Organisatorisches

Heute

- Programmiersprache C, Kommazahlen
- Lehrevaluation

Nächste Woche

- Keine Vorlesung am 7.12.! Aber Praktikum!
- Nächste Vorlesung am 14.12.

12.4 Mindestanforderungen an höhere, maschinennahe Sprachen

effizient

Gutes Laufzeitverhalten, wenig Overhead.

typisiert

Operationen und Unterprogrammaufrufe werden zur Compilezeit daraufhin überprüft, ob die Parameter den vereinbarten Typ haben.

strukturiert

Es stehen Sprachkonstrukte im Sinne der Nassi-Shneiderman-Strukturblockarten zur Verfügung.

prozedural

Das Programm kann in benennbare Funktionsblöcke zerlegt werden. Von einer einmal geschriebenen Funktion muss nur noch das Schnittstellenverhalten bekannt sein.

modular

Der Quellcode kann in sinnvolle Einheiten unterteilt werden, z.B. mit dem Ziel der Wiederverwertbarkeit von Codeteilen. → Bibliotheken
Beispiele: Mathematische Funktionen, Grafikfunktionen, …..

13. Programmiersprache C

13.1 Historie

- C wurde 1972 von Dennis Ritchie bei den AT&T Bell Lab als Systemprogrammier-sprache zur Implementierung von UNIX für die PDP-11 entwickelt.

- Ziel von C war die Entwicklung einer Hochsprache für lesbare und portable Systemprogramme, aber einfach genug, um auf die zugrunde liegende Maschine abgebildet zu werden.

- In 1973/74 wurde C von Brian Kernighan verbessert. Daraufhin wurden viele Unix-Implementierungen von Assembler nach C umgeschrieben.

- Um die Vielzahl der entwickelten Compiler auf einen definierten Sprachumfang festzulegen, wurde C 1983 durch die amerikanische Normbehörde ANSI normiert. Diese Sprachnorm wird als ANSI-C bezeichnet. Die aktuelle Norm stammt aus dem Jahr 2011 (ANSI = **A**merican **N**ational **S**tandardisation **I**nstitute).

- Erweiterung für objektorientierte Programmierung (Objekte, Klassen, Vererbung): Sprache „C++“ (oft nicht unterschieden und gemeinsam betrachtet: „C/C++“)

13.2 Einführende Anmerkungen

13.2.1 Stärken und Schwächen

Stärken:

- standardisiert (ANSI)
- sehr effizient (hardwarenah implementiert)
- universell verwendbar
- sehr weit verbreitet, speziell in technischen Anwendungen
- Typüberprüfung (weitgehend stark typisiert, strong typing)

Schwächen:

- Code kann beliebig unlesbar geschrieben werden, da Fragen des Programmierstils nur in (freiwilligen) Konventionen festgelegt sind
  → Erfahrung und persönlicher Stil haben entscheidenden Einfluss auf die Softwarequalität !
- teilweise etwas kryptische Notation

13.2.2 Die größte Gefahr: Stilloser Code

```
/* ASCII to Morsecode                                          */
/* Obfuscated C Code Contest :   „Best“ small Program */

#include<stdio.h>
#include<string.h>

main()
{
        char*O,l[999]="'`acgo\177~|xp .-\0R^8)NJ6%K4O+A2M(*0ID57$3G1FBL";
        while(O=fgets(l+45,954,stdin)){
                *l=O[strlen(O)[O-1]=0,strspn(O,l+11)];
                while(*O)switch((*l&&isalnum(*O))-!*l){
                case-1:{char*I=(O+=strspn(O,l+12)+1)-2,O=34;
                        while(*I&3&&(O=(O-16<<1)+*I---'-')<80);
                        putchar(O&93?*I&8||!(I=memchr(l,O,44))?'?':I-l+47:32);
                        break;
                case 1:     ;}*l=(*O&31)[l-15+(*O>61)*32];
                        while(putchar(45+*l%2),(*l=*l+32>>1)>35);
                case 0:     putchar((++O,32));}
        putchar(10);}
}
```

13.2.3 Eigenschaften

C ist .....

klein

- 32 Schlüsselwörter und
- 40 Operatoren

modular

- alle Erweiterungen stecken in Funktionsbibliotheken
- unterstützt das Modulkonzept

maschinennah

- geht mit den gleichen Objekten um wie die Hardware:
  Zeichen,
  Zahlen,
  Adressen,
  Speicherblöcke.

13.3 Ein erster Blick auf (einfache) C-Programme

13.3.1 Beispiel eines einfachen C-Programms

```
#include <stdio.h>

int main( )
{
    printf("Hello World\n");
    return 0;
}
```

Eine **main-Funktion** wird immer benötigt, damit der Compiler den Beginn des Hauptprogramms erkennt. Das Programm steht zwischen { ... }.

Der Rückgabewert der Funktion ist **„int“** (ganze Zahl). In diesem Programm ist der Rückgabewert der **main()** - Funktion 0:

```
return 0;
```

Nach Konvention bedeutet das, dass das Programm fehlerfrei beendet wurde.

13.3.2 Komponenten eines C-Programms

Präprozessor

`#include <stdio.h>` ist kein direkter Bestandteil der Sprache C, sondern ein Befehl des sogenannten **Präprozessors**. Er führt vor dem eigentlichen Übersetzungsvorgang Textersetzungen durch und erlaubt die Steuerung des Übersetzungsvorganges. Präprozessorbefehle erkennt man am # in der ersten Spalte.

`#include` kopiert vor der Übersetzung den Text der Datei `<stdio.h>` an die Stelle der Präprozessoranweisung „`#include`“ (vor den Programmtext).

Ergebnisausgabe

Mit `printf("Hello World\n")` wird der in Anführungsstrichen “ “ stehende Text ausgegeben. `"\n"` ist eine sog. Escape-Sequenz und bedeutet „Zeilenumbruch“.

Zeilenende “;“

Alle Kommandos werden mit einem “;“ abgeschlossen.

Kommentare /* ..... */ und //

Kommentare sind in `/* Kommentar ... */` eingeschlossen, einzeilige Kommentare beginnen mit `//` .

13.3.3 Einfache Datentypen und formatierte Terminalausgabe

```
#include <stdio.h>
int main ()  /* Beginn des Hauptprogramms */
{
      int       i = 10;
      double    db = 1.23;
      printf("Zahl=%d",i);
      printf("\n");
      printf("%10.2lf",db);
      return 0;
}
```

Variablendeklaration

Alle verwendeten Variablen müssen deklariert werden, d.h. der Datentyp und Name der Variablen werden bekannt gemacht.

`double` deklariert z.B. eine reelle Zahl. Das Dezimaltrennzeichen ist der ".".

Formatierte Ergebnisausgabe

Ergebnisse können durch Formatbeschreiber (z.B. %d, %2d, %8.3lf) im Formatbeschreibungs-String formatiert ausgegeben werden.

%2d bedeutet, eine Integerzahl wird in einem Feld von 2 Zeichen ausgegeben.
%8.3lf bedeutet, eine double-Zahl wird in einem Feld von 8 Zeichen (Vor- und Nachkommazahlen und das Komma selbst), mit 3 Nachkommastellen ausgegeben.

13.3.4 Tastatureingabe

```
int main ()
{
      int i;
      printf("Bitte geben Sie eine Zahl ein : ");
      scanf("%d",&i);       /*Wartet auf Eingabe*/
      printf("Die Zahl die Sie eingegeben haben war %d\n",i);
      return 0;
}
```

Mit `scanf()` können Texte von der Tastatur eingelesen werden.
Der Formatbeschreiber (hier "%d") gibt den Typ der eingelesenen Werte an.

Die Variable muss den Adressoperator `&` vorangestellt haben (d.h. übergeben wird nicht der momentane Wert der Variablen `i`, sondern die Adresse von `i`).

13.4 Sprachelemente von C

13.4.1 Übersicht

C besitzt 6 **Wortklassen**:

- Bezeichner
- reservierte Worte
- Konstanten
  - Ganzzahlkonstanten
  - Gleitkommakonstanten
  - Zeichenkonstanten
- Strings
- Operatoren
- Trenner (Leerstellen, Zeilentrenner, Kommentare)

*13.4.2 **Bezeichner** (vom Programmierer vergebbare Namen)*

Bezeichner (engl.: Identifier) benennen auf eindeutige Weise

- Variablennamen,
- Konstantenbezeichner,
- Typbezeichner und
- Funktionsnamen.

Syntax:

identifier : *buchstabe* { *buchstabe* | *ziffer* } .

Unter Buchstaben wird die Menge der grossen und kleinen Buchstaben des lateinischen Alphabetes verstanden; zuzüglich des Zeichen '_' (*underscore*)

- C unterscheidet zwischen Gross- und Kleinschreibung (***case-sensitiv***).
- Die Namen können beliebig lang sein, wobei mindestens die ersten 31 Zeichen signifikant sind.
- Schlüsselwörter dürfen nicht verwendet werden (s.u.)

13.4.3 Reservierte Worte (Schlüsselwörter)

Folgende Bezeichner gehören in C zur Menge der **reservierten Wörter:**

auto	default	float	long	sizeof	union
break	do	for	register	static	unsigned
case	double	goto	return	struct	void
char	else	if	short	switch	volatile
const	enum	int	signed	typedef	while
continue	extern				

Darüberhinaus kann es Compiler-spezifische reservierte Wörter geben.

ÜBUNG Bezeichner

Welche der folgenden Bezeichner sind nicht erlaubt:

- `Mitgliedsnummer`
- `4U`
- `Auto`
- `Read_Me`
- `double`
- `Laurel&Hardy`
- `Ergänzung`
- `a`
- `ist_Null_wenn_die_Linie_laenger_als_MAXLEN_ist`

Einschub / Nachtrag: Zahlendarstellung

Was fehlt noch?

- Kommazahlen
- Was für Kommazahlen können unterschieden werden?

Vorschlag? Wie zu realisieren bzw. wie zu speichern?

EINSCHUB: ZAHLENDARSTELLUNG

3.3.4 Festkommazahlen

Festkommazahlen lassen sich ebenfalls durch das Stellenwertsystem darstellen. Die Nachkommastellen haben dann die Wertigkeit 2^{-1}, 2^{-2}, 2^{-3}, u.s.w..

Für eine Binärzahl mit n Vorkommastellen und m Nachkommastellen gilt somit:

$$\sum_{i=-m}^{n-1} a_i \cdot 2^i$$

Beispiel: 9.75_D = **1***2^3 + **0***2^2 + **0***2^1 + **1***2^0 + **1***2^{-1} + **1***2^{-2} = 1001.11_B

In Rechnern ist die Stellenzahl meist begrenzt (z.B. 8 bit vor und nach dem Komma). Daraus resultiert eine größte und kleinste darstellbare Zahl.

Beispiel:

Vorzeichenlose 16 bit Festkommazahl mit 8 bit Vor-/Nachkommaanteil

größte Zahl: `1111 1111.1111 1111`$_B$ = 255.99609375_D

kleinste Zahl (ungl. 0): `0000 0000.0000 0001`$_B$ = 0.00390625

….. Festkommazahlen: Diskussion

Vorteile:

- ermöglicht schnelle Berechnungen auch auf einfachen Prozessoren ohne Fließkommarecheneinheit (Floatingpoint-Unit, FPU)
- einfache Arithmetik
- Umrechnung (z.B. dezimal→binär) mit bekannten Verfahren, z.B. Addition von Zweierpotenzen und „Modulo-Division“ (→Modulo-Multiplikation)

Nachteile:

- trotz begrenzter Dezimalstellenzahl kann die Zahl der notwendigen Binärstellen für eine fehlerlose Darstellung unendlich sein.

  Beispiel: $0.8_D = 0.\overline{1100}_B$

- Einsetzbarkeit und notwendige Stellenzahl sind für den jeweiligen Einsatzfall sehr genau zu prüfen (Rundungsfehler).

  Beispiel: aus 0.8_D wird bei 8 Nachkommabitstellen 0.796875_D (err: 0.003125)

- Für hohe Genauigkeiten sind sehr viele Stellen notwendig.

ÜBUNG: Festkommazahlen

Geben Sie zu folgenden Dezimalzahlen die Festkomma-Binärzahlen (8 Vorkomma- und 8 Nachkommastellen) an.

15.6875_D , 37.1875_D , 10.2_D

Verwandeln Sie die zu 10.2_D korrespondierende Binärzahl wieder zurück in Dezimalzahlen. Wie groß ist der Fehler?

3.3.5 Gleitkommazahlen

Gleitkommazahlen (Floatingpoint) dienen zur Codierung (einer Teilmenge) der reellen Zahlen.

Prinzip: Jede Zahl, unabhängig von ihrer Größe, wird durch eine feste Anzahl *signifikanter Ziffern* dargestellt.

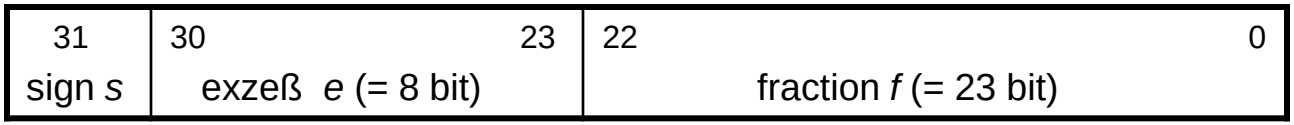
Beispiele: $0.31415 * 10^{1}$
$0.12345 * 10^{-23}$
$0.43434 * 10^{16}$

In Rechnern gilt: Zahl $\approx m * b^{e}$ mit m: Mantisse (Stellenzahl begrenzt)
b: Basis (= 2 bei Binärzahlen)
e: Exponent (Stellenzahl begrenzt)

Weltweit akzeptierter Standard wurde definiert vom IEEE (*Institut of Electrical and Electronics Engineers*) → **Standard 754**

….. Gleitkommazahlen : interne Darstellung

Darstellung von 4-Byte-Gleitkommazahlen (einfache Genauigkeit):



Normalisierte Zahlen werden dann wie folgt dargestellt:

$$Zahl = (-1)^s \cdot 2^{e-127} \cdot 1.f$$

sign (s) : Vorzeichen der Zahl (0: positiv, 1: negativ)

exzeß (e) : Vorzeichen des Exponenten wird durch Addition des sog. *Exzeß* (=127) berücksichtigt. ($e \in [1, 254]$)

Beispiel:
$2^0 = 2^{(0+127)} - 127$ → e = 127
$2^{-7} = 2^{(-7+127)} - 127$ → e = 120
$2^{15} = 2^{(15+127)} - 127$ → e = 142

fraction (f) : Nachkommaanteil der Mantisse, wobei die Zahl so normiert wird, daß genau eine 1 links neben dem Komma steht.

Beispiel:
$1101011.101 = \mathbf{1}.101011101 * 2^6$
$0.001011011 = \mathbf{1}.011011 * 2^{-3}$

ÜBUNG: Gleitkommazahlen

Geben Sie zu folgenden dezimalen Gleitkommazahlen die entsprechende binäre Darstellung (im Floatingpointformat nach IEEE Standard 754) an:

12.75_D , -128.8_D

Geben Sie zu folgenden Gleitkommazahlen (im Floatingpointformat nach IEEE Standard 754) die entsprechenden dezimalen Gleitkommazahlen an:

1 01111011 00000000000000000000_B

0 10000100 00101100000000000000_B

….. Gleitkommazahlen : Zahlenbereich

Zahlenbereich:

größte normalisierte Zahl (**Anm.:** e darf nicht 0 oder 255 sein !)

0 11111110 11111111111111111111111 = 2^{127} * 1,999999881
= 3.40282347 * 10^{38}

kleinste pos. normalisierte Zahl (Anm.: e_{Min} = 1 = (-126+Exzeß))

0 00000001 00000000000000000000000 = 2^{-126} * 1,00000000
= 1.17549435 * 10^{-38}

Sonderfälle:

e = 0 und f = 0	→ Zahl ist +/-0
e = 0 und f ≠ 0	→ Zahl ist denormalisiert
e = 255 und f = 0	→ Zahl ist +/- unendlich
e = 255 und f ≠ 0	→ Zahl ist „not a number“ (-nan)

Zu beachten auch in C und Java:

Abbruch von Schleifen NIE auf Gleichheit prüfen

ENDE EINSCHUB ZAHLENDARSTELLUNG