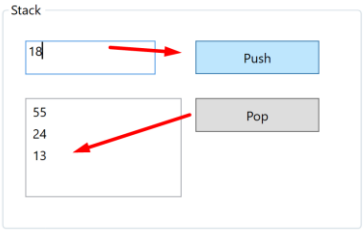
Datastructures, Oefeningen Hoofdstuk 1

Stack

1. Bouw een Stack (versie 1) met behulp van een array

- Een stack heeft 2 Methodes: **Push & Pop**
- We bouwen een stack waar we getallen (**int**) kunnen in bijhouden.
- Zorg dat je elementen kan "**push"en tot de stack vol is** (negeer de waarde gewoon als de stack vol is en er wordt opnieuw push aangeroepen)
- Zorg dat je **geen elementen meer kan "pop"pen als de stack leeg is**
  (geef dan bv. -99999999 terug als waarde)
- Kies **5** voor de **grootte van je array**

Voorzie in je WPF MainWindow de nodige controls om je stack te testen, bv.

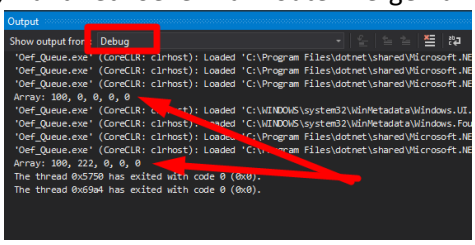


Door in te schrijven op de events van de buttons kan je dan de Push en Pop methodes van je stack gaan aanroepen en zo je stack testen.

Tip: om de array (mijn array noemt 'queue') zichtbaar te maken kan je in de code dit toevoegen op de plaatsen waar je array wordt aangepast (of je maakt er een aparte methode van die je telkens aanroept):

```
Debug.WriteLine($"Array: {string.Join(", ", queue)}");
```

Onderaan in het Output window zal je dan ook de inhoud van je array kunnen bekijken terwijl je de applicatie test, want het zoeken van fouten vergemakkelijkt:



<u>Mogelijke verbeteringen:</u>

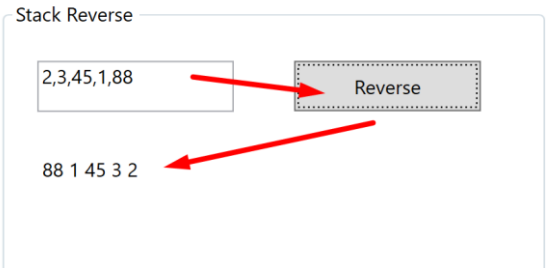
- Maak het invoerveld leeg, telkens je een push hebt gedaan, ter bevestiging dat de push gelukt is, maw. de gebruiker **heeft en geldig getal ingevoerd en de stack was niet volzet**
- Zorg dat de **knop 'Pop' inactief** (disabled) is wanneer de **stack leeg** is.
- Zorg dat de **knop 'Push' inactief** is wanneer er **geen (geldig) getal is ingevoerd** in het input veld.

2. Bouw een Stack (versie 2), of pas versie 1 aan

- Als de stack vol is en er wordt een nieuwe waarde gepushed, gaan we dit wel toelaten.
- Maak dan een nieuwe array die **2x de grootte** heeft van de huidige array
- De aanroeper merkt hier niets van want de push zal nu gewoon altijd lukken.
- We kunnen nu dus getallen blijven pushen in de stack tot het geheugen van onze laptop volzet is….. ? (en wat met onze 'Top' variabele ???)

Test nu of je wel degelijk meer dan 5 getallen op de stack kan pushen. Meer dan tien ? 20 ?

3. Een stack heeft de eigenschap dat een reeks getallen kan worden omgekeerd van volgorde.
    - We hergebruiken de stack die we reeds gebouwd hebben, maar…
    - Breid de UI uit zodat je ook een reeks van getallen in 1 maal kan invoeren (bv. gescheiden door een komma)
    - Bij het drukken op de knop zal de reeks worden ingelezen, de getallen zullen vervolgens 1 voor 1 op de stack (die je reeds hebt gemaakt) worden gepushed.
    - Onmiddellijk nadien zullen alle getallen van de stack worden gehaald (pop) tot de stack terug leeg is. De uitvoer van de stack wordt terug getoond op het scherm.

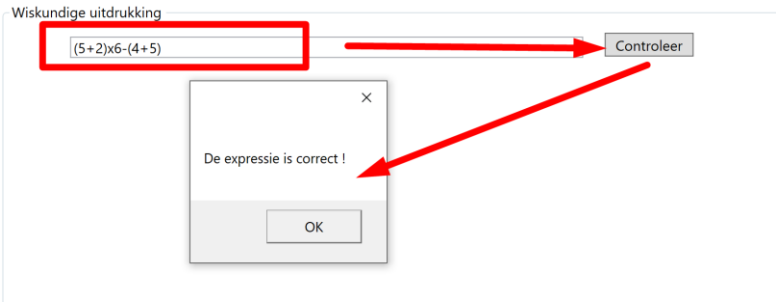


4. Geldigheid van een wiskundige uitdrukking valideren

We gaan voor deze oefening een nieuw **Window** toevoegen aan het WPF project => **Add Window (WPF)**

- Om te zorgen dat dit venster wordt getoond bij het opstarten moet je dit aanpassen in het **App.xaml** bestand => **StartupUri**
- We willen de geldigheid nakijken van een **eenvoudige** wiskundige uitdrukking, dwz. **nakijken of de haakjes in balans zijn**.
- Aangezien we nu niet meer met getallen gaan werken maar met karakters, hebben we een bijkomende stack nodig
- We voorzien hiervoor dus een 2e stack in het .cs bestand van dit nieuwe window met uiteraard ook weer een Push en een Pop methode (je kan de code van Stack1 uiteraard kopiëren).
- De werking blijft identiek aan stack 1, echter nu met **type string ipv. int**
- Maak voor deze oefening nu gebruik van je stack met string types. Werk in een eerste versie enkel met ronde haakjes. Bouw nadien eventueel een 2e versie met ook rechte haakjes.

Voorzie je window van de nodige UI controls om de expressie te kunnen invoeren en een knop om de controle te starten. Geef een boodschap als de expressie werd gecontroleerd en ze al dan niet geldig is.



Tip: De boodschap tonen, kan heel eenvoudig met **Messagebox.Show("dit is een boodschap !");**

Tip: je kan met dezelfde methode nu ook vrij eenvoudig de code schrijven om na te gaan of een HTML document geldig is (dus of de nodige "closing tags" aanwezig zijn en op de juiste plaats staan)

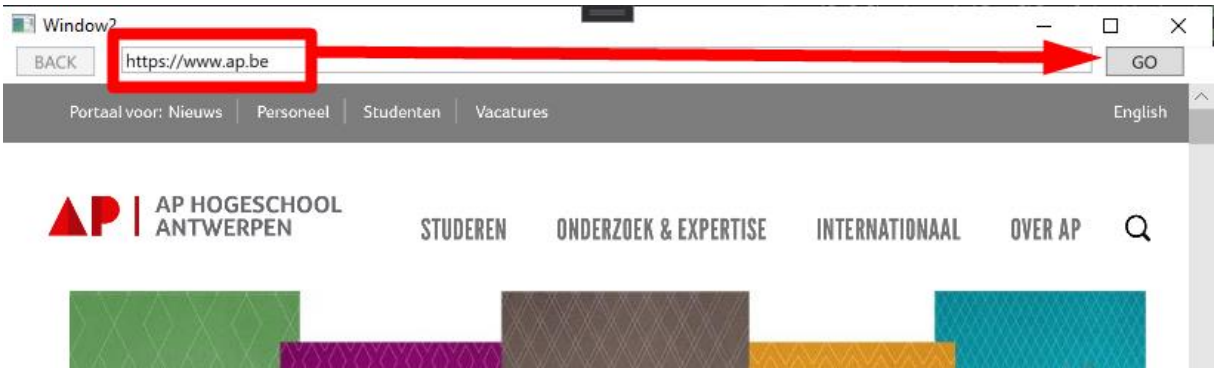
```
<html>
   <head>
      <title>
         Example
      </title>
   </head>

   <body>
      <h1>Hello, world</h1>
   </body>
</html>
```

5. Internet Browser

We gaan voor deze oefening wederom een nieuw **Window** toevoegen aan het WPF project => Add Window (WPF)

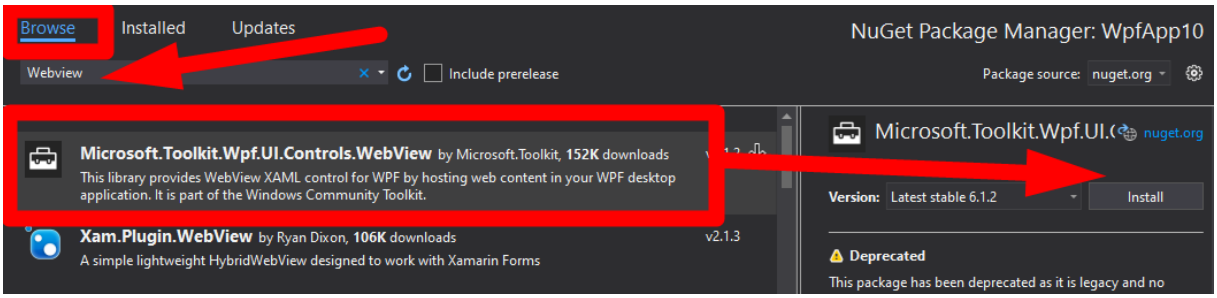
- Om te zorgen dat dit venster wordt getoond bij het opstarten moet je dit aanpassen in het **App.xaml** bestand => **StartupUri**
- Om het gedrag van een internet browser na te bootsen hebben we eveneens een stack nodig waar we url's (dus type **string**) kunnen in opslaan.
- Kopieer je stack code van de vorige oefening in het .cs bestand van dit nieuwe window.

Vervolgens gaan we onze WPF window3 voorzien van de nodige UI controls: textbox, buttons en een WebView control *(deze laatste is apart te installeren, zie hieronder)*

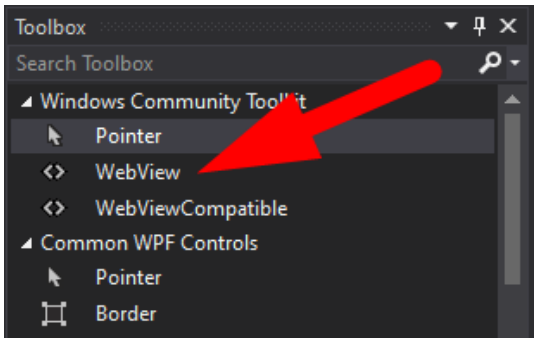


Om de **webview browser control** te kunnen gebruiken moet je eerst een **nuGet package** installeren. Deze control zit immers niet standaard bij onze applicatie maar moeten we apart toevoegen.

Open hiervoor de package manager via het popup menu van je project -> **Manage NuGet Packages**, zoek naar "Webview" en installeer de juiste package.



Na installatie mag je dit venster sluiten en krijg je de "**Webview**" control te zien in de toolbox onder **Windows Community Toolkit.** Deze kan je vervolgens in je window slepen zoals de andere controls. Geef deze uiteraard ook een naam.



Om de URL in te stellen kan je vanuit c# code kan je de Navigate("…") methode gebruiken.

Je kan nu de browser afwerken zodat:

- De gebruiker een URL kan invoeren, en met de knop '**GO**' deze site kan openen
- Als de gebruiker meer dan 1 site bezoekt zal hij/zij met de **BACK** knop terug kunnen navigeren. Gebruik hiervoor uiteraard jouw Stack ! (en kijk ook even terug naar de animatie in de presentatie van de theorie sessie)

Mogelijke verbeteringen:

- Onze **Back** knop werkt enkel voor URL's die we hebben ingegeven **via de adres balk bovenaan**. Als de gebruiker echter **klikt op een hyperlink in de pagina zelf** willen we deze ook kunnen registreren als een bezochte website (en deze zouden dus ook bij op onze stack moeten geplaatst worden). We kunnen dit realiseren door in te schrijven op het **NavigationStarting event** van onze browser control. Dit event zal immers afgaan bij **elke adres wijziging** in de browser. Breid uit zodat ook dit werkt (merk op: dit is niet zo super eenvoudig…)
- Uiteraard is de GO knop overbodig. Zorg dat onze browser ook zonder deze knop werkt.

6. Ze je stack om naar een OO versie

- Maak een **StackInt** en een **StackString** klasse
- Voorzie deze van de nodige methodes (Push, Pop, …)
- Maak er een werkende stack van.
- Vervolgens kan je je oefeningen aanpassen zodat ze met deze klassen werken