Introductie Programmeertalen

Het doel van deze korte samenvattingen is tweeledig:

- mensen d ie evt. willen beginnen met programmeren een handvat te geven waar te beginnen
- mensen die al een programmeertaal kennen aan andere talen laten ruiken

Slide 3
Het plaatje komt uit een boek over programmeertalen uit de jaren 60. Toen al waren er heel veel programmeertalen.

Slide 4
Een verzameling van programmeertalen van Wikipedia geplukt en dat is nog maar een kleine selectie. Waarom zoveel? Komen we nog op

Slide 5
Allereerst: waarom wil je programmeren?

Slide 6
Geschiedenis van programmeertalen
Allereerste programmeurs codeerden rechtstreeks in de machinetaal. De bytes (of wat het ook waren) werden via knopjes en schakelaars gevuld.

Al snel werden er assemblers gemaakt. Dan hoefde je niet meer de machinecodes uit je hoofd te leren en zelf de geheugenadressen bij te houden.

Slide 7
Rond 1960 werden de eerste hogere programmeertalen bedacht en ook geïmplementeerd.
Toen onstonden er ook talen die op een bepaald type problemen toegesneden waren.

Cobol
De grote animator hiervan was Grace Hopper. Een echte computerpionier. Ze was rear admiral in de jaren 50. Het DoD eiste dat iedere leverancier die wilde leveren aan het DoD Cobol voor Cobol moest zorgen.
Cobol was bedoeld voor administratieve toepassingen. Het idee was dat de taal zo beschrijvend was dat ook niet-programmeurs het konden begrijpen.
Er is nog heel veel Cobol-programmatuur in omloop mn in de financiële wereld. De verwachting is dat er op korte termijn tekort aan Cobol-programmeurs zal ontstaan. De mensen die nog Cobol kennen gaan of zijn met pensioen en jongeren leren geen Cobol meer.

Fortran
Fortran was bedoeld voor het hard-core rekenwerk. De eerste versie van Fortran bevatte wat merkwaardige restricties die voorkwamen uit de beperkingen van de IBM-computer waarop de eerste compilers gemaakt zijn.
Er bestaat nog heel veel Fortran-programmatuur.
Astronomen programmeren nog steeds veel in Fortran. De Fortran-compilers zijn vaak heel goed in het optimaliseren van programma’s.

Algol
Algol was ontworpen als publicatietaal voor wiskundigen. De taal bevatte ook elementen die theoretisch heel mooi waren maar uiterst moeilijk te implementeren waren.
Algol zelf zal naw niet meer gebruikt worden. Maar in andere vormen leeft Algol nog steeds.
Een heleboel concepten die in Algol het eerst opduiken komen terug in talen als Pascal, C, Python.
Denk daarbij aan begrippen als

– blokstructuren
– lokale en globale variabelen
– recursie

Hier zien we al dat talen uiteen gaan lopen: je kunt best een administratieve toepassing in Fortran schrijven maar echt handig is het niet.
Ik heb zelf programmatuur gemaakt die gegevens uit kassa's via netwerkverbindingen binnenhaalde. Dat was in Cobol geschreven. De enige compiler die we aangeschaft hadden.

Slide 8
De volgende stap is een nog hoger abstractieniveau en dan komen we bij talen die bedoeld zijn voor een specifiek probleemdomein: SQL en Postgress 4GL

Slide 9
In alle programmeertalen die we tot dusverre behandeld hebben beschrijven we de manier waarop we een probleem willen oplossen. Er zijn ook talen als Prolog waarbij we het probleem beschrijven. Dan komen we op het terrein van de KI.

Slide 10
Er zijn programmeertalen die gecompileerd moeten worden. Dat houdt in dat het programma in een aantal stappen wordt omgezet in machinecode.
Er zijn ook talen die geïnterpreteerd worden. Het programma wordt ingelezen en direct uitgevoerd.
In sommige situaties wordt zelfs telkens de brontekst bekeken. Mn command-shells werken vaak zo.
Andere interpreters vertalen eerst de brontekst in een interne code.
Als een interpreter dat niet doet dan kun je bijv. de situatie krijgen dat er al wat van je programma uitgevoerd wordt en de uitvoering wordt afgebroken omdat je een syntax-fout hebt gemaakt.

Slide 11
Een groot voordeel van talen die gecompileerd worden is dat het programma sneller zal lopen, dwz minder CPU-tijd verbruikt.
Maar hoe belangrijk is dat? Maakt het uit of de quad cores van je PC voor 99% niets staan te doen of voor 90%?
Als het programma voornamelijk bestanden leest of gegevens via internet naar binnenhaalt dan is de CPU niet de belemmerende factor. Als je zware beelden van CT-scanners verwerkt wellicht wel.

Een groot voordeel van scripttalen als Perl, Python en Ruby is dat de talen zelf mogelijkheden bieden die de gecompileerde talen alleen via libraries kennen. Bijv. array's die automatisch groeien, ingebouwde garbage collection waardoor de programmeur zich niet met het beheer van geheugen hoeft bezig te houden. Één van de grootste veroorzakers van bugs in C-programma's..

Een taal als Java neemt een eigen positie in. De Java-compiler vertaalt het programma naar de interne code voor een JVM (een Java Virtual Machine). Dat betekent als je het vertaalde programma naar een heel ander platform overbrengt waar ook een JVM is, het programma het doet. Het "Write once, run

everywhere”-principe.

Slide 12
Een ander onderscheid is het paradigma waar de taal vanuit gaat: het denkkader.

Imperatieve talen als C, Fortran e.d. resulteren een stijl van programmeren waarbij je gegevens hebt en programmacode die iets met die gegevens doet.

Als je object-geörienteerd denkt, dan denk je in classes en objecten. Bijv. artikelen, orders, klanten. Die objecten bevatten gegevens (artikelnummers, aantallen, prijzen) en methoden die iets met die gegevens doen.
Talen als Smalltalk en Eiffel zijn puur object georienteerd.
Talen als C++, Python en Ruby ‘verleiden’ je tot een object-georienteerde programmeerstijl.
Talen als Perl en PHP geven je de mogelijkheid om, als je dat wilt, object-georienteerd te werken.
In een taal als C kun je wel object-georienteerde technieken toepassen maar de taal ondersteunt het niet.

Functioneel programmeren houdt in dat je denkt in termen van functies zoals dit begrip binnen de wiskunde gehanteerd wordt. Variabelen e.d. zijn uit de boze.
De functionele taal is Lisp maar er zijn er meer: APL, Erlang, F♯, Haskell, ML, Scala en Scheme, waarvan Haskell de puurste is.
Het grote voordeel van functioneel programmeren is dat het gemakkelijker aan te tonen is dat een programma correct is.

Bij logisch programmeren denken we aan Prolog. Ik heb daar geen enkele kennis over.

Slide 13
In de presentaties die volgen hebben we, vanwege de tijd, geen ruimte voor twee belangrijke talen die we toch in het kort willen aanstippen.

Perl is een scripttaal uit de Unix-wereld. Kenmerken zijn

- uiterst krachtig
- voor een scripttaal heel efficient
- heel sterk in het verwerken van tekststromen: bestanden, output uit commando’s etc.

Het antwoord op de vraag “Kun je Perl gebruiken voor ….” is meestal ja. Alleen bij sommige zaken moet je zeggen: het kan wel maar het is misschien niet zo slim.
Perl heeft de naam cryptische code op te leveren. Mijn stelling is dat dat meer te maken heeft met de kwaliteit van de programmeur dan van de taal.

Slide 14
Voorbeeld

Slide 15
Bij het begin van het World Wide Web was Perl de taal waarin webapplicaties geschreven werden. Tegenwoordig wordt die positie door twee talen ingenomen: ASP (Active Server Pages) voor applicaties die onder IIS van MS moeten draaien en PHP voor applicaties die onder Apache moeten draaien.

Een kenmerk van PHP is dat je code en HTML kunt mengen. Een kanttekening daarbij is dat, zodra je wat grotere applicaties gaat ontwikkelen je HTML en code gaat scheiden. Een model wat daarbij vaak gebruikt wordt is het MVC-model: models, views en controllers.
Models representeren de data waar de applicatie mee werkt (bijv. tabellen uit een database).
Views zorgen voor het genereren van HTML en controllers bevatten de applicatielogica.

Slide 16
Een vraag die aspirant-programmeurs vaak stellen: met welke taal moet ik beginnen?
Ik ben zelf een Perl-junk maar ik zal niemand aanraden om met Perl te beginnen.
Het antwoord hangt ook af van wat je wilt bereiken: als je wel eens incidenteel een programmaatje wilt maken: begin met Basic of Scratch
Als je professionele ambities hebt, zou ik een scripttaal als Python of Ruby aanraden. Als je dat beheerst zou ik verder gaan met C of C++. Als je web-applicaties wilt ontwikkelen: leer PHP.

En als je een goede boterham wilt verdienen kun je dat waarschijnlijk heel goed bereiken door te zorgen dat je Cobol en/of Fortran beheerst: de vraag op die arbeidsmarkt wordt kleiner maar het aanbod waarschijnlijk nog veel sneller. In die niche is nmm geld te verdienen.