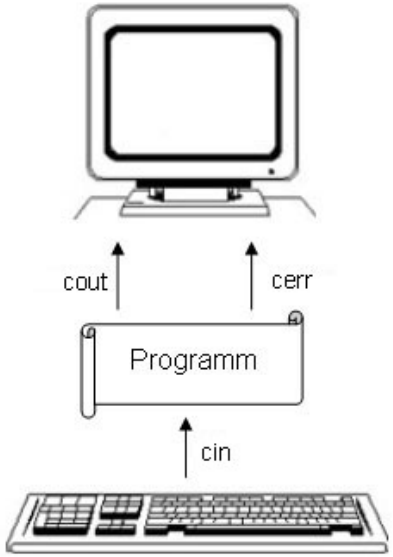
C++ - Einführung in die Programmiersprache

„Ein- und Ausgabe in der Konsole“



Konsolen

- Kommandozeile. Terminal. Shell.
- Pro „Zeile“ werden textbasierte Befehle zur Steuerung von Softwareprogrammen oder zur Navigation in Ordnern eingegeben.
- Das Betriebssystem Windows nutzt zum Beispiel die Microsoft Eingabeaufforderung oder PowerShell als Konsole.

Streams (Datenströme)

- Transport von Daten.
- „Leitung“ nach außen, um Daten zu lesen und zu schreiben.
- Input-Streams (Eingabe) lesen Daten von einer Quelle.
- Output-Streams (Ausgabe) schreiben Daten in eine Senke.

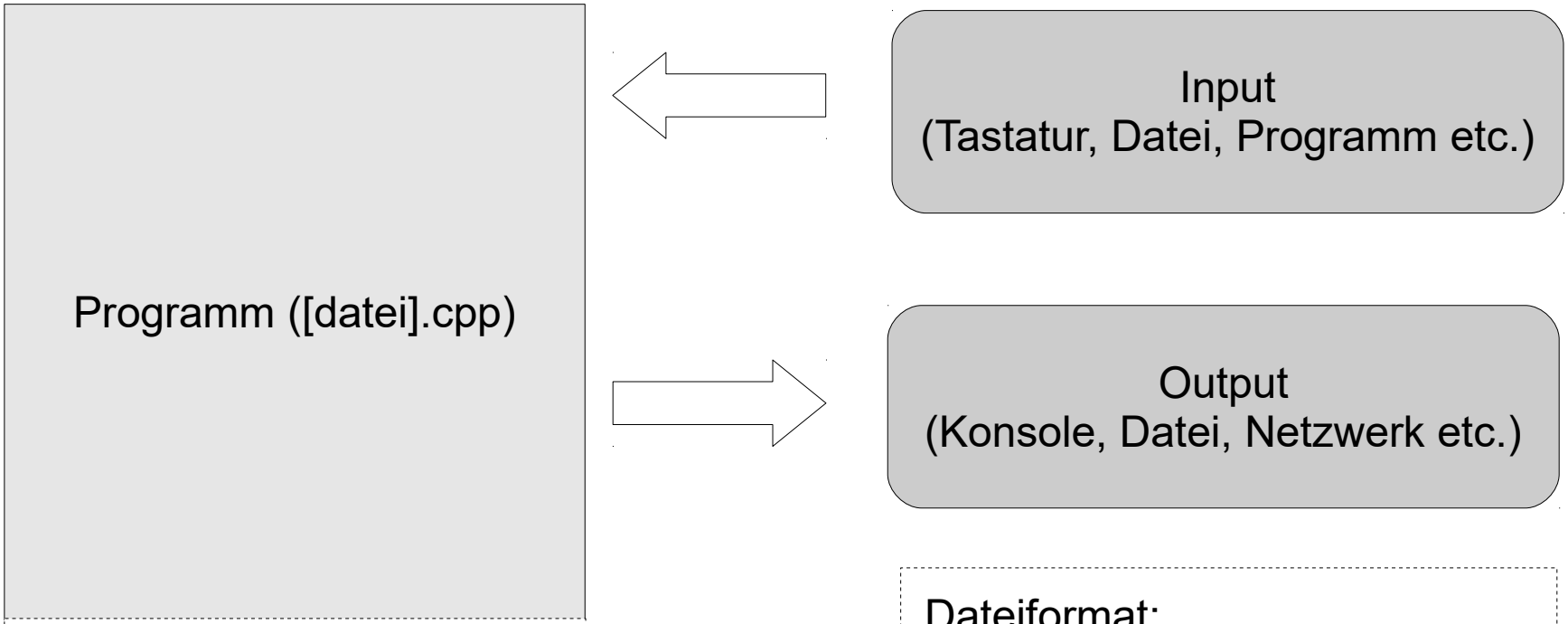
Zeichen- und byteorientierte Streams

- Zeichenorientierte Streams transportieren Buchstaben, Wörter oder Texte.
- Byteorientierte Streams werden für alle anderen Arten genutzt. Beispiel: Austausch von Grafiken zwischen verschiedenen Programmen.

Ein- und Ausgabe in C++

- Typsichere und erweiterbare Klassen und Funktionen in der Standard-Bibliothek.
- Basis der Ein- und Ausgabe: Byte-orientierte Streams. Verschiedene Geräte zur Ein- und Ausgabe nutzen die gleiche Funktionalität.

Grafische Darstellung



Dateiformat:
- `char, wchar_t, char32_t etc.`
- `int, float etc.`

Dateiformat:
- Text in verschiedenen Zeichenkodierungen (UTF8, ASCII etc.
- Binäres Format

Header-Dateien für die Ein- und Ausgabe

Header-Dateien

- C++-Dateien mit der Endung „.h“.
- Deklarationsdatei. Kopf einer Quelldatei.
- Funktionsdeklaration und so weiter.
- Definition von Schnittstellen nach außen.

Informationen im Web

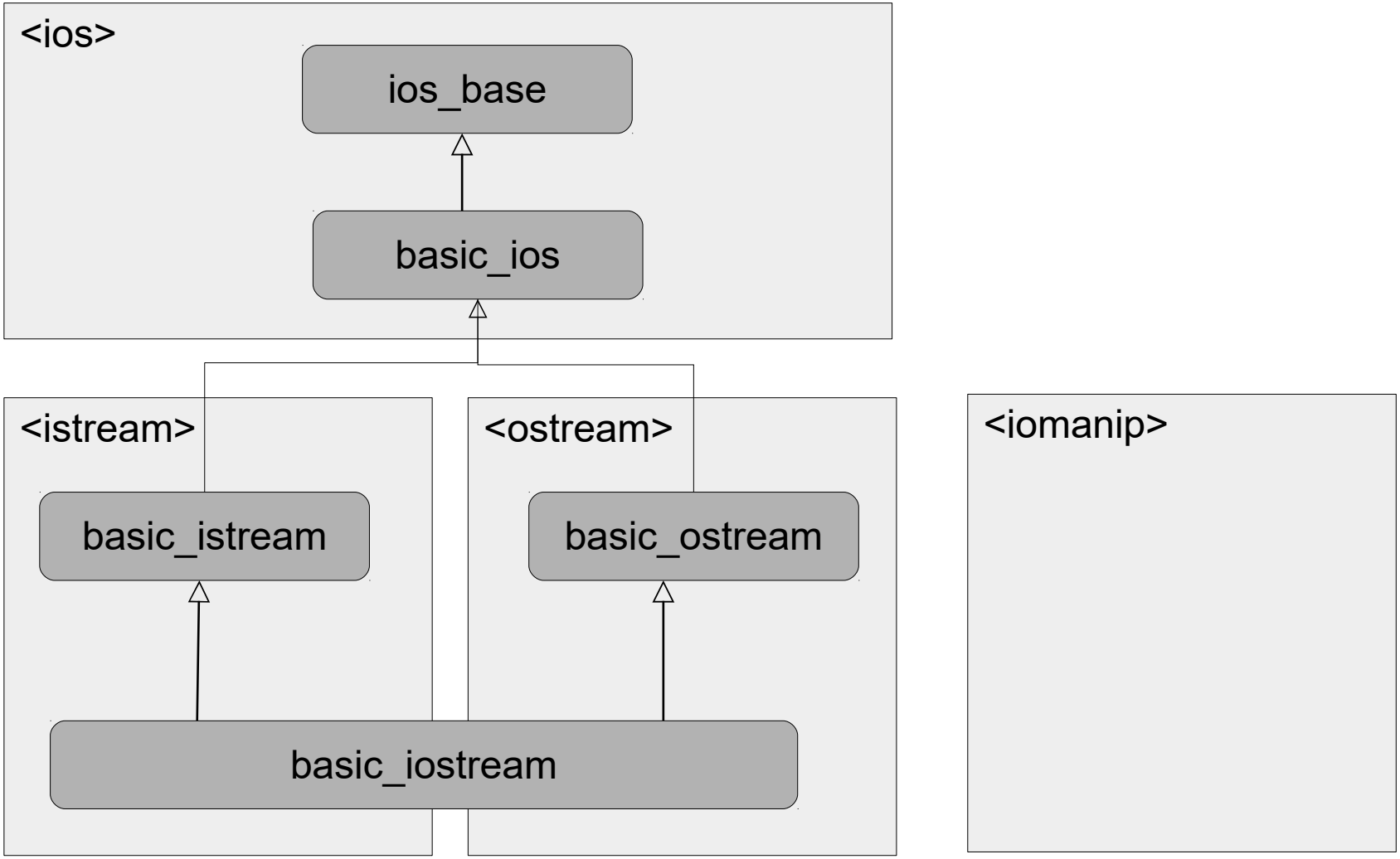
- https://www.ntu.edu.sg/home/ehchua/programming/cpp/cp10_IO.html
- https://en.cppreference.com/w/cpp/io
- https://en.cppreference.com/w/cpp/io/manip

… einbinden

```
#include <iostream>
```

- `#include` ist eine Präprozessor-Anweisung am Anfang einer Quelldatei.
- Spitze Klammerung des Dateinamens einer Standard-Headerdatei.
- Die Standard-Headerdateien werden automatisiert mit dem Compilier installiert.

Header <iostream>



Streams (Datenströme)

- Transport von Daten als Byte-Stream.
- Input-Streams (Eingabe). Von der Daten-Quelle (Tastatur, Datei etc.) zum Programm.
- Output-Streams. Vom Programm zu einer beliebigen Daten-Senke (Bildschirm, Drucker etc).

Standard-Streams

- Output-Stream: `cout` oder `wcout` für die Ausgabe auf die Konsole.
- Input-Stream: `cin` oder `wcin` für das Lesen von Tastatureingaben.
- Ausgabe von Fehlermeldungen auf die Konsole: `cerr` oder `wcerr`.
- Ausgabe von Log-Informationen auf die Konsole: `clog` oder `wclog`

Ausgabe auf die Konsole

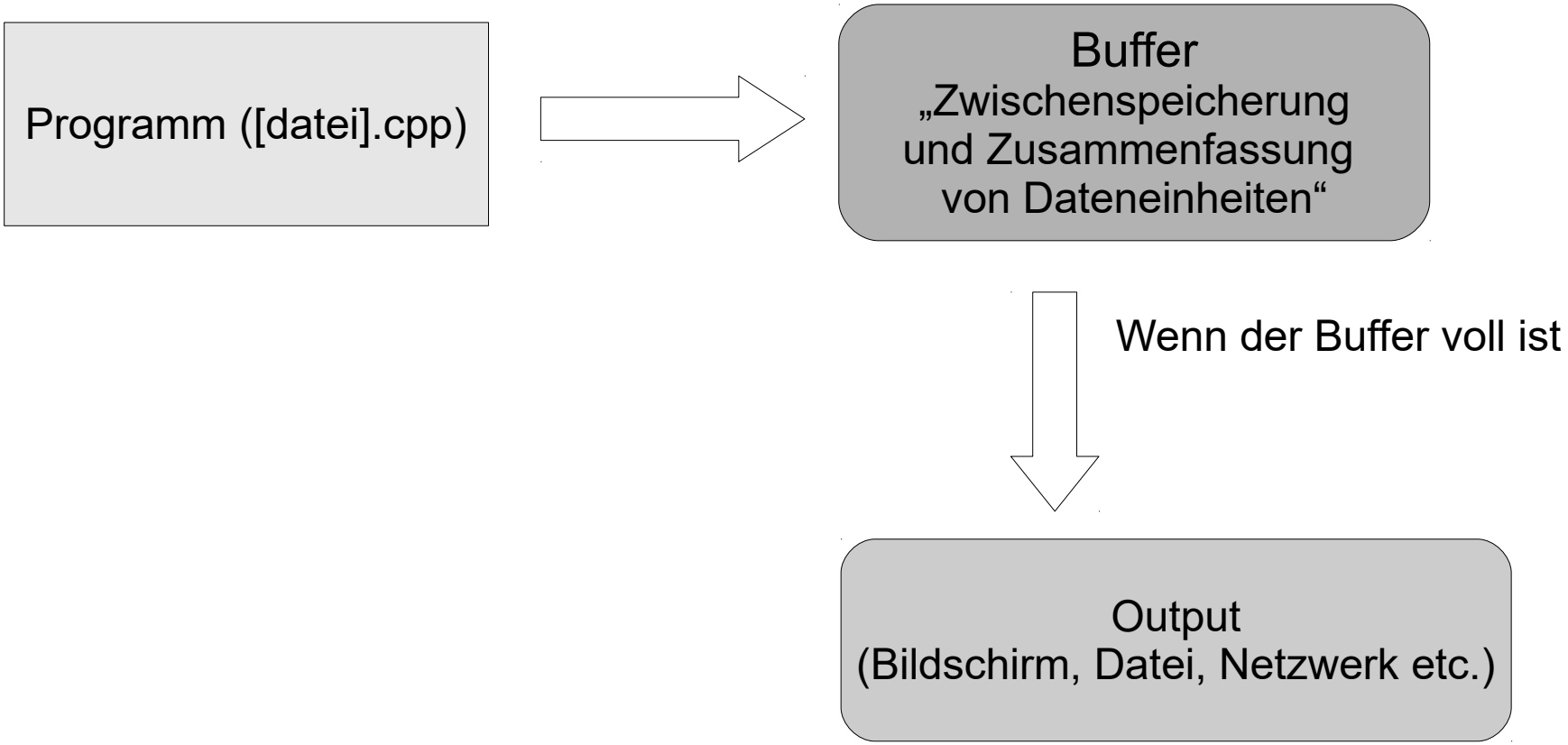
```
#include <iostream>

int main()
{
  std::cout << "Hello World!" << std::endl;
  return 0;
}
```

Voraussetzung

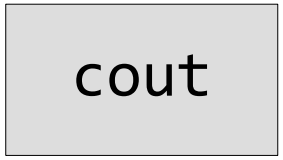
```
#include <iostream>
```

- Alle Elemente zum Lesen und Schreiben auf der Konsole sind in der Header-Datei `<iostream>` definiert.
- Keinerlei Elemente zur Ein- und Ausgabe im Sprachumfang von C++.

Grafische Darstellung

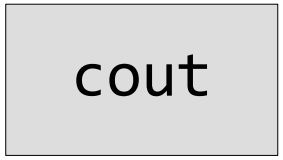


Ausgabe-Stream



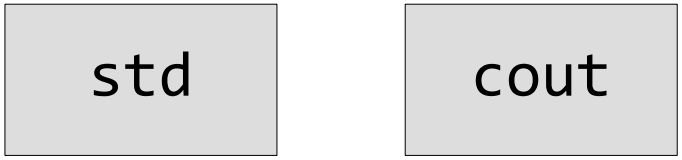
- `cout` beschreibt Eigenschaften und Methoden der Standardausgabe.
- Gepufferter Ausgabe-Stream auf die Konsole, in eine Datei etc.
- Ausgabe von Strings, Zeichen aus dem ASCII-Zeichensatz, Ganzzahlen und Gleitkommazahlen.

Gültigkeit des Namens



- Namen wie `cout` können in verschiedenen Programmen genutzt werden. Der Compiler weiß aber nicht, welcher Name aktuell gemeint ist.
- Um Verwechselungen zu vermeiden, werden die Elemente verschiedenen Namensräumen zugeordnet.
- Jeder Name ist in seinem Raum eindeutig.

Standard-Namensraum



- `std` bezeichnet den Standard-Namensraum in C++.
- Nutzung von Elemente, die in der Standard-Bibliothek deklariert sind.

Element x befindet sich im Namensraum y

`std`	`::`	`cout`

- Qualifizierungsoperator: `::`.
- Qualifizierung eines Elements mit seinem übergeordneten Element.
- Zuordnung eines Elements zu einem bestimmten Raum.
- Beispiel: `cout` ist in dem Namensraum `std` abgelegt.

… global für die Datei angeben

```
using namespace std;
```

- Nutze (`using`) den Namensraum im gesamten Programm.
- Alle Elemente aus dem Standard-Namensraum sind qualifiziert.

Fehler bei Nicht-Definition

- Wenn die passende Bibliothek eingebunden ist: „was not declared in this scope“.
- Wenn keine passende Bibliothek eingebunden ist: „is not a member of std“.

Beispiel

```
#include <iostream>

using namespace std;

int main()
{
  cout << "Hello World!" << endl;
  return 0;
}
```

Umleitung der gewünschten Ausgabe

```
std::cout << "Hello World!"
```

- Umleitung von Werten auf die Standard-Ausgabe.
- In diesem Beispiel wird der String „Hello World“ auf der Konsole ausgegeben.

Umleitungsoperator

```
std::cout | << | "Hello World!"
```

- Zusammengesetzter Operator. Zwei direkt aufeinanderfolgende Kleiner-Zeichen.
- Die Pfeilspitzen zeigen die Fließrichtung des Stroms an.
- Der Strom fließt auf die Standardausgabe.

Verkettung von „Umleitungen“

`std::cout`	`<<`	`"Hello World!"`	`<<`	`std::endl`

- Die Ausgabe wird in der entsprechenden Reihenfolge in den Puffer geschrieben.
- Beliebig viele Elemente können gemeinsam umgeleitet werden.

Ausgabe von Zahlen

```
double zahl = .5;
int hexazahl = 0X42;

std::cout << "\n Int: " << 5;
std::cout << "\n Hexadezimal: " << hexazahl;
std::cout << "\n Double: " << zahl;
std::cout << "\nExponential: " << 3.25e2;
std::cout << std::endl;
```

- Zahlen werden entsprechend des Dezimalzahlen-Systems ausgegeben.

Ausgabe von boolschen Werten

```
bool aus = false;
bool an = true;

cout << "\n Wahr: " << an;
cout << "\n Falsch: " << aus;
```

- Boolsche Werte werden als Ganzzahlen ausgegeben.
- Falsch, False wird als Zahl 0 ausgegeben.
- Wahr, True wird als Zahl 1 dargestellt.

Ausgabe des Datentyps „char“

```
std::cout << '\n' <<  'p' << '*'
          << 'a' << '=' << '\u20AC';
```

- In welchen Zeichensatz ist das Zeichen im Programm codiert?
- Welche Codepage für Zeichen nutzt die Konsole?
- Kann die Schriftfamilie der Konsole die Zeichen darstellen? Zum Beispiel kann die Rasterschriftart der MS DOS Eingabeaufforderung keine Umlaute und so weiter darstellen.

Codepage

- Zeichensatztabelle einer Konsole.
- Zum Beispiel nutzt die MS DOS Eingabeaufforderung standardmäßig die Codepage 850.

… im Web

- https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/chcp
- https://en.wikipedia.org/wiki/Windows_code_page.

… ändern

```
#include <iostream>
#include <windows.h>

int main()
{

    SetConsoleOutputCP(65001);
    SetConsoleCP(65001);
```

Einbinden der Header-Datei

```
#include <windows.h>
```

- Für Dateien aus der C-Bibliothek muss die Dateiendung „.h“ angegeben werden.
- Windows32-API.
- Elemente speziell für das Betriebssystem Windows.

Befehle

```
SetConsoleOutputCP(65001);
SetConsoleCP(65001);
```

- Die Funktion `SetConsoleOutputCP` ändert die Codepage für die Ausgabe.
- Die Funktion `SetConsoleCP` ändert die Codepage für die Eingabe.
- In den runden Klammern wird die gewünschte Codepage übergeben.

Weitere Möglichkeit

```
#include <iostream>

int main()
{

    system("chcp 1252");
```

Erläuterung

- Die Funktion `system()` führt Befehle in Abhängigkeit des Betriebssystems aus.
- Der Funktion wird der Systembefehl in runden Klammern übergeben.
- `chcp 1252` ist ein Befehl zum Ändern der Codepage. Systembefehle werden immer als String (begrenzt durch die Anführungsstriche) an die Funktion `system()` übergeben.

Escape-Sequenzen

- Zeichen vom Datentyp `char`, die durch das Apostroph begrenzt werden.
- Steuerzeichen für den Drucker etc.
- Nicht druckbare Zeichen eines Zeichensatzes.
- Maskierung von Zeichen, die in C++ in einer besonderen Funktion genutzt werden.

Maskierung von Zeichen

```
cout << '\n';
```

- Mit Hilfe des Schrägstrichs wird ein Zeichen maskiert.
- Zeichen, die mit einem Schrägstrich beginnen, haben eine besondere Bedeutung für den Compiler und stellen ein Zeichen dar.

Maskierung von Steuerzeichen

\	b
\	f
\	n
\	r
\	t
\	v

Back space. Gehe einen Schritt zurück.
Form feed. Seitenumbruch.
New Line. Gehe zum Anfang einer neuen Zeile.
Carriage return. Gehe zum Anfang der aktuellen Zeile.
Horizontal Tabulator.
Vertical Tabulator.

Maskierung von weiteren Zeichen

\	"
\	'
\	\
\	?
\	0

Anführungszeichen.
Apostroph.
Umgekehrter Schrägstrich. Backslash.
Fragezeichen.
Endezeichen eines Strings.

Escape-Sequenz als Zeilenende

```
std::cout << "1. Zeile" << std::endl;
std::cout << "2. Zeile" << '\n';
```

- Die Escape-Sequenz `\n` definiert einen Zeilenumbruch. Die Zeile wird umgebrochen.
- Sobald der Ausgabepuffer hinreichend gefüllt ist, wird der Inhalt des Puffers auf die Ausgabe geschrieben.

Manipulator als Zeilenende

```
std::cout << "1. Zeile" << std::endl;

std::cout << "entspricht:\n";
std::cout.flush();
```

- Die Bibliothek `<iomanip>` sollte eingebunden werden.
- Der Manipulator `endl` definiert einen Zeilenumbruch. Die Zeile wird umgebrochen.
- Die Elemente im Ausgabepuffer werden sofort auf die Ausgabe geschrieben.

Ausgabe von mehreren Zeichen

```
std::cout << "\n Preis(p) * Stück(a) = Gesamt €";
```

- Eine Zeichenkette (String) wird durch die Anführungszeichen begrenzt.
- Ein String kann x beliebige alphanumerische und numerische Zeichen enthalten.
- Strings sind keine Standard-Datentypen in C++.

Löschen des Puffers

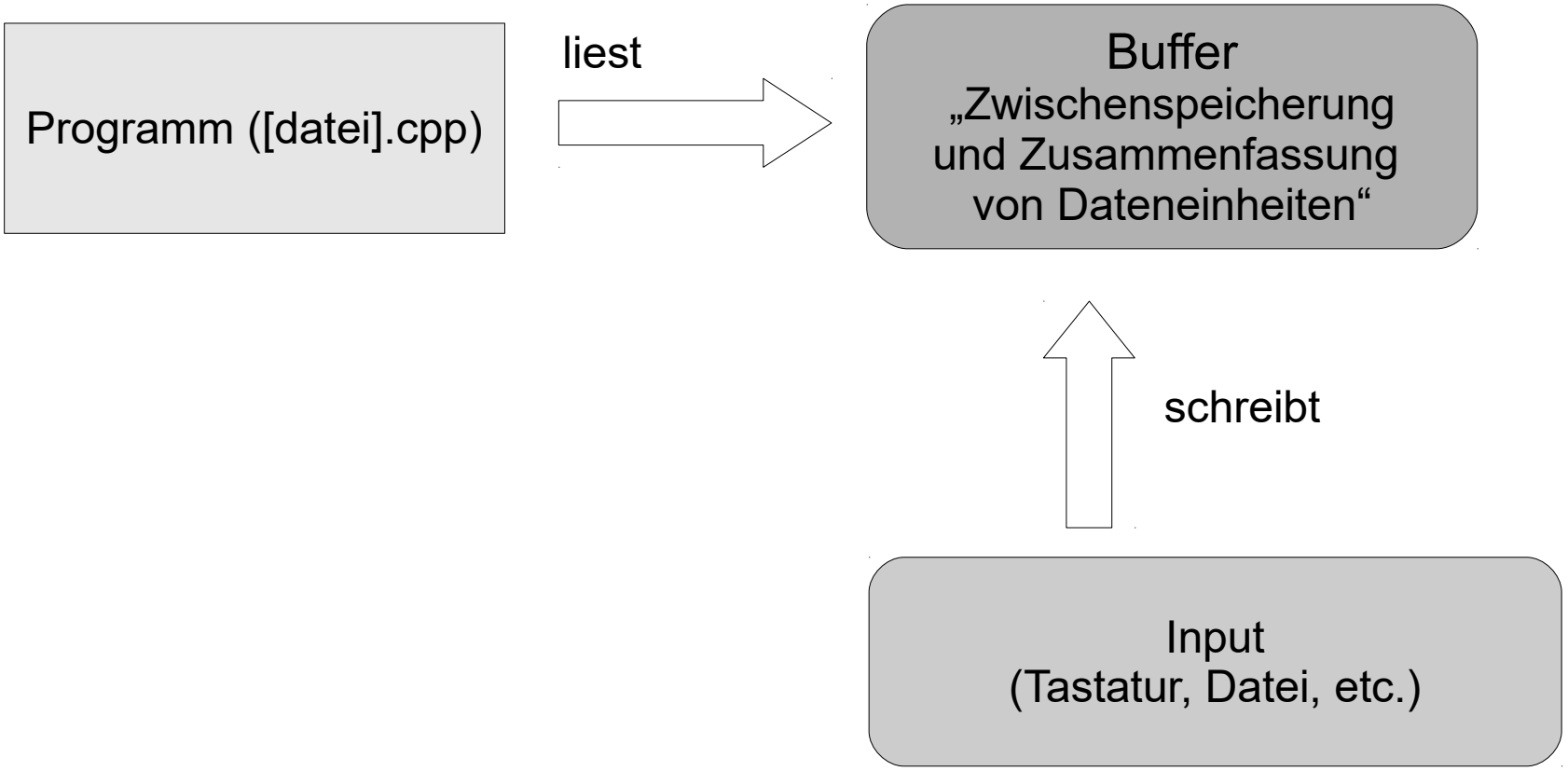
```
std::cout.flush();
```

- Das Objekt `cout` besitzt verschiedene Methoden, um seine Eigenschaften zu verändern.
- Mit Hilfe der Methode `.flush()` wird der Ausgabepuffer geleert. Die Elemente im Puffer werden direkt auf die Ausgabe geschrieben.
- Der Punktoperator verbindet die Methode mit seinem Objekt.
- Dem Namen der Methode `flush` folgen immer die runden Klammern. Die runden Klammern sind leer. Für die Ausführung werden keinerlei Parameter benötigt.

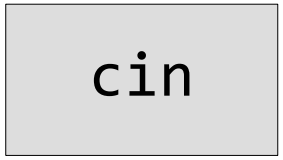
Nutzung der Standardeingabe

```
#include <iostream>

int main() {
    double gramm = 1.0;
    double kilo = 0.0;

    std::cout << "Wert in Gramm: ";
    std::cin >> gramm;
    kilo = gramm / 1000;
    std::cout << "\nIn Kilogramm: " << kilo;
    std::cout << std::endl;
    return 0;
}
```

Grafische Darstellung



Eingabe-Stream



- `cin` beschreibt Eigenschaften und Methoden der Standardeingabe.
- Mit Hilfe der Tastatur werden Zeichen in den Puffer geschrieben. Sobald die Eingabetaste gedrückt wird, wird die Eingabe beendet und das Programm kann diese lesen.

Funktion x befindet sich im Namensraum y

`std`	`::`	`cin`

- Die zwei direkt nacheinander geschriebenen Doppelpunkte stellen einen Operator dar.
- `cin` ist in dem Namensraum `std` abgelegt.
- Falls der Namensraum global angegeben ist, kann die Qualifizierung entfallen.

Umleitung der Eingabe

`std::cin`	`>>`	`gramm`

- Die Standard-Eingabe wird auf eine Variable umgeleitet.
- Die Eingabe wird entsprechend des Datentyps der Variablen interpretiert.
- Sobald das Zeichen nicht entsprechend des Datentyps interpretiert werden kann, wird der Lese-Vorgang beendet.

Umleitungsoperator

`std::cin`	`>>`	`gramm`

- Zusammengesetzter Operator. Zwei direkt aufeinanderfolgende Größer-Zeichen.
- Die Pfeilspitzen zeigen die Fließrichtung des Stroms an. Falls die Fließrichtung nicht korrekt angegeben ist, wird der Fehler „no match for operator“ angezeigt.
- Der Strom fließt in diesem Beispiel von der Standard-Eingabe in eine Variable.

Verkettung der Eingabe

`std::cin`	`>>`	`wert`	`>>`	`faktor`

- Für jede angegebene Variable muss der Benutzer einen Wert eingeben.
- Trennzeichen zwischen der Eingabe der verschiedenen Werte: Leerzeichen, Tabulator oder die Eingabetaste.

Umleitungsoperatoren und Bit-Operatoren

```
cout << "\n nach links: " << (intLWert << 2);
cout << "\n nach rechts: " << (intRWert >> 2);
```

- Der binäre Wert wird um x Positionen nach links oder rechts verschoben.

Linksverschiebung

```
cout << "\n nach links: " << (intLWert << 2);
```

- `intRWert` hat den Wert 2 = 0000 0010 binär.
- Multiplikation mit 2 bei jedem Schritt.

Rechtsverschiebung

```
cout << "\n nach rechts: " << (intRWert >> 2);
```

- `intRWert` hat den Wert 2 = 0000 0010 binär.
- Division durch 2.

Buffer löschen

```
#include <iostream>
#include <limits>

int main() {
  double faktor = 0.0;
  double wert = 0.0;

  std::cout << "Bitte geben Sie einen Wert: \n";
  std::cin >> wert;
  std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
                  '\n');
  std::cout << "Bitte geben Sie einen Faktor ein: \n";
  std::cin >> faktor;
  std::cout << (wert * faktor) << std::endl;
  return 0;
}
```

Objekt cin

```
std::cin.ignore(          );
```

- Das Objekt `cin` beschreibt den Standard-Eingabestrom.
- Objekte basieren auf Klassen. Klassen fassen Attribute und Methoden zu einem Objekt zusammen.
- Attribute beschreiben ein Objekt.
- Methoden verändern die Attribute eines Objekts. Aktionen, die auf einem bestimmten Objekt angewendet werden.

Methode „Buffer löschen“

```
cin.ignore(numeric_limits<streamsize>::max(), '\n');
```

- Mit Hilfe der Methode `.ignore()` werden x Zeichen aus dem Buffer entfernt.
- Die Zeichen im Buffer werden von der Eingabe ignoriert.
- Mit Hilfe des Punktoperators wird das Objekt mit der Methode verbunden.

Parameterliste der Methode

```
cin.ignore(numeric_limits<streamsize>::max(), '\n');
```

- Dem Namen der Methode `.ignore` folgen zwei runde Klammern.
- In diesen runden Klammern wird eine Parameterliste definiert.
- Die Parameter in der Liste werden durch ein Komma getrennt.
- Die Informationen werden beim Start der Funktion benötigt.

… der Methode „Buffer löschen“

```
cin.ignore(numeric_limits<streamsize>::max(), '\n');
```

- Der erste Parameter gibt die Anzahl der maximal zu löschenden Zeichen an. In diesem Beispiel wird die maximale Größe eines Puffers genutzt.
- Der zweite Parameter definiert ein Zeichen, dass das Ende der Löschung kennzeichnet. In diesem Beispiel wird das Zeilenende genutzt.

Voraussetzung zur Ermittlung der Größe

`numeric_limits`	`<`	`streamsize`	`>`	`::`	`max`	`(`	`)`
`template`	`<`	`datentyp`	`>`	`::`	`methode`	`(`	`)`

- Die Vorlage `numeric_limits` benötigt die Header-Datei `<limits>`.
- Der Type `streamsize` benötigt die Header-Datei `<iostream>`.

Template „numeric_limits“

`numeric_limits`	`<`	`streamsize`	`>`
`template`	`<`	`datentyp`	`>`

- Die Vorlage ermittelt den Wertebereich eines beliebigen Datentyps.
- Der Datentyp wird in spitzen Klammern an das Template übergeben.

… wird angewendet auf

`numeric_limits`	`<`	`streamsize`	`>`
`template`	`<`	`datentyp`	`>`

- In diesem Beispiel wird der Typ `streamsize` genutzt.
- Das Template `numeric_limits` generiert für diesen Datentyp alle vordefinierten Methoden.
- Der Datentyp `streamsize` definiert die Anzahl von Zeichen, die in einer IO-Operation transferiert werden dürfen.

Nutzung der Methode ...

`numeric_limits`	`<`	`streamsize`	`>`	`::`	`max`	`(`	`)`
`template`	`<`	`datentyp`	`>`	`::`	`methode`	`(`	`)`

- In diesem Beispiel wird die Methode `max()` genutzt, um die maximale Anzahl von Zeichen zu ermitteln.
- Jede Methode benötigt eine Parameterliste. Für diese Methode ist diese leer.
- Die Methode wird mit dem Template durch den Qualifizierungsoperator `::` verbunden. Die Methode `max()` ist in der Vorlage `numeric_limits` definiert.

Deaktivierung des Fehlerflags

```
std::cin.clear();
```

- Das Error-Flag der Standard-Eingabe `cin` wird deaktiviert.
- Bis zu diesem Zeitpunkt gespeicherte Fehler haben keine Auswirkungen auf den nächsten Einlese-Vorgang.