Języki programowania #4

a. Asembler
b. Abstrakcji
c. Języki programowania wysokiego poziomu
d. Kompilacja

Uczynić kontakt człowiek - komputer bardziej ludzkim

Od kiedy powstały komputery, ich twórcą (a szczególnie twórcą oprogramowania) towarzyszyła chęć i potrzeba uczynienia programowania bardziej ‘ludzkim’. Pierwszym krokiem do tego celu stały się języki niższego poziomu, zwane językami symbolicznymi (językami asemblera, ang. assembly language).

Uczynić kontakt człowiek - komputer bardziej ludzkim - SUKCES!

Najpopularniejsze obecnie języki programowania powstały przede wszystkim w latach 90-tych (Java, Python, JavaScript, PHP). Jednak historia języków programowania zaczęła się długo wcześniej, już w latach 50-tych. Bardzo często twórcy popularnych obecnie język programowania opierali je o różnych poprzedników (np. Java czerpie z C++, C++ jest rozwinięciem C, a C inspirowało się B, zaś język B inspirował się ...).

Języków programowania (których używa więcej osób niż tylko ich twórca) są tysiące. Te najbardziej popularne są mocno rozwijane. Pojawiają się wciąż także nowe (już w XXI wieku powstały np. C# czy TypeScript).

Świat współczesnych języków programowania

Języków programowania (których używa więcej osób niż tylko ich twórca) są tysiące. Te najbardziej popularne są mocno rozwijane. Pojawiają się wciąż także nowe (już w XXI wieku powstały np. C# czy TypeScript).

Niskopoziomowy język programowania

Języki niskiego poziomu, czyli prawie język maszynowy.

“Niskopoziomowość” można tłumaczyć jako niski poziom względem ‘odległości’ od sprzętu, czy bardziej precyzyjnie odległość od instrukcji/rozkazów procesora.

Rozkazy dla procesora są w nim wydawane nie w systemie binarnym, a bliższym ludziom wyrazom-symbolom oraz możliwe jest użycie systemu szesnastkowego czy dziesiętnego.

Czyli zamiast 0101110101001001 programista może napisać już np. Move, Push, Delete, Add.

*Mała uwaga. Może się zdarzyć że ktoś nazywa językiem niskiego poziomu takie języki jak **C (a nawet C++)**. To zawsze kwestia konwencji, więc nie ma o co kruszyć kopii.

Niskopoziomowy a wysokopoziomowy - ABSTRAKCJA

Poziom abstrakcji - brak abstrakcji oznacza pokrycie 1 do 1 z tym co dzieje się na niskim poziomie sprzętowym czyli na poziomie samego procesora (podstawowe instrukcje/rozkazy). Czyli 0 i 1 zdefiniowane dla rozkazów dla procesora i przekazywanych wartości (argumentów). Język maszynowy jest więc językiem o zerowym poziomie abstrakcji.

Pewien poziom abstrakcji zaczyna się gdy odpowiedni rozkaz dla procesora nie jest wyrażony bezpośrednio w języku maszynowym, ale za pomocą symboli (nazw) zrozumiałych dla człowieka (ADD zamiast 01000001), czy np. w systemie dziesiętnym, a nie dwójkowym (czyli 5 a nie 00000101).

Niskopoziomowy a wysokopoziomowy - abstrakcja

W językach wysoko poziomowych poziom abstrakcji rośnie i przejawia się, czy to w pojawiających się m.in. w kolejnych typach danych czy instrukcjach sterujących (tablice, listy, pętle, instrukcje warunkowe, funkcje, klasy itd.). Pojawiają się również instrukcje i wyrażenia, które korzystają z logiki i słownictwa języków naturalnych. W praktyce jedna instrukcja w języku programowania wysokiego poziomu przekłada się na mnóstwo instrukcji w języku maszynowym.

We współczesnych językach wysokopoziomowych w ogóle nie pracuje się (wręcz nie ma się świadomości) działań na poziomie procesora i nic o procesorze nie trzeba (w 99.9 proc. przypadków) wiedzieć.

Czy abstrakcja jest dobra?

Zdecydowanie! To proces, który uprościł, oczywiście z punktu widzenia człowieka, nie komputera ;), tworzenia oprogramowania.

Abstrakcja ma wiele korzyści, umożliwia:

- Szybsze programowanie - jedna instrukcja zastępuje mnóstwo instrukcji w kodzie maszynowym
- Bardziej ludzki programowanie - sposób programowania mający analogie, ze światem rzeczywistym, a więc i bardziej zrozumiały dla umysłu człowieka (np. Programowanie strukturalne, obiektowe)
- Skupienie się na logice programu - możliwość koncentrowania się na tym co ma robić komputer, a nie na działaniu procesora czy pisanie mnóstwa instrukcji
- Zmniejszenie liczby szczegółów, którymi musi zajmować się programista
- Pisania kodu, który w zakresie słów kluczowych składa się przede wszystkim ze słów w języku angielskim (np. if, function, class, int, string, for) a nazwy mogą być wyrażane w którymś z języków (najczęściej również angielskim)
- Zastosowanie różnych technik programowania (paradygmatów programowania) z których najpopularniejsze współcześnie to programowanie obiektowe.

Assembler (pol. Asembler)

1. Język Asemblera - inne określenie, to język symboliczny
2. Istnieje wiele języków Asemblera
3. Pojawił się już w latach 40-tych dając możliwość napisania kodu maszynowego w bardziej ‘ludzki sposób’.
4. Język Asemblera wymagał dodatkowego programu (nazywanego asemblerem), który tłumaczył język Asemblera (symbole - mnemoniki) na język maszynowy.
5. Język Asemblera nie rozwiązywał problem pisania na różne procesory oraz pisania instrukcji dla każdego rozkazu procesora. Ale pozwalał tworzyć program trochę łatwiej.
6. W swojej pracy 99.9 proc. Programistów nie będzie używać Asemblera.

Słowniczek Asemblera

Język Asemblera (czasami po prostu Asembler) - niskopoziomowe języki programowania.

asembler(y) - program(y) zmieniający kod źródłowy napisany w języku Asemblera na kod maszynowy. Ten proces zamiany określamy asemblacją.

Mnemonik – w językach Asemblera jest to składający się z kilku liter kod-słowo (symbol, stąd język symboliczny), który oznacza konkretną czynność, jaką ma wykonać procesor. Zastępuje rozkaz wyrażony w kodzie binarnym.

Języki wysokiego poziomu

Język wysokiego poziomu, charakteryzujący się kilkoma cechami, z których warto wymienić dwie:

-> wymaga kompilacji - kompilacja jest bardziej skomplikowanym procesem niż asemblacja (obejmuje wielostopniową analizę oraz optymalizację). W wyniku kompilacji powstaje kod wynikowy, który może mieć charakter kodu maszynowego, ale często powstaje też kod pośredni.
-> ma wysoki poziom abstrakcji

Najpopularniejsze języki programowania (kol. alfabetyczna)

C#

C++

Java

JavaScript

PHP

Python

Najpopularniejsze języki programowania (kol. alfabetyczna)

C#

C++

Java

JavaScript

PHP

Python

Więcej na ich temat dowiesz się już w przedmiocie “Fundamenty Programowania”
Teraz skupmy się na najważniejszych, uniwersalnych aspektach języków programowania.

Słowa kluczowe (semantyka)

W każdym języku programowania istnieje lista określonych słów i symboli, które mają znaczenie z punktu widzenia programu.

W wielu języka te słowa kluczowe się pokrywają np. int, string, function, public, void, class if, while, main czy symbole “ ` ; | & itd.

Języki programowania nadają więc znaczenie pewnym słowom.

Składnia (syntax)

Mając już słownictwo potrzebujemy jeszcze zasad ich używania czy łączenia. Szukając analogii znajdziemy je w gramatyce języków naturalnych.

Kolejność, w jaki sposób łączyć, czego nie wolno używać, a co w danej sytuacji jest wymagane oraz jaki będzie efekt użycia. I znaczenie, znacznie więcej...

Język naturalny też ma zasady. Jeśli nawet używamy słów, ale nie łączymy ich prawidłowo, to jest to niezrozumiałe, przy czym ludzie potrafią ‘zrozumieć’ mimo błędów. Komputer (najczęściej kompilator wcześniej) ma znacznie niższy próg tolerancji.

Typy danych

Program operuje na danych, te typy są zbliżone w różnych językach programowania, ale zdarzają się różnice. Najczęściej występują takie typy danych jak liczby całkowite, liczby zmiennoprzecinkowe, znaki, stringi (ciągi tekstowe), wartości logiczne. Dostępne są także bardziej złożone struktury jak tablice, funkcje, klasy czy obiekty.

Operatory

= + - | === & > ! new ++

-- arytmetyczne (matematyczne)

-- logiczne (wynik końcowy to prawda i fałsz np. Koniunkcja i alternatywa)

-- relacji (np. równe, negacja, większe)

Prawda - fałsz

Czy języki są do siebie podobne?

Prawda - fałsz

Czy języki są do siebie podobne?

Tak, bardzo, szczególnie z tych samych paradygmatów (większość popularnych języków, to tzw. języki obiektowe).

Różnice są, ale małe.

Prawda - fałsz

Czy języki są do siebie podobne?

Tak, bardzo, szczególnie z tych samych paradygmatów (większość popularnych języków, to tzw. języki obiektowe). Różnice są, ale małe.

Czy trzeba znać język programowania aby programować?

Prawda - fałsz

Czy języki są do siebie podobne?

Tak, bardzo, szczególnie z tych samych paradygmatów (większość popularnych języków, to tzw. języki obiektowe). Różnice są, ale małe.

Czy trzeba znać język programowania aby programować?

Tak, ale... rozwiązania dla dzieci czy program z GUI tworzący kod.

Dziękuję za dzisiaj
i do zobaczenia jutro!