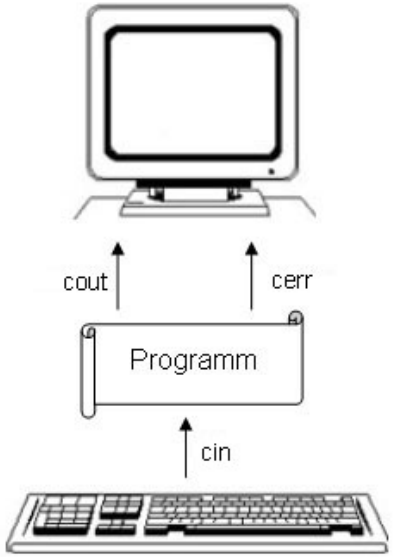
C++ - Einführung in die Programmiersprache

„Ein- und Ausgabe in der Konsole / Shell“



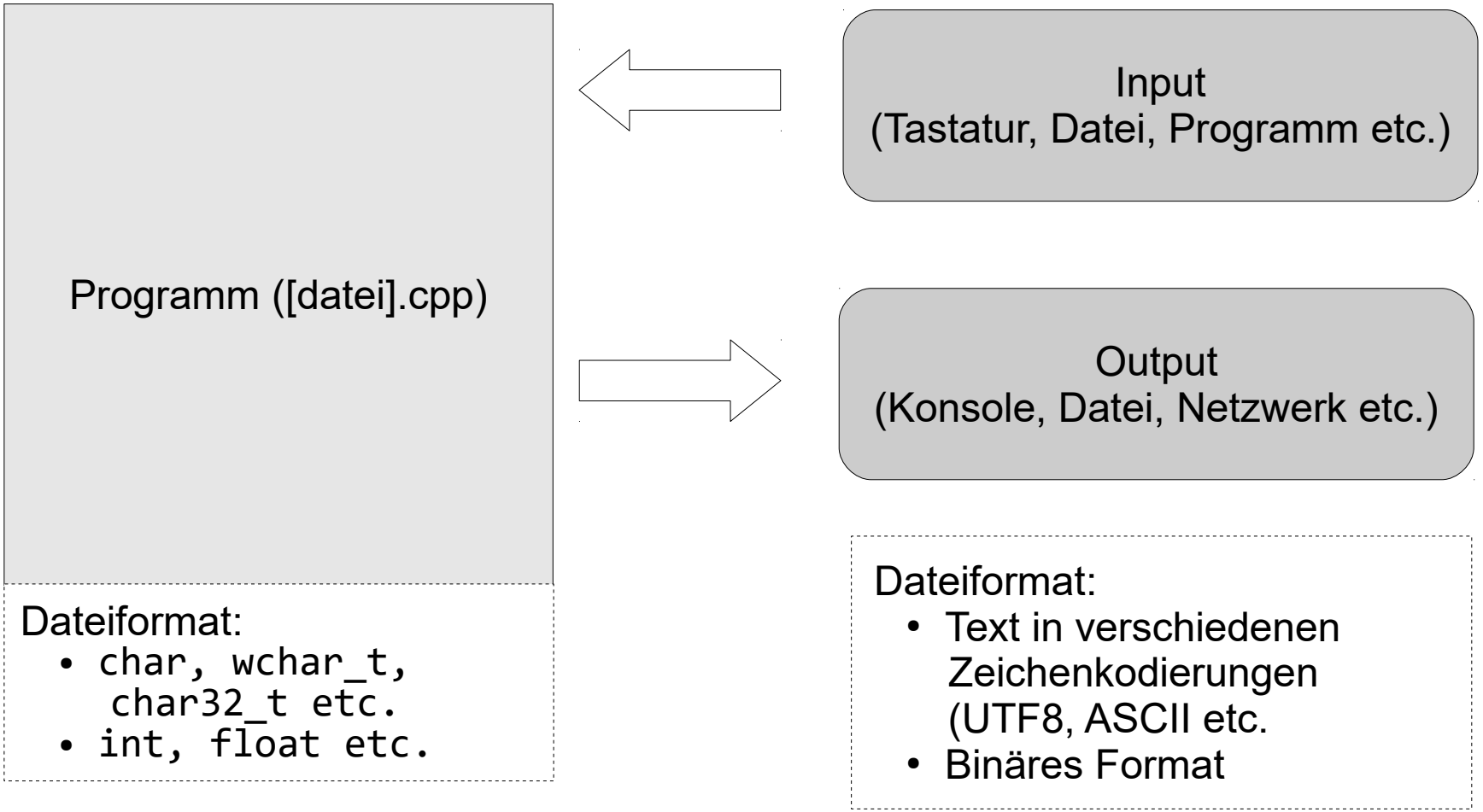
Konsolen

- 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.

Ein- und Ausgabe in C++



Header-Dateien für die Ein- und Ausgabe

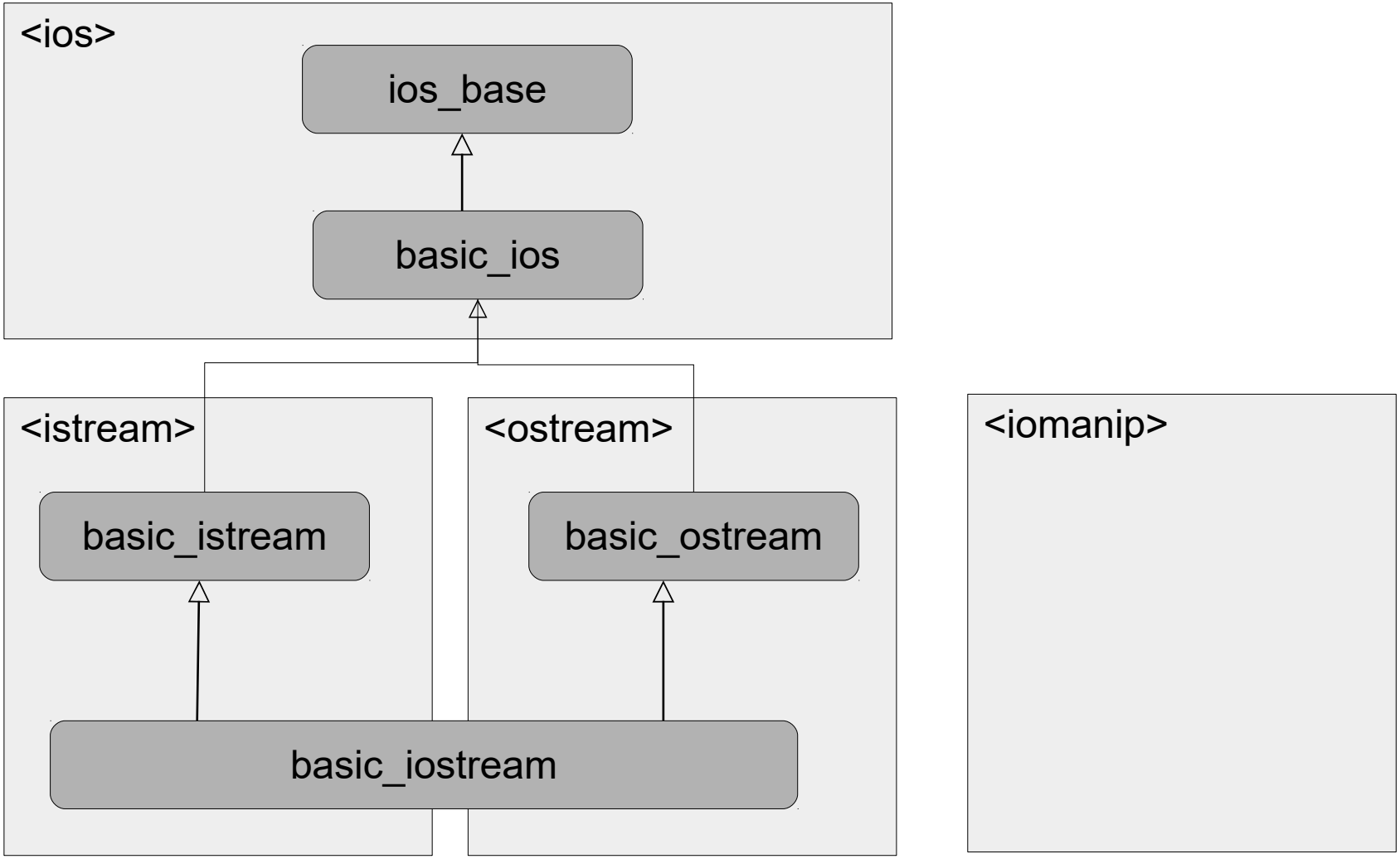
Header-Dateien

- C++-Dateien mit der Endung „.h“.
- Deklaration von Funktionen und Klassen.
- Einbindung in eine „.cpp“-Datei: `#include <header>`.
- Alle Standard-Header-Dateien bilden die Standard-Bibliothek von C++.

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

Header <iostream>



… einbinden

```
#include <iostream>
```

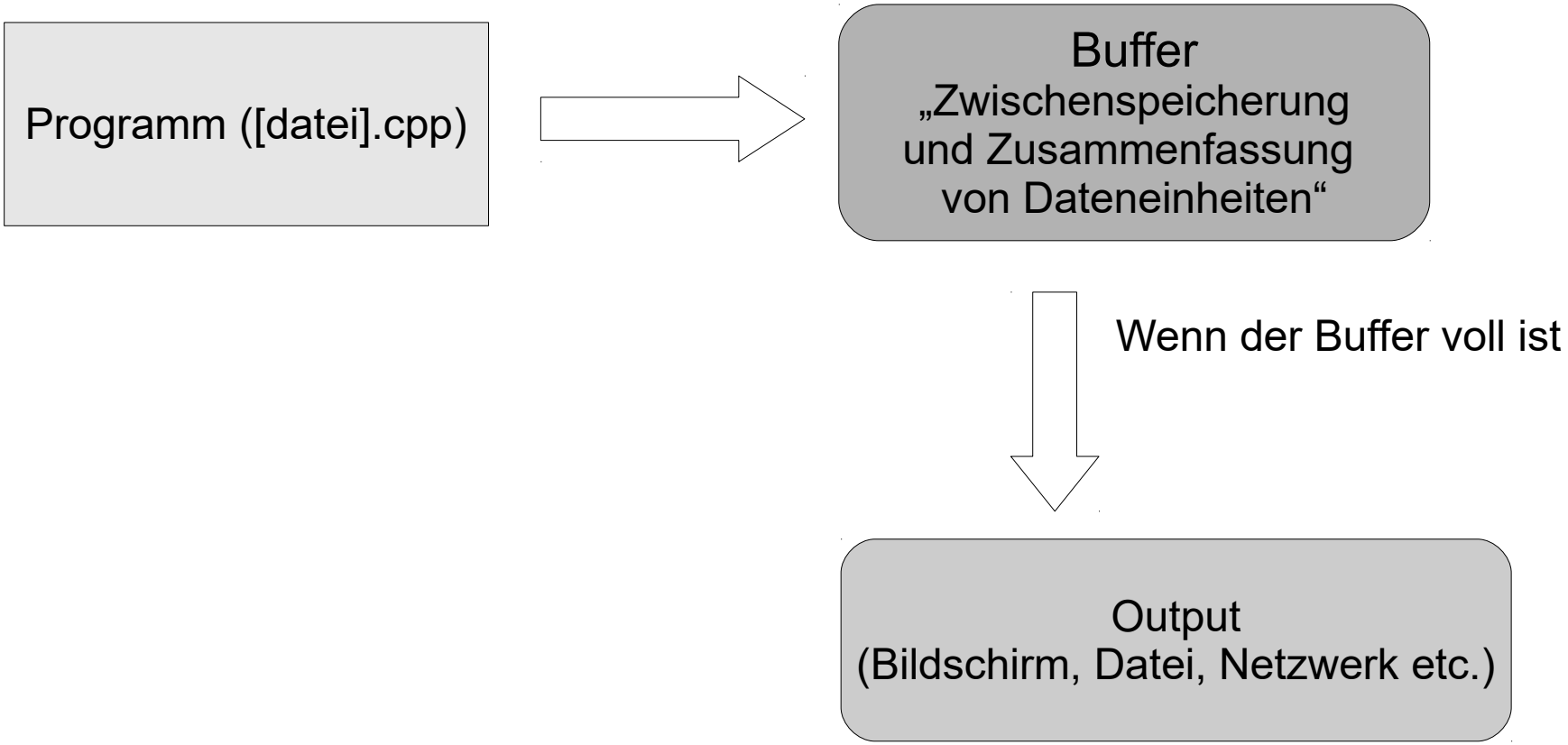
- Die Präprozessor-Anweisung `#include` bindet Header-Dateien am Anfang eines Programms ein.
- Der Anweisung folgt in spitzen Klammern der Name der einzubindenden Datei.
- Header-Dateien aus der Standard-Bibliothek beginnen und enden mit den spitzen Klammern.
- Die Datei `iostream` muss für die Ein- und Ausgabe auf die Konsole eingebunden werden

Ausgabe auf die Konsole

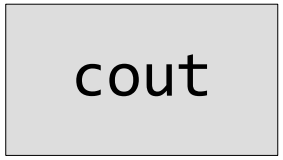
```
#include <iostream>

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

Grafische Darstellung

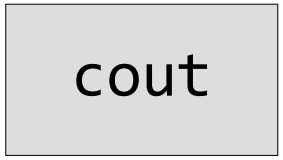


Ausgabe-Stream



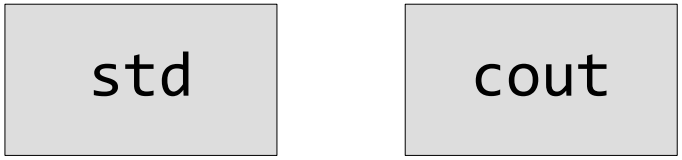
- `cout` beschreibt Eigenschaften und Methoden der Standardausgabe.
- Gepufferter Ausgabe-Stream auf die Konsole, in eine Datei etc.
- Ausgabe von Strings, Zeichen aus dem ASCII-Zeichensatz, Ganzzahlen und Gleitkommazahlen.

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 Namen von Funktionen etc. verschiedenen Namensräumen zugeordnet.
- Jeder Name ist in seinem Raum eindeutig.

Standard-Namensraum



- `std` bezeichnet den Standard-Namensraum in C++.
- Alle Funktionen etc., die in der Standard-Bibliothek deklariert sind, sind in diesem Namensraum abgelegt.

Funktion x befindet sich im Namensraum y

```
std  ::  cout
```

- Die zwei direkt nacheinander geschriebenen Doppelpunkte stellen einen Operator dar.
- `cout` ist in dem Namensraum `std` abgelegt.
- Qualifizierung eines Elements mit seinem übergeordneten Element.
- Auflösung eines Bereichs.

Umleitung der gewünschten Ausgabe

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

- Mit Hilfe des Umleitungsoperators werden Variablen etc. auf den Ausgabestrom umgeleitet.
- Der Wert der Variablen wird zum Beispiel 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 in diesem Beispiel auf die Standardausgabe.

Umleitung der gewünschten Ausgabe

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

- Mit Hilfe des Umleitungsoperators werden Variablen etc. auf den Ausgabestrom umgeleitet.
- Der Wert der Variablen wird zum Beispiel auf der Konsole ausgegeben.

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.

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

\"	Anführungszeichen.
\'	Apostroph.
\\	Umgekehrter Schrägstrich. Backslash.
\?	Fragezeichen.
\0	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";
```

- 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++.

Formatierte Ausgabe

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

int main() {
    double dblZahl = 12.123456789;
    int intZahl = 12;

    cout << setfill('-');
    cout << setw(10) << "12345";
    cout << "\n" << setprecision(5) << dblZahl;

    cout << '\n' <<  setiosflags(ios::left)
         << setw(10) << "NWTC-20" << '\t' << "Produkt A";
    cout << '\n' <<  setiosflags(ios::left)
         << setw(10) << "NWTCA-40" << '\t' << "Produkt B";
    cout << resetiosflags(ios::left);
```

Einbindung der Bibliothek iomanip

```
#include <iostream>
#include <iomanip>

using namespace std;
```

- Formatierung der Ausgabe mit Hilfe von Manipulatoren.
- Alle Manipulatoren nutzen de Standard-Namensraum.

Mindestanzahl von Zeichen

```
std::cout << std::setw(10) << 12345;
```

- `setw(int)` .
- Der Parameter legt die Mindestbreite der Ausgabe fest.
- Wie viele Zeichen werden mindestens ausgeben? In dem obigen Beispiel werden mindestens 10 Zeichen ausgegeben.

Nutzung eines Füllzeichens

```
cout << setfill('-');
cout << "\n" << setw(10) << "12345";
```

- Anzahl der ausgegebenen Zeichen kleiner als die Mindestanzahl von Zeichen: Auffüllung mit Leerzeichen
- `setfill(char)` legt ein beliebiges andere Füllzeichen fest.
- Das Füllzeichen ist immer vom Datentyp `char`.

Formatierung von Gleitkommazahlen

```
#include <iostream>
#include <iomanip>

using namespace std;

int main() {
    std::cout << "\n\n" << "Gleitkommazahlen: ";
    std::cout << "\n" << std::setprecision(5) << 12.123456789;
    std::cout << "\n" << std::fixed << 12.123456789;
    std::cout << "\n" << std::scientific << 12.123456789;
```

Darstellung ohne Exponenten

```
std::cout << std::fixed << 0.012345 << '\n';
```

- +-[Vorkommastellen].[Nachkommastellen]

Festlegung der Genauigkeit

```
std::cout << std::fixed
          << setprecision(4)
          << 0.012345 << '\n';
```

- `setprecision(int)`.
- Die Anzahl der Nachkommastellen werden festgelegt. In diesem Beispiel hat die Gleitkommazahl vier Nachkommastellen.

Darstellung mit Exponenten

```
std::cout << std::scientific << 0.012345 << '\n';
```

- Darstellung in der Exponentialschreibweise.
- Nutzung bei sehr großen oder kleinen Gleitkommazahlen.

Festlegung der Genauigkeit

```
std::cout << std::scientific
          << setprecision(4)
          << 0.012345 << '\n';
```

- `setprecision(int)`.
- Die Anzahl der Nachkommastellen werden festgelegt. In diesem Beispiel hat die Gleitkommazahl vier Nachkommastellen.

Standardeinstellung

```
std::cout << std::defaultfloat << 0.012345 << '\n';
```

- Einführung mit C++ 11.
- Entspricht weder der Einstellung `fixed` noch `scientific`.

Festlegung der Stellen

```
std::cout << std::fixed
          << defaultfloat(4)
          << 0.012345 << '\n';
```

- `setprecision(int)`.
- Keinerlei Auswirkungen, wenn die Anzahl der Stellen vor und nach dem Komma plus das Dezimaltrennzeichen größer als die Angabe ist.

Nutzung von Formatierungsflags

```
cout << '\n' <<  setiosflags(ios::left)
   << setw(10) << "NWTC-20" << '\t' << "Produkt A";

cout << resetiosflags(ios::left);
```

- Mit Hilfe der Funktion `setiosflags()` werden Flags aktiviert.
- Die Funktion `resetiosflags()` deaktiviert das übergebene Flag. Die Standardeinstellung für das Flag wird genutzt.
- Mögliche Parameter der Funktionen: http://www.cplusplus.com/reference/ios/ios_base/fmtflags/.

Beispiel: Linksbündige Ausrichtung

```
std::cout << '\n' <<  setiosflags(std::ios::left)
   << std::setw(10) << "NWTC-20"
   << '\t' << "Produkt A";

cout << resetiosflags(ios::left);
```

- Das Flag `left` beginnt den Text am linken Rand.
- Das Flag `left` ist in dem Namensraum `ios` eindeutig.
- Der Namensraum `ios` ist im Standard-Namensraum `std` gekapselt.

Formatierungen mit Hilfe von cout

```
#include <iostream>

using namespace std;

int main() {
    std::cout.left;
    std::cout << '\n';
    std::cout.width(10);
    std::cout << "NWTC-20"   << '\t' << "Produkt A";

    std::cout.right;
    std::cout.fill('*');
    std::cout.width(10);
    std::cout.fixed;
    std::cout.precision(2);
    std::cout << 12.40 << "\n";
```

Objekt cout

```
std::cout.left;
std::cout << '\n';
```

- Das Objekt `cout` beschreibt den Standard-Ausgabestrom.
- Mit Hilfe von Attributen wird das Objekt beschrieben.
- Das Objekt kann bestimmte Aktivitäten ausführen. Die Aktivitäten verändern häufig den Wert eines Attributs des Objekts.

Methoden eines Objektes

```
std::cout.left;
std::cout.width(10);
```

- Mit Hilfe des Punktoperators werden Methoden und Objekte verbunden.
- Die Methode wird rechts vom Punkt angegeben.
- Das Objekt wird links vom Punkt definiert.
- Die Aktion führt das Objekt links vom Punkt aus und ändert damit seine Attribute.

Feldbreite

```
std::cout.width(10);
```

- `.width(int)`.
- Mindestanzahl der auszugebenden Zeichen.
- Mindestanzahl der auszugebenden Stellen.

Füllzeichen

```
cout.fill('*');
```

- `.fill(char)`.
- Standardmäßig wird die Mindestbreite, falls nötig, mit einem Leerzeichen aufgefüllt.
- Die Methode legt ein neues Füllzeichen fest.

Ausgabe von Wide character type

```
wchar_t zeichenLocal;

zeichenLocal = L'A';
std::cout << "\n wchar_t: " << zeichenLocal;
std::wcout << "\n wchar_t (Zeichen): " << zeichenLocal;
```

- Mit Hilfe von `cout` werden ASCII-Zeichen auf der Konsole ausgegeben.
- Mit Hilfe von `wcout` werden Zeichen ausgegeben, die einen Speicherbedarf größer als 1 Byte haben.

Wide character type

```
wchar_t zeichenLocal;

zeichenLocal = L'\u0041';
zeichenLocal = L'A';
```

- Zeichen, die einen Speicherbedarf größer als 1 Byte haben.
- Jedes Zeichen hat das Präfix `L`.
- Abbildung von länderspezifischen Zeichen oder anderen Sonderzeichen wie Euro.

Eingabegebietsschema

- Einbindung der Bibliothek `<locale>`.
- Beeinflussung der Zahlendarstellung, Datumseingaben und länderspezifische Zeichen.
- Siehe: https://en.cppreference.com/w/cpp/locale/setlocale, http://www.cplusplus.com/reference/locale/.

Eingabegebietsschema

```
    std::cout << "\n Schema: "
              << std::setlocale(LC_ALL, "");
```

- `setlocale` gibt das aktuelle Eingabegebietsschema zurück oder legt dieses fest.
- Die Methode `setlocale` ist in der Bibliothek `locale` definiert.
- Die Methode ist im Standardnamensraum abgelegt.
- Der erste Parameter legt die Kategorie des Schemas fest.
- Der zweite Parameter definiert das gewünschte Schema..

Kategorie LC_ALL

```
    std::cout << "\n Schema: "
              << std::setlocale(LC_ALL, "");
```

- Das Makro `LC_ALL` gibt das aktuell genutzte Schema von C++ zurück.
- Der zweite Parameter muss ein leerer String (zwei aufeinanderfolgende Anführungszeichen) sein.

Kategorie LC_CTYPE

```
std::setlocale(LC_CTYPE,"de_DE.utf8");
```

- Das Makro `LC_CTYPE` legt das aktuelle Schema für Zeichen fest.
- Der zweite Parameter gibt das Schema an. In diesem Beispiel wird Unicode 8 aus dem deutschsprachigen Raum genutzt.
- Die Namen der Schemata werden durch die Anführungszeichen begrenzt.
- Hinweis: Die Namen der Schemata sind nicht normiert.

Unicode-Zeichen

```
char unicode = = '\u0041';
char zeichen = '\u02C3';
```

- Maskierung mit \u.
- Angabe als Hexadezimal-Wert.
- Das erste Zeichen im Unicode-Zeichensatz wird folgendermaßen kodiert: `'\u0000'`.
- Unicode-Zeichen können die Warnung `multi-character character constant` auslösen. Das Unicode-Zeichen wird nicht als ein Zeichen interpretiert.

Unicode-Zeichensatz

- Das erste Zeichen im Zeichensatz wird mit Hilfe von `'\u0000'` angegeben.
- Die ersten 127 Zeichen des UTF-8-Zeichensatzes sind mit dem ASCII-Zeichensatz identisch.
- Die ersten 256 Zeichen des UTF-8-Zeichensatzes entsprechen dem ISO 8859-1 (Latin 1)-Zeichensatz.
- Siehe https://unicode-table.com/de/.

Datentyp char16_t

```
char16_t u16;

u16 = u'A';
cout << "\n Unicode (Character) 16 Bit: " << u16;
u16 = u'\u0041';
cout << "\n Unicode 16 Bit: " << u16;
```

- Das Zeichen benötigt 16 Bit zur Speicherung.
- Jedes Zeichen hat das Präfix u.

Datentyp char32_t

```
char32_t u32;

u32 = U'@';

std::cout << "\n Unicode (Codierung): " << u32;
std::cout << "\n Unicode (Character): " << (char)u32;
```

- Das Zeichen benötigt 32 Bit zur Speicherung.
- Jedes Zeichen hat das Präfix `U`.

… in der Konsole ausgeben

```
char32_t u32;

u32 = U'@';

std::cout << "\n Unicode (Codierung): " << u32;
std::cout << "\n Unicode (Character): " << (char)u32;
```

- Standardmäßig wird die Codierung des Unicode-Zeichens in der Konsole ausgegeben.
- Durch die Angabe `(char)` wird die Codierung in das entsprechende Zeichen konvertiert.

Wide character type

```
wchar_t zeichenLocal;

zeichenLocal = L'\u0041';
zeichenLocal = L'A';
```

- Keine festgelegte Länge.
- Jedes Zeichen hat das Präfix `L`.
- Abbildung von länderspezifischen Zeichen.

… auf der Konsole ausgeben

```
wchar_t zeichenLocal;

zeichenLocal = L'A';
std::cout << "\n wchar_t: " << zeichenLocal;
std::wcout << "\n wchar_t (Zeichen): " << zeichenLocal;
```

- Mit Hilfe von `cout` wird die Codierung des Zeichens entsprechend der länderspezifischen Einstellung in der Konsole ausgegeben.
- Mit Hilfe von `wcout` wird das Zeichen entsprechend der länderspezifischen Einstellung in der Konsole ausgegeben.