Programowanie I
Wykład 9

dr inż. Rafał Brociek

Wydział Matematyki Stosowanej
Politechnika Śląska

16.12.2019

Operacje wejścia/wyjścia na plikach

Do programu należy włączyć plik nagłówkowy biblioteki `fstream`.

```
#include<fstream>
```

Strumienie

- ofstream (output file stream) - zapis do plików,
- ifstream (input file stream) - odczyt z pliku,
- fstream (file stream) - zapis i odczyt jednocześnie.

Uwaga. Nazwy zadeklarowane w pliku `fstream` należą do przestrzeni nazw `std`, zatem użycie `using namespace std;` pozwala na niepisanie kwalifikatora zakresu `std::`

Zapis do pliku

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
        ofstream plik("tekst.txt");
        // tryb tekstowy domyślnie
        // nadpisuje zawartość pliku
        plik << "Wszyscy␣studenci␣";
        plik << "zalicza␣I␣semestr.";
        plik.close();
        plik.open("tekst.txt");
        plik << "Przepraszam,␣jednak␣nie␣wszyscy.";
        plik.close();
}
```

Tryby otwarcia

Tryb	Skrót od	Opis
in	input (wejście)	otwórz plik do czytania
out	output (wyjście)	otwórz plik do zapisania
ate	at end (na końcu)	otwórz plik i ustaw wskaźnik pliku na końcu
app	append (dołączenie)	otwórz plik i pozwól na dopisanie na jego końcu
trunc	truncate (odcięcie)	otwórz plik i skasuj jego zawartość
binary		zapis w trybie binarnym

```
fstream plik;
plik.open("proba.txt", ios::out | ios::app);
// sprawdzamy, czy plik został otwarty
if (plik.is_open())
{
        plik << "Napis ";
        plik.close();
}
else
        cout << "Nie można otworzyć pliku." << endl;
system("pause");
```

Odczyt z pliku (znak po znaku)

```
char z;
ifstream plik("tekst.txt");
if (plik.is_open())
{
        // sprawdzamy, czy wskaźnik pliku
        // znajduje się na końcu pliku
        while (!plik.eof())
        {
                plik.get(z);
                cout << z;
        }
}
else
        cout << "Nie mozna otworzyc plik." << endl;
plik.close();
cout << endl;
system("pause");
```

Odczyt z pliku (słowo po słowie)

```
// zapisujemy plik
fstream plik("tekst.txt", ios::out);
plik << "Taki sobie\n tekst \n z kilkoma";
plik << "\n liniami.";
plik.close();

// odczytujemy z pliku
string s;
plik.open("tekst.txt", ios::in);
if (plik.is_open()) {
        while (!plik.eof()) {
                plik >> s;
                cout << s << endl;
        }
}
else
        cout << "Nie mozna otworzyc plik." << endl;
plik.close();
system("pause");
```

Odczyt z pliku (linia po linii)

```
// zapisujemy plik
fstream plik("tekst.txt", ios::out);
plik << "Taki sobie\n tekst \n z kilkoma";
plik << "\n liniami.";
plik.close();

// odczytujemy z pliku
string linia;
plik.open("tekst.txt", ios::in);
if (plik.is_open()) {
        while (!plik.eof()) {
                getline(plik, linia);
                cout << linia << endl;
        }
}
else
        cout << "Nie mozna otworzyc plik." << endl;
plik.close();
system("pause");
```

Tryb binarny

Konieczny w przypadku systemów operacyjnych, które inaczej obchodzą się z plikami tekstowymi niż binarnymi np. system operacyjny Windows. Dla strumienia nie ma znaczenie co nim płynie, czy są to dane binarne czy tekstowe, ale ... w przypadku trybu tekstowego (domyślnego) dla systemu Windows:

- każdy wystawiony do strumienia znak '\n' zostanie zastąpiony przez dwa znaki '\r' '\n'
- w przypadku pobrania ze strumienia znaków '\r' '\n' zastąpione zostaną przez znak '\n'

`https://pl.wikipedia.org/wiki/CRLF`

Dane zapisane w trybie binarnym są w niezmienionej formie zapisywane do pliku, podczas gdy w trybie tekstowym, dane są modyfikowane, by pasowały do konwencji zapisu tekstu w danym środowisku np. w systemie Windows.

Tryb binarny

```
fstream plik;
plik.open("tekst.txt", ios::out);
plik << "AB\nCD";
plik.close();
plik.open("tekst.txt", ios::in | ios::binary);
char z;
if (plik.is_open())
        while (true) {
                plik.get(z); // z=plik.get();
                if (plik.eof()) break;
                cout << static_cast<int>(z) << ' ';
        }
else
        cout << "Nie można otworzyć pliku" << endl;
cout << endl;
plik.close();
```

```
// 65 66 10 67 68
```

Tryb tekstowy

```
fstream plik;
plik.open("tekst.txt", ios::out);
plik << "AB\nCD";
plik.close();
plik.open("tekst.txt", ios::in);
char z;
if (plik.is_open())
        while (true) {
                plik.get(z); // z=plik.get();
                if (plik.eof()) break;
                cout << static_cast<int>(z) << ' ';
        }
else
        cout << "Nie można otworzyć pliku" << endl;
cout << endl;
plik.close();
```

```
// 65 66 13 10 67 68
```

Ustawienie pozycji w pliku i jej określenie

Dla ifstream

```
plik.seekg(2, ios::beg); //2 znak od początku
plik.seekg(6, ios::cur); //6 znaków dalej
//od bieżącego
plik.seekg(-3, ios::end); //3 znaki przed końcem
plik.tellg(); //aktualna pozycja w pliku
```

Dla ofstream

```
plik.seekp(2, ios::beg); //2 znak od początku
plik.seekp(6, ios::cur); //6 znaków dalej
//od bieżącego
plik.seekp(-3, ios::end); //3 znaki przed końcem
plik.tellp(); //aktualna pozycja w pliku
```

Zastosowanie

Określenie liczby znaków w pliku do odczytu:

```
ifstream plik;
int n{};
plik.open("tekst.txt", ios::binary | ios::ate);
if (plik.is_open())
        n = plik.tellg();
else
        cout << "Nie mozna otworzyc pliku.";
cout << "liczba znakow: " << n << endl;
plik.close();
```

Zastosowanie

Określenie liczby znaków w pliku do odczytu (inny sposób):

```
ifstream plik;
int n{};
plik.open("tekst.txt", ios::binary);
if (plik.is_open())
{
        plik.seekg(0, ios::end);
        n = plik.tellg();
}
else
        cout << "Nie mozna otworzyc pliku.";
cout << "liczba znakow: " << n << endl;
plik.close();
```

Zapis/odczyt binarny bloku danych

Metody read, write

```
plik.read(wskaźnik, rozmiar),
plik.read(char* s, streamsize n),
plik.write(wskaźnik, rozmiar),
plik.write(const char* s, streamsize n).
```

```
char z;
// czytaj z pliku do zmiennej z
plik.read( &z , sizeof( z ));

const int n=10;
int tab[n]{};
// zapisz do pliku tablice tab
plik2.write( reinterpret_cast<char*>(tab), n*sizeof(int) );
```

Zapis binarny

```
ofstream zapis_do_plik;
// dane do zapisu do pliku
double tab[3] = { 1.1,2.2,3.3 };
zapis_do_plik.open("tab.bin", ios::out | ios::binary);
if (zapis_do_plik.is_open())
{
        // zapisanie bloku danych
        zapis_do_plik.write(reinterpret_cast<char*>(tab),
                3*sizeof(double));
}
else
        cout << "Nie mozna otworzyc pliku.";
zapis_do_plik.close();
```

Odczyt binarny

```
ifstream odczyt_z_plik;
odczyt_z_plik.open("tab.bin", ios::in
        | ios::binary | ios::ate);
// jaki rozmiar, ile ,,miejsca" musimy przygotowac
int rozmiar = odczyt_z_plik.tellg();
odczyt_z_plik.seekg(0, ios::beg);
// zarezerwowanie miejsca na dane
char* dane = new char[rozmiar];

if (odczyt_z_plik.is_open())
{
        odczyt_z_plik.read(dane, rozmiar);
        odczyt_z_plik.close();
        int rozmiar_tab_double = rozmiar / sizeof(double);
        double* odczytane = reinterpret_cast<double *>(dane);
        for (int i = 0; i < rozmiar_tab_double; i++)
                cout << odczytane[i] << endl;
}
else
        cout << "Nie mozna otworzyc pliku.";
```

Zadanie

Zadanie. Zdefiniuj strukturę `Osoba`, zawierającą pola:

- `string` imie,
- `string` nazwisko,
- `int` wiek,
- `void info()`

Stwórz obiekt tej struktury, zapisz ten obiekt do pliku `osoba.bin`, a następnie odczytaj ten obiekt z pliku do nowego obiektu struktura.

Rozwiązanie 1/3

Struktura:

```
struct Osoba
{
        string imie;
        string nazwisko;
        int wiek;

        void info();
};

void Osoba::info()
{
        cout << imie << " " << nazwisko;
        cout << ", lat: " << wiek << endl;
}

// w funkcji main tworzymy ,,kibica"
Osoba kibic{ "Janusz", "Znawca", 45 };
```

Rozwiązanie 2/3

Zapis do pliku:

```
fstream plik;
plik.open("osoba.bin", ios::out | ios::binary);
if (plik.is_open())
{
        plik.write(reinterpret_cast<char*>(&kibic),
                sizeof(kibic));
}
else
        cout << "Nie mozna otworzyc pliku." << endl;
plik.close();
```

Rozwiązanie 3/3

Odczyt z pliku:

```
plik.open("osoba.bin", ios::in
        | ios::binary | ios::ate);
int rozmiar = plik.tellg();
plik.seekg(0, ios::beg);
Osoba* nowaOsoba = new Osoba;

if (plik.is_open())
{
        plik.read(reinterpret_cast<char*>(nowaOsoba),
                rozmiar);
        plik.close();
        nowaOsoba->info();
}
```

Dziękuję za uwagę