Implementatieplan voor een efficiënte en veilige container

Patrick Dekker
Mark Gasse
Datum: 14 april 2019

Inhoudsopgave

Doel

Het doel van de implementate is om een container te bouwen voor RGB en grayscale images die snelle acces time en met weinig kans op segmentatie fouten doormiddel van misbruik.

Methoden

- 2D Array
    - Pros
        - Acces is complexiteit is O(1) waardoor snelheid voor get- en setPixel redelijk hoog is.
        - Kan makkelijk gebruikt worden door stl functions.
    - Cons
        - Arrays zijn fixed size hierdoor is uitbreiding een zware operatie
        - Insertion en deletion zij zeer traag
        - Overhead
- 2D Vector
    - Pros
        - Acces complexiteit is O(1)
        - De vector hoeft niet helemaal opnieuw gegenereerd worden
        - Kan makkelijk gebruikt worden door stl functions
    - Cons
        - Insertion en deletion kunnen O(n) zijn
        - Overhead
- Linked list
    - Pros
        - Is memory efficiënt
        - Is efficiënt als het wordt uitgebreid
    - Cons
        - Kan niet door alle stl functions worden gebruikt
        - Insertion, deletion zijn erg hoog
        - Heeft geen random access
- 1D Array
    - Pros
        - Iets snellere acces doordat er minder handelingen zijn
        - Alle andere pros van de 2d array
    - Cons
        - Implementatie is iets lastiger dan een 2D array
        - Handmatige memory management
        - Alle andere cons van 2D vector
- 1D Vector
    - Pros
        - Snellere acces ten opzichte van de 2D vector
        - Alle andere Pros van 2D Vector
    - Cons
        - Implementatie is iets lastiger
        - Alle Cons van de 2D vector

Keuze

Uiteindelijk hebben we gekozen voor een 1-dimensionale vector, de overhead nemen we dan voorlief. De vector bevat RGB-objecten. Verder kunnen we de vector sneller maken door de gewenste grootte va tevoren te initialiseren.

Implementatie

Wij gaan RGBImageStudent implementeren met een 1D vector met RGB pixels. In de vector worden sequentieel opgeslagen alsof alle rijen onder elkaar staan.

De Intensity image kan op dezelfde manier worden geïmplementeerd maar in plaats van RGB pixels zullen er dan intensity pixels in de vector staan. Ook bij deze implementatie staan de rijen onder elkaar.

Het is nodig om zowel een get en set voor x/y en i te maken omdat door de default implementatie x/y word gebruikt. In onze eigen functies zullen we get en setPixel met i gebruiken omdat deze sneller is. Bij de get en set met x/y word x eerst vermenigvuldigd met de hoogte waarna y erbij op word geteld waarna er word gekeken of deze waarde niet groter is dan de vector. Bij set en get met alleen de index word er alleen gekeken of deze in vector past.

Evaluatie

We zullen testen of een vector met pointers naar RGB pixels wel of niet sneller is dan een vector met gewoon RGB pixels er in.

Ook zullen we tests uitvoeren over hoe RGBStudentImage presteert tegenover de default implementatie.