helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

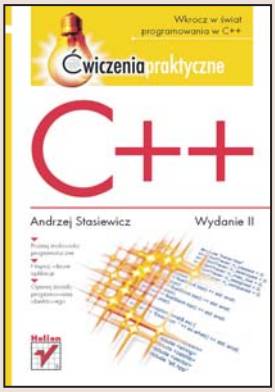
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

C++. Ćwiczenia praktyczne. Wydanie II



Autor: Andrzej Stasiewicz
ISBN: 83-246-0619-X
Format: A5, stron: 152

Wkrocz w świat programowania w C++

- Poznaj środowisko programistyczne
- Napisz własne aplikacje
- Opanuj zasady programowania obiektowego

C++ to jeden z najbardziej popularnych języków programowania. Przyczyny jego popularności to przede wszystkim niewielka liczba słów kluczowych, ogromna liczba bibliotek umożliwiających zastosowanie C++ w wielu dziedzinach, a przede wszystkim ogromne możliwości języka, pozwalające na stworzenie praktycznie dowolnej aplikacji. Systemy operacyjne, aplikacje użytkowe, gry — twórcy wszystkich tych programów wykorzystują właśnie język C++.

„C++. Ćwiczenia praktyczne. Wydanie II" to kolejna edycja książki, która zyskała ogromną popularność wśród osób chcących nauczyć się języka C++. Każde z zawartych w niej ćwiczeń zapozna Cię z elementami tego języka programowania. Nauczysz się tworzyć aplikacje konsolowe i opanujesz zasady projektowania obiektowego. W każdym z ćwiczeń znajdziesz również informacje o najczęściej popełnianych błędach i rady, jak ich unikać. Nowe wydanie książki zostało dostosowane do najnowszych wytycznych komitetu standaryzacyjnego języka C++, dzięki czemu możesz mieć pewność, że wykonując ćwiczenia zawarte w książce, poznajesz najnowszą technologię.

- Konfiguracja środowiska programistycznego
- Standardowe wejście i wyjście
- Składnia programu
- Sterowanie wykonywaniem programu
- Funkcje
- Typy danych
- Podstawy programowania obiektowego

Po lekturze tej książki zdobędziesz niezbędne podstawy do dalszej nauki i tworzenia prawdziwych aplikacji

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Pliki źródłowe w języku C++

Pliki jako nośniki programów

Treść programu komputerowego zazwyczaj umieszczamy w plikach dyskowych. Zawartość tych plików będzie odczytywana przez kompilator języka C++ i tłumaczona na ciąg binarnych poleceń dla procesora komputera.

Programowanie nie zawsze jest równoznaczne z zapisywaniem czegoś w plikach — np. w przemyśle spotykamy się z sytuacjami wprowadzania programu do komputera za pomocą odpowiedniego ustawiania mikroprzełączników. Kiedyś powszechne było umieszczanie programu na odpowiedniej ilości dziurkowanych kart.

Przygotowywanie programu w formie zapisów umieszczanych w plikach jest bardzo wygodne, tanie i uniwersalne. Zawsze można taki program odtworzyć, poprawić, zlecić jego wykonanie, zarchiwizować na całe lata.

Zapis programu dla komputera zazwyczaj ma strukturę zwykłego tekstu — mamy zatem do czynienia z plikami tekstowymi. Rzeczywiście, program napisany w zdecydowanej większości znanych języków daje

się otworzyć i przeczytać za pomocą zwykłego Notatnika. Jest to dodatkowe uproszczenie sposobu kodowania i przechowywania współczesnych programów.

Skoro pliki źródłowe są zwyczajnymi plikami tekstowymi, do programowania wystarczy najzwyklejszy edytor tekstowy — np. popularny Notatnik. Jednak większość współczesnych środowisk programistycznych udostępnia programiście własne, wbudowane edytory. Są to edytory tekstowe, ale „znające" składnię języka i na przykład odpowiednio kolorujące niektóre frazy języka. Praca nad programem w takim edytorze jest prawdziwą przyjemnością! Pamiętajmy jednak, że poradzilibyśmy sobie także, dysponując zwykłym Notatnikiem.

Nośniki programów w C++

W języku C++ przyjęto powszechnie konwencję, że głównym nośnikiem algorytmów jest plik o rozszerzeniu *cpp*, czyli np. plik o nazwie *test.cpp*. Spotkamy się także z plikami o rozszerzeniu *h*, czyli np. o nazwie *test.h*, które są nośnikami nie tyle algorytmów, ile ich zapowiedzi lub ściślej — deklaracji. Wiadomo, skąd pochodzi nazwa *cpp*, natomiast literka *h* w nazwie pliku z deklaracjami wzięła się od słowa *header* — nagłówek.

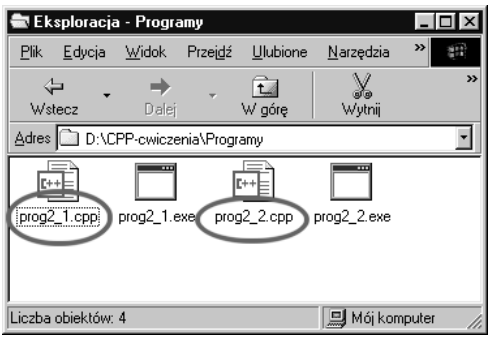
Swoje programy będziemy spisywać w pliku o nazwie np. *test.cpp* lub *przyklad.cpp*, lub *cokolwiek.cpp*. Plik ten powinien mieć strukturę zwykłego pliku tekstowego i mógłby być przygotowany w dowolnym edytorze, potem odczytany przez kompilator języka C++, skompilowany i uruchomiony.

ĆWICZENIE

3.1 Pliki źródłowe na dysku

Pliki źródłowe naszych programów:

1. Po wykonaniu ćwiczeń z poprzedniego rozdziału na dysku Twojego komputera powinny pojawić się ich pliki źródłowe. Odszukaj katalog, w którym środowisko DEV zapisało te pliki (rysunek 3.1).
2. Spróbuj otworzyć swoje pliki źródłowe za pomocą zwykłego Notatnika.



Rysunek 3.1. *Oto rzut oka na katalog roboczy — widzimy tutaj dwa pliki źródłowe (są to programy napisane w poprzednim rozdziale) i utworzone w wyniku ich kompilacji dwa finalne pliki* exe*, nadające się do uruchamiania w systemie Windows*

Najprostsze programy w całości spisuje się w pliku *cpp*. Jeśli zachodzi konieczność zadeklarowania czegokolwiek, odpowiednie frazy umieszcza się raczej w górnej części tego pliku (gdzieś przed zasadniczą funkcją `main()`) niż w oddzielnym pliku *h*. Umieszczanie deklaracji w pliku nagłówkowym jest wyrazem profesjonalizmu programisty, jego wysokiej kultury, dobrego smaku i zamiłowania do porządku. Jednak drobniutkie algorytmy z całym spokojem możemy umieszczać wyłącznie w pliku *cpp*.

Postarajmy się zapamiętać, że język C++ w najlepszym, profesjonalnym wydaniu operuje parą plików *cpp* i *h* oraz że para ta nazywa się **modułem**.

Dyrektywa #include i scalanie plików cpp i h

A oto następny szczegół, na który powinniśmy zwrócić uwagę. Skoro język C++ wprowadza do gry dwa pliki źródłowe, tym samym rozdzielając tak zwane *deklaracje* (zapowiedzi algorytmów) od tak zwanych *implementacji* (algorytmów), to w strukturze języka powinna znaleźć się dyrektywa łączenia pary takich plików w całość. Dla porównania — w Pascalu, Fortranie czy Basicu problem ten nie występuje, bo zarówno deklaracje, jak i implementacje umieszczamy w jednym i tym samym pliku.

Zazwyczaj na samej górze pliku *cpp* — czyli na samym początku spisywanych algorytmów — pojawia się dyrektywa nakazująca kompilatorowi „spojrzeć" w jakiś plik nagłówkowy. Przypomnijmy sobie którykolwiek z programów z poprzedniego rozdziału:

```
#include <iostream>
using namespace std;
int main()
{
 ...
```

Widoczna tutaj w pierwszej linii dyrektywa nakazuje kompilatorowi przeczytanie pewnego innego pliku, tutaj o nazwie *iostream*. Co prawda plik ten nie ma oczekiwanej nazwy *iostream.h*, a tylko *iostream*, i jest to swego rodzaju naruszenie zasad uświęconych tradycją, ale dyrektywie wklejania w niczym to nie przeszkadza. Rzeczywiście — wklejaniu za pomocą dyrektywy `#include` podlegają wszelkie pliki, nawet te o najdziwniejszych nazwach, czego raczej nie należy nadużywać.

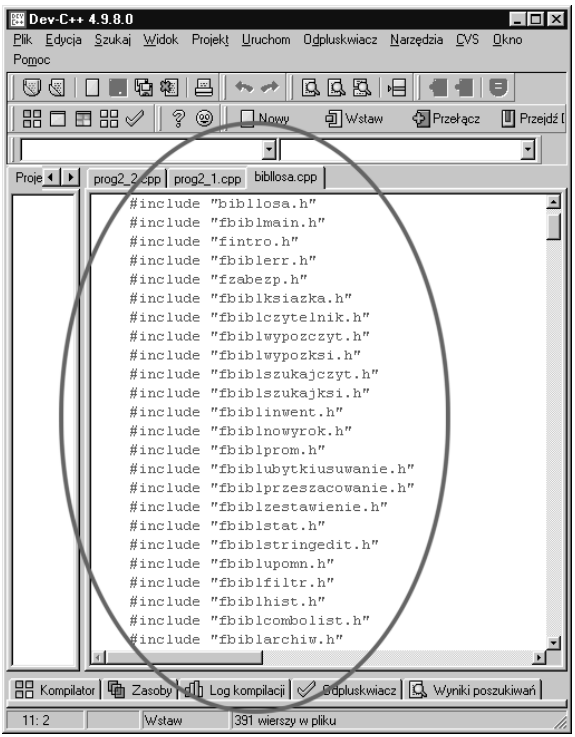
Wklejać pliki nagłówkowe trzeba praktycznie zawsze. Oddzielną sprawą jest orientowanie się, jaki plik akurat należy wkleić. Jeśli potrzebne jest wyprowadzanie informacji na ekran — chętnie wklejamy plik nagłówkowy *iostream*. Jeśli potrzebujemy jakiejś funkcji matematycznej, wkleimy prawdopodobnie plik *math.h* lub jego nowszą wersję *cmath* zawierające zapowiedzi instrumentów matematycznych — stałej *pi*, sinusa, pierwiastka czy logarytmu (porównaj rysunek 3.2).

ĆWICZENIE

3.2 Dołączanie plików nagłówkowych

Teraz zrobimy coś złego. Z któregoś z poprzednich programów usuńmy dyrektywę `#include` i poddajmy program kompilacji oraz uruchomieniu (porównaj rysunek 2.3):

```
//#include <iostream>
using namespace std;
int main()
{
 cout << "Twoje imie i nazwisko";
 return 0;
}
```



Rysunek 3.2. *Poważniejsze programy intensywnie wykorzystują dyrektywę wklejania. Niech nas nie zaniepokoją cudzysłowy zamiast ostrych nawiasów, jak w naszych mikroprogramach — jest to pozostałość po dawnych czasach, gdy maszyny nie były tak szybkie jak dzisiaj. Ostre nawiasy nakazują rozpoczęcie wyszukiwania od katalogów bibliotecznych kompilatora. Kultura nakazuje, by swoje deklaracje wklejać w średnikach, biblioteczne — w nawiasach ostrych*

Pierwsza linia została poprzedzona podwójnym ukośnikiem, zatem jest zamieniona na komentarz i nie podlega kompilacji. Czy ten program się uruchamia? Nie. Nawet się nie kompiluje — kompilacja kończy się komunikatem „niezadeklarowane cout”, „nie rozumiem cout”, „nie wiem, co znaczy cout”! Widocznie w pliku *iostream* znajdował się opis algorytmu `cout`.

Częsty błąd: *zapomnienie o dyrektywie* `#include`*, dołączającej deklaracje napisów wykorzystywanych w dalszej części programu.*

Wklejanie starszej wersji plików nagłówkowych, np. `math.h`*, zamiast współczesnej* `cmath`.

Podsumowanie

Nośnikami współczesnych programów komputerowych są zazwyczaj zwykłe pliki tekstowe.

Języki rezerwują sobie rozszerzenia nazw plików — i tak pliki w języku C++ mają nazwy *.*cpp* i *.*h*, pliki pascalowe nazywają się *.*pas*, pliki z Fortranem *.*for* itd.

Zapisuj swoje algorytmy w swoich plikach i strzeż ich jak oka w głowie!

Staraj się wyrobić w sobie nawyk, by pliki każdego programu umieszczać w oddzielnym katalogu. Jest to ważne, dlatego że współczesne, duże programy zazwyczaj składają się z wielu plików źródłowych i trudno jest lokalizować je, gdy mieszają się z plikami innych programów.

Plik *.*cpp* jest głównym nośnikiem algorytmów spisanych w języku C++.

Plik *.*h* zwyczajowo mieści deklaracje (zapowiedzi) algorytmów w języku C++. Plik *.*h* jest włączany do pliku głównego *.*cpp* za pomocą dyrektywy `#include "`*`nazwa_pliku`*`.h"` lub `#include <`*`nazwa_pliku`*`.h>`.

Niekiedy, szczególnie przy małych programach, pomija się plik *.*h* i deklaracje umieszcza bezpośrednio w pliku *.*cpp*.

Deklaracje algorytmów biblioteki standardowej języka C++ zwyczajowo umieszcza się w plikach, w których nazwach pomijamy kropkę i literę h, np. `#include <iostream>`. By jednak zapewnić wsteczną zgodność z wcześniejszymi dialektami języka, ciągle poprawnie skompiluje się stara, „klasyczna” dyrektywa wklejania `#include <iostream.h>`.