1. Vector geheugen

1.1. Namen en datum

Teamlid 1: Jip Galema

Teamlid 2: Tim IJntema

Datum: 23-2-2017

1.2. Doel

Wij gaan bij deze meting kijken wat de verschillen zijn in geheugengebruik tussen 1D arrays op de heap die handmatig aangemaakt zijn of 1D arrays die gemaakt zijn door middel van de STL vector. Hierbij vragen wij ons dus af welke implementatie minder geheugen gebruikt, de 1D array die in de standaard implementatie gebruikt wordt of de STL-vector die wij willen gebruiken.

1.3. Hypothese

Wij verwachten dat de vector minder geheugen gebruikt. Dit baseren we op het feit dat de vector gemaakt is om zo efficient mogelijk met geheugen om te gaan. Daarbij zal de vector altijd alles correct dealloceren waardoor er nooit geheugen lekken kunnen ontstaan. Deze kans is er wel bij de handmatig aangemaakte 1D array aangezien je heel makkelijk één foutje over het hoofd ziet.

1.4. Werkwijze

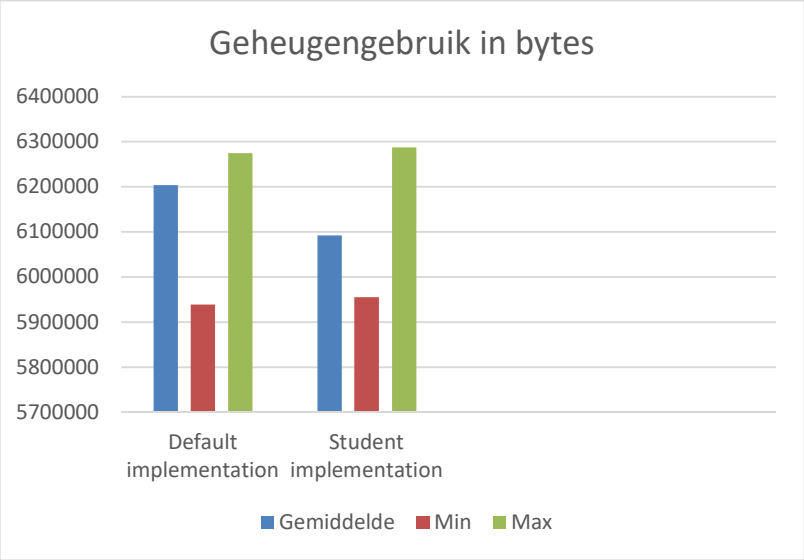
Wij gaan onze imageshellversie met STL-vector implementeren en deze dan vergelijken met de al geïmplementeerde imageshellversie. We gaan de geheugengebruik meet implementatie doen die beschreven staat op de volgende 3 links: https://msdn.microsoft.com/en-us/library/ms683219.aspx, https://msdn.microsoft.com/en-us/library/windows/desktop/ms683179(v=vs.85).aspx, http://stackoverflow.com/questions/8644110/how-to-use-getprocessmemoryinfo-in-c. Hierbij volgen we het volgende stappenplan:

- Start het programma
- Wacht tot de meetwaarde op het scherm komt
- Noteer deze waarde
- Herhaal tot het 5 keer gedaan is

We meten alleen de RGB imageshell aangezien de intensity imageshell hetzelfde geimplementeerd is.

1.5. Resultaten

In de grafiek aan de linkerkant kan je zien dat de student implementation gemiddeld minder geheugen gebruikt. De minimum en maximum waardes zijn wel vergelijkbaar met de default implementation.



1.6. Verwerking

In de resultaten kon je duidelijk zien dat de vector implementatie minder geheugen gebruikt en daarom is onze hypothese juist.

1.7. Conclusie

Uit deze metingen kunnen we concluderen dat de vector implementatie voor geheugengebruik de betere implementatie zou zijn.

1.8. Evaluatie

Hoewel volgens de metingen de hypothese juist was zaten de metingen met minimale en maximale waardes behoorlijk dicht bij elkaar. Dit zou gewoon toeval kunnen zijn. Ondanks dat zou een verbetering van het experiment het doen van meer metingen kunnen zijn. Verder is er nog een andere verbetering mogelijk, namelijk het toevoegen van een peak meting. De peak meting zou laten zien wat het hoogste geheugen gebruik was tijdens het afspelen van het algoritme. Hierdoor zou je te weten kunnen komen hoeveel geheugen een systeem minimaal nodig zou hebben om dit algoritme te kunnen afspelen.