Ein/Ausgabe

C: Die Ein/Ausgabe erfolgt durch Standardfunktionen in `<stdio.h>`: `scanf` und `printf`. Die Anzahl der Argumente ist veränderlich, das erste Argument ist aber jeweils eine Formatzeichenkette, die angibt, wie die weiteren Argumente zu interpretieren sind.

```
scanf("%d%lf",&i,&x);       // i und x einlesen (Zeiger!)
printf("i=%d x=%f\n",i,x)   // Ausg. z.B. i=4 x=0.123456
```

C++: Hauptsächlich wird die typsichere C++-Ausgabe mit überladenen Shiftoperatoren benutzt.

```
cin >> i >> x;
cout << "i=" << i << " x=" << x << endl;
```

Java: Wie bei C++ gibt es Stromobjekte für die Ein/Ausgabe. Sie können durch Methoden angesprochen werden, die sich insbesondere bei der Ausgabe an den C-Funktionen orientieren.

```
Scanner scannner = new Scanner(System.in);
i=scanner.nextInt(); x=scanner.nextDouble();
System.out.printf("i=%d x=%f%n",i,x);
```

Vektoren

C: Die Länge von C-Vektoren wird zur Übersetzungszeit festgelegt, ihre Länge ist unveränderbar. Dynamische Allokation ist mit Hilfe geeigneter Funktionen möglich.

```
double x[N];                // C-Vektor, N Konstante (Uebersetzungszeit)
double *xp = malloc(n*sizeof(double))
                            // Dynam. Alloz. eines C-Vektors mit n Komp.
xp[n-1]=x[0];               // Wertzuweisung
```

Die Länge des dynamischen C-Vektors ist nicht abfragbar.

C++: Vorhanden sind die aus C bekannten Vektoren („C-Vektoren“) und die dynamischen Vektoren aus der Standard Template Library („STL-Vektoren“). Letztere unterliegen im Unterschied zu C der bei Standardklassen üblichen Wertsemantik.

```
double *xp = new double[n]; // Dynam. Alloz. eines C-Vektors mit n Komp.
vector<double> x(n);        // C++-Vektor mit n Komp.
```

Die Länge des STL-Vektors ist über die Komponentenfunktion `size` abfragbar.

Java: Vektoren können als Klassen angesehen werden, es gilt Referenzsemantik.

```
double x[] = new double[n]  // Vektorvereinbarung
double[] x = new double[n]  // Vektorvereinbarung (bedeutungsgleich)
```

Die Länge des Vektors ist über die Komponente `length` abfragbar.

Zeichenketten (`char`-basiert)

C: C-Zeichenketten sind `'\0'`-terminierte C-Vektoren aus 8-Bit-Zeichen (`char`). Ihre Verarbeitung erfolgt durch Standardfunktionen `strlen,strcpy,strcat,...`), die *keinen* Speicherplatz allozieren.

C++: Die Standard Template Library (STL) stellt zusätzlich die bequemer handhabare Klasse `string` bereit. Diese Zeichenketten basieren auch auf dem Datentyp `char`, sind aber nicht nullterminiert und ihre Länge ist veränderbar.

Java: Die Klasse `String` basiert auf 16-Bit-Zeichen (ebenfalls `char` genannt). Ihre Objekte sind Konstanten (“immutable“), die bei ihrer Erzeugung festgelegt werden.

Fehlerbehandlung

C: In der imperativen Programmierung werden Fehler durch Rückgabewerte von Funktionen, Referenzparameter (bzw. in C Zeiger) oder globale Fehlervariablen (errno) mitgeteilt.

C++: Zusätzlich existiert ein Exception-Mechanismus zur Behandlung von Fehlern, die in Klassen auftreten.

Java: In der Regel werden Fehler mit Hilfe des aus C++ übernommmenen Exception-Mechanismus behandelt. Allerdings unterscheiden sich die Exception-Klassen erheblich.

Dreiecksflächenberechnung mittels der Heronformel

- C

```
#include <stdio.h>
#include <math.h>

int main(void)
{
  double a,b,c,s,F;
  printf("a b c: "); scanf("%lf%lf%lf",&a,&b,&c);
  s = (a+b+c)/2.0; F = sqrt(s*(s-a)*(s-b)*(s-c));
  printf("F = %8.4f\n",F);
  return 0;
}
```

- C++

```
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;

int main()
{
  double a,b,c,s,F;
  cout << "a b c: "; cin >> a >> b >> c;
  s = (a+b+c)/2.0; F = sqrt(s*(s-a)*(s-b)*(s-c));
  cout << "F = " << fixed << setprecision(4) << setw(8) << F << endl;
  return 0;
}
```

- Java

```
import java.util.*;

public class Heron {
  public static void main(String[] args)
  {
    double a,b,c,s,F;
    Scanner scanner = new Scanner(System.in);
    System.out.print("a b c: ");
    a = scanner.nextDouble(); b = scanner.nextDouble(); c = scanner.nextDouble();
    s = (a+b+c)/2.0; F = Math.sqrt(s*(s-a)*(s-b)*(s-c));
    System.out.printf("F = %8.4f%n",F);
  }
}
```

Dateikopie

- C

```c
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>

int main(void)
{
  char ifname[1024],ofname[1024]; // Dateipfad max. 1024 Zeichen inkl \0
  FILE *ifp,*ofp;
  char c;
  printf("Eingabedatei: "); scanf("%s",ifname);
  if ( (ifp=fopen(ifname,"rb")) == NULL ) { perror(ifname); exit(1); }
  printf("Ausgabedatei: "); scanf("%s",ofname);
  if ( (ofp=fopen(ofname,"wb")) == NULL ) { perror(ofname); exit(1); }
  while( fscanf(ifp,"%c",&c) != EOF ) fprintf(ofp,"%c",c);
  return 0;
}
```

- C++

```cpp
#include <iostream>
#include <cmath>
#include <cerrno>
#include <cstdlib>
#include <string>
#include <cstring>
#include <fstream>

using namespace std;

int main()
{
  string ifname,ofname;
  char c;
  cout << "Eingabedatei: "; cin >> ifname;
  ifstream is(ifname.c_str(),ios::binary);
  if (!is) { cerr << ifname << ": " << strerror(errno) << endl; exit(1);}
  cout << "Ausgabedatei: "; cin >> ofname;
  ofstream os(ofname.c_str(),ios::binary);
  if (!os) { cerr << ofname << ": " << strerror(errno) << endl; exit(1);}
  is >> noskipws;
  while(is >> c)
    os << c;
  return 0;
}
```

- Java

```
import java.io.*;
import java.util.*;

public class Copy {
  public static void main(String[] args)
  { // Block der main-Methode
    try{
      String ifname,ofname;
      int c;
      Scanner scanner = new Scanner(System.in);
      System.out.print("Eingabedatei: "); ifname = scanner.next();
      FileInputStream is = new FileInputStream(ifname);
      System.out.print("Ausgabedatei: "); ofname = scanner.next();
      FileOutputStream os = new FileOutputStream(ofname);
      while( (c=is.read()) != -1) os.write(c);
      is.close(); os.close();
    }
    catch (IOException e) { System.out.println(e); }
  }
}
```