P1
werkcollege 2

algoritmen en programma's
leer Karel lopen
leer Karel rekenen

opgave 1 vierkantje

- wat stapjes vooruit
- vier keer een zijkant
- maak een herbruikbare procedure om *n* ballen te leggen

n ballen leggen

```
void ballen ( int aantal )
{
  for ( int t = 1 ; t <= aantal ; t = t + 1 )
  {
      legBal ( );
      stap ( ) ;
  }
}
```


opgave 1 vierkantje in C++

```
void vierkantje ( )
{
  stappen ( 8 );
  for ( int i=0; i<4; i+=1 )
  {
      ballen ( 6 );
      rechtsom ( );
  }
}
```

ruitje

- zelfde idee als vierkantje, maar randen diagonaal

```
void ballenDiagonaal ( int stappen )
{
  for ( int t = 1 ; t <= stappen ; t = t + 1 )
  {
      legBal ( );
      stap ( ) ;
      linksom ( );
      stap ( );
      rechtsom ( );
  }
}
```

5

ruitje 2

```
void ruitje ( )
{
  stappen ( 9 );
  for ( int i=0; i<4; i+=1 )
  {
      ballenDiagonaal ( 6 );
      rechtsom ( );
  }
}
```

6

akkertje zaaien

- meer van het zelfde
- 16 keer een voor van 20 ballen

```
for ( int v = 0; v<16; v = v+ 1 )
  zaaiVoor ( 20 );

void zaaiVoor ( int n )
  {
     ballen ( n ); keerLinksOm ( );
     stappen ( n ); keerRechtsOm ( );
  }
```

7

naar beginpositie

- draai naar noord
- loop tot muur
- draai linksom
- loop tot muur
- links om
- doe 13 stappen
- draai linksom

8

hulpfuncties

```
void naarNoord ( )
{
   while ( ! noord ( ) )
        linksom ( );
}

void loopTotMuur ( )
{
   while ( ! muurVoor ( ) )
        stap ( );
}
```

9

naar beginpositie in C++

```
void naarBegin ( )
{
   naarNoord ( );
   loopTotMuur ( );
   linksom ( );
   loopTotMuur ( );
   linksom ( );
   stappen ( 13 );
   linksom ( );
}
```

10

Spiraal

- hier moeten we voor het eerst even nadenken
- er zijn minstens 2 algoritmen:

1. leg reeks ballen tot aan muur en ga rechts zodra dat kan
2. bereken hoe lang de takken moeten worden

- wat kiezen we?

11

spiraal 1

```
void spiraal ( )
{
   stappen ( 4 );
   ballen ( 3 );

   while ( ! muurVoor ( ) || rechtsvrij ( ) )
   {
        if ( rechtsvrij ( ) )
              rechtsom ( );
        legBal ( );
        stap ( );
   }
}
```

12

rechts vrij ?

```
bool rechtsvrij ( )
{
   rechtsom ( );
   stap ( );
   if (opBal ( ))
   {   gaTerug ( );
       return false;
   }
   else
   {   gaTerug ( );
       return true;
   }
}
```

```
void gaTerug ( )
{
   rechtsom ( );
   rechtsom ( );
   stap ( );
   rechtsom ( );
}
```

13

spiraal 2

```
void spiraal2 ( )
{
   stappen ( 4 );
   for ( int i=0; ! muurVoor ( ) ; i=i+1 )
   {
       ballen ( i );
       rechtsom ( );
       ballen ( i );
       rechtsom ( );
   }
}
```

beter nog een for-loopje

14

pas op voor de muur

```
void ballenVeilig ( int aantal )
{
   for ( int t = 1 ; t <= aantal && ! opBal ( )
       ; t = t + 1)
   {
       legBal ( );
       if ( ! muurVoor ( ))
             stap ( ) ;
   }
}
```

15

spiraal 2 poging 2

```
void spiraal2 ( )
{
   stappen ( 4 );
   for ( int i=0; ! muurVoor ( ) ; i=i+1 )
   {
       for ( int z = 0; z<2; z=z+1 )
       {
             ballenVeilig ( i );
             rechtsom ( );
       }
   }
}
```

16

algemeen

- geef herkenbare onderdelen een naam (*procedure*)
- hergebruik deze procedures waar mogelijk
- liever herhaling met een loopje i.p.v. copy-paste van de code

17

doolhof

- bekendste algoritme

1. loop rechtdoor tot een muur
2. leg hand op muur en loop tot de uitgang gevonden is

- werkt niet bij een 'eiland'

18

algemeen

1. begrijp de opdracht/het probleem
2. verzin algoritme
   - de belangrijkste stap
   - bekijk voorbeeldjes, zoek algemeen patroon
3. bedenk of het echt werkt
4. code kloppen
5. testen + review van code
   - werkt het; moet het echt zo

19

nieuwe opgave: leer Karel rekenen

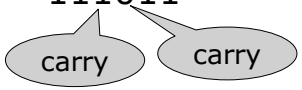
- van de kleuterschool:
  - 1234 = 1*1000+2*100+3*10+4*1
  - = $1*10^3+2*10^2+3*10^1+4*10^0$
- 10 heet hier het grond tal, zelfde truc kan ook met ander grondtal
- populaire grondtallen in informatica:
  - 2 binair, cijfers 0,1
  - 8 octaal, cijfers 0..7
  - 16 hexadecimaal, cijfers 0..9,A..F

20

karel rekent binair

- 0 = geen bal,
  1 = wel bal
- verder alle regels als bij gewoon rekenen
- optellen:

```
  101110
  001101 +
  111011
```

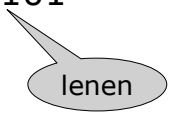


21

aftrekken

- indien nodig lenen bij linker buur
- aftrekken:

```
  1101011
  0110110 -
  0110101
```



22

negatieve getallen

- aftrekken:

```
  0000000
  0000111 -
  1111001
```

- lenen uit de linkermuur
- *two's complement* notatie voor negatieve getallen
- alle bitjes flippen en er 1 bij tellen

23

vermenigvuldigen

- herhaald optellen
  - te veel werk voor grote getallen
- slimmer vermenigvuldigen:

```
      111
    10101 *
      111
    11100
  1110000 +
 10010011
```

24

Karel's wereld heeft vaste breedte

- ballen die niet meer passen worden weggelaten (*overflow*)
- bij aftrekken kun je altijd lenen bij de muur
- normale getallen in C++ gebruiken 32 bits en hebben dezelfde regels bij overflow

25

invoer

- er is een globale variabele getal
- via menu en dialoog kun je de waarde zetten
- leg getallen op vaste plek in de wereld
  - bovenste twee regels

26