**50**

HBO ICT

PropedeusE - TI

*Practicum*handleiding

Object Oriented Programming in C++

**TICT-V1OOPC-15**

Studiejaar 2018-2019

|  |  |
| --- | --- |
|  |  |
| Cursuseigenaar | Wouter van Ooijen |
| **Auteur(s)** | Wouter van Ooijen |
| **Datum** | http://enyze.com/wp-content/uploads/2015/12/due1.jpg2018-04-24 |
| Versie | 1.4 |
|  | |

© Institute for ICT, Hogeschool Uterecht, 2018

Contents

[2. Inleiding 4](#_Toc512331151)

[3. Hardware 5](#_Toc512331152)

[4. Software 6](#_Toc512331153)

[4.1. Python 6](#_Toc512331154)

[4.2. CodeLite (IDE) 6](#_Toc512331155)

[4.3. GIT (versie beheer tool) 7](#_Toc512331156)

[4.4. MinGW (GCC voor windows) 7](#_Toc512331157)

[4.5. GCC voor ARM eabi (Arduino Due C++ compiler) 8](#_Toc512331158)

[4.6. Doxygen (documentatie formatter) 8](#_Toc512331159)

[4.7. Gemeenschappelijke code binnenhalen met GIT 8](#_Toc512331160)

[4.8. Catch2 (test harness) 9](#_Toc512331161)

[4.9. Hwlib (C++ hardware library) 9](#_Toc512331162)

[4.10. Voorbeeld code bij deze cursus 9](#_Toc512331163)

[4.11. Templates voor practicum 9](#_Toc512331164)

[4.12. Bmptk (compilation scripts en tools) 9](#_Toc512331165)

[4.13. Arduino Due device driver 10](#_Toc512331166)

[4.14. Configureren 10](#_Toc512331167)

[4.15. Gebruik van CodeLite met bmptk 11](#_Toc512331168)

[5. Practicumopgaven week 1 13](#_Toc512331169)

[5.1. Installeer 13](#_Toc512331170)

[5.2. Boeken 13](#_Toc512331171)

[5.3. Main 15](#_Toc512331172)

[5.4. Meubels 15](#_Toc512331173)

[5.5. Gevulde rechthoek 15](#_Toc512331174)

[5.6. Samengesteld object 15](#_Toc512331175)

[6. Practicumopgaven week 2 17](#_Toc512331176)

[6.1. Katten 17](#_Toc512331177)

[6.2. Doxygen 18](#_Toc512331178)

[6.3. Catch2 18](#_Toc512331179)

[6.4. ADT: vector 19](#_Toc512331180)

[7. Practicumopgaven week 3 20](#_Toc512331181)

[7.1. Auto 20](#_Toc512331182)

[7.2. Oefening met pointers en overerving 21](#_Toc512331183)

[7.3. Dikkere muren 22](#_Toc512331184)

[7.4. Stuiteren 22](#_Toc512331185)

[7.5. Verdwijnen 23](#_Toc512331186)

[8. Practicumopgaven week 4 24](#_Toc512331187)

[8.1. Vragen 24](#_Toc512331188)

[8.2. Installeer 24](#_Toc512331189)

[8.3. Patroon 24](#_Toc512331190)

[8.4. Input 24](#_Toc512331191)

[9. Practicumopgaven week 5 26](#_Toc512331192)

[9.1. Vragen 26](#_Toc512331193)

[9.2. HC595 26](#_Toc512331194)

[9.3. Decorator 27](#_Toc512331195)

[9.4. Links-rechts Knipperen met kitt 27](#_Toc512331196)

[9.5. Idem met blink() 27](#_Toc512331197)

[10. Practicumopgaven week 6 28](#_Toc512331198)

[10.1. OLED 28](#_Toc512331199)

[11. Appendix : Enkele veel voorkomende problemen 29](#_Toc512331200)

[12. Appendix: Checklist bij Installatie 30](#_Toc512331201)

[12.1. “Some of the Compilers No Longer Exist” 30](#_Toc512331202)

[12.2. Locaties 30](#_Toc512331203)

[12.3. 2. PATH 30](#_Toc512331204)

[12.4. 3. Update ook je python-path in een MakeFile 30](#_Toc512331205)

[12.5. Unplug – Replug Due 30](#_Toc512331206)

[12.6. (Re-) run de batchfile in je workspace folder 30](#_Toc512331207)

[12.7. Herstart CodeLite of console na aanpassing van environment variabelen 30](#_Toc512331208)

# Inleiding

Dit document beschrijft het practicum bij de cursus V1OOPC:

* de gebruikte hardware;
* de gebruikte software en de procedures voor het installeren daarvan;
* de opgaves.

De ga de gang van zaken rond het inleveren van de practica is beschreven in een apart document dat je kan vinde op canvas.

De herkansingsmogelijkheid voor de practica is in het blok volgend op het blok waarin de cursus gegeven wordt. Hiervoor hoeven alleen die practica die niet als voldoende waren beoordeeld te worden getoond.

# Hardware



In deze cursus wordt gebruikt gemaakt van de volgende hardware, die door de student zelf moet worden aangeschaft:

* Een Windows PC met een vrije USB poort (Linux is toegestaan, maar vereist zelfredzaamheid van de student)
* Een Arduino Due met USB kabel
* Een solderless breadboard (‘prikbordje’)
* Een 128x64 I2C OLED display

Aanvullend materiaal is (kostenloos) beschikbaar via het Techlab:

* Verbindingsdraadjes (male-male)
* Chips, bv. 74HC595
* Potentiometer
* LEDjes, weerstanden, drukknopjes

Een deel van deze materialen is al aangeschaft of ter beschikking gesteld voor eerdere cursussen.

Met uitzondering van de PC is deze hardware pas in week 4 nodig. Een goedkope (maar niet heel snelle!) leverancier voor deze zaken is [www.aliexpress.com](http://www.aliexpress.com). Nederlandse alternatieven (sneller maar duurder) zijn [www.antratek.nl](http://www.antratek.nl) en <http://www.miniinthebox.com/nl/>.

|  |  |
| --- | --- |
| Arduino Due: een veel krachtigere broer van de bekendere Arduino Uno. Je hebt er een (micro) USB kabel bij nodig, die wordt er meestal bij geleverd. Prijsindicatie: E 15 (aliexpress) .. E 35 (antratek). |  |
| Een 128x64 I2C OLED display. We gebruiken de versie met (alleen) I2C aansluiting. Ze zijn er ook met SPI en dubbele (SPI en I2C) aansluiting. Ze zijn er in 1 kleur (voorkeur) en in meerdere kleuren (bovenste helft ene kleur, onderste helft andere kleur). Koop geen LCD zonder printje of printje zonder LCD. Prijsindicatie: E 2.50 (aliexpress) .. E 7.00 (miniinthebox). |  |

# Software



In deze cursus wordt gebuik gemaakt van de volgende, vrij verkrijgbare software:

* Python
* CodeLite
* Git
* MinGW
* GCC voor ARM
* Doxygen
* Catch2
* Hwlib
* Bmptk
* Arduino Due device driver
* C++ code voorbeelden voor deze cursus

In de volgende secties wordt aangegeven waar deze software voor dient en hoe je het moet installeren. De links naar de betreffende internet pagina’s staan ook op sharepoint. Python, GIT, MinGW en CodeLite heb je mogelijk al geinstalleerd en gebruikt in vorige cursussen. Je hoeft niet meteen alles te installeren (bij de practica staat aangegeven welke zaken je moet installeren), maar het is wellicht handiger om alles in 1 keer te doen.

## Python

Python wordt gebruikt om Python scripts (applicaties) uit te voeren. Waarschijnlijk heb je al een Python versie geinstalleerd. Zo niet, installeer dan een recente Python van

<https://www.python.org>

Bmptk (dat moet je verderop installeren) neemt aan dat er een Python interpreter in je PATH staat, of anders in C:/Python36. Als je niet aan één van beide voorweaarden voldoet dan moet je iets aanpassen (zie verderop bij bmptk).

## CodeLite (IDE)

CodeLite is een editor/IDE die je kan vinden op:

<https://codelite.org>

CodeLite heeft de irritante eigenschap bij het opstarten te checken of er een nieuwe versie is, en dan is die er niet of het lukt het niet om die te installeren. Dit kan je uitzetten via Settings 🡺 Preferences 🡺 Misc. 🡺 Check for new version at startup.

## GIT (versie beheer tool)

Git is een versiebeheervtopol. Als je software aan het ontwikkelen bent is het vaak nodig om versies van deze software bij te houden. Als er problemen zijn met de software wil je altijd kunnen kijken wanneer deze fout in de software is gekomen om zo snel en efficiënt de fout te kunnen herstellen. Een ander voordeel van versiebeheer is als je samen wilt werken bij het ontwikkelen van de software. In onze opleiding zal je hier veel mee gaan werken.

De website met alle informatie over GIT is <https://git-scm.com/>.

Voor je begint met werken met GIT is het nodig dat je een tool installeert waarmee je dit kan doen. De beste optie hiervoor is om het standaard command-line programma hiervoor te gebruiken. Zeker als je begint met werken met GIT is het verstandig om juist deze tool te gebruiken. Grafische programma’s voor het gebruik met GIT zijn er wel, maar geven vaak niet alle controle die je wilt hebben.

Om de standaard tool te installeren ga je naar de website van GIT:

<https://git-scm.com/downloads>

Kies daarbij het platform (Windows, Mac, Linux, …) en download de passende versie. Installeer deze client op jouw computer, met de default opties.

Binnen de cursus gebruiken we GIT voor het binnenhalen van software (bmptk, voorbeeld code, hwlib) en voor het inleveren van de huiswerkopgaven. Voor het binnenhalen heb je geen GIT account nodig, voor het inleveren van de huiswerkopgaven wel.

Om een account aan te maken ga je naar <https://github.com> en maak een account aan. Ga voor het ‘free’ account/plan. Zorg voordat je doorgaat dat je je email adres heb gevalideerd. TIP: gebruik je HU email adres zodat je later makkelijk een ‘Educational Licence’ kan krijgen. Nadat je het account hebt aangemaakt kan je naar <https://education.github.com/> gaan en daar een discount aan te vragen.

Er zijn op internet veel resources te vinden om te leren wat GIT is en hoe het werkt. Voor deze cursus hoef je niet veel te weten, alleen de meeste basiskennis. Je hoeft niet samen te werken, je hoeft geen zogenaamde ‘branches’ te maken. Het enige is dat je moet zorgen dat je lokale repository synchroneert met GITHUB en dat je updates van de basis repository kan doorvoeren in je lokale repository.

Om een idee te krijgen wat GITHUB is is er een uitgebreide online cursus:

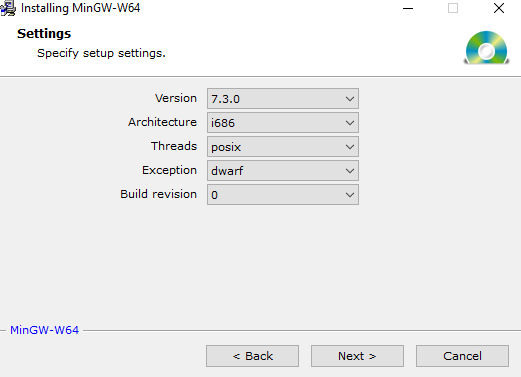
<https://www.codecademy.com/learn/learn-git>

Voor deze cursus staat bij de installatie instructies en practicum-opgaven wat je moet doen met GIT, dus het is niet strict noodzakenlijk dat je de online cursus doet.

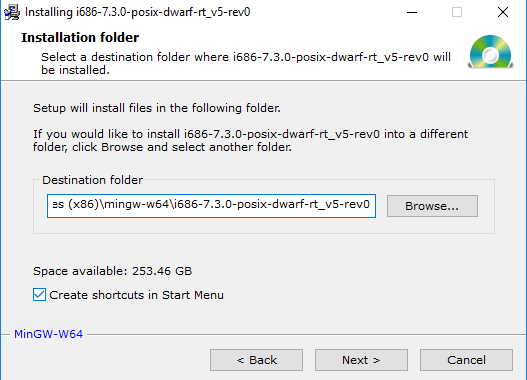
## MinGW (GCC voor windows)

MinGW is een GCC compiler voor Windows. Je kan MinGW installeren via de link op:

<https://mingw-w64.org/doku.php/download/mingw-builds> ; klik op die magina op de Sourceforge link. Dit download de installer. Run die. Selecteer de i686 (32 bits) versie.



Installeer in de default plek (onder C:/Program Files (x86)/…)



## GCC voor ARM eabi (Arduino Due C++ compiler)

Dit is de gcc (C++ compiler) die we gebruiken voor de Arduino Due. Je kan dit pakket installeren via de link op

<http://gnutoolchains.com/arm-eabi/>

## Doxygen (documentatie formatter)

Doxygen is een tool die documentatie genereert uit je source files. Installeer doxygen van

<http://www.stack.nl/~dimitri/doxygen/download.html>

## Gemeenschappelijke code binnenhalen met GIT

Er wordt gebruik gemaakt van een aantal pakketten die je met GIT download van github, en ergens op je PC moet plaatsen. De default locatie voor deze cursus (en volgende curssen die dezelfde pakketten gebruiken) is in de directory C:\ti-software. Als je deze pakketten ergens anders plaatst dan zal je een aantal instellingen moeten aanpassen.

Je haalt zo’n pakket binnen door die directory waar je het wilt plaatsen (indien nodig) aan te maken, daar (dus in die directory) een command window (DOS prompt) op te starten, en dan het commando

git clone <path>

uit te voeren. Hiermee maak je een kopie (in een subdirectory) van de <path> repository op github. (Hiervoor moet je wel eerst GIT hebben geinstalleerd. Als later de github repository is aangepast (er is bv. een fout verbeterd) dan kan je in de betreffende directory met het commando

git pull

jouw kopie up-to-date maken.

## Catch2 (test harness)

Catch is een unit test harness (software die het makkelijker maakt unit tests te schrijven). Catch haal je op van github, het git path is

<https://github.com/catchorg/Catch2>

## Hwlib (C++ hardware library)

Dit is een C++ library voor de Arduino Due die voor deze cursus en voor IPASS gebruikt wordt. Hwlib haal je op van github, het git adres is

<https://github.com/wovo/hwlib>

## Voorbeeld code bij deze cursus

Deze download bevat de (volledige) sources van alle in de reader getoonde voorbeelden. Je kan het ophalen op de zelfde wijze als de andere git pakketten, maar aangezien deze code specifiek is voor de cursus is het niet logisch om het in C:\ti-software directory op te staan, wellicht is het handiger dat je een directory specifiek voor deze cursus maakt. Het git path is

<https://github.com/wovo/v1oopc-examples>

## Templates voor practicum

Deze download geeft een opzet voor CodeLite/bmptk projecten. Het git path is

<https://github.com/wovo/v1oopc-practica>

## Bmptk (compilation scripts en tools)

Bmptk is een verzameling tooltjes die de commando’s bevatten voor het compileren, linken en opstarten (voor een micro-controller applicatie: downloaden) van een applicatie, voor een reeks targets. Bmptk haal je op van github, het git path is

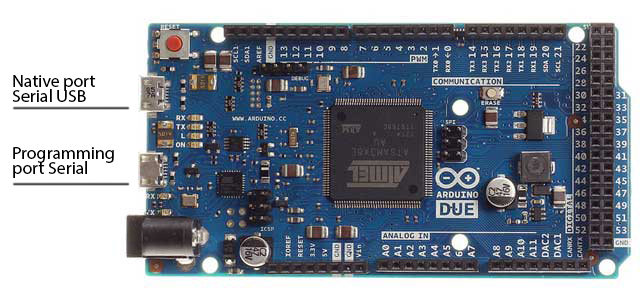
<https://github.com/wovo/bmptk>

Bmptk werkt via makescripts, die gestart worden met het commando bmptk-make. [[1]](#footnote-1) Om dit mogelijk te maken moet je zorgen dat de bmptk-make.exe file (staat in bmptk/tools) via je PATH te vinden is. Dit kan op twee manieren:

* Zorg dat de bmptk/tools directory in je PATH komt te staan. Dit is de nette manier, maar als je bmptk verplaatst moet je dus wel je PATH aanpassen.
* Copieer bmptk-make.exe en de beide .dll files naar een plek die al in je PATH staat, bv. C:/Windows. Dit is een nogal botte manier, maar je bent wel vrij om bmptk te plaatsen en te verplaatsen waar je wil.

## Arduino Due device driver

Sluit je de Arduino Due aan op je PC (gebruik op de Due de programming connector, die zit vlak naast de zwarte voedingsconnector, dus niet de ‘native’ poort).



Waarschijnlijk vindt Windows geen passende driver. Als er geen pop-up komt waarmee je de driver kan aangeven, dan kan je via de Device Manager (Control Panel 🡺 System and Security 🡺 System 🡺 Device Manager) de driver aangeven (Other Devices 🡺 Arduino Due Prog. Port; Click rechts 🡺 Update Driver Software 🡺 Browse My Computer). De driver vind je in de directory bmptk\drivers\arduino.

Windows genereert nu een Virtuele Communicatie Poort met als naam COM met het eerst volgende vrije nummer. Je moet deze poort hernoemen naar COM2. De poort vind je in de Device Manager (Aparaatbeheer)onder Ports (Poorten). Klik rechts op de poort, kies properties (eigenschappen), Port Configuration (poortinstellingen), advanced (geavanceerd), en pas de poort aan. Trek je niets aan van de klachten van Windows dat die poort al ingebruik is, tenzij je een vaste (hardware) COM2 hebt.

Koppel de Arduino Due nu even los en sluit hem weer aan.

## Configureren

Als je alles op de default plekken hebt opgeslagen en/of hebt vermeld in het PATH dan hoef je als het goed is niets aan te passen. Zo niet:

* ga naar de bmptk direcory
* kopieer de file Makefile.local naar Makefile.custom (je maakt dus een nieuwe file)
* edit die Makefile.custom:
  + Als je Python interpreter niet in het bPATH staat en ergens anders staat dan bmptk aanneemt moet je deze regel aanpassen:

# python interpreter

PYTHON ?= "C:/Python36/Python.exe"

* + Als je MinGW ergens anders staat moet je deze regel aanpassen:

# the windows toolchain

C:\Program Files (x86)\mingw-w64\i686-7.3.0-posix-dwarf-rt\_v5-rev0\mingw32

* + Als je GCC voor ARM ergens anders staat moet je deze regel aanpassen:

# the ARM toolchain

GCC-ARM ?= C:/SysGCC/arm-eabi

## Gebruik van CodeLite met bmptk

CodeLite organizeert je code in een workspace, die bestaat uit een aantal projecten, die ieder in een subdirectory staan. CodeLite kan zelf een compiler aansturen, maar in deze cursus wordt dat gedaan door bmptk make scripts.

Gebruik http:://github.com/wovo/v1oopc-practica als uitgangspunt.

In de workspace directory staat een Makefile.link file, met daarin een verwijzing naar de bmptk directory. Als je bmptk, catch of hwlib niet op de standaard plek hebt geplaatst moet je deze file aanpassen:

* + Als de ti-software directory anders hebt genoemd of ergens anders hebt geplaats pas je die locatie aan:

# the location of the ti software directory

TI-SOFTWARE := C:/ti-software

* + Als je een van de onderdelen (Catch2, bmptk, etc) anders hebt geplaats dan moet je de regel voor dat onderdeel aanpassen.

Om het aanroepen van de make scripts vanuit CodeLite te laten werken moet de workspace en de projecten daarin worden geconfigureerd. Dit gebeurt door de update\_codelite\_workspace.bat file uit te voeren. Als je een nieuwe project directory aan je workspace wil toevoegen (doe dit in de verkenner, dus niet in CodeLite) dan moet je die directory aanmaken, met de juiste Makefile er in. Het makkelijkst is een \_template\_\* directory te copiëren en de copie een passende naam te geven.

Als het goed is merkt CodeLite dat de workspace is gewijzigd, en veraagt het je om de nieuwe woerkspace in te lezen. Als dat niet lukt, sluit je CodeLite af, en dan start je CodeLite weer op.

Vanuit Codelite gebruik je de clean, build en run commando’s onder de build tab gebruiken. Die roepen de Makefile aan die in de project directory staat. Deze Makefile roept (via een paar tussenstappen) de Makefile.inc van bmptk aan, waar het echte werk gebeurt. Je kan ook met de hand de make scripts aanroepen, dit kan handig zijn als er iets fout gaat (want dan zie je de foutmelding): open een cmd window in je project directory, en geef dan een van de commando’s ‘bmptk-make clean’, ‘bmptk-make build’, ‘bmptk-make run’.

De genenereerde \_codelite.workspace file in de hoofddirectory kan je associeren met de codelite.exe, dan kan je op een workspace file dubbel-klikken om de workspace te openen.

De project Makefile bepaalt welke files onderdeel van het project zijn, of het een native (Windows) of een Arduino Due project is, en nog wat dingen. De source en header files geef je aan in de SOURCES en HEADERS variabelen. Een main.c of main.cpp hoef je niet op te geven, die wordt automatisch meegenomen.

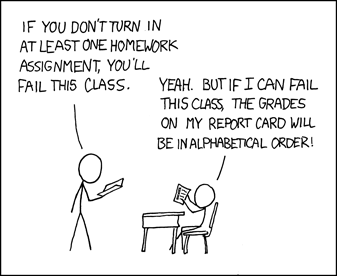
De laatste regels starten de Makefile.native die een stap hoger (in de workspace directory) staat. Voor een Arduino Due project is dit de file Makefile.due.

|  |
| --- |
| # source files in this project (main.cpp is automatically assumed)  SOURCES :=  # header files in this project  HEADERS :=  # other places to look for files for this project  SEARCH :=  # set RELATIVE to the next higher directory  # and defer to the Makefile.\* there  RELATIVE := ..  include $(RELATIVE)/Makefile.native |
| Makefile voor een project met alleen een main.cpp |

Na het bouwen kan je de applicatie ‘starten’. Voor een native (Windows) applicatie wordt dan gewoon de applicatie gestart. Voor een Arduino Due applicatie start het downloaden van de applicatie naar het Flash geheugen van de Due, en daarna wordt de Due ge-reset zodat de applicatie start. Dit gebeurt in een zwarte console window, die open blijft en eindigt met een terminal applicatie, waarin je kan zien wat de Due (over de USB/seriele verbinding) naar de PC stuurt. Dit scherm kan je afsluiten door op Esc te drukken.

|  |
| --- |
|  |
| Downloaden en runnen van een Due applicatie |

# Practicumopgaven week 1



Bestudeer de hoofstukken van de reader zoals aangeven in het weekoverzicht.

## Installeer

Installeer en configureer (voor zover je dit nog niet hebt gedaan voor vorige cursussen) Python, CodeLite, GIT, MinGW, bmptk, en de voorbeeld code.

## Boeken

Onderstaant zie je de code van de class *book*. De code is verdeelt over een .hpp bestand en een .cpp bestand. Lees de code en beantwoord de volgende vragen:

1. Wat doet de contructor van de klasse book?
2. Hoe noemen we de tekst tussen de : (dubbelepunt) en de { (krulhaak) bij de constructor?
3. Welke functies van de klasse book zijn van buitenaf toegankelijk?
4. Waarom denk je de schrijver van deze code er voor heeft gekozen sommige functies private te maken?
5. Waarom staat er in de .cpp voor elke functie “book::”?

|  |
| --- |
| #ifndef BOOK\_HPP  #define BOOK\_HPP  #include <ostream>  class book {  private:  string text;  string author;  string title;  void print\_text();  void print\_author();  void print\_title();  public:  book(  const std::string & text,  const std::string & author,  const std::string & title  ):  text( text ),  author( author ),  title( title )  {}    void print();  };  #endif // BOOK\_HPP |
| book.hpp |

|  |
| --- |
| #include <ostream>  #include "book.hpp"  void book::print(){  print\_title();  print\_author();  print\_text();  }  void book::print\_text(){  cout << text << "\n";  }  void book::print\_title(){  cout << title << "\n";  }  void book::print\_author(){  cout << author << "\n";  } |
| book.cpp |

## Main

Maak een project aan met de book.cpp en book.hpp, en een main.cpp waarin de main functie een book aanmaakt en deze print.

## Meubels

1. Maak een project aan met de klassen: *stool*, *table* en *furniture*. *Stool* heeft als member variabelen n\_legs en n\_seats. *Table* heeft als member variabelen n\_legs, length en width. De constructors krijgen de aantallen mee en slaan die op. De membervariabelen zijn allemaal private.
2. *Furniture* bestaat uit 4 variabelen van het type *stool* en 1 variabele van het type *table*. De constructor van *Furniture* initializeert de meubelen met een redelijke waarden. De membervariabelen van *Furiniture* zijn ook private.
3. *Furniture* heeft de methode *makeMoreHipster*(). Deze functie haalt van alle 4 de stoelen één poot af en geeft de tafel deze 4 poten. Schrijf deze functie en de benodigde functie’s bij *stool* en *table.*
4. Schrijf de main functie. Deze maakt een *Furniture* aan, print zijn eigenschappen, roept zijn *makeMoreHipster*() aan, en print weer de eigenschappen. Om dit te kunnen doen moet je print functies toevoegen aan alle klassen.
5. Teken een UML-diagram met *stool*, *table* en *furniture*.

## Gevulde rechthoek

|  |
| --- |
| D:\2017-2018\vakken\V1OOPC\images\h1\filled-rectangle.png |
| UML diagram van een opgevulde rechthoek (en de window waarop hij wordt afgebeeld) |

Neem de 04-09-window voorbeeldcode als uitgangspunt. Schrijf (voeg er aan toe) een klasse (hpp en cpp files) voor een opgevulde rechthoek. De constructor ziet er hetzelfde uit als van de rechthoek in de voorbeelden maar bij het afdrukken moeten alle pixels die deel uitmaken van de rechthoek (inclusief de randen) getekend worden. Je maakt dus geen gebruik van de lijn klasse, gebruik direct de methode(n) van window. Demonstreer je klasse in een applicatie die een aantal gevulde rechthoeken toont. In het UML diagram zijn de attributen van de gevulde rechthoek weggelaten: die moet je zelf bedenken.

## Samengesteld object

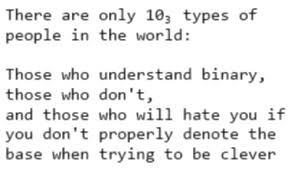
Gebruik de code van de vorige opgave als uitgangspunt. Maak een klasse voor een grafisch object (cpp en hpp files) dat (ten minste)

* 1 rechthoek (de bestaande of de in de vorige opgave gemaakte gevulde rechthoek),
* 1 lijn en
* 1 circkel bevat (dus een compositie)
* en van ten minste 1 van deze basis objecten meer dan 1.

Verdere vereisten:

* Het grafische object moet iets herkenbaars zijn, bv. een huis, trein, wolkenlucht, fabriek, pacman.
* De constructor van jouw object moet (ten minste) 1 parameter hebben die iets aan het object aanpast, bv voor een trein het aantal wagons.
* Maak ook een UML diagram met jouw klasse, de basis klassen die jouw object gebruikt en de window klasse.
* Demonstreer jouw klasse in een applicatie die er (ten minste) twee verschillende varianten van afbeeldt.

# Practicumopgaven week 2



Bestudeer de hoofstukken van de reader zoals aangeven in het weekoverzicht.

## Katten

Lees de onderstaande code met de class *cat* en de main functie die deze class gebruikt. Wat zijn de waarden van de attributen *pianocat*, *ceilingcat* en *happycat* na het uitvoeren van de code?

|  |
| --- |
| #include "cat.hpp"  int main(int argc, char \*\*argv){  cat pianocat(230,10,6, 1);  cat ceilingcat;  cat happycat = pianocat + ceilingcat;  pianocat \*= ceilingcat;  } |
| main.cpp |

|  |
| --- |
| #ifndef CAT\_HPP  #define CAT\_HPP  #include <algorithm> // std::min  #include <cstdint> // uint8\_t  class cat {  private:  int n\_lives;  // colour of the cat:  uint8\_t red;  uint8\_t green;  uint8\_t blue;  public:  cat( uint8\_t red, uint8\_t green, uint8\_t blue, int n\_lives = 9 ):  n\_lives( n\_lives ), red( red ), green( green ), blue( blue )  {}  cat():  n\_lives(9), red(7), green(4), blue(5)  {}  cat operator+( const cat& rhs ) const {  return cat(  (red + rhs.red ) / 2,  ( green + rhs.green ) / 2,  (blue + rhs.blue ) / 2,  n\_lives + rhs.n\_lives );  }  cat & operator\*=( const cat & rhs ){  red = std::min( red \* rhs.red, 255 );  green = std::min( green \* rhs.green, 255 );  blue = std::min( blue \* rhs.blue, 255 );  n\_lives \*= rhs.n\_lives;  return \*this;  }  };  #endif // CAT\_HPP |
| cat.hpp |

## Doxygen

Installeer Doxygen. Copieer de inhoud van de week-2-rational directory, run Doxygen, bekijk de documentatie die wordt aangemaakt. Er ontbreekt Doxygen documentatie voor één operator (bij het genereren geeft Doxygen een warning). Voeg die documentatie toe en genereer de documentatie opnieuw. Check in de Doxygen uitvoer dat die operator nu wel goed gedocumenteerd is. Zorg dat bij het inleveren de gegenereerde documentatie \*niet\* wordt meegnomen. (zie .gitignore)

## Catch2

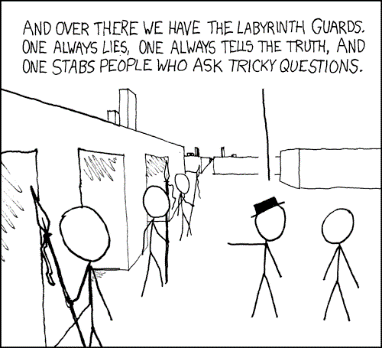
Installeer Catch. De week-2-rational directory bevat een Catch test voor een library voor rationele getallen (breuken). Zoals je kan zien als je de test runt zitten er een aantal fouten in de library. Verbeter die fouten. Implementeer daartoe eerst een operator<< voor rationele getallen, die de beide getallen (teller en noemer) print in vier posities, hexadecimaal, met voorloopnullen, en 0x ervoor.

## ADT: vector

* Schrijf een vector klasse (hpp en cpp file) met de functionaliteit die aangegeven is in de volgende tabel. Je mag daarvoor code uit de reader en de voorbeelden kopieren.
* Documenteer je klasse met Doxygen commentaar en genereer de documentatie.
* Test je implementatie met Catch2. Schrijf hiertoe voor iedere operator (ten minste) 1 test.
* Gebruik je tests op de ADT implementatie van (ten minste) 1 andere student, en zorg dat (ten minste) 1 andere student zijn implementatie test met jouw tests. Wijzig zo nodig je implementatie en/of tests naar aanleiding van de resultaten. Noteer wie die andere student was, en wat de resultaten waren.

|  |  |  |
| --- | --- | --- |
| **Feature** | **Functie parameters** | **Functie resultaat** |
| atributes | x int |  |
| y int |  |
| constructors | - |  |
| int, int |  |
| + | v | v |
| v, v | v |
| += | v &, v | v & |
| \* | v, int | v |
| int, v | v |
| \*= | v &, int | v & |
| << | std::ostream &, v | std::ostream & |
| v = vector parameter | | |

# Practicumopgaven week 3



Bestudeer de hoofstukken van de reader zoals aangeven in het weekoverzicht.

## Auto

|  |
| --- |
| class course {  public:  const std::vector< char > name;  };  class mark {  public:  course & course;  uint8\_t number;  };  class student {  public:  const std::string name;  int number;  std::vector< mark > marks;  };  class klas {  public:  const std::array< char, 3 > name;  std::vector< student > students;  klas(){}  }; |
| class definitions |

Gegeven deze definities, wat is het type van de variabelen a .. x?

|  |
| --- |
| klas v1a;  auto a = v1a;  auto b = a.name;  auto c = a.students[ 3 ];  auto d = c.name;  auto e = c.number;  auto f = c.marks[ 7 ];  auto g = f.course;  auto h = f.number;  auto i = g.name; |
| variables |

## Oefening met pointers en overerving

Geef van de onderstaande code aan of de regels met een getal in het commentaar een error opleveren of niet, en waarom.

|  |
| --- |
| class vehicle { . . . };  class car : public vehicle { . . . };  class train : public vehicle { . . . };  class intercity : public train { . . . };  void travel( vehicle & v );  void ovchipkaart( train & t );  void file( car & c );  car batmobiel;  intercity utr\_amst;  train ICE;  file( batmobiel ); // 1  ovchipkaart( utr\_amst ); // 2  file( ICE ) // 3  ovchipkaart( batmobiel ) // 4  travel( utr\_amst ) // 5  travel( batmobiel ) // 6  intercity & ref7 = utr\_amst; // 7  car & ref8 = utr\_amst; // 8  train & ref9 = utr\_amst; // 9  train & ref10 = batmobiel; // 10  **intercity** & ref11 = batmobiel; // 11  **intercity** & ref12 = ICE; // 12 |
| Overerving, parameters en references |

## Dikkere muren

|  |
| --- |
| D:\2017-2018\vakken\V1OOPC\images\week3\wall.png |
| Een muur |

Neem de 12-10-bounce code als uitgangspunt. Voeg daar een muur subklasse (cpp en hpp files) aan toe. Een muur is een rectangle, maar hij heeft als extra een vlag die aangeeft of hij gevuld of ongevuld (zoals de huidige rechthoek) moet worden afgebeeld. Pas de main aan zodat de muren 4 pixels dik en (initieel) gevuld zijn. Het is niet erg als de muren in de hoeken overlappen. Een muur wisselt iedere toggle\_interval update calls tussen gevuld en ongevuld. Dit aantal is een constructor parameter van de muur.

## Stuiteren

|  |
| --- |
| D:\2017-2018\vakken\V1OOPC\images\week3\bounce.png |
| Een betere stuiter-richting |

Neem de code van de vorige opgave als uitgangspunt. Voeg daar in de drawable een stuiter vector toe, met als default waarde (1,1). Als de bal botst met een object, dan vermenigvuldigd hij zijn speed x en y met de x en y van deze vector in drawable. De default (1,1) heeft dus geen enkel effect. De waarden (-1,1) en (1,-1) zorgen voor een realistisch stuiter effect voor een horizontale resp. een vertikale muur.

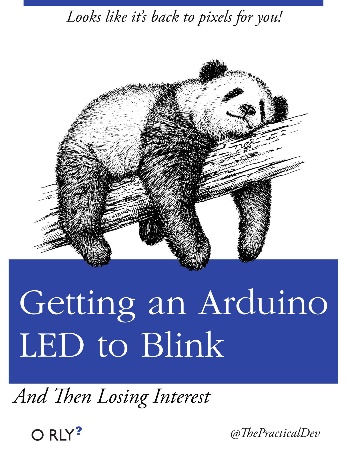
Pas de interactie van de bal aan.

Pas de constructor van de muur aan zodat een stuiter-waarde wordt meegegeven. Geef in de main de muren een passende stuiter waarde, zodat de bal realistischer tegen een muur stuitert.

## Verdwijnen

Kopieer de code van je vorige opgave naar een nieuwe directory. Voeg een nieuwe klasse slachtoffer (victim) toe. Een victim is een (ongevulde) rechthoek die zodra de bal hem heeft aangeraakt bij iedere update stapje kleiner wordt. Hiervoor moet je zowel de plek als de omvang van het slachtoffer aanpassen. Teken (mag met een pen, in een afdruk van het UML diagram met de hiërargie van drawables) deze nieuwe klasse. De bal hoeft niet tegen een slachtoffer te stuiteren (dat is best lastig omdat je de richting niet van te voren weet), hij kan er gewoon overheen gaan.

# Practicumopgaven week 4



Bestudeer de hoofstukken van de reader zoals aangeven in het weekoverzicht.

## Vragen

Beantwoord de volgende vragen:

* Wat betekent de term ‘bare metal’?
* Wat is het nut van een ‘weak pull-up’?
* Wat is het nut van een namespace, welk probleem kan je ermee vermijden?

## Installeer

Installeer GCC voor ARM, hwlib, en de Arduino Due device driver (voor zover je dat nog niet gedaan hebt). Test dat alles werkt door de knipper voorbeeldcode te runnen.

## Patroon

Sluit 4 LEDs (met weerstanden!) aan op Due pinnen. Schrijf een programma dat, gebruikmakend van hwlib::target::pin\_out, Het volgende patroon laat zien:

|  |  |
| --- | --- |
| tijd 🡺 | XX--  -XX-  --XX  -XX-  XX-- |

etc.

## Input

Sluit naast de LEDs van de vorige opdracht, nog twee drukschakelaars aan. Schrijf een programma dat zorgt dat de ene schakelaar een LED meer aan zet, en de andere een LED minder. Indrukken van de ‘meer’ schakelaar maakt dus de stappen

----

X---

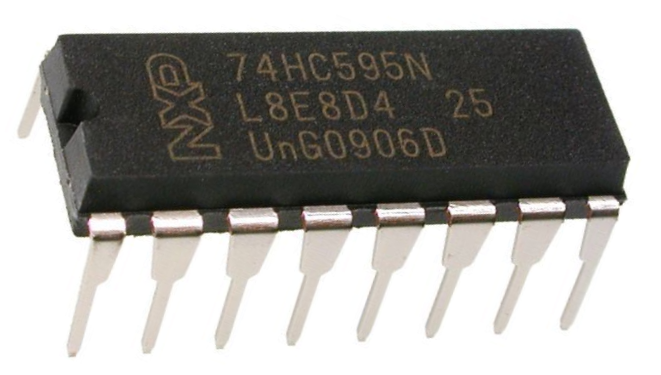
XX--

XXX-

XXXX

Bouw je programma zo op dat alleen de main() ‘weet’aan welke pinnen de LEDs en schakelaars zitten, en dat het eigenlijk werk wordt gedaan in een functie die als parameters een port\_out en twee pin\_in’s mee krijgt.

# Practicumopgaven week 5



Bestudeer de hoofstukken van de reader zoals aangeven in het weekoverzicht.

## Vragen

Beantwoord de volgende vragen:

* Hoeveel GPIO pinnen heb je nodig om 3 SPI peripherals (met zowel input als output) aan te sluiten? Maak een lijstje van de functies van deze GPIO’s.
* Wat is de functie van de Q7’ output pin van een 74HC595 chip?

## HC595

Sluit, naast de LEDs en schakelaars van de vorige opdrachten, nog een 74HC595 chip aan, met LEDs (en weerstanden!!) op 4 van zijn uitgangen. Test je schakeling door je meer/minder programma uit te breiden naar 8 LEDs (4 op de Due pinnen, 4 op de HC595).

Let op dat je alle pinnen van de HC595 chip goed gebruikt. Vul daartoe de onderstaande tabel in, dat maakt het ook makkelijker voor jezelf om de chip aan te sluiten. Inputs moet je altijd aansluiten, outputs alleen als je die gebruikt. Raadpleeg de datasheet van de chip voor de details die je niet uit de reader kan halen.

|  |  |  |
| --- | --- | --- |
| aansluiten aan |  | aansluiten aan |
|  |  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## Decorator

Schrijf een twee-input AND decorator voor de pin\_in klasse. De constructor van deze decorator heeft twee pin\_in’s als parameters. De get() van de decorator geeft 1 terug als de get()’s van beide ‘slaaf’ pin\_in’s 1 teruggeven (anders geeft hij 0 terug). Demonstreer deze decorator.

## Links-rechts Knipperen met kitt

Schrijf een pin\_out\_invert decorator. Schrijf een programma dat zorgt dat de kitt() functie op de 8 LEDs het volgende patroon laat zien:

XXXX----

----XXXX

XXXX----

----XXXX

etc.

Doe dit door de 8 pin objecten te combineren via jouw pin\_out\_invert() decorator en de bestaande all() decorator.

## Idem met blink()

Bereik het zelfde als in de vorige opgave, maar nu door blink() te gebruiken ipv. kitt().

# Practicumopgaven week 6



Bestudeer de hoofstukken van de reader zoals aangeven in het weekoverzicht.

## OLED

Sluit de OLED aan en run de voorbeelden uit de reader. Doe zelft iets met de OLED, teken bv. een huisje.

# Appendix : Enkele veel voorkomende problemen

|  |  |
| --- | --- |
| symptoom | oplossing |
| CodeLite blijft zeuren om een update | Uncheck settings🡺preferences🡺geberal🡺”check for new version at startup” |
| Declaraties in een header file worden niet herkend | Check of je na het copiëren de #ifndef etc. wel hebt aangepast aan de nieuwe file naam |
| De compiler klaagt op de eerste (echte) regel van een file | Je bent in de header die je #include de ; aan het einde van een klasse declaratie vergeten |
| error: stray '\342' in program | Heb je ge copy-pasted vanuit de reader? De quote characters die daar in staan zijn geen ASCII characters. |
| Rare fouten bij compileren | Zorg dat er geen spaties in je path staan |
| Bij het runnen flitst even een schermpje op maar het verdwijnt meteen weer | Open een DOS box (command window) en cd naar de directory van je project. Run daar het commando ‘bmptk-make run’. Je ziet nu wat het commado te melden heeft. |
| Als je de applicatie probeert te runnen gebeurt er niets | Je bent vergeten het pop-up schermpje van de vorige keer runnen te sluiten. |

# Appendix: Checklist bij Installatie

## “Some of the Compilers No Longer Exist”

Als je deze foutmelding krijgt bij het opstarten van CodeLite, en daarbij een lijst met compilers om uit te kiezen, kies dan een willekeurige compiler uit die lijst.  
Uiteindelijk wordt toch de bmptk compiler gebruikt.

## Locaties

Verifieer dat je (behalve evt Python) alles op standaard locaties hebt geinstalleerd:

* C:\SysGCC
* C:\TDM-GCC-32
* C:\ti-software\hwlib
* C:\ti-software\Catch2
* C:\ti-software\bmptk

## 2. PATH

Verifieer dat je je windows environment PATH hebt geupdate. Er moet zijn toegevoegd/aanwezig zijn:

* C:\ti-software\bmptk\tools
* Het pad van je Python installatie

## 3. Update ook je python-path in een MakeFile

Dat kan de Makefile\_link of Makefile\_shared in je workspace directory zijn.  
Voeg daaraan de volgende regel toe:  
PYTHON = C:\.....(je python path)

Wat ook kan is het juiste python-path in een Makefile in je bmptk/tools directory zetten.

## Unplug – Replug Due

Dus… als je in apparaatbeheer de COM-poort van je Due hebt aangepast naar COM2, unplug en replug dan je Due, zodat hij daadwerkelijk ook die poort gaat gebruiken (anders werkt het uploaden van binaries naar je Due niet).

## (Re-) run de batchfile in je workspace folder

Als je iets hebt veranderd aan de configuratie: andere locatie voor bmptk, hwlib, andere environment-variabelen of wat dan ook – sluit dan Codelite en je openstaande consoles af, en executeer opnieuw de batch-file in je workspace folder – zodat die de makefiles en alles goed kan instellen.

## Herstart CodeLite of console na aanpassing van environment variabelen

Als je environment variabelen (de PATH variabele in dit geval) hebt aangepast, kunnen CodeLite of je console daar pas gebruik van maken nadat je ze opnieuw gestart hebt.

1. Bmptk-make is gewoon een copie van GNU make, maar met een andere naam om te voorkomen dat een andere versie die ergens op je computer staat wordt gebruikt. [↑](#footnote-ref-1)