

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

Autor: Andrzej Stasiewicz

ISBN: 83-7361-479-6

Format: B5, stron: 120



Język C++ jest obecnie najpopularniejszym językiem programowania. Powodów jest kilka: niewielka liczba słów kluczowych, ogromna ilość 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” to książka przeznaczona dla osób, które chcą rozpocząć naukę tego języka. Opisuje najważniejsze zagadnienia niezbędne do pisania programów w C++. Każde z nich zilustrowane jest prostym przykładem. Po lekturze niniejszych ćwiczeń zdobędziesz podstawy niezbędne do dalszej nauki i tworzenia prawdziwych aplikacji.

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



Spis treści

Wprowadzenie	7
Dlaczego język C++ jest tak ważny?	7
Co da Czytelnikowi ta książka?	8
Co będzie potrzebne do korzystania z książki?	8
Jak uczyć się języka z tej książki?	9
Rozdział 1. Nasz programistyczny warsztat	11
Rozdział 2. Nasz pierwszy program	15
Czy to działa?	15
Sposób na znikanie okienka konsoli	18
Podsumowanie	19
Rozdział 3. Pliki źródłowe w języku C++	21
Pliki jako nośniki programów	21
Nośniki programów w C++	22
Dyrektywa #include i scalanie plików cpp i h	23
Podsumowanie	24
Rozdział 4. Więcej o strumieniach cin i cout	25
Standardowe strumienie wejścia i wyjścia	26
Kaskadowe posługiwanie się strumieniami	28
Odrobina formatowania	29
Podsumowanie	32
Rozdział 5. Przestrzeń na Twoje algorytmy	33
Początek — najlepsze miejsce na dyrektywy #include	33
Po nagłówkach — dostęp do biblioteki standardowej	35
Po bibliotece standardowej — nasze własne deklaracje	36
Funkcja main() — centrum programu	36
Po funkcji main() — definicje innych funkcji	37
Podsumowanie	38
Rozdział 6. Algorytmy	39
Zwrotnica if() ... else	39
Zwrotnica switch{ ... }	43
Pętla for(...; ...; ...).....	48

Pętla while(...)	52
Pętla do {...} while(...)	54
Instrukcje break i continue	55
Podsumowanie	59
Rozdział 7. Funkcje	61
Deklarowanie funkcji	61
Definiowanie funkcji	63
Argumenty funkcji i referencja	68
Podsumowanie	71
Rozdział 8. Dane	73
Typy danych	73
Deklarowanie i inicjowanie prostych danych	75
Deklarowanie i inicjowanie danych tablicowych	77
Deklarowanie i inicjowanie danych wskaźnikowych	80
Operacje na danych	84
Podsumowanie	89
Rozdział 9. Klasy i obiekty	91
Klasa jako nowy typ danych	91
Wewnętrzny ustrój klasy — dane	93
Wewnętrzny ustrój klasy — algorytmy	95
Pewien specjalny algorytm, zwany konstruktorem	99
Podsumowanie	105
Rozdział 10. Kontenery na dane	107
Podsumowanie	115
Zakończenie	117

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 poradziłibyśmy 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.

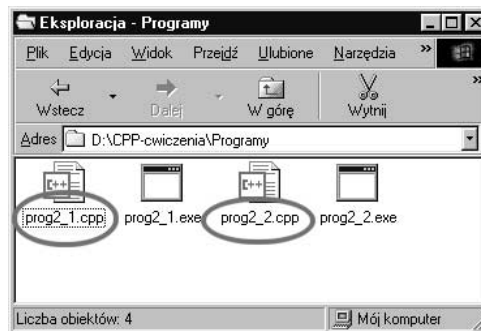
Ćwiczenie 3.1.

Pliki źródłowe 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).

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



2. Spróbuj otworzyć swoje pliki źródłowe za pomocą zwykłego Notatnika.

Najprostsze programy w całości spisuje się w pliku *cpp*. Jeśli zachodzi konieczność zadeklarowania czegośkolwiek, 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 drobnitkie 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 spojrzenie w jakiś plik nagłówkowy. Przypomnijmy sobie którykolwiek z programów z poprzedniego rozdziału:

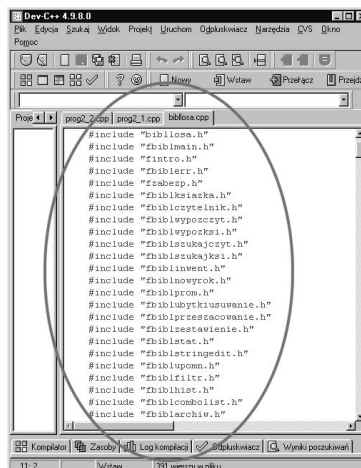
```
#include <iostream>
using namespace std;
int main()
{
    ...
}
```

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* zawierający zapowiedzi instrumentów matematycznych — stałej *pi*, sinusa, pierwiastka czy logarytmu (porównaj rysunek 3.2).

Rysunek 3.2.

Poważniejsze programy intensywnie wykorzystują dyrektywę wklejania. Niech nas nie zaniepokoją cudzy stowy 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 średnikami, biblioteczne — nawiasami ostrymi



Ćwiczenie 3.2.

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

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

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`.

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`.