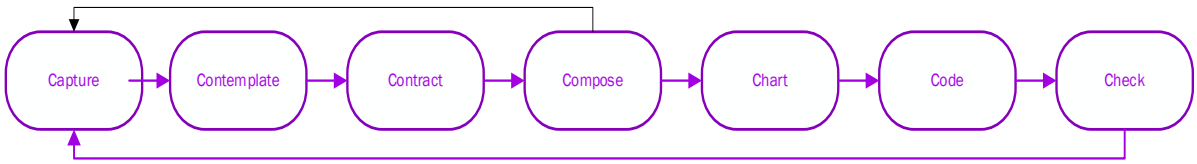
Imperatief programmeren

= recept maken = stappenplan volgen => 7 C's

Programmeren is als het schrijven van een recept.
Een recept beschrijft de wijze waarop ingrediënten getransformeerd worden tot een gerecht.
Een programma beschrijft de wijze waarop inputdata verwerkt wordt tot outputdata.
Een recept kan gebruikmaken van hulprecepten die een deel van het gerecht maken.

Een recept maken in 7 stappen, Conquer the 7 C's of programming:

1. Capture. Beschrijf de **requirements**, geef een eenduidige definitie van het probleem.
2. Contemplate. **Analyseer** het probleem:
   a. welke data is nodig,
   b. welke oplossingsstrategieën zijn er,
   c. welke deelproblemen zijn te onderkennen?
3. Contract. Bepaal de **interface**, beschrijf input- en outputdata (inhoud en structuur).
4. Compose. Deel het probleem eventueel op in **deelproblemen**: bepaal welke (bestaande of nieuwe) hulprecepten gebruikt kunnen worden.[1] *Herhaal* de 7 stappen voor elk nieuw hulprecept.[2]
5. Chart. Gebruik de 3 **besturingsstructuren** (iteratie, selectie, sequentie) om de volgorde van de instructies te beschrijven.
6. Code. Codeer de besturing in een **programmeertaal**.
7. Check. **Test** de code conform de requirements. Ga indien nodig *terug* naar stap 1.



[1] Classificeer een hulprecept als bestaand of nieuw, bestaande hulprecepten zijn (naar verwachting) al beschikbaar in de programmeertaal.

[2] Depth-first aanpak: doorloop eerst voor elk nieuw deelprobleem de 7 stappen alvorens het hoofdprobleem verder uit te werken.
Breadth-first aanpak: doorloop voor elk nieuw deelprobleem eerst de stappen 1 t/m 3, zodat de interface gebruikt kan worden, ga vervolgens verder met het hoofdprobleem. Werk na het hoofdprobleem de deelproblemen verder uit.
Wat zijn de voor- en nadelen van depth- vs breadth-first?

Voorbeeld probleem: FizzBuzz

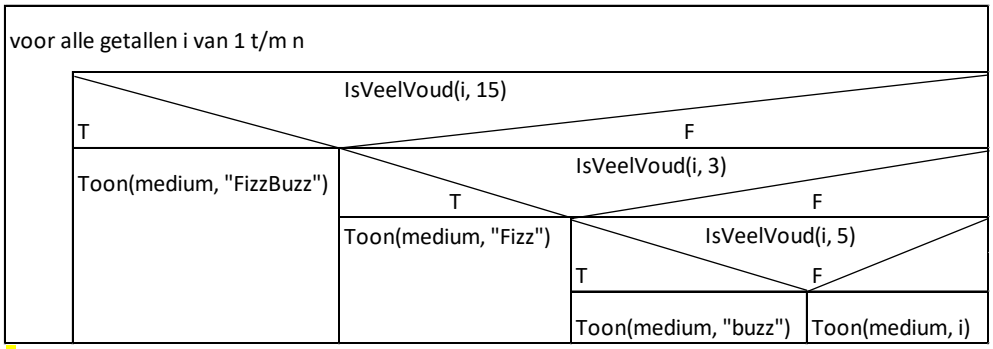
1 Toon alle getallen van 1 t/m n, maar toon voor veelvouden van 3 het woord *Fizz* en voor veelvouden van 5 het woord *Buzz* (en voor veelvouden van 15 *FizzBuzz*).

2 - inputdata: *n* en het *medium* (bijv. beeldscherm of file) voor de output
- outputdata: *medium*
- oplossingsstrategie: doorloop alle getallen van 1 t/m n, bepaal per getal of het een veelvoud is van 3, van 5, van beide of van geen en toon de bijbehorende waarde.
- deelproblemen: “bepaal of iets een veelvoud is van” en “Toon waarde op medium”

3 FizzBuzz(*in out* medium, *in* n), waarvoor verder geldt:
Op *medium* kan iets getoond worden
n is een geheel getal >= 1

4 IsVeelVoud(*in* a, *in* b): *boolean.*
Toon(*in out* medium, *in* waarde).
Beide zijn naar verwachting bestaande hulprecepten.

5 *Chart in de vorm van een Nassi-Shneiderman diagram (NSD)*



[3]

6 *Programmeertaal Python*

```
def FizzBuzz(medium, n):
  for num in range(1, n+1):
    if num % 15 == 0: print("FizzBuzz", file=medium)
    elif num % 3 == 0: print("Fizz", file=medium)
    elif num % 5 == 0: print("Buzz", file=medium)
    else: print(num, file=medium)
```

7 import sys

```
FizzBuzz(sys.stdout, 20)                 //output correct op beeldscherm
FizzBuzz(open('FizzBuzz.txt', 'w'), 15)  //output correct naar bestand FizzBuzz.txt
FizzBuzz(sys.stdout, 1)                  //output correct op beeldscherm
FizzBuzz(sys.stdout, 3.5)                //error[4]
```

[3] *Kan het ook met minder selectiestatements?* Verklaar!
[4] *Is deze fout acceptabel of niet?* Design by Contract (DbC) vs Defensive Programming!