

Datentypen

Array	“Massenvariable” eines bestimmten Typs
<p>Wichtige Befehle:</p> <p>Definition: <code>int my_arr[5] = {2, 3, 8, -1, 3};</code> Zugriff: <code>my_arr[2] = 8 * my_arr[3];</code> (siehe auch <code>[]</code>-Operator)</p> <p>(Anstatt <code>int</code> gehen natürlich auch andere Typen.) (Die Definition kann auch ohne Initialisierung erfolgen: <code>int my_arr[5];</code>)</p> <p>Die Indizes beginnen bei 0.</p> <p>Der Programmierer muss selber sicherstellen, dass die Indizes nicht über den Array hinausgehen.</p> <p>Zuweisungen (ausser Initialisierung), Vergleiche, etc. müssen elementweise erfolgen.</p> <p>Die Länge des Arrays muss zum Kompilierzeitpunkt eindeutig bestimmbar sein. (z.B. Literal oder <code>const</code>-Variable, die mittels Literal eingelesen wurde, etc.)</p>	
<pre>float a[10]; for (int i = 0; i < 10; ++i) a[i] = i; // a becomes {0 1 2 ... 9} float b[10] = {2, 3, 4, 5, 6, 7, 8, 9, 10, 11}; a = b; // NOT valid: array-copying is forbidden // (should copy element-wise).</pre>	

Programmier-Befehle - Woche 07

Vektoren	komfortabler Array eines bestimmten Typs
<p>Erfordert: <code>#include <vector></code></p> <p>Wichtige Befehle:</p> <p>Definition: <code>std::vector<int> my_vec (length, init_value);</code> Zugriff: (wie Array) neues Element hinten: <code>my_vec.push_back(5)</code></p> <p>(Anstatt <code>int</code> gehen natürlich auch andere Typen.) (Die Definition kann auch nur mit Längenangabe erfolgen: <code>std::vector<int> my_vec (length)</code>)</p> <p>Unterschied zu Arrays: Die Länge des Vektors muss NICHT zum Kompilierzeitpunkt eindeutig bestimmbar sein. Ausserdem besitzen Vektoren "Komfortfunktionen" wie beispielsweise <code>push_back()</code>.</p>	
<pre>int len; std::cin >> len; // Assume: len > 2 std::vector<int> my_vec (len, 0); // my_vec: 0, 0, 0, ..., 0 my_vec[1] = 3; // my_vec: 0, 3, 0, ..., 0 my_vec.push_back(5); // my_vec: 0, 3, 0, ..., 0, 5 // [length increases by 1]</pre>	

char	Datentyp für Zeichen
<p>Literal: <code>'a'</code> für Zeichen (<i>einfache</i> Anführungszeichen) Literal: <code>"Hello World"</code> für Strings (<i>doppelte</i> Anführungszeichen)</p> <p>Ein String-Literal wird als Array des Typs <code>char</code> mit passender Länge gespeichert. Es wird immer mit <code>'\0'</code> terminiert (hat also ein Zeichen mehr als "Buchstaben").</p>	
<pre>char a[] = "my text"; // "my text" generates: my text\0 a[3] = '5'; // Changes a to: "my 5ext" // output for (int i = 0; a[i] != '\0'; ++i) std::cout << a[i] << "\n"; // Note: '\0' marks the end of the string.</pre>	

Operatoren

<code>my_array[...]</code>	Array- und Vektor-Zugriff (Subskript-Operator)
Präzedenz: 17 und Assoziativität: links	
<pre>int a[] = {1, 2, 3, 4}; a[3] = 5; // a is 1, 2, 3, 5</pre>	