1. Podstawy programowania

Języki używane na co dzień, by komunikować się z innymi ludźmi, to – krótko ujmując – system znaków i zasad służących porozumiewaniu się. W takim znaczeniu języki są otwarte i dynamiczne. Codziennie powstają nowe słowa i pojęcia. Natomiast zasady gramatyczne zmieniają się rzadko. Czym są języki programowania? To również zbiór słów i zasad składniowych. Co więcej, te zasady obowiązują tylko dla danej wersji kompilatora. Zapoznaj się z charakterystyką języka maszynowego, asemblera oraz języka wysokiego poziomu.

Język maszynowy jest zrozumiały praktycznie tylko przez komputery; jego postać zależy od typu procesora. Jest on zbudowany ze słów binarnych odpowiadających poszczególnym rozkazom procesora. Nie używa się go do programowania. Zazwyczaj powstaje w procesie kompilacji programu pisanego w innym języku programowania.

Asembler to tzw. język niskiego poziomu (używa bezpośrednio rozkazów mikroprocesora); trudny w użyciu – częściej stosują go elektronicy programujący mikrokontrolery itp. Składnia i lista rozkazów są ściśle związane z typem procesora; rozkazy asemblera zapisuje się w postaci mnemoników (kilkuliterowych skrótów ich angielskich nazw, np. ADD, MOV); pozostałe elementy języka są poleceniami sterującymi ustalającymi np. adres fizyczny, nazwy zmiennych. Kompilację programu

w języku asembler nazywa się asemblacją, a proces odwrotny – przejście od kodu maszynowego do programu w asemblerze – deasemblacją (lub dezasemblacją).

W języku wysokiego poziomu poszczególne instrukcje są zapisane za pomocą zrozumiałych i łatwych do zapamiętania słów. Instrukcja może realizować od kilku do nawet kilku tysięcy poleceń z listy instrukcji procesora. Programista nie musi znać poszczególnych funkcji procesora i jego urządzeń peryferyjnych ani wiedzieć, jak będzie wyglądało wykonywanie jego programu – musi tylko zadbać o to, by jego działanie w pełni realizowało założenia i algorytm. Programy napisane w językach wysokiego poziomu są łatwe w implementowaniu na różne rodzaje komputerów.

Najpowszechniej stosowane języki są uniwersalne. Używa się ich do pisania programów z różnych dziedzin i dla różnych systemów operacyjnych.

W latach 80. i 90. XX w. Pascal był jednym z najpopularniejszych języków do zastosowań profesjonalnych. Obecnie dzięki swojej składni dobrze spełnia swoją funkcję edukacyjną. Jest typowym przedstawicielem języków strukturalnych. Oznacza to, że program zbudowany jest z podprogramów – mniejszych bloków realizujących określone zadania, a za ich uruchamianie odpowiada program główny. Oto przykład kilku instrukcji napisanych w języku Pascal:

```
Var
      a,b,c,x : Integer;
begin
  a:=1;
  b:=2;
  c:=a + 2;
  x:=a + b + c;
end.
```

Jednym z najbardziej popularnych języków programowania na świecie dla profesjonalistów i amatorów jest język C++. Nadaje się on do tworzenia programów z każdej dziedziny – od sterujących do skomplikowanych systemów biurowych i operacyjnych. Programy w języku C++ są zazwyczaj kompilowane bezpośrednio do postaci kodu maszynowego, co oznacza, że są bardzo szybkie w działaniu. Daje to programiście możliwość pisania programów odnoszących się bezpośrednio do poleceń procesora bez używania asemblera. A oto przykład instrukcji napisanych w języku C++.

```
int a,b,c,x;
main ()
{
  a=1;
  b=2;
  c=a + 2;
  x=a + b + c;
}
```

1 Edytor i kompilator Dev-C++

Jak napisać program, który policzy głosy w wyborach do samorządu uczniowskiego? Najpierw trzeba się zastanowić, jak ten program powinien działać, stworzyć jego algorytm, a następnie zapisać ten program w postaci kodu źródłowego. Na końcu można go przetestować i sprawdzić jego poprawność. Przedstawiliśmy ten proces w wielkim uproszczeniu. Do poszczególnych elementów tworzenia programu jeszcze wrócimy, a teraz zajmiemy się kodem źródłowym i kompilacją.

Kod źródłowy i kompilator

Kod źródłowy to tekst programu przed procesem kompilacji, napisany w jednym z języków programowania. Podlega ochronie prawnej i stanowi jedną z najpilniej strzeżonych tajemnic każdej firmy tworzącej oprogramowania komercyjne. Kody źródłowe często są także opatentowane. Możliwość korzystania z kodów innych programistów (lub firm, jeśli są ich właścicielami) zależy od rodzaju licencji, którą są objęte tekst programu i rozwiązania w nim stosowane.

Kompilatory zwykle mają zaimplementowany własny edytor służący do pisania kodów źródłowych. Edytory takie posiadają opcje dotyczące procesu kompilacji i uruchamiania programów oraz automatycznie formatują tekst programu, zwiększając w ten sposób jego czytelność.

W procesie kompilacji w pierwszej kolejności następuje sprawdzenie składni kodu źródłowego – czy nie popełniono błędów polegających na użyciu niewłaściwych znaków lub słów itp. Gdy cały kod źródłowy jest bezbłędny, wówczas kompilator przystępuje do zamiany go na kod maszynowy, którego instrukcje będą już bezpośrednio wykonywane przez mikroprocesor. Kod maszynowy zawiera binarne odpowiedniki rozkazów z listy rozkazów danego mikroprocesora. Są one rozpoznawane i dekodowane przez dekoder rozkazów i wykonywane przez inny blok procesora (najczęściej jednostkę arytmetyczno-logiczną ALU).

Programy dla systemu Windows są zapisywane z rozszerzeniem .com lub .exe. Rozszerzenie .com mają pliki zawierające tylko kod maszynowy programu. W plikach typu .exe są zapisywane jeszcze inne informacje, np. dotyczące wywoływania plików z danymi i bibliotek .dll. Biblioteki takie często są związane ze sterownikami urządzeń peryferyjnych.

Implementacja
Wdrożenie, realizacja – to proces pisania programu w konkretnym języku programowania lub wynik w postaci działającego programu.

Kompilacja
Proces tłumaczenia kodu napisanego w jednym języku (źródłowym) na kod w innym języku (wynikowym).

Dekodowanie
Proces zamiany zakodowanej informacji na zrozumiałą dla urządzenia czy programu.

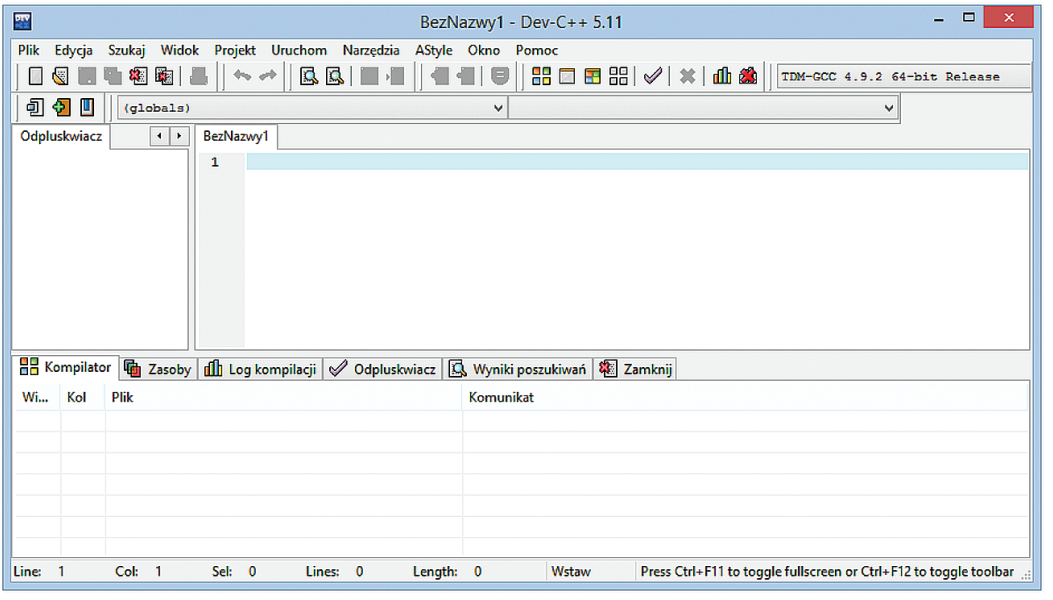
Kompilator Dev-C++

Kompilatorów C/C++ jest kilka. Każdy z nich oferuje wiele możliwości. Do najbardziej znanych zalicza się zamknięte oprogramowanie Microsoft Visual Studio, także otwarte oprogramowania oparte na licencji GPL: Code::Blocks czy Dev-C++, z którego będziemy korzystać.

Uruchom program za pomocą ikony (rys. 1.1.) i z menu **Plik** wybierz polecenie **Nowy** oraz **Plik źródłowy.**

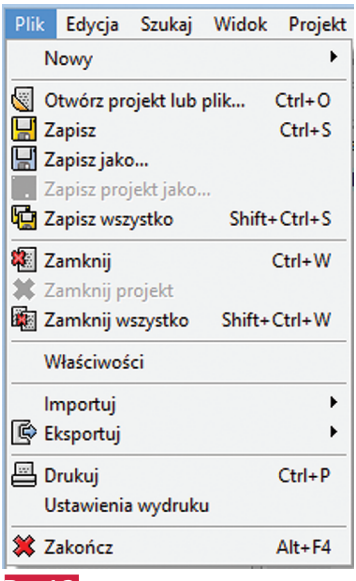
Rys. 1.1. Ikona do uruchamiania Dev-C++

Przy otwieraniu okna edycyjnego edytor automatycznie nadał plikowi nazwę **BezNazwy1**. Kody źródłowe należy zapisywać pod nazwami informującymi o przeznaczeniu programu i jego funkcjach. Okno (rys. 1.2.) przedstawia widok edytora kodu źródłowego kompilatora Dev-C++.

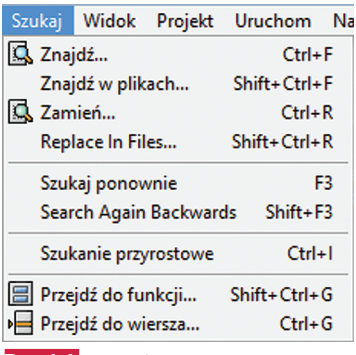


Rys. 1.2. Widok okna edytora kodu źródłowego kompilatora Dev-C++

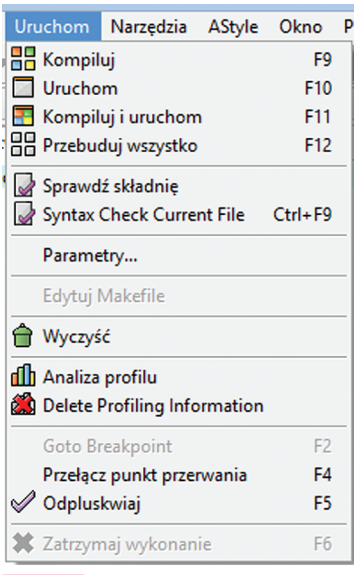
W górnej części okna znajduje się menu. Niektóre opcje z menu przedstawiamy poniżej. Pod menu można umieścić zestawy pasków narzędzi. Widoczność poszczególnych zestawów zależy od indywidualnych ustawień definiowanych z poziomu menu. Główna część ekranu to miejsce na wpisywanie kodu (tzw. obszar roboczy).



Rys.1.3. Menu Plik



Rys.1.4. Menu Szukaj



Rys.1.5. Menu Uruchom

W dolnej części okna znajdują się zakładki przydatne podczas kompilacji i uruchamiania programu. Do minimalizacji wszystkich zakładek służy ostatnia z nich – **Zamknij**.

Menu Plik

Na rysunku 1.3. przedstawiono otwarte menu **Plik**. Większość opcji zapewne znasz i nie trzeba ich dokładniej omawiać.

Nowy – otwiera nowy plik.

Otwórz projekt lub plik... – otwiera istniejący plik lub projekt z kodem źródłowym.

Zapisz – umożliwia zapisanie aktualnie otwartego pliku z kodem źródłowym. Tej opcji (lub jej skrótu **Ctrl+S**) należy używać często. Jeśli w lewym górnym rogu pojawi się znak [*], to oznacza, że zmiany nie zostały zapisane.

Zapisz jako... – umożliwia zapisanie pliku jako kodu źródłowego C/C++ (np. z rozszerzeniem .cpp), pliku nagłówkowego (np. z rozszerzeniem hpp) czy pliku zasobu (np. z rozszerzeniem .rc).

Importuj – umożliwia import projektu z MS Visual C++.

Eksportuj – umożliwia eksport do plików w postaci HTML (HyperText Markup Language – hipertekstowy język znaczników) czy RTF (Rich Text Format).

Menu Szukaj

Menu **Szukaj** (rys. 1.4.) powinno być ci znane.

Menu Uruchom

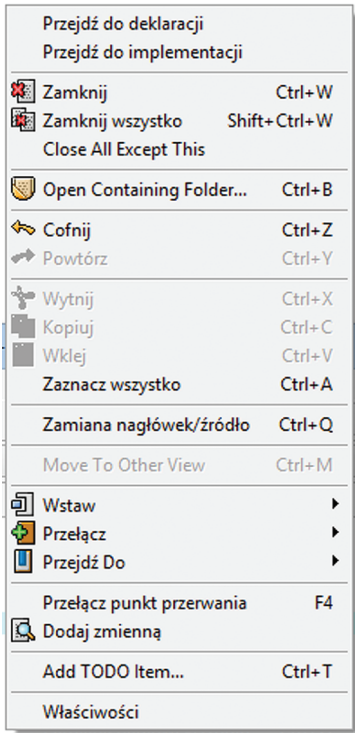
W menu **Uruchom** (rys. 1.5.) znajdują się opcje związane z kompilacją programu.

Kompiluj – kompiluje program bez uruchamiania go. Informacje o procesie kompilacji są zawarte w dolnej zakładce **Logi kompilacji**. Informacje o błędach widać w zakładce **Kompilator**.

Uruchom – uruchamia ostatnio skompilowany program; podczas uruchamiania uwzględnia ewentualne parametry określone w opcji **Parametry**. Polecenie **Kompiluj i uruchom** kompiluje program, a jeśli zostanie on poprawnie skompilowany – uruchamiania go.

Przebuduj wszystko – kompiluje wszystkie moduły wchodzące w skład programu.

Sprawdź składnię – weryfikuje składnię kodu źródłowego. Ewentualne błędy umieszcza w dolnej zakładce **Kompilator**.



Rys. 1.6. Menu kontekstowe

Parametry – tu określa się parametry wywołania programu.
Wyczyść – powoduje usunięcie skompilowanej wersji programu.
Przełącz punkt przerwania i Odpluskwiaj – opcje związane z procesem krokowego uruchamiania programu (debug), linia po linii, co jest bardzo przydatne w przypadku analizy kodu i poszukiwania błędu.
Pod prawym przyciskiem myszy znajduje się menu kontekstowe (rys. 1.6).
Większości opcji ma swoje odpowiedniki w postaci skrótów klawiszowych.
Jak zapewne zauważyliście, mimo że edytor Dev-C++ jest dostępny w polskiej wersji językowej, to nie wszystkie opcje zostały przetłumaczone.

Debugowanie
Proces, który ma na celu redukcję błędów. Polega na wykonywaniu kodu w sposób kontrolowany.

Zapamiętaj

- Kod źródłowy podlega ochronie prawnej i jest jedną z najpilniej strzeżonych tajemnic każdej firmy tworzącej oprogramowania komercyjne.
- Dev-C++ jest jednym z dostępnych środowisk programistycznych dla języka C++.
- W Dev-C++ większość ważniejszych opcji ma polskie tłumaczenia.
- Skrót Ctrl+W zamyka aktywne okno z kodem źródłowym.

Zadania

1. Wyszukaj w internecie program instalacyjny Dev-C++. Zainstaluj go na swoim komputerze.
2. Uruchom Dev-C++. Przejrzyj zawartość menu pod kątem skrótów klawiszowych oraz ikonek na paskach narzędzi. Przećwicz użycie niektórych skrótów i ikonek.
3. Przećwicz zapisywanie plików z różnymi rozszerzeniami.

2 Edytor programu Dev-C++

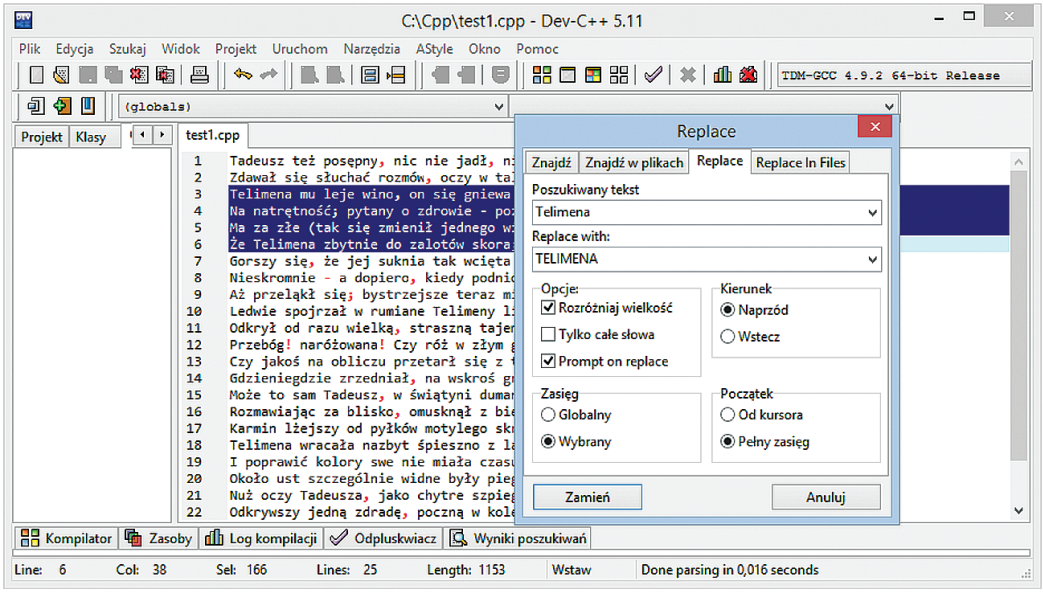
Masz już za sobą pierwszy kontakt z kompilatorem, którego będziemy używać podczas nauki programowania. Jednak zanim rozpoczniesz programować, poznaj dobrze edytor zaimplementowany w Dev-C++.

Edycja tekstu w Dev-C++

Do pisania kodów programów z powodzeniem można używać najprostszych edytorów tekstowych, np. Notatnika – będącego składnikiem większości systemów operacyjnych. Później jednak trzeba użyć kompilatora, który skompiluje napisany program i umożliwi jego uruchomienie. Jednak edytory specjalistyczne są wyposażone w funkcje usprawniające pisanie programów – rozpoznają niektóre słowa, mogą podpowiadać składnię, instrukcje itd.

Uruchom program Dev-C++, utwórz nowy plik i zapisz go pod nazwą test1.cpp. Następnie wklej (użyj opcji **Edycja→Wklej** lub **Ctrl+V**) do niego tekst, np. fragment wiersza czy powieści. My posłużymy się fragmentem tekstu *Pana Tadeusza* Adama Mickiewicza.

Zwróć uwagę, że niektóre symbole (przecinek, średnik, nawias) są w pewien sposób wyróżnione. Dlaczego? Edytor tekstowy zaimplementowany w kompilatorze Dev-C++ rozpoznaje pewne składniki tekstu i koloruje je dla zwiększenia czytelności programu. Kolory-

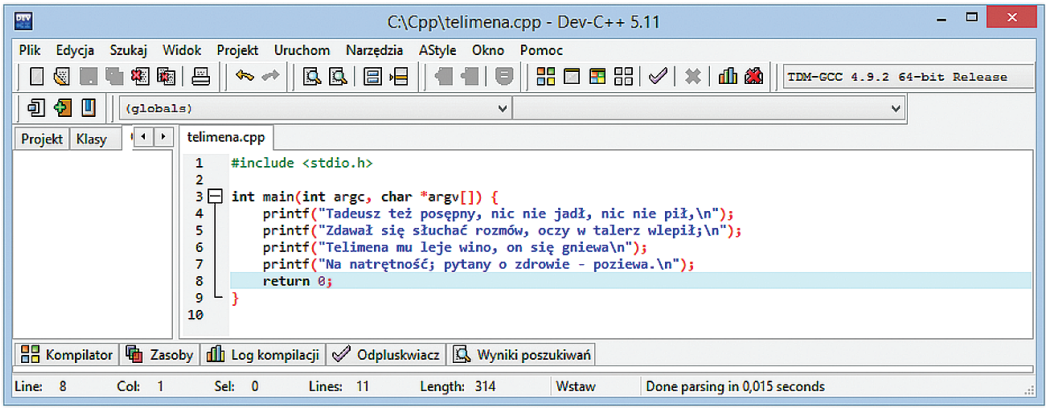


Rys. 2.1. Dev-C++ – zamiana tekstu w zadanym obszarze

stykę można konfigurować na różne sposoby w opcji **Narzędzia → Opcje edytora**, w zakładce **Kolorowanie składni**.
Zaznacz myszką fragment tekstu i wszystkie zapisy słowa „Telimena" wymień na „TELIMENA". Użyj do tego celu opcji **Szukaj → Zamień** (rys. 2.1.). Odszukaj i ustaw odpowiednią opcję tak, by program za każdym razem pytał, czy rzeczywiście wymienić wyrazy. Sprawdź, jak działa zamiana wyrazów przy różnych ustawieniach opcji: **Opcje** (szczególnie **Rozróżniaj wielkość liter**), **Kierunek**, **Zasięg** oraz **Początek**. Umiejętności związane z wyszukiwaniem tekstów przydadzą się do ujednolicenia kodu programu. Przećwicz jeszcze użycie przydatnej kombinacji klawiszy działającej na bloku tekstu zaznaczonym myszką:
Tab - przesuwa zaznaczony blok w prawo.
Shift + Tab - przesuwa zaznaczony blok w lewo.
Na koniec zapisz plik pod nazwą **telimena.cpp.** (przy użyciu **Plik → Zapisz jako**).

Kod źródłowy

Aby poznać nie tylko edycyjne możliwości programu Dev-C++, wykorzystamy posiadany już tekst do stworzenia programu komputerowego. W tym celu użyjemy kilku nowych słów, np.: `include`, `main`, `int`, `agrc`, `char`, `argv`, `printf`, `return`. Poza nimi potrzebne jeszcze będą nawiasy okrągłe, klamrowe i kwadratowe, przecinki, średniki, podwójne apostrofy, ukośnik lewy (`\`), kratka (ang. *hash* `#`) oraz znaki mniejszości i większości. Utwórz teraz plik zgodnie z rysunkiem 2.2. Zrób to bardzo uważnie, gdyż nawet mały błąd sprawi, że program nie będzie działał. Zwróć uwagę, gdzie są użyte okrągłe czy klamrowe nawiasy prawe i lewe, w którym miejscu są umieszczone podwójne apostrofy, przecinki i średniki.
Zauważ, że wiersz `#include<stdio.h>` został dodany na początku. Kolejny wiersz (ze słowem `main`) zawiera znaczki i wyrazy



Rys. 2.2. Fragment wiersza „zamieniony" w program

w dziwnej kombinacji (`*`, `[`, `]`). Następnie zaczyna się pewien blok od nawiasu klamrowego. Na początku każdej kolejnej linii z treścią wiersza znajduje się słowo `printf(”`, a na końcu – `”)` i średnik, a dodatkowo – kombinacja znaków `\n`. Pod tekstem wiersza jest umieszczone słowo `return`. Całość zamyka nawias klamrowy `}`. Wcięcie można uzyskać, zaznaczając blok ze słowami `printf` i przesuwając całość w prawo za pomocą klawisza **Tab**.

Kompilacja kodu źródłowego

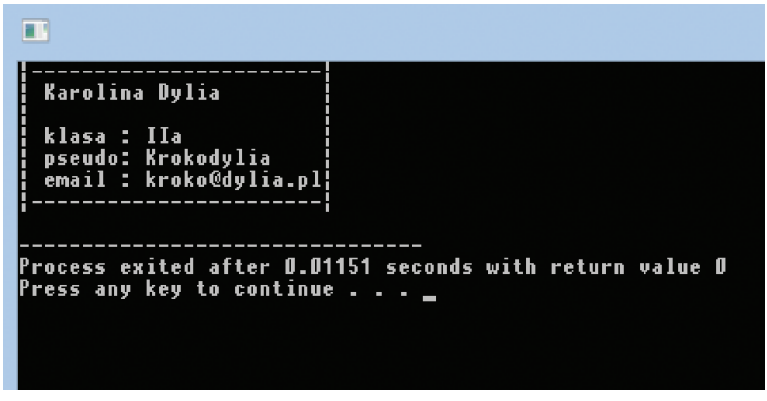
Został już napisany pełen kod źródłowy pierwszego programu. Teraz wybierz opcję **Uruchom → Kompiluj** (**F9**). W dolnym okienku, w zakładce **Kompilator** powinny pojawić się pewne informacje. Jeśli kompilacja zostanie przeprowadzona poprawnie, powstanie program. Otwórz go za pomocą opcji **Uruchom → Uruchom** (**F10**). Po tym na ekranie pojawi się fragment wiersza.

Popatrz teraz jeszcze raz na efekt działania programu. Zwróć uwagę na to, że:

- polskie znaki nie są wyświetlane poprawnie; aby je uzyskać, trzeba to oprogramować, do czego jest potrzebna praktyka programistyczna;
- po uruchomieniu programu na ekranie wyświetlają się cztery wiersze; jest to zasługa dodania `\n` wewnątrz polecenia `printf`.

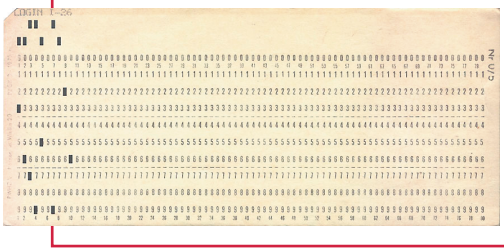
Wizytówka

Potrafisz już napisać program wyświetlający tekst na ekranie. Teraz zaprojektuj samodzielnie swoją wizytówkę i napisz program, który wyświetli ją na ekranie. Przykład podano na rysunku 2.3.



Rys. 2.3. Dev-C++ – przykład wizytówki

CIEKAWOSTKA



Pierwszym nośnikiem do zapisu danych była karta perforowana. Została wymyślona przez Josepha Marie Jacquarda w 1801 r. i zastosowana w krosnach do produkcji tkanin o skomplikowanym splocie (tkaniny żakardowe). W XX w. za pomocą kart perforowanych wprowadzano programy do komputerów. Był to kartonik o wymiarach 187,325 mm na 82,55 mm, który miał dziurki w określonych miejscach.

Zapamiętaj

- Dev-C++ może pełnić funkcję edytora plików tekstowych.
- Jak najczęściej używaj kombinacji klawiszy Ctrl+S, aby na bieżąco zapisywać swoją pracę.
- Przydatne kombinacje klawiszy to Tab - przesuwa zaznaczony blok w prawo oraz Shift + Tab - przesuwa zaznaczony blok w lewo.
- Niektóre ze słów, które rozumie kompilator, to: main, printf, return. Poza tym są w nim używane: nawiasy okrągłe, klamrowe oraz kwadratowe, podwójny apostrof, przecinek, gwiazdka, średnik, znaki \, # oraz mniejszości i większości.
- Program kompilujemy klawiszem F9.
- Program uruchamiamy klawiszem F10.
- Program kompilujemy i uruchamiamy klawiszem F11.

Zadania

1. Otwórz plik telimena.cpp. Utwórz nowy plik i zapisz go pod nazwą test2.cpp. Zaznacz i skopiuj fragment z pliku telimena.cpp. Wklej ten fragment do pliku test2.cpp i go zapisz.

2. Otwórz plik telimena.cpp. Otwórz plik test2.cpp. Zaznacz i wytnij fragment z pliku telimena.cpp. Wklej ten fragment do pliku telimena.cpp. Zapisz oba zmienione pliki.

3. Otwórz zapisany plik telimena.cpp. Przećwicz pozostałe opcje z menu Szukaj. Wyszukaj ciąg znaków (Znajdź), ciąg znaków w plikach (Znajdź w plikach...) i zamień ciąg znaków w plikach (Replace In Files...).

4. Otwórz plik telimena.cpp. Otwórz plik test2.cpp. Przećwicz wybrane opcje z menu Okno - zmieniaj rozmiary okna, przechodź między oknami.

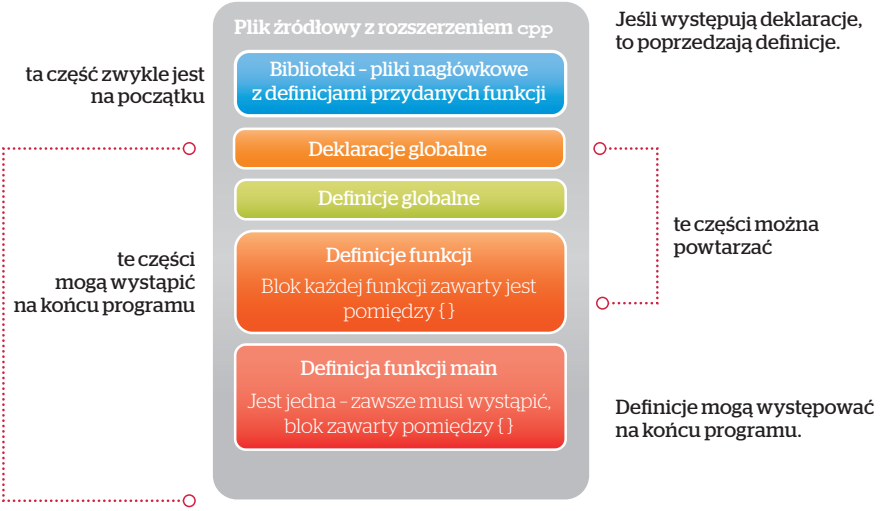
5. Napisz program wyświetlający na ekranie twój plan lekcji.

3 Tworzenie kodu źródłowego i budowa programu

Czas zacząć pisać pierwsze programy, które nie będą ograniczały się do wyświetlania tekstu na ekranie. W kolejnych rozdziałach nauczysz się przekształcać specyfikację algorytmu i opis metody w działający program. Dowiesz się, w jaki sposób buduje się kod programu w C/C++, jak kompiluje się napisany program i go uruchamia, a w przypadku błędu – debuguje (czyli uruchamia program krokowo).

Struktura blokowa programu

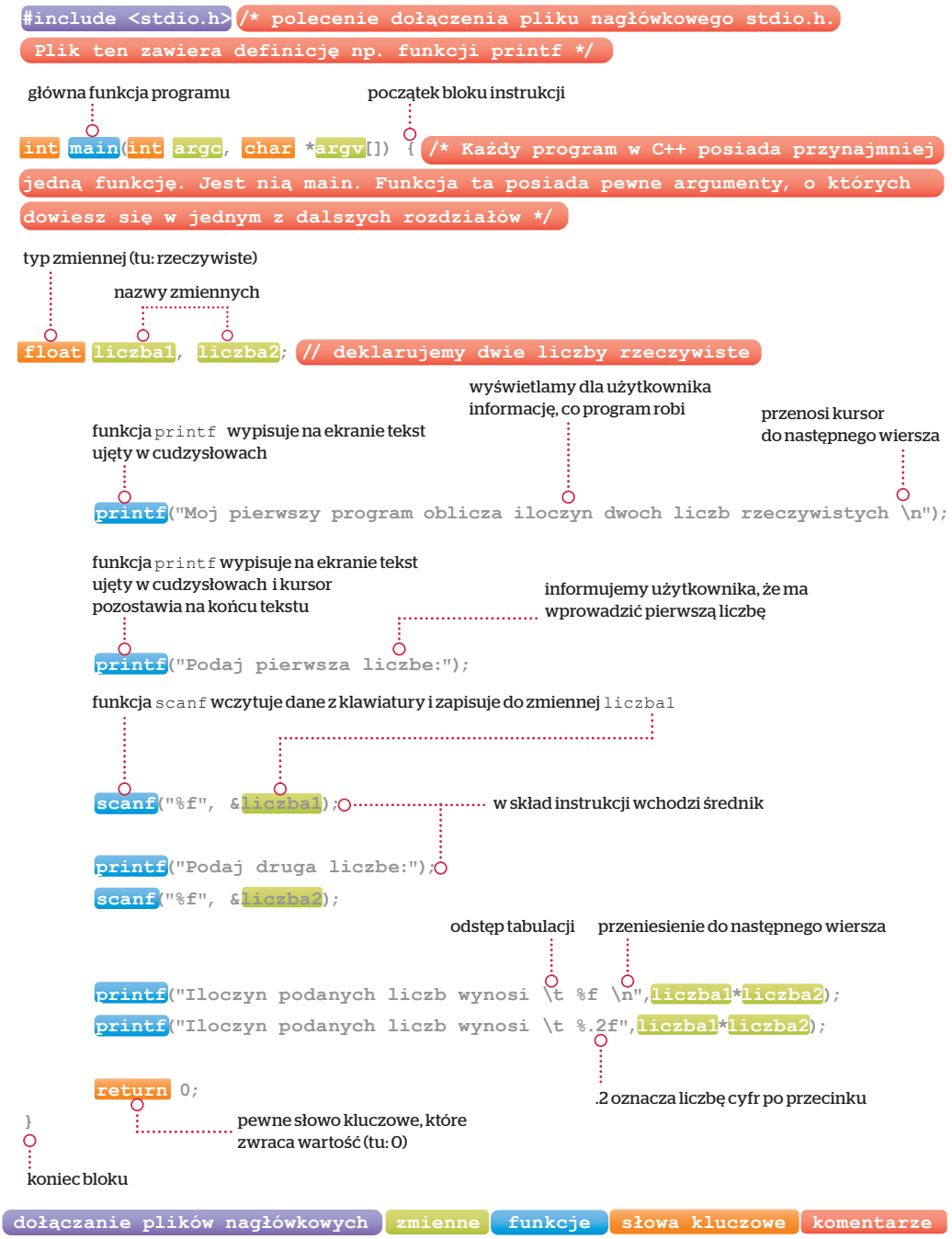
Kod programu (zwany też kodem źródłowym) napisany w języku C/C++ umieszcza się w pliku z rozszerzeniem .cpp. Każdy program ma określoną budowę (strukturę programu) i składa się z pewnych stałych elementów. Spójrz na poniższy rysunek stałych elementów (rys. 3.1.).



Rys. 3.1. Struktura programu w C/C++

Poszczególne elementy języka omówimy na przykładzie programu, którego zadaniem – po wpisaniu przez użytkownika dwóch liczb z klawiatury – będzie wyświetlenie ich iloczynu na ekranie. Uruchom Dev-C++, utwórz nowy plik (**Nowy → Plik → Nowy plik źródłowy** lub **Ctrl+N**) i wpisz treść programu podanego poniżej. Zapisz go w pliku o nazwie **Pierwszy.cpp** (**Plik → Zapisz jako**).

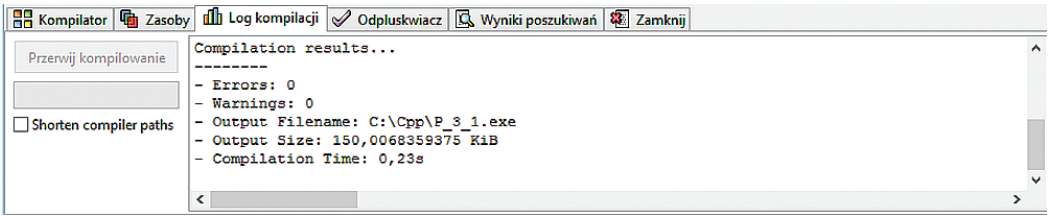
Zanim go skompilujesz, wyjaśnimy w skrócie, co oznaczają poszczególne linie. Zwróć uwagę na wcięcia w tekście, mające na celu wyróżnianie poszczególnych bloków programu (rys. 3.2.).



Rys. 3.2. Opis poszczególnych linii programu

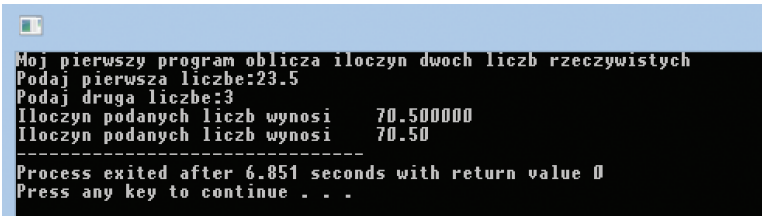
Kompilacja i uruchomienie programu

Użyj opcji **Uruchom → Kompiluj** albo klawisza **F9**, aby skompilować program. Po skompilowaniu otrzymasz komunikat o kompilacji zakończonej sukcesem (rys. 3.3.). Jeśli nie, to wówczas przeczytaj wyświetloną informację.



Rys. 3.3. Komunikat informujący o poprawnie zakończonej kompilacji

Taki program możesz już uruchomić (**Uruchom → Uruchom** lub **F10**). Jeśli w trakcie działania programu podasz liczby 23.5 (funkcję przecinka spełnia kropka) oraz cyfrę 3, to otrzymasz efekt pokazany na rysunku 3.4.



Rys. 3.4. Ekran programu uruchomionego za pomocą Dev-C++

Zwróć uwagę, że pod wyświetlonym wynikiem iloczynu podanych liczb automatycznie zostały dodane komunikaty:

```
Process exited after 6.851 seconds with return value 0
Press any key to continue …
```

Jest to informacja dotycząca czasu wykonywania programu oraz wskazówka, co należy zrobić, aby kontynuować (w tym przypadku należy nacisnąć klawisz lub zamknąć okno).

Komunikaty te pojawiają się w zależności od ustawionego parametru **Pause Program after return option** w opcji **Narzędzia → Opcje środowiska → Ogólne**. Skompilowany program (z rozszerzeniem .exe) możesz uruchomić, nie korzystając z Dev-C++.

W 1983 r. Rick Mascitti zaproponował nazwę języka C++. Nazwa ta została po raz pierwszy użyta poza laboratorium naukowym. Miała ona nawiązywać do faktu, że język ten jest rozszerzeniem języka C. Nazwa wcześniejsza to „C z klasami". Użycie w nazwie operatora inkrementacji (czyli zwiększenie liczby o 1) „++" jest symbolicznym zaznaczeniem, że jest to język o znacznie większych możliwościach.

Zapamiętaj

- Na początku programu w C/C++ z reguły są dodawane informacje o dołączanych plikach nagłówkowych (tzw. bibliotekach).
- Każdy program napisany w języku C/C++ zawiera przynajmniej jedną funkcję o nazwie main. Funkcja ta zawiera główny kod programu.
- Bloki instrukcji są umieszczane pomiędzy nawiasami klamrowymi `{...}`.
- Program kompilujemy za pomocą klawisza F9, a uruchamiamy za pomocą F10.
- Każda instrukcja kończy się średnikiem (wchodzi w jej skład). Pełni on funkcję ogranicznika.
- Stosuj wcięcia w kodzie programu. Sprzyja to czytelności programu.
- Stosuj komentarze, czyli fragmenty kodu ignorowane przez kompilator. Są przydatne programistom do opisywania źródeł - tego, w jaki sposób działa dany fragment programu. Komentarz jednowierszowy rozpoczyna się dwoma prawymi ukośnikami (slash): `//`. Treść dłuższego komentarza umieszczamy pomiędzy prawymi ukośnikami z gwiazdką: `/*treść komentarza*/`.

Zadania

1 Usuń znaki \n w programie Pierwszy.cpp w funkcji printf, w której są umieszczone. Dodaj je w funkcjach `printf`, które ich pierwotnie nie miały. Sprawdź efekty tych działań.

2 W programie Pierwszy.cpp popełnij błąd i dokonaj kompilacji. Możesz:

- usunąć wiersz z #include,
- usunąć jeden z nawiasów w instrukcjach printf lub scanf,
- usunąć jeden z podwójnych apostrofów,
- zrobić błąd literowy w słowach kluczowych, nazwach funkcji lub zmiennych,
- opuścić jedno ze słów kluczowych lub dodać podwójnie.

Przeanalizuj błędy powstające w przypadku poszczególnych sytuacji. W ten sposób przekonasz się, z jakimi typami błędów możesz się zetknąć w przyszłości, i odkryjesz, jak je rozwiązywać. Zobaczysz też, w jaki sposób kompilator będzie o nich informował.

3 Napisz program (pamiętając o nazwach zmiennych, wcięciach i komentarzach), który zadeklaruje trzy zmienne rzeczywiste float. Wczytaj je z klawiatury za pomocą funkcji scanf. Wyświetl zawartości tych trzech zmiennych za pomocą jednej instrukcji printf. Każdą z nich wyświetl z różną liczbą cyfr po przecinku.

4 Identyfikatory w C/C++

Mamy już pierwsze linie kodu C++. Z nauką języka programowania jest jak z nauką języka obcego – aby go dobrze poznać i nabrać biegłości w posługiwaniu się nim, trzeba wiele ćwiczyć, a następnie odpowiednio często używać, żeby go nie zapomnieć. Zanim zajmiesz się pisaniem kolejnych programów, zapoznaj się z pojęciem nazwy (inaczej identyfikatora).

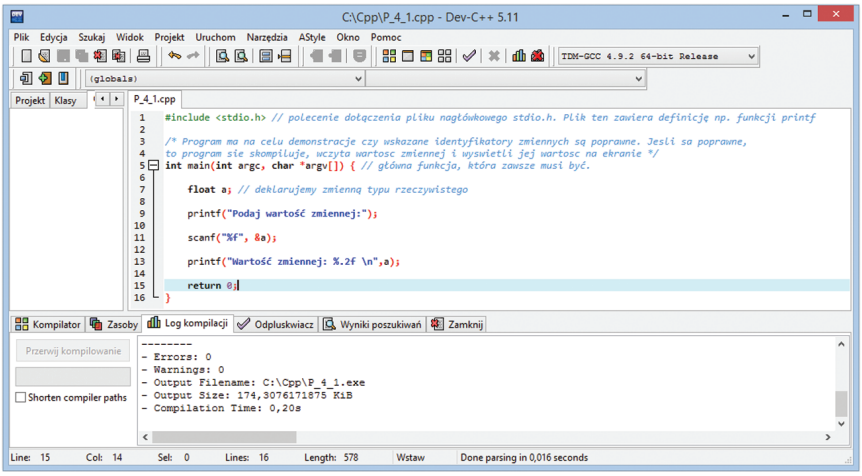
Identyfikatory, czyli nazwy

W swoich programach na każdym kroku będziesz tworzyć nazwy: programów, zmiennych, procedur, funkcji, a nawet nazwy swoich własnych typów. Przy pisaniu pierwszych programów stosowane już były nazwy zmiennych (`liczba1` oraz `liczba2`). Nazwy w C/C++ tworzy się według odpowiednich zasad.

Identyfikatorem nazywamy ciąg liter (bez polskich znaków) i cyfr zaczynający się od litery. Do liter zaliczamy również znak podkreślenia (_). Identyfikatorem nie może być wyraz będący słowem kluczowym. Dodatkowo rozróżniane są małe i duże litery. Zatem `liczba1` i `Liczba1` to nazwy dwóch zmiennych. Liczba znaków w nazwie może być dowolna.

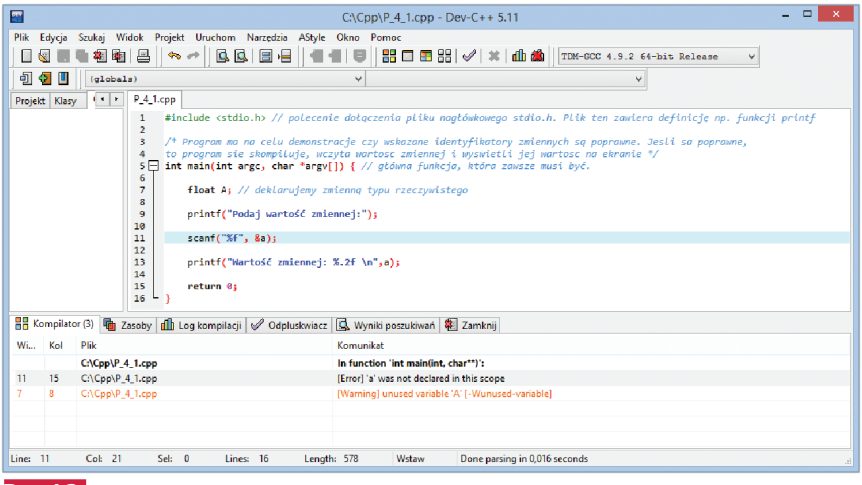
Przeanalizuj przykład, w którym została zadeklarowana zmienna o identyfikatorze `a`, typu `float`.

```
#include <stdio.h>

int main(int argc, char *argv[]) {

     float a;

     printf("Podaj wartosc zmiennej\n");

     scanf("%f", &a);

     printf("Wartosc: %f \n",a);

     return 0;
}
```



Rys. 4.1. Przykład użycia poprawnego identyfikatora

Program skompilował się (rys. 4.1.) i uruchomił. Jeśli zmienisz nazwę deklarowanej zmiennej i zamiast `a` użyjesz `A`, to program już się nie skompiluje (rys. 4.2.). Dla niego `a` i `A` to inne nazwy zmiennych. Zadeklarowana została zmienna `A`, natomiast w funkcjach `printf` i `scanf` jest użyta zmienna `a`. Ze względu na to, że kompilator rozróżnia małe i wielkie litery, bez problemu możesz w programie zadeklarować dwie zmienne o identyfikatorach `a` i `A`.



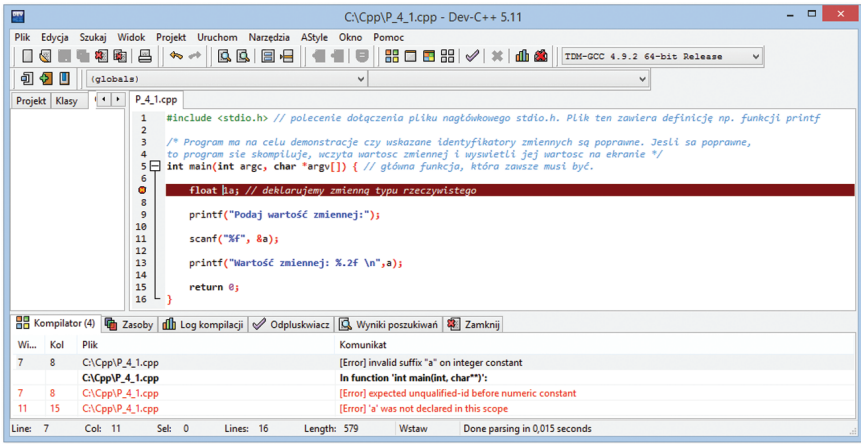
Rys. 4.2. Rozróżnianie małych i wielkich liter przez kompilator

Poprawnymi identyfikatorami będą na przykład `zmienna` lub `Zmienna` - są to po prostu ciągi liter zaczynające się od litery. Sprawdź to.

Przykładami poprawnych identyfikatorów będą także:

- `iloczyn_macierzy` czy `ILOCZYN_MACIERZY` - ciągi liter zaczynające się od litery z umieszczonym w środku znakiem podkreślenia,
- `x15` - ciąg liter i cyfr zaczynającym się od litery,
- `_2K` - ciąg liter i cyfr zaczynający się od znaku podkreślenia.

Natomiast `1x` jest przykładem błędnego identyfikatora, gdyż jest to ciąg rozpoczynający się cyfrą. Zwróć uwagę na komunikat kompilatora w opisie błędu - zakładka **Kompilator** (rys. 4.3.).



Rys. 4.3. Błędny identyfikator - komunikat o błędzie

Błędnymi identyfikatorami są także:
`aA*B` - w ciągu występuje znak specjalny (gwiazdka),
`a_b_c[d]` - w ciągu występują znaki specjalne (nawiasy kwadratowe).
Sprawdź, jaki komunikat wyświetli kompilator po znalezieniu takich nazw.
Tworząc nazwy zmiennych, konstruuj je w taki sposób, aby łatwo kojarzyły się z tym, do czego będą służyć. Dzięki temu zwiększa się czytelność programu. Przyjęło się, że:
- nazwy zmiennych zaczynają się z małej litery,
- nazwy stałych pisane są dużymi literami,
- nazwy zmiennych całkowitych zwykle zaczynają się od liter `i` do `n`.

Zapamiętaj

- Identyfikator może składać się tylko z liter (bez polskich znaków), cyfr i znaku podkreślenia. Musi zaczynać się od litery lub znaku podkreślenia.
- Identyfikator nie może być słowem kluczowym.
- Wielkość liter ma znaczenie.

Zadania

1. Czy można zadeklarować zmienne: main, Main, Printf, scanf? Uzasadnij swoją odpowiedź.

2. Podaj po pięć przykładów poprawnych i błędnych identyfikatorów.