Je moet altijd een header file maken, in de header file geef je aan welke variabelen, functies etc je hebt. In de cpp file include je de header en andere dingen en die kun je vervolgens functionaliteit kan geven zoals functies. In de main file moet je alle andere files ook includen.

Pragma once zorgt ervoor dat je header file maar 1x geimporteerd en gebruikt wordt. Header moet altijd eindigen met ;

<https://gyazo.com/49a4cbea57fe82c22b96c8d58622dc3d>   
<https://gyazo.com/e0544982e3f17f5157b85a3d78260192>

Als je curly braces maakt binnen je main dan wordt iets uitgevoerd erbinnen en daarna wordt het ook gelijk weer uitgevoerd, waardoor je 2x een destructor dus hebt. Dit gebeurd dus bij sub scopes. Alles erbinnen wordt altijd opgeruimd wanneer alles in de haakjes is uitgevoerd. Het is makkelijk om voor de main dus een aparte file te hebben omdat het altijd als 1e wordt uitgevoerd en zodat zowel jij als het programma het makkelijk kan terug vinden.

<https://gyazo.com/83e5d68229fda65d58dc1f59a6ba89b9>

De header is de definitie en de cpp is de implementatie. Alle c++ en c talen hebben ook een main class nodig. Die moet altijd 0 returnen.

Als je geen default constructor/destructor maakt dan doet de compiler het voor je, dus er is er altijd 1. Heel vaak ga je het echter zelf definieren. De constructor wordt als eerste uitgevoerd en de destructor wordt als laatste uitgevoerd.

<https://gyazo.com/8f25807b9662ddf25de33abfbaeef043>

De hello:hello etc zijn constructors. Constructors zijn speciaal. Dit zijn dingen die aangemaakt en aangeroepen wordt op het moment dat een class wordt aangemaakt. Dit is dus de initialiseer functie, vergelijkbaar met void start. Je kan constructors ook een parameter geven zodat ze een specifieke taak hebben.

<https://gyazo.com/e5049e7acb257097756ab86082edbdc6>

Er bestaat ook een destructor. Dit wordt aangeroepen wanneer een object bijvoorbeeld verwijderd wordt. Dit voorkomt memory leaks en zorgt ervoor dat je project clean blijft. De destructor heeft altijd een ~ voor de naam.

De void is een method. Dus gewoon een functie.

<https://gyazo.com/62b820130dc48f2a354a56ddd9021092>   
<https://gyazo.com/d31e5652e78360ac5f0bd1ba005c7c38>   
<https://gyazo.com/b690bb0d65d8d972c5be34cf6f635acb>   
<https://gyazo.com/4e24f102537e9aadc05e69c108b80b19>   
<https://gyazo.com/f10eb15d380566b10c9368e8b193c6af>

We hebben ook inheritence; overerfen. Het heeft kleine verschillen maar wel overlap. Alle gemeenschappelijke gedrag verwerk je in een “superclass”. Hoe ze specifiek zijn is een ander verhaal. De nieuwe superclass shape heeft dus ook een method met settransparancy. Doordat je virtual ervoor zit wordt het geimplementeerd door alle dingen die het gebruiken.

<https://gyazo.com/711636f749f72347ecc540489ab95ef2>

We hebben ook overriding. Dit zorgt ervoor dat je gedrag uit een superclass vervangt voor specifieke situaties. In de superclass zet je virtual ervoor waardoor je aangeeft dat het vervangen wordt in de subclass.

<https://gyazo.com/3881241ea58b138a61615b4365be0b87>

In C++ heb je geheugen management nodig. Een method is de stack, dit is een beperkte werk ruimte en dit wordt automatisch geallocate. Dit heet dus ook automatic allocation, dit noemt iedereen ook wel gewoon de stack. Aan het einde van de stack wordt alles dus ook weer netjes opgeruimd.

<https://gyazo.com/78da664a7c071854c3170959d8466f32>

Je hebt ook een heap, dit is de dynamic allocation. Dit zorgt ervoor dat je de maximale geheugen van je computer kunt gebruiken. Hierdoor kan je bijvoorbeeld grote aantallen van objecten maken. Bij heaps heb je dus pointers nodig. Je hebt new nodig om iets aan te maken. De ster om aan te geven dat het om een heap gaat. Het resultaat van de functie is een pointer ookwel een verwijzing naar het type student. Dit komt omdat de student op een ander stuk in het geheugen zit. Om de method aan te roepen gebruik je dan ook een pijltje, dit is omdat je een functie op een pointer aanroept. De variabele student hier etc zitten niet meer in de variabele. De variabele dat is aangemaakt is dus een pointer en een referentie, niet de variabele zelf. Je moet tot slot ook zelf je pointer opruimen om memory leaks te voorkomen. Als je programma sluit dan worden wel alle heaps opgeruimd, dit gebeurd ook bij een crash.

Je hebt ook een system class genaamd Vector (system dingen geef je aan met < >, eigen dingen met “”). Hierdoor kan je bijvoorbeeld arrays maken. Een vector is eigenlijk een list dat vergelijkbaar is met C#.

Les van 11-05-2021  
<https://gyazo.com/07f815230dccae3a68dcf5cbd6847a35>

De pointers, deference, reference en address of operator

Operator is een + of een – en een compound assignment operator is een += bijvoorbeeld  
<https://gyazo.com/9f9858fa0b06f8ff6bac7154defa3a8c>   
iets met een & is een reference dus int& is een reference naar een int  
\* is pointer naar waar iets wordt verwezen

Operator precedence laat de volgorde zien van welke operators als eerste worden uitgevoerd

Overload voorbeeld:  
<https://gyazo.com/e5a9d5600e7522c28d76074d658a0f25>

Config file voorbeelden  
<https://gyazo.com/835e8a0ac7a09c05fcc8c12fee405475>

<https://gyazo.com/0de319c39fddc2556ff101b0e91ce0aa>

Maak een vector2 met magnitude, normalize functies, alles ermee optellen, dot product opvragen. Een goeie werkende vector2 class maken. Angles tussen 2 vectoren berekenen is ook belangrijk.