§1 Einführung – Programmaufbau in C++

Leitidee: *Genaueres Verständnis der C++-Syntax an Hand von Syntaxdiagrammen*

- Lexikalische Struktur von C++
- Vereinbarungen: Definition / Deklaration
- Beispiele für Variablen- und Funktionsdefinition
- Übersetzungseinheit
- Beispiel: hello.cpp
- Ausdrucksanweisung, insb. Zuweisung
- Laufanweisung: `while` *(Zusatzfolie!)*
- Beispiele: Quadratzahlen, Arithmetische Reihe, Exponentialreihe
- Bedingte Anweisung: `if` *(Zusatzfolie!)*
- Beispiel: Überprüfung von Eingabedaten
- Ein/Ausgabe *(Zusatzfolie!)*

Lexikalische Struktur von C++

- Zeichensatz:
  Druckbare Zeichen des ASCII-Zeichensatzes *ohne* `$,\`,@`
  Zwischenraumzeichen, Zeilenende

  Weitere druckbare Zeichen sind möglich in

Zeichenkettenliteralen	`"Eine Zeichenkette"`
Zeichenliteralen	`'u'`
Kommentaren	`/* C/C++ Kommentar */`
	`// C++ Kommentar`

- Mit dem Zeichensatz werden die Eingabesymbole (syntaktische Grundeinheiten der Programmiersprache) geschrieben:

Namen	`x1 i main Gamma_Funktion`
Reservierte Worte	`return while if`
Literale	`"Text" 'z' 539 12.75`
Operatoren	`+ * ++`
Trennzeichen	`; , ( ) { }`

Lexikalische Struktur von C++ - Fortsetzung

- Namen bestehen aus Buchstaben, Ziffern und _
  Namen dürfen *nicht* mit Ziffern beginnen!
- Zwischenraum (einschl. Zeilenumbrüchen) kann beliebig *zwischen* Eingabesymbolen eingefügt werden.
- Ausnahme: Präprozessoranweisungen müssen jeweils in einer Zeile stehen.
  Bsp.:
  ```
  #include <iostream>
  #include <cmath>
  ```
- Zwischen direkt benachbarten Namen, reservierten Worten und Literalen muss Zwischenraum stehen.

Programmaufbau I

Syntaxdiagramm „Übersetzungseinheit“ auf Infoblatt!

- C++-Programme bestehen (nach Auswertung der Präprozessoranweisungen) aus Funktionsdefinitionen und einfachen Vereinbarungen.
- *Vereinbarungen* sind entweder *Definitionen* oder *Deklarationen*.

  Unterschied:

  Definitionen reservieren auch Speicherplatz
  Deklarationen machen Namen bekannt, die an anderer Stelle definiert sind

  Bsp.: `sqrt` — deklariert in `cmath`
  — definiert in `libm.a`

  `std::cout,std::cin` — definiert in `iostream`

  Wegen der Anweisung `using namespace std;` kann `std::` weggelassen werden!

Programmaufbau II

Syntaxdiagramm „einfache Vereinbarung“ auf Infoblatt!

- Bsp. für Variablendefinition

  `int i=4, j;` Definiert i mit Wert 4
  und j ohne Wert
  i und j sind ganzzahlig

  `int` Deklarationsangabe, hier: Typname
  `i` Deklarator, hier: Variablenname

- Weiteres Bsp. für Variablendefinition

  `double x, y=3.7;` Definiert x ohne Wert
  und y mit Wert 3.7
  x und y sind Gleitpunktzahlen

Programmaufbau III

Syntaxdiagramme „Funktionsdefinition“ und „Block“ auf Infoblatt!

- Bsp. für Funktionsdefinition: `int main(){ ... }`
  `int` Deklarationsangabe, hier: Typname
  `main()` Deklarator, hier: Fkt.name mit leerer Parameterliste
  `{ ... }` Block
- Erforderlich: Es muss *genau eine* Funktion mit Namen `main` („Hauptprogramm“) geben.
- Besonderheit: Rückgabewert (ganzzahlig) von `main` wird im Programm nicht weiter benutzt.
  Zweck: Information des Betriebssystems über etwaige Fehler beim Programmlauf
  Konvention: 0 – erfolgreich
  $\neq 0$ – Fehler

Programmaufbau IV

- C++ lässt getrenntes Übersetzen zu: Ein Programm kann über mehrere Dateien verteilt sein, die zu unterschiedlichen Zeitpunkten übersetzt werden.
  Übersetzungseinheit = Datei nach Ausführung der Präprozessoranweisungen
- Momentan: 1 Übersetzungseinheit mit der Funktion `main`.
  [Erste Hälfte des Semesters: Beschränkung auf 1 Übersetzungseinheit und 2-3 Funktionsdefinitionen.]

Bsp.:

```
#include <iostream>                    Präprozessoranweisung
using namespace std;                   Namespace-Anweisung

                                       optionale Leerzeile
int main()                             Funktionskopf von main
{                                      Beginn Funktionsblock
  cout<<"Hello, world!"<<endl;         Ausgabeanweisung
  return 0;                            Rückgabewert setzen
}                                      Ende Funktionsblock
```

Ausdrucksanweisung

Syntaxdiagramm in „Anweisung“ auf Infoblatt enthalten!

- Mit Operatoren und Funktionen können aus Literalen und Namen kompliziertere Ausdrücke aufgebaut werden.
- *Ausdrucksanweisung:* Ausdruck, gefolgt von Strichpunkt. Bewirkt die Auswertung des Ausdrucks.

  Bsp.: `cout << "sqrt(2)=" << sqrt(2) << endl;`
- Ein sehr wichtiger Spezialfall der Ausdrucksanweisung ist die Zuweisung:

  Variablenname = *Ausdruck*`;`

  Wirkung: Der Ausdruck auf der rechten Seite wird ausgewertet und in dem durch Variablenname bezeichneten Speicherplatz gespeichert.

  Bsp.:

  ```
  i = 4;      i erhält den Wert 4
  j = i*i;    j erhält den Wert 16
  i = i+1;    i erhält den Wert 5
  ```

  Zuweisungen dürfen nur erfolgen, wenn die Größen auf der rechten Seite mit Werten versehen wurden.