CHAPTER 1

OIS11-dictaat

FHICT

1.1 Werken met variabelen in C#

Dit hoofdstuk is geschreven als een *naslagwerk*, het is niet geschikt om uit te leren hoe je met variabelen programmeert.

Typen variabelen

Een variabele is een stukje geheugen waarin tijdelijk een waarde kan worden opgeslagen. De veelgebruikte typen variabelen zijn:

Inhoud	Naam	Voorbeelden
Stukje tekst	String	”abcde” ”dit is een tekst” ”” etc.
Geheel getal	Int	12 -1337 0 etc.
Komma getal	Double	10.2 -12.3 5.0 etc.
Waar of niet waar	Bool	true, false

Variabele aanmaken (declareren)

Variabelen kunnen op verschillende manieren worden aangemaakt, enkele voorbeelden staan hier-onder. Merk op dat:

- Je de variabele naam zelf kunt kiezen
- De regel moet worden beeindigd met een ";"-teken

Op verschillende manieren kunnen variabelen worden aangemaakt. Programmeer op een lege regel het type van de variabele (zie hierboven), de naam die je de variabele wil geven (deze kies je zelf) en een ";" teken om het programmeercommando af te sluiten.

Voorbeeld	Effect
String s;	Variabele met de naam s wordt aangemaakt. De default waarde is "".
`int i;`	Variabele met de naam `i` wordt aangemaakt. De default waarde is 0.
`double d;`	Variabele met de naam "d" wordt aangemaakt. De default waarde is 0.0
Bool b;	Variabele met de naam "b" wordt aangemaakt. De default waarde is false
String mijnString;	Variabele met de naam "mijnString" wordt aangemaakt. De default waarde is ""
int getal;	Variabele met de naam "getal" wordt aangemaakt. De default waarde is 0
double straal;	Variabele met de naam "straal" wordt aangemaakt. De default waarde is 0.0

Direct na het aanmaken heeft een `variabele` een `waarde` die we de `default waarde` noemen. Dit kan per programmeertaal enigszins verschillen. Daarom is het een goede gewoonte variabelen waarvan je wil dat ze een specifieke waarde hebben deze waarde expliciet toe te kennen.

Waarde aan variabele geven (toekenning of `assignment`)

Als een variabele eenmaal is aangemaakt kan hier een waarde aan worden toegekend. Merk op:

- Alleen geldige waarden kunnen worden toegekend (string waarden aan strings, getallen aan int, etc.), het programmeren van een niet geldige toekenning levert een fout op waardoor het programma niet kan worden uitgevoerd.
- De variabele waaraan een waarde moet worden toegekend staat aan de linkerkant van het "=" teken, en de waarde welke in de variabele moet worden gestopt staat rechts van het "=" teken.

- De regel code wordt weer beeindigd met het ";"-teken.

Hier volgen enkele voorbeelden. In commentaar staat erbij uitgelegd wat het betekent.

```
String s;     // maak een variabele aan met naam "s".
s = "test";   // Variabele met de naam "s" krijgt de waarde "test".
```

```
int i;
i = 10; // maak variabele met naam "i" aan en geef die waarde 10
```

```
double d;
d = 1.52; //  Nieuwe variabele genaamd "d" krijgt de waarde 1,52
```

```
bool b;
b = true; // Nieuwe variabele "b" krijgt de waarde true
```

```
String string1;
string 1 = "abc";
String string2;
string2 = string1;  // Variabele met de naam "string2" krijgt
                    // de waarde van "string1", namelijk "abc"
```

```
int getalA;
getalA = 5;
int getalB;
getalB  = getalA; // Variabele met de naam "getalB" krijgt
                  // de waarde van "getalA", namelijk 5
```

```
double kommaGetalA;
kommaGetalA = 1.32;
double kommaGetalB;
kommaGetalB  = kommaGetalA; // Variabele met de naam "kommaGetalB"
    krijgt
                            // de waarde van "kommaGetalA",
                            // namelijk 1.32
```

```
String s;
s = textBox1.Text;
    // Variabele met de naam "s" krijgt
    // als waarde de tekst die in de
    // TextBox genaamd "textBox1" staat.
```

Dit werkt omdat de `Text property` van de *TextBox* ook van het `type string` is.

Variabele aanmaken en direct een waarde geven (`declare` en `initialize`)

Variabele met de naam *s* aanmaken en waarde "test" toekennen:

```
String s = "test";
```

Variabele met de naam *i* aanmaken en waarde `10` toekennen:

```
int i =10;
```

Variabele met de naam *d* aanmaken en waarde `1,52` toekennen:

```
double d = 1.52;
```

Variabele met de naam *b* aanmaken en waarde `true` toekennen:

```
bool b = true;
```

Waarden omzetten naar andere typen (`convert`)

> **Note** Merk op: het omzetten van een `int` of `double` naar een `String` lukt altijd, andersom lukt niet altijd en kan een foutmelding opleveren tijdens het uitvoeren van het programma (`crash` of `Unhandled Exception`).

Note Een `bool` variabele kan niet worden geconverteerd.

Zet de waarde van *i* om naar een tekst met dezelfde waarde. Het resultaat van de laatste regel is dat variabele *s* de waarde `81` krijgt.

```
int i = 81;
String s;
s = Convert.ToString(i);
```

Zet de waarde van *d* om naar een tekst met dezelfde waarde. Het resultaat van de laatste regel is dat variabele *s* de waarde `"12.33"` krijgt:

```
double d =12.33;
String s;
s = Convert.ToString(d);
```

Zet de waarde van *s* om naar een geheel getal (`integer`) met dezelfde waarde als dat lukt (anders krijg je een foutmelding). Het resultaat van de laatste regel is dat variabele *i* de waarde `7` krijgt:

```
int i;
String s = "7";
i  = Convert.ToInt32(s);
```

Zet de waarde van *s* om naar een *kommagetal* met dezelfde waarde als dat lukt (anders krijg je een foutmelding). Het resultaat van de laatste regel is dat variabele *d* de waarde `12.129` krijgt:

```
double d;
String s = "12.129";
d  = Convert.ToDouble(s);
```

String bewerkingen (String functies)

Hieronder worden enkele veelgebruikte String functies gedemonstreerd en kort toegelicht.

String's samenvoegen

Met het `plus` teken kunnen strings aan elkaar worden geplakt.

```
string tekst = "een tekst.";
string woorden = "Hier staat";
string s = woorden+tekst;
```

De `s variabele` krijgt hier de waarde `"Hier staateen tekst."` Merk op dat niet automatisch spaties worden toegevoegd.

```
  string tekst = "tekst.";
  string woorden = "Hier staat";
  string s = woorden + " een " + tekst;
```

Met het "+"-teken kunnen strings aan elkaar worden geplakt. De "s" variabele krijgt hier de waarde "Hier staat een tekst."

`IndexOf`

De plaats van een String binnen een andere String bepalen:

De *Positie* variabele krijgt de waarde `1`. Merk op dat de positie van de eerste gevonden "e" in de String wordt gevonden (waarbij vanaf `0` wordt geteld):

```
string tekst = "regel tekst";
int positie = tekst.IndexOf("e");
```

Er kan ook naar meerdere letters achter elkaar gezocht worden:

```
string tekst = "regel tekst";
int positie = tekst.IndexOf("tek");
```

De "Positie" variabele krijgt de waarde 6.

Niet gevonden geeft `-1`:

```
string tekst = "regel tekst";
int positie = tekst.IndexOf("a");
```

De "Positie" variabele krijgt de waarde -1. De waarde -1 betekent dus: de String komt niet voor binnen de andere String.

Substring

Een stukje uit een string kopiëren:

```
string tekst = "regel tekst";
string deelTekst = tekst.Substring(0, 1);
```

wat heeft als resultaat dat in deelTekst de waarde ”r” komt te staan omdat van de oorspronkelijke tekst vanaf positie `0` precies `1` letter gekopieerd wordt.

```
string tekst = "regel tekst";
string deelTekst = tekst.Substring(6, 5);
```

Deze code heeft als resultaat dat in deelTekst de waarde ”tekst” komt te staan omdat van de oorspronkelijke tekst vanaf positie `6` precies `5` letters gekopieerd worden.

Length

Aantal tekens van de String bepalen. Achter `Length` hoeven geen haakjes openen en sluiten geplaatst te worden omdat het een `property` (eigenschap) van de string is en niet een `method` die je uitvoert.

```
string tekst = "regel tekst";
int lengte = tekst.Length;
```

Deze code heeft als resultaat dat in lengte de waarde `11` komt te staan omdat de tekst precies elf lang is. Merk op: dit is inclusief spaties in de tekst. De `double quotes` om begin en einde van de String waarde aan te geven worden niet meegeteld.

```
string tekst = "";
int lengte = tekst.Length;
```

Deze code heeft als resultaat dat in *lengte* de waarde `0` komt te staan omdat geen tekens in de string staan.

int en double bewerkingen (operatoren)

Onderstaande bewerkingen zijn zowel op `int typen` als op `double` typen van toepassing:

```
int k;
k = 5 + 10;
```

Aan variabele *k* wordt in de laatste regel code de waarde `15` toegekend omdat het `+`teken de waarden `5` en `10` bij elkaar optelt.

```
int i = 2;
int k;
k = i + 1;
```

Aan variabele *k* wordt in de laatste regel code de waarde `3` toegekend omdat het `+`teken de waarden `2` en `1` bij elkaar optelt.

```
int i = -8;
int k;
k = 1 + i;
```

Aan variabele *k* wordt in de laatste regel code de waarde `-7` toegekend omdat het `+`teken de waarden `1` en `-8` bij elkaar optelt.

```
int i = 5;
int j = 3;
int k;
k = i + j;
```

Aan variabele *k* wordt in de laatste regel code de waarde `8` toegekend omdat het `+`teken de waarden uit *i* en *j* bij elkaar op telt.

Bij bovenstaande voorbeelden kan de operator (het `+`teken) worden vervangen door één van de volgende mogelijkheden:

Symbool	Uitwerking
+	Optellen
-	Aftrekken
*	Vermenigvuldigen
/	Delen
%	Geeft de rest na deling. Bijvoorbeeld: 7 % 5 = 2 11 % 2 = 1 6 % 2 = 0

1.2 Werken met keuzestructuren in C#

Dit hoofdstuk is geschreven als een *naslagwerk*, het is niet geschikt om uit te leren hoe je met variabelen programmeert.

Als een stukje code soms wel en soms niet moet worden uitgevoerd, dan heb je een `if` of `if … else statement` nodig. Moet een stukje code soms één keer en soms vaker worden herhaald, dan heb je een `for` of `while statement` nodig.

`if-statement`

Deze structuur wordt gebruikt om een stukje code uit te voeren afhankelijk van een bepaalde situatie (de `conditie` genoemd). Algemene vorm:

```
if ([conditie])
{
  [Uit te voeren code als conditie waar is]
}
```

waarbij `conditie` is een stelling die de waarde `true` (*waar*) of `false` (*niet waar*) heeft.

Voorbeelden van condities:

Conditie	Betekenis
true	Waar
false	Niet waar
i > 5	Is i groter dan 5?
i < 7	Is i kleiner dan 7?
i >= 1	Is i groter of gelijk aan 1?
i <= 2	Is i kleiner of gelijk aan 2?
i == 3	Is i precies gelijk aan 3?
i != 3	Is i ongelijk aan 3?
stukjeText == ”abcde”	Is *stukjeText* precies gelijk aan ”abcde”?
stukjeText < ”abcde”	Komt *stukjeText* eerder in het alphabet dan ”abcde”?
etc.	

Verder staat *[Uit te voeren code als conditie waar is]* voor een stukje code (dit kunnen meerdere regels code zijn) dat moet worden uitgevoerd als de conditie `true` (*waar*) is.

Als precies één regel code moet worden uitgevoerd zou je ervoor kunnen kiezen de accolades openen en sluiten weg te laten, maar dit maakt de kans op bugs een stuk groter, dus dat raden we af.

`if … else … statement`

Een if statement kan uitgebreid worden met een ”else” blok. Als de conditie niet ”waar” oplevert dan wordt de code in het else blok uitgevoerd. Algemene vorm:

```
if ([conditie])
{
[Uit te voeren code als conditie waar is]
}
else
{
 [Uit te voeren code als conditie niet waar is]
}
```

Merk op: of de conditie nu wel of niet waar is, altijd wordt één van de twee stukjes code uit-ge-voerd.

Voorbeelden ”if …” statement en ”if … else …” statement

```
if (true)
{
  TextBox1.Text = "test";
}
```

Het stukje code tussen { en } wordt altijd uitgevoerd, dus de `Text` van de *TextBox* wordt altijd ”test” gemaakt.

```
if (false)
{
  TextBox1.Text = "test";
}
```

Het stukje code tussen { en } wordt nooit uitgevoerd.

```
bool b = true;
if (b)
{
  TextBox1.Text = "test";
}
```

Als `b` de waarde `true` (= *waar*) heeft wordt de `Text` in de *TextBox* ”test” gemaakt. Dit is hier nu altijd het geval omdat in dit stukje code aan variabele `b` alleen de waarde ”true” wordt toegekend.

```
int i = 10;
if (i < 5)
{
  i  = i + 1;
}
```

Als getal `i` kleiner dan `5` is, dan wordt bij de waarde van `i` één opgeteld, anders gebeurt er niets.

```
TextBox1.Text = "test2";
if (TextBox1.Text !=  "test")
{
  TextBox1.Text = "test3";
}
```

Als de tekst in de textbox niet gelijk is aan ”test” (dat is hier het geval) dan wordt de tekst van de textbox veranderd in ”test3”.

```
if (true)
{
  TextBox1.Text = "test";
}
else
{
  TextBox1.Text = "test2";
}
```

Het stukje code tussen de eerste { en } wordt altijd uitgevoerd, dus de `Text` van de `TextBox` wordt altijd ”test” gemaakt. Het stukje code tussen de tweede { en } wordt nooit uitgevoerd.

```
int i = 5;
if (i >= 10)
{
  i  = i + 1;
}
else
{
  i = i + 5;
}
```

Als getal *i* groter of gelijk aan `10` is dan wordt bij getal *i* `1` opgeteld. Dit is hier niet het geval, dus wordt bij *i* `5` opgeteld. Resultaat: *i* krijgt de waarde `10`.

```
int i = 5;
if (i >= 10)
{
  i  = i + 1;
}
else
{
  i = i + 5;
  if (i >= 10)
  {
    i = 20;
  }
}
```

Als getal *i* groter of gelijk aan `10` is dan wordt bij getal *i* `1` opgeteld. Dit is hier niet het geval, dus wordt bij *i* `5` opgeteld. Resultaat: *i* krijgt de waarde `10`, vervolgens wordt gecontroleerd of `i >= 10`, dat is nu het geval dus krijgt *i* uiteindelijk de waarde `20` toegekend.

`while` statement

Deze structuur wordt gebruikt om een stukje code uit te voe-ren zolang aan bepaalde voorwaarden is voldaan. Dit varieert van `0` keer de code uitvoeren tot het in de oneindigheid aantal keer uitvoeren van de code).

Algemene vorm:

```
  while ([conditie])
  {
  [Uit te voeren code zolang de conditie waar is]
  }
```

Note Na de eerste regel staat geen ”;” teken.

Note Eerst wordt gecontroleerd of aan een voorwaarde is voldaan, dan pas wordt eventueel code uitgevoerd.

`do while statement`

Deze structuur wordt gebruikt om een stukje code uit te voeren. Elke keer nadat het stukje code is uitgevoerd wordt gecontroleerd of nog aan bepaalde voorwaarden is voldaan, zo ja, dan wordt de code opnieuw uitgevoerd. Het aantal keer uitvoeren van de code varieert van 1 keer de code uitvoeren tot het in de oneindigheid aantal keer uitvoeren van de code.

Algemene vorm:

```
do
{
  [Uit te voeren code zolang de conditie waar is]
} while ([conditie]);
```

Note Na de laatste regel staat een ”;” teken.

Note Eerst wordt de code één keer uitgevoerd, dan pas wordt gecontroleerd of de code eventueel vaker moet worden uitgevoerd.

Voorbeelden `while` en `do while statement`

```
int i = 0;
while(i < 10)
{
  MessageBox.Show("Test");
  i = i + 1;
}
```

Variabele *i* krijgt in het begin de waarde `0` en er wordt net zo lang doorgegaan met *MessageBoxes* weergeven totdat *i* kleiner dan `10` is. De code wordt dus doorlopen met achtereenvolgens de waarden `0` , `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8` en `9`. Er worden daarom `10` *Messageboxes* getoond met de tekst `"Test"`.

```
int i = 5;
while(i > 0)
{
  MessageBox.Show("Test");
  i = i – 1;
}
```

Variabele *i* krijgt in het begin de waarde `5` en er wordt direct gestopt als *i* de waarde `0` krijgt toegekend. De code wordt dus doorlopen met de waarden `5`, `4`, `3`, `2`, `1`. Er worden daarom `5` *Messageboxes* getoond met de tekst `"Test"`.

```
int i = 10;
do
{
  MessageBox.Show("Test");
  i = i + 1;
}
while (i < 5);
```

Variabele *i* krijgt in het begin de waarde `10`, de code wordt uitgevoerd, en vervolgens wordt net zo lang doorgegaan met *Messageboxes* weergeven totdat *i* kleiner dan `5` is. De code wordt dus doorlopen met de waarde `10`. Er wordt daarom `1` *Messagebox* getoond met de tekst `"Test"`.

`for statement`

Deze structuur wordt gebruikt om een stukje code een vooraf bekend aantal keer uit te laten voeren.

Algemene vorm:

```
for([teller variabele aanmaken]; [herhalingsconditie]; [teller
    variabele aanpassen])
{
  [herhaaldelijk uit te voeren code]
}
```

waarbij *[teller variabele aanmaken]* een `variabele` met zelf te kiezen variabelenaam wordt aangemaakt en van een waarde voorzien. Veel gebruikte variabele namen voor een for statement zijn ”i”, ”j”, ”k” omdat deze een hele korte naam hebben, dat leest in veel gevallen prettig. Ook ”index”, ”count” of ”teller” worden vaak gebruikt. Het type variabele is meestal int. De waarde waarmee de teller wordt gevuld is afhankelijk van wat je aan het programmeren bent. In veel gevallen heeft deze de waarde `0`.

Voorbeelden:

```
int i = 0
```

```
int j = 100
```

Dan *[herhalingsconditie]*: deze uit te voeren code wordt net zo lang herhaald als uit de voorwaarde de waarde `true` komt. Hierin verwijs je naar de *teller* variabele.

Voorbeelden:

```
i < 10
j > 0
```

[teller variabele aanpassen] Het verhogen of verlagen van de teller. Vaak wordt deze met 1 verhoogd of verlaagd, soms in grotere stappen (bijv. 10).

Voorbeelden:

```
i = i + 1
j = j – 10
```

[herhaaldelijk uit te voeren code] Het stukje code (dit kunnen meerdere regels code zijn) dat moeten worden uitgevoerd zolang de herhalingsconditie ”true” (waar) is.

Ieder `for`statement is om te zetten naar een `while` statement dat hetzelfde doet, en andersom.

Voorbeelden `for statement`

```
for(int i =0 ; i < 10 ; i = i + 1)
{
  MessageBox.Show("Test");
}
```

Variabele `i` krijgt in het begin de waarde 0 en er wordt direct gestopt als i de waarde 10 krijgt toegekend. De code wordt dus doorlopen met de waarden 0,1,2,3,4,5,6,7,8 en 9. Er worden daarom 10 messagebox-en getoond met de tekst ”Test”.

```
for(int i =5;i > 0; i = i - 1)
{
  MessageBox.Show("Test");
}
```

Variabele `i` krijgt in het begin de waarde 5 en er wordt direct gestopt als i de waarde 0 krijgt toegekend. De code wordt dus doorlopen met de waarden 5,4,3,2,1. Er worden daarom 5 messagebox-en getoond met de tekst ”Test”.

```
for(int i =0;i < 10;i++)
{
  MessageBox.Show("Test");
}
```

Hetzelfde resultaat als het eerste voorbeeld, maar dan in een verkorte schrijfwijze:

```
i = i + 1;
```

is hetzelfde als

```
i++;
```

Hetzelfde resultaat als het tweede voorbeeld, maar dan in een verkorte schrijfwijze:

```
for(int i =5;i > 0; i--)
{
  MessageBox.Show("Test");
}
```

```
i=i-1;
```

is hetzelfde als

```
i--;
```

De code

```
for(int i =0;i < 10; i++)
{
  MessageBox.Show("Test "+i);
}
```

heeft als resultaat dat *MessageBoxes* verschijnen met achtereenvolgens:

```
"Test 0"
"Test 1"
"Test 2"
"Test 3"
"Test 4"
"Test 5"
"Test 6"
"Test 7"
"Test 8"
"Test 9"
```

De code

```
for(int i =5;i > 0; i = i - 2)
{
  MessageBox.Show("Test "+i);
}
```

laat messagebox-en verschijnen met achtereenvolgens:

```
"Test 5"
"Test 3"
"Test 1"
```

en tot slot geeft

```
for(int y =0;y < 2; y++)
{
  for(int x =0;x < 3; x++)
  {
    MessageBox.Show("("+x+","+y+")");
  }
}
```

als resultaat *MessageBoxes* verschijnen met:

```
"(0,0)"
"(1,0)"
"(2,0)"
"(0,1)"
"(1,1)"
"(2,1)"
```

1.3 Werken met methoden in C#

Dit hoofdstuk is geschreven als een *naslagwerk*, het is niet geschikt om uit te leren hoe je met variabelen programmeert.

Algemene structuur methoden

Een methode is een stukje code dat vanuit een ander stukje code is aan te roepen. Als een methode een waarde terug geeft welke gebruikt gaat worden in het stukje code waar vanuit deze is aangeroepen spreek je over een methode welke "een waarde teruggeeft". Ook kunnen aan een methode één of meer waarden worden meegegeven. Dit worden argumenten genoemd.

Belangrijkste voordelen van het gebruik van methoden:

1. Overzichtelijkheid: Als alle code in één enkele event handler (bijv. *ButtonX_Click*) wordt geplaatst wordt je code al snel erg overzichtelijk.
2. Werk verdelen: Als je voordat je gaat programmeren het programmeerwerk wilt verdelen kun je de te maken code opdelen in methoden en deze met verschillende programmeurs tegelijkertijd programmeren.
3. Onderhoudbaarheid & herbruikbaarheid: Als je op verschillende plaatsen in je programma hetzelfde stuk code vaker uit wilt voeren kun je vanaf de verschillende plaatsen een methode aanroepen die je maar één keer hoeft te programmeren. Dat scheelt code en is gemakkelijker te onderhouden dan dat je code verschillende keren in je programma kopiert en plakt.

Een methode heeft de volgende structuur:

```
private [returnType] [methodeNaam]([parameters])
{
  …
  [return returnWaarde]
}
```

`private` geeft aan dat de `methode` alleen binnen het huidige bestand (lees: `Form1`) kan worden aangeroepen. Wat dit precies inhoudt is voor dit vak niet interessant, hier wordt later op teruggekomen.

[returnType] Het type van de waarde die de methode terug geeft. Als de methode geen waarde terug geeft, is dit `void` (*niets*). Voorbeelden: `int`, `double`, `bool`, `string`, `void`.

[methodeNaam] Zelf gekozen naam voor de methode, te vergelijken met een zelf gekozen naam voor een variabele. Voorbeelden: *TelOp*, *ToonMelding*, *Methode1*, *Abc*.

[parameters] Optioneel onderdeel. Hiermee wordt opgegeven welk(e) type(n) waarde(n) moet worden meegegeven aan de methode en onder welke naam deze waarde binnen de method kunnen worden gebruikt. Meerdere parameters worden gescheiden met een ","-teken. Voorbeelden: `int deler`, `int getalA`, `int getalB`, `bool isIngelogd`, `double eenKommaGetal`.

[return returnWaarde] Met `return`" gevolgd door een waarde die aan het *[returnType]* voldoet wordt een waarde teruggegeven vanuit de methode aan het stukje code dat de methode heeft aangeroepen. Als *[returnType]* `void` is hoeft er geen return worden gebruikt. Voorbeelden: `return uitkomst;`, `return 10;`, `return mijnTekst;`, `return "Hallo"+" daar!";`, `return getal > 10;`

Voorbeelden Methoden

Een aantal voorbeeldmethoden:

```
    private int AddTwoNumbers(int number1, int number2)
    {
        int som;
        som = number1 + number2
        return som;
    }

    private int SquareANumber(int number)
    {
        return number * number;
    }
```

Bovenstaande methoden zijn als volgt aan te roepen:

```
  int sum = AddTwoNumbers(8765, 287);
```

of:

```
  int kwadraat = SquareANumber(63);
```

of, beiden:

```
  int total = SquareANumber(AddTwoNumbers(1, 2));
```

Tekst en uitleg (engelstalig) over methoden en parameters

Zie bijvoorbeeld C-sharpcorner over methods[1] voor meer uitleg.

1.4 Handige sneltoetsen en opties in Visual Studio

> **Note** Als in een sneltoets combinatie een komma staat, dan moet eerst het deel voor de komma worden ingedrukt, dan laat je de toetsen los en druk je daarna de letter in die na de komma staat.

Sneltoets	Menu	Toelichting
CTRL-W, X	View Toolbox	Maakt het Toolbox scherm met alle visuele objecten zichtbaar.
CTRL-W, P	View Properties	Window Scherm met de eigenschappen van visuele objecten tonen. Hier vind je ook de events van de visuele objecten.
CTRL-W, E	View Error List	Toont het scherm met de eventuele fouten die in je code gevonden zijn (ververs het scherm door je project opnieuw te build-en met de F6 knop, zie hieronder). Dubbelklikken op een error navigeert naar de plaats in je code waar de fout is gevonden. Tip: bekijk eerst de eerste fout, andere fouten kunnen hier een gevolg van zijn.
F6	Build Build Solution	Controleert je code op fouten en bouwt vervolgens een uitvoerbaar bestand.
F5	Debug Start Debugging	Build je project (zie F6 hierboven) als dat nog niet gebeurd is, en voert het uitvoerbaar bestand vervolgens uit. Dit doet hetzelfde als een klik op de knop met het groene pijltje (`play`knop).
Shift-F5		Stop de uitvoer van je programma. Handig als je programma vast loopt.

[1] http://www.c-sharpcorner.com/UploadFile/myoussef/CSharpMethodsP_111152005003025AM/CSharpMethodsP_1.aspx

1.5 Online C-sharp tutorials

MSDN tutorial

MSDN How do I Learn C# tutorials (Engelstalig)[2]

Bruikbaarheid: *

Toelichting: Enkele gedetailleerde walkthroughs. Aanrader als je al bekend bent met programmeren in een andere programmeertaal.

techzine tutorial

Les 1: Beginnen met C# (Nederlandstalig)[3]

Bruikbaarheid: ****

Toelichting: Uitleg over het maken van een programma aan de hand van een voorbeeldprogramma dat telkens een stukje wordt uitgebreid. Het gebruik van variabelen, FOR en WHILE lus worden uitgelegd. Let op: in deze tutorial wordt een Console applicatie gemaakt, dit is iets anders dan een Form applicatie.

Webbrowser tutorial

Zelf een webbrowser maken (Nederlandstalig)[4]

Bruikbaarheid: *

Toelichting: Tutorial waarin een webbrowser gebouwd wordt. Weinig toelichting op wat er gebeurt maar wel een leuk eindresultaat. Deze tutorial is vanaf lesweek 3 redelijk goed te maken.

Blackwasp tutorial

BlackWasp[5]

Bruikbaarheid: ***

Toelichting: Verzameling tutorials en artikelen (Engelstalig).

1.6 Online C-sharp boeken

C# Station Tutorial (Engelstalig)[6] Bruikbaarheid: *** Toelichting: Uitleg over de basis onderdelen van C# zoals expressies, typen, variabelen en controlestructuren. De ”lessons” 1 t/m 5 zijn interessant voor OIS.

[2] http://msdn.microsoft.com/en-us/vcsharp/aa336766.aspx
[3] http://www.techzine.nl/tutorials/358/3/c-les-1-beginnen-met-c-de-eerste-stapjes.html
[4] http://www.sitemasters.be/tutorials/17/1/564/CSharp.NET/CSharp_Zelf_een_WebBrowser_maken
[5] http://www.blackwasp.co.uk/
[6] http://csharp-station.com/

C# Yellow Book (Engelstalig)[7] Bruikbaarheid: **

Toelichting: Compleet boek over programmer constructies en onderwerpen. Wellicht handig als naslagwerk, niet geschikt als leerboek voor OIS. Let op: in deze tutorial wordt uitgegaan van een Console applicatie als beginpunt, dit is iets anders dan een Form applicatie.

[7] http://www.robmiles.com/c-yellow-book/Rob%20Miles%20CSharp%20Yellow%20Book%202010.pdf