Basisinstructies

Bewegen	`vooruit(n)`	Beweeg *n* stappen vooruit
	`achteruit(n)`	Beweeg *n* stappen achteruit
	`links()`	Draai 90 graden naar links
	`rechts()`	Draai 90 graden naar rechts
	`noord(n)`	Draai richting het noorden en doe vervolgens *n* stappen naar voren.
	`zuid(n)`	Draai richting het zuiden en doe vervolgens *n* stappen naar voren.
	`oost(n)`	Draai richting het oosten en doe vervolgens *n* stappen naar voren.
	`west(n)`	Draai richting het westen en doe vervolgens *n* stappen naar voren.
Verven	`verfWit()`	Zet de kwast neer met witte verf
	`verfZwart()`	Zet de kwast neer met zwarte verf
	`stopVerven()`	Berg de kwast weer op
Grijpen	`pakOp()`	Pak het baken recht voor je op
	`zetNeer()`	Zet het baken recht voor je neer
Willekeurig	`kopOfMunt()`	Dit commando zal willekeurig de waarde 'waar' of 'onwaar' aannemen, alsof er een munt is opgegooid. Bijvoorbeeld:

```
als(kopOfMunt()){
    vooruit(1)
}
anders{
    achteruit(1)
}
```

zorgt ervoor dat de robot met 50% vooruit en 50% achteruit zal gaan.

Zien	**Links**	**Voor**	**Rechts**
	`linksIsObstakel()`	`voorIsObstakel()`	`rechtsIsObstakel()`
	`linksIsVrij()`	`voorIsVrij()`	`rechtsIsVrij()`
	`linksIsBaken()`	`voorIsBaken()`	`rechtsIsBaken()`
	`linksIsWit()`	`voorIsWit()`	`rechtsIsWit()`
	`linksIsZwart()`	`voorIsZwart()`	`rechtsIsZwart()`

Programmeerstructuren

Commentaar

```
# tekst die niet door de robot gelezen wordt
```

alle tekst in een programma dat na het hekje, '#', staat wordt niet geinterpreteerd als instructies. Pas op de volgende regel worden de instructies weer gelezen door de robot. Gebruik deze mogelijkheid om een programma desgewenst te voorzien van extra informatie.

Herhalingen

```
herhaal(n){...instructies...}
```

herhaalt de instructies tussen accolades precies n keer.

Voorbeeld:

```
# een 3x3 wit vierkantje
verfWit()
herhaal(4)
{
    vooruit(2)
    rechts()
}
stopVerven()
```

```
herhaal(){...instructies...}
```

blijft de instructies tussen accolades telkens herhalen.

Voorbeeld:

```
# blijf alsmaar rechtdoor gaan
# (zal uiteindelijk blijven botsen)
herhaal()
{
    vooruit()
}
```

```
herhaalZolang(conditie){...instructies...}
```

herhaalt de instructies tussen accolades net zo lang totdat de conditie niet meer opgaat. De conditie moet een logische expressie zijn die de waarde waar of onwaar aanneemt (bijv. `voorIsVrij()`)

Voorbeeld:

```
# blijf rechtdoor gaan, totdat je niet verder kunt
herhaalZolang(voorIsVrij())
{
    vooruit(1)
}
```

`doorbreekLus`

zorgt ervoor dat de lus waar deze instructie in staat wordt beëindigd, en er wordt verder gegaan met de eerste instructie na deze lus.

Voorbeeld:

```
# blijf rechtdoor gaan, totdat je niet verder kunt
herhaal()
{
    als(voorIsObstakel())
    {
        doorbreekLus
    }
    anders
    {
        vooruit(1)
    }
}
```

Als-structuren

`als(`*`conditie`*`){`*`...instructies...`*`}`

voert de instructies tussen accolades alleen uit als de conditie opgaat. De conditie moet logische expressie zijn die de waarde waar of onwaar aanneemt (bijv. `voorIsVrij()`)

Voorbeeld:

```
# als je links een witte stip ziet, maak hem zwart
als(linksIsWit())
{
    links()
    vooruit(1)
    verfZwart()
    stopVerven()
    achteruit(1)
    rechts()
}
```

`als(conditie){...instructies...}anders{...instructies...}`
voert de instructies tussen het eerste paar accolades alleen uit als de conditie opgaat, anders voert het alleen de instructies uit tussen het tweede paar accolades. De conditie moet logische expressie zijn die de waarde waar of onwaar aanneemt (bijv. `voorIsVrij()`)

Voorbeeld:

```
# blijf rechtdoor gaan en
# draai 180 graden wanneer je niet meer verder kunt
herhaal(){
	als(voorIsVrij())
	{
		vooruit(1)
	}
	anders
	{
		links()
		links()
	}
}
```

`als(conditie){...instructies...} anders als {...instructies...}`
is de makkelijkere notatie zonder extra accolades voor: `als(conditie){...instructies...} anders { als {...instructies...}}`. Het codeblok van `anders` wordt alleen uitgevoerd als zijn overeenkomende conditie het geval is. Deze constructie is met name nuttig wanneer er meerdere verschillende gevallen moeten worden gecontroleerd en uitgevoerd.

Voorbeeld:

```
# blijf rechtdoor gaan als dat kan, controleer anders of
# links vrij is en draai naar links, anders draai naar rechts
herhaal(){
	als(voorIsVrij())
	{
		vooruit(1)
	}
	anders als(linksIsVrij())
	{
		links()
	}
	anders
	{
		rechts()
	}
}
```

Logische expressies

De *conditie* van *als*- en *herhaalZolang*-structuren zijn zogenaamde logische expressies. Deze expressies zijn een voorwaarde die de waarden waar of onwaaraannemen, waarna naar het overeenkomende deel van de code kan worden gesprongen alvorens de uitvoering te hervatten .

Een logische expressie kan een van de waarnemingsinstructies zijn, zoals bijvoorbeeld `linksIsWit()`. Waarnemingsinstructies kunnen ook worden samengevoegd met de booleaanse operatoren niet, en, of.
Voorbeeld:

```
# als je links een witte stip ziet, maak hem zwart
als(linksIsWit())
{
    links()
    vooruit(1)
    verfZwart()
    stopVerven()
    achteruit(1)
    rechts()
}
```

Operatie	Aantal argumenten	Uitleg
`niet`	1	Keert de waarde van het argument om: *Waarheidstabel:* niet waar = onwaar niet onwaar = waar *Voorbeeld:* `niet voorIsVrij()`
`en`	2	Alleen waar als beide argumenten waar zijn. *Waarheidstabel:* waar en waar = waar waar en onwaar = onwaar onwaar en waar = onwaar onwaar en onwaar = onwaar *Voorbeeld:* `voorIsVrij() en rechtsIsWit()`
`of`	2	Waar als een van beide argumenten waar is. Waarheidstabel: waar of waar = waar waar of onwaar = waar onwaar en waar = waar onwaar en onwaar = onwaar Voorbeeld: `voorIsVrij() of rechtsIsWit()`

Einde

Zorgt ervoor dat het hele programma direct stopt met de uitvoer als deze instructie wordt bereikt.

Voorbeeld:

```
# stop na 5 stappen, of eerder als je rechts een baken ziet
herhaal(5)
{
      vooruit(1)
      als(rechtsIsBaken())
      {
            einde # breek het programma af
      }
}
# normaal einde van het programma
```