Строки – 1

Denis Bakin

std::string – особый вектор

- `std::string` — контейнер символов, похожий на `std::vector<char>`
- Многие методы вектора доступны: `size()`, `push_back()`, `pop_back()`, `resize()`
- Дополнительные методы для работы с текстом

Конкатенация строк

```cpp
#include <iostream>
#include <string>

int main() {
    std::string s = "Some string";

    s += ' ';           // добавляем символ (аналог push_back)
    s += "functions";   // добавляем строку в конец

    std::cout << s << "\n";  // Some string functions
}
```

- Оператор += работает и с `char`, и с `const char*`

Выделение подстроки: substr

```
std::string s = "Some string functions";

// substr(pos, len) -- подстрока длины len с позиции pos
std::string sub1 = s.substr(5, 6);   // "string"

// substr(pos) -- подстрока с позиции pos до конца
std::string sub2 = s.substr(12);     // "functions"
```

- Первый аргумент – начальная позиция (с 0)
- Второй аргумент – длина (необязательный)

Поиск в строке: find

```cpp
std::string s = "Some string functions";

size_t pos1 = s.find(' ');          // 4 (первый пробел)
size_t pos2 = s.find(' ', pos1 + 1); // 11 (следующий пробел)
size_t pos3 = s.find("str");        // 5 (позиция подстроки)
size_t pos4 = s.find("#");          // std::string::npos
```

- Возвращает позицию первого вхождения
- `std::string::npos` — если не найдено

Проверка результата find

```
std::string s = "Hello, world!";

size_t pos = s.find("world");

if (pos != std::string::npos) {
    std::cout << "Найдено на позиции " << pos << "\n";
} else {
    std::cout << "Не найдено\n";
}
```

- Всегда проверяйте результат на npos!

Вставка подстроки: insert

```
std::string s = "Some string functions";

// insert(pos, str) -- вставляет str перед позицией pos
s.insert(5, "std::");

std::cout << s << "\n";  // Some std::string functions
```

- Первый аргумент – позиция вставки
- Второй аргумент – вставляемая строка

Замена подстроки: replace

```
std::string s = "Some std::string functions";

// replace(pos, len, str) -- заменяет len символов с позиции pos
s.replace(0, 4, "Special");

std::cout << s << "\n";  // Special std::string functions
```

- Первый аргумент – начальная позиция
- Второй аргумент – длина заменяемой части
- Третий аргумент – новая подстрока

Удаление подстроки: erase

```
std::string s = "Special std::string functions";

// erase(pos, len) -- удаляет len символов с позиции pos
s.erase(8, 5);  // удаляем "std::"

std::cout << s << "\n";  // Special string functions
```

- Аналогичный синтаксис: позиция + длина

Проверка префикса и суффикса (C++20)

```cpp
#include <iostream>
#include <string>

int main() {
    std::string phrase;
    std::getline(std::cin, phrase);

    if (phrase.starts_with("hello")) {
        std::cout << "Greeting\n";
    }

    if (phrase.ends_with("bye")) {
        std::cout << "Farewell\n";
    }
}
```

- `starts_with` – проверка начала строки
- `ends_with` – проверка конца строки

Бесконечный ввод: концепция

- В Unix “everything is a file”
- Потоки ввода/вывода – как бесконечные файлы
- EOF (End Of File) – специальный символ конца
- В консоли: `Ctrl+D` (Unix)

Чтение до конца ввода

```
int a = 0, b = 0;

// Цикл продолжается, пока чтение успешно
while (std::cin >> a >> b) {
    // обработка a и b
}
```

- Оператор >> возвращает `true` при успешном чтении
- Цикл завершается при EOF или ошибке

Чтение строки целиком: getline

```cpp
#include <iostream>
#include <string>

int main() {
    std::string line;

    // Читает до символа '\n'
    std::getline(std::cin, line);

    std::cout << "Вы ввели: " << line << "\n";
}
```

- `std::getline` читает до конца строки
- Включает пробелы (в отличие от >>)

Разбор строки: stringstream

```cpp
#include <iostream>
#include <sstream>
#include <vector>

int main() {
    std::string line;
    std::getline(std::cin, line);  // "1 2 3 4 5"

    std::stringstream ss(line);
    std::vector<int> numbers;
    int num;

    while (ss >> num) {
        numbers.push_back(num);
    }
    // numbers = {1, 2, 3, 4, 5}
}
```

stringstream – поток из строки

- `std::stringstream` – поток, работающий со строкой
- Поддерживает оператор >> как `std::cin`
- Удобен для разбора строки на части

```
std::stringstream ss("42 3.14 hello");
int i;
double d;
std::string s;

ss >> i >> d >> s;
// i = 42, d = 3.14, s = "hello"
```

Полный пример: чтение по строкам

```cpp
#include <iostream>
#include <vector>
#include <sstream>

int main() {
    std::vector<int> numbers;
    std::string line;

    while (std::getline(std::cin, line)) {
        std::stringstream ss(line);
        int num;
        while (ss >> num) {
            numbers.push_back(num);
        }
    }
}
```

Сводка методов std::string

Метод	Описание
`+=`	Добавление символа/строки в конец
`substr(pos, len)`	Выделение подстроки
`find(str, pos)`	Поиск подстроки
`insert(pos, str)`	Вставка подстроки
`replace(pos, len, str)`	Замена подстроки
`erase(pos, len)`	Удаление подстроки