* Probleem oplossen
  + Specifieke aanpak vs Algemene aanpak
  + [D] Een **programma** (of computerprogramma) is een lijst van instructies die door de computer kunnen worden uitgevoerd, en die tot doel hebben een probleem op te lossen.
  + Invoervariabelen -> verwerkingsproces -> uitvoervariabelen
  + [D] Een **algoritme** is een exacte beschrijving van een gegevensverwerkend proces en van de invoer- en uitvoervariabelen van dat proces.
  + Een algoritme moet aan volgende vereisten voldoen
    - **Invoer** en **uitvoer** moeten duidelijk gespecifieerd zijn
    - Het moet **robuust** zijn
    - Het moet **uitvoerbaar** zijn
    - Het moet **eenduidig** zijn
    - Het moet **eindig** zijn
    - Het moet **algemeen** zijn
    - Het moet **correct** zijn
    - Het moet **efficiënt** zijn
  + Gestructureerd programmeren, uitgedrukt dmv
    - Pseudocode
    - Stroomdiagram
    - Nassi-Schneidermann-diagrammen (= PSD, **P**rogramma **S**tructuur **D**iagrammen)
  + Computertaal
    - Eerste generatie (1GL)
      * ‘Enen en nullen’ -> binaire code / machinetaal
      * **Compiler**: programma om hoger niveau taal om te zetten in machinetaal
    - Assembleertalen (2GL)
    - COBOL, Java, C, … (3GL)
    - Peoplesoft, SQL, … (4GL)
    - Prolog (5GL)
* Variabelen
  + [D] Een **variabele** is een gereserveerde geheugenplaats en de naam die daaraan wordt gegeven in het programma. Een variabele wordt gekenmerkt door de naam, de geheugenplaats, de waarde en het gegevenstype.
  + Naam verwijst naar een geheugenplaats waar een waarde gestockeerd wordt van een bepaald gegevenstype.
  + Een naam van een variabele moet **betekenisvol** en **uniek** zijn
  + De cursus programmatielogica gebruikt **lowerCamelCase**
  + **Toekenningsopdracht / Assignment**
    - <variabele> := <waarde> getal := 27
    - Of algemener: <variabele> := <expressie>
  + Variabelen moeten eerst **geïnitialiseerd** worden, bijvoorbeeld met een constante waarde of met een Read command.
  + Nog voor de initialisatie, is het nodig de variabele te **declareren**. Declaraties staan best aan het begin van het programma.
* PSD syntax
  + **<variabele> := <expressie>** ASSIGNMENT
  + **Read** mijnVariabele WACHTEN OP INVOER
    - Meerdere variabelen: Read v1, v2, v3
  + **Write** mijnVariabele GEGEVENS UITVOEREN
  + **var** <variabele1>, …, <variabeleN>: <gegevenstype> DECLARATIE
  + In lazarus moet je haakjes gebruiken bij combinaties van logische operatoren. Je kan niet zuiver vertrouwen op de prioriteiten.
* Gegevenstypes
  + [D] Een **gegevenstype** is een waardenverzameling met een aantal daarop gedefinieerde bewerkingen.
  + De **waardenverzameling** omvat alle mogelijke waarden die in een variabele van dat type kunnen worden opgeslagen. De grootte ervan is afhankelijk van de hoeveelheid geheugenruimte er aan dat gegevenstype gekoppeld is.
  + Enkelvoudig
    - **Integer**
      * = Gehele getallen, maar dan begrensd door de hoeveelheid geheugenruimte die toegekend is aan het type
      * ‘Kleine’ int gebruikt 2 bytes, met 1 bit voor het teken. 215 = 32768 🡺 [-32768, 32768]
      * ‘Grote’ int gebruikt direct 4 bytes, met 1 bit voor het teken. 231 = 2147483648 🡺 [-2147483648, 2147483648]
      * **Overflow** kan leiden tot een verkeerd resultaat of een error
      * **Deling door 0** leidt tot een error
    - **Real**
      * = Reële getallen, maar dan begrensd zowel in magnitude als in nauwkeurigheid
      * Vaak wordt er gebruik gemaakt van de **floating point** notatie. Dit is dezelfde manier van representatie als de **wetenschappelijke notatie**
        + Een aantal bits dienen dus voor de **beduidende cijfers**. Een tweede aantal voor de **exponent**. Doorgaans wordt dit nog aangevuld met één bit voor het **teken**.
    - **Char**
      * De waardenverzameling heet hier een **alfabet**
        + Bv ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange)

Eerst waren dit 7 bits + 1 controle => 27 = 128 mogelijkheden

Dan Extended ASCII of ASCII-II met 28 = 256 mogelijkheden

Nu Unicode met 216 = 65536 mogelijkheden

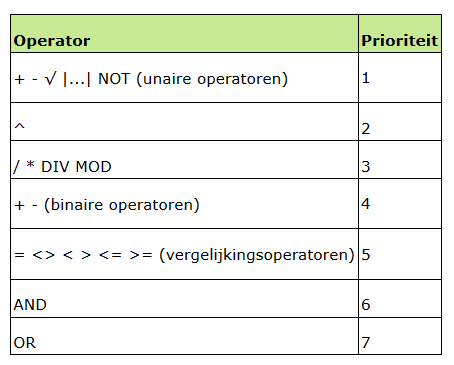
* + - **Boolean**
      * 1 of 0, True of False. Theoretisch heb je maar één bit nodig om dit weer te geven. In de praktijk vermijdt men de vertragingen die met *bit shift operations* gepaard gaan door een hele byte te gebruiken (kleinste adresseerbare geheugenruimte). De waardenverzameling blijft echter maar twee elementen bevatten.
  + Samengesteld
    - **String**
      * Reeks karakters uit een bepaalde waardenverzameling
    - **Array**
      * Niet echt een gegevenstype, maar een lijst van gelijksoortige gegevens, gekenmerkt door een **naam** en een **index**.
      * Elk gegeven is een **element**
* Operatoren
  + [D] **Operatoren** zijn bewerkingen die op één of meer **operanden** kunnen worden toegepast. Operanden zijn **variabelen, constanten of expressies**.
  + Operatoren kunnen aangeduid worden dmv symbolen of gereserveerde woorden
  + Opdeling op basis van
    - **Aantal operanden** waarop de operator werkt (/kan werken)
      * **Unaire** operatoren werken op één operand
        + bv negatief-teken (-) of wortelteken
      * **Binaire** operatoren werken op twee operanden.
        + Bv som-teken (+) of groter dan teken (>)
      * **Ternaire** operatoren werken op drie operanden
        + Bv ternary operator (<boolean expression> ? <result if true> : <result if false>)
    - **Soort bewerking** dat de operator doet, en bij gevolg, **soort operanden** waarop de operator werkt (/kan werken)
      * **Wiskundige operatoren**
        + Van toepassing op getallen, bijvoorbeeld optelling
      * **Logische operatoren**
        + Van toepassing op logische waarden, bijvoorbeeld

**Nevenschikking (AND)**

**Alternatief (OR)**

**Ontkenning (NOT)**

* + - * **Vergelijkingsoperatoren**
        + Van toepassing op alle soorten gegevens
  + Zoals in de wiskunde, hebben operatoren **prioriteiten**.
    - **Operatoren met gelijke prioriteit** worden geëvalueerd **van links naar rechts**.
    - Om af te wijken van de geijkte prioriteiten, gebruik je **haakjes**
    - Prioriteitentabel toont de operatoren met de hoogste prioriteit (1) tot en met de laagste (7)
    - !(A && B) == !A || !B
    - !(A || B) == !A && !B



* Basisstructuren van een programma
  + Sequentie
  + Selectie
  + Iteratie