een switch is eigenlijk een anders geschreven if-statement

```
switch($test) {
case 1: echo 'a'; break;
case 2: echo 'b'; break;
}
```

is dus gelijk aan:

```
if($test == 1) echo 'a';
if($test == 2) echo 'b';
```

als je dus

```
switch ($test) {
case $test < 10: echo 'a'; break;
case $test > 10; echo 'b'; break;
}
```

doet is dit dus herschreven

```
if($test == $test < 10) echo 'a';
if($test == $test > 10) echo 'b';
```

in het script staat dit:

```
switch($bestelling)
{
 case($bestelling>$drempel_3):
  $stukprijs=$prijs_4;
 break;
 case($bestelling>$drempel_2):
  $stukprijs=$prijs_3;
 break;
 case($bestelling>$drempel_1):
  $stukprijs=$prijs_2;
 break;
 case($bestelling<$drempel_1):
  $stukprijs=$prijs_1;
 break;
}
```

waardoor alles dus vertaald wordt naar if($bestelling == $bestelling > $drempel_3) etc.

als je naar de precendence van operators kijkt:
http://nl.php.net/manual/en/language.operators.precedence.php zie je dat < boven == staat. dus eerst wordt $bestelling > $drempel_3 gedaan. resultaat is een boolean. daardoor moet $bestelling ook een boolean worden (je kan geen appels met peren vergelijken)

als $bestelling 0 is krijg je bij de eerste case 0 > 250 dit heeft als resultaat een boolean false.
daarom maakt hij van je variabele in de switch case ook een boolean. $bestelling is 0 wat dus false is deze zijn gelijk dus matched deze met de eerste statement

als $bestelling niet 0 is (bijvoorbeeld 260) dan geeft 260 > 250 true. je zou dan misschien verwachten dat: 26 == 260 > 250 false oplevert (260 > 250 is false dus 0 en 260 == 0 is ook false) maar omdat false een boolean is maakt hij van de variabele in de switch case ook een boolean. 260 wordt dan true en deze zijn gelijk dus matchen ze.

hierdoor gaat deze statement dus alleen fout als $bestelling = 0 want dat is de enige waarde die als boolean false oplevert

jouw oplossing hier: http://www.informaticavo.nl/php/bloembollen2.txt lost dit dus op door 0 als eerste case toe te voegen. en is de vertaling naar een if statement if($bestelling == 0) waardoor je dus twee integers met elkaar vergelijkt.

de oplossing met switch(true) werkt omdat je altijd vergelijkingen doet in de vorm if(true == $bestelling > $drempel_3) wat dus nooit waar kan worden omdat 0 > $drempel_3 false is maar 0 < $drempel_1 true is.

maar een vrij interessante case want hier zie je dat php in de achtergrond toch nog steeds met booleans en integers e.d. werkt terwijl je ze niet ziet. je dus ziet meteen hoeveel de php parser in de achtergrond voor je oplost in termen van type casting.

hoop dat het een beetje duidelijk is deze uitleg en niet te wiskundig. normaal hoef ik aan niemand uit te leggen waarom iets werkt, alleen dat iets werkt ;-)

Groeten,

Johnny van de Laar