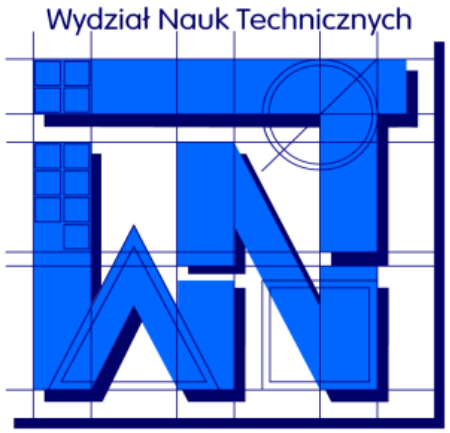


UNIVERSITY OF WARMIA AND MAZURY IN OLSZTYN
The Faculty of Technical Sciences
POLAND, 10-957 Olsztyn, M. Oczapowskiego 11
tel.: (48)(89) 5-23-32-40, fax: (48)(89) 5-23-32-55
URL: http://www.uwm.edu.pl/edu/sobieski/ (in Polish)

Języki Programowania

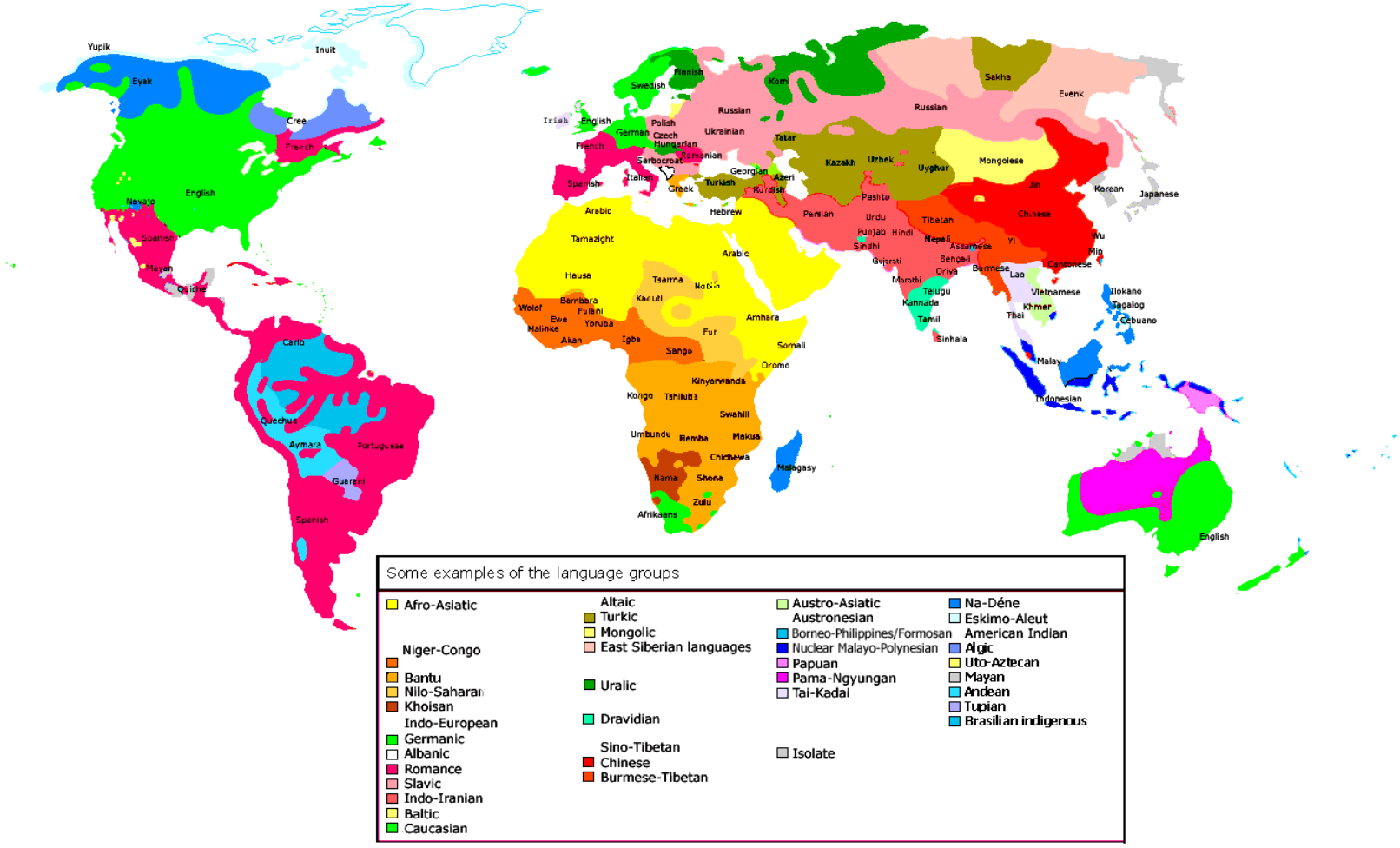
Wprowadzenie do programowania

Wojciech Sobieski

Olsztyn, 2001-2021

Język

Język – jest to ogólna nazwa zdefiniowanego zbioru znaków i symboli oraz reguł określających sposoby i kolejność ich użycia.



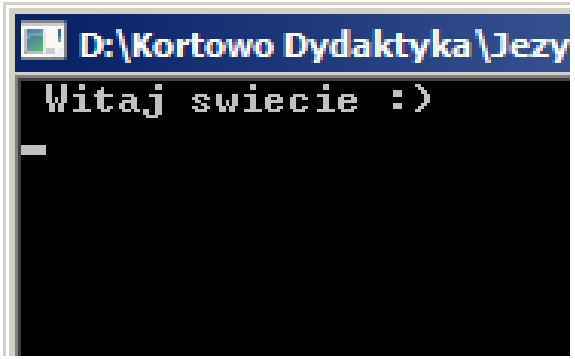
Język programowania

Język programowania – zbiór zasad (znaków i symboli oraz reguł ich stosowania), dzięki którym powstaje kod źródłowy programu komputerowego. Kod źródłowy musi zostać przetworzony na kod maszynowy, wykonywany przez konkretny procesor. Etap ten nosi nazwę **translacji**.

```
program hello
print *, 'Witaj swiecie :)'
read *
end
```

kod źródłowy

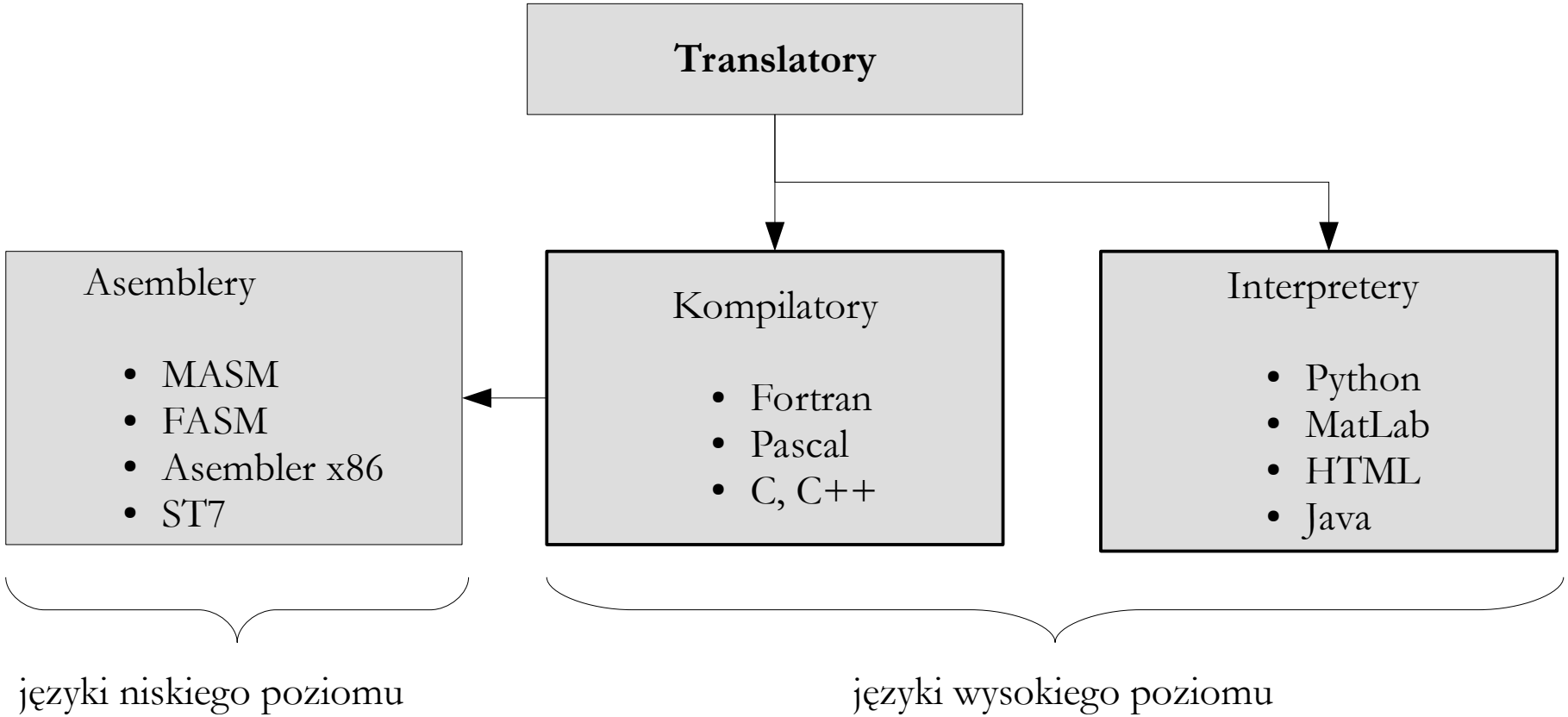
translacja



wykonany
kod maszynowy

Translator

Translator – program służący do automatycznego tłumaczenia kodu źródłowego na kod maszynowy. Istnieją dwa rodzaje translatorów: **kompilatory** (w tym **asemblery**) oraz **interpretery**.

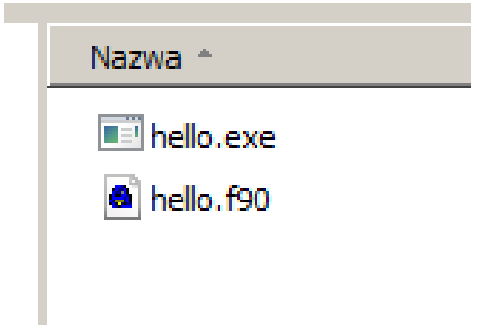


Kompilator

Kompilator – program jednorazowo tłumaczący cały kod źródłowy (napisany w języku wysokiego poziomu) na kod maszynowy i zapisujący go do pliku wykonywalnego. Kolejne uruchomienia programu nie wymagają powtórzenia etapu kompilacji (kompilator nie jest potrzebny do uruchamiania programu).

```
program hello
print *, 'Witaj swiecie :)'
read *
end
```

kod źródłowy (Fortran)



plik źródłowy i plik wynikowy (Windows)

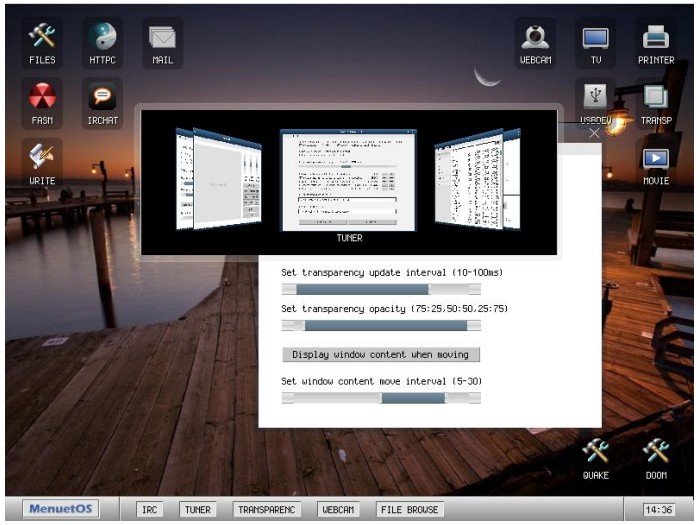
Asembler

Asembler – program jednorazowo tłumaczący cały kod źródłowy (napisany w języku niskiego poziomu) na kod maszynowy i zapisujący go do pliku wykonywalnego. Kolejne uruchomienia programu nie wymagają powtórzenia etapu kompilacji (kompilator nie jest potrzebny do uruchamiania programu).

```
mov ax, 0D625h
mov es, ax
mov al, 24
mov ah, 0
int 21h
```

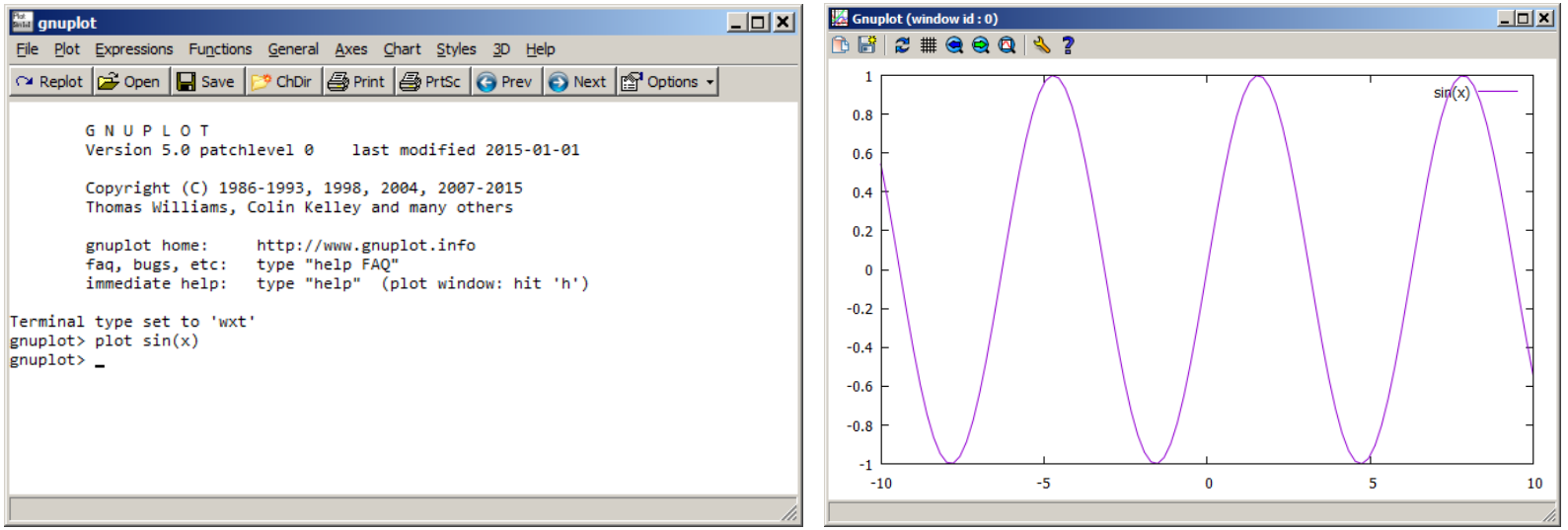
kod źródłowy (Asembler x86)

MenuetOS – 64-bitowy system operacyjny mieszczący się na dyskietce 1.44 MB!!



Interpreter

Interpreter – program tłumaczący i wykonujący kod źródłowy linia po linii. W tym przypadku nie ma pliku wykonywalnego, a każde uruchomienie wymaga ponownego przeprowadzenia etapu interpretacji (interpreter jest niezbędny do uruchomienia i wykonania programu).



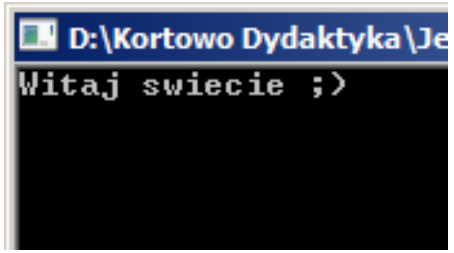
Przykład interpretacji instrukcji w środowisku Gnuplot

Kod źródłowy

Kod źródłowy – zapis programu komputerowego przy pomocy określonego języka programowania, opisujący operacje jakie powinien wykonać komputer na zgromadzonych lub otrzymanych danych. Kod źródłowy zapisywany jest w plikach źródłowych.

```
program hello;
begin
WriteLn('Witaj swiecie ;)');
end.
```

kod źródłowy (Pascal)

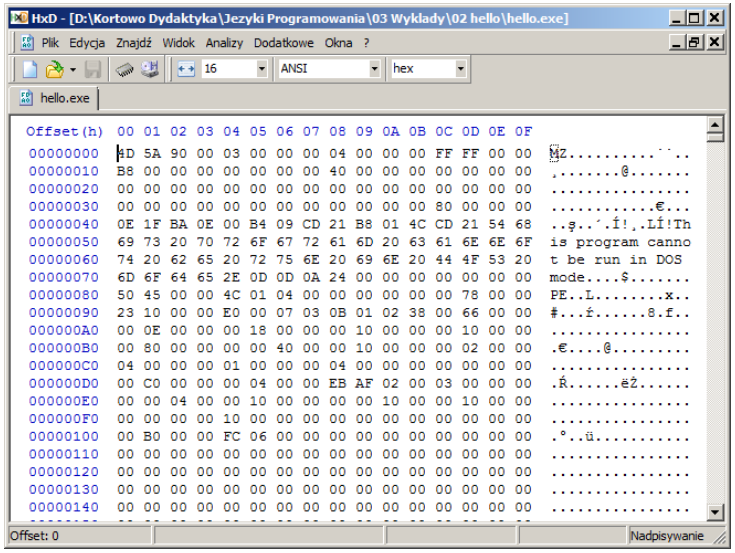


efekt uruchomienia pliku wykonywalnego (Windows)

Kod maszynowy

Kod maszynowy – kod generowany w procesie kompilacji (lub asemblacji). W trakcie procesu generowania kodu maszynowego często tworzony jest przenośny kod pośredni zapisywany w pliku obiektowym. Następnie kod ten pobrany z pliku obiektowego poddawany jest konsolidacji (linkowaniu) z kodem w innych plikach, w celu utworzenia ostatecznej postaci kodu maszynowego, który będzie zapisany w pliku wykonywalnym (wynikowym). Język maszynowy jest nieprzenośny, ponieważ każda architektura procesora ma swój własny zestaw rozkazów maszynowych.

przykładowy kod maszynowy
programu z poprzedniego slajdu



Plik dołączany (nagłówkowy)

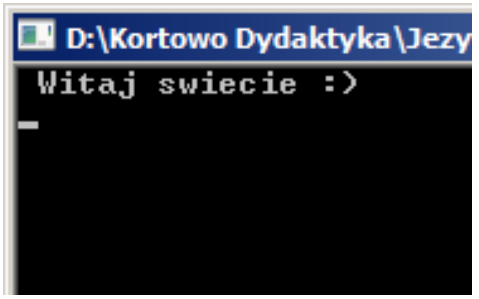
Plik dołączany (nagłówkowy) – jest to zewnętrzny plik tekstowy zawierający fragmenty kodu źródłowego (np. deklaracje typów zmiennych, wartości stałych, funkcji, procedur itp.). Zawartość plików dołączanych dodawana jest do innych plików źródłowych odpowiednim poleceniem, przy czym odbywa się to automatycznie na etapie tworzenia kodu maszynowego. Pliki nagłówkowe mają zazwyczaj rozszerzenie *.inc lub *.h.

```
program hello
include 'hello.inc'
read *
end
```

kod źródłowy (hello.f90)

```
print *, 'Witaj swiecie :)'
```

kod źródłowy w pliku dołączanym (hello.inc)

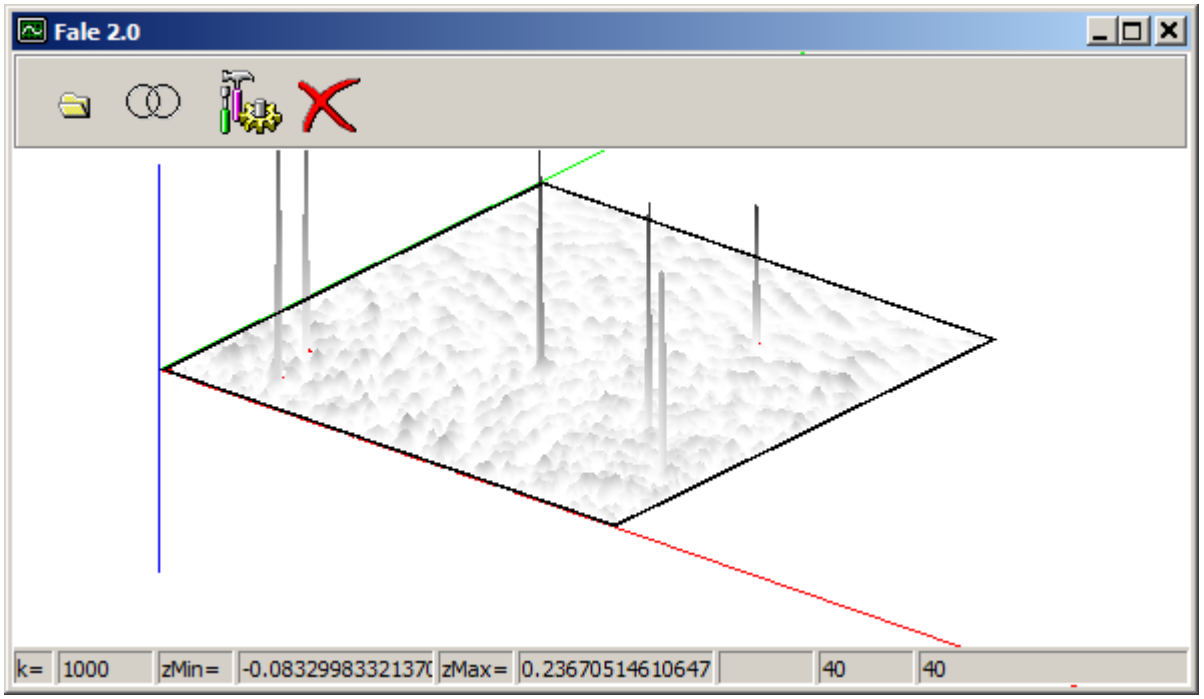


efekt uruchomienia pliku wykonywalnego (Windows)

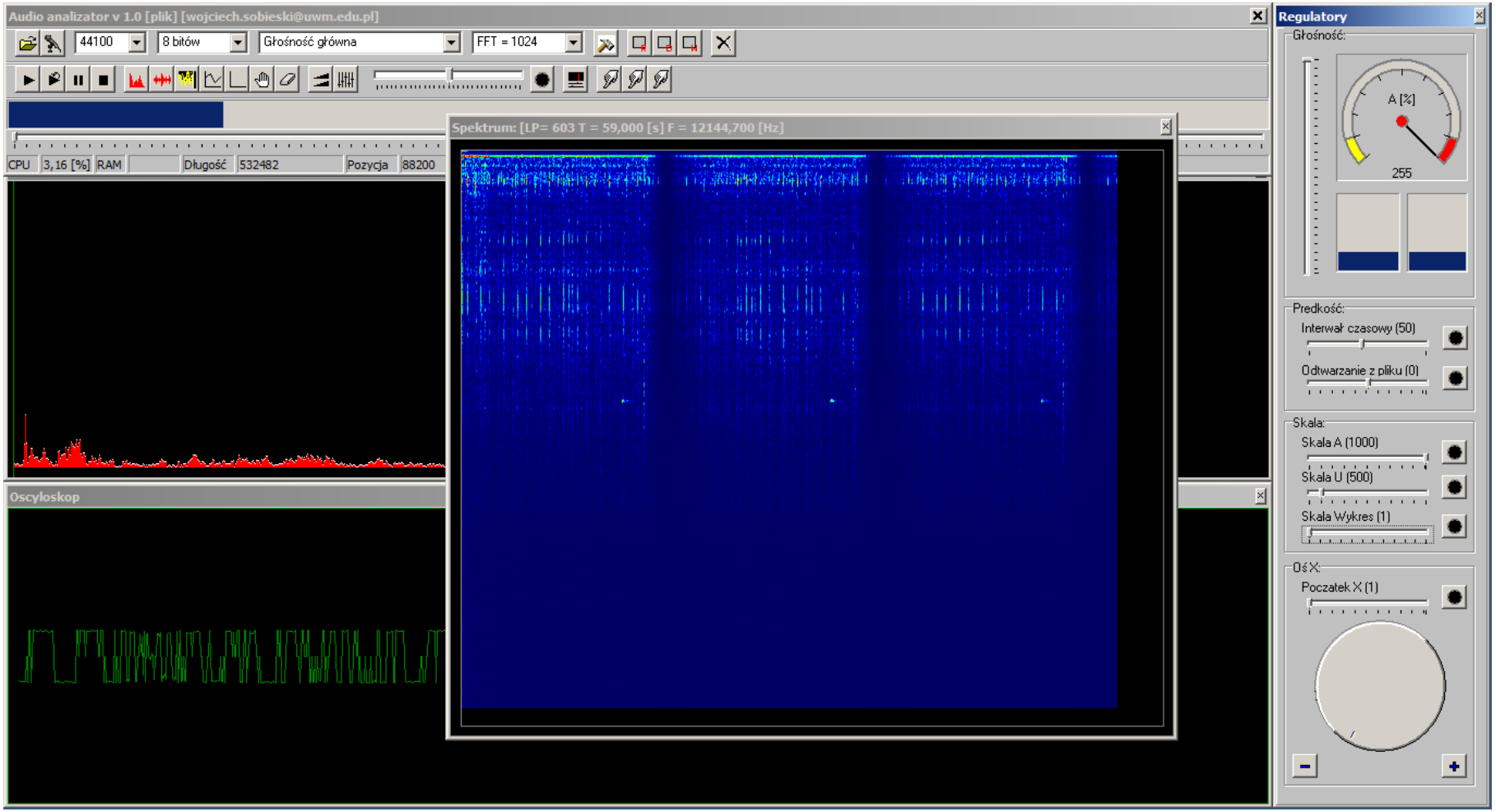
Biblioteka

Biblioteka – plik obiektowy zawierający fragmenty kodu źródłowego (najczęściej procedury lub funkcje) realizujące określone zadania: matematyczne (np. biblioteka LAPACK, SLATEC), graficzne (np. DISLIN, OpenGL) i inne.

przykład programu rozwiązującego tzw. równanie falowe napisanego w środowisku Borland Delphi z użyciem biblioteki dynamicznej OpenGL

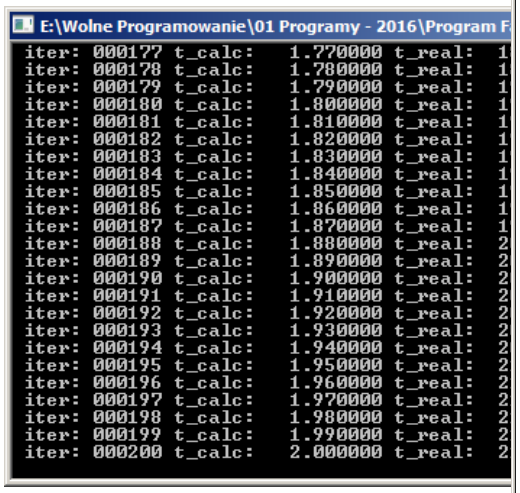


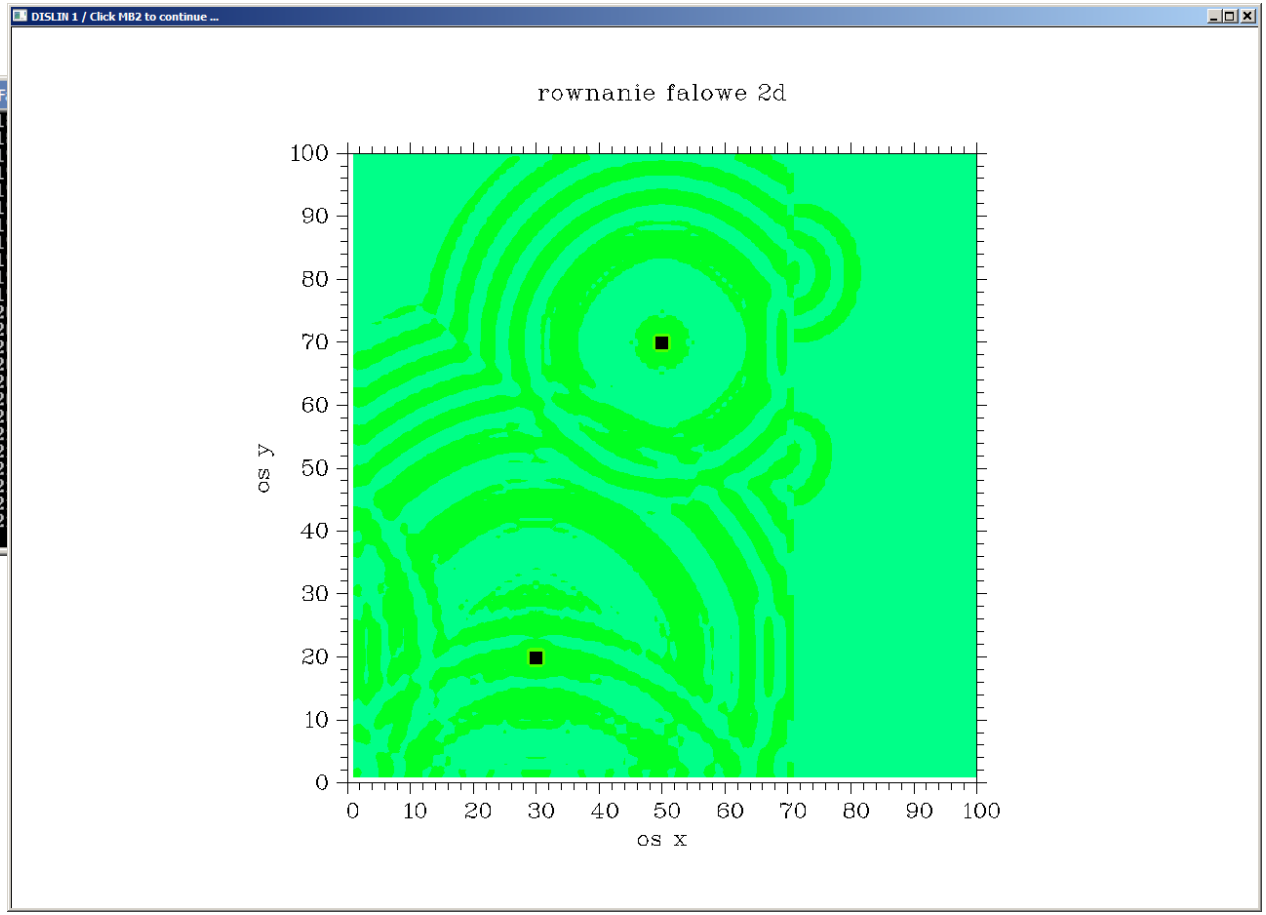
Biblioteka



przykład programu służącego do analizy sygnału akustycznego (FFT)
napisanego w środowisku Borland Delphi z użyciem biblioteki dynamicznej BASS

Biblioteka

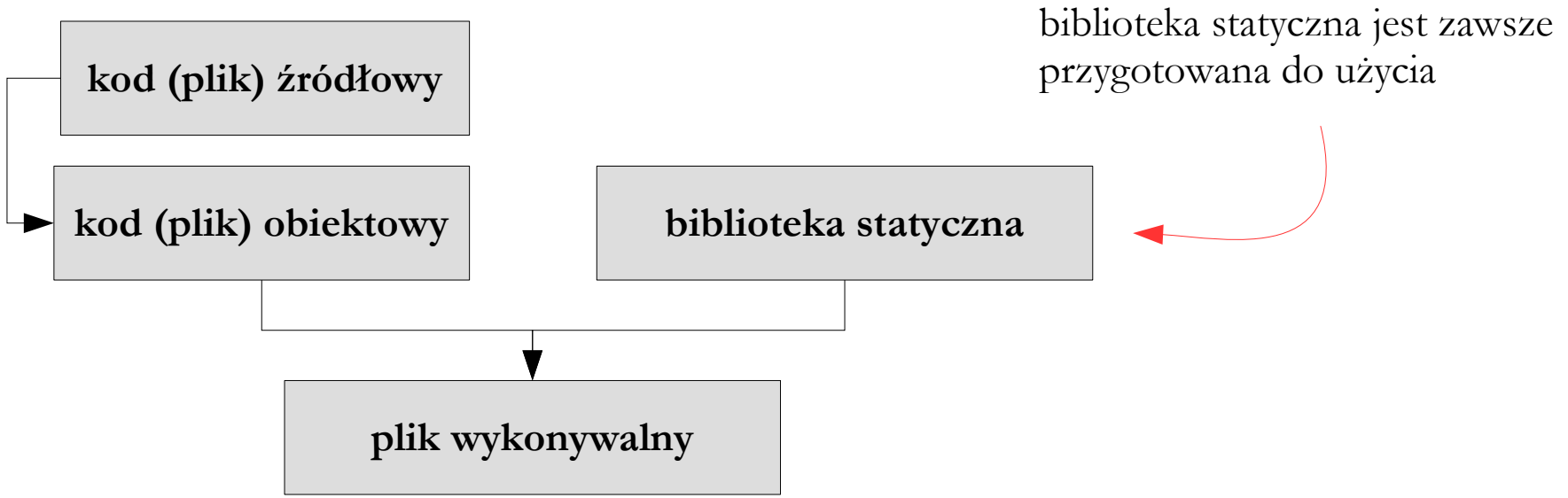




przykład programu rozwiązującego tzw. równanie falowe
napisanego w języku Fortran z użyciem biblioteki statycznej DISLIN

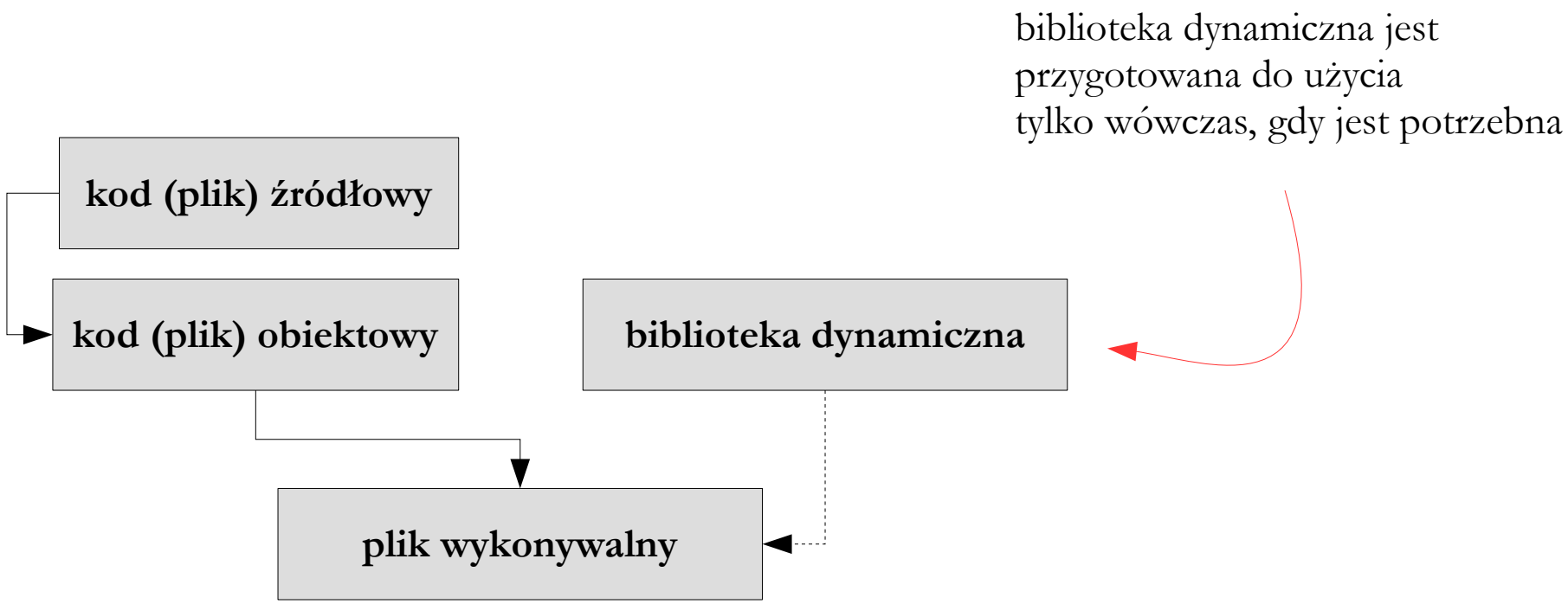
Biblioteka statyczna

Biblioteka statyczna – jest to rodzaj biblioteki, która łączona jest z programem w etapie konsolidacji (linkowania). W systemach z rodziny Windows zwykło nadawać się im rozszerzenia *.lib lub *.o, natomiast w systemach z rodziny Unix/Linux są to zwykle *.a lub *.o. Biblioteki statyczne, w przeciwieństwie do bibliotek dynamicznych, nie wymagają pomocy systemu operacyjnego komputera – po połączeniu z danym programem są od razu gotowe do użycia.

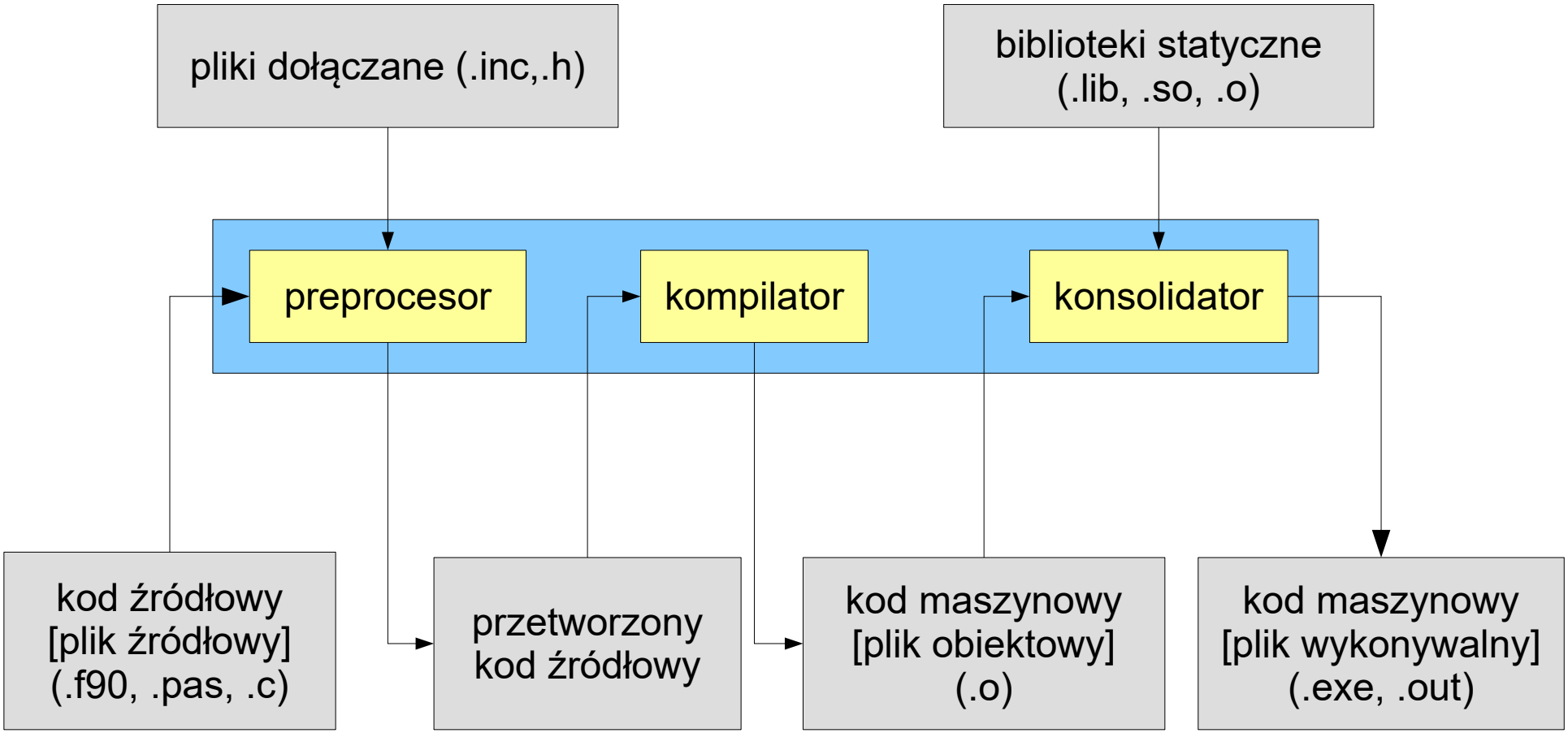


Biblioteka dynamiczna

Biblioteka dynamiczna – jest to rodzaj biblioteki, która łączona jest z programem dopiero w momencie jego wykonania. Aby tego dokonać system operacyjny komputera musi posiadać pewne funkcje, które umożliwiają łączenie dynamiczne. W systemach Windows biblioteki dynamiczne mają zazwyczaj rozszerzenie *.dll.



Etapy kompilacji



schemat procesu translacji (tu: kompilacji)

Preprocesing

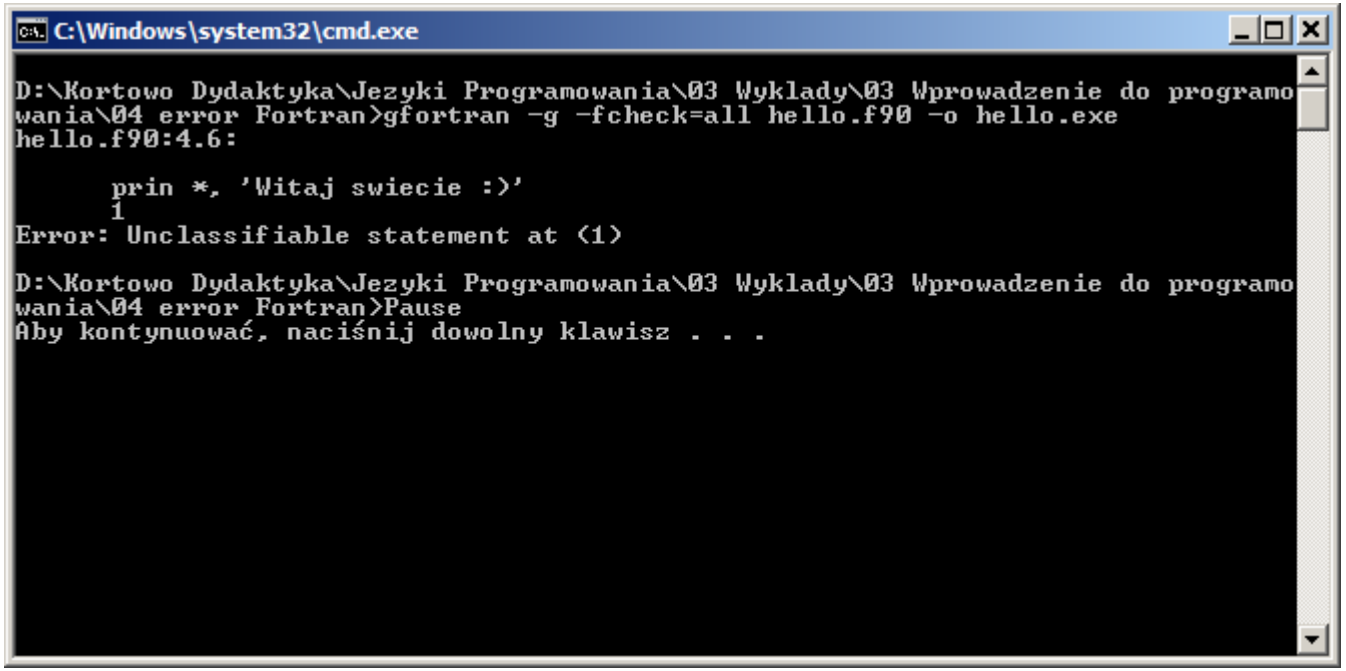
Etap preprocesingu – obejmuje odczyt i analizę kodu źródłowego na podstawie reguł danego języka. W tym etapie dokonywana jest:

- analiza leksykalna – rozdzielenie kodu źródłowego na elementarne jednostki języka programowania zwane tokenami;
- analiza składniowa – kontrola, czy ułożenie tokenów nie łamie reguł danego języka programowania (czy kod źródłowy był poprawny składniowo);
- analiza semantyczna – określenie znaczenia poszczególnych tokenów. Przykładem może być sprawdzanie, czy nie występuje niezgodność typów.

Na etapie preprocesingu do kodu źródłowego wstawiana jest również zawartość plików dołączanych.

Preprocesing

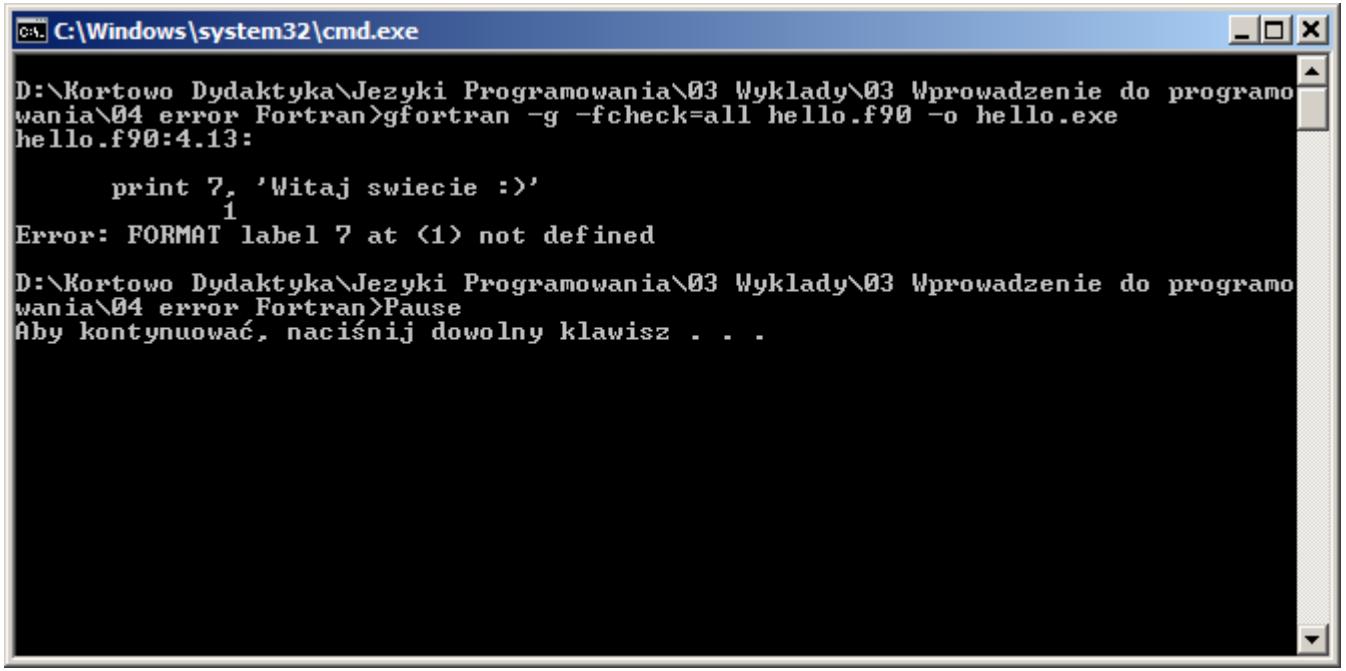
```
program hello
prin *, 'Witaj swiecie :)'
read *
end
```



Przykład błędu wykrytego przez preprocesor (język Fortran) – użycie nieznanej instrukcji (**prin**)

Preprocesing

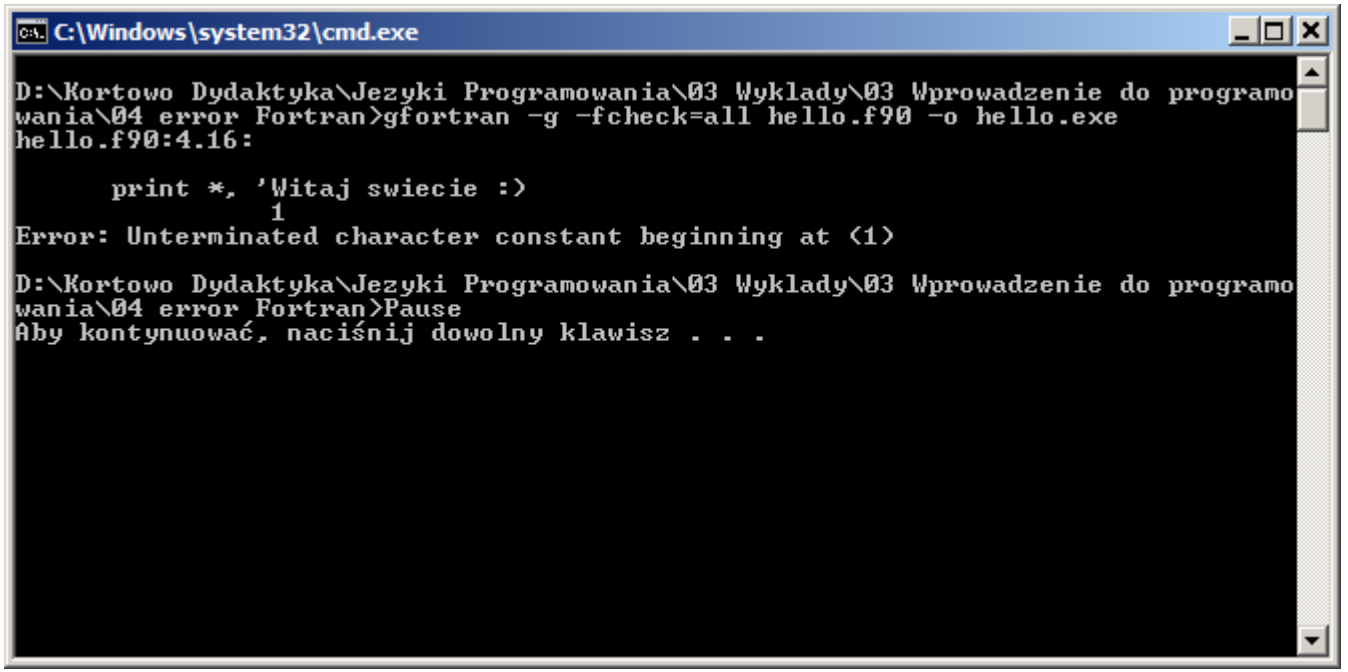
```
program hello
print 7, 'Witaj swiecie :)'
read *
end
```



Przykład błędu wykrytego przez preprocesor (język Fortran) – brak definicji formatu I/O (**7**)

Preprocesing

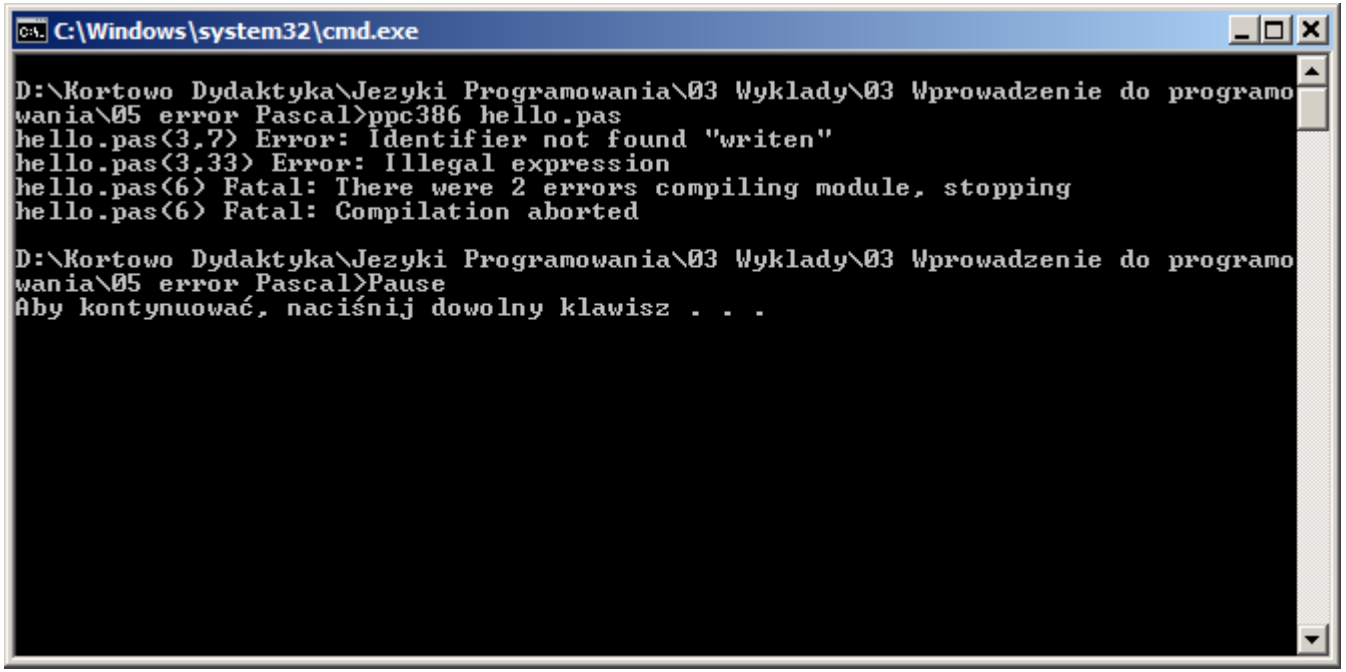
```
program hello
print *, 'Witaj swiecie :)
read *
end
```



Przykład błędu wykrytego przez preprocesor (język Fortran) – niezamknięty łańcuch znaków ('...')

Preprocesing

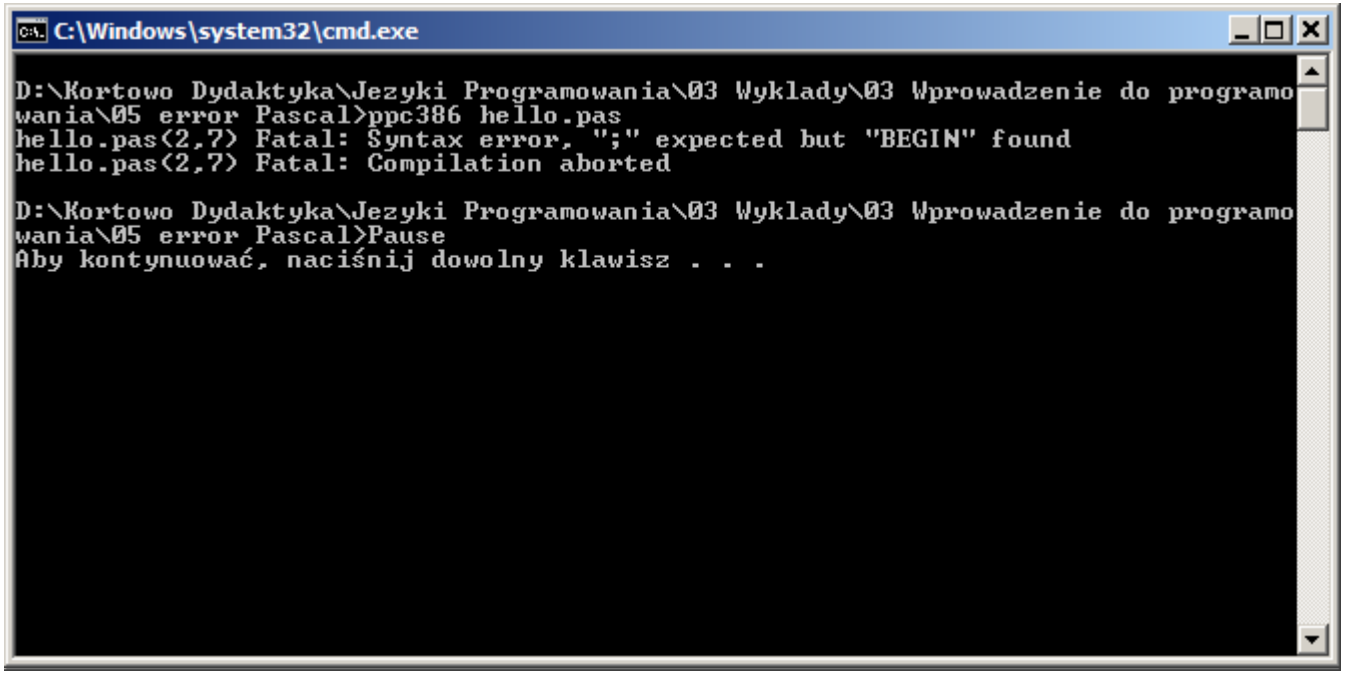
```
program hello;
begin
Writen('Witaj swiecie ;)');
end.
```



Przykład błędu wykrytego przez preprocesor (język Pascal) – użycie nieznanej instrukcji (**Writen**)

Preprocesing

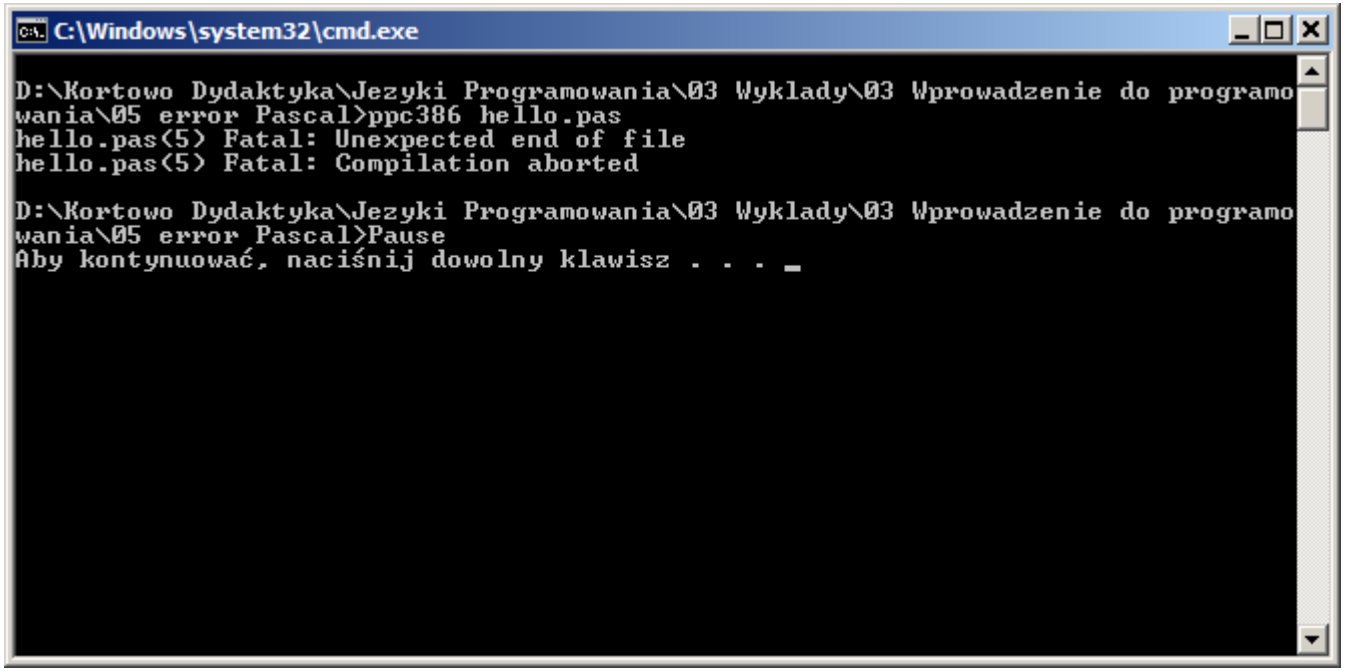
```
program hello
begin
WriteLn('Witaj swiecie ;)');
end.
```



Przykład błędu wykrytego przez preprocesor (język Pascal) – brak średnika na końcu 1-go wiersza

Preprocesing

```
program hello;
begin
WriteLn('Witaj swiecie ;)');
```



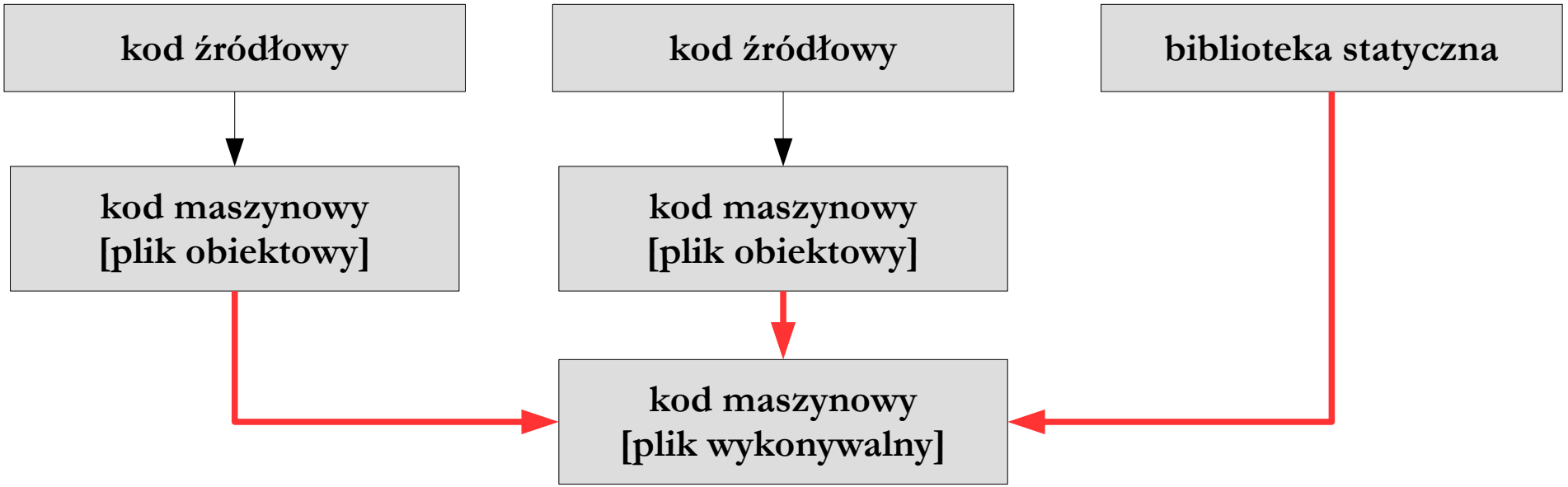
Przykład błędu wykrytego przez preprocesor (język Pascal) – nieoczekiwany koniec pliku (brak **end.**)

Kompilacja

Etap kompilacji – służy do przetłumaczenia kodu źródłowego na plik obiektowy. W przypadku jednoczesnej kompilacji wielu plików źródłowych, każdy z nich przetworzony jest na oddzielny plik obiektowy. Na tym etapie nie są analizowane zależności pomiędzy poszczególnymi segmentami programu.

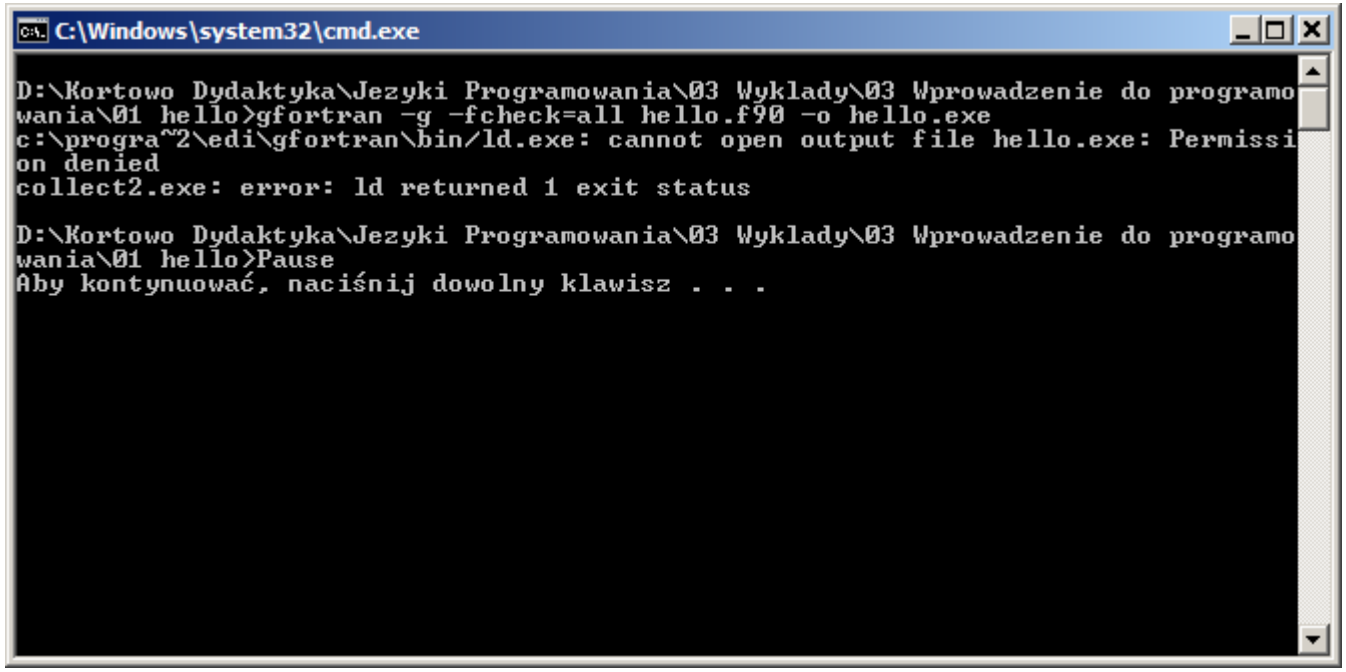
Konsolidacja

Etap konsolidacji (linkowania) – służy do połączenia wszystkich plików obiektowych oraz dodatkowych bibliotek statycznych w jeden kod wynikowy (wykonywalny). Na tym etapie sprawdzane są zależności pomiędzy poszczególnymi segmentami programu. Dodanie biblioteki statycznej wymaga podania odpowiednich opcji kompilacji. Po konsolidacji program jest gotowy do uruchomienia.



Konsolidacja

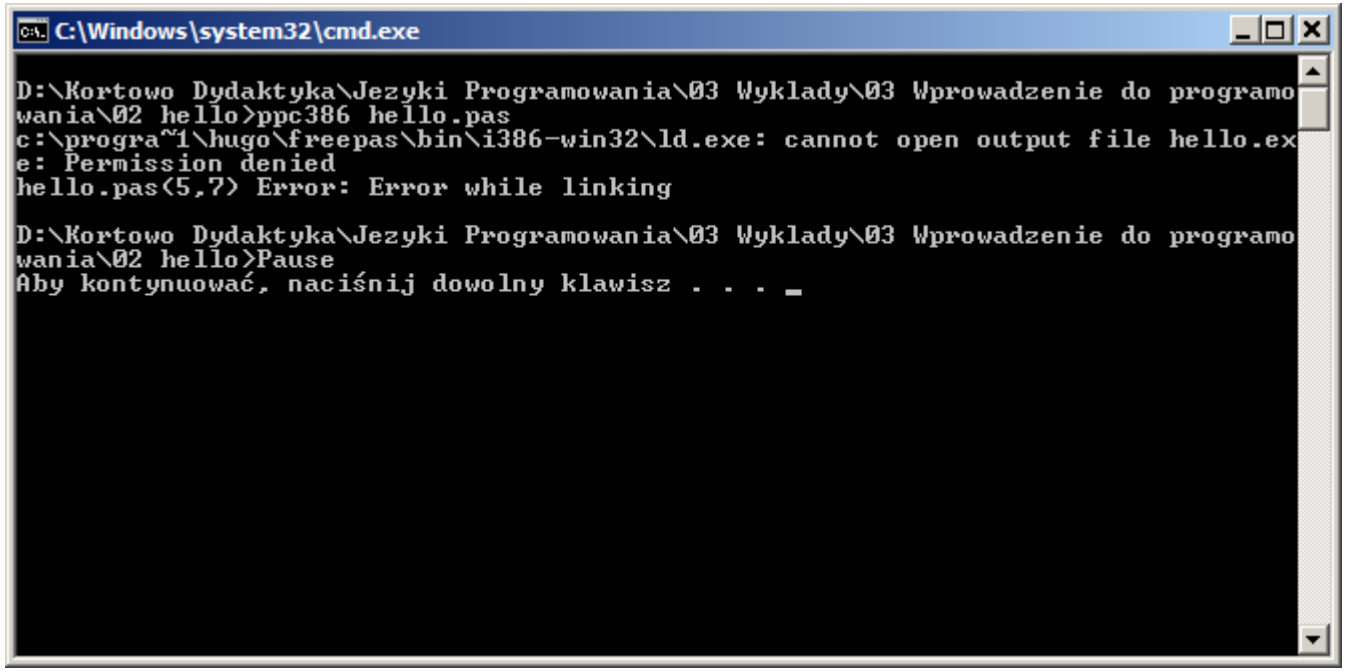
```
program hello
print *, 'Witaj swiecie :)'
read *
end
```



Przykład błędu na etapie konsolidacji pojedynczego kodu źródłowego (język Fortran) – plik wykonywalny jest właśnie uruchomiony i nie ma do niego dostępu

Konsolidacja

```
program hello;
begin
Writen('Witaj swiecie ;)');
end.
```

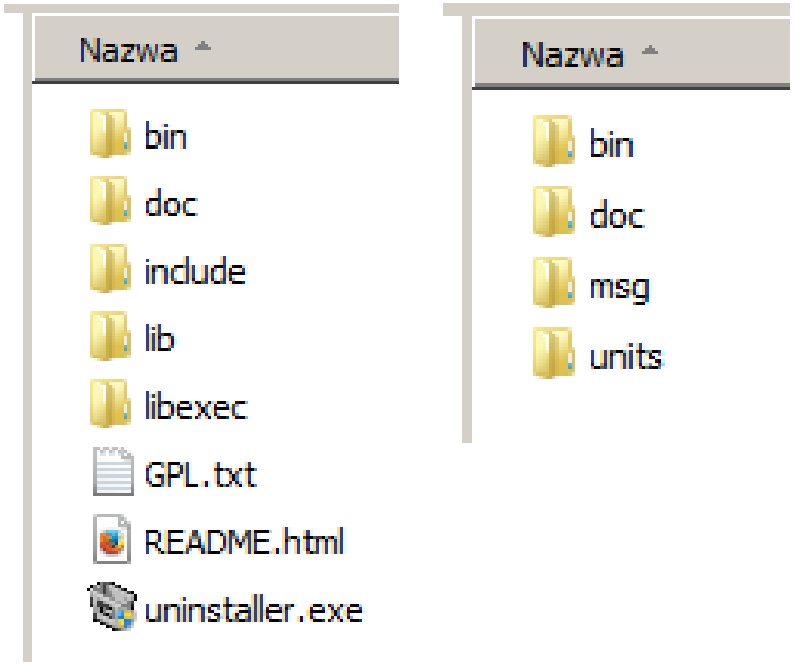


Przykład błędu na etapie konsolidacji pojedynczego kodu źródłowego (język Pascal) – plik wykonywalny jest właśnie uruchomiony i nie ma do niego dostępu

Środowisko programistyczne

1. Translator – program służący do wykonania kompilacji lub interpretacji kodu źródłowego. Standardowym katalogiem, w których umieszcza się pliki translatora jest katalog BIN w głównym katalogu aplikacji. Aby proces translacji był możliwy, system operacyjny musi znać dokładne położenie translatora i jego bibliotek.

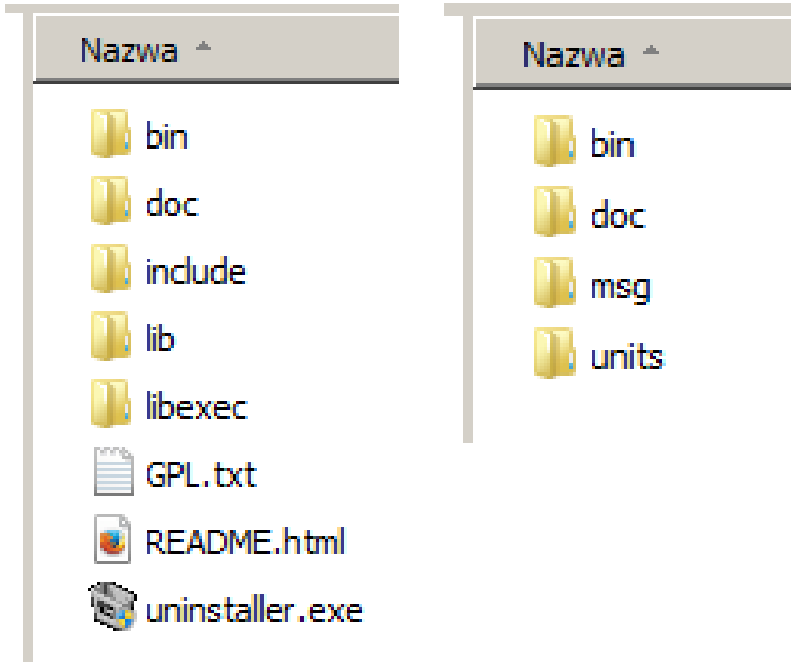
przykładowa
zawartość katalogu
kompilatora:
gfortran (z lewej) oraz
free pascal (z prawej)



Środowisko programistyczne

2. Biblioteki i dodatkowe pliki nagłówkowe – służą do rozszerzania możliwości języka, szczególnie w zakresie zastosowań specjalistycznych (np. obliczeń matematycznych, numerycznych, obróbki grafiki). Standardowo instalowane są w katalogach LIBRARY i INCLUDE w głównym katalogu aplikacji. Biblioteki podstawowe dostarczane są wraz z środowiskiem programistycznym, biblioteki dodatkowe rozprowadzane są w postaci osobnych pakietów.

przykładowa zawartość katalogu kompilatora: gfortran (z lewej) oraz free pascal (z prawej)



Środowisko programistyczne

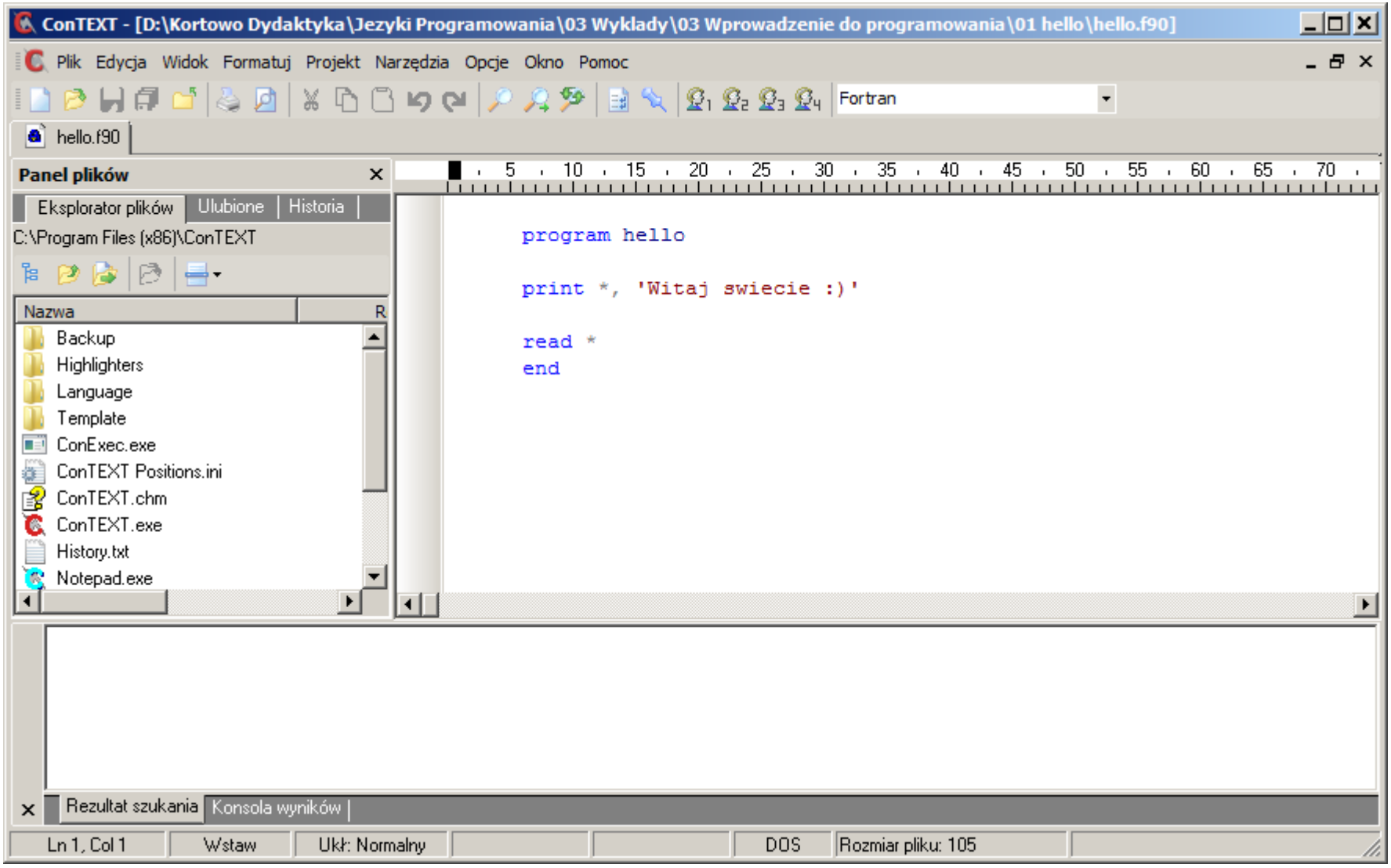
3. Edytor kodu – program służący do edycji kodu źródłowego.

Rodzaje edytorów kodu:

- uniwersalne (np. ConText, Amigo, Crimson, Codex, EditPlus, UltraEdit, Editeur, EmEditor i inne). Edytory uniwersalne pozwalają na podłączanie kompilatorów jednego lub wielu języków i są rozprowadzane jako oddzielne programy.

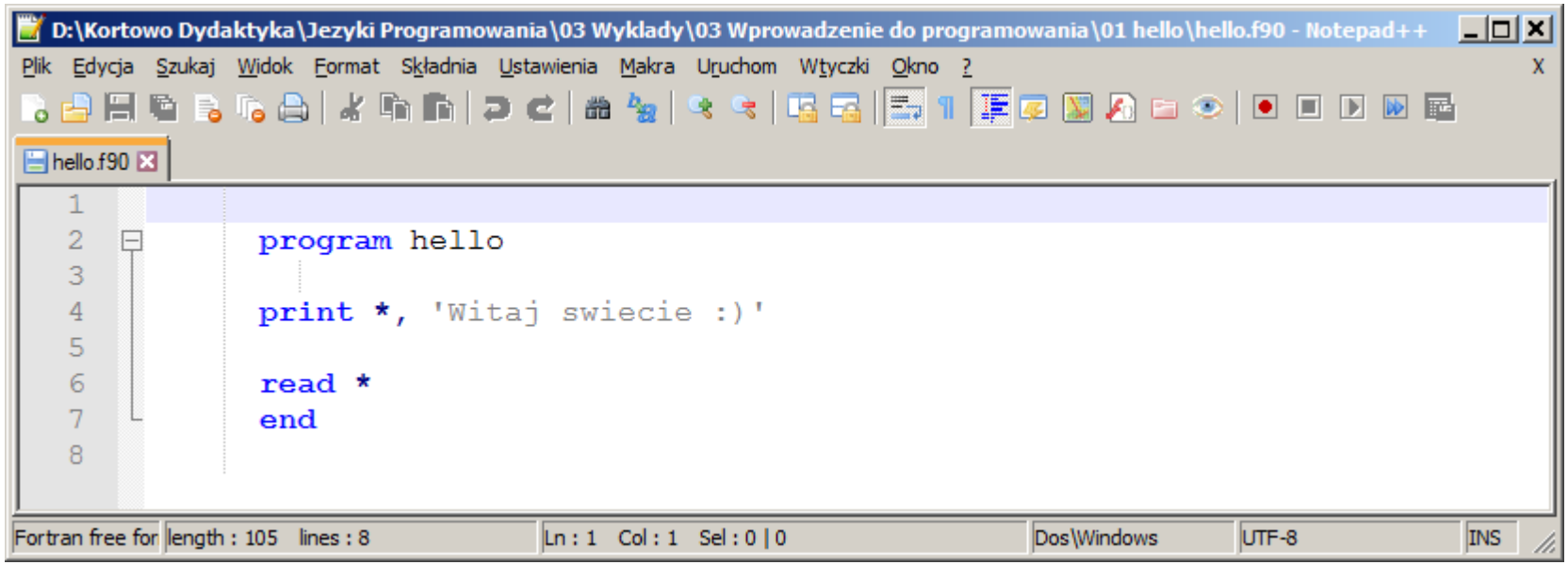
- zintegrowane z konkretną implementacją języka (np. Borland Delphi, Compaq Visual Fortran i inne).

Środowisko programistyczne



przykład uniwersalnego edytora kodu – ConTEXT

Środowisko programistyczne



przykład uniwersalnego edytora kodu – Notepad++

Środowisko programistyczne

4. Debugger (analizator kodu) – program służący do analizy poprawności tworzonego kodu źródłowego i stanowiący zazwyczaj integralną część translatora (ale może to być również oddzielny program).

Analizator kodu generuje:

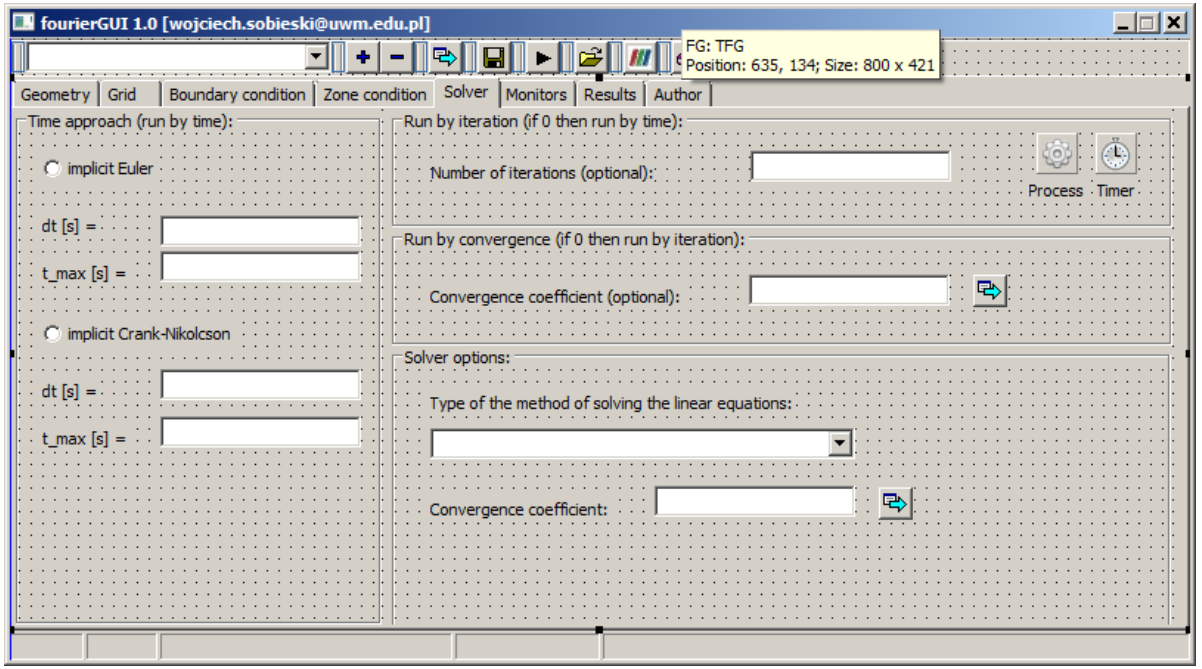
- **błędy** – komunikaty o złamaniu zasad pisaniu kodu źródłowego (program się nie skompiluje)

- **ostrzeżenia** – komunikaty o naruszeniu zasad pisaniu kodu źródłowego (program się skompiluje, ale może działać niepoprawnie lub niestabilnie)

Środowisko programistyczne

5. Edytor formularzy – program służący do budowy okien widzianych przez użytkownika po uruchomieniu tworzonej aplikacji (formularzy). Element ten występuje jedynie w językach wizualnych, takich jak Visual Basic, Borland Delphi, Compaq Visual Fortran i innych.

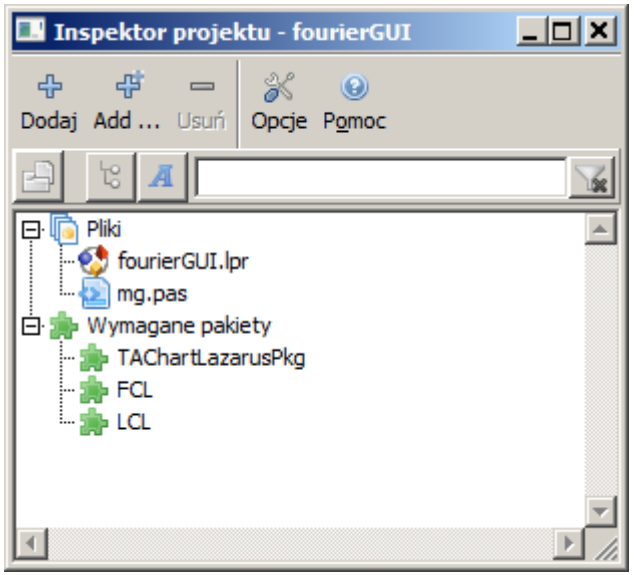
przykładowe okno programu zaprojektowane w edytorze formularzy środowiska Lazarus (program fourierGUI)



Środowisko programistyczne

6. Menadżer projektu – program służący do zarządzania modułami i plikami projektu (kodami źródłowymi, bibliotekami, komponentami oraz zasobami dodatkowymi).

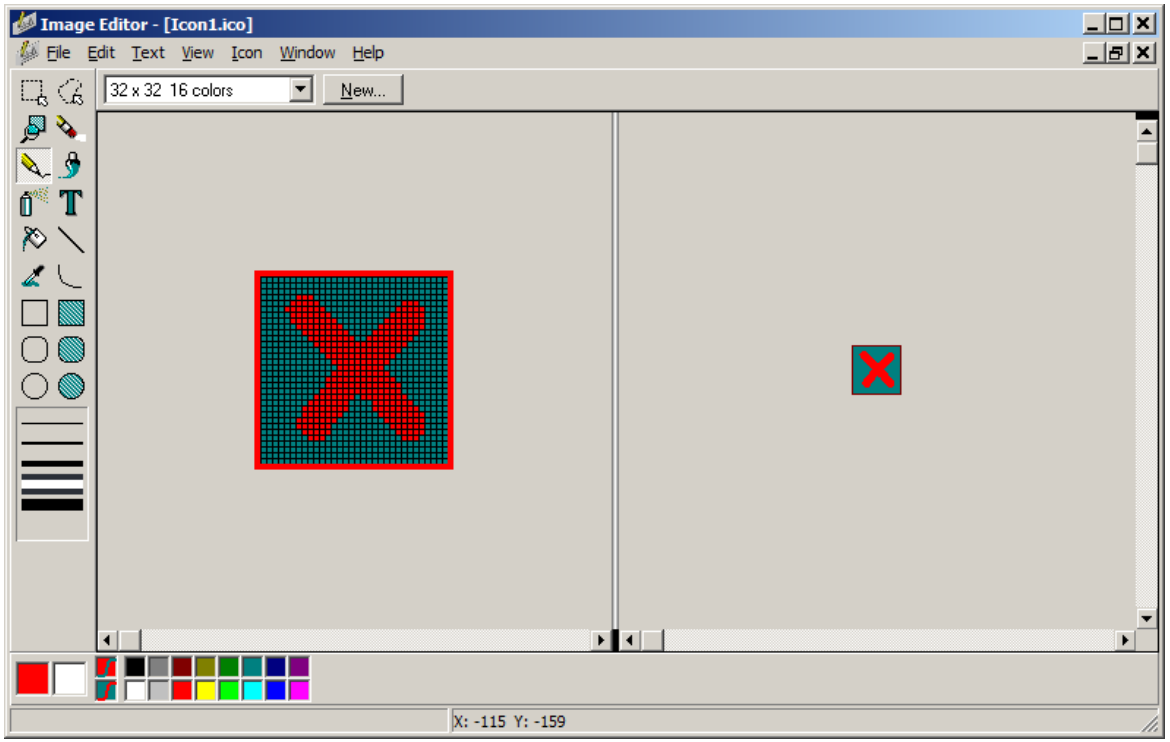
manager projektu
środowiska Lazarus



Środowisko programistyczne

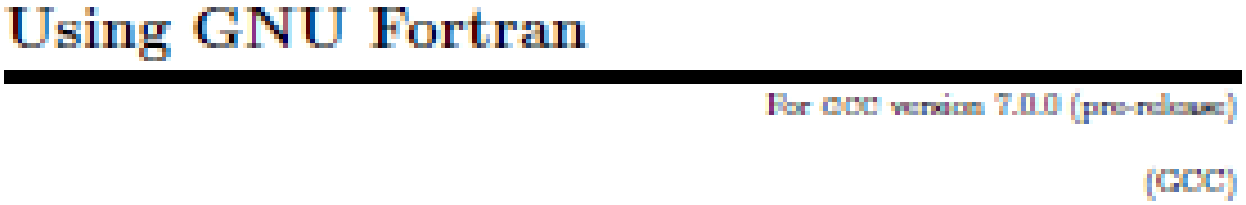
7. Narzędzia dodatkowe – służą do tworzenia systemu pomocy, ikon i kursorów, programów instalacyjnych. Ilość narzędzi i poziom ich zaawansowania zależy od implementacji języka.

ImageEditor
– narzędzie z pakietu Borland Delphi służące do tworzenia ikon i bitmap



Środowisko programistyczne

8. System pomocy – służy do uzyskiwania informacji o środowisku programistycznym, zasadach jego użytkowania, elementach języka (wraz z przykładami), rodzaju licencji, autorach i kontaktach. Zależnie od implementacji języka oraz jego rodzaju pomoc może być mniej lub bardziej rozwinięta. Dobrze zorganizowanie, obszerne systemy pomocy zawierają często kompendium wiedzy na temat danego języka programowania.


Using GNU Fortran

For GCC version 7.0.0 (pre-release)

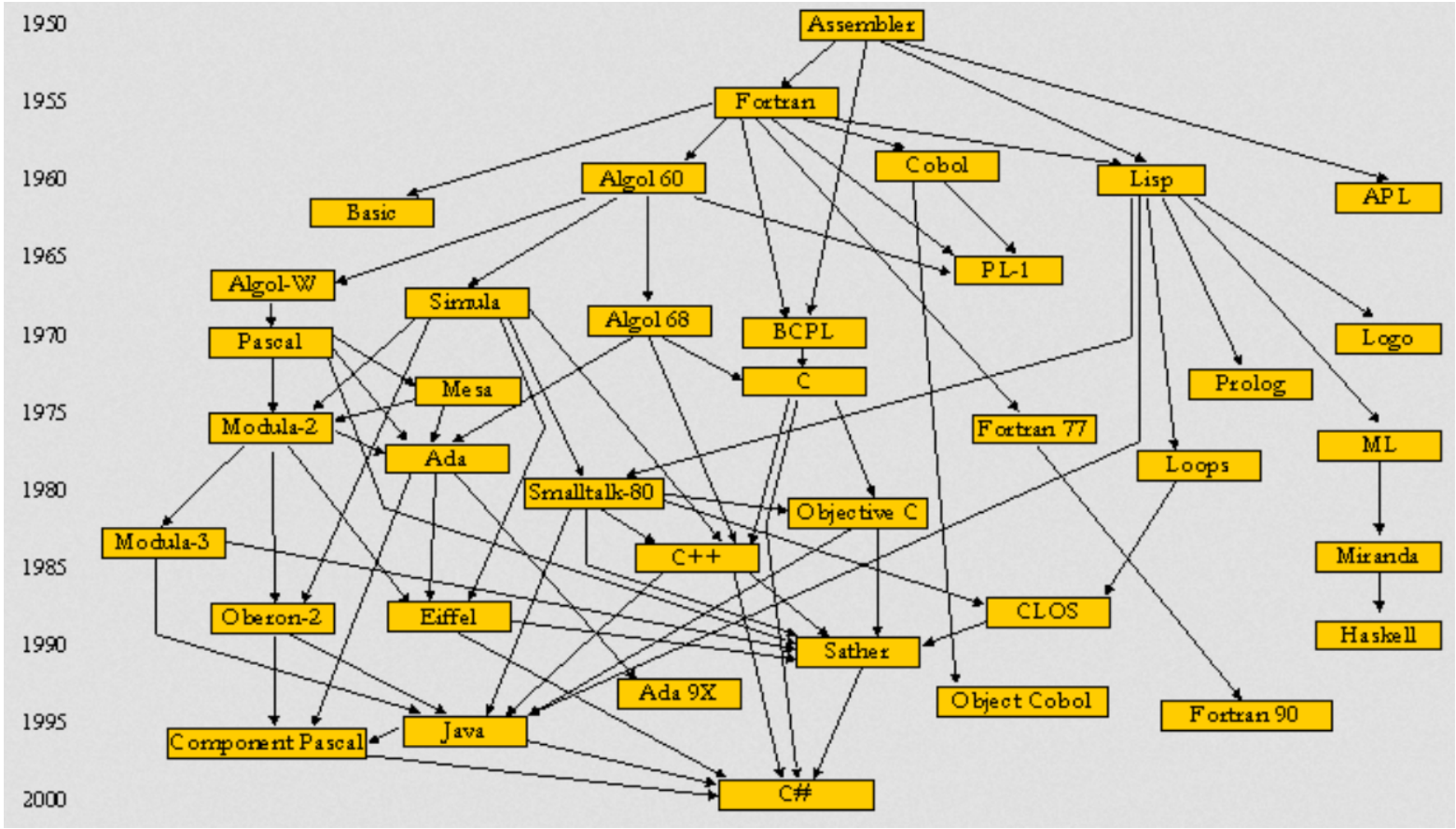
(GCC)

dobrze jest zaopatrzyć się w szczegółowy opis języka bezpośrednio od jego twórców, np.:
https://gcc.gnu.org/onlinedocs/gfortran.pdf

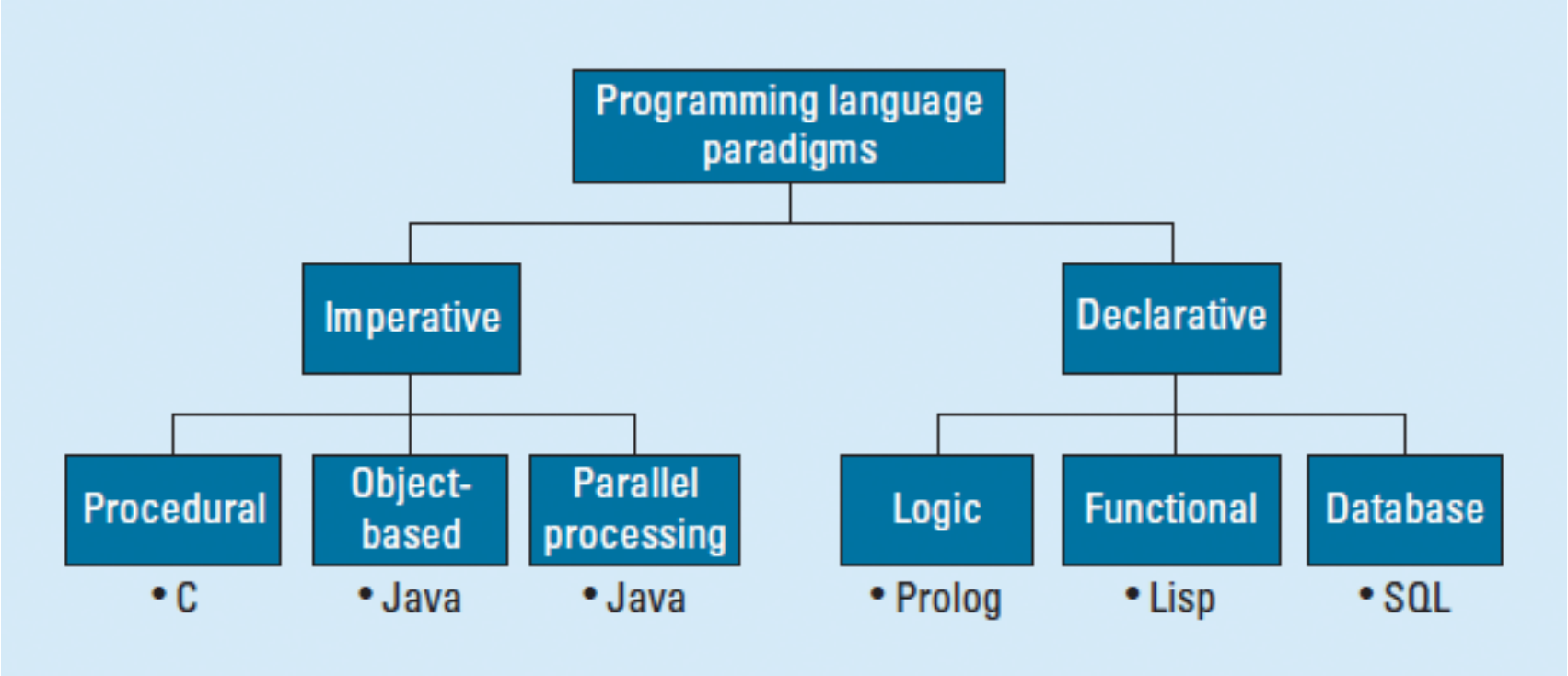
Klasyfikacja języków programowania

Problem:

ile jest języków programowania i wg jakich kategorii je klasyfikować?

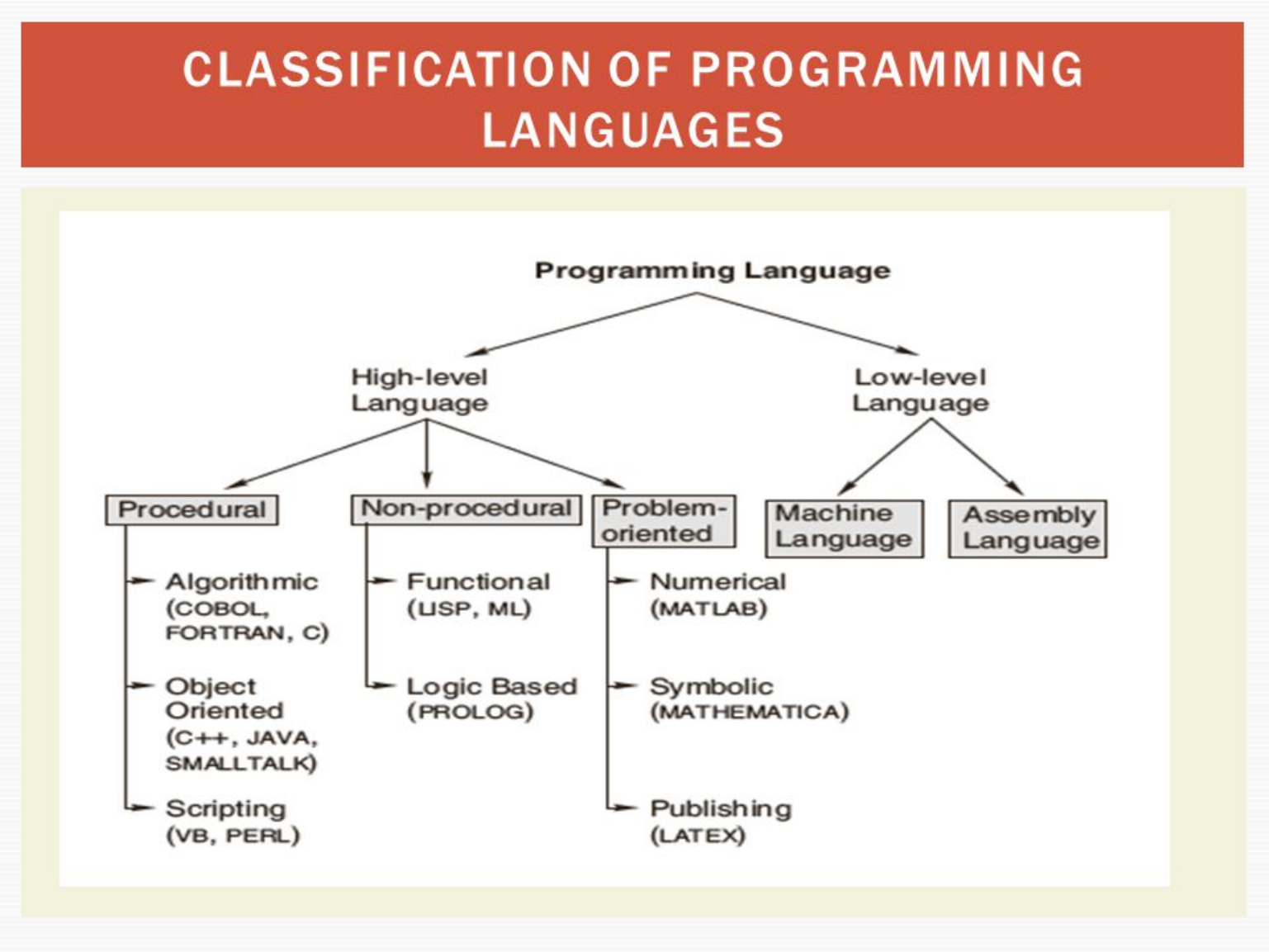


Klasyfikacja języków programowania



przykład drzewka obrazującego podział języków programowania

Klasyfikacja języków programowania



przykład drzewka obrazującego podział języków programowania

Przykłady języków programowania

FORTRAN (od FORmula TRANslator) – pierwszy w historii język wysokiego poziomu, stworzony przez zespół Johna Backusa na początku lat 50-tych XX wieku. Kompilator języka został starannie zoptymalizowany, ponieważ autorzy obawiali się, że nikt nie będzie go używał, jeśli szybkość programów nie będzie porównywalna z szybkością programów napisanych w asemblerze.

```
PROGRAM HELLOWORLD
10 FORMAT (1X,11HHELLO WORLD)
WRITE(6,10)
END
```

Przykłady języków programowania

LISP (od LISt Processing) – drugi z kolei pod względem wieku język programowania wysokiego poziomu. Lisp powstał jako wygodna matematyczna notacja dla programów komputerowych, oparta na rachunku lambda. Lisp szybko został najchętniej wybieranym językiem do badania i rozwoju sztucznej inteligencji. Podstawową strukturą danych w Lispie jest lista.

Lista – struktura danych służąca do reprezentacji zbiorów dynamicznych, w której elementy ułożone są w liniowym porządku.

```
(DEFUN HELLO-WORLD ()
(PRINT (LIST 'HELLO 'WORLD)))
```

Przykłady języków programowania

COBOL (od COmmon Business Oriented Language) – język programowania stworzony w roku 1959 z głównym przeznaczeniem do prac programistycznych w dziedzinach ekonomii i biznesu; powstał z inicjatywy amerykańskiego departamentu obrony; mocno krytykowany przez programistów i teoretyków programowania.

Osobliwością języka COBOL jest składnia, którą starano się uczynić jak najbardziej podobną do naturalnego języka angielskiego. Np. dodanie do siebie wartości zmiennych A i B z umieszczeniem wyniku w zmiennej C zapisuje się w COBOLu następująco: ADD A TO B GIVING C.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. Hello.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
    Display 'Hello, World'.
    STOP RUN.
```

Przykłady języków programowania

BASIC (Beginner's All-purpose Symbolic Instruction Code) – język programowania wysokiego poziomu, opracowany w 1964 przez Johna George'a Kemeny'ego i Thomasa E. Kurtza w Dartmouth College w oparciu o Fortran i Algol-60.

Założenia projektantów BASIC-a uwzględniały łatwość użytkowania, wszechstronność zastosowań, interaktywność i dobrą komunikację z użytkownikiem poprzez jasne komunikaty błędów.

BASIC wybił się na czoło języków do zastosowań amatorskich i półprofesjonalnych po wprowadzeniu na rynek mikrokomputera Altair 8800.

```
10 PRINT "HELLO WORLD"
```

Przykłady języków programowania

Pascal – język programowania wysokiego poziomu opracowany przez Niklausa Wirtha w 1970 roku.

Pierwotnie Pascal służył celom edukacyjnym do nauki programowania strukturalnego. Popularność Pascala w Polsce była większa niż w innych krajach ze względu na dostępność kompilatorów w pirackich wersjach (zanim pojawiło się prawo ochrony własności intelektualnej), prostotę języka oraz jego popularyzację przez wyższe uczelnie. Szczyt popularności tego języka przypadł na lata 80-te i początek lat 90-tych XX wieku.

```
program HelloWorld;

begin
  writeln('Hello World');
end.
```

Przykłady języków programowania

SQL – deklaratywne, strukturalny, opracowany w latach 70-tych w firmie IBM, język zapytań używany do tworzenia, modyfikowania baz danych oraz do umieszczania i pobierania danych z baz danych.

```
SELECT 'Hello World!';
```

Przykłady języków programowania

C – imperatywny, strukturalny język programowania wysokiego poziomu stworzony na początku lat siedemdziesiątych XX w. przez Dennisa Ritchiego do programowania systemów operacyjnych i innych zadań niskiego poziomu.

C stał się popularny poza Laboratoriami Bella (gdzie powstał) po 1980 roku i stał się dominującym językiem do programowania systemów operacyjnych i aplikacji. Na bazie języka C w latach osiemdziesiątych Bjarne Stroustrup stworzył język C++, który ułatwia znacząco programowanie obiektowe.

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
 {
  printf("Hello, world\n");
  return EXIT_SUCCESS;
 }
```

Przykłady języków programowania

Prolog (od francuskiego Programmation en Logique) – deklaratywny język programowania logicznego. Język powstał na początku lat 70-tych jako język programowania służący do automatycznej analizy języków naturalnych. Obecnie Prolog jest językiem ogólnego zastosowania, szczególnie dobrze sprawdzającym się w programach związanych ze sztuczną inteligencją.

Program w Prologu składa się z faktów oraz reguł wnioskowania. Aby go uruchomić, należy wprowadzić odpowiednie zapytanie.

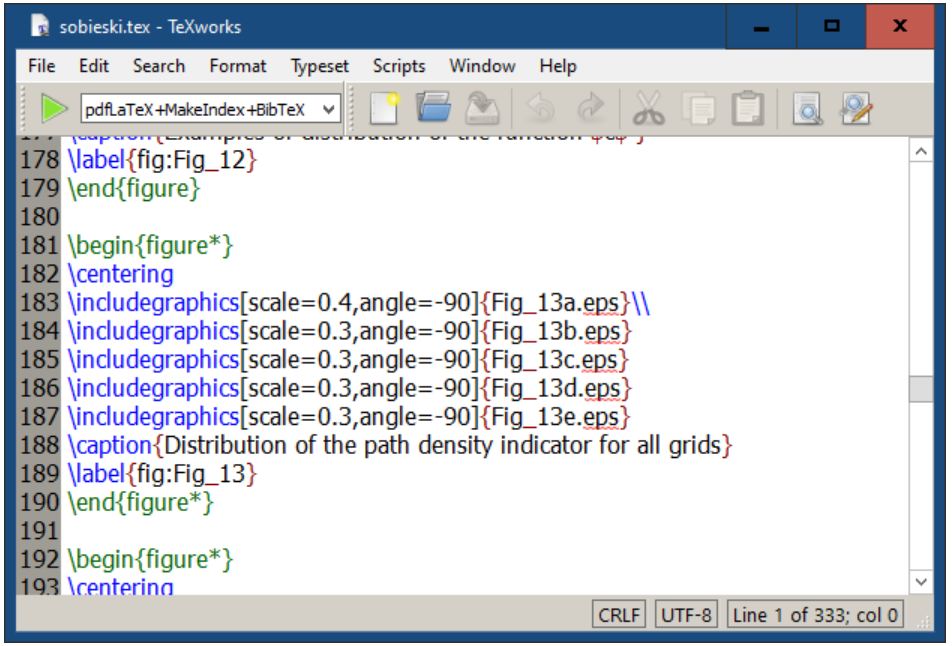
```
go :-
writeln('Hello World').
```

Przykłady języków programowania

TeX – komputerowy system profesjonalnego składu drukarskiego (przeznaczonego z założenia do składu publikacji naukowych), obejmujący zarówno specjalny język, jak i kompilator przygotowujący pliki w formatach wymaganych przez urządzenia graficzne (drukarki, naświetlarki).

Twórcą TeX-a jest Donald E. Knuth, amerykański matematyk i informatyk. Tworzenie programu zajęło mu 8 lat (1977-1985), choć pierwotnie szacował, że zadanie zrealizuje w pół roku.

```
\documentclass[12pt]{article}
\begin{document}
Hello world!
$Hello world!$ %math mode
\end{document}
```



Przykłady języków programowania

bash – jedna z najpopularniejszych powłok systemów uniksowych. Jest domyślną powłoką w większości dystrybucji systemu GNU/Linux oraz w systemie OS X od wersji 10.3, istnieją także wersje dla większości systemów uniksowych. Bash jest także domyślną powłoką w środowisku Cygwin i MinGW dla systemów Win32.

Nazwa jest akronimem od Bourne-Again Shell (angielska gra słów: fonetycznie brzmi tak samo, jak born again shell, czyli odrodzona powłoka). Wywodzi się od powłoki Bourne'a sh, która była jedną z pierwszych i najważniejszych powłok systemu UNIX oraz zawiera pomysły zawarte w powłokach Korna i csh. Bash był pisany głównie przez Briana Foksa i Cheta Rameya w 1987.

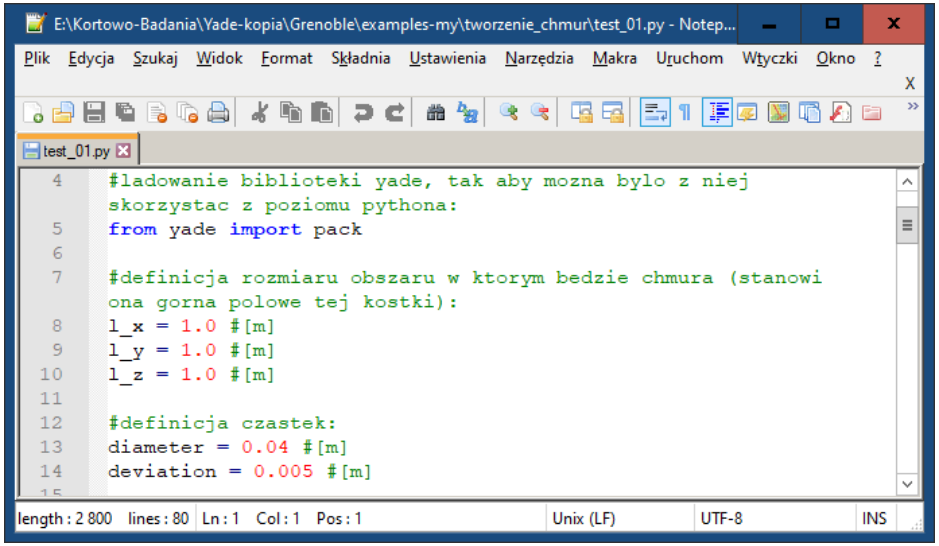
```
#!/bin/bash
echo "Hello world"
```

Przykłady języków programowania

Python – język programowania wysokiego poziomu ogólnego przeznaczenia, o rozbudowanym pakiecie bibliotek standardowych, którego ideą przewodnią jest czytelność i klarowność kodu źródłowego. Jego składnia cechuje się przejrzystością i zwięzłością. Pythona stworzył we wczesnych latach 90-tych Guido van Rossum.

Python rozwijany jest jako projekt Open Source zarządzany przez Python Software Foundation, która jest organizacją non-profit.

```
print "hello world"
```

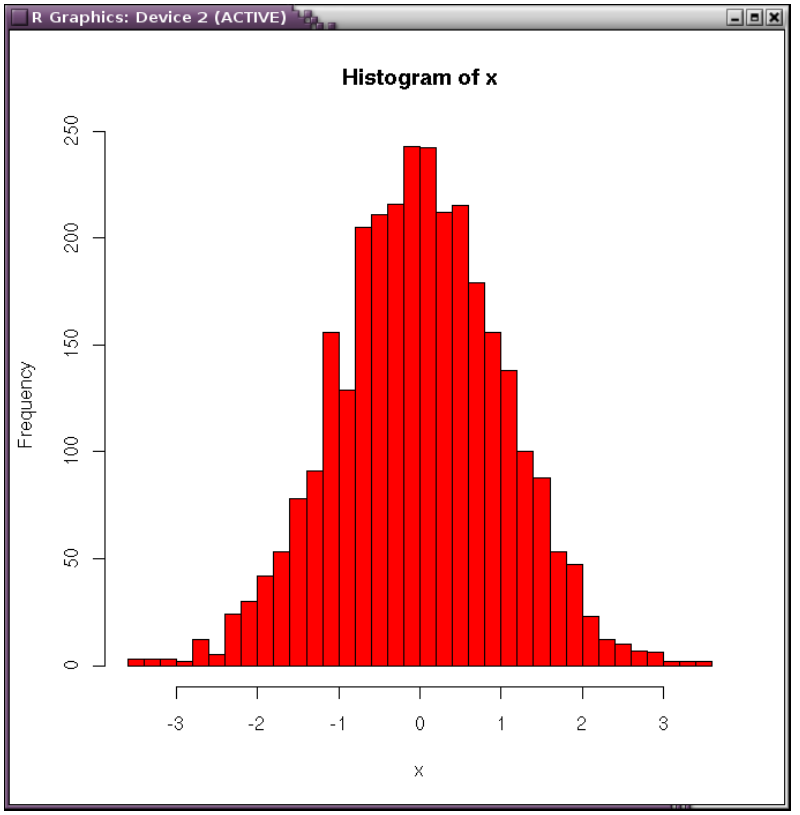


Przykłady języków programowania

R – interpretowany język programowania oraz środowisko do obliczeń statystycznych i wizualizacji wyników.

Kod źródłowy R opublikowany jest na zasadach licencji GNU GPL.

```
print("Hello World")
```



Przykłady języków programowania

PHP (od Personal Home Page) – interpretowany skryptowy język programowania zaprojektowany do generowania stron internetowych i budowania aplikacji internetowych działających w czasie rzeczywistym. PHP został stworzony przez Rasmusa Lerdorfa w roku 1994 jako zestaw skryptów Perla służący do monitorowania internautów odwiedzających jego witrynę

PHP jest najczęściej stosowany do tworzenia skryptów po stronie serwera WWW, ale może być on również używany do przetwarzania danych z poziomu wiersza poleceń, a nawet do pisania programów pracujących w trybie graficznym (np. za pomocą biblioteki GTK+, używając rozszerzenia PHP-GTK).

```
<?="Hello World"?>
```

Przykłady języków programowania

Java – powstały na początku XXI wieku obiektowy język programowania, w którym kod źródłowy kompiluje się najpierw do kodu pośredniego (niezależnego od systemu operacyjnego i procesora), a następnie wykonuje na tzw. wirtualnej maszynie Javy; maszyna ta tłumaczy kod uniwersalny na kod dostosowany do specyfiki konkretnego systemu operacyjnego i procesora.

Podstawowe koncepcje Javy zostały przejęte z języka Smalltalk (maszyna wirtualna, zarządzanie pamięcią) oraz z języka C++ (duża część składni i słów kluczowych).

```
public class Hello {
      public static void main(String []args) {
            System.out.println("Hello World");
      }
}
```

Przykłady języków programowania

Szacuje się, że do tej pory (rok 2021) opracowano na świecie około **9000** języków programowania, z czego szeroko używanych jest około 50.

Wydział Nauk Technicznych

UNIVERSITY OF WARMIA AND MAZURY IN OLSZTYN
The Faculty of Technical Sciences
POLAND, 10-957 Olsztyn, M. Oczapowskiego 11
tel.: (48)(89) 5-23-32-40, fax: (48)(89) 5-23-32-55
URL: http://www.uwm.edu.pl/edu/sobieski/ (in Polish)

Dziękuję

Wojciech Sobieski

Olsztyn, 2001-2021