Grundstruktur:

Liste von Compilerdirektiven
(jede in 1 Zeile, jede beginnt mit **#**)

Liste von *Funktionen*

hier nur eine Funktion: **`main`** (Hauptfunktion) –
diese *muss* vorhanden sein.
Hier ist stets der Einstiegspunkt des Programms!

"**`int main()`**" bedeutet:

- **`main`** erwartet keine Argumente ()
- **`main`** gibt einen **`int`**-Wert (= eine ganze Zahl) als Ergebnis zurück – dieser ist hier bedeutungslos (deshalb **`return 0`**); das eigentliche Ergebnis wird innerhalb von **`main`** mittels "**`printf`**" ausgedruckt!

`main` hat die Struktur:

`int main()`
Block

Block : {
Folge-von-Deklarationen-und-Anweisungen
}

{ ... } geschweifte Klammern (hier nicht als Mengen-Klammern gebraucht!):

- umgrenzen einen Block
- in anderen Programmiersprachen stattdessen:
BEGIN ... END (z.B. Pascal)

Deklaration:
Jede Variable (zum Einlesen, Rechnen, Ausgeben) muss *deklariert* (erklärt, eingeführt) sein (in der Funktion, wo sie gebraucht wird, oder außerhalb = global *)

(*) globale Variablendeklaration: auch möglich

```
#include .....
float a, flaeche, vol;  char puffer[130];
int main()
    {
    printf .....
```

Deklaration : *Typ Bezeichner-Liste* **;**

Bezeichner-Liste : *Bezeichner* { **,** *Bezeichner* }*

Typ: Schlüsselwort für den Datentyp,
gibt an, welche Werte die Variable annehmen kann.

Beispiele:

```
float b;
float a, flaeche, vol;
int n1;
char puffer[130];
unsigned int k_1;
```

(Array (Feld) von 130 **char**-Variablen hintereinander (siehe später))

Schlüsselwörter:

- haben feste Bedeutung
- dürfen nicht als Bezeichner verwendet werden

Liste aller C-Schlüsselwörter: siehe Folie

Literale: Werte im Programm, die nicht Variable sind
(sondern "buchstabengetreu" dastehen)

z.B. **`130`**, **`3.0`**, **`"\Ergebnis: "`**, **`0`**

Literal :

- *Integer-Konstante*
- *Gleitkomma-Konstante*
- *Zeichenkonstante*
- *String-Konstante*
- *Aufzählungstypen-Konstante*

Näheres zur Konstanten-Syntax später,
zunächst zur Informationsdarstellung im Rechner