Prozedurale Programmierung
Einfache C-Programme

Hochschule Rosenheim – University of Applied Sciences

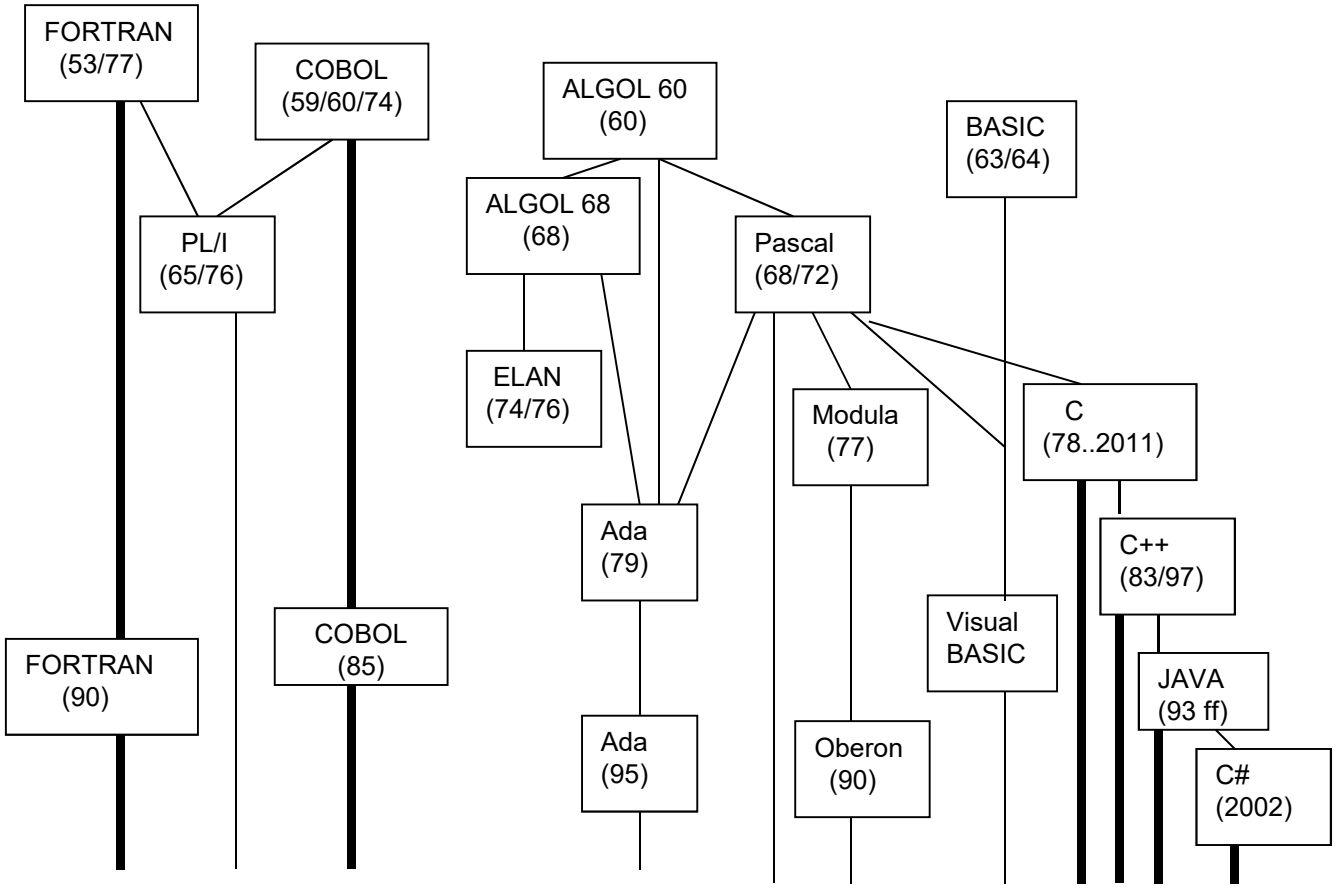
WS 2018/19

Prof. Dr. F.J. Schmitt

Überblick

- Einführung und Grundlagen der Programmiersprache C
  - historische Entwicklung
  - Aufbau und Bestandteile von C-Programmen

Prozedurale Programmiersprachen



Historische Entwicklung von C (1)

- Ursprünglich aus den Programmiersprachen BCPL (Basic Combined Programming Language) und B entstanden

- Bei der Entwicklung von Unix entstand der Bedarf nach neuer Programmiersprache
  - Möglichkeiten der HW-nahen Programmierung vgl. mit Assembler
  - Performance des Laufzeitcodes vgl. mit Assembler
  - Unterstützung der Sprachmittel der strukturierten Programmierung

Historische Entwicklung von C (2)

- 1971/72: Weiterentwicklung von B zu C (Ritchie)
- 1973: Realisierung von Unix in C
  (1/10 bleibt Assembler-Code)
- Bis 1978: Weiterentwicklung in mehreren Schritten durch Kernighan und Ritchie („Sprachbibel“)
- Verschiedene Dialekte entstehen in den folgenden Jahren
  - Erhebliche Einschränkungen in der Portablilität
- 1989: Standardisierung durch ANSI-C
  - Normierung der Sprache und der Standard-Bibliothek

Historische Entwicklung von C (3)

- 1990: internationaler Standard ISO/IEC 9899
  → C90 „ANSI C“
  - entspricht weitgehend ANSI C von 1989

- 1999: Überarbeitung des internationalen Standards ISO/IEC 9899:1999, übernommen als ANSI Standard 2000
  → C99

- 2011: neueste Überarbeitung des Standards, ISO/IEC 9899:2011; ersetzt C99
  → C11

Einordnung der Sprache C (1)

- Prinzipielle Unterscheidung:
  - **Imperative** Programmiersprachen
    - Geprägt durch die ***von-Neumann-Architektur***
      (Befehle und Daten werden im gleichen Speicher bearbeitet)
    - Programm besteht aus
      - Variablen (stellen die Speicherzellen dar)
      - einer Folge von Befehlen (verarbeiten die Daten)
    - Zentraler Ansatz: Algorithmus
    - Verschiedene Untertypen:
      - maschinenorientierte (Assembler)
      - prozedurale (Fortran, ALGOL, Pascal oder C)
      - objektorientierte Sprachen (Smalltalk, C++, Java)

Einordnung der Sprache C (2)

- Deklarative Programmiersprachen
  - Nicht die Verarbeitungsschritte werden angegeben
  - Direkte Beschreibung des Ergebnisses (Deklaration)
  - Übersetzer/Interpreter leitet die Verarbeitungsschritte ab
  - Beispiele: LISP, PROLOG, OPS5

Charakteristische Eigenschaften von C

- C besitzt Doppelnatur
  - ***prozedurale*** Hochsprache
  - ermöglicht aber ***maschinennahe*** Programmierung
- Unterstützung von ***strukturierter*** und ***modularer Programmierung***
- Sprache mit ***wenig*** „Worten“, sogenannten ***Schlüsselwörtern*** ↔ ***mächtige*** und flexible Programmiersprache
- ***Portable*** Sprache → Übersetzung und Ausführung auf anderen Systemen mit geringen Modifikationen möglich

Minimales C-Programm

- Jedes C-Programm besteht aus Funktionen, die sich gegenseitig aufrufen
- Erste Funktion, die bei Programmstart ausgeführt wird: `main-Funktion` (Startpunkt – Hauptfunktion)
  - Auszuführenden Anweisungen stehen innerhalb der Klammern { } (Funktionsblock)
  - Jede Anweisung wird mit Semikolon ; abgeschlossen
  - `return`-Anweisung: Funktion wird verlassen

```
int main()
{
    return 0;
}
```

Kommentare

- Ziel: Dokumentation / Erläuterung des Programmcodes
  - Mehrzeilenkommentare

```
/*
Dieser Kommentar belegt
insgesamt
fünf Zeilen.
*/
```

Verwenden Sie ausreichend Kommentare!

  - Einzeilenkommentare

```
// Dieser Kommentar belegt eine Zeile.
```

- Kommentare sind Zeichenfolgen, die vom Präprozessor vor Übersetzung entfernt werden

Einfache C-Programme: Hallo Welt

```
/*************************************\
*Dateiname: hello_world.c
*Beschreibung: Hallo Welt Programm
*Verfasser: C. Förster
*Datum: 8.10.2010
\*************************************/

#include <stdio.h>

int main()
{
   //Ausgabe
   printf("Hallo Welt!");
   return 0;
}  //end main
```

Sechszeiliger Kommentar

Einbinden der Header-Datei `stdio.h`, die zur Bekanntmachung der `printf`-Funktion benötigt wird

Hauptprogramm `main`

Ausgabe von „Hallo Welt!“

Einbinden von Header-Dateien

- Sollen Funktionen aus Bibliotheken verwendet werden, so müssen die zugehörigen Header-Dateien eingebunden werden
  - Einbinden einer System-Header-Datei

```
#include <dateiname.h>
```

Präprozessor-Anweisung

  - Einbinden einer selbstdefinierten Header-Datei

```
#include "dateiname.h"
```

Präprozessor-Anweisung

Woraus besteht ein C-Projekt?

- Programmtext (Quelltext-Datei(en))
  - besteht aus einer oder mehreren Dateien mit der Endung `.c`
- Header-Dateien
  - enthalten Informationen, die in mehreren C-Dateien gebraucht werden und tragen die Endung `.h`
    - Deklaration (Bekanntmachung) von Funktionen
    - Definition von Datentypen
    - **KEINE** ausführbaren Programmtexte!

Einfache C-Programme: Reklameschrift

```
/*********************************\
*Dateiname: tom.c
*Beschreibung: Ausgabe von "TOM"
               als Reklameschrift
*Verfasser: Förster
*Datum: 8.10.2010
\*********************************/

#include <stdio.h>

int main()
{
   //Ausgabe von "TOM"
   printf("*****   ***   *   *\n");
   printf("  *    *   *   ** **\n");
   printf("  *    *   *   * * *\n");
   printf("  *    *   *   *   *\n");
   printf("  *     ***   *   *\n\n");
   return 0;
}  // end main
```

`\n` ist eine von mehreren möglichen **Steuerzeichen** (Escape-Sequenzen) und bewirkt einen Zeilenvorschub

Die Anweisung
`printf("\n");`
bewirkt die Ausgabe einer Leerzeile!

Empfehlung – Aufbau eines Programms

- Empfehlung (für 1 Datei):
  (1) Datei-Header (Kommentar – Datei-Beschreibung)
  (2) Include-Anweisungen
  (3) Definition von symbolischen Konstanten, globalen Datentypen und externen (globalen) Variablen
  (4) Prototypen-Deklarationen der eigenen Funktionen
  (5) Definition der main-Funktion
  (6) Definition der eigenen Funktionen
  (die in main-Funktion aufgerufen werden)

Beispielprogramm – Berechnung

```
/*      Datei: rechnen.c
        Verfasser:
*/
#include <stdio.h>
int main(void) // Beginn Hauptprogramm
{
  double x,y;  // Variablendefinition

  printf ("Bitte gib ein: \n1. Zahl: ");
  scanf ("%lf",&x);  // Eingabefunktion
  printf ("2. Zahl: ");
  scanf ("%lf",&y);
  printf ("Summe:      %10.4lf \n",x+y);
  printf ("Produkt:    %10.4lf \n",x*y);
  printf ("Differenz: %10.4lf \n",x-y);
  printf ("Quotient:  %10.4lf \n",x/y);
}  // Ende Hauptprogramm
```

Zusammenfassung

- historische Entwicklung/Einordnung der Sprache
- Aufbau einfacher C-Programme
  - .c/.h Dateien
  - main
  - Ein-/Ausgabe printf() / scanf()