WYBRANE ŚRODOWISKA PROGRAMISTYCZNE (IDE)

Podręczny pendrive początkującego programisty

Większość prezentowanych środowisk może być zainstalowana na pendrivie i następnie używana w dowolnym miejscu, gdzie mamy dostęp do komputera z systemem operacyjnym Windows (w wersji XP i nowszych — w starszej wesji Windows 98 możemy mieć problemy ze sterownikami do pendrive'a). Turbo Pascal może natomiast być uruchamiany nawet z dyskietki, na której umieścimy niezbędne pliki.

Na pendrivie umieścimy kilka przydatnych plików wsadowych (*.bat*) oraz skrótów do zainstalowanych programów. Kopiowanie zmodyfikowanych plików wsadowych i skrótów do różnych folderów naszego pendrive'a niesamowicie ułatwi nam pracę.

Plik: *console.bat*

```
cmd
```

Zawiera jedną komendę `cmd` uruchamiającą konsolę w bieżącym folderze.

Plik: *xDriveOn.bat*

```
subst X: \
X:
path x:\tp;%PATH%
cmd
```

Uruchomienie pliku wsadowego tworzy dysk wirtualny *X:*, przypisując mu główny folder bieżącego dysku, czyli pendrive'a z którego był uruchomiony plik wsadowy. Następnie ten dysk jest wybierany jako dysk bieżący. Do ścieżki wyszukiwań dodawany jest folder *x:\tp* (to tylko przykład, dodany z myślą o zainstalowaniu Turbo Pascala w folderze *x:\tp*). Na koniec otwierana jest konsola (z czego można ewentualnie zrezygnować).

Kolejny plik wsadowy odłącza dysk wirtualny

Plik: *xDriveOff.bat*

```
C:
subst X: /D
```

Wszystkie potrzebne aplikacje będziemy instalowali i konfigurowali na dysku *X:*. W ten sposób dostaniemy przenośne środowiska, możliwe do używania z pendrive'a na dowolnym kompute-

rze z systemem Windows XP i w nowszych wersjach. Według potrzeb możemy tworzyć własne wersje plików wsadowych i skrótów do instalowanych aplikacji.

Srodowisko TP 5.5 lub inna wersja oprogramowania dla systemu operacyjnego DOS, może być używana w Windows 7, po zainstalowaniu emulatora DOS BOX version 0.74 (http://www.dosbox.com/). Pobieramy plik instalacyjny, np. DOSBox0.74-win32-installer.exe (1415 kB)i instalujemy na dysku X: w folderze DOSBox (rozmiar na dysku 4,48 MB). Program instalujący utworzy na pulpicie skrót DOSBox074.lnk, który skopiujemy do głównego folderu dysku X: (naszego pendrive'a). Jesteśmy już gotowi do uruchamiania starych DOS-owych aplikacji w w systemie Windows 7. W ten sposób posiadacze komputera z innymi systemami (np. Linux) mogą programować w Turbo Pascalu.

Na pendrivie możemy umieścić ikonę *pppp.ico* (własnego projektu) i plik *autorun.inf,* co ułatwi nam odnajdowanie naszego pendrive'a w systemie:

```
[AutoRun]
OPEN=
ICON=pppp.ico
ACTION=""
LABEL=PPPP
```

Turbo Pascal 5.5

Turbo Pascal firmy Borland jest wersją języka przeznaczoną dla mikrokomputerów zgodnych z IBM PC. Pakiet zawiera interakcyjny system programowania, złożony z kompilatora języka Turbo Pascal, ekranowego edytora tekstów oraz programu do wyszukiwania i usuwania usterek w kodzie (ang. *debugger*). Wersja 5.5 pojawiła się pod koniec maja 1989 r. Od tej wersji TP oferuje możliwość programowania zorientowanego obiektowo[1].

Wadą systemu jest interfejs nieobsługujący myszki i brak możliwości uruchomienia w nowszych wersjach systemu Windows (np. Windows 7) — problem dotyczy edytora, kompilatora i programów wynikowych (skompilowanych we wcześniejszych wersjach, np. Windows XP).

Do zalet należy niewątpliwie zaliczyć szybki kompilator, stabilnie działające środowisko IDE i niewielki rozmiar całości.

Środowisko można bezpłatnie pobrać ze strony firmy Embarcadero *(http://edn.embarcadero.com/article/20803)* — potrzebna jest jedynie darmowa rejestracja. Zainteresowani historią programowania w Pascalu, znajdą tu również wcześniejsze wersje Turbo Pascala (1.0 i 3.02). Nowsze wesje jezyka, Turbo Pascal 6.0, Turbo Pascal 7.0 i Borland Pascal 7.0 (na tym rozwój środowiska dla DOS zakończył się) nie uzyskały statusu freeware, ale sporo licencji znajduje się w szkołach i na uczelniach. Jeśli ktoś ma do nich dostęp, to może je do nauki programowania wykorzystać.

[1] Na podstawie przedmowy do książki: A. Marciniak, *Turbo Pascal 5.5*, Wydawnictwo Nakom, Poznań 1993.

Ponieważ edycja kodu źródłowego i uruchomienie programu odbywa się w konsoli, to nie ma problemu z polskimi znakami diakrytycznymi (ta sama strona kodowa w obu przypadkach). Wystarczy w kosoli ustawić czcionkę true type Lucida Console.

Instalacja programu. Wersja instalacyjna TP5.5 mieści się na dwóch dyskietkach i w tej postaci ją otrzymamy – pobrany plik *tp55.zip* zawiera spakowane dwa foldery *Disk1* i *Disk2*. Kopiujemy zawartość folderów do jednego folderu, np. *PASCAL55* na pendrivie (18 plików, razem 1,07 MB). Wersja instalacyjna zawsze może się przydać.

Uruchamiamy plik wsadowy *xDriveOn.bat*. Z folderu *PASCAL55* uruchamiamy aplikację *INSTALL.EXE* (np. poleceniem *pascal55\install* w konsoli, z głównego folderu dysku *X:*). W trakcie instalacji odpowiadamy na kilka pytań ustalając kolejno:

Enter the SOURCE drive to use: X

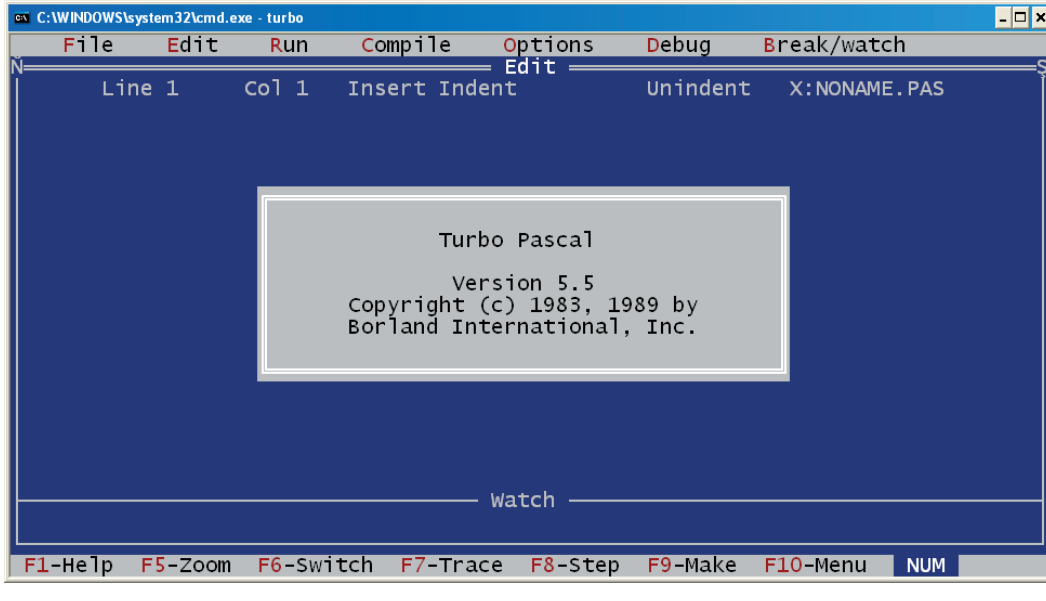
Enter the SOURCE path: \pascal55

Install Turbo Pascal on a Hard Drive

Turbo Pascal Directory: X:\TP (i pozostałe ścieżki).

Program został pomyślnie zainstalowany. Ponieważ plik wsadowy *xDriveOn.bat* dodał do ścieżki przeszukiwań folder *X:\tp*, wystarczy w konsoli wpisać polecenie *turbo* i uruchomimy IDE Turbo Pascala 5.5 (rysunek 1.).

Przed uruchomieniem programu, proponujemy utworzenie na dysku *X:* folderu *pascal* do przechowywania plików źródłowych i wynikowych (z konsoli, polecenie: *md X:\pascal*).



Rysunek 1. Zintegrowane środowisko Turbo Pascala 5.5.

Klawiszem Esc zamykamy informację o prawach autorskich – okienko w centrum ekranu. W menu systemowym okna, możemy ukryć wskaźnik myszki (IDE nie jest przez myszkę obsłu-

giwane). Musimy nauczyć się kilku zasad obsługi programu bez pomocy myszki (w razie awarii myszki ta umiejętność przyda się w Windows – inne mogą być tylko skróty klawiaturowe). Ważny w tym wszystkim jest lewy klawisz *Alt*, w kombinacji z innymi klawiszami znakowymi lub funkcyjnymi lub klawisze funkcyjne (opis na pasku statusu). Oto niektóre przydatne funkcje:

F1 – Help — pomoc i na tym można by zakończyć, resztę można przeczytać w systemie pomocy (pomoc ma charakter kontekstowy i jest dostępna w dowolnym momencie).

Esc (ang. *escape* – ucieczka) — wycofywanie się z podjętych działań.

F2 – Save — zapisanie pliku źródłowego.

F3 – Load — ładowanie pliku, obecnie częściej stosujemy zwrot otwieranie (ang. *open*).

Alt – X — zamknięcie programu (środowiska IDE).

Alt – F5 (ang. *user screen* — ekran użytkownika) oglądanie efektów pracy programu na konsoli.

F10 – Menu — uaktywnienie menu, wybór opcji klawiszami sterującymi kursorem (strzałki) i zatwierdzenie klawiszem *Enter*, *Esc* – rezygnacja.

Alt – F – uaktywnienie menu File w głównym menu, dalszy wybór *klawiszami sterującymi kursorem* (strzałki) i zatwierdzenie klawiszem *Enter*, klawiszami odpowiadającymi wielkim literom w nazwach opcji lub przy użyciu skrótów odpowiadających tym opcjom.

Alt – F, *C* (opcja *Change dir*) — ustawimy folder *X:\pascal*, jako folder do bieżącej pracy z plikami źródłowymi.

Alt – E (*Edit*) — przejście do edytora, w tej opcji nie znajdziemy funkcji typu kopiuj, wklej. Poważny mankament stosowanego edytora. W systemie pomocy odszukajmy polecenia Block Comands — pozostanie nauczyć się pewnych skrótów *Ctrl-K*, ... i problem rozwiązany (Pamiętajmy, środowisko powstało 22 lata temu).

Wpisujemy kod źródłowy...

Alt – C, *Enter* (menu *Compile*) lub *Ctrl – F9* (skrót) — kompilujemy program, domyślnie kod źródłowy umieszczany jest w pamięci.

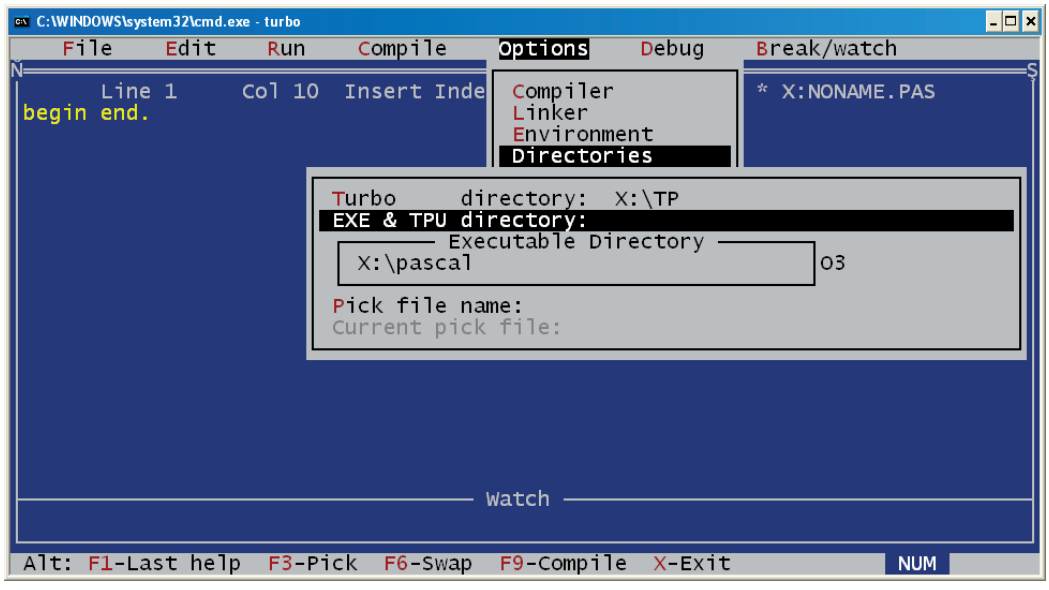
Alt – C, *D* (opcja: *Destination* w menu *Compile*) — przełączamy miejsce kompilowania kodu pomiędzy pamięcią a dyskiem (powstanie plik wykonywalny *.exe*). Czynność tę wykonujemy przed kompilacją.

Jeśli program skompilował się poprawnie, to możemy go uruchomić...

Alt – R, *Enter* lub *Alt – R*, *R* lub *Ctrl – F9* — opcja *Run* z menu *Run*.

Warto jeszcze wspomnieć o możliwościach konfiguracji środowiska, np:

Alt – O, *D*, *E* (menu: *Options* | *Directories* | *EXE & TPU directory*) — ustawienie folderu, w którym będą zapisywane pliki wykonywalne (*.exe*) i skompilowane moduły (*.tpu*).



Rysunek 2. Przykład konfiguracji środowiska Turbo Pascal 5.5.

Rozpracowanie pozostałych opcji pozostawiamy Czytelnikowi. Pomimo historycznego już znaczenia środowiska Turbo Pascal 5.5, warto poświęcić nieco czasu na pracę z tym IDE. Zdobyte tutaj nawyki i doświadczenia, zaowocują w przyszłości. Większe możliwości zaoferują nam środowiska Turbo Pascal 6.0, 7.0 i Borland Pascal 7.0 (nie są one niestety dostępne jako freeware).

Free Pascal

Free Pascal jest 32- oraz 64-bitowym kompilatorem języka Pascal, dostępnym na wiele różnych platform sprzętowych i systemów operacyjnych. Jest to kontynuacja Turbo Pascala. Pracę nad 16-bitowym kompilatorem dla DOS-a, rozpoczął student Florian Klaempfl. Po opublikowaniu prac w Internecie, do projektu przyłączyło się wielu innych programistów.

Kompilator Free Pascal jest rozpowszechniany zgodnie z licencją GPL. Biblioteki wykonawcze oraz dodatkowe pakiety rozpowszechniane razem z kompilatorem objęte są zmodyfikowaną licencją LGPL - autorzy wyrazili zgodę na dołączanie (linkowanie) tych bibliotek do innych programów bez względu na licencję tworzonego programu i sposób łączenia (statyczne lub dynamiczne).

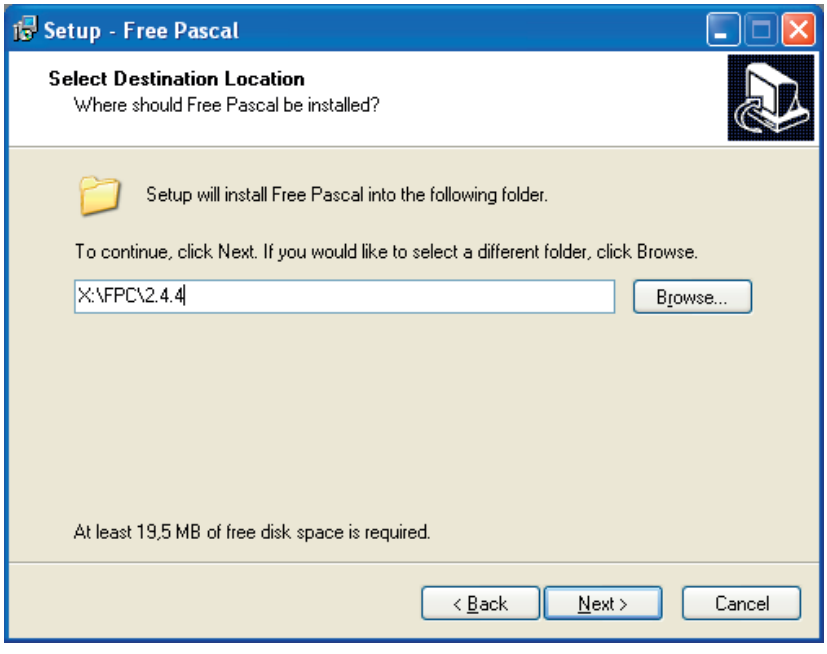
Składnia języka jest zgodna z TP 7.0 oraz z większością wersji Delphi. Free Pascal udostępnia pełny zakres programowania obiektowego. Możliwe jest przeciążanie funkcji i przeciążanie operatorów oraz zagnieżdżanie komentarzy. Free Pascal obsługuje arytmetykę 64-bitową, SSE. Wbudowana jest optymalizacja dla procesorów Pentium, AMD, AMD64[2].

Wersję instalacyjną (dla wybranego systemu operacyjnego) można pobrać ze strony *http://www.freepascal.org/download.var* lub wielu innych źródeł. Najnowsza wersja dla systemu Win-

[2] Informacje na podstawie Wikipedii: *http://pl.wikipedia.org/wiki/Free_Pascal*

dows zawarta jest w pliku: *fpc-2.4.4.i386-win32.exe* (*40.4* MB) — *http://sourceforge.net/projects/freepascal/files/*.

Program instalacyjny skopiujemy na nasz dysk *X:* do folderu *Instalator* (nazwa jest bez znaczenia, natomiast proponujemy porządkowanie naszych zbiorów na dysku) i zainstalujemy w folderze *X:\fpc\2.4.4* zmieniając w sugerowanej ścieżce literę dysku (*C:* na *X:*).



Rysunek 3. Instalacja programu Free Pascal.

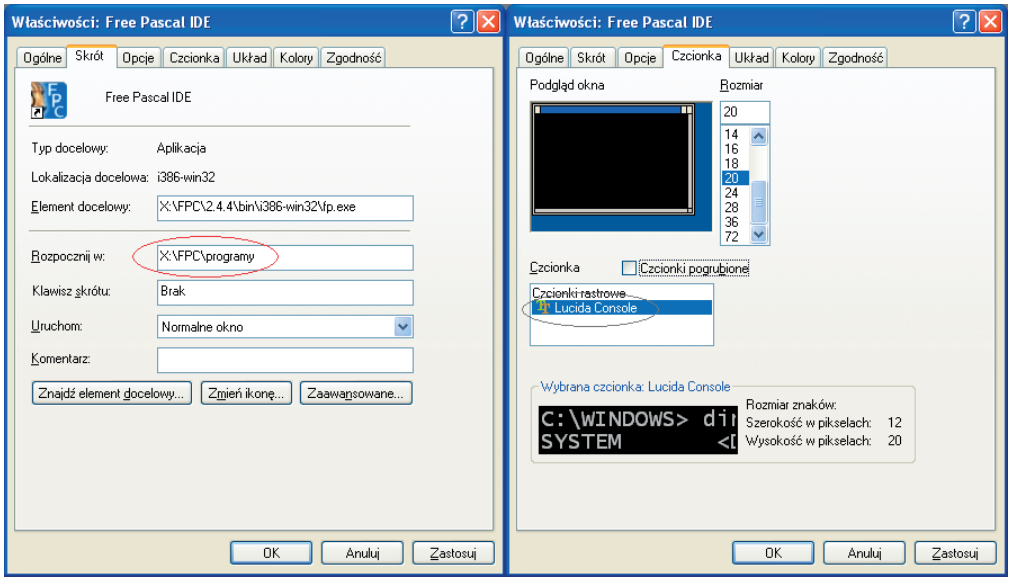
W trakcie instalacji (w kolejnych okienkach) wybieramy opcje:

Full instalation

Don't create a Start Menu folder

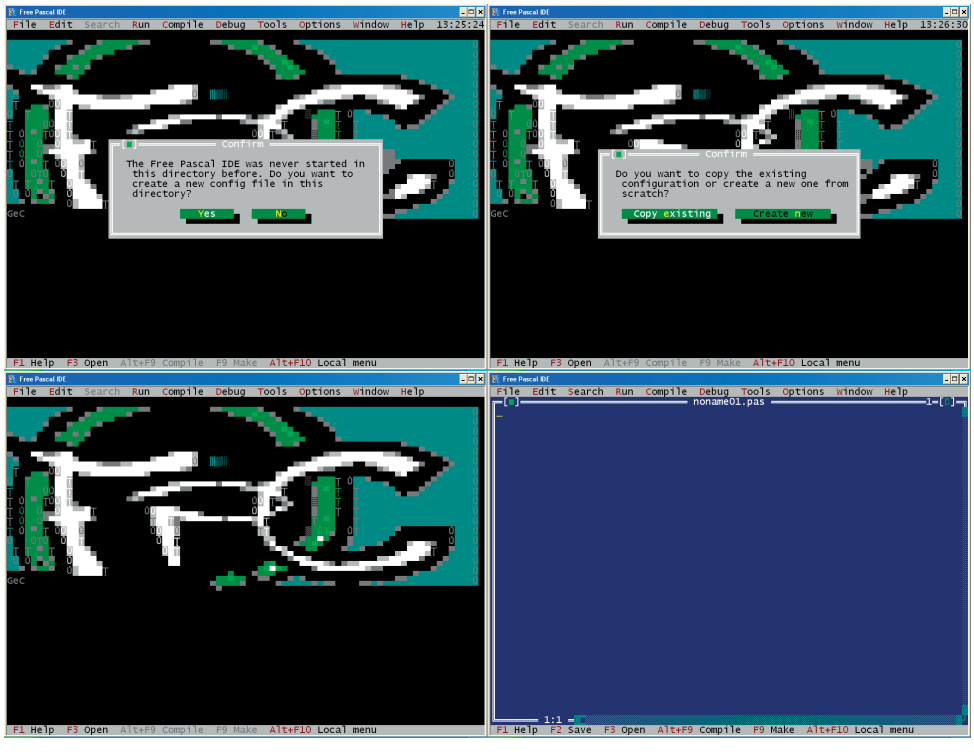
W oknie *Select Additional T*ask, zezygnujemy ze skojarzeń rozszerzeń (będą zapisane w komputerze, na którym dokonujemy instalacji — dysk *X:* będzie przenoszony), skrót na pulpicie utworzymy, aby następnie go sobie przenieść na dysk *X:*, pozostawimy również tworzenie plików konfiguracyjnych...

Po ukończeniu instalacji przenosimy skrót *Free Pascal IDE* (plik: *Free Pascal IDE.lnk*) do głównego folderu dysku *X:*. Tworzymy folder na pliki źródłowe i wykonywalne, np. *X:\FPC\programy* i modernizujemy skrót (rysunek 4.) — ustalamy folder, w którym będą przechowywane pliki jako folder startowy, wybieramy czcionkę *TT Lucida Console*, zawierającą polskie znaki diakrytyczne. Możemy również dokonać innych zmian, wg własnych upodobań.



Rysunek 4. Modyfikacja skrótu Free Pascal IDE.

Program jest gotowy do pierwszego uruchomienia (rysunek 5).



Rysunek 5. Pierwsze uruchomienie Free Pascal IDE.

Podczas pierwszego uruchomienia, w folderze startowym tworzymy nowy plik konfiguracyjny, wybierając w kolejnych oknach opcje (przycisk) *Yes* i *Create New*. Ekran startowy nie wygląda zbyt estetycznie (powiększona ikona nie prezentuje się najlepiej). Mamy jednak do dyspozycji myszkę i możemy wybrać z menu opcję: *File* | *New*. Jesteśmy gotowi do pracy.

Kombinacje klawiszy i ważniejsze opcje menu omówione dla Turbo Pascala 5.5 pozostają aktualne. Nie ma opcji przełączania kompilacji pomiędzy pamięcią a dyskiem (znanej z Turbo Pascala).

Każdy plik źródłowy jest *kompilowany* do pliku *obiektowego* (*.o*), a następnie plik obiektowy jest łączony z niezbędnymi *bibliotekami* (*konsolidacja* lub inaczej *linkowanie*) i tworzony jest *plik wykonywalny* (*.exe*).

Do pliku wsadowego tworzącego dysk wirtualny *X:* (*xDriveOnn.bat*) warto dopisać polecenie:

```
path X:\FPC\2.2.4\bin\i386-Win32;%PATH%
```

Umożliwi tu uruchamianie Free Pascala poleceniem *fp* w konsoli oraz korzystanie z kompilatora (polecenie: *fpc*) w linii komend.

Należy dodać, że system pomocy użytkownik musi skonfigurować samodzielnie. Podane informacje w zupełności wystarczą do wykonania ćwiczeń podanych w książce.

Delphi

Środowisko powstało w roku 1995, w firmie Borland, jako kontynuacja środowiska Borland Pascala stwarzającego możliwości programowania aplikacji do systemu Windows 3.1.Rozwój kolejnych wersji Delphi spowodował, że obecnie mówimy o *języku Delphi* bazującym na Pascalu. Delphi jest narzędziem typu *RAD* (*Rapid Application Development* — szybkie tworzenie aplikacji przy zestawie gotowych komponentów...). Możemy korzystać z wersji demonstracyjnych środowiska, pobranych ze strony firmy Embarcadero. W naszym przypadku jest to jednak nieuzasadnione, gdyż budowane programy będą wyłącznie aplikacjami dla konsoli.

Turbo Delphi Explorer – jako darmowe środowisko programistyczne oparte na języku Object Pascal zostało udostępnione przez firmę Borland w 2006 r. z licencją na 100 lat. Obecnie oferta nie jest aktualna — pobranie i zarejestrowanie aplikacji nie jest możliwe. W środowisku tym można tworzyć programy okienkowe i aplikacje konsolowe. Przykład przedstawiono na rysunku 6. Dla prostej aplikacji, zostanie utworzony projekt złożony z kilku plików:

```
2011-06-28  14:54               160 hello.dpr
2011-06-28  14:54               434 hello.cfg
2011-06-28  14:54             8˙417 hello.bdsproj
2011-06-28  14:54            44˙544 hello.exe
2011-06-28  14:54               435 hello.bdsproj.local
2011-06-28  14:59               102 hello.identcache
```

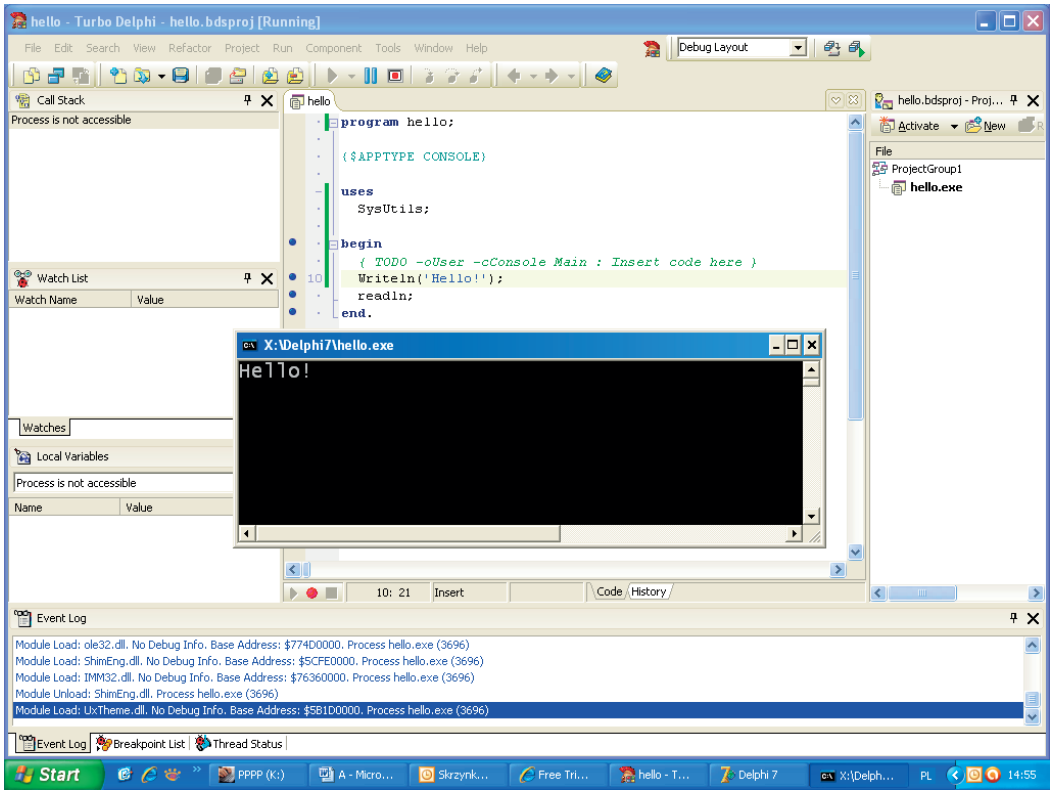
Kod źródłowy programu znajduje się w pliku *hello.dpr*

```
program hello;

{$APPTYPE CONSOLE}

uses
  SysUtils;

begin
  { TODO -oUser -cConsole Main : Insert code here }
  Writeln('Hello!');
  readln;
end.
```



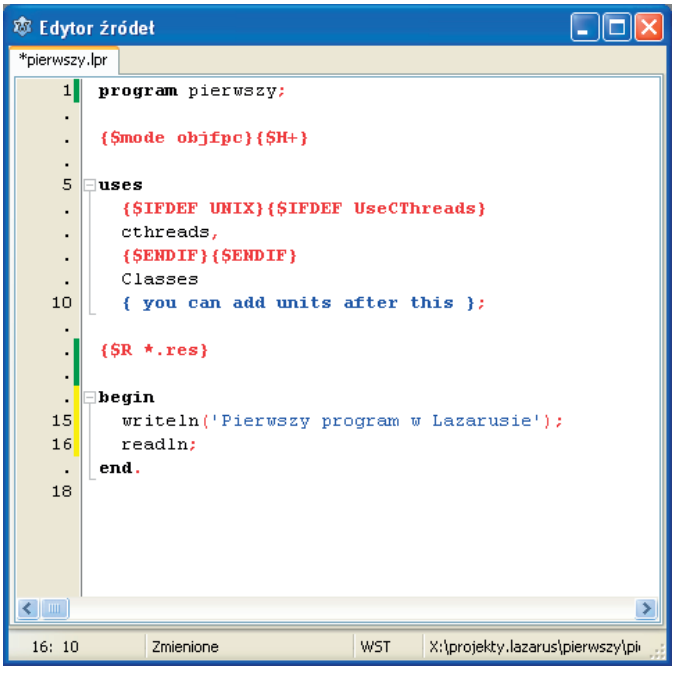
Rysunek 6. Prosta aplikacja konsolowa napisana w Turbo Delphi.

Lazarus

Lazarus jest wzorowanym na Delphi środowiskiem programistycznym. Wykorzystuje kompilator Free Pascal oraz bibliotekę komponentów *LCL* (*Lazarus Component Library*). Podobnie jak Free Pascal, program rozpowszechniany jest na licencji *GPL*, a biblioteki *LGPL*. Plik instalacyjny (dla Windows *lazarus-0.9.30-fpc-2.4.2-win32.exe*, ok. *72,6* MB) można pobrać ze strony projektu: *http://www.lazarus.freepascal.org/*. Środowisko jest dostępne dla wielu platform i systemów operacyjnych.

Plik instalacyjny kopiujemy na dysk *X:* do folderu *instalator* i instalujemy program na dysku *X:*. Pomijamy tworzenie menu Start, natomiast skrót utworzony na pulpicie kopiujemy na dysk *X:*. Na dysku *X:* tworzymy folder *projekty.lazarus* do przechowywania efektów naszej pracy oraz folder tymczasowy dla projektów *projekty.lazarus\temp*. Modyfikujemy skrót *Lazarus.lnk* wpisując ścieżkę "Rozpocznij w: *X:\projekty.lazarus*". Uruchamiamy środowisko i w konfiguracji (menu: *Środowisko* | *Opcje* | *Środowisko* | *Pliki* | *Katalog do budowania testowych projektów* wpisujemy ścieżkę *X:\ projekty.lazarus\temp*. Jesteśmy gotowi do pracy[3].

W menu wybieramy *Projekt* | *Nowy projekt* | *Program* i zapisujemy projekt (*Projekt* | *Zapisz projekt*...) w wybranym folderze, np: *X:\projekty.lazarus\pierwszy*. Wpisujemy do gotowego szablonu kod programu (rysunek 7). Wystarczy nacisnąć klawisz funkcyjny *F9*, by skompilować i uruchomić program.



Rysunek 7. Program w Pascalu — zapisany w środowisku Lazarus.

[3] Środowisko zajmuje 562 MB (733 MB na dysku) i mieści się w 1285 folderach – razem 16075 plików (dokonaliśmy pełnej instalacji).

Pierwszą niespodzianką będzie zawartość folderu *pierwszy* (bez komentarza!):

```
X:\projekty.lazarus\pierwszy
2011-06-28  18:04           137˙040 pierwszy.ico
2011-06-28  18:04           138˙128 pierwszy.res
2011-06-28  18:03             2˙183 pierwszy.lpi
2011-06-28  18:03               254 pierwszy.lpr
2011-06-28  18:04           409˙890 pierwszy.exe
X:\projekty.lazarus\pierwszy\backup
2011-06-28  17:57             2˙181 pierwszy.lpi.bak
2011-06-28  17:57               199 pierwszy.lpr.bak
X:\projekty.lazarus\pierwszy\lib
2011-06-28  18:03    <DIR>          i386-win32
X:\projekty.lazarus\pierwszy\lib\i386-win32
2011-06-28  18:04             4˙983 pierwszy.o
2011-06-28  18:04           138˙128 pierwszy.res
2011-06-28  18:04           138˙478 pierwszy.or
2011-06-28  18:04               347 pierwszy.compiled
```

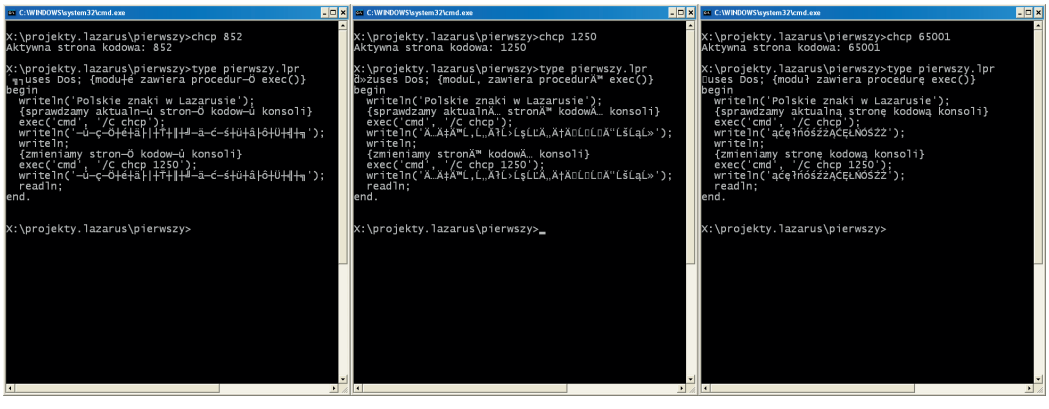
Poważniejszym problemem jest uzyskanie polskich znaków diakrytycznych. Interfejs programu nie umożliwia wprowadzania polskich znaków przy użyciu klawiatury programisty (prawy Alt+...). Wyjściem z sytuacji jest nauczenie się wprowadzania znaków przy użyciu kombinacji klawiszy: *Shift+~*, *znak* — znak oznacza odpowiednio *a*, *c*, *e*, *l*, *n*, *o*, *s*, *x* i *z* dla znaków *ą*, *ć*, *ę*, *ł*, *ń*, *ó*, *ś*, *ź* i *ż* lub *A*, *C*, *E*, *L*, *N*, *O*, *S*, *X* i *Z* dla wielkich liter *Ą*, *Ć*, *Ę*, *Ł*, *Ń*, *Ó*, *Ś*, *Ź* i *Ż*. Można nauczyć się kodów znaków i wprowadzać je z klawiatury numerycznej z naciśniętym lewym klawiszem *Alt*. Polskie znaki w edytorze (w kodzie źródłowym) to dopiero połowa sukcesu. Znaki w edytorze zapisane są w Unicode (UTF-8), a w konsoli obowiązuje strona kodowa 852. Rozwiązanie problemu polega na zmianie strony kodowej 852 na stronę kodową 1250, przy zastosowaniu procedury exec().

```
uses Dos; {moduł zawiera procedurę exec()}
begin
  writeln('Polskie znaki w Lazarusie');
  {sprawdzamy aktualną stronę kodową konsoli}
  exec('cmd', '/C chcp');
  writeln('ąćęłńóśźżĄĆĘŁŃÓŚŹŻ');
  writeln;
  {zmieniamy stronę kodową konsoli}
```

```
    exec('cmd', '/C chcp 1250');
    writeln('ąćęłńóśźżĄĆĘŁŃÓŚŹŻ');
    readln;
end.
```

Polecenie `exec('cmd', '/C chcp');` powoduje wykonanie w konsoli (`cmd`) komendy `chcp`, czyli sprawdzenia aktualnej strony kodowej (początkowo jest to 852). Polskie znaki diakrytyczne nie są wyświetlane poprawnie. Polecenie `exec('cmd', '/C chcp 1250');` natomiast wywołuje komendę `chcp 1250` zmieniającą stronę kodową. Reszta należy do kompilatora i systemu operacyjnego. Znaki diakrytyczne wyświetlane są poprawnie.

Próba ustawienia bezpośrednio strony kodowej 65001[4] nie daje spodziewanych rezultatów. Program przerywa działanie sygnalizując błąd. Możemy natomiast w konsoli obejrzeć kod programu zmieniając stronę kodową (rysunek 8).



Rysunek 8. Polskie znaki diakrytyczne w kodzie programu – listingi w różnych stronach kodowych

Przedstawione rozwiązanie można stosować budując aplikacje konsolowe w Delphi lub innych środowiskach zintegrowanych Pascala działających w Windows.

Uwaga. Używając procedury `exec()` do uruchamiania innych programów z poziomu naszego programu stosujemy dodatkowo bezparametrową procedurę `SwapVectors` (przed i po wywołaniu. procedury `exec()`). W naszym przykładzie tę procedurę pominęliśmy. Postępowanie takie zabezpiecza przed użyciem wektorów przerwań naszego programu przez uruchamiany proces.

Dev-C++

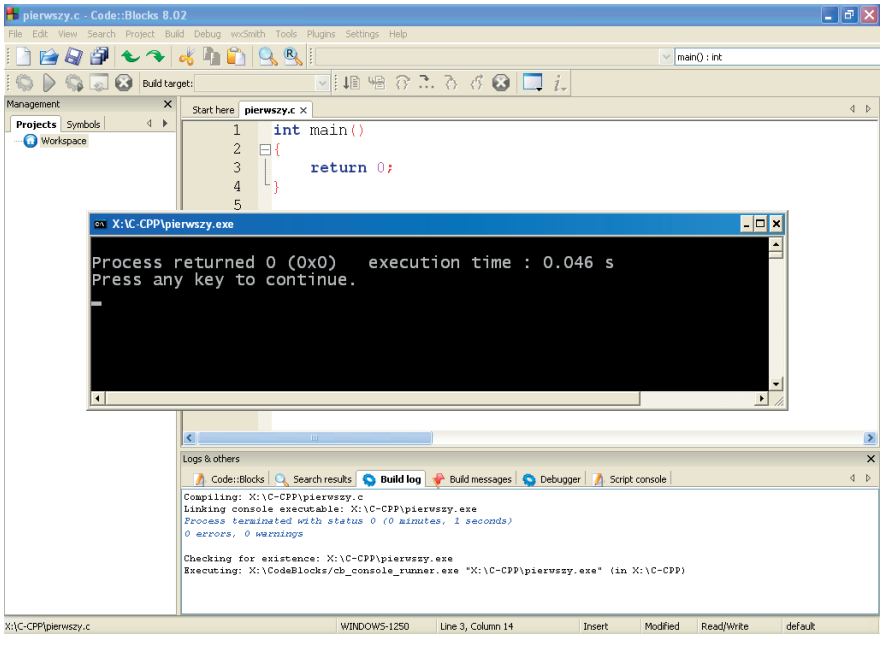
Dev-C++ 5 9.2 (4.9.9.2) jest w pełni funkcjonalnym darmowym środowiskiem programistycznym C/C++, z interfejsem w języku polskim, zawierającym wielookienkowy edytor kodu źródłowego z podświetlaniem składni, kompilator, debbuger i linker. Środowisko posiada narzędzia do tworzenia pakietów instalacyjnych napisanych programów. Dec-C++ umożliwia tworzenie

[4] Unicode (UTF-8)

aplikacji konsolowych oraz okienkowych (w WinApi). Możliwa jest też kompilacja i uruchamianie pojedynczych plików źródłowych (nie musimy tworzyć wieloplikowych projektów do prostych przykładów). W środowisku używany jest kompilator MinGW (GCC). Środowisko aktualnie nie jest już rozwijane. Z sieci pobieramy plik instalacyjny *devcpp-4.9.9.2_setup.exe* (*8,9* MB), zapisujemy w folderze *Instalator* i instalujemy na dysku *X:* w folderze *X:\Dev-Cpp*. Skrót do programu (*Dev-C++.lnk*) został utworzony w menu Start i stamtąd przenosimy go na dysk *X:*. Tworzymy folder *X:\C-CPP*, w którym będziemy umieszczali pliki źródłowe oraz modernizujemy odpowiednio skrót do aplikacji.

Uruchamiamy program, czytamy wskazówkę dnia — zawsze można dowiedzieć się czegoś nowego. Zamykamy okienko ze wskazówką i wybieramy z menu: *Plik* | *Nowy* | *Plik źródłowy* (skrót *Ctrl+N*). Wpisujemy kod źródłowy i kompilujemy program — menu: *Uruchom* | *Kompiluj* (skrót *Ctrl+F9*). Jeśli plik źródłowy nie był wcześniej zapisany, to będziemy musieli teraz to zrobić — wybrać folder i nazwę pliku. Ważny jest wybór rozszerzenia: *.c* dla plików kompilowanych kompilatorem języka C, *.cpp* — kompilacja w języku C++. Skompilowany program uruchamiamy poleceniem z menu *Uruchom* | *Uruchom* (skrót *Ctrl+F10*). Możemy też zrealizować dwie czynności, jedna po drugiej — z menu *Uruchom* | *Kompiluj i uruchom* (skrót *F9*).

Ponieważ okno konsoli jest zamykane, gdy program przestanie działać, to możemy mieć problemy z obejrzeniem efektów pracy. Nie ma dostępu do ekranu użytkownika, tak jak to było w Pascalu. Musimy zatem w naszym programie umieścić instrukcję zatrzymującą działanie programu do chwili naciśnięcia jakiegoś klawisza. Są różne rozwiązania tego problemu. Proponujemy wykorzystanie polecenia `pause` z konsoli, wywołanego przy pomocy funkcji `system()` — wymagane jest użycie pliku nagłówkowego `<cstdlib>` w C++ lub `<stdlib.h>` w C. Efekt działania programu przedstawiono na rysunku 9.



Rysunek 9. Przykład działania prostego programu w środowisku Dev-C++

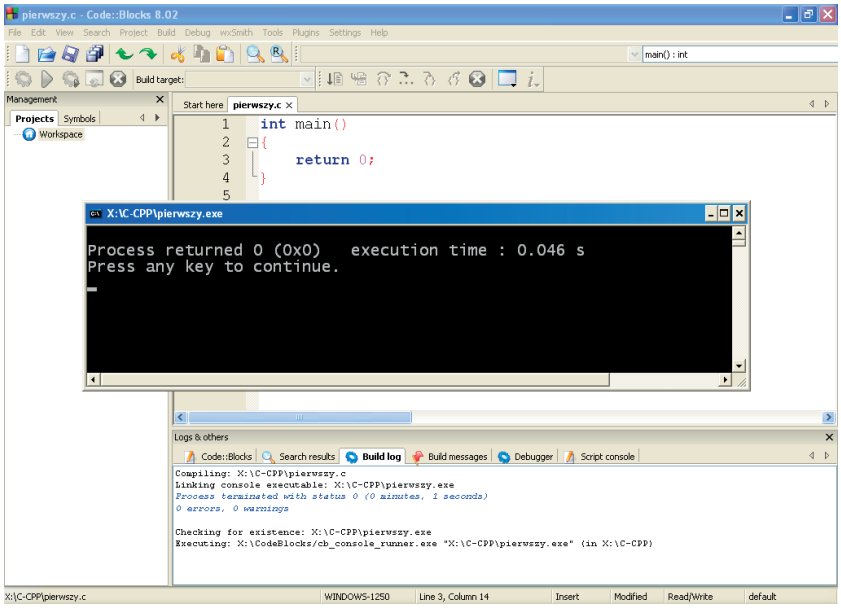
Pozostaje problem polskich znaków diakrytycznych. W edytorze polskie znaki kodowane są w charakterystycznej dla systemu Windows stronie kodowej *1250*. W konsoli mamy typowo ustawioną stronę kodową *852*. Najprostszym rozwiązaniem wydaje się używanie wspomnianej funkcji `system()` i polecenia zmieniającego stronę kodową konsoli: `chcp 1250`. Oczywiście powinniśmy też zmienić w konsoli czcionkę rastrową na czcionkę *TT Lucida Console*.

Czytelnik, poszukując Dev-C++ w sieci, może spotkać nowszą wersję wxDev-C++, czyli odmianę środowiska Dev-C++, stworzoną z myślą o projektowaniu aplikacji okienkowych opartych na bibliotece wxWidgets — ta wersja nie jest nam jednak potrzebna.

Code::Blocks

Code::Blocks jest wieloplatformowym IDE dla programistów C/C++. Wspiera wiele kompilatorów (GCC, Borland C++, Microsoft Visual C++ i inne). Posiada wbudowany, zaawansowany i wydajny debugger. Ponadto, umożliwia importowanie projektów z Dev-C++ oraz Microsoft Visual C++. Interfejs (w języku angielskim) jest prosty w obsłudze i intuicyjny, a rozbudowana pomoc ułatwia używanie programu. Kolorowanie składni, a także automatyczne uzupełnianie w trakcie pisania kodu, przyspieszają pracę. Środowisko jest nadal rozwijane przez dużą grupę programistów (*www.codeblocks.org*).

Mamy do wyboru samo środowisko IDE bez kompilatora lub środowisko z kompilatorem MinGW (tym samym co Dev-C++). Wybierzemy ten drugi wariant. Możemy wybrać wersję wcześniejszą *codeblocks-8.02mingw-setup.exe* (19775 kB) lub najnowszą *codeblocks-10.05mingw-setup.exe* (72293 kB). Kopiujemy wersję instalacyjną do folderu *Instalator* i instalujemy na dysku *X:* w folderze *CodeBlocks*. Skrót do programu (*CodeBlocks.lnk*) został utworzony w menu Start. Kopiujemy skrót na dysk *X:* i modyfikujemy folder startowy, np. *X:\C-CPP*, w którym będziemy zapisywali pliki źródłowe. Uruchamiamy aplikację, otwieramy nowy plik (menu: *File* | *New* | *Empty file* — skrót *Ctrl+Shift+N*). W kodzie nie musimy używać funkcji `system("pause")` — środowisko nie zamyka konsoli z efektami pracy programu, czeka aż użytkownik naciśnie dowolny klawisz. Natomiast będziemy używali funkcji `system("chcp 1250")` w celu poprawnego wyświetlania polskich znaków w konsoli. Przykład działania prostego programu zapisanego w języku C widzimy na rysunku 10.



Rysunek 10. Przykład działania prostego programu w środowisku Code::Blocks.

Borland C++ 5.5

Kompilator firmy Borland, dostępny na stronie *http://forms.embarcadero.com/forms/BCC32CompilerDownload* możemy pobrać (*freecommandLinetools.exe*, *8727* KB) i zainstalować na dysku *X:* w folderze *BCC55*. Jest do bardzo dobre narzędzie do kompilowania programów w języku C i C++ w linii komend (w konsoli).

Do pliku wsadowego *xDriveOn.bat* dopiszemy linię `path X:\BCC55\Bin;%PATH%`. Będziemy mieli dostęp do kompilatora *bcc32.exe* z dowolnego folderu. Wpisujemy w linii komend *bcc32* i odczytujemy z konsoli podstawowe informacje o składni komendy kompilatora:

```
Borland C++ 5.5.1 for Win32 Copyright (c) 1993, 2000 Borland

Syntax is: BCC32 [ options ] file[s]     * = default; -x- = turn switch
x off

  -3    * 80386 Instructions        -4      80486 Instructions
  -5      Pentium Instructions      -6      Pentium Pro Instructions
  -Ax     Disable extensions        -B      Compile via assembly
  -C      Allow nested comments     -Dxxx   Define macro
  -Exxx   Alternate Assembler name  -Hxxx   Use pre-compiled headers
  -Ixxx   Include files directory   -K      Default char is unsigned
  -Lxxx   Libraries directory       -M      Generate link map
  -N      Check stack overflow      -Ox     Optimizations
```

```
-P      Force C++ compile          -R      Produce browser info
-RT   * Generate RTTI              -S      Produce assembly output
-Txxx   Set assembler option       -Uxxx   Undefine macro
-Vx     Virtual table control      -X      Suppress autodep. output
-aN     Align on N bytes           -b    * Treat enums as integers
-c      Compile only               -d      Merge duplicate strings
-exxx   Executable file name      -fxx    Floating point options
-gN     Stop after N warnings      -iN     Max. identifier length
-jN     Stop after N errors        -k    * Standard stack frame
-lx     Set linker option          -nxxx   Output file directory
-oxxx   Object file name          -p      Pascal calls
-tWxxx  Create Windows app         -u    * Underscores on externs
-v      Source level debugging     -wxxx   Warning control
-xxxx   Exception handling         -y      Produce line number info
-zxxx   Set segment names
```

Na podstawie tej informacji oraz analizy rozmieszczenia plików w folderze BCC55 tworzymy plik wsadowy *kompilacja.bat*:

```
bcc32 -I"X:\BCC55\include" –L"X:\BCC55\lib" %1
```

Od tej chwili poleceniem `kompilacja nazwa_pliku.c` lub `kompilacja nazwa_pliku.cpp` możemy kompilować w konsoli pliki źródłowe programów zapisanych w języku C lub C++. Plik wynikowy będzie miał taką samą nazwę jak plik źródłowy (można nazwę zmienić dodając opcję: `-exxx` *Executable file name*) i zostanie umieszczony w bieżącym folderze (to też można zmienić — opcja: `-nxxx` *Output file directory*).

Oczywiście nie musimy kompilować programów w linii komend — kompilator można dołączyć do środowiska IDE, np. Code::Blocks. Pozostawiamy to jako ćwiczenie Czytelnikowi.