W PROSTOCIE TKWI SIŁA

wydanie VII

C++

dla bystrzaków

Wprowadzenie do języka C++

Programowanie funkcjonalne

Używanie klas nadrzędnych i dziedziczenia

Stephen R. Davis
autor bestsellera *C# dla bystrzaków*

septem
septem.pl
Helion

Tytuł oryginału: C++ For Dummies, 7th edition

Tłumaczenie: Michał Sternik

ISBN: 978-83-283-5987-1

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://dlabystrzakow.pl/user/opinie/cppby7
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/cppby7.zip

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *dlabystrzakow@dlabystrzakow.pl*
WWW: *http://dlabystrzakow.pl*

Printed in Poland.

Spis treści

W TYM ROZDZIALE:

- Podstawy C++.
- Instalowanie *Code::Blocks* w systemach Windows, Ubuntu Linux lub Macintosh OS X.
- Tworzenie pierwszego programu w C++.
- Uruchamianie programu.

Rozdział 1

Pierwszy program w C++

Oto jesteśmy: tylko Ty i ja. Nie zwlekajmy, czas zacząć. Na początek przedstawię kilka podstawowych pojęć.

Komputer to niesamowicie szybka, ale zarazem niezwykle głupia maszyna. Komputer może zrobić wszystko, co mu powiesz (w granicach rozsądku), ale robi **dokładnie** to, co mu powiesz — nic więcej ani nic mniej.

Zła wiadomość dla nas, ludzi, jest taka, że komputery nie rozumieją niestety żadnego rozsądnego języka ludzkiego — nie mówią też po polsku. No dobrze, wiem, co powiesz: „Widziałem komputery, które rozumiały polski". Właściwie jednak widziałeś komputer wykonujący **program**, który potrafił zrozumieć język polski.

Komputery rozumieją język, który nazywamy **językiem komputerowym** lub **językiem maszynowym**. Człowiek mógłby posługiwać się językiem maszynowym, ale byłoby to bardzo trudne. Dlatego komputery i ludzie spotykają się pośrodku i używają języków pośrednich, takich jak C++. Ludzie rozumieją C++ (poniekąd), a C++ może zostać przekonwertowany na język maszynowy, aby mógł go zrozumieć komputer.

Pojęcia dotyczące języka C++

Program w C++ jest plikiem tekstowym zawierającym sekwencję komend C++ zgodnych z gramatyką języka C++. Ten plik tekstowy jest określany jako **plik źródłowy** (prawdopodobnie dlatego, że jest źródłem wszystkich frustracji). Pliki źródłowe w C++ zwykle zawierają rozszerzenie *.CPP*, podobnie jak pliki Adobe Acrobat zawierają rozszerzenie *.PDF*, a pliki wsadowe MS-DOS (pamiętasz je?) mają rozszerzenie *.BAT*.

Celem programowania w C++ jest napisanie sekwencji poleceń, które można przekształcić na program w języku maszynowym. Dopiero on faktycznie robi to, co chcemy. Ta konwersja nazywana jest *kompilacją* i jest zadaniem *kompilatora*. Kod maszynowy, który napisałeś, musi być połączony z pewnymi instrukcjami instalacji i usuwania oraz niektórymi standardowymi procedurami bibliotecznymi w procesie zwanym *łączeniem* lub *konsolidacją* (ang. *linking*). Kompilacja razem z konsolidacją to *budowanie*. Powstałe pliki **wykonywalne maszynowo** mają w systemie Windows rozszerzenie *.EXE*. W systemie Linux lub w komputerach Macintosh nie zawierają one żadnego konkretnego rozszerzenia.

To brzmi dość prosto — co w tym trudnego? Zatem czytaj dalej.

Aby napisać program, potrzebujesz dwóch specjalistycznych programów komputerowych. Jeden (edytor) służy do pisania kodu podczas tworzenia pliku źródłowego *.CPP*. Drugi (kompilator) konwertuje plik źródłowy na plik wykonywalny, który faktycznie wykonuje polecenia (otwórz arkusz kalkulacyjny, wydawaj nieprzyzwoite odgłosy, odbijaj nadlatujące asteroidy czy cokolwiek innego).

Obecnie twórcy narzędzi zazwyczaj łączą kompilator i edytor w jeden pakiet, nazywany **zintegrowanym środowiskiem programistycznym**. W takim środowisku, po zakończeniu wprowadzania poleceń, które tworzą program, wystarczy kliknąć jeden przycisk, aby utworzyć plik wykonywalny.

Na szczęście istnieją darmowe środowiska dla języka C++. W tej książce używam jednego z nich — Code::Blocks. Ten edytor działa z wieloma różnymi kompilatorami, ale wersja Code::Blocks w połączeniu z kompilatorem GNU gcc użytym do napisania tej książki jest dostępna dla systemów Windows i Macintosh oraz dla różnych wersji systemu Linux — tak jak zostało to opisane w punkcie niniejszego rozdziału poświęconym instalacji.

Code::Blocks jest darmową aplikacją Jednak jeśli chcesz, możesz wesprzeć jej dalszy rozwój. Zachęcam do wpłacenia niewielkiej darowizny na ten cel. Wcale nie **musisz** płacić, żeby używać Code::Blocks, ale możesz mieć swój wkład, jeśli chcesz. Szczegółowe informacje można znaleźć w witrynie Code::Blocks.

Przetestowałem programy w tej książce za pomocą Code::Blocks 13.12, który jest wyposażony w gcc w wersji 4.7.1. Ta wersja gcc implementuje większość dodatków ze standardu C++ 2011.

Możesz użyć innych wersji gcc lub nawet innych kompilatorów, ale mogą one nie implementować pełnego standardu 2011. Z tego powodu rozszerzenia 2011 są oznaczone ikoną „11” widoczną tutaj.

Użyty w tej książce kompilator gcc nie implementuje żadnego z rozszerzeń dodanych w standardzie C++ 2014, ale uwzględniłem je tam, gdzie ma to zastosowanie, ponieważ są już dostępne nowsze wersje.

Muszę się do czegoś przyznać: ta książka jest w pewnym sensie ukierunkowana na Windowsa. Wszystkie programy w książce przetestowałem w systemach Windows 2000/XP/Vista/7/8, Ubuntu Linux i Macintosh OS X. Różnice występujące pomiędzy systemami operacyjnymi zaznaczyłem w tekście.

Dodatkowo w tym rozdziale zamieściłem instrukcje instalacji dla każdego z trzech wymienionych powyżej systemów operacyjnych. Wersje Code::Blocks i gcc są dostępne dla innych odmian systemu Linux i innych wersji systemu operacyjnego Macintosh. Programy powinny także z nimi współpracować.

OSTRZEŻENIE

Pakiet Code::Blocks/gcc generuje programy 32-bitowe. Nie umożliwia jednak łatwego tworzenia programów „okienkowych”. Programy w tej książce uruchamiane są z wiersza poleceń i wyświetlają wynik w wierszu poleceń. Mimo że wydaje się to nudne, stanowczo zalecam, abyś najpierw zapoznał się z przykładami z tej książki, aby nauczyć się C++, zanim zajmiesz się tworzeniem aplikacji okienkowych. C++ i programowanie w systemie Windows to dwie różne rzeczy i (z uwagi na Twoje zdrowie psychiczne) powinny jako takie pozostać w Twojej głowie.

Postępuj zgodnie z procedurą opisaną w następnym podrozdziale, aby zainstalować Code::Blocks, a następnie zbuduj swój pierwszy program w C++. Zadaniem tego programu jest konwersja temperatury wprowadzonej przez użytkownika ze stopni Celsjusza na stopnie Fahrenheita.

Instalacja Code::Blocks

Ze strony *http://www.codeblocks.org/downloads/binaries* możesz pobrać najnowszą wersję środowiska Code::Blocks dla systemów: Windows, Ubuntu Linux i Macintosh OS X. Postępuj zgodnie z zamieszczonymi poniżej instrukcjami instalacji dotyczącymi Twojego systemu operacyjnego.

Windows

Środowisko Code::Blocks jest dostępne w łatwym do zainstalowania, skompresowanym pliku wykonywalnym, który jest zgodny ze wszystkimi wersjami systemu Windows począwszy od wersji Windows 2000. Oto sposób instalacji środowiska:

1. **Pobierz plik wykonywalny *codeblocks-13.12.mingw-setup.exe* ze strony *http://www.codeblocks.org/downloads/source/5*.**

   Zapisz plik wykonywalny na pulpicie lub w innym miejscu, które można łatwo znaleźć.


WSKAZÓWKA

   Wspomniany plik wykonywalny zawiera wersję kompilatora GCC 4.71. Nie jest to najnowsza wersja GCC, ale właśnie tej wersji użyłem podczas pisania tej książki. W chwili powstawania jej tłumaczenia najnowszą dostępną wersją Code::Blocks jest wersja 17.12. Można ją pobrać ze strony *http://www.codeblocks.org/downloads/binaries*. Ostatnia dostępna wersja dla Mac OS X to 13.12. To tej wersji będziemy używać.

2. **Po zakończeniu pobierania kliknij dwukrotnie ikonę programu.**

3. **W zależności od używanej wersji systemu Windows może pojawić się wyskakujące okienko „*Niezidentyfikowany program chce uzyskać dostęp do komputera*". Jeśli tak się stanie, kliknij *Zezwól*, aby rozpocząć instalację.**

4. **Po zamknięciu wszystkich aplikacji zewnętrznych, tak jak zostanie to wyświetlone w oknie dialogowym *Welcome to the CodeBlocks Setup Wizard* (witaj w instalatorze CodeBlocks), kliknij przycisk *Next* (dalej).**

5. **Przeczytaj *Umowę licencyjną użytkownika końcowego* (powszechnie znaną jako *EULA*), a następnie kliknij przycisk *I Agree*, jeśli zgadzasz się z jej postanowieniami.**

   Nie masz dużego wyboru — jeśli nie zaakceptujesz licencji, pakiet się nie zainstaluje. Załóżmy jednak, że się zgodzisz. Instalator Code::Blocks otworzy okno dialogowe z opcjami instalacji. Opcje domyślne są dla nas wystarczające.

6. **Kliknij przycisk *Next*.**

   Program instalacyjny pozwala zainstalować tylko niektóre podzbiory funkcjonalności. Musisz wybrać przynajmniej instalację domyślną i pakiet *MinGW Compiler Suite*. Najlepszym wyjściem jest zainstalowanie wszystkich opcji, tak jak jest to zaznaczone domyślnie.


OSTRZEŻENIE

   Jeśli nie ma opcji *MinGW Compiler Suite*, prawdopodobnie pobrałeś wersję Code::Blocks, która nie zawiera gcc. Ta wersja nie będzie działać poprawnie.

7. **Po kolejnym kliknięciu przycisku *Next* pojawi się zapytanie o folder instalacji; kliknij *Install*, aby zaakceptować domyślny folder docelowy.**

   Instalator Code::Blocks rozpocznie kopiowanie wszystkich potrzebnych plików na Twój dysk twardy. Po zakończeniu zapyta: *Do you want to run Code::Blocks now?* (Czy chcesz teraz uruchomić Code::Blocks?).

8. **Kliknij *Yes* (lub *Tak*), aby uruchomić Code::Blocks.**

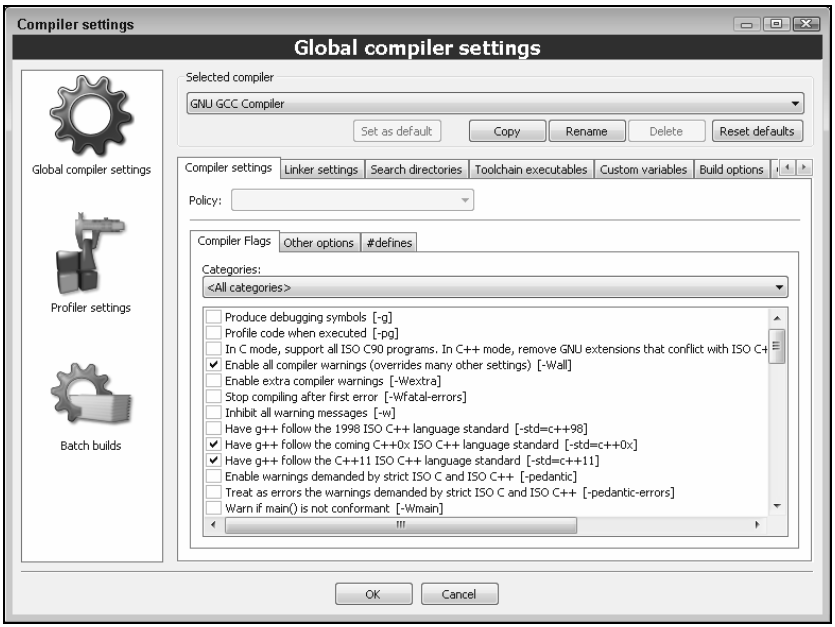
   Code::Blocks zapyta teraz, którego kompilatora użyć. Domyślny kompilator to GNU GCC. To jest właściwy wybór.

9. **Z poziomu okna głównego Code::Blocks wybierz *Settings/Compiler*.**

10. **Wybierz zakładkę *Compiler Flags* (flagi kompilatora).**

11. **Upewnij się, że wybrano następujące trzy flagi (patrz rysunek 1.1):**
    - *Enable All Compiler Warnings* (włącz wszystkie ostrzeżenia kompilatora);
    - *Have g++ Follow the Coming C++0x ISO C++ Language Standard* (niech g++ postępuje zgodnie z nadchodzącym standardem C++ 0x ISO C++);
    - *Have g++ Follow the C++11 ISO C++ Language Standard* (niech g++ postępuje zgodnie ze standardem C++11 ISO C++).

    Standard C++ 2011 pierwotnie miał być standardem C++ 2008 lub 2009. Ponieważ nie było to jasne, standard zyskał popularność pod nazwą *0x*. Standard nie został całkowicie zaakceptowany do 2011 roku. W środowisku kompilatora gcc języki C++ 0x i C++ 11 odnoszą się do tego samego standardu.



RYSUNEK 1.1. Upewnij się, że ustawiona jest flaga Enable All Compiler Warnings

12. **Wybierz zakładkę *Toolchain Executables* (pliki wykonywalne zestawu narzędzi). Upewnij się, że wygląda tak, jak pokazano na rysunku 1.2.**

    Domyślną lokalizacją kompilatora gcc jest podkatalog *MinGW\bin* katalogu Code::Blocks.

OSTRZEŻENIE

Jeśli domyślna lokalizacja jest pusta, Code::Blocks nie będzie wiedział, gdzie jest kompilator gcc, i nie będzie mógł budować programów. Upewnij się, że pobrałeś wersję Code::Blocks, która zawiera gcc, oraz że zaznaczyłeś właściwą opcję podczas instalacji. Jeśli używasz kompilatora gcc zainstalowanego wcześniej, powinieneś wskazać programowi Code::Blocks miejsce na dysku twardym, w którym się on znajduje.

RYSUNEK 1.2. Upewnij się, że katalog instalacyjny kompilatora jest poprawny

13. **Zamknij okno dialogowe ustawień — *Compiler Settings*.**

14. **Kliknij *Next* w oknie dialogowym instalatora Code::Blocks, a następnie kliknij *Finish* (zakończ), aby zakończyć program instalacyjny.**

    Program instalacyjny zakończy działanie.

Ubuntu Linux

W systemie Linux instalacja jest procesem dwuetapowym, ponieważ Code::Blocks dla Linuksa nie zawiera gcc. Najpierw trzeba zainstalować gcc, a następnie Code::
↪Blocks.

Instalacja gcc

Kompilator gcc dla Linuksa jest łatwo dostępny. Aby go zainstalować, uruchom terminal i wykonaj następujące czynności:

1. **Wprowadź następujące polecenia z wiersza poleceń:**

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install g++
```

Standardowa dystrybucja Ubuntu Linux zawiera kompilator GNU C, ale nie obejmuje rozszerzeń C++. W szczególności nie zawiera rozszerzeń standardu C++ 2011. Pierwsze dwa polecenia aktualizują narzędzia, które już masz. Trzecie polecenie instaluje C++.

2. **Wprowadź następujące polecenie z wiersza poleceń:**

```
gcc —-version
```

Mój Ubuntu 18.04 pobrał GNU C++ w wersji 7.4.0. Na potrzeby przykładów z tej książki wersja 4.7.1 jest wystarczająca. Jeśli masz wcześniejszą wersję, niektóre funkcje C++ 2011 mogą nie działać poprawnie. Poza tym wszystko powinno działać jak należy.

Jeśli używasz Debiana, polecenia są takie same. Jeśli używasz Red Hata, zamiast programu `apt-get` zastosuj program `yum`. Polecenie instalacji dla Red Hata wygląda następująco:

```
sudo yum install g++
```

Instalacja Code::Blocks

Wersja Code::Blocks gotowa do użycia w Ubuntu jest dostępna w *Ubuntu Software Center* (Centrum Oprogramowania Ubuntu). Na szczęście dla wszystkich zainteresowanych wiele innych wersji systemu Linux zawiera coś podobnego do Centrum Oprogramowania. Aby zainstalować Code::Blocks, wykonaj poniższe czynności:

1. **Kliknij ikonę Centrum Oprogramowania na pulpicie Ubuntu.**
2. **Wybierz Code::Blocks z listy dostępnego oprogramowania.**

   Rozpocznie się proces instalacji.

   Code::Blocks przeszukuje dysk twardy w poszukiwaniu kompilatora C++. Powinien być w stanie go znaleźć bez problemu, ale jeśli tego nie zrobi, wykonaj poniższe kroki.
3. **Uruchom Code::Blocks.**
4. **Wybierz *Settings*/*Compiler* (ustawienia kompilatora).**
5. **Wybierz zakładkę *Compiler Flags* (flagi kompilatora).**
6. **Upewnij się, że wybrałeś następujące trzy flagi (patrz rysunek 1.1):**
   - *Enable All Compiler Warnings* (włącz wszystkie ostrzeżenia kompilatora);
   - *Have g++ Follow the Coming C++0x ISO C++ Language Standard* (niech g ++ postępuje zgodnie z nadchodzącym standardem C++ 0x ISO C++);
   - *Have g++ Follow the C++11 ISO C++ Language Standard* (niech g++ postępuje zgodnie ze standardem C++11 ISO C++).
7. **Wybierz zakładkę *Toolchain Executables* (pliki wykonywalne zestawu narzędzi).**
8. **Kliknij przycisk ....**
9. **Przejdź do */usr*, chyba że zainstalowałeś kompilator gcc w innym miejscu niż domyślna lokalizacja */usr/bin*.**

10. **W formularzu, w polu *C compiler* należy wpisać `gcc`, w polu *C++ compiler* należy wybrać *g++*, w polu *Linker for dynamic libs* również należy wybrać *g++*.**

11. **Wybierz *OK*, aby zamknąć okno dialogowe.**

Macintosh

Wersja Code::Blocks dla komputerów Macintosh używa dystrybucji Xcode firmy Apple jako kompilatora. Z tego powodu podzieliłem instalację na trzy części.

Instalacja Xcode

Potrzebne jest darmowe środowisko programistyczne oferowane przez Apple, czyli *Xcode*. Aby je zainstalować, wykonaj poniższe czynności:

1. **Otwórz przeglądarkę *Safari* i przejdź na stronę *http://developer.apple.com*.**

2. **Kliknij *Download Xcode* (pobierz Xcode), aby pobrać najnowszą wersję programu.**

   Spowoduje to otwarcie okna dialogowego pobierania Xcode, pokazanego na rysunku 1.3.

RYSUNEK 1.3. Okno dialogowe pobierania Xcode pozwala zainstalować Xcode za darmo

3. **Kliknij przycisk *Free*, aby zmienił się na *Install App* (zainstaluj aplikację).**

4. **Możesz zostać poproszony o wprowadzenie hasła systemowego (którym logujesz się podczas uruchamiania komputera Mac).**

   Ikona zmieni się na *Installing*.

   Pobieranie i instalacja zajmuje trochę czasu (w chwili pisania przeze mnie tego tekstu archiwum z Xcode miało nieco ponad 2 GB).

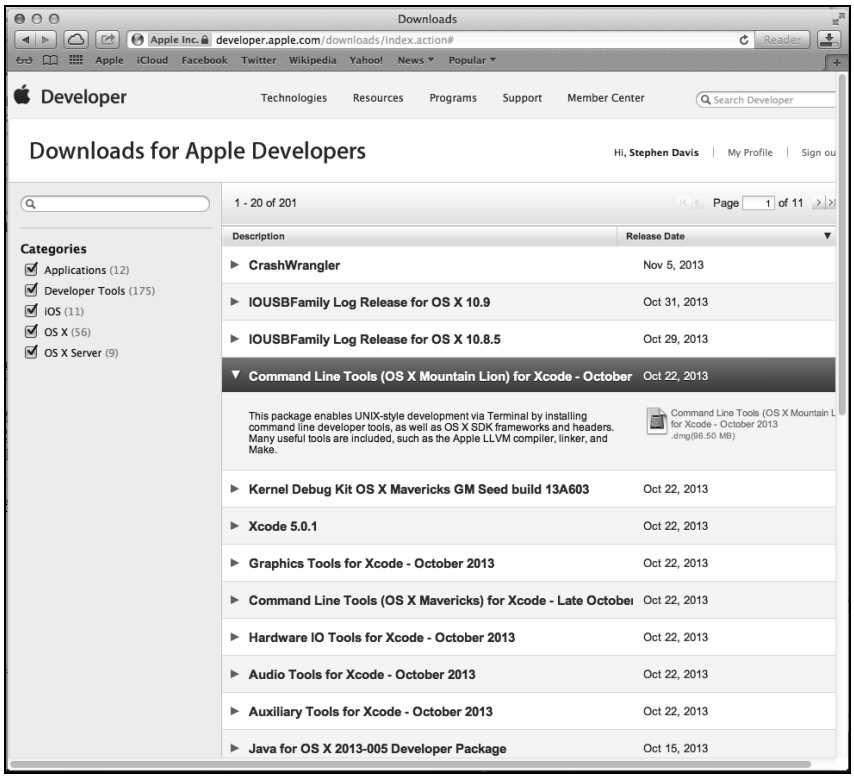
Instalowanie narzędzi wiersza polecenia

Można by się spodziewać, że tak duży program jak Xcode ma wszystko, czego potrzebuje. Nic bardziej mylnego. Do uzyskania działającego kompilatora gcc na komputerze Macintosh potrzebny jest jeszcze jeden pakiet firmy Apple. Aby zainstalować narzędzia wiersza poleceń dla Xcode, postępuj zgodnie z poniższą procedurą:

1. **Otwórz przeglądarkę Safari i przejdź na stronę *http://developer.apple.com/downloads*.**

   Możesz zostać poproszony o zarejestrowanie identyfikatora *Apple Developer ID*. Bez obaw, to nic nie kosztuje.

2. **Wyszukaj narzędzia wiersza poleceń dla Xcode. Wybierz aplikację pokazaną na rysunku 1.4. Następnie kliknij ikonę *Download*.**



RYSUNEK 1.4. Aby przygotować kompilator dla macintosha, musisz zainstalować zarówno Xcode, jak i Command Line Tools dla Xcode

3. **Kliknij dwukrotnie pobrany pakiet mpkg, aby go zainstalować.**
4. **Zaakceptuj wszystkie wartości domyślne.**

   Instalacja powinna zakończyć się powodzeniem.

Instalacja Code::Blocks

Teraz możesz dokończyć instalację, pobierając i instalując edytor Code::Blocks:

1. **Otwórz przeglądarkę Safari i przejdź na stronę *www.codeblocks.org/downloads*.**
2. **Kliknij *Downloads*/*Binaries* (pobierz/binaria).**
3. **Kliknij Mac OS X.**
4. **Wybierz jedno ze źródeł plików, aby pobrać najnowszą wersję.**

   W chwili powstawania tego tekstu najnowsza dostępna wersja to *CodeBlocks-13.12-mac.zip*.
5. **Zainstaluj pobrany plik *Zip* w folderze *Applications*.**

   Jeśli nigdy nie instalowałeś aplikacji z witryny innej firmy, zanim będziesz mógł to zrobić, być może będziesz musiał wykonać poniższe dodatkowe operacje:

   a) kliknąć *System Preferences* (preferencje systemowe);

   b) kliknąć *Security and Privacy* (bezpieczeństwo i prywatność);

   c) kliknąć kłódkę w lewym dolnym rogu okna dialogowego, aby zezwolić na zmiany;

   d) w formularzu *Allow applications downloaded from* (zezwalaj na pobieranie aplikacji z) kliknąć opcję *Anywhere* (skądkolwiek), tak jak pokazano na rysunku 1.5.

RYSUNEK 1.5. Przed instalacją Code::Blocks w Twoim systemie Macintosh trzeba zainstalować inne aplikacje

Po zakończeniu instalacji Code::Blocks możesz powrócić do tego okna dialogowego i przywrócić domyślną wartość ustawień (*Mac App Store*).

6. **Kliknij dwukrotnie ikonę Code::Blocks.**

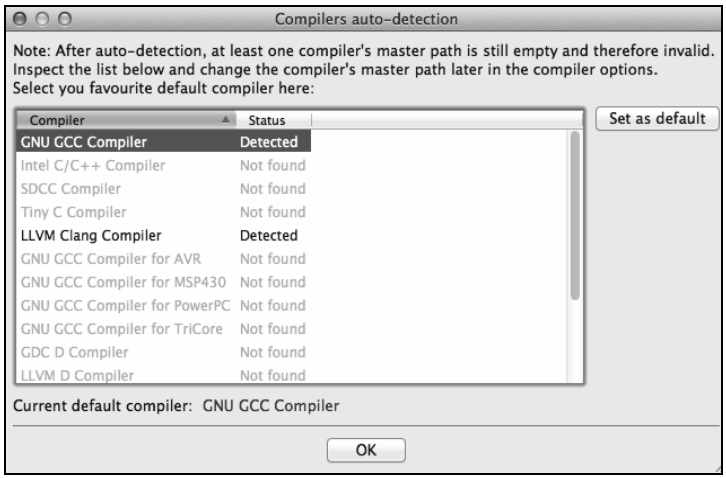
   Przy pierwszym uruchomieniu system Mac OS zapyta Cię, czy na pewno chcesz go otworzyć.

7. **Wybierz *Don't Warn Me When Opening Applications on This Disk Image* (nie ostrzegaj mnie przy otwieraniu aplikacji na obrazie dysku) i kliknij *Open* (otwórz).**

   Code::Blocks powinien się uruchomić i znaleźć zainstalowany kompilator gcc wraz z narzędziami wiersza polecenia.

8. **Wybierz kompilator gcc, jak pokazano na rysunku 1.6. Kliknij *Set as Default* (ustaw jako domyślny). Aby kontynuować uruchamianie Code::Blocks, kliknij *OK*.**

   Code::Blocks otworzy stronę z banerem, a następnie wyświetli menu w górnej części okna dialogowego.



RYSUNEK 1.6. Code::Blocks przy pierwszym uruchomieniu automatycznie znajduje zainstalowany kompilator gcc

9. **Wybierz *Settings/Compiler* (ustawienia kompilatora), następnie kliknij opcję *Have g ++ Follow the Coming C++ 0x ISO C++ Language Standard* (niech g++ postępuje zgodnie z nadchodzącym standardem C++ 0x ISO C++). Kliknij *OK*, aby zamknąć okno dialogowe.**

   Teraz jesteś gotowy do utworzenia pierwszego programu w C++.

Tworzenie pierwszego programu w C++

W tym podrozdziale utworzysz swój pierwszy program w C++. Kod C++ wprowadzimy w pliku o nazwie *CONVERT.CPP*, a następnie skonwertujemy ten kod na program wykonywalny.

Tworzenie projektu

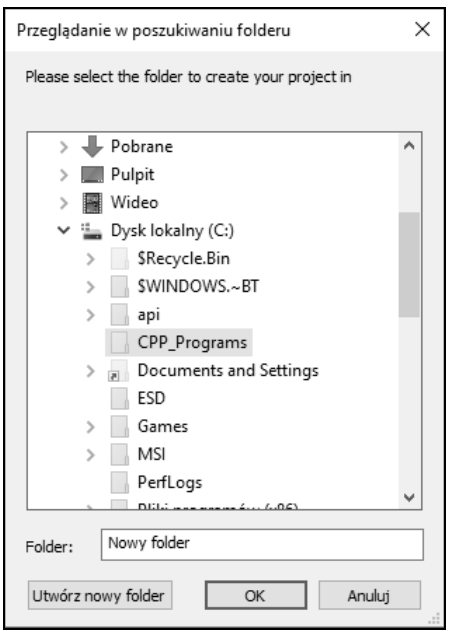
Pierwszym krokiem na drodze do stworzenia programu w C++ jest opracowanie projektu. Projekt w Code::Blocks definiuje nazwy plików źródłowych *.CPP*, które mają zostać dołączone, oraz typ programu do utworzenia. Większość programów w książce składa się z jednego pliku źródłowego i jest uruchamiana z wiersza poleceń.

1. **Uruchom środowisko Code::Blocks.**
2. **W oknie Code::Blocks wybierz *File/New/Project* (plik/nowy/projekt).**
3. **Wybierz ikonę *Console Application* (aplikacji konsoli), a następnie kliknij przycisk *Go* (przejdź).**
4. **W następnym oknie dialogowym wybierz *C++* jako język, którego chcesz używać. Kliknij *Next* (następny).**

   Code::Blocks i gcc umożliwiają również pisanie programów w języku C.
5. **W polu *Folder to Build Project In* (folder do budowy projektu) wybierz ikonę ....**
6. **W systemie Windows kliknij *Ten Komputer*, a następnie dysk *C:*.**

   W systemach Linux i Macintosh możesz wybrać Pulpit.
7. **Wybierz przycisk *Utwórz nowy folder* w lewym dolnym rogu ekranu.**
8. **Nazwij nowy folder *CPP_Programs*.**

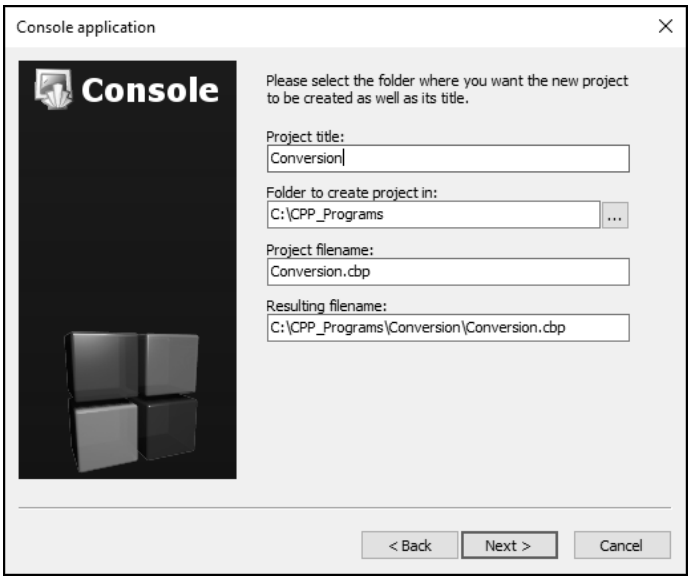
   Wynik powinien wyglądać tak, jak pokazano na rysunku 1.7.



RYSUNEK 1.7. W systemie Windows umieść swój projekt w folderze C:\CPP_Programs

9. **W polu *Project title* (tytuł projektu) wpisz nazwę projektu, w tym przypadku *Conversion*.**

   Ekran wynikowy w systemie Windows pokazano na rysunku 1.8. Wersje dla systemów Linux i Macintosh nie niczym się od niego różnią, z wyjątkiem ścieżki.



RYSUNEK 1.8. Tworzenie projektu Conversion dla pierwszego programu

10. **Kliknij *Next*.**

    Następne okno dialogowe pozwala na modyfikację nazwy pliku i ścieżki, gdzie będą zapisywały się skompilowane pliki programu, zarówno wersji do testowania, jak i ostatecznej. Możesz zostawić domyślne wartości.

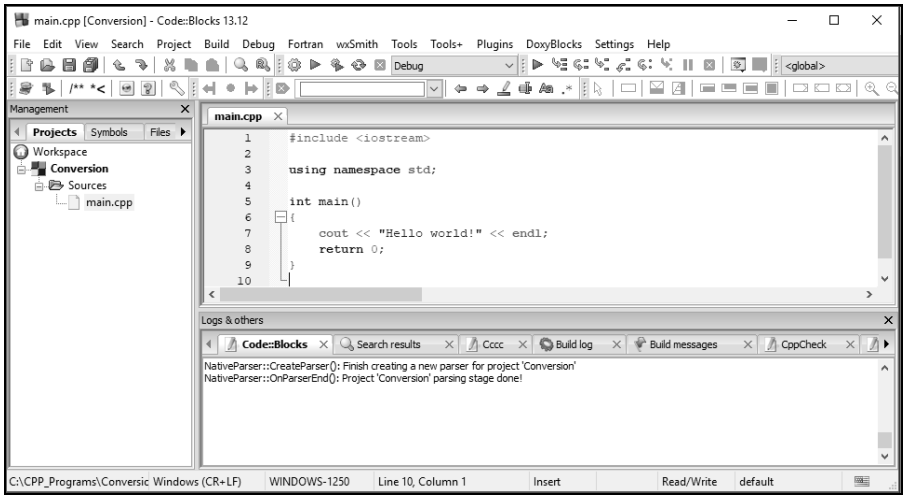
11. **Kliknij przycisk *Finish* (zakończ), aby utworzyć projekt *Conversion*.**

Wprowadzanie kodu w języku C++

Projekt *Conversion*, który stworzył Code::Blocks, składa się z pojedynczego pliku *main.cpp*, wyświetlającego komunikat `Hello, world`. Teraz wprowadź kod programu:

1. Kliknij dwukrotnie plik *main.cpp*, który znajduje się w oknie dialogowym *Management* (zarządzanie) po lewej stronie. Plik jest zagnieżdżony w węzłach *Conversion/Sources*.

   Jak pokazano na rysunku 1.9, Code::Blocks otwiera w edytorze kodu domyślny program *main.cpp*.



RYSUNEK 1.9. Okno dialogowe Management wyświetla strukturę katalogów wszystkich dostępnych programów

2. Zastąp zawartość pliku *main.cpp* kodem podanym poniżej.

Nie przejmuj się zbytnio wcięciami lub odstępami — nie ma znaczenia, czy dana linia jest wcięta na dwie spacje, czy na trzy. Tak samo nie ma znaczenia, czy między słowami jest jedna spacja, czy są dwie spacje. Musisz się upewnić, że wszystkie litery są małe, ponieważ kompilator C++ rozróżnia wielkość liter.


WSKAZÓWKA

Możesz użyć ściągawki dostępnej w plikach na stronie *www.dummies.com/extra/cplusplus*, tak jak to zostało opisane w następnym punkcie.

```
//
// Conversion — Program do konwersji temperatury ze
// Stopni Celsjusza na stopnie Fahrenheita:
// Fahrenheit = Celsjusz * (212 - 32)/100 + 32
//
#include <cstdio>
#include <cstdlib>
#include <iostream>
using namespace std;
int main(int nNumberofArgs, char* pszArgs[])
{
    // wprowadź temperaturę w stopniach Celsjusza
    int celsius;
    cout << "Wprowadź temperaturę w stopniach Celsjusza:";
    cin >> celsius;
    // dokonaj konwersji stopni Celsjusza
    // na stopnie Fahrenheita
    int factor;
    factor = 212 - 32;
    // użyj przelicznika do przeliczenia stopni Celsjusza
    // na stopnie Fahrenheita
    int fahrenheit;
    fahrenheit = factor * celsius/100 + 32;
```

```
    // wyświetl wyniki (a następnie znak przejścia do nowego wiersza)
    cout << "Wartość w stopniach Fahrenheita to:";
    cout << fahrenheit << endl;
    // Aby pozwolić użytkownikowi zobaczyć wyniki programu,
    // poczekaj, aż użytkownik będzie gotowy przed zakończeniem programu.
    cout << "Naciśnij Enter, aby kontynuować..." << endl;
    cin.ignore(10, '\n');
    cin.get();
    return 0;
}
```

3. **Wybierz *File/Save*, aby zapisać plik źródłowy.**

   Może nie wydaje Ci się to ekscytujące, ale właśnie stworzyłeś swój pierwszy program C++!

Korzystanie ze ściągawki

Wszystkie programy w książce są dostępne online wraz z plikami projektu do ich tworzenia. Aby z nich skorzystać, musisz je pobrać i zainstalować. Następnie postępuj zgodnie z poniższą procedurą:


WSKAZÓWKA

Poniższe instrukcje dotyczą systemu Windows. Czynności, które należy wykonać dla Linuksa czy Macintosha, są bardzo podobne.

1. **Otwórz przeglądarkę internetową.**
2. **Przejdź na stronę** *https://www.dummies.com/store/product/C-For-Dummies-6th-Edition.*
   *↪productCd-0470317264,navId-322468,descCd-DOWNLOAD.html***.**
3. **Kliknij link do *C++ Code files*.**

   Może wyświetlić się okno dialogowe z pytaniem, gdzie zapisać pobrany plik.
4. **Kliknij *Zapisz plik*.**

   System operacyjny skopiuje plik *CPP_programs.zip* do wskazanej w poprzednim kroku lokalizacji lub do domyślnej lokalizacji pobierania.
5. **Kliknij prawym przyciskiem myszy plik *CPP_programs.zip* i wybierz *Otwórz*.**

   Otworzy się okno dialogowe zawierające pojedynczy katalog *CPP_Programs*.
6. **Skopiuj ten folder na dysk *C*.**

   Spowoduje to skopiowanie wszystkich plików źródłowych użytych w książce do katalogu *C:\CPP_Programs*.


OSTRZEŻENIE

Możesz umieścić folder *CPP_Programs* w innym miejscu. Pamiętaj, aby nie umieszczać plików źródłowych w katalogu, do którego ścieżka zawiera spację. W systemie Windows uważaj, żeby nie umieszczać plików, których używa Code::Blocks, w folderach takich jak *Moje dokumenty* czy *Pulpit*, ponieważ ścieżka do nich zawiera spację.

Są dwa sposoby korzystania z tych plików. Jednym z nich jest utworzenie programu ręcznie poprzez wykonanie procedury opisanej w tej książce. W razie popełnienia błędu możesz skopiować i wkleić kod z pobranych plików do swojego programu. Jest to sposób zalecany.

Drugie podejście polega na korzystaniu bezpośrednio z pobranych źródeł i pliku projektu:

1. **Kliknij dwukrotnie plik *AllPrograms.workspace* w folderze *C:\CPP_Programs*.**

   Rozszerzenie *workspace* charakteryzuje plik, który odwołuje się do jednego lub większej liczby projektów. Plik *AllPrograms.workspace* zawiera odwołania do wszystkich projektów z tej książki.

2. **Kliknij prawym przyciskiem myszy projekt *Conversion* w oknie dialogowym *Management* po lewej stronie. Następnie z wyświetlonego menu kontekstowego wybierz *Activate project*.**

   Program, który jest aktywny, ma pogrubioną etykietę. W tym przypadku jest to projekt *Conversion*. Gdy wydajesz polecenie budowania — *Build*, Code::Blocks zawsze tworzy aktywny projekt.

3. **Aby otworzyć plik w edytorze, kliknij dwukrotnie plik *main.cpp*.**

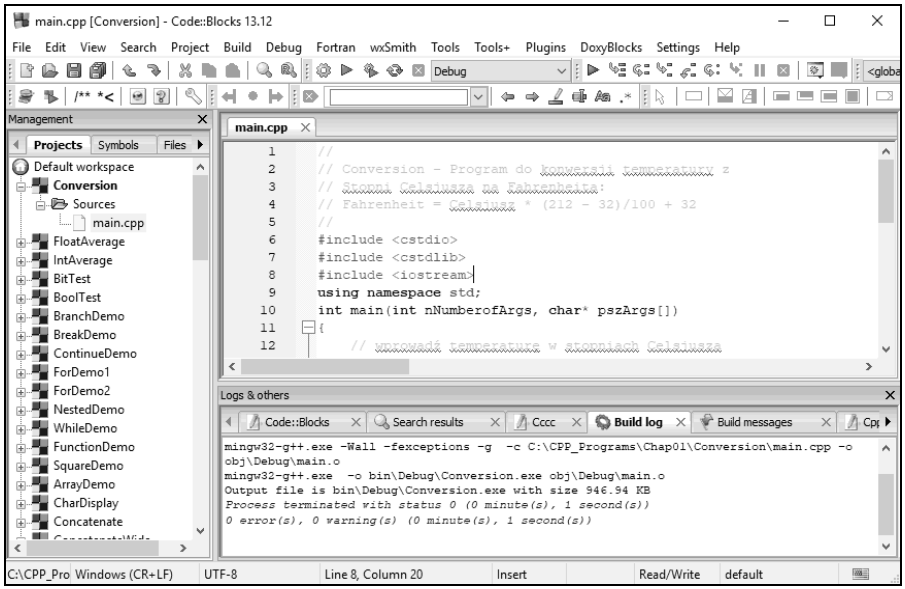
Problem z tym podejściem polega na tym, że jeśli sam nie wprowadzasz kodu w C++, nie uczysz się języka zbyt skutecznie.

Budowanie programu

Po zapisaniu pliku źródłowego w C++ na dysku twardym nadszedł czas na wygenerowanie wykonywalnych instrukcji maszynowych.

Aby zbudować program *Conversion*, wybierz polecenie *Build/Build* z paska poleceń lub wciśnij *Ctrl+F9*. Niemal natychmiast Code::Blocks rozpocznie proces kompilacji programu. Jeśli wszystko pójdzie dobrze, w prawym dolnym oknie dialogowym wyświetli się komunikat z liczbą błędów i liczbą ostrzeżeń równymi 0. Przykład przedstawiono na rysunku 1.10.

Jeśli Code::Blocks wykryje jakikolwiek błąd w programie C++ — a błędy kodowania są tak samo powszechne jak piasek na plaży — poinformuje Cię o tym za pomocą stosownego komunikatu. Bez wątpienia napotkasz w przyszłości wiele ostrzeżeń i komunikatów o błędach. Błędy są prawdopodobne nawet podczas pisania tak prostego programu jak *Conversion.cpp*. Aby zobaczyć, jak wygląda proces zgłaszania błędów, zamień w wierszu 16. instrukcję `cin >> celsius;` na `cin >>> celsius;`.



RYSUNEK 1.10. Code::Blocks buduje program Conversion

Być może dla Ciebie i dla mnie jest to błąd „do wybaczenia”, ale nie dla kompilatora C++. Wybierz *Build/Build*, aby uruchomić proces kompilacji. Prawie natychmiast Code::Blocks umieści czerwony kwadrat obok wiersza z błędem. Komunikat `error: expected primary-expression before '>' token` w zakładce *Build messages* może Ci nic nie mówić. Usuń dodatkowy znak > i ponownie skompiluj, aby pozbyć się tego błędu.

WSKAZÓWKA

Prawdopodobnie uważasz, że komunikat o błędzie wygenerowany przez przykład jest tajemniczy. Daj sobie trochę czasu — programujesz dopiero jakieś 30 minut. Z biegiem czasu będziesz znacznie lepiej rozumiał komunikaty o błędach generowane przez Code::Blocks i gcc.

OSTRZEŻENIE

Tym razem Code::Blocks potrafił wskazać błąd, ale nie zawsze mu się to udaje. Czasami błąd zostaje zauważony dopiero w którymś z kolejnych wierszy. Zatem jeśli wiersz oznaczony jako błędny wygląda dobrze, to aby znaleźć błąd, sprawdź wiersze poprzednie.

Uruchamianie programu

Nadszedł czas, aby sprawdzić, jak działa Twoje nowe dzieło, czyli uruchomić program. Teraz uruchomimy plik programu *Conversion* i wprowadzimy dane wejściowe, aby zobaczyć, jak świetnie działa.

Aby uruchomić program Conversion w Code::Blocks w systemie Windows, wybierz *Build/Build and Run* lub naciśnij *F9*. Jeśli coś się zmieniło, program ponownie zostanie skompilowany, a następnie, jeśli kompilacja się powiedzie, uruchomi się.

Na początek otworzy się okno dialogowe proszące o podanie temperatury w stopniach Celsjusza. Wprowadź znaną temperaturę, np. 100 stopni. Po naciśnięciu klawisza *Enter* program zwróci temperaturę 212 stopni Fahrenheita, która jest równoważna temperaturze 100 stopni Celsjusza. Oto przykład:

```
Wprowadź temperaturę w stopniach Celsjusza:100
Wartość w stopniach Fahrenheita to:212
Naciśnij Enter, aby kontynuować...
```

Wiadomość `Naciśnij Enter, aby kontynuować...` daje możliwość przeczytania wpisanego tekstu, zanim zniknie. Naciśnij *Enter*, a okno dialogowe (wraz z zawartością) zniknie. Gratulacje! Właśnie napisałeś, zbudowałeś i uruchomiłeś swój pierwszy program w C++.

Zauważ, że Code::Blocks nie jest przeznaczony do tworzenia programów okienkowych, takich jak te używane w systemie Windows. Napisanie aplikacji okienkowej dla Windows przy użyciu Code::Blocks nie jest łatwe, ale możliwe. Znacznie prościej jest stworzyć aplikację okienkową w Visual Studio.

Programy okienkowe Windows pokazują użytkownikowi dane w sposób czytelny wizualnie. Są one estetycznie rozmieszczone w oknach na ekranie. *Conversion.exe* to 32-bitowy program, który jest *wykonywany w systemie* Windows, ale nie jest programem Windows w sensie wizualnym.

Nie przejmuj się, jeśli nie wiesz, co oznacza program 32-bitowy. Jak wspomniałem wcześniej, ta książka nie dotyczy pisania programów okienkowych w systemie Windows. Programy C++, które piszesz w tej książce, mają *interfejs wiersza poleceń* wykonywalny w oknie wiersza poleceń.

Jeśli jesteś początkującym programistą aplikacji okienkowych dla Windowsa, nie przejmuj się — nie zmarnowałeś pieniędzy. Nauczenie się C++ jest niezbędne przed przystąpieniem do pisania programów dla Windowsa. Moim zdaniem powinieneś uczyć się programowania w następującej kolejności: najpierw C++, potem Windows.

Przeglądanie programu z komentarzami

Wprowadzanie danych do czyjegoś programu jest równie ekscytujące jak oglądanie kogoś, kto jeździ samochodem. Aby było to ekscytujące, musisz sam zająć miejsce za kierownicą. Programy są trochę podobne do samochodów. Wszystkie samochody są w zasadzie takie same, z małymi różnicami. Być może francuskie samochody różnią się trochę bardziej od innych, ale wyjątek potwierdza regułę. Samochody są stworzone według tego samego wzoru — kierownica z przodu, fotel pod kierowcą, dach nad kierowcą i tak dalej.

Podobnie wszystkie programy C++ mają wspólny wzorzec. Ten wzorzec jest już obecny w naszym pierwszym programie. Aby znaleźć elementy wspólne dla wszystkich programów, spróbujmy spojrzeć na program *Conversion*.

Analiza struktury programów w C++

Każdy program w C++ opisany w tej książce opiera się na tej samej podstawowej strukturze. Wygląda ona podobnie jak w tym przykładzie:

```
//
// Szablon — zapewnia szablon do użycia jako punkt wyjścia
//
// następujące dołączone pliki definiują większość
// funkcji potrzebnych każdemu programowi
#include <cstdio>
#include <cstdlib>
#include <iostream>
using namespace std;
int main(int nNumberofArgs, char* pszArgs[])
{
    // Twój kod C++ zaczyna się tutaj
    // aby pozwolić użytkownikowi zobaczyć wyniki programu,
    // poczekaj przed zakończeniem programu, aż użytkownik będzie gotowy
    cout<< "Naciśnij Enter, aby kontynuować..." <<endl;
    cin.ignore(10, '\n');
    cin.get();
    return 0;
}
```

Bez wdawania się we wszystkie nudne szczegóły: wykonywanie programu zaczyna się od kodu zawartego między znakami otwarcia i zamknięcia nawiasu klamrowego, bezpośrednio po wierszu definiującym funkcję `main()`.

Skopiowałem ten kod do pliku o nazwie *Template.cpp*, znajdującego się w głównym folderze *CPP_Programs*.

Opatrywanie kodu źródłowego komentarzami

Pierwsze kilka wierszy w programie *Conversion* wydaje się zwykłym tekstem. Albo ten tekst był przeznaczony dla ludzkich oczu, albo kompilator C++ jest o wiele mądrzejszy, niż mi się wydawało. Te sześć pierwszych wierszy to **komentarze**. Pisząc segment kodu, programista może opisać to, co robi lub myśli, i umieścić to w komentarzu przed nim. Kompilator ignoruje komentarze. Programiści (przynajmniej ci dobrzy) tego nie robią.

Komentarz w C++ zaczyna się podwójnym ukośnikiem (//) i kończy znakiem przejścia do nowego wiersza. W komentarzu możesz umieścić dowolne znaki. Komentarz nie ma ograniczenia długości, ale zazwyczaj nie dodaje się komentarzy, których długość przekracza 80 znaków. W dawnych czasach — przymiotnik „dawnych" jest tu pojęciem względnym — ekrany komputerów były ograniczone do 80 znaków szerokości. Niektóre drukarki nadal mają ustawione 80 znaków jako domyślną długość wiersza wydruku. Obecnie utrzymywanie pojedynczej linii krótszej niż 80 znaków jest po prostu dobrą praktyką. Z tych samych powodów co zawsze: tekst jest łatwiejszy do odczytania, a oczy mniej się męczą.

W czasach maszyn do pisania *znak końca wiersza* był znany jako *powrót karetki* (ang. *carriage return*). *Znak przejścia do nowego wiersza* to znak kończący wiersz kodu.


OSTRZEŻENIE

C++ pozwala na stosowanie innego sposobu wprowadzania komentarzy. Polega on na umieszczeniu komentarza pomiędzy znakami /* i */. Wszystkie znaki między tymi dwoma znacznikami są ignorowane przez kompilator. Taka forma komentarzy nie jest jednak powszechnie używana w C++.

Istnienie w C++ (lub jakimkolwiek innym języku programowania) polecenia, które jest specjalnie ignorowane przez komputer, może wydawać się dziwne. Jednak wszystkie języki programowania mają jakiś sposób dodawania komentarzy. Wyjaśnienie tego, co działo się w umyśle programisty, gdy pisał kod, jest bardzo ważne. Myśli programisty mogą nie być oczywiste dla jego kolegi po fachu, który próbuje zastosować lub zmodyfikować program. Nawet ten sam programista, który napisał program, gdy spojrzy na kod kilka miesięcy później i nie znajdzie żadnej wskazówki, może sobie nie przypomnieć, co miał na myśli, tworząc go.

Tworzenie programów z instrukcji C++

Wszystkie programy C++ bazują na tak zwanych instrukcjach C++. W tym punkcie omówiono instrukcje składające się na strukturę programu użyte w programie *Conversion*.

Instrukcja to pojedynczy zestaw poleceń. Prawie wszystkie instrukcje C++, poza komentarzami, kończą się średnikiem (jeszcze jeden wyjątek opisany został w rozdziale 10.). Wykonywanie programu w C++ rozpoczyna się od pierwszej instrukcji po otwierającym nawiasie klamrowym. Kolejne instrukcje są wykonywane po kolei, tak jak zostały wymienione w listingu.

Podczas przeglądania kodu możesz zobaczyć, że w całym programie pojawiają się spacje, tabulatory i znaki nowego wiersza. Po każdej instrukcji występuje nowy wiersz. Wszystkie te znaki nazywane są *białymi znakami* (ang. *whitespace*), ponieważ nie widać ich na monitorze.


WSKAZÓWKA

Białe znaki możesz umieszczać w dowolnym miejscu w swoim programie. Zwiększa to czytelność kodu. Nie możesz jednak umieszczać ich w środku wyrazu:

```
Wid zisz o c
o mi chod zi?
```

Chociaż kompilator C++ ignoruje białe znaki, jest bardzo czuły na wielkość liter: `pelnapredkosc` i `PelnaPredkosc` to dwie różne zmienne, które nie mają ze sobą nic wspólnego. Słowo kluczowe `int` jest zrozumiałe, ale kompilator C++ nie ma pojęcia, czym jest `INT`. Wiesz już, co miałem na myśli, mówiąc o szybkich, ale głupich komputerach?

Deklaracja zmiennych

Wiersz kodu zawierający `int celsius;` to deklaracja. *Deklaracja* jest instrukcją, która definiuje zmienną. Zmienna to kontener do przechowywania wartości określonego typu. Zmienna zawiera *wartość*, taką jak liczba lub znak.

Termin *zmienna* wywodzi się z formuł algebry następującego typu:

```
x = 10
y = 3 * x
```

Zmienna `y` jest w drugim wyrażeniu ustawiona na `3` razy `x`, ale co to jest `x`? Zmienna `x` działa jako kontener przechowujący wartość. W tym przypadku wartość `x` wynosi `10`. Równie dobrze `x` może równać się `20`, `30` lub `–1`. Drugie sformułowanie ma sens bez względu na wartość `x`.

W algebrze nie trzeba rozpoczynać od instrukcji takiej jak `x = 10`. Za to w C++ programista musi zdefiniować zmienną `x`, zanim będzie mógł z niej skorzystać.

W C++ definicja zmiennej składa się z typu i nazwy. Zmienna zdefiniowana w wierszu 11. nazywa się `celsius` i zadeklarowano, że zawiera liczbę typu `int`, czyli liczbę całkowitą (ang. *integer*) (nigdy się nie dowiemy, dlaczego w tym przypadku nie użyto po prostu *integer* zamiast *int*; to jedna z tych rzeczy, z którymi trzeba po prostu żyć).

Dla kompilatora C++ nazwa zmiennej nie ma szczególnego znaczenia. Zmienna musi rozpoczynać się od dużej lub małej litery z zakresu od A do Z lub od znaku podkreślenia (_). Wszystkie kolejne znaki muszą być małymi lub dużymi literami, cyframi od 0 do 9 lub znakami podkreślenia. Nazwy zmiennych mogą być dowolnie długie.


WSKAZÓWKA

Konwencja mówi, że nazwy zmiennych powinny się zaczynać od małej litery, a każde nowe słowo w zmiennej powinno się zaczynać się od dużej litery, na przykład `mojaZmienna`.


WSKAZÓWKA

Staraj się nazywać zmienne krótko, ale opisowo. Unikaj takich nazw jak `x`, ponieważ widząc `x`, możemy się nie domyślić jego znaczenia. Nazwa zmiennej taka jak `dlugoscSegmentuLinii` dużo lepiej opisuje, co może oznaczać wartość zmiennej.

Generowanie wyjścia programu

Wiersze zaczynające się od `cout` i `cin` są znane jako instrukcje wejścia-wyjścia. Często posługujemy się skrótem „instrukcje *I/O*" (ang. *input/output*) (programiści uwielbiają skróty i akronimy, tak jak wszyscy inżynierowie).

Pierwsza instrukcja I/O mówi: „Przekaż frazę `Wprowadź temperaturę w stopniach Celsjusza:` do `cout`" (wymawiane jak angielskie „see-out" — widoczne na zewnątrz). `cout` to w C++ nazwa standardowego urządzenia wyjściowego. W tym przypadku standardowym urządzeniem wyjściowym C++ jest Twój monitor, a dokładniej okno konsoli, w której uruchamiasz program.

Następny wiersz opisuje odwrotne działanie. Mówi: „Pobierz wartość z urządzenia wejściowego do C++ i zapisz ją w zmiennej `celsius` typu `int`". Standardowym urządzeniem wejściowym w C++ jest zwykle klawiatura. Mamy tutaj analogię C++ do wspomnianego wcześniej wzoru algebraicznego `x = 10`. W pozostałej części programu wartość zmiennej `celsius` to wartość liczbowa, którą wprowadził użytkownik.

Obliczanie wyrażeń

Wszystkie programy, oprócz tych najbardziej podstawowych, wykonują jakieś obliczenia. W C++ *wyrażenie* jest instrukcją, która wykonuje obliczenia. Mówiąc inaczej, wyrażenie jest instrukcją, która ma wartość. Za to *operator* to polecenie, które generuje wartość.

Na przykład w programie *Conversion* — w szczególności w dwóch wierszach oznaczonych jako wyrażenie obliczeniowe — program deklaruje zmienną, a następnie przypisuje jej wartość wynikającą z obliczeń. To polecenie oblicza różnicę liczb `212` i `32`; operatorem jest znak minus (`-`), a całe wyrażenie to `212–32`.

Przechowywanie wyników wyrażenia

Język mówiony może być bardzo niejednoznaczny. Jedną z takich niejednoznaczności jest termin *równy*. Zwrot *równa się* może oznaczać, że dwie rzeczy mają tę samą wartość, na przykład „złotówka jest równa stu groszom". Równość może także oznaczać przypisanie, jak w matematyce, gdy mówisz, że y jest równe 3 razy x.

Aby uniknąć niejednoznaczności, programiści C++ znak równości (`=`) nazywają **operatorem przypisania**. Mówi on: „wyniki na prawo od znaku *przypisania* wyrażenia przechowuj w zmiennej po lewej stronie". Programiści powiedzą: „*zmiennej* `factor` przypisujemy wartość 212 minus 32." W skrócie można powiedzieć: „do `factor` *podstaw* 212 minus 32".

OSTRZEŻENIE

Nigdy nie mów: „`factor` jest równy 212 minus 32”. Możesz usłyszeć takie sformułowanie od jakiegoś leniucha, ale my nie jesteśmy leniuchami.

Pozostała część programu Conversion

Drugie wyrażenie w programie Conversion jest nieco bardziej skomplikowane niż pierwsze. To wyrażenie również używa symboli matematycznych — `*` do mnożenia, `/` do dzielenia i `+` do dodawania. Jednak w tym przypadku obliczenia są wykonywane na zmiennych, a nie na stałych.

Wartość zawarta w zmiennej o nazwie `factor` (która, nawiasem mówiąc, została obliczona jako wynik `212 - 32`) jest mnożona przez wartość zawartą w zmiennej `celsius` (która została wprowadzona z klawiatury). Wynik jest dzielony przez 100 i sumowany z 32. Końcowy wynik wyrażenia jest przypisany do zmiennej całkowitej `fahrenheit`.

Kolejne dwa polecenia wyświetlają na standardowym urządzeniu wyjściowym napis: *Wartość w stopniach Fahrenheita to:*, a następnie wartość zmiennej `fahrenheit`. Wszystko to dzieje się tak szybko, że użytkownik nie wie, że zostało wykonanych tyle różnych czynności.

Ostatnie trzy instrukcje przypominają użytkownikowi o naciśnięciu klawisza *Enter*, a następnie czekają, aż to zrobi. Jest to konieczne, ponieważ w niektórych systemach program może wyświetlić wyniki, a następnie zamknąć okno dialogowe konsoli tak szybko, że nawet nie wiadomo, czy coś się stało.

WSKAZÓWKA

W wielu systemach możesz pominąć te trzy wiersze — Code::Blocks i tak zachowa otwarte okno dialogowe, dopóki nie naciśniesz *Enter*. Mimo wszystko dodanie tych wierszy kodu nie zaszkodzi.

Końcowa instrukcja `return 0;` zwraca sterowanie do systemu operacyjnego.

Skorowidz

F

G

H

I

K

L

Ł

M

N

O

P

R

S

Ś

T

U

W

Z


PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA
Helion

Język C++ od podstaw

Bez względu na to, czy jesteś początkującym czy średnio zaawansowanym programistą, który chce doszlifować swoje umiejętności, dzięki *C++ dla bystrzaków* możesz zostać sprawnym programistą. Ten podręcznik, zaktualizowany, uwzględniający zmiany z rozszerzenia do języka C++ z 2014 roku, pomoże Ci zrozumieć klasy, dziedziczenie, pokaże, jak sprawić, by Twój program wykonywał się w sposób bezpieczny — i wiele więcej.

W książce:

- Twój pierwszy program
- Operatory matematyczne i logiczne
- Programowanie obiektowe
- Obsługa błędów i obrona przed hakerami

Jonathan Landaw

jest autorem wielu bestsellerów i artykułów. Programowaniem zajmuje się od ponad 30 lat, obecnie pracuje w korporacji Booz Allen Hamilton dla Departamentu Obrony Stanów Zjednoczonych.

dla bystrzaków

Zamówienia telefoniczne:
0 801 339900 0 601 339900

septem
septem.pl

Sprawdź najnowsze promocje:
http://dlabystrzakow.pl/promocje
Książki najchętniej czytane:
http://dlabystrzakow.pl/bestsellery
Zamów informacje o nowościach:
http://dlabystrzakow.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: rady@dlabystrzakow.pl
http://dlabystrzakow.pl

Helion

Cena 69,00 zł

ISBN 978-83-283-5987-1

9 788328 359871