2022

Autoverhuur & Gameswinkel in C++

CEYHUN ÇAKIR | 1784480

Inhoud

Inleiding

In dit verslag leg ik uit hoe ik de Autoverhuur en Gameswinkel opdracht heb gemaakt in C++. Verder ga ik uitleggen welke functies ik heb gebruikt om dit opdracht te kunnen realiseren. Ook komt er een uitleg over wat mijn gedachtegang en logica was om deze opdracht te kunnen realiseren. Er komt uiteindelijk ook een samenvatting over hoe ik deze opdracht kan verbeteren.

Deel 1 (Autoverhuur)

Wat moest ik maken?.

In het opdracht ben ik eerst begonnen met het maken van het deel van Autoverhuur. In dit deel werd er verwacht dat een auto gehuurd kon worden door een klant. Ook werd er verwacht dat klanten korting kunnen krijgen op de huurprijs van een auto. Uiteindelijk moest de klant getoond worden waar er staat welke auto er is gehuurd voor de klant.

Hoe heb ik het gemaakt?.

Ik ben begonnen met het bekijken van mijn vorige opdracht die ik had gemaakt in Java. Dit opdracht was identiek aangezien het de zelfde opdracht was. Na dat ik de opdracht had geanalyseerd wat er gedaan moest worden ben ik begonnen met het creëren van (.h) en (.cpp) bestanden. Deze bestanden zijn noodzakelijk om in C++ bestanden te compileren.

Ik begon met het declareren van de member functies die nodig waren binnen de (.h) bestanden. Dit was nodig aangezien er declaratie nodig is om de functionaliteit van het functie te laten werken. Na dat ik alles had gedeclareerd binnen de diverse (.h) bestanden ben ik begonnen met de maken van de constructor's binnen de respectievelijke (.cpp) bestanden. Hier onder zie je een van mijn constructor die ik geschreven heb.

```
AutoHuur::AutoHuur(Auto& gehuurde_auto, Klant& huurder, int dagen):
gehuurde_auto(gehuurde_auto),
huurder(huurder),
aantal_dagen(dagen)
{}
```

De constructor's zijn van belang om klassen objecten te kunnen maken die we later nodig zouden hebben bij het creëren van een klant, auto en een autoverhuur. Verder heb ik ook alle member functies die opgeroepen moeten worden door een ander member functie een constante declaratie gegeven. Ook is het belangrijk dat er een default constructor gegeven kan worden. Dit is zo zodat wanneer een object geinitialiseerd kan worden zonder enige initialisatie waardes de object een default constructor kan toewijzen.

Binnen de (.h) bestanden ben ik ook bezig geweest met het aanmaken van header guards om definitie duplicatie te voorkomen. Uiteindelijk heb ik ook een operator overloading functie gemaakt om objecten te kunnen uitprinten. Hieronder zie je een van de header guards die ik heb geschreven.

```
#ifndef AUTO_H
#define AUTO_H
```

Waar had ik problemen mee?.

In het begin van het opdracht had ik voornamelijk problemen mee dat ik de syntax van het programmeer taal moet opzoeken om efficiënt code te kunnen schrijven. Ook moest ik opzoeken hoe sommige functies binnen de standaard bibliotheek van C++ ter werking ging. Verder had ik de meeste moeite bij het maken van de custom operator overloading functie die er voor zorgt dat er objecten uitgeprint kunnen worden.

Dit was voornamelijk lastig aangezien ik niet wist waar de functie en de declaratie moest komen binnen de bestanden.

```
friend std::ostream& operator<<(std::ostream& out, const Klant& k);
```

Ik kwam er achter dat er ten eerste een declaratie moest komen van de custom operator overloading functie.

```
std::ostream& operator<<(std::ostream& out, const Klant& k){

  return out << k.naam << " (korting: " << k.korting_percentage << "%)" << "\n";
}
```

Na dat ik de declaratie had gemaakt kon ik verder de functie uitschrijven zodat we de object kunnen uitprinten. Ik kwam op dit oplossing door veel research te doen over hoe operating overloading functies werken.

Hoe kan ik deel 1 verbeteren.

Er zijn momenteel geen verbeterpunten die ik bedenken op het moment.

Uitkomst

Hieronder in de afbeelding zien we de uitkomst van deel 1.

```
Eerste autohuur:
  autotype: Peugeot 207 met prijs per dag: 50
  op naam van: Mijnheer de Vries (korting: 10%)
  aantal dagen: 4 en dat kost 180

Tweede autohuur:
  autotype: Ferrari met prijs per dag: 3500
  op naam van: Mijnheer de Vries (korting: 10%)
  aantal dagen: 1 en dat kost 3150

Gehuurd: Peugeot 207 met prijs per dag: 50
Gehuurd: Ferrari met prijs per dag: 3500
```

Deel 2 (Gameswinkel)

Wat moest ik maken?.

Na dat ik klaar was met het eerste deel van de opdracht, ben ik begonnen aan het tweede deel dat gaat over een Gameswinkel waar games verkocht kunnen worden aan personen. Ook is het zo dat binnen de opdracht transacties uitgeprint kunnen worden en dat personen meerdere games kunnen kopen.

Hoe heb ik het gemaakt?.

Ik ben begonnen net als deel 1 van het opdracht met het aanmaken van (.h) en (.cpp) bestanden wat nodig zijn in het compileren van het opdracht. Ik begon met het declareren van de member functies die nodig waren binnen de (.h) bestanden. Dit was nodig aangezien er declaratie nodig is om de functionaliteit van het functie te laten werken.

```
class Persoon {

  private:
    std::string name;
    double money;
    std::vector<Game> boughtgames;
  public:
    Persoon(std::string name, double money);
    bool koop(const Game &game);
    bool verkoop(const Game &game, Persoon &persoon);
    friend std::ostream& operator<<(std::ostream& out, const Persoon& k);
};
```

Snel kwam ik achter, dat ik een vector nodig had om dit deel van het opdracht te kunnen realiseren. Dit is zo aangezien elke gekochte game van een persoon opgeslagen moet kunnen worden. Hieronder zie je hoe ik een vector heb aangemaakt.

```
std::vector<Game> boughtgames;
```

Na dat ik klaar was met het declareren van mijn (.h) bestanden ben ik begonnen met het maken van constructors en functies binnen mijn (.cpp) bestanden.

Waar had ik problemen mee?.

Ik had voornamelijk problemen met het vector. De probleem ontstond voornamelijk toen ik een object binnen een lijst probeerde te vinden. Het is zo dat de std::find functie het niet goed vind dat er een vergelijking word gemaakt zonder dat er een custom operator< functie er voor gemaakt wordt waar er word vergeleken of een object gelijkwaardig staat aan een ander object binnen de vector. Om deze probleem op te lossen heb ik de volgende functie gedeclareerd binnen mijn game.h bestand.

```
bool operator==(const Game &other) const {
    return game == other.game && release == other.release && price == other.price;
}
```

Hoe kan ik deel 2 verbeteren?.

Aangezien ik eerst std::cout gebruikte binnen mijn koop en verkoop functies en nu het verplaats hebt naar main, heb ik geen verbeterpunten op het moment waar ik aan kan denken. Eventueel kan je ook een set of unordered set gebruiken inplaats van een vector.

Uitkomst

```
p1: Eric heeft een budget van 142.479 en bezit de volgende games:
Just Cause 3, uitgegeven in 2021; nieuwprijs: 49.98 nu voor 34.986
Need for Speed: Rivals, uitgegeven in 2020; nieuwprijs: 45.99 nu voor 22.5351

p2: Hans heeft een budget van 32.4649 en bezit de volgende games:
Need for Speed: Rivals, uitgegeven in 2020; nieuwprijs: 45.99 nu voor 22.5351

p3: Arno heeft een budget van 162.465 en bezit de volgende games:
Need for Speed: Rivals, uitgegeven in 2020; nieuwprijs: 45.99 nu voor 22.5351

Eric verkoopt Just Cause 3 aan Arno: gelukt
Hans verkoopt Need for Speed: Rivals aan Arno: niet gelukt
Hans verkoopt Just Cause 3 aan Eric: niet gelukt

p1: Eric heeft een budget van 177.465 en bezit de volgende games:
Need for Speed: Rivals, uitgegeven in 2020; nieuwprijs: 45.99 nu voor 22.5351

p2: Hans heeft een budget van 32.4649 en bezit de volgende games:
Need for Speed: Rivals, uitgegeven in 2020; nieuwprijs: 45.99 nu voor 22.5351

p3: Arno heeft een budget van 127.479 en bezit de volgende games:
Need for Speed: Rivals, uitgegeven in 2020; nieuwprijs: 45.99 nu voor 22.5351
Just Cause 3, uitgegeven in 2021; nieuwprijs: 49.98 nu voor 34.986
```