

Leerdoelen per week, week 5 t/m 8 – EMS30

In dit document worden alle leerdoelen die aan bod zijn gekomen in de lessen van week 5 tot en met 8 overzichtelijk weergegeven. Mocht je ergens tegenaan lopen tijdens het maken van de eindopdrachten, dan kun je dit document gebruiken om de nodige informatie te vinden.

Inhoudsopgave

Week 5	1
Les 1 – Van C naar C++	1
Les 2 – User-defined Data Type in C++	2
Week 6	2
Les 1 – Templates	2
Les 2 – Overerving en polymorfisme	3
Week 7	3
Les 1 – Datastructuren	3
Les 2 – Algoritmen, performance en big-O-notatie	3
Week 8	4
Les 1 – UML use cases en klassediagram	4
Les 2 – UML sequentie-, toestands- en activiteitendiagram	4

Week 5

Les 1 – Van C naar C++

Je hebt deze les geleerd hoe je:

- in C++ iets naar een outputdevice (bijvoorbeeld je beeldscherm) kunt sturen en iets van een inputdevice (bijvoorbeeld je toetsenbord) kunt inlezen;

- in C++ standaard classes (bijvoorbeeld `string`) kunt gebruiken.

Les 2 – User-defined Data Type in C++

Je hebt deze les geleerd hoe je:

- een UDT genaamd `Tijdsduur` kunt definiëren in de vorm van een **class**;
- de *implementatie* van de class `Tijdsduur` kunt afschermen van de gebruiker door **private** datavelden en **private** memberfuncties te definiëren;
- de *interface* van de class `Tijdsduur` beschikbaar kunt maken voor de gebruiker door **public** memberfuncties te definiëren;
- een object van de class `Tijdsduur` kunt *initialiseren* (door middel van *constructors*);
- memberfuncties kunt definiëren die ook voor *read-only* objecten van de class `Tijdsduur` gebruikt kunnen worden;
- er met behulp van *operator overloading* voor kunt zorgen dat een `Tijdsduur` op dezelfde wijze gebruikt kan worden als een ingebouwd datatype (b.v. **int**);
- de implementatie van het UDT `Tijdsduur` kunt *wijzigen* zonder dat de code die deze UDT gebruikt aangepast hoeft te worden.

Week 6

Les 1 – Templates

Je hebt deze les geleerd hoe je:

- de herbruikbaarheid en aanpasbaarheid van een functie kunt verbeteren door een functietemplate te gebruiken;
- de standaard template class `std::vector<T>` gebruikt.

Les 2 – Overerving en polymorfisme

Je hebt deze les geleerd hoe je:

- met behulp van overerving een ‘is een’-relatie tussen classes implementeert;
- een polymorfe functie kunt definiëren die je zowel voor objecten van een bepaalde class als voor objecten van de, van deze class, afgeleide classes kunt gebruiken;
- er voor kunt zorgen dat software eenvoudig uit te breiden is, zonder dat bestaande code gewijzigd hoeft te worden, door overerving en polymorfisme toe te passen.

Week 7

Les 1 – Datastructuren

Je hebt deze les geleerd hoe je:

- een aantal datastructuren uit de standaard C++ library kunt gebruiken;
- een aantal algoritmen uit de standaard C++ library kunt gebruiken.

Les 2 – Algoritmen, performance en big-O-notatie

Je hebt deze les geleerd hoe je:

- je de orde (big-O-notatie) van een algoritme kunt bepalen;
- verschillende implementaties van algoritmen die functioneel gelijk zijn, toch in orde kunnen verschillen.

Week 8

Les 1 – UML use cases en klassediagram

Je hebt deze les geleerd hoe je:

- UML kunt gebruiken om de statische structuur van software te modelleren;
- een UML klassediagram kunt maken.

Les 2 – UML sequentie-, toestands- en activiteitendiagram

Je hebt deze les geleerd hoe je:

- UML kunt gebruiken om het dynamisch gedrag van software te modelleren;
- een UML use-case-diagram kunt maken;
- een UML sequentiediagram kunt maken;
- een UML communicatiediagram kunt maken;
- een UML toestandsdiagram kunt maken;
- een UML activiteitsdiagram kunt maken.