STRING

W C++ mamy dwa główne sposoby pracy z łańcuchami znaków:

1. **Tradycyjne tablice znaków (char[])** – starsza metoda, ale wciąż używana w niektórych przypadkach.
2. **Klasa std::string** – nowocześniejsze, wygodniejsze i bezpieczniejsze rozwiązanie.

Poniżej znajdziesz opis najważniejszych operacji na obiektach klasy std::string w C++.

---

1. Tworzenie i inicjalizacja std::string

```
#include <iostream>

#include <string>


int main() {
   std::string s1 = "Hello";     // Inicjalizacja bezpośrednia
   std::string s2("World");      // Konstruktor z `const char*`
   std::string s3(s1);          // Kopia innego stringa
   std::string s4(5, 'A');       // "AAAAA" (5 razy 'A')

   std::cout << s1 << " " << s2 << " " << s3 << " " << s4 << std::endl;
}
```

---

2. Podstawowe operacje na std::string

Operacja	Przykład	Opis
Konkatenacja (+)	s3 = s1 + s2;	Łączy dwa stringi
Dodawanie (+=)	s1 += "!!!";	Dopisuje tekst na końcu
Długość (size())	s1.length(); lub s1.size();	Zwraca liczbę znaków
Opróżnianie	s1.clear();	Usuwa zawartość stringa
Sprawdzanie pustego	s1.empty();	Zwraca true, jeśli string jest pusty
Dostęp do znaku	s1[0], s1.at(0);	Pobiera znak o danym indeksie

```
std::string s = "Hello";
std::cout << s.size() << " " << s[1] << " " << s.at(1) << std::endl;
```

---

3. Porównywanie łańcuchów

```
std::string s1 = "abc";
std::string s2 = "xyz";

if (s1 == s2) std::cout << "Równe";
if (s1 < s2) std::cout << "s1 jest mniejsze"; // Porównanie leksykograficzne
```

---

4. Modyfikacja stringa

Operacja	Przykład	Opis
Dodawanie (+=)	s += "ABC";	Dopisuje tekst
Wstawianie (insert)	s.insert(3, "XYZ");	Wstawia ciąg na pozycji
Zastępowanie (replace)	s.replace(0, 2, "Hi");	Zastępuje fragment
Usuwanie (erase)	s.erase(2, 3);	Usuwa fragment

```
std::string s = "abcdef";
s.insert(3, "XYZ");  // "abcXYZdef"
s.replace(0, 3, "Hi");  // "HiXYZdef"
s.erase(2, 3);  // "HiXdef"
```

---

5. Wyszukiwanie w stringu

Operacja	Przykład	Opis
Znajdź (find)	s.find("abc");	Zwraca pozycję podciągu
Znajdź od końca (rfind)	s.rfind("abc");	Szuka od końca
Czy zawiera (std::string::npos)	if (s.find("x") != std::string::npos)	Sprawdza, czy podciąg istnieje

```
std::string s = "Hello World";
size_t pos = s.find("World");

if (pos != std::string::npos) {
   std::cout << "Znaleziono na pozycji: " << pos << std::endl;
}
```

---

6. Podciągi i podział stringa

```
std::string s = "abcdef";
std::string sub = s.substr(2, 3);  // "cde"
```

Aby podzielić string na słowa (np. według spacji), można użyć std::stringstream:

cpp

KopiujEdytuj

```
#include <sstream>
#include <vector>
std::string text = "C++ jest fajny";
std::stringstream ss(text);
std::vector<std::string> words;
std::string word;
while (ss >> word) {
   words.push_back(word);
}
```

---

7. Konwersja liczb do stringa i odwrotnie

Konwersja	Przykład	Opis
Liczba → string	std::to_string(123);	Zwraca "123"
String → int	std::stoi("123");	Zwraca 123
String → float	std::stof("12.34");	Zwraca 12.34

```
int x = 42;
string s = to_string(x); // "42"

string str = "123";
int num = stoi(str); // 123
```

---

8. Zamiana wielkości liter

```
#include <algorithm>
#include <cctype>
std::string s = "AbC";
std::transform(s.begin(), s.end(), s.begin(), ::toupper); // "ABC"
std::transform(s.begin(), s.end(), s.begin(), ::tolower); // "abc"
```

---

9. Usuwanie białych znaków

```
#include <cctype>

std::string trim(const std::string &s) {
    size_t start = s.find_first_not_of(" \t\n");
    size_t end = s.find_last_not_of(" \t\n");
    return (start == std::string::npos) ? "" : s.substr(start, end - start + 1);
}
```

---

10. Iterowanie po znakach

```
std::string s = "text";

for (char c : s) {

    cout << c << " ";

}
```