

Dr inż. Andrzej Czerepicki

Języki i paradygmaty programowania

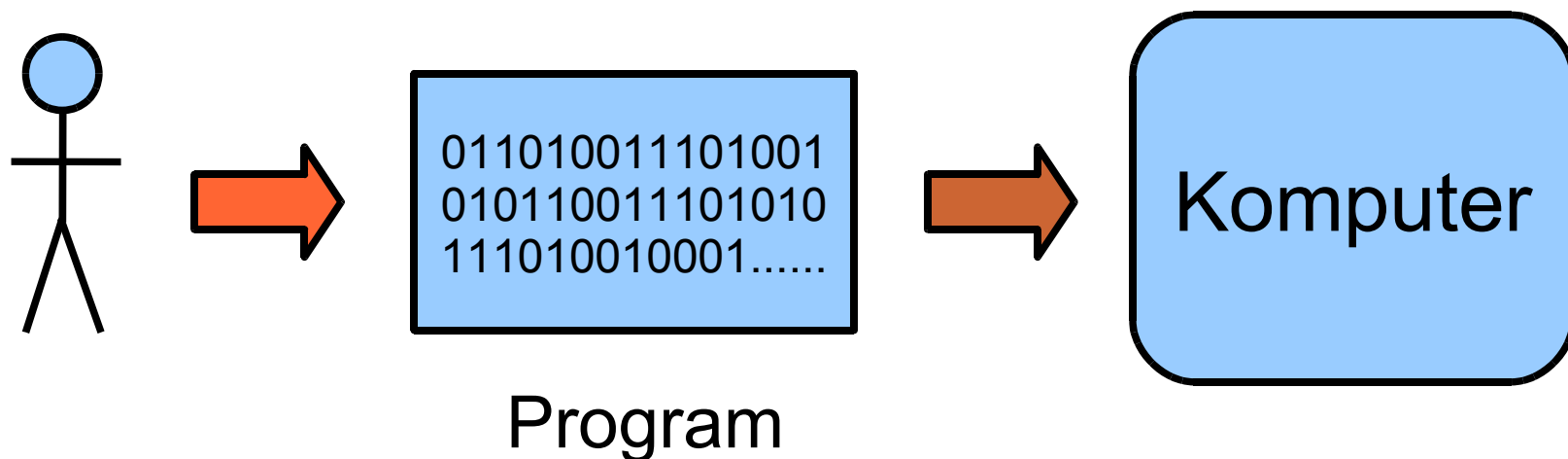
Studia podyplomowe

Wykład 3.

Języki programowania

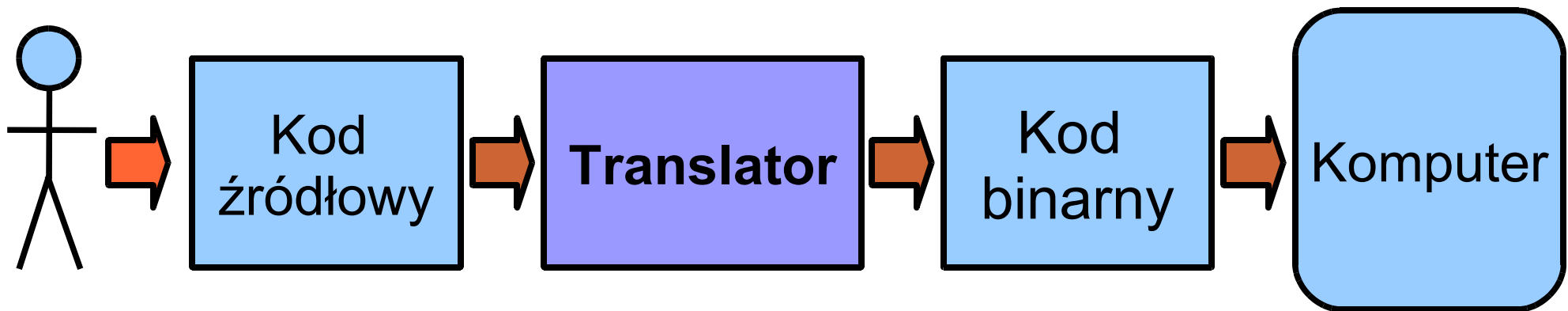
Wstęp

- Programowanie algorytmów w postaci kodów maszynowych (binarnych) okazało się zbyt trudnym i czasochłonnym zajęciem
- Zaczęto więc poszukiwania metod optymalizacji całego procesu



Wstęp (c.d.)

- Podjęto próbę stworzenia sposobu zapisu algorytmów, spełniającego jednocześnie kryteria:
 - Jednoznacznego tłumaczenia na kod maszynowy
 - Łatwiejszego opanowania przez człowieka



1. Język programowania

- Notację pozwalającą na zapisanie algorytmu w postaci innej niż kod maszynowy nazwano **językiem programowania**
- Język programowania jest językiem sztucznym, zorientowanym na komunikację pomiędzy człowiekiem a komputerem
- Języki programowania mają takie cechy języków naturalnych, jak:
 - Składnia
 - Semantyka

2. Składnia języka programowania

- Składnia języka programowania pokazuje jak należy tworzyć poprawne syntaktycznie konstrukcje języka, za pomocą:
 - **Alfabetu języka**
 - Zbiór symboli dopuszczalnych w konstrukcjach języka
 - **Gramatyki języka**
 - Zbiór reguł opisujących zasady tworzenia poprawnych konstrukcji języka

2.1. Alfabet języka programowania

W skład alfabetu większości języków programowania wchodzi następujące symbole:

- **cyfry** { 0..9 }
- **litera** { 'a..z', 'A..Z' }
- **znaki specjalne** { +/-%*^?!/ ... }

2.2. Gramatyka języka programowania

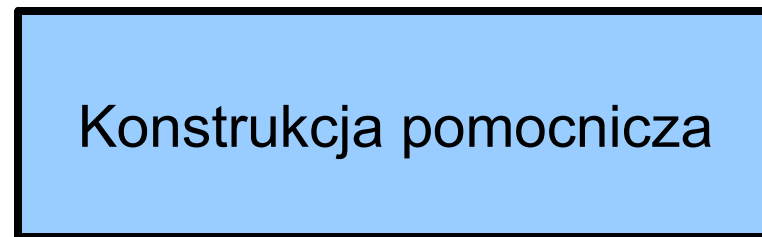
- Na bazie skończonej ilości symboli z **alfabetu języka** można stworzyć nieograniczoną ilość dowolnie złożonych konstrukcji
- Czy każda z nich będzie miała sens?
 - Nie!
- Do określenia, jakie konstrukcje są dopuszczalne, stosuje się **gramatykę języka programowania**

2.2. Gramatyka języka programowania (c.d.)

- **Gramatyka języka** jest *zbiorem reguł*, jednoznacznie określających, jak z symboli alfabetu języka powstają poprawne konstrukcje języka
- Podstawową formą zapisu gramatyki języków programowania jest tzw. **notacja Backusa-Naura** (*ang.* BNF, Backus-Naur Form)
 - Opracowana w latach 50. i szeroko stosowana do dziś
 - Ze względu na dość złożony charakter zagadnienia, szczegółowy opis notacji BNF wykracza poza zakres wykładu

2.2. Gramatyka języka programowania (c.d.)

- Graficznym odpowiednikiem BNF są **diagramy syntaktyczne**
- Pozwalają one na zapis reguł gramatyki języka w łatwej do zrozumienia formie graficznej
- Elementy diagramów syntaktycznych:



Łącznik



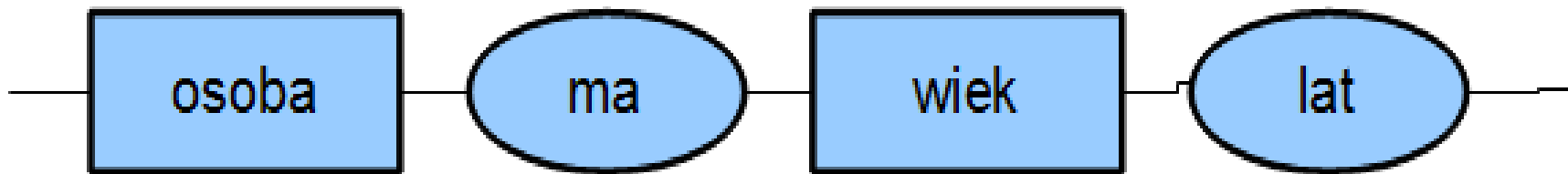
A horizontal line with a vertical drop at its right end, representing a connector.

2.2. Gramatyka języka programowania (c.d.)

- Zasada czytania diagramów syntaktycznych:
 - Każda konstrukcja wynikająca z diagramu syntaktycznego metodą przejścia dowolną dozwoloną ścieżką, jest poprawna
- Przykład:
 - Alfabet języka: { A .. Z, 0 ... 9 }
 - Gramatyka języka (p. następną stronę)

2.2. Gramatyka języka programowania (c.d.)

- Przykład (gramatyka języka)



Osoba:



Wiek:

2.2. Gramatyka języka programowania (c.d.)

- Przykład (c.d.)
 - Sekwencje będące poprawnymi dla omawianego języka:
 - ALA MA 20 LAT
 - PIOTR MA 15 LAT
 - NIKT MA 0 LAT
 - ABCDEF..Z MA 0123456789 LAT
- Wniosek:
 - Gramatyka pozwala określić, czy dana konstrukcja jest poprawna z punktu widzenia języka, lecz nie mówi o tym **czy ma ona jakikolwiek sens (!)**

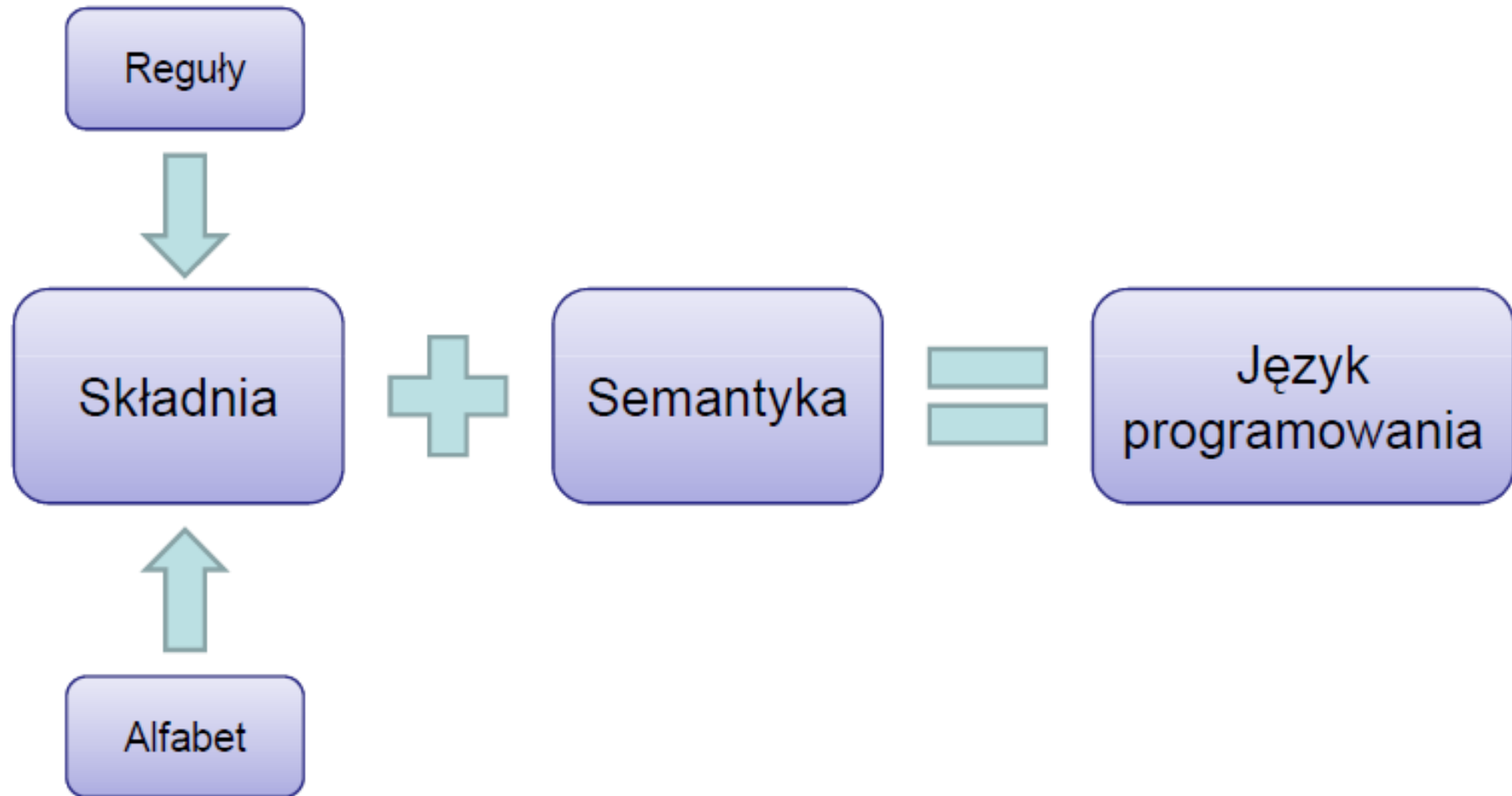
3. Semantyka języka programowania

- **Semantyka języka** – zbiór reguł opisujących co oznaczają poprawne konstrukcje języka i jak mają być interpretowane przez komputer
- Przykład:
 - Konstrukcja „ $A := B$ ” oznacza „wartość A przyjmuje wartość B”
 - Konstrukcja „IF($A < B$) .. ” oznacza, że „kod zostanie wykonany tylko jeśli wartość A będzie mniejsza od wartości B”
 - Konstrukcja $A < B > C$ nie ma sensu

3. Semantyka języka programowania (c.d.)

- Semantyka jest ściśle związana z konkretnym językiem programowania
 - Przykład: wyrażenia „ $A = B$ ” lub „ $B(10)$ ” w różnych językach programowania może oznaczać zupełnie co innego
- Do przechowania semantyki języka są stosowane różne metody formalnego zapisu
 - Stanowi to duży obszar wiedzy teoretycznej
 - Najbardziej prostą (i dość często stosowaną w praktyce) jest zwykła słowna postać

Składnia i semantyka języka programowania. Podsumowanie



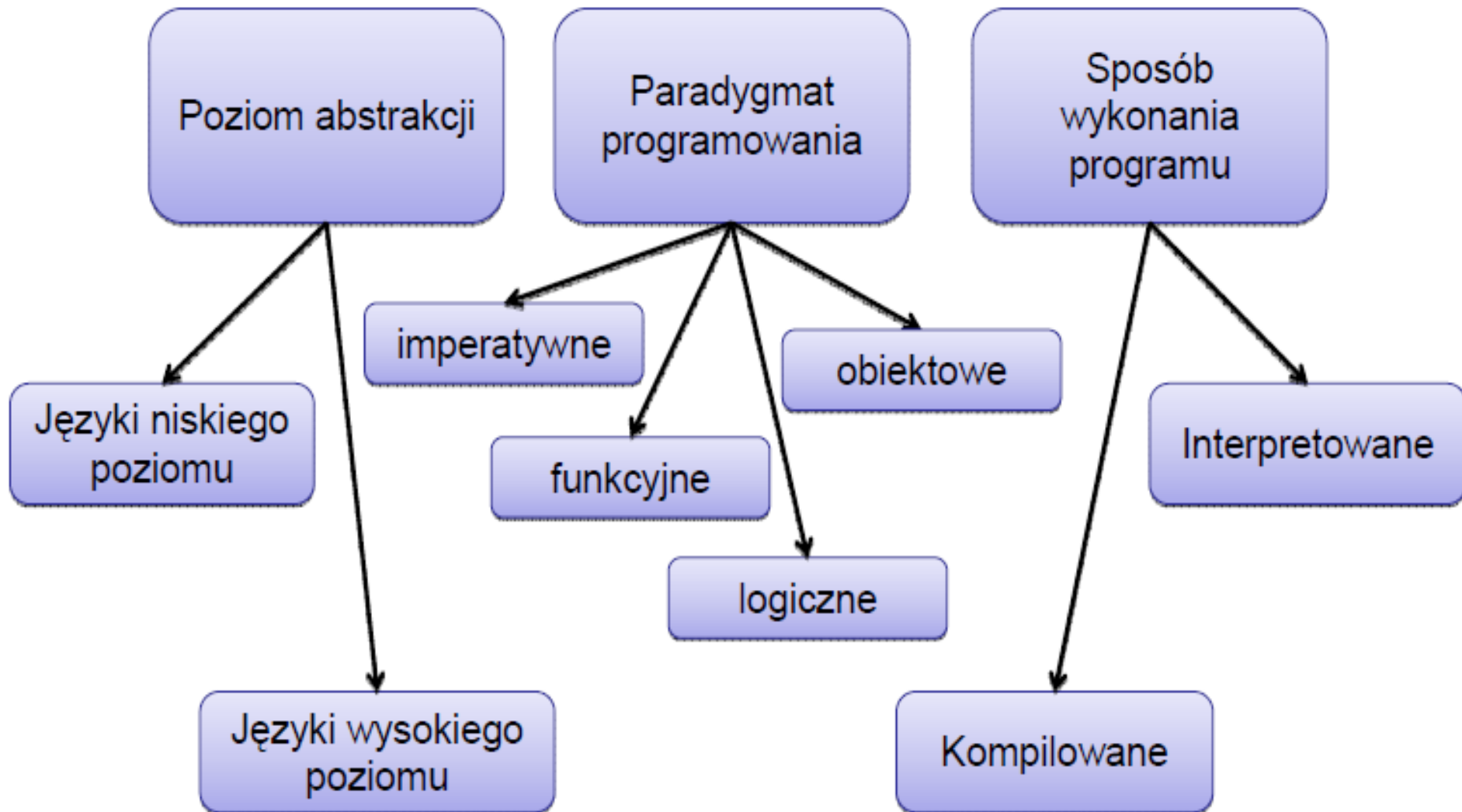
4. Klasyfikacja języków programowania

- Proces powstawania języków programowania trwa od lat 50. XX wieku aż do dziś
- W międzyczasie powstało kilka tysięcy języków programowania
 - w zależności od źródła oraz kryteriów ilość waha się w granicach 3 ... 8 tys.
 - Z tego zaledwie kilkaset znalazło (w swoim czasie) chociażby minimalne zastosowanie
- Aktywnie wykorzystywanych obecnie jest kilkadziesiąt języków programowania
- 90% kodu powstaje w 10 najbardziej popularnych

4. Klasyfikacja języków programowania (c.d.)

- W zależności od kryteriów porównania, wyróżniają kilka **kategorii klasyfikacji** języków programowania:
 - Poziom abstrakcji języka programowania
 - Generacja języka programowania
 - Sposób wykonania programu
 - Przynależność do paradygmatu programowania
- Inne (mniej istotne) kategorie klasyfikacji:
 - Przeznaczenie (uniwersalny / specjalistyczny)
 - Forma kontroli typów danych (silne / słabe)
 - itp.

4. Klasyfikacja języków programowania (c.d.)



4.1. Klasyfikacja języków programowania wg poziomu abstrakcji

- Języki programowania niskiego poziomu (języki maszynowe oraz tzw. języki Asemblera)
 - Zorientowane na konkretny typ procesora, instrukcje są dostosowane do jego specyfiki i są zapisywane w formie jednoznacznie (1:1) przekładającej się na kod binarny
- Języki programowania wysokiego poziomu
 - Gramatyka języka jest zorientowana na człowieka, język może nie być przywiązany do konkretnej platformy sprzętowej

4.1. Klasyfikacja języków programowania wg poziomu abstrakcji

- Języki niskiego poziomu

- Wydajny kod
- Krótki kod
- Optymalny kod
- Specjalistyczne zastosowania
- Przywiązanie do platformy sprzętowej

- Języki wysokiego poziomu

- Czytelny kod
- Łatwe programowanie
- Mniejsza ilość błędów
- Możliwość przeniesienia na inne platformy sprzętowe

4.1.1. Języki niskiego poziomu

- **Język maszynowy** – zapis programu w formie binarnej, który może być wykonany na komputerze bez dodatkowych translacji
 - W celu kompaktowego zapisu kodu binarnego jest on przedstawiany za pomocą systemu szesnastkowego (HEX)
 - W systemie szesnastkowym cztery bity kodu binarnego składają się na jedną wartość szesnastkową: 0000 .. 1111 => 0 .. A
- Przykład zapisu programu w kodzie maszynowym:

0D 0A 24 21 C3 48 BA 08 01 B4 09 CD 65 6C 6C 6F
2C 20 57 6F 72 6C 64 21

4.1.1. Języki niskiego poziomu (c.d.)

- **Język Asembler**

- Pisanie programów w kodzie maszynowym jest trudne, jeszcze zaś trudniejsze jest ich zrozumienie oraz poprawianie
- Wkrótce zasugerowano zastąpić kod szesnastkowy notacją symboliczną, elementy której byli by bardziej czytelne
- Przykład:
 - Instrukcja porównania: CMP (*compare*)
 - Instrukcja przejścia: JMP (*jump*)
 - Instrukcja dodawania ADD (*addition*)
 - Instrukcja kopiowania MOV (*move*)

4.1.1. Języki niskiego poziomu (c.d.)

- **Język Asembler (c.d.)**

- Pierwszy Asembler był programowany na taśmie perforowanej za pomocą urządzenia elektromechanicznego (rok 1945)
- Przykład kodu źródłowego programu

„Hello world”

```
org 100h
mov dx,msg
mov ah,9
int 21h
mov ah,4Ch
int 21h
msg db 'Hello, World! ',0Dh,0Ah,'$'
```


4.2. Klasyfikacja języków programowania wg sposobu wykonania programu

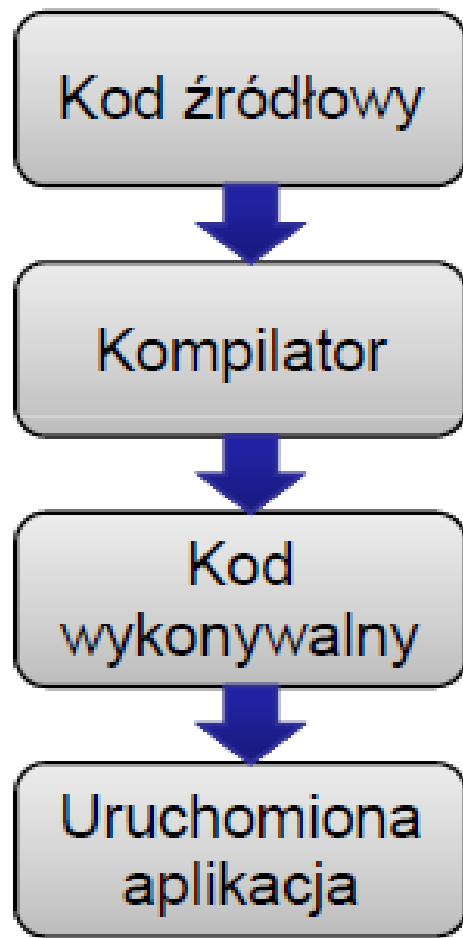
- **Języki kompilowane**

- Kod źródłowy programu jest **jednorazowo w całości** translowany do kodu maszynowego
- Następnie kod maszynowy jest wykonywany

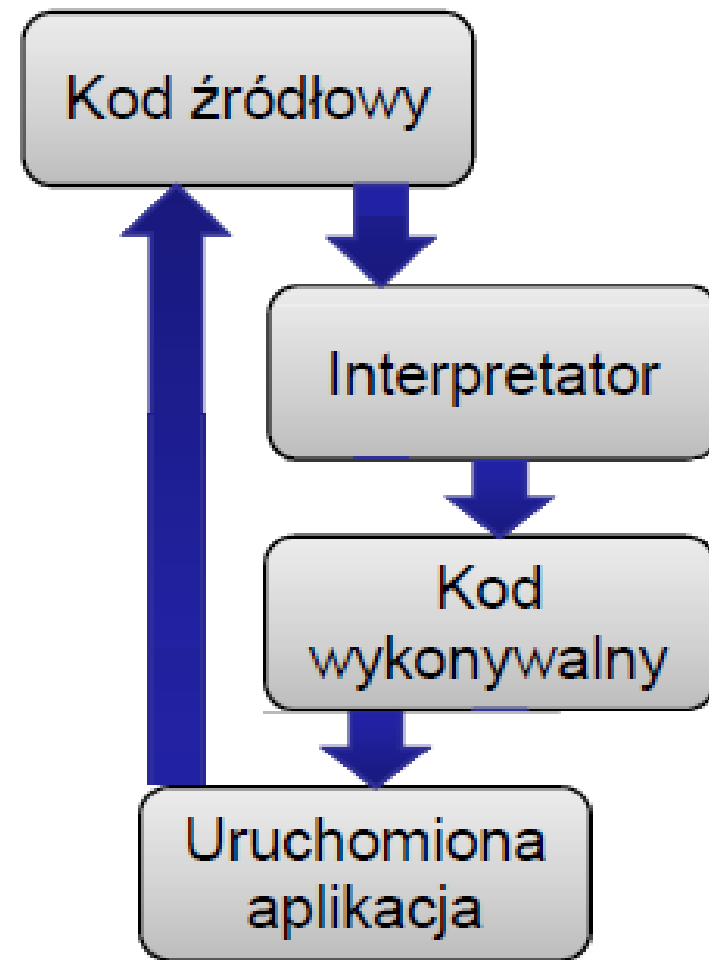
- **Języki interpretowane**

- Translacja kodu źródłowego na kod binarny odbywa się w momencie uruchomienia programu i wykonuje się stopniowo, instrukcja po instrukcji

4.2. Klasyfikacja języków programowania wg sposobu wykonania programu (c.d.)



Kompilowane



Interpretowane

4.2.1. Kompilowane języki programowania

- Cały kod źródłowy jest translowany na kod binarny
 - Program z błędami nie skompiluje się
 - Każda poprawka w tekście programu wymaga ponownej kompilacji
 - Jeśli program składa się z wielu modułów, zazwyczaj wystarczy skompilować tylko część zmienioną
 - Kod binarny jest kompilowany pod konkretną platformę sprzętową
 - Inna platforma często wymaga innego kompilatora
 - Do wykonania pliku binarnego programu nie są potrzebne pliki źródłowe

4.2.2. Interpretowane języki programowania

- Do wykonania programu jest potrzebny specjalny program – **interpretator** danego języka, zainstalowany na komputerze klienta
- Kod jest translatowany „w locie”, co pociąga za sobą następujące efekty:
 - Wykonanie programu rozpoczyna się natychmiast po uruchomieniu kodu
 - Optymalizacja kodu binarnego odbywa się względem konkretnego procesora (nie ogólnej platformy, jak to ma miejsce przy kompilacji)
 - Napotkany w tekście programu błąd może zatrzymać wykonanie całego programu

4.2.2. Interpretowane języki programowania (c.d.)

- Język BASIC (1964)
 - *ang.* **B**eginners **A**ll-purpose **S**ymbolic **I**nstruction **C**ode
 - Powstał w oparciu o języki Fortran i Algol, przy jednoczesnym założeniu:
 - Łatwości programowania
 - Uniwersalności zastosowań
 - Komunikatywności z użytkownikiem
 - Pierwsza wersja języka została uruchomiona na mocnych komputerach obliczeniowych komputerach typu *mainframe*

4.2.2. Interpretowane języki programowania (c.d.)

- Język BASIC (c.d.)
 - W połowie lat 70. zaczyna się epoka komputerów personalnych
 - Komputer staje się narzędziem nie tylko dla profesjonalistów, lecz również dla amatorów
 - Jest potrzebny nowy, prosty w oswajaniu oraz wygodny język programowania
 - Do wzrostu popularyzacji języka BASIC przyczynia się jego wersja napisana dla komputera personalnego Altair
 - Wersja nosi skromną nazwę GW BASIC
 - Pytanie: kto jest autorem języka?

4.2.2. Interpretowane języki programowania (c.d.)

- Język BASIC (c.d.)
 - W latach 80. rynek zalewają interpretatory BASIC dla komputerów 8-bitowych typu ZX Spectrum, Atari, Commodore itp.
 - Większość wersji języka nie jest kompatybilna
 - Microsoft przyczynia się do rozwoju BASIC, wypuszczając kolejne wersje języka:
 - QuickBasic (DOS)
 - Visual Basic (Windows)
 - Aktualna wersja Visual Basic .NET ma mało wspólnego z poprzednikami, stając się językiem obiekowym, kompilowanym, o silnej kontroli typów itp.

4.2.2. Interpretowane języki programowania (c.d.)

- Język BASIC (c.d.)
 - Wbrew zamiarom autorów, BASIC stał się synonimem amatorskiego podejścia do programowania:
 - Język opiera się na numeracji wierszy programu (wcześniejsze wersje), co utrudnia modyfikację programu
 - Nadużycie instrukcji przejścia GO TO (kod określany terminem „*spaghetti code*”)
 - Składnia języka nie trzyma się żadnych standardów (ponad 200 dialektów!)
 - Słaba kontrola typów danych
 - Dziwaczne nazewnictwo zmiennych itp.

4.2.2. Interpretowane języki programowania (c.d.)

- Język BASIC (c.d.)
 - Przykład programu „Hello, world!” wyjaśnia, dlaczego język stał się tak popularny:

```
| Print "Hello, world!"
```

4.3. Kompilacja vs. Interpretacja

- Każda technologia ma swoje niezaprzeczalne zalety, oraz niestety dość specyficzne wady
- **Język programowania może być jednocześnie kompilowanym oraz posiadać interpreter**
 - Praktyka pokazała, iż kompilowane programy są stosowane przeważnie w rozwiązaniach typu „desktop” (program uruchamiany lokalnie na komputerze), zaś interpretacja jest szeroko rozpowszechniona na platformach internetowych
 - Prawie każdy język interpretowany posiada obecnie również kompilator

4.3. Kompilacja vs. Interpretacja (c.d.)

- Obecnie szybko rozwijają się technologie (Java, .NET) łączące zalety obu podejść:
 - Kod źródłowy jest kompilowany do kodu pośredniego na komputerze programisty
 - Eliminowane są błędy, kod jest częściowo optymalizowany
 - Na komputerze docelowym instaluje się interpretator (*ang.* Framework)
 - Wykonanie programu polega na interpretacji kodu pośredniego
 - Translacja tylko niezbędnych fragmentów
 - Kod jest dalej optymalizowany pod konkretny procesor oraz system operacyjny

4.3. Klasyfikacja języków programowania wg generacji

- Wyróżniają 5 generacji języków programowania
- O przynależności języka do określonej generacji mogą decydować:
 - Okres powstania języka
 - Specyfika konstrukcji gramatycznych języka
 - np. sposób nazewnictwa słów kluczowych języka
 - Obszar zastosowania
 - Możliwości funkcjonalne itp.
- Granice generacji są nieco „rozmyte”
- Język może posiadać cechy kilku generacji

4.3. Klasyfikacja języków programowania wg generacji (c.d.)

- 1 generacja – języki maszynowe
- 2 generacja – języki Asemblera
- 3 generacja – języki zastosowania uniwersalnego
- 4 generacja – języki zorientowane problemowo
- 5 generacja – języki sztucznej inteligencji / języki wizualnego projektowania

4.3.1. Języki programowania 1. generacji

- Języki maszynowe

- Od momentu powstania komputerów do pojawienia się języków Asemblera reprezentują jedyny możliwy sposób programowania komputerów
- Zasada „1 wiersz programu = 1 instrukcja”
- Nie wymagają procesu translacji kodu źródłowego na kod binarny
- Obecnie nie są stosowane przez programistów, tylko automatyczne systemy generacji kodu (translatory z innych języków)

4.3.2. Języki programowania 2. generacji

- Języki Asemblera (*ang.* assembly – zbiórka, montaż)
 - Instrukcje procesora zapisane w kodzie binarnym zostają zastąpione przez nazwy symboliczne
 - Zwiększa się czytelność kodu, łatwiej i szybciej się programuje
 - Pierwsze języki powstają w latach 50., do dziś są aktywnie wykorzystywane w zastosowaniach wymagających szybkiego oraz lakonicznego kodu (sterowniki urządzeń, fragmenty systemów operacyjnych, systemy czasu rzeczywistego itp.)

4.3.3. Języki programowania 3. generacji

- Języki uniwersalne (początek lat 60.)
 - Pojawiają się w związku z wyjściem komputerów poza środowiska wojskowe oraz akademickie
 - Każdy chce się nauczyć programować!
 - Celem jest stworzenie języków pozwalających rozwiązać problemy z różnych dziedzin
 - Coraz bardziej złożone konstrukcje semantyczne przyczyniają się do zmniejszenia rozmiaru kodu
 - Wzrasta wydajność pracy programistów
 - Spora ilość tych języków jest stosowana do dziś

4.3.3. Języki programowania 3. generacji (c.d.)

- Język FORTRAN (FORmula TRANslator)
 - Pierwszy język programowania 3 generacji (!)
 - Powstał w latach 1954 – 1957
 - Przeznaczony do złożonych obliczeń naukowych
 - Posiada ogromną bibliotekę gotowych formuł stosowanych do dziś
 - Pierwszy język posiadający własny kompilator
 - Opracowany i rozwijany przez IBM
 - Oraz w różnych czasach przez inne firmy
 - Są komercyjne oraz bezpłatne wersje

4.3.3. Języki programowania 3. generacji (c.d.)

Język FORTRAN (c.d.)

- Wcześniejsze wersje języka nie pozwalały na zapis więcej niż 1 instrukcji w wierszu
- Swoistą „wizytówką” języka Fortran stała się bardzo często stosowana instrukcja przejścia bezwarunkowego GOTO, co znacznie utrudniało zrozumienie programu
- Przykład kodu źródłowego programu „Hello, World!”:

```
program hello  
print *, "Hello, World!"  
end
```

4.3.3. Języki programowania 3. generacji (c.d.)

- Język ALGOL (**ALGO**rithmic **L**anguage)
 - Pierwsza wersja opracowana w 1958 r., najbardziej znana Algol-68
 - Zorientowany na programowanie formuł matematycznych, przez dłuższy czas pozostawał standardem notacji oficjalnej
 - Bardzo popularny w Europie, nie zdobył uznania w USA (IBM promował Fortran)
 - Jako pierwszy pozwolił na stosowanie *rekurencji*
 - Problemem okazał się brak standardu implementacji na różnych komputerach

4.3.3. Języki programowania 3. generacji (c.d.)

Język ALGOL (c.d.)

- Koniec epoki języka ALGOL nastąpił w latach 80. XX wieku
- Program typu „Hello, World!” dla różnych wersji języka ALGOL

```
program HiFolks;  
begin  
    print "Hello, world";  
end;
```

```
'BEGIN'  
    OUTSTRING (1, '('HELLO, WORLD!')');  
'END'
```

4.3.3. Języki programowania 3. generacji (c.d.)

- Język PASCAL

- Autorem języka jest N. Wirth
 - Pierwszy język opracowany de-facto przez jedną osobę (wraz z grupą studentów)
- Pierwsza wersja została opublikowana w 1970r.
- Pierwotnie język planowano używać wyłącznie do celów edukacyjnych
- **Kompilator Pascala został napisany w języku Pascal (!)**
- Opracowano kilka różniących się standardów języka (co stało się przyczyną problemów)

4.3.3. Języki programowania 3. generacji (c.d.)

- Język PASCAL (c.d.)
 - W latach 70-80. XX wieku zyskał ogromną popularność
 - Brak komercyjnego podejścia do rozwoju języka sprawił jednak, iż różne firmy wycofały się z inwestycji w Pascala (m. innymi Microsoft)
 - Język posiadał też pewne wady konstrukcyjne, eliminacja których trwała zbyt długo
 - Największym sukcesem Pascala stała się jego implementacja pod nazwą Delphi
 - Obecnie język został zaimplementowany na platformie .NET, więc można uznać że istnieje

4.3.3. Języki programowania 3. generacji (c.d.)

- Język C (1972)
 - D. Ritchie (jeden z autorów systemu Unix) opracowuje nowoczesny język programowania
 - Pierwotnie mając na celu programowanie systemów operacyjnych, więc język musiał być
 - Stabilnym
 - Szybkim w działaniu
 - Uniwersalnym
 - Język po kilku niezbyt udanych próbach (wersje A i B) otrzymuje nazwę C
 - W roku 1980 język staje się ogólnie znanym i popularnym językiem
 - Standard języka ukazuje się w 1990 r.

4.3.3. Języki programowania 3. generacji (c.d.)

- Język C (c.d.)
 - Dzięki standardowi język C unika powtórzenia losu BASIC (mającego ogromną ilość niekompatybilnych dialektów)
 - Składnia języka jest bardzo lakoniczna (często ponad miarę), przy zachowaniu wysokiej funkcjonalności konstrukcji
 - np. „n++” w Pascal pisze się jako „n := n + 1”
 - Posiada możliwości programowania niskopoziomowego (z zachowaniem składni C zarówno jak za pomocą wstawek w języku Asembler)

4.3.3. Języki programowania 3. generacji (c.d.)

- Język C (c.d.)
 - Właśnie zdolność do programowania na niskim poziomie prowadzi do dominacji C jako języka do napisania sterowników urządzeń (sprzętu)
 - Język ma też sporo istotnych wad:
 - Wbudowane operacje na złożonych typach danych są prymitywne (np. nie można kopiować tablic)
 - Brak kontroli wykroczenia poza zakres typu danych
 - Podprogramy nie mogą być zagnieżdżone
 - Itp.
 - Większość z wad można wyeliminować

4.3.3. Języki programowania 3. generacji (c.d.)

- Język C (c.d.)
 - Język nie jest zalecany do nauczania się programowania (w odróżnieniu od Pascal np.)
 - Program napisany w C może być podatny na ataki hakerów wykorzystujących nieprzewidywane zachowanie się programu w przypadku wystąpienia błędu podczas wykonania
 - Dlatego nie jest stosowany w rozwiązaniach internetowych!
 - Posłużył bazą do języka obiektowego C++ (kompatybilność wsteczna niemal że pewna)

4.3.3. Języki programowania 3. generacji (c.d.)

- Język C (c.d.)
 - Kod źródłowy programu „Hello, world!”

```
#include <stdio.h>
main() {
    printf("Hello, world!\n");
}
```

4.3.3. Języki programowania 3. generacji (c.d.)

- Język BASIC (szczegółowo opisany w poprzednich rozdziałach)
 - Niewątpliwie jest (a raczej był) jednym z najbardziej popularnych języków 3. generacji
 - Ewoluował w kierunku amatorskiego programowania, często kojarząc się z programami prymitywnymi oraz niestabilnymi w działaniu
 - Obecnie nie jest rozwijany (jako język 3. gen), lecz na rynku wciąż pozostaje sporo programów w nim napisanych (a nawet wspieranych przez producenta)

4.3.4. Języki programowania 4. generacji

- Języki problemowo-zorientowane (koniec lat 70 – do dziś)
 - Przeznaczone do realizacji dużych, dynamicznie rozwijających się projektów
 - Problemy programowania dużych systemów informatycznych zostały genialnie opisane w książce F.Brooksa „Mityczny osobo-miesiąc”
 - Posiadają instrukcje, do realizacji których w językach wcześniejszych generacji potrzebne były setki bądź tysiące linii kodu
 - Orientacja na programowanie obiektowe (inne niż dotychczas podejście do konstruowania programu)

4.3.4. Języki programowania 4. generacji

- Języki 4. generacji charakteryzuje
 - Automatyczne zarządzanie pamięcią operacyjną komputera przy wykonaniu programu
 - Obiektowe podejście do programowania
 - O tym dalej w wykładzie „Paradygmaty ...”
 - Zaawansowana kontrola jakości kodu źródłowego
 - Możliwość kompilacji oraz interpretacji
 - Zastosowanie do programowania zarówno aplikacji klasycznych tak i internetowych
 - Możliwość przeniesienia kodu na inne platformy

4.3.4. Języki programowania 4. generacji (c.d.)

- Najbardziej popularne języki:
 - Java
 - C#
 - C++
 - Python
 - Ruby
 - Ada
 - Delphi
 - Itp.

4.3.4. Języki programowania 4. generacji (c.d.)

- Język C# (2001)
 - Jest bazowym językiem programowania platformy .NET firmy Microsoft
 - Swoją składnię (jak sama nazwa wskazuje) wywodzi z języka C / C++
 - Język uniwersalny, do wszechstronnych zastosowań
 - Posiada rozbudowane mechanizmy ułatwiające programowanie
 - Obecna wersja 4.0 realizuje większość znanych standardów programowania, zweryfikowanych czasem (w tym również na innych językach)

4.3.4. Języki programowania 4. generacji (c.d.)

- Język C# (c.d.)
 - Przykład kodu źródłowego aplikacji pobierającej z serwera WWW zawartość strony internetowej i wyświetlającej ją w konsoli

```
static void Main(string[] args) {
    if (args.Length != 1)
        Console.WriteLine("WebConsole [http://www.....]");
    else {
        try {
            WebRequest wr = WebRequest.Create(args[0]);
            WebResponse ws = wr.GetResponse();
            StreamReader sr = new StreamReader(ws.GetResponseStream());
            String content = sr.ReadToEnd();
            Console.Write(content);
        } catch (Exception ex) {
            Console.Write(ex.Message);
        }
    }
    Console.ReadLine();
}
```

4.3.5. Języki programowania 5. generacji

- Kolejna wpadka teoretyków od prognozowania!
 - W latach 90. wydawało się iż za chwilę nastąpi przełom w programowaniu, dzięki językom programowania sztucznej inteligencji (AI) – to one miały należeć do 5. generacji
- Obecnie do tej generacji zaliczają **języki wizualnego programowania**, wymagające minimalnej wiedzy o procesie kodowania
- Promuje się koncepcja odejścia od „kodowania” na rzecz „projektowania” systemów informatycznych (reszty ma dokonać komputer)

4.4. Klasyfikacja języków programowania wg kontroli typów danych

- Obiekty świata rzeczywistego należą do pewnych kategorii
 - Moneta: 1, 2, 5, 10, 20, 50
 - Data: 12/06/2010, 13/06/2010, ...
 - Imię: Jan, Łukasz, Piotr, ...
- **Dane** reprezentujące ww. obiekty w programie komputerowym, również należą do odpowiednich kategorii
- Kategoria danych nazywa się **typem danych**

4.4. Klasyfikacja języków programowania wg kontroli typów danych (c.d.)

- W zależności od tego, czy język zezwala na wykonanie operacji na danych różnych typów, rozróżniają:
 - Języki z silną kontrolą typów
 - Data nie może być porównana bezpośrednio np. z nazwiskiem, od imienia nie można odjąć wieku osoby itp.
 - Języki ze słabą kontrolą typów
 - Dopuszcza się automatyczna konwersja pomiędzy danymi różnych typów, jeśli jest ona jednoznaczna, np. sumowanie wiersza z liczbą automatycznie konwertuje liczbę na tekst:
 - „Jaś ma ” + 10 + „ lat”

4.4. Klasyfikacja języków programowania wg kontroli typów danych (c.d.)

- W przyrodzie nie ma języków z absolutnie silną bądź słabą kontrolą typów:
 - Absolutnie silny język nie byłby w stanie funkcjonować ze względu na konieczność opracowania dużej ilości mechanizmów konwersji pomiędzy wieloma typami
 - Jednym z najbardziej rygorystycznych języków jest PASCAL, do silnych zaliczamy również język C
 - Absolutnie słaby język zrealizować jest łatwiej, lecz działanie programu w nim napisanego często kończyłoby się błędem ze względu na nieudaną próbę konwersji
 - Język Basic we wcześniejszych realizacjach

4.5. Klasyfikacja języków programowania wg paradygmatu

Szczegółowo zostanie omówiono w następnym wykładzie

Podsumowanie

- Języki programowania powstały jako zbiory instrukcji, mające na celu zapis algorytmu w formie zrozumiałej dla człowieka oraz umożliwiającej jego jednoznaczną translację do postaci wykonywalnej na komputerze
- Języki programowania mają takie cechy języków naturalnych, jak alfabet, gramatyka, składnia oraz semantyka
- W zależności od momentu powstania oraz oferowanej funkcjonalności dzielą się na 5 generacji

Podsumowanie (c.d.)

- W zależności od sposobu wykonania na komputerze, dzielą się na języki kompilowane oraz interpretowane
- Wszystkie języki programowania należące do 3. lub późniejszej generacji nazywane są językami wysokiego poziomu, zaś języki Asemblera oraz kody maszynowe tworzą grupę języków niskiego poziomu abstrakcji
- Języki programowania wciąż są aktywnie modyfikowane i rozwijane, powstają również całkowicie nowe języki