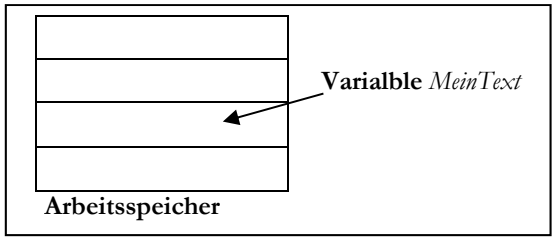
2.2 Variablen

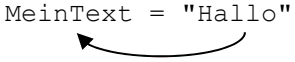


Variablen sind eines der wichtigsten Hilfsmittel eines Programmierers. Immer dann, wenn Sie sich in Ihrem Programm einen Wert merken müssen, verwenden Sie eine Variable. Eine Variable ist im Prinzip ein Name für ein Stück Arbeitsspeicher des Rechners, in dem Sie sich einen beliebigen passenden Wert merken und in einer späteren Anweisung in Ihrem Programm weiterverwenden können. Jede Variable besitzt einen **Datentyp**, der angibt, welche Art von Information gespeichert werden soll (z.B.: Text oder Zahlen). Dies ist ganz klar, da sie ja für einen langen Text viel mehr Arbeitsspeicher benötigen als für eine kurze Zahl.
Variablen müssen vor ihrer Verwendung vereinbart (=deklariert), d.h. bekannt gemacht werden, damit der Compiler den benötigten Speicherplatz reserviert. Dies geschieht in C# durch Angabe der Datentyps gefolgt vom Namen der Variable.

Bsp. 4: *Einen Wert in einer Variablen speichern*

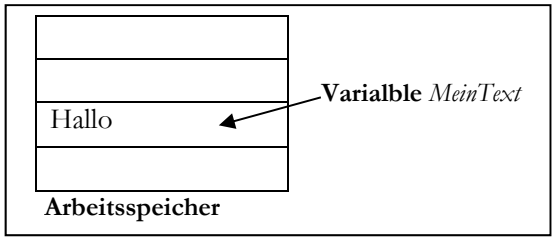
```
static void Main(string[] args)
{ string MeinText;
  MeinText="Hallo";
}
```

Irgendwo im Arbeitsspeicher wird mit dem Befehl `string MeinText` der Platz für eine String-Variable mit dem Namen *MeinText* reserviert.
String- Variable können Zeichenketten (also mehrere Buchstaben, Ziffern und Symbole) speichern.
Mit dem nächsten Befehl wird der Variable *MeinText* der Wert *Hallo* zugewiesen.
Da es in dem Beispiel keine Ausgabe gibt können wir dies aber natürlich nicht sehen.
Wichtig in diesem Zusammenhang ist, dass die Wertzuweisung **immer von rechts nach links** erfolgen. Es wird also der Text "Hallo" in der Variablen MeinText gespeichert und nicht ungekehrt.



Bsp. 5: *Einen Wert in einer Variablen ausgeben*

```
static void Main(string[] args)
{string MeinText;
 MeinText="Hallo";
 Console.WriteLine("MeinText");
 Console.WriteLine("{0}",MeinText);
 Console.ReadLine();
}
```



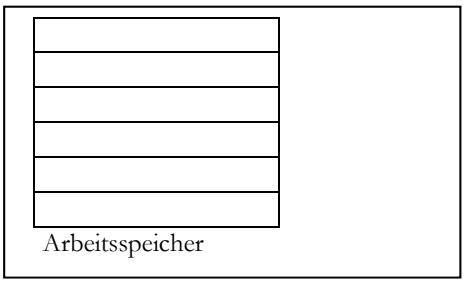
Wenn wir das Programm starten, so erkennen wir, dass das erste `WriteLine` nur *MeinText* auf den Bildschirm schreibt, da wir ja einfach in Anführungszeichen eine Zeichenkette ausgeben.
Das zweite `WriteLine` gibt den Wert der Variablen, also *Hallo* aus.

```
MeinText
Hallo
```

Bei Variablennamen ist die Groß- und Kleinschreibung wichtig, so sind *Name* und *name* zwei unterschiedliche Variablen.

Die Variablennamen sollten aussagekräftig sein. Meist wird zur besseren Lesbarkeit Groß- und Kleinschreibung gemischt z.B. *einSehrLangerAbereinSehrGutLesbarerVariablenname.*
Damit man aus den Variablennamen sofort den Typ erkennen kann wird in manchen Projekten auch die so genannte *Ungarische Notation* verwendet. Eine Variable vom Typ *string* die Namen speichert könnte dann zum Beispiel *strNamen* heißen.

Ü 2: *Wertzuweisungen. Überlegen Sie wie die Ausgabe des folgenden Programms lautet und zeichnen Sie die Werte in das Arbeitsspeicherdiagramm ein.*




```
static void Main(string[] args)
{ string name1, name2;
  name1 = "Franzi";
  name2 = "Hansi";
  name1 = name2;
  Console.WriteLine("{0}\n{1}",name1,name2);
  Console.ReadLine();
}
```

Bsp. 6: *Schreiben Sie ein Programm, das den Benutzer begrüßt*

```
Wie heißt du? Thorsten
Hallo Thorsten
```

```
static void Main(string[] args)
{ string name;
  Console.Write("Wie heißt du? ");
  name = Console.ReadLine();
  Console.WriteLine("Hallo {0}",name);
  Console.ReadLine();
}
```

`Console.ReadLine()` liest eine Zeichenkette von der Tastatur ein. In diesem Beispiel wird die Eingabe des Benutzers in der Variable *Name* zwischengespeichert.
Die Zuweisung erfolgt wiederum von rechts nach links: `Name = Console.ReadLine();`

`Console.Write` schreibt einen Text ohne in die nächste Zeile zu springen.

Ü 3: *Schreiben Sie ein Programm, das den Benutzer auffordert, seinen Vor und Nachnamen einzugeben und ihn anschließend auf den Bildschirm begrüßt.*
Hinweis: Sie benötigen 2 Variablen, da Sie ja 2 Werte speichern wollen.

Wenn sich unser Programm ganze Zahlen merken soll, so müssen wir den Variablentyp **Integer** verwenden.
Eine Übersicht über alle Variablen Typen finden Sie in 10.1 auf der Seite 197.
Integer Werte müssen nach dem Einlesen mit `Console.ReadLine()` von einem String in einen Integer Typ umgewandelt werden, da Console.Readline immer eine Variable vom Typ String ergibt.
Dies geschieht mit `Convert.ToInt32()`. Der in den Klammern stehende Ausdruck wird konvertiert.

Bsp. 7: *Eingabe einer Zahl*

```
static void Main(string[] args)
{  int i;
   Console.Write("Geben Sie eine ganze Zahl ein");
   i = Convert.ToInt32(Console.ReadLine());
}
```

Bsp. 8: *Eingabe einer Zahl, Ausgabe des doppelten der Zahl*

```
static void Main(string[] args)
{ int eingabe, ergebnis;
  Console.Write("Zahl ?");
  eingabe = Convert.ToInt32(Console.ReadLine());
  ergebnis = eingabe * 2;
  Console.WriteLine("Das Doppelte von {0} ist {1}", eingabe, ergebnis);
  Console.ReadLine();
}
```

Ü 4: *Schlagen Sie auf der Seite 197 die weiteren Datentypen nach. Überlegen Sie bei jedem Typ in welcher Situation jeder einsetzbar ist. Wie würden die Convert – Funktionen lauten?*