кодом 103(‘g’) > символ с кодом 37(‘%’)

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

int main()
{
    string s1 = "Hello";
    string s2("World!");
    string s3 = "Hello";

    bool eq = s1 == s2; //false
    bool eq1 = s1 == s3; //true
    bool eq2 = s1 < s2; //true

    cout << eq2;
}
```


Операции над строками

Оператор точка позволяет нам использовать методы различных структур, например:

```
s1.size() //узнать количество символов строки (число)
```

```
s1.empty() //пустая ли строка (true или false)
```

Т.е. мы буквально говорим “у этой строки вызвать этот функционал”

```
переменная.метод()
```

Все методы которые мы вызываем обязательно заканчиваются скобками.

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

int main()
{
    string s1 = "Hello";
    cout << s1.size() << '\n';
    string s2;
    cout << s2.empty();
}
```

Методы строки

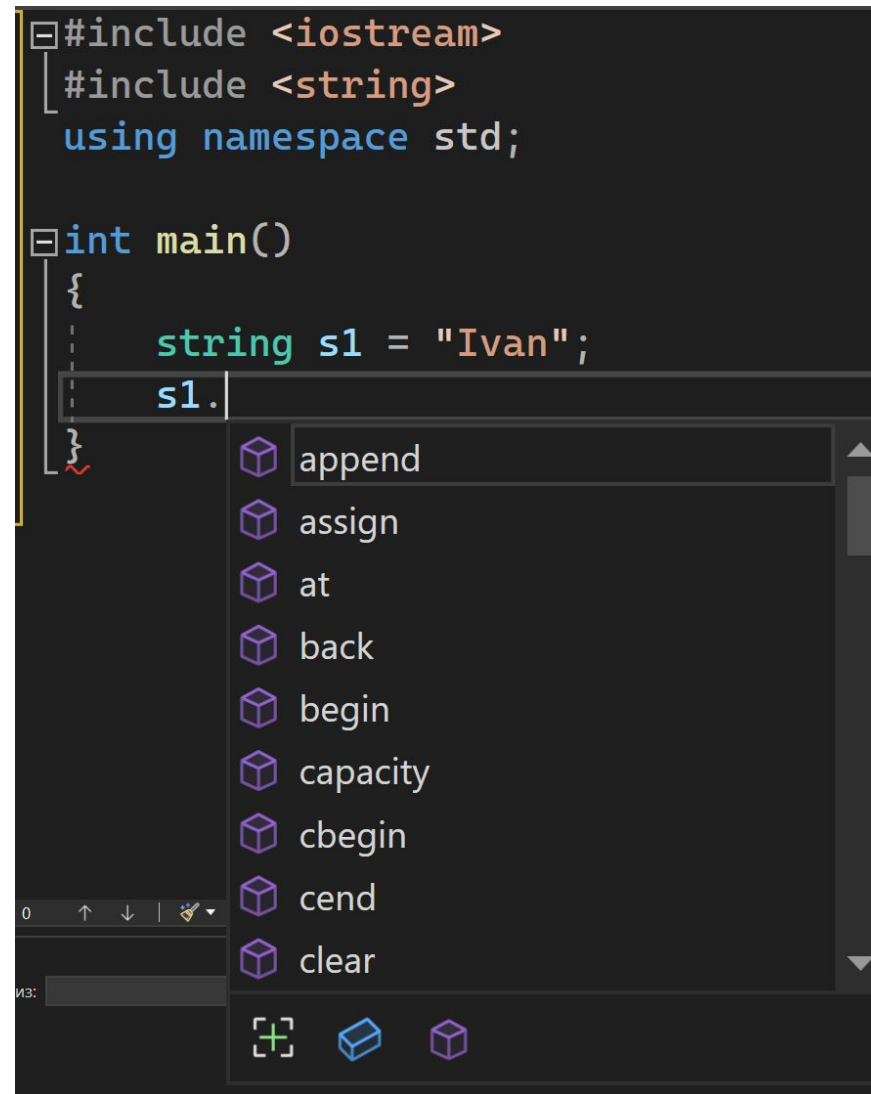
Когда мы используем оператор точка почти в любой IDE может возникнуть подсказка: Какие методы есть у данного объекта?

Эту подсказку можно вызвать горячими клавишами Ctrl+Пробел

Все эти методы созданы разработчиками C++ и выполняют четко определенные задачи.

Некоторые методы позволяют узнать какие-нибудь свойства объекта (например для строки: размер, последний символ, пустая ли строка)

Но есть также методы, которые изменяют объект (например для строки: удалить символы, очистить строку, добавить символ в середину строки)



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

int main()
{
    string s1 = "Ivan";
    s1.
```

The screenshot shows a C++ IDE with a code completion menu open for the string variable `s1`. The menu lists the following methods: `append`, `assign`, `at`, `back`, `begin`, `capacity`, `cbegin`, `cend`, and `clear`. At the bottom of the menu, there are icons for adding new entries, a box, and a cube.

Чтение строки с консоли

при обычно чтении с помощью `cin` в переменную типа `string` будет записана строка до первого пробела (так можно например ввести ФИО в разные строки)

для получения всей строки (учитывая пробелы) используется метод `getline()`

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

int main()
{
    string s1;
    getline(cin, s1);
    cout << s1 << endl;
}
```

Консоль отладки Microsoft Visual Studio

```
Ivanov Ivan Ivanovich
Ivanov Ivan Ivanovich
```

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

int main()
{
    string s1;
    string s2;
    string s3;
    cin >> s1 >> s2 >> s3;
    cout << s1 << endl;
    cout << s2 << endl;
    cout << s3 << endl;
}
```

Консоль отладки Microsoft Visual Studio

```
Ivanov Ivan Ivanovich
Ivanov
Ivan
Ivanovich
```

Чтение строки с консоли

Если сначала вводили через `cin`, а потом через `getline()`, то может произойти следующая ситуация:

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

int main()
{
    string s1;
    string s2;
    cin >> s1; //введу "Hello World!" но сюда попадет только "Hello"
    getline(cin, s2); //сюда попадет " World!" автоматически
    cout << s1 << endl << s2 << endl;
}
```

Консоль отладки Microsoft Visual Studio

```
Hello world!
Hello
world!
```

Чтение строки с консоли

Чтобы избежать такой ситуации можно использовать метод `ignore()`, который очищает буфер ввода до символа новой строки (256 на скриншоте это случайное число, всё зависит от размера строки которую ввел пользователь)

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

int main()
{
    string s1;
    string s2;
    cin >> s1; //введу "Hello World!" но сюда попадет только "Hello"
    cin.ignore(256, '\n'); //очищаем 256 символов до переноса строки
    getline(cin, s2); //" World!" теперь потерян, но можно вводить дальше
    cout << s1 << endl << s2 << endl;
}
```

Консоль отладки Microsoft Visual Studio

```
Hello world!
New String!
Hello
New String!
```

Метод erase()

`s1.erase(индекс)` - удаляет из строки `s1` начиная с выбранного индекса и до конца строки.

`s1.erase(индекс, количество)` - удаляет из строки `s1` определенное количество символов, начиная с выбранного индекса

Во всех языках программирования индекс считается с 0, а не с 1 как мы привыкли в реально жизни

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

int main()
{
    string s1 = "0123456789";
    cout << s1 << endl;
    s1.erase(4);
    cout << s1 << endl;
}
```

Консоль отладки Microsoft Visual Studio

0123456789
0123

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

int main()
{
    string s1 = "0123456789";
    cout << s1 << endl;
    s1.erase(1, 5);
    cout << s1 << endl;
}
```

Консоль отладки Microsoft Visual Studio

0123456789
06789

Метод insert()

s1.insert(индекс, строка) - вставляет строку в указанный индекс

s1.insert(индекс, количество, символ) - вставляет данный символ столько раз сколько указано в количестве в указанный индекс

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

int main()
{
    string s1 = "0123456789";
    cout << s1 << endl;
    s1.insert(5, "Hello");
    cout << s1 << endl;
}
```

Выбрать Консоль отладки Microsoft Visual Studio

0123456789
01234Hello56789

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

int main()
{
    string s1 = "0123456789";
    cout << s1 << endl;
    s1.insert(5, 3, '!');
    cout << s1 << endl;
}
```

Консоль отладки Microsoft Visual Studio

0123456789
01234!!!56789

Метод insert()

`s1.insert(индекс_s1, строка_s2, индекс_s2, количество_s2)` - вставляет данную строку в указанный индекс, но сначала указываем начиная с какого символа и сколько символов нужно вставить из указанной строки

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

int main()
{
    string s1 = "Hello ";
    string s2 = "My name is Ivan";
    cout << s1 << endl;
    s1.insert(6, s2, 11, 4);
    cout << s1 << endl;
}
```

Консоль отладки Microsoft Visual Studio

Hello
Hello Ivan

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

int main()
{
    string s1 = "0123456789";
    cout << s1 << endl;
    s1.insert(5, "987654321", 3, 4);
    cout << s1 << endl;
}
```

Консоль отладки Microsoft Visual Studio

0123456789
01234654356789

Метод find()

s1.find(строка) - найти первое вхождение выбранной строки в строку s1 и вернуть его индекс

s1.find(строка, индекс) - найти первое вхождение выбранной строки в строку s1, начиная с выбранного индекса, и вернуть его индекс

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

int main()
{
    //string s1 = "012345"
    string s1 = "Hello ";
    cout << s1.find("l") << endl;
```

Консоль отладки Microsoft Visual Studio

2

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

int main()
{
    //string s1 = "0123456789"
    string s1 = "HelloHello";
    cout << s1.find("l", 5) << endl;
```

Консоль отладки Microsoft Visual Studio

7

Использование методов вместе

Вставить одну строчку ровно посередине другой:

```
int main()
{
    string s1 = "ByeBye";
    string s2 = "Hello";
    cout << "s1.size: " << s1.size() << endl;
    cout << "s1.size/2: " << s1.size() / 2 << endl;
    s1.insert(s1.size() / 2, s2);
    cout << "s2 in s1: " << s1 << endl;
}
```

Консоль отладки Microsoft Visual Studio

```
s1.size: 6
s1.size/2: 3
s2 in s1: ByeHelloBye
```

Использование методов вместе

Удалить всё начиная с символа k

```
int main()
{
    string s1 = "delovaya kolbasa";
    string s2 = "ne moloko";
    string s3 = "morkov'";
    cout << s1 << endl << s2 << endl << s3 << endl;
    s1.erase(s1.find('k'));
    s2.erase(s2.find('k'));
    s3.erase(s3.find('k'));
    cout << "_____" << endl;
    cout << s1 << endl << s2 << endl << s3 << endl;
}
```

Консоль отладки Microsoft Visual Studio

```
delovaya kolbasa
ne moloko
morkov'

delovaya.
ne molo.
mor.
найлены.
```