|  |  |
| --- | --- |
|  | Języki i paradygmaty programowania – zjazd 4 |

# Przykłady

## String

Stringi w C++ są używane do przechowywania i manipulowania ciągami znaków. Są reprezentowane przez klasę std::string z biblioteki standardowej C++, co daje dostęp do wielu przydatnych funkcji i operacji na tekście.

### Inicjalizacja

#include <iostream>

#include <string>

int main() {

// Inicjalizacja pustego stringa

std::string pustyString;

// Inicjalizacja stringa z wartością

std::string imie = "Jan";

// Inicjalizacja stringa z powieloną wartością

std::string powielony(5, 'x');

// Wypisanie wartości stringów

std::cout << "Pusty string: " << pustyString << std::endl;

std::cout << "Imię: " << imie << std::endl;

std::cout << "Powielony string: " << powielony << std::endl;

return 0;

}

### Operacje na stringach

#include <iostream>

#include <string>

int main() {

std::locale::global(std::locale(""));

std::string imie = "Jan";

std::string nazwisko = "Kowalski";

// Łączenie stringów

std::string imie\_nazwisko = imie + " " + nazwisko;

std::cout << "Pełne imię i nazwisko: " << imie\_nazwisko << std::endl;

// długość stringu

std::cout << "Długość imienia: " << imie.length() << std::endl;

std::cout << "Długość imienia i nazwiska: " << imie\_nazwisko.length() << std::endl;

// Dostęp do znaku na konkretnej pozycji

char litera = imie[0];

std::cout << "Pierwsza litera: " << litera << std::endl;

// Zmiana znaku na konkretnej pozycji

imie[0] = 'H';

std::cout << "Nowe imię: " << imie << std::endl;

// Podciągu / substring - wybieranie części napisu

std::string podciag = imie\_nazwisko.substr(4, nazwisko.length());

std::cout << "Substring: " << podciag << std::endl;

// Wyszukiwanie wzorca

size\_t pozycja = imie\_nazwisko.find("Kowalski");

if (pozycja != std::string::npos) {

std::cout << "Znaleziono wzorzec na pozycji: " << pozycja << std::endl;

}

else {

std::cout << "Nie znaleziono wzorca." << std::endl;

}

std::string zdanie = "To jest przykładowe zdanie";

// Iteracja przez znaki

for (char znak : zdanie) {

std::cout << znak << " ";

}

std::cout << std::endl;

return 0;

}

## Pliki

W C++ do odczytu i zapisu plików tekstowych używamy [std::ifstream](https://en.cppreference.com/w/cpp/io/basic_ifstream) i [std::ofstream](https://en.cppreference.com/w/cpp/io/basic_ofstream). Istnieje jeszcze [std::fstream](https://en.cppreference.com/w/cpp/io/basic_fstream), ale na razie skupimy się na plikach tekstowych.

### Zapis

#include <fstream>

#include <iostream>

#include <string>

int main() {

std::ofstream file("output.txt"); // Otwarcie pliku do zapisu

if (file.is\_open()) {

file << "Przykładowa treść do zapisu w pliku.\n"; // Zapisanie do pliku

file.close(); // Zamknięcie pliku

}

else {

std::cerr << "Nie można otworzyć pliku do zapisu." << std::endl;

}

return 0;

}

### Odczyt

#include <fstream>

#include <iostream>

#include <string>

int main() {

std::locale::global(std::locale(""));

std::ifstream file("tu wpisz ścieżkę pliku"); // Otwarcie pliku do odczytu

if (file.is\_open()) {

std::string line;

while (std::getline(file, line)) {

std::cout << line << std::endl; // Wyświetlanie zawartości pliku

}

file.close(); // Zamknięcie pliku

}

else {

std::cerr << "Nie można otworzyć pliku." << std::endl;

}

return 0;

}

### Wczytywanie z pliku danych i sprawdzanie czy są liczbami

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

#include <iomanip> // Dla setprecision

#include <algorithm> // all\_of

bool isFloatingPoint(const std::string& s) {

size\_t dotCount = std::count(s.begin(), s.end(), '.'); // Liczenie wystąpień kropki w ciągu znaków

return !s.empty() && std::all\_of(s.begin(), s.end(), [](char c) { // Sprawdzenie wszystkich znaków w ciągu; "[](char c)" to początek definicji lambdy (funkcji anonimowej)

return std::isdigit(c) || c == '.'; // Sprawdzenie czy znak jest cyfrą lub kropką

}) && dotCount <= 1; // Sprawdzenie, czy jest co najwyżej jedna kropka

}

int main() {

std::ifstream file("data.txt");

if (!file.is\_open()) {

std::cerr << "Nie można otworzyć pliku." << std::endl;

return 1;

}

std::string line;

while (std::getline(file, line)) {

std::istringstream iss(line);

std::string word;

while (iss >> word) {

if (isFloatingPoint(word)) {

// Przetwarzanie jako liczba zmiennoprzecinkowa

double number = std::stod(word); // Konwersja na double

std::cout << std::fixed << std::setprecision(2); // Ustawienie precyzji

std::cout << "Liczba zmiennoprzecinkowa: " << number << std::endl;

}

}

}

file.close();

return 0;

}

# Zadania

1. Przeżyj dokumentacje stringa (np. [tu](https://cplusplus.com/reference/string/string/)) i sprawdź co robi funkcja replace oraz napisz kod, który pokaże działanie replace. Odnajdź jeszcze 2 inne funkcje których nie ma w przykładach i napisz jak one działają oraz napisz kod, który to pokaże. Odpowiedź może być w komentarzu kodu.
2. Napisz czym się różnią get, getline, read oraz wczytywanie za pomocą operatora >>. Napisz kod który pokaże różnice. Odpowiedź może być w komentarzu kodu.
3. Przenalizuj i uruchom z różnymi parametrami poniższy kod i dodaj mu komentarze wyjaśniające co robią poszczególne funkcje z filestream, tj. opisz co się dzieje w blokach kodu rozpoczętych komentarzem „// ???”. Zastąp znaki zapytania odpowiedzią. Kod:

#include <iostream>

#include <filesystem>

#include <fstream>

#include <locale>

namespace fs = std::filesystem;

int main(int argc, char\* argv[]) {

std::locale::global(std::locale(""));

if (argc != 3) {

std::cout << "Użycie: " << argv[0] << " <katalog> <plik>" << std::endl;

return 1;

}

// Pobieranie nazw katalogu i pliku z argumentów wiersza poleceń

fs::path directoryPath = argv[1];

fs::path filePath = argv[2];

// ???

if (!fs::exists(directoryPath)) {

if (fs::create\_directory(directoryPath)) {

std::cout << "Utworzono katalog: " << directoryPath << std::endl;

}

else {

std::cout << "Nie udało się utworzyć katalogu." << std::endl;

return 1;

}

}

else {

std::cout << "Katalog już istnieje." << std::endl;

}

// ???

std::ofstream outputFile(filePath);

if (outputFile.is\_open()) {

outputFile << "To jest zawartość pliku." << std::endl;

outputFile.close();

std::cout << "Utworzono plik: " << filePath << std::endl;

}

else {

std::cout << "Nie udało się utworzyć pliku." << std::endl;

return 1;

}

// ???

fs::path copiedFilePath = directoryPath / "skopiowany\_plik.txt";

try {

fs::copy\_file(filePath, copiedFilePath, fs::copy\_options::overwrite\_existing);

std::cout << "Skopiowano plik do: " << copiedFilePath << std::endl;

}

catch (const fs::filesystem\_error& e) {

std::cout << "Błąd podczas kopiowania pliku: " << e.what() << std::endl;

return 1;

}

// ???

std::cout << "Zawartość katalogu:" << std::endl;

for (const auto& entry : fs::directory\_iterator(directoryPath)) {

std::cout << entry.path().filename() << std::endl;

}

// ???

std::cout << "Atrybuty pliku " << filePath << ":" << std::endl;

if (fs::exists(filePath)) {

std::cout << "Rozmiar: " << fs::file\_size(filePath) << " bajtów" << std::endl;

std::cout << "Czas ostatniej modyfikacji: " << fs::last\_write\_time(filePath).time\_since\_epoch().count() << std::endl;

}

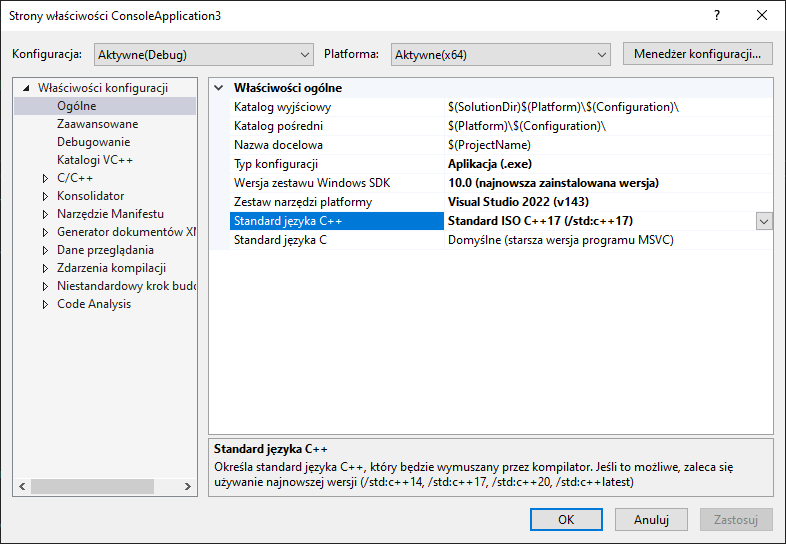
else {

std::cout << "Plik nie istnieje." << std::endl;

}

return 0;

}

Jeżeli będzie problem z filesystem to należy zmienić ustawienia kompilatora tak, by używał standardu C++17 lub wyższego. Dla Visual Studio jest to kwestia ustawień we właściwościach projektu. 

1. Przeżyj dokumentację filestream na <https://en.cppreference.com/w/cpp/filesystem> i odnajdź trzy możliwości jego użycia. Opisz je i napisz kod, który je zademonstruje.
2. Napisz program, który będzie wypisywał zawartość plików. Nazwy plików powinny być podane jako argumenty wywołania. Program powinien obsługiwać dowolną liczbę argumentów i wypisywać zawartość plików jeden po drugim.
3. Napisz program, który wczyta wszystkie liczby z pliku oraz poda dla nich średnią i sumę.
4. Napisz program, który będzie przyjmował nazwę pliku jako argument a następnie zliczał dla niego liczbę znaków, liczbę słów oraz liczbę linii i wyświetli te zliczone dane.
5. Napisz program, który będzie wczytywał plik i będzie każdą liczbę wypisywał oraz poda informację czy liczba jest parzysta czy nieparzysta, czy jest podzielna przez 2, 3, 5 i 9. Przykładowa zawartość pliku (można dopisać swoje przykłady, ale poniższe linie muszą być w pliku):

1 2 3 4 5 6 7 8 9 10 213 134324 12

21321 455678 234234 1223345678

1234567890

912345678129 435435 234 5235 325345 3453453543543243 5235 235 32 5345345345354543453

9999999999999999999999999999999999999999999999999999999999999999999999999999990

9999999999999999999999999999999999999999999999999999999999999999999999999999995

9999999999999999999999999999999999999999999999999999999999999999999999999999991

9999999999999999999999999999999999999999999999999999999999999999999999999999992

9999999999999999999999999999999999999999999999999999999999999999999999999999993

1. Napisz program, który będzie wczytywał tablice z pliku (jedna linia na jedną tablicę) i będzie je sortował:
   1. Za pomocą algorytmu sortowania bąbelkowego (Bubble Sort) lub sortowania przez wstawianie (Insertion Sort) lub sortowania przez wybieranie (Selection Sort) – należy wybrać 1 z trzech i w implementacji użyć pętli. (1 punkt)
   2. Za pomocą sortowania szybkiego (Quick Sort) lub sortowania przez scalanie (Merge Sort) – należy wybrać jedno do implementacji i użyć rekurencji. (1 punkt)