Zadania

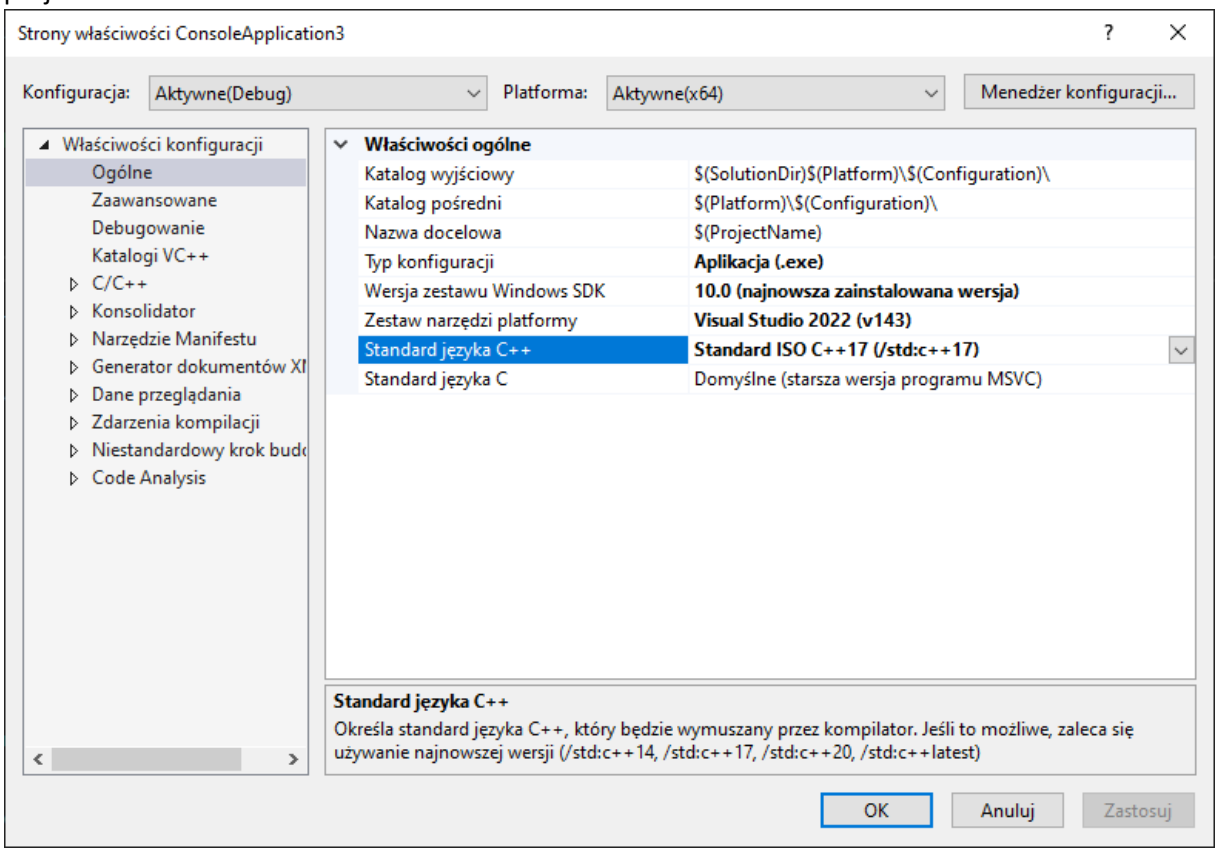
1. Przeżyj dokumentacje stringa (np. tu) i sprawdź co robi funkcja replace oraz napisz kod, który pokaże działanie replace. Odnajdź jeszcze 2 inne funkcje których nie ma w przykładach i napisz jak one działają oraz napisz kod, który to pokaże. Odpowiedź może być w komentarzu kodu.
2. Napisz czym się 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.



4. 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.
5. 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.
6. Napisz program, który wczyta wszystkie liczby z pliku oraz poda dla nich średnią i sumę.
7. 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.
8. 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

99999999999999999999999999999999999999999999999999999999999999999999999999999990

99999999999999999999999999999999999999999999999999999999999999999999999999999995

99999999999999999999999999999999999999999999999999999999999999999999999999999991

99999999999999999999999999999999999999999999999999999999999999999999999999999992
```

```
99999999999999999999999999999999999999999999999999999999999999999999999999999993
```

9. Napisz program, który będzie wczytywał tablice z pliku (jedna linia na jedną tablicę) i będzie je sortował:
    a. 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)
    b. Za pomocą sortowania szybkiego (Quick Sort) lub sortowania przez scalanie (Merge Sort) – należy wybrać jedno do implementacji i użyć rekurencji. (1 punkt)

Wybrane 1 zadanie 5, 6 lub 7 można zastąpić poniższym zadaniem:
Napisz w C++ dwie funkcje które będą implementowały szyfrowanie typu Cezar. Jedna powinna szyfrować a druga odszyfrowywać.