Practicum P1 — Herfst 2006 **Opgave 2**

Leerdoelen: Representatie, Algoritmen, Controlstructuren, Variabelen, Functies

Leer Karel rekenen

In de eerste opgave heb je Kareltje de robot wat laten tekenen en z'n weg door een doolhof laten zoeken. In deze opdracht ga jij hem enige rekenvaardigheden bijbrengen. Karel's wereld veranderen we daarvoor niet, dat houdt in dat we geen gebruik gaan maken van decimale cijfers, maar ballen gaan gebruiken om getallen mee te representeren. De makkelijkste manier is om gebruik te maken van een zogenaamd unair talstelsel waarin bijvoorbeeld het getal 13 wordt gerepresenteerd door 13 ballen. Tijdens het Introductie Informatica en Informatiekunde-college heb je geleerd dat deze manier van representeren niet erg efficiënt is wat betreft het ruimte gebruik, vandaar dat we een andere methode zullen gebruiken. Het idee is om met een *tweetallig* of *binair* talstelsel te gaan werken. Voor het getal N hebben we dan slechts $\log(N)$ velden nodig.

Zoals je weet uit III heb je in het binaire stelsel twee cijfers, te weten 0 en 1. Het cijfer 0 representeren we in Karel's wereld met een lege plek, het cijfer 1 met een bal. In plaats van cijfers spreken we in het tweetallig talstelsel ook wel van *bits*. Het is vrij eenvoudig om een binair getal naar een decimaal om te rekenen. Bijvoorbeeld het binaire getal 101010 komt overeen met het decimale getal 42. Voor wie het even vergeten is, deze omrekening gaat als volgt:

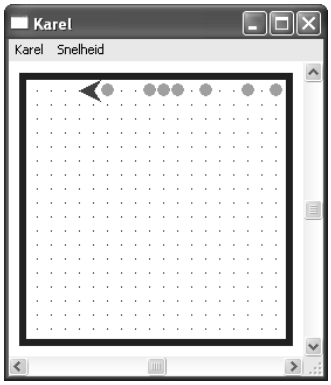
$$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 1 \times 32 + 1 \times 8 + 1 \times 2 = 42.$$

Van decimaal naar binair lijkt misschien wat lastiger. Stel je hebt een decimaal getal d. In de binaire representatie van d wordt het i^e bit gegeven door de formule

$$b_i = (d \text{ DIV } 2^i) \text{ MOD } 2$$

Hierbij moet nog opgemerkt worden dat we de bits nummeren vanaf 0 (zoals meestal in de informatica). Voorbeeld: in de binaire representatie van 42 is het 4^e bit van rechts (bit 3) $(42 \text{ DIV } 2^3) \text{ MOD } 2 = (42 \text{ DIV } 8) \text{ MOD } 2 = 5 \text{ MOD } 2 = 1$.

En nu terug naar Karel. Ieder getal heeft een vaste lengte die overeenkomt met de breedte van de wereld. We beginnen steeds vanaf de rechter muur van Karel's wereld. Hier beneden zie je hoe het getal 1001110100101 is weergegeven (voor de duidelijkheid heeft Karel een extra stapje gedaan).



Opdrachten

- Definieer een procedure **`void`** `schrijfGetal` (**`int`** `getal`) die Karel het decimale `getal` binair door middel van ballen laat weergeven vanaf de positie waar Karel zich op dat moment bevindt in de

richting waarin z'n neus wijst.

De waarde van `getal` kun je veranderen via `Vraag gebruiker om getal` in het `Karel` menu.

- Implementeer de procedures `getal1` en `getal2` die de getallen om mee te rekenen op de afgesproken plaats neerleggen.

- Definieer een procedure **`void`** `telOp ()`, die Karel twee binaire getallen laat optellen en wel op de volgende wijze: de getallen staan onder elkaar en worden bit voor bit opgeteld net zoals je vroeger geleerd hebt om decimale getallen decimaal voor decimaal op te tellen. De regels hier zijn precies hetzelfde: als je tijdens het sommeren twee enen bij elkaar optelt is het resultaat op die positie 0 en moet je 1 'onthouden', dat wil zeggen, optellen bij het resultaat van de som van de volgende twee bits. De 1 die je moet onthouden en bij de cijfers een naar links weer moet meetellen het in goed nederlands informatica jargon *carry*. Dus bijvoorbeeld:

$$\begin{array}{rr} & 0001110100101 \\ + & 1001110101101 \\ \hline & 1011101010010 \end{array}$$

  Bits (ballen) die niet meer passen worden weggegooid. Indien we bij het optellen een bal moeten weggooien hebben we *overflow*.

- Definieer op soortgelijke wijze een procedure **`void`** `trekAf ()`.

  Als het lenen van de linkerbuur stopt als je bij de muur komt vindt de meest gebruikte representatie voor negatieve getallen. Aan het meest linkse bit zie je dan of de bitrij als een negatief of positief getal gelezen wordt.

- Definieer ook de procedure **`void`** `vermenigvuldig ()` die twee binaire getallen met elkaar vermenigvuldigt. Ook hierbij dien je de 'basisschoolmethode' te gebruiken, die voor alle duidelijkheid nog eens wordt geïllustreerd aan de hand van het volgende voorbeeld. Het idee is om, bijvoorbeeld, 1011×100101 te schrijven als $1 \times 100101 + 10 \times 100101 + 1000 \times 100101$ en de tussenresultaten vervolgens op te tellen.

$$\begin{array}{rr} & 100101 \\ \times & 1011 \\ \hline & 100101 \\ & 1001010 \\ + & 100101000 \\ \hline & 110010111 \end{array}$$

  Voor het optellen kun je gebruik maken van de eerder gedefinieerde `telOp` procedure, maar je kunt ook een procedure maken die n getallen in een keer optelt. Overtuig je allereerst zelf van de overeenkomst tussen deze wijze van binair vermenigvuldigen met de normale decimale vermenigvuldiging zoals je die gewend bent.

  Gebruik dus *niet* herhaald optellen voor de implementatie van de vermenigvuldiging. Bij het optellen van N en M volgens de herhaald optellen methode moeten we N keer een optelling doen. Volgens het hier beschreven algoritme hebben we slechts $^2\log N$ optellingen nodig.

- Definieer de procedure `gemiddelde` die het gemiddelde van twee gegeven getallen berekent.

- Definieer tot slot de procedure `grootste` die de grootste van twee gegeven getallen berekent en kopieert naar de derde rij van Karel's wereld. Dit onderdeel is optioneel, je hoeft het dus niet te maken. Neem aan dat de getallen positief zijn.

Aanwijzingen

- In deze opgave is het van groot belang dat je van tevoren nadenkt over waar getallen neergezet worden en waar Karel zich bij aanvang en aan het einde van de rekenopgave bevindt. Bij het vermenigvuldigen moet je meerdere optellingen achter elkaar uitvoeren waarbij je het tussenresultaat steeds gebruikt bij de volgende optelling. Ook hier is het van belang dat je goed vastlegt waar alles staat/moet komen te staan.

- Voor deze opgave hebben we de mogelijkheden van de robot wat uitgebreid. De benodigde files staan samen in een nieuwe zip-file. Neem alle gegeven files op in je project. In versie 2 van de robot kent Karel de volgende extra mogelijkheden:

  wisRij Met de methode `wisRij` worden alle ballen van de rij waar de robot op staat weggehaald.

  Ballen De robot kent het type `Ballen`. Dit type bevat een rijtje met ballen en lege posities.

  legBallen Met de methode **`void`** `legBallen(Ballen ballen)`, wordt een hele rij ballen in een keer gelegd. De ballen worden altijd in westelijke richting neergelegd. Als Karel's neus niet naar het westen wijst zal hij zich eerst in de juiste richting draaien. Na het neerleggen van iedere bal maakt Karel steeds een stap westwaarts, met uitzondering van de laatste bal: zodra Karel die heeft neergelegd stopt hij met als gevolg dat hij altijd bovenop de laatst neergelegde bal zal staan. Je kunt zo ballen tot aan de muur leggen.

  leesBallen Met de methode `Ballen leesBallen()` worden alle ballen vanaf de huidige positie in westelijke richting gelezen. De ballen blijven hierbij gewoon liggen. Ook hier geldt dat Karel vanaf z'n huidige positie begint te lezen en ophoudt zodra hij de laatste bal gevonden heeft. Deze positie is tevens de nieuwe positie van Karel.

  Conversie string naar ballen Je kunt ook zelf een ballenrij maken van een string. De letters `1*Oo` worden geïnterpreteerd als een bal, de letters `0 _.` representeren een lege plaats. Met de aanroep `legBallen(Ballen("* * * "))`; kun je dus de binaire representatie van 42 neerleggen.

  Grootte van de wereld Voor deze opgave kan het handig zijn om de grootte van de wereld van de Robot wat aan te passen. Tijden het testen van je programma wil je misschien een kleine wereld, en tijdens het rekenen misschien een extra hoge wereld. Je kunt dit doen door `WereldBreedte` en `WereldHoogte` in `Robot.h` aan te passen.

  Binair rekenen Het is dus *niet* de bedoeling dat je een bitrij omzet in een integer, met de integers rekent en het resultaat weer omzet naar een bitrij. Voor bitrijen langer dan 32 werkt dit overigens niet eens.

Inleveren

Vóór woensdag 4 oktober, 8:30 uur via Blackboard. De beoordeling van deze opgave telt mee voor je practicumcijfer. Deze opgave is dus niet optioneel, maar moet je inleveren. Vergeet niet om **in** je uitwerking duidelijk je naam en die van je practicumpartner te vermelden. We hebben genoeg aan het door jullie veranderde `Karel2.cpp`.