Odczytywanie i zapisywanie plików

Wszystkie przedstawione do tej pory programy są względnie proste, czyli w sam raz dla kogoś, kto dopiero uczy się programować. Dzięki zdobytym podstawom języka C++ przy odrobinie praktyki będziesz w stanie pisać także bardziej zaawansowane programy. Brakuje Ci jednak jeszcze wiedzy na jeden ważny temat: pracy z plikami.

Umiesz już wyświetlać dane na ekranie i pobierać informacje od użytkownika z konsoli. Na pewno się zgodzisz, że interakcja z samą konsolą to trochę za mało. Pomyśl o takich programach, jak Notatnik czy środowisko programistyczne, którego używasz albo arkusz kalkulacyjny. Wszystkie one potrafią zapisywać i odczytywać pliki. Także w programowaniu gier jest to bardzo potrzebne. Oprócz plików do zapisywania aktualnego stanu gry w grach wykorzystywane są też pliki graficzne, wideo, muzyczne i inne. Mówiąc krótko, program który nie współpracuje z plikami ma poważnie ograniczoną funkcjonalność.

Zatem bierzemy się do pracy. Jeśli wiesz jak posługiwać się instrukcjami `cin` i `cout`, to bez trudu nauczysz się też działać na plikach.

Zapisywanie danych w pliku

Pracę z plikiem rozpoczyna się od jego otwarcia. Gdy plik jest otwarty, dalsze czynności wykonuje się tak samo, jak przy użyciu instrukcji `cin` i `cout`, tzn. za pomocą operatorów `<<` i `>>`. Uwierz mi, nauczysz się tego w mig.

Kiedy jest mowa o komunikacji programu z obiektami zewnętrznymi, używa się pojęcia **strumienia**. W tym rozdziale interesować nas będą **strumienie plikowe**, czyli strumienie umożliwiające zapis i odczyt plików.

Nagłówek fstream

Jak to zwykle bywa w języku C++, gdy potrzebujemy jakiejś funkcjonalności, musimy dołączyć do programu specjalny plik nagłówkowy. Narzędzia do pracy z plikami znajdują się w nagłówku `fstream`, który dołączamy na początku programu za pomocą dyrektywy `#include <fstream>`.

Znasz już nagłówek `iostream` zawierający narzędzia pozwalające wysyłać i pobierać dane z konsoli. Nazwa `iostream` jest skrótem od angielskich słów input/output stream oznaczających strumień wejścia i wyjścia. Natomiast nazwa `fstream` pochodzi od angielskich słów file stream oznaczających strumień plikowy.

Najważniejsza różnica między strumieniem wejścia i wyjścia a strumieniem plikowym polega na tym, że w tym drugim przypadku **każdy plik musi mieć utworzony osobny strumień**. Najpierw nauczymy się tworzyć strumienie wyjściowe, czyli umożliwiające odczyt danych z plików.

Otwieranie pliku do zapisu

Faktycznie strumienie są **obiektami**. Przypomnę, że język C++ jest językiem obiektowym i strumienie są po prosty jednymi z jego wielu obiektów.

Nie martw się, jeśli nie wiesz co to jest obiekt. Wszystkiego dowiesz się później, a na razie obiekty wyobrażaj sobie jako duże zmienne. Zawierają one wiele informacji o otwartych plikach i udostępniają różne funkcje, takie jak np. do zamykania plików, przechodzenia na ich początek itp.

Co ciekawe, strumienie plikowe definiuje się dokładnie tak samo, jak zwykłe zmienne. Po prostu definiujemy zmienną typu `ofstream` o wartości będącej ścieżką do pliku, który chcemy otworzyć:

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ofstream mojStrumien("C:/Nanoc/scores.txt");     // Deklaracja strumienia do zapisu w pliku
                                                     // C:/Nanoc/scores.txt

    return 0;
}
```

Ścieżkę do pliku należy podać w cudzysłowie. Można stosować dwa rodzaje ścieżek:

- **Ścieżka bezwzględna**: jest to ścieżka odnosząca się do katalogu głównego na dysku, np. C:/Nano/C++/Pliki/scores.txt
- **Ścieżka względna**: jest to ścieżka określająca położenie pliku względem miejsca, w którym znajduje się program, np. Pliki/scores.txt, jeśli program znajduje się w folderze C:/Nanoc/C++.

Teraz można użyć utworzonego strumienia do zapisania danych w pliku.

Jeśli plik o podanej nazwie nie istnieje, zostanie on automatycznie utworzony!

Najczęściej nazwa pliku jest określona jako łańcuch znaków. Wówczas do jego otwarcia należy użyć funkcji `c_str()`:

```
string const nazwaPliku("C:/Nanoc/scores.txt");

ofstream mojStrumien(nazwaPliku.c_str());     // Definicja strumienia do zapisu danych w pliku.
```

Podczas otwierania pliku mogą mieć miejsce niespodziewane zdarzenia, np. plik nie zostanie odnaleziony albo dysk twardy będzie zapełniony. Dlatego też zawsze przy otwieraniu pliku należy sprawdzić, czy operacja się powiodła. Do tego celu można użyć konstrukcji `if(mojStrumien)`. Jeśli wynik tego testu jest negatywny, wiadomo, że wystąpił jakiś problem i nie można użyć żądanego pliku.

```
ofstream mojStrumien("C:/Nanoc/scores.txt"); // Próbujemy otworzyć plik
```

```
if(mojStrumien)     // Sprawdzamy czy plik został otwarty.
{
     // Wszystko jest w porządku, można pracować na pliku
}
else
{
     cout << "BŁĄD: nie można otworzyć pliku." << endl;
}
```

Możemy w końcu zapisać jakieś dane w pliku. Jak się przekonasz, nie jest to wcale trudne.

Zapisywanie danych w pliku

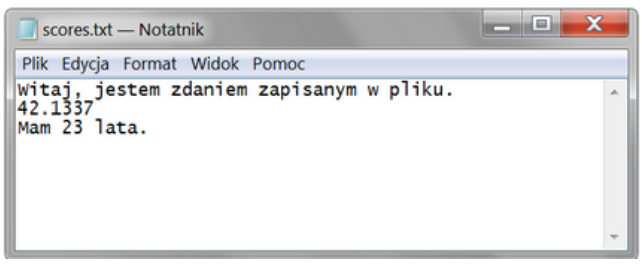
Napisałem, że z plikami pracuje się podobnie, jak ze strumieniami `cout` i `cin`. Tak jest rzeczywiście, ponieważ w przypadku zapisu do pliku używa się operatora <<:

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main()
{
     string const nazwaPliku("E:/scores.txt");
     ofstream mojStrumien(nazwaPliku.c_str());

     if(mojStrumien)
     {
          mojStrumien << "Witaj, jestem zdaniem zapisanym w pliku." << endl;
          mojStrumien << 42.1337 << endl;
          int age(23);
          mojStrumien << "Mam " << age << " lata." << endl;
     }
     else
     {
          cout << "BŁĄD: nie można otworzyć pliku." << endl;
     }
     return 0;
}
```

Po uruchomieniu tego programu na moim dysku znalazł się plik o nazwie scores.txt o następującej zawartości:



Wypróbuj go sam! Możesz na przykład napisać program pobierający od użytkownika imię i wiek i zapisujący te informacje w pliku tekstowym.

Tryby otwarcia pliku

Pozostała nam jeszcze jedna kwestia do wyjaśnienia. Co się dzieje, gdy plik o podanej nazwie już istnieje? Zostanie on wyczyszczony i zapisany od nowa, co nie zawsze jest tym, czego byśmy chcieli. Wyobraź sobie na przykład plik do przechowywania listy czynności wykonanych przez użytkownika. Nie chcemy za każdym razem kasować jego zawartości, tylko dodawać na jego końcu coraz to nowe wiersze danych.

Aby zapisać informacje na końcu pliku, należy wyrazić ten zamiar podczas jego otwierania. W tym celu dodaje się specjalny parametr do instrukcji tworzącej strumień:

```
ofstream mojStrumien("C:/Nanoc/scores.txt", ios::app);
```

Słowo `app` jest skrótem od angielskiego słowa append oznaczającego dołącz.

Teraz nowe dane nie będą powodowały skasowania starych, tylko będą dodawane na końcu pliku.

Odczytywanie pliku

Wiesz już jak zapisać dane w pliku, a więc czas nauczyć się także je odczytywać. Czynność tę wykonuje się bardzo podobnie do poprzedniej.

Otwieranie pliku do odczytu

Zasada działania jest dokładnie taka sama, jak przy zapisywaniu plików, tylko zamiast strumienia `ofstream` będziemy używać strumienia `ifstream`. Oto ogólna struktura kodu do odczytywania zawartości pliku:

```
ifstream mojStrumien("C:/Nanoc/C++/data.txt");   // Otwarcie pliku do odczytu

if(mojStrumien)
{
    // Wszystko gotowe do odczytu.
}
else
{
    cout << "BŁĄD: nie można otworzyć pliku do odczytu." << endl;
}
```

Jak widać, nic nowego tu nie ma. Plik można odczytać na trzy sposoby:

1. Linijka po linijce przy użyciu funkcji `getLine()`.
2. Słowo po słowie przy użyciu operatora `>>`.
3. Znak po znaku przy użyciu funkcji `get()`.

Poniżej znajduje się szczegółowy opis każdej z tych metod.

Odczytywanie pliku linijka po linijce

Ta metoda polega na odczytaniu jednej linijki tekstu i zapisaniu jej jako łańcucha znaków.

```
string linia;
getline(mojStrumien, linia); // Odczytanie jednej linii tekstu
```

Efekt zastosowania tej techniki jest identyczny, jak użycia instrukcji `cin`.

Odczytywanie pliku słowo po słowie

Ta metoda również jest łatwa do opanowania. Spójrz na poniższy przykład:

```
double liczba;
mojStrumien >> liczba; // Odczytanie liczby zmiennoprzecinkowej
string slowo;
mojStrumien >> slowo; // Odczytanie słowa z pliku
```

W tym przypadku odczytywane jest wszystko, co znajduje się między miejscem w pliku, w którym aktualnie się znajdujemy a najbliższą spacją. Odczytany tekst jest traktowany jako typ `double`, `int` lub `string` w zależności od typu użytej zmiennej.

Odczytywanie pliku znak po znaku

Ta metoda polega na odczytywaniu z pliku po jednym znaku i jako jedyna z wszystkich opisanych jest dla nas całkiem nowa. Ale oczywiście tak jak poprzednie, również łatwo ją opanować.

```
char a;
mojStrumien.get(a);
```

Powyższy kod odczyta jeden znak i zapisze go w zmiennej *a*.

Ta metoda odczytuje wszystkie rodzaje znaków, a więc także spacje, znaki nowego wiersza, znaki tabulacji itd.

Odczytywanie całego pliku

Często trzeba odczytać całą zawartość pliku. Na razie pokazałem Ci jak odczytywać dane z plików, ale jeszcze nie wyjaśniłem, jak zakończyć wczytywanie, gdy dojdzie się do końca pliku.

Aby dowiedzieć się czy można kontynuować odczytywanie danych, należy użyć wartości zwrotnej funkcji `getline()`. Funkcja ta nie tylko wczytuje linie zawartości plików, ale dodatkowo zwraca wartość logiczną informującą, czy można kontynuować odczyt. Zatem jeśli funkcja ta zwróci wartość `true`, to wiadomo, że można kontynuować odczytywanie. Jeśli natomiast zwróci `false`, jest to znak, że został osiągnięty koniec pliku albo wystąpił błąd. W obu tych przypadkach należy zakończyć operację odczytu.

Pamiętasz jeszcze pętle? Użyjemy jednej z nich, aby odczytać zawartość pliku do samego końca. Najlepiej nadaje się do tego pętla `while`:

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main()
{
    ifstream plik("C:/Nanoc/plik.txt");

    if(plik)
    {
         // Udało się otworzyć plik, a więc można rozpocząć odczytywanie

        string linia;     // Zmienna do przechowywania odczytanych wierszy
tekstu

        while(getline(plik, linia))    // Jeśli jeszcze nie nastąpił koniec
pliku, czytamy dalej
        {

            cout << linia << endl; // Wyświetlamy odczytany tekst w konsoli
                                   // Można też zrobić z nim coś innego
        }
    }
    else
    {
        cout << "BŁĄD: nie można otworzyć pliku do odczytu." << endl;
    }

    return 0;
}
```

Gdy wczytamy linię tekstu, możemy z nią zrobić, co tylko nam się podoba. Ten program tylko wyświetla po kolei każdy wiersz treści, ale w prawdziwym programie byłyby one używane w inny sposób. Jedynym ograniczeniem jest Twoja wyobraźnia.

Powyższa metoda odczytywania zawartości plików jest stosowana najczęściej ze wszystkich. Po wczytaniu linii tekstu do łańcucha znaków można na niej pracować przy użyciu funkcji operujących na łańcuchach.