Einführung in die Programmierung mit ANSI-C

Übungen 1 – Grundlagen der Programmerstellung

Sommersemester 2015- Dr. Henry Herper

Imperative und objektorientierte Programmiersprachen – C/C++/C#

C (1972), C++ (1980), C# (2001)

Anwendungsgebiet: Systemprogrammiersprache, Echtzeitprogrammierung

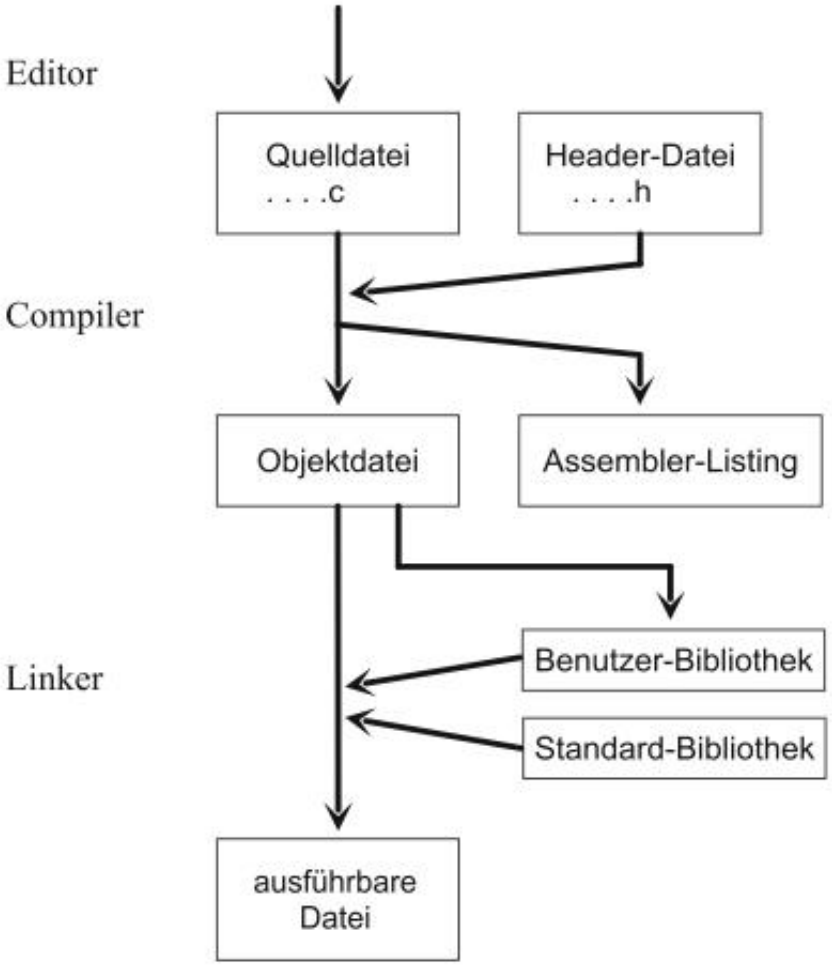
Sprachklasse: prozedural (imperativ), objektorientiert (ab C++)

Die Programmiersprache C wurde in den Bell Laboratories von Dennis Ritche entwickelt.

Die Sprache ist blockorientiert und erlaubt eine sehr kompakte Schreibweise. Ein C-Programm ist eine Menge von Funktionen, die einander aufrufen können.

Es wurde in den frühen achtziger Jahren zur Version C++ weiterentwickelt. Damit steht ein Werkzeug für die Entwicklung großer Softwaresysteme zur Verfügung. C++ besitzt leistungsfähige Komponenten zur objektorientierten Programmierung.

Erstellung und Übersetzung eines C-Programms



Komponentenbasierte Softwareentwicklung

traditionelle Softwareentwicklung

- aus wenigen Sprachelementen werden komplexe Strukturen erstellt
- Applikationen können sehr effektiv implementiert werden, aber sehr großer Aufwand, da alle Elemente neu erstellt werden müssen

komponentenbasierte Softwareentwicklung

- aus vielen komplexen Komponenten werden die geeigneten ausgewählt
- Softwareprodukte werden größer, aber komplexe Applikationen können schneller entwickelt werden

Ereignisorientierte Softwareentwicklung

Moderne Anwendungen auf graphischen Oberflächen arbeiten ereignisorientiert. **Anwendungen reagieren auf Ereignisse**. Diese Ereignisse können vom Betriebssystem, von anderen Anwendungen oder von der Anwendung selbst erzeugt werden.

Visual C++ – Grundlagen – Editor

Die **maximale Zeilenlänge**, die vom Editor verarbeitet werden kann, beträgt **256 Zeichen**. Im Interesse der Lesbarkeit sollten aber nur Zeilen mit einer Länge von 80 Zeichen verwendet werden. Ein Zeilenumbruch ist an beliebigen Positionen des Quelltextes (außer in Zeichenketten) möglich.

Visual C++ – Programmstruktur

Häufig werden in den ersten Zeilen eines Programms **Preprozessor -Anweisungen** zusammengefasst. Diese beginnen mit einem #. Mit der Anweisung

#include <dateiname.h> bzw.

#include “dateiname.h“

kopiert der Preprozessor die angegebene Datei an diese Stelle in den Quellcode. Damit stehen dem Programm alle Informationen zur Verfügung, die in dieser Datei enthalten sind. Es kann zum Beispiel die Header-Datei stdio.h eingebunden werden, die die Standardvereinbarungen für die Ein- und Ausgabe mit Standardfunktionen enthält.

Visual C++ – Konsolenanwendung

```
#include "stdafx.h"


int _tmain(int argc, _TCHAR* argv[])
{
      return 0;
}
```

ANSI-C – Programmstruktur

Der Ausführungsteil einer Funktion besteht aus einer ***Verbundanweisung***, deren Beginn und das Ende durch geschweifte Klammern {} markiert sind. Verbundanweisungen können geschachtelt werden. Jede Ausdrucksanweisung wird durch ein ***Semikolon*** abgeschlossen.

In einem Programm können ***Kommentare*** verwendet werden. Ein Kommentar beginnt mit /* und wird mit */ abgeschlossen. Kommentare können sich über mehrere Zeilen erstrecken, dürfen jedoch nicht geschachtelt werden.

Einzelne Zeilen werden mit // auskommentiert.

ANSI-C – Grundlagen - Bezeichner

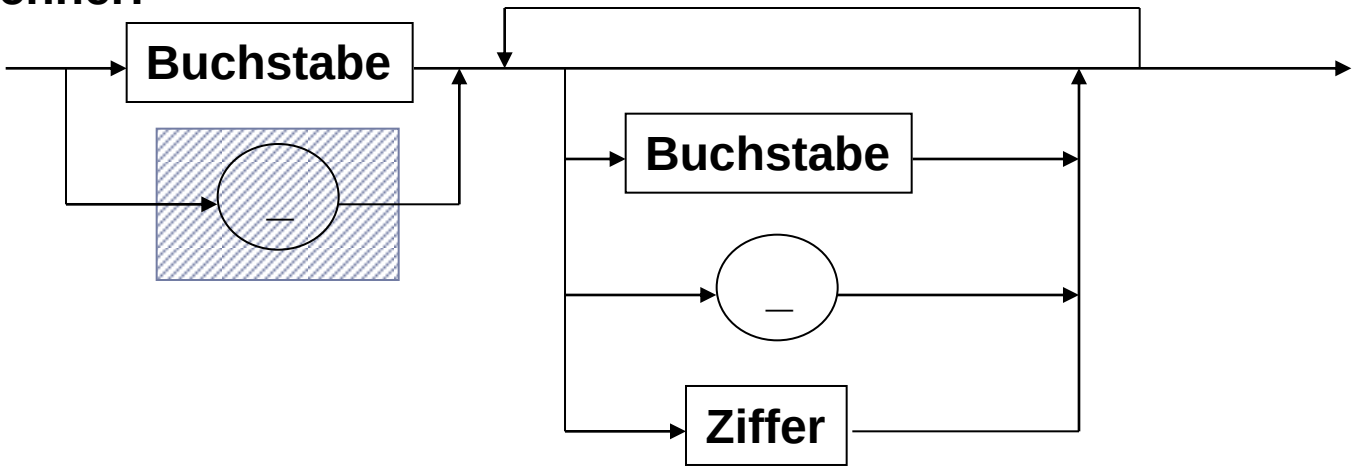
Der ***Zeichenvorrat*** in C umfasst die Großbuchstaben A..Z, die Kleinbuchstaben a..z, die Ziffern 0..9 und Sonderzeichen. In der Regel werden **Groß- und Kleinbuchstaben** strikt **unterschieden**. Quelltext wird normalerweise in Kleinbuchstaben geschrieben.

Aus den Zeichen werden ***Namen bzw. Bezeichner*** zusammengesetzt. Jeder Name muss mit einem Buchstaben beginnen. Das Underline-Zeichen (_) wird dabei wie ein Buchstabe verwendet.

Ein Name kann beliebig lang sein. Einige C-Implementierungen unterscheiden jedoch nur die ersten acht Zeichen. Die meisten Implementierungen unterscheiden jedoch die ersten 31 Zeichen (lt. ANSI-Standard). Da der Linker in der Regel nur die ersten acht Zeichen eines Namens beachtet, sollten Namen für globale Objekte so gewählt werden, dass die **ersten acht Zeichen signifikant** sind.

ANSI-C – Grundlagen – Bezeichner

Bezeichner:



ANSI-C – Grundlagen – Bezeichner

Bestimmte Bezeichner, die ***reservierten Worte bzw. Schlüsselworte***, haben eine standardadisierte und vorausbestimmte Bedeutung. Sie können vom Programmierer nicht als Bezeichner verwendet werden. ***Reservierte Worte müssen immer klein geschrieben werden!***

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

Ausgabe von Daten

C enthält keine Schlüsselwörter zur Ein- und Ausgabe von Daten. Es stehen aber Routinen und Funktionen in Standardbibliotheken zur Verfügung. Dazu ist es notwendig, diese über Compilerdirektiven die Datei einzubinden. Ein Beispiel ist die Datei <stdio.h>, deren Routinen im folgenden vorgestellt werden.

Ausgabe von Daten – ganze Zahlen

Typ	Argumententyp	Darstellung
c	char oder int (<=255)	einzelnes Zeichen
d	int	vorzeichenbehafteter Dezimalwert
u	unsigned int	dezimal
o	unsigned int	oktal
x	unsigned int	hexadezimal mit a,b,c,d,e,f
X	unsigned int	hexadezimal mit A,B,C,D,E,F

Ist ein Argument vom Datentyp *long* oder *unsigned long*, so muss der Typangabe ein l vorangestellt werden.
Der in der printf()-Funktion angegebene Formatstring kann *Formatelemente* enthalten ,die die Ausgabe steuern. Formatelemente beginnen immer mit dem Prozentzeichen % und enden stets mit einer Typangabe. Die Typangabe legt fest, wie der angegebene Wert interpretiert wird.

Ausgabe von Daten

Weiterhin wird mit den Formatelementen die *Breite des Ausgabefeldes* festgelegt. Die Zahl vor der Typangabe gibt die Länge des Ausgabefeldes an. Standardmäßig erfolgt die Ausgabe in dieses Feld rechtsbündig. Wird der Feldbreite ein Minus vorangestellt, so erfolgt die Ausgabe linksbündig.
Ist die auszugebende Zeichenfolge länger als die angegebene Feldbreite, so wird die Zeichenkette *nicht abgeschnitten*, sondern die Formatlänge ignoriert.
Beispiele:

```
%5d    %8x
```

Ausgabe von Daten - Gleitkommazahlen

Typ	Argumententyp	Darstellung
f	float / double	Gleitpunktzahl [-] ddd.dd
e E	float / double	Exponenten-Schreibweise [-] d.dd e±dd [-] d.dd E±dd
g G	float / double	wie e, E oder f, je nachdem was kürzer ist
s	String	Ausgabe bis ‘\0’

Für die Ausgabe des Typs *long double* wird den Typenangaben f, e oder g der Buchstabe L vorangestellt.
Auch bei der *Ausgabe von Gleitpunktzahlen* kann die Breite des Ausgabefeldes festgelegt werden. Zusätzlich kann durch einen Dezimalpunkt die Länge der Mantisse angegeben werden. Die angegebene Ausgabegenauigkeit wird durch Rundung erzeugt.

Ausgabe von Daten - Strings

Bei der *Ausgabe von Strings* kann man neben der Ausgabefeldbreite angeben, wie viele Zeichen maximal ausgegeben werden sollen. Die Angabe erfolgt analog zur Mantissenlänge der Gleitkommazahlen. Standardmäßig erfolgt die Ausgabe rechtsbündig, durch das Voranstellen des Minuszeichens kann eine linksbündige Ausgabe erfolgen.

Ausgabe von Daten

```
#include <stdio.h>
int main()
{
        float wert1 = 1234.5678;
        float wert2 = 1e5;
        int   wert3 = 8765;
        int   wert4 = 6789;
        char  zeichen = 'X';
        printf("\n TESTAUSGABE \n");
        printf("----0----|----1----|----2----|----3----|\n");
        printf("\n wert1 : %f   wert1 : %5.2f\n", wert1,wert1);
        printf("%15.4f \n%15.4f \n", wert1,wert2);
        printf(" wert3 dezimal     %12d \n       oktal
%12o\n", wert3, wert3);
        printf("       hexadezimal %12X \n\n", wert3);
        printf(" <%-10d><%10d>\n\n", wert4, wert4);
        printf(" Zeichen %4c\n\n", zeichen);
        printf(" TESTENDE \n");
        return 0;
}
```

Aufgaben

1. Erweitern Sie das Programm um eine weitere Variable mit dem Name variable2
2. Lesen Sie auch für diese Variable einen Wert ein
3. Geben Sie diesen Wert auch auf die Konsole aus
4. Definieren Sie vier weitere Variable mit dem Namen "ergebnis1>4" vom gleichen Typ integer
5. Realisieren Sie nacheinander die Grundrechenarten mit den Operatoren +, -, *, /, indem Sie die Eingangsvariablen miteinander verknüpfen und das Ergebnis den entsprechenden Ergebnisvariablen zuweisen
6. Geben Sie diese Ergebnisvariablen auf dem Bildschirm wieder mit Kommentar aus
7. Was passiert, wenn Sie statt gerader Zahlen Kommawerte an der Konsole angeben?
8. Was geschieht bei der Division, wenn gebrochene Werte zu erwarten sind?
9. Wechseln Sie vom Datentyp integer auf den Typ float