

Datentypen

<code>std::stringstream</code>	Datentyp für String-Streams
<p>Erfordert: <code>#include <sstream></code></p> <p>Man kann Objekte dieses Typs beispielsweise verwenden, um dem Programm eine Eingabe via <code>std::cin</code> vorzutauschen.</p> <p>Beachte: “<i>String-Stream</i>” heisst, dass im Objekt ein String <i>enthalten</i> ist. Es heisst aber nicht, dass nur Strings (oder chars) daraus <i>extrahiert</i> werden können. Darin können beispielsweise auch Zahlen als String vorliegen. Diese kann man als String (oder char) oder als Zahl extrahieren.</p> <p>Objekte des Typs <code>std::stringstream</code> können nicht direkt kopiert werden. Deshalb sollte man sie immer via Call-by-Reference an Funktionen übergeben.</p>	
<pre>char c; std::cin >> c; // e.g. value: 'b' std::stringstream s ("b345e"); // stringstream with value "b345e" s >> c; // c gets value 'b' and s is now "345e" s >> c; // c gets value '3' (!as char! since type of c is char) int n; s >> n; // n gets value 45 (!as int! since type of n is int) // (This works since the computer sees that the next // 2 characters in the string "45e", namely '4' and '5', // can be used as the int 45. So after this operation // s is "e".)</pre>	

<code>std::istream</code>	Datentyp für Input-Streams
<p>Erfordert: <code>#include <istream></code> oder <code>#include <iostream></code></p> <p>Beispielsweise <code>std::cin</code> hat den Typ <code>std::istream</code>. Objekte des Typs <code>std::stringstream</code> können auch als <code>std::istream</code> verwendet werden (siehe <code>calculator.cpp</code>).</p> <p>Objekte des Typs <code>std::istream</code> können nicht direkt kopiert werden. Deshalb sollte man sie immer via Call-by-Reference an Funktionen übergeben.</p>	

(...)

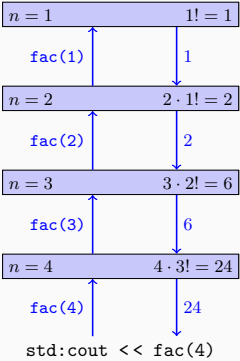
Programmier-Befehle - Woche 10

(...)

```
// POST: Two characters are removed from is. If is contains less
//       characters it is emptied.
void remove_two (std::istream& is) {
    char a;
    is >> a >> a; // remove two chars
}

int main () {
    // Assume that the user enters "Informatics".
    remove_two(std::cin);
    char out;
    while (std::cin >> out)
        std::cout << out; // Output: formatics
    return 0;
}
```

Funktionen

Rekursion	Selbstaufzuruf einer Funktion
<p>Jeder rekursive Funktionsaufruf hat seine eigenen, unabhängigen Variablen und Argumente. Dies kann man sich sehr gut anhand des in der Vorlesung gezeigten Stacks vorstellen (<code>fac</code> ist im Beispiel unten definiert):</p>	
 <p>Das Diagramm zeigt den rekursiven Aufruf der Fakultätsfunktion <code>fac</code> für <code>n=4</code>. Es besteht aus vier horizontalen Ebenen, die von unten nach oben absteigend beschriftet sind: <code>n=4</code>, <code>n=3</code>, <code>n=2</code> und <code>n=1</code>. Jede Ebene enthält die Berechnung <code>n * (n-1)!</code>. Aufsteigende Pfeile (Rückgabe) zeigen die Werte 1, 2, 6 und 24 von den unteren Ebenen nach oben an. Absteigende Pfeile (Parameterübergabe) zeigen die Werte 1, 2, 6 und 24 von den oberen Ebenen nach unten an. Am unteren Ende des Diagramms steht <code>std::cout << fac(4)</code>.</p>	
<pre>// POST: return value is n! unsigned int fac (const unsigned int n) { if (n <= 1) return 1; return n * fac(n-1); // n > 1 }</pre>	

Input/Output

<code>my_stream.eof()</code>	Prüfe, ob das Ende des Streams erreicht worden ist.
<p>Erfordert: <code>#include <iostream></code></p> <p>Ein Stream kann sich in verschiedenen Zuständen befinden. Und es gibt Funktionen, um den aktuellen Zustand abzufragen. <code>eof()</code> ist eine solche Funktion. Mit ihr prüft man, ob sich der Stream im Zustand "Ende der Eingabe erreicht" befindet.</p> <p><i>EoF</i> bedeutet "End-of-File".</p>	
<pre>std::stringstream b ("33"); int z; b >> z; std::cout << b.eof(); // true (>> read 33 and reached // end of b)</pre>	

<code>my_stream.fail()</code>	Prüfe, ob im Stream ein ungültiges Zeichen hätte gelesen werden sollen.
<p>Erfordert: <code>#include <iostream></code></p> <p>Ein Stream kann sich in verschiedenen Zuständen befinden. Und es gibt Funktionen, um den aktuellen Zustand abzufragen. <code>fail()</code> ist eine solche Funktion. Mit ihr prüft man, ob sich der Stream im Zustand "gescheiterte Eingabe" befindet.</p>	
<pre>std::stringstream b ("33a"); int z; b >> z; std::cout << b.fail(); // false (>> read 33) b >> z; std::cout << b.fail(); // true (>> ought to read 'a' // which is not an int.)</pre>	

<code>my_stream.peek()</code>	Im Stream nächstes Zeichen anschauen, ohne es zu entfernen.
<p>Erfordert: <code>#include <istream></code> oder <code>#include <iostream></code></p> <p>Der Rückgabewert ist die int-Repräsentierung des nächsten Zeichens (als char) im Stream. Der Datentyp des Rückgabewerts ist also int.</p> <p>Diese Funktion ignoriert Whitespaces nie (unabhängig davon, ob der Stream zuerst an <code>std::noskipws</code> übergeben wurde oder nicht).</p>	
<pre>std::stringstream str ("my subst"); str >> std::noskipws; // Do not ignore whitespaces. char c; // output everything before the first 's' (but leave 's' in str) while (str.peek() != 's') // Removes: "my " str >> c; // output remaining string while (str >> c) // Output: "subst" std::cout << c; // miscellaneous std::stringstream num ("3 a"); std::cout << num.peek() << "\n"; // Output: 51 and NOT '3' num >> c; std::cout << c << "\n"; // Output: '3' char next = num.peek(); std::cout << next << "\n"; // Output: ' ' num >> c; std::cout << c << "\n"; // Output: 'a'</pre>	

<code>std::ws</code>	Entfernt Whitespaces am Anfang eines Input-Streams.
Erfordert: <code>#include <iostream></code> oder <code>#include <iostream></code>	
Unter einem <i>Whitespace</i> versteht man <i>nicht-darstellbare</i> Zeichen wie zum Beispiel Leerschläge, <i>Zeilenumbrüche</i> , etc. Dem Computer werden solche Zeichen als eine Sequenz von anderen Zeichen übergeben, welche er dann interpretiert (z.B. " <code>\n</code> " interpretiert er als einen <i>Zeilenwechsel</i>).	
<pre>char c; std::stringstream t ("d a\n\nb"); t >> std::noskipws; // Do not ignore whitespaces. // Output t without whitespaces t >> c; std::cout << c; // Output: 'd' t >> std::ws; // Remove: " " t >> c; std::cout << c; // Output: 'a' t >> std::ws; // Remove: "\n\n" t >> c; std::cout << c; // Output: 'b' // Output in total: dab</pre>	