Einführung in die Programmierung

Grundbegriffe

Prof. Dr. Peter Jüttner

Inhalt

- **Grundbegriffe**
  - Grundstruktur eines C Programms
  - Variable
  - Konstante
  - Typen
  - Ein-/Ausgabe
  - Operatoren
  - Kontrollstrukturen
- **Komplexe Elemente**
  - Weitere Typen
  - Pointer
  - Funktionen

Kontrollstrukturen

Hochschule Deggendorf

bisher ...

- lineare Abläufe, d.h. eine Anweisung wird nach der anderen ausgeführt (linearer Kontrollfluss)
- Unterscheidungen im Ablauf nur durch ?: Operator

➔ wenig Flexibilität

➔ wiederholte Abläufe mehrfach zu codieren

Kontrollstrukturen

Hochschule Deggendorf

Grundsätzliches

- flexible Gestaltung des Programmablaufs
- Programmablauf steuerbar durch Ereignisse, z.B. (eingelesene) Werte von Variablen
- wiederholte Abläufe einfach programmierbar
- kompakte Programme
- übersichtliche Programme

Kontrollstrukturen

Hochschule Deggendorf

Überblick

- if ... else
- switch ... case ...
- while ...
- do ... while ...
- for ...
- goto

Kontrollstrukturen

Hochschule Deggendorf

if ... else ... Verzweigung

- Syntax: if (<Bedingung>) <Anweisungsblock> else <Anweisungsblock> wobei
  - <Anweisungsblock> eine einzelne Anweisung oder eine durch { ... } geklammerte Folge von Anweisungen ist
  - der „else-Zweig“ (else <Anweisungsblock>) auch komplett entfallen kann

Kontrollstrukturen

if ... else ... Verzweigung

- Motivation: Der Programmablauf soll abhängig von einer zur Laufzeit des Programms überprüften Bedingung in einen von zwei möglichen Abläufen verzweigen.

Kontrollstrukturen

if ... else ... Verzweigung

- Syntax: if (<Bedingung>) <Anweisungsblock> else <Anweisungsblock>
  - falls <Bedingung> wahr (d.h. Wert ungleich 0) wird der Anweisungsblock nach <Bedingung> ausgeführt, sonst der Anweisungsblock nach dem else
  - falls der else-Zweig leer ist und die Bedingung falsch (d.h. Wert gleich 0) wird nicht ausgeführt

Kontrollstrukturen

Hochschule Deggendorf

if ... else ... Verzweigung

```
#include <stdio.h>

int main(void)
{
      int a, b, min;

      printf("Bitte Wert fuer a eingeben:\n");
      scanf("%d", &a);

      printf("Bitte Wert fuer b eingeben:\n");
      scanf("%d", &b);

      if (a<b) min = a; else min = b;

      printf("Das Minimum von %d und %d ist %d\n", a, b, min);

};
```

Kontrollstrukturen

Hochschule Deggendorf

if ... else ... Verzweigung

- Bedingung muß immer mit (...) geklammert sein
- if ... else Verzweigungen können beliebig geschachtelt werden, d.h. die Anweisungsblöcke nach der Bedingung und nach else dürfen wiederum if Anweisungen (und andere Kontrollstrukturen) enthalten
  ➔Klammerung beachten

Kontrollstrukturen

Hochschule Deggendorf

if ... else ... Verzweigung

```
#include <stdio.h>

int main(void)
{
      int a, b, min;
      printf("Bitte Wert fuer a eingeben:\n"); scanf("%d", &a);
      printf("Bitte Wert fuer b eingeben:\n"); scanf("%d", &b);

      if (a<b) { if (a>0)
                     min = a;
                  else min = -a; }
      else { if (b>0) min = b;
               else min = -b; }

      printf("Der Betrag des Minimums von %d und %d ist %d\n", a, b, min);
};
```

Kontrollstrukturen

Hochschule Deggendorf

if ... else ... Verzweigung

```
#include <stdio.h>

int main(void)
{
      int a, b, min;
      printf("Bitte Wert fuer a eingeben:\n"); scanf("%d", &a);
      printf("Bitte Wert fuer b eingeben:\n"); scanf("%d", &b);

      if ((a<b) && (a<0)) min = -a;
      if ((a<b) && (a>=0)) min = a;
      if ((a>=b) && (b<0)) min = -b;
      if ((a>=b) && (b>=0)) min = b;

      printf("Der Betrag des Minimums von %d und %d ist %d\n", a, b, min);
};
```

Kontrollstrukturen

Hochschule Deggendorf

if ... else ... Verzweigung

- Unterschied zum ?: Operator:
  - ?: Operator liefert abhängig von der Bedingung ein Ergebnis zurück (entweder des Ergebnis des Ausdrucks nach ? oder das Ergebnis des Ausdrucks nach dem :)
  - if ... else führt abhängig von der Bedingung einen Anweisungsblock aus. Es wird kein (!) Ergebnis zurückgeliefert, d.h.

    min = if (a<b) a; else b; ist syntaktisch nicht korrekt

    min = (a<b) ? a : b; OK

Kontrollstrukturen

Hochschule Deggendorf

Zum Schluss dieses Abschnitts ...

Noch Fragen ??

Kontrollstrukturen

switch ... case ... Verzweigung

- Motivation: Der Programmablauf soll abhängig von einem zur Laufzeit des Programms entstehenden ganzzahligen Wert in einen von mehreren möglichen Abläufen verzweigen

Kontrollstrukturen

switch ... case ... Verzweigung

- Syntax:

```
switch (Wert)
{
  case Wert1 : <Anweisungsblock>

  case Wert2 : <Anweisungsblock>

  ...

  case Wertn : <Anweisungsblock>

  default : <Anweisungsblock>
}
```

Kontrollstrukturen

switch ... case ... Verzweigung

- Syntax:

```
switch (Wert)
{ case Wert_1 : <Anweisungsblock>
  case Wert_2 : <Anweisungsblock>
  ...
  case Wert_n : <Anweisungsblock>
  default : <Anweisungsblock>
} wobei
```

  - <Anweisungsblock> eine einzelne Anweisung oder eine durch { ... } geklammerte Folge von Anweisungen ist
  - der „default-Zweig“ (default: <Anweisungsblock>) kann auch entfallen kann (sollte aber nicht!)
  - $Wert_1$, $Wert_2$, ..., $Wert_n$ müssen Konstante sein, Wert von einem ganzzahligen Typ (char, short, long, ...)

Kontrollstrukturen

switch ... case ... Verzweigung

- Syntax:

```
switch (Wert)
{ case Wert1 : <Anweisungsblock1>; break;
  case Wert2 : <Anweisungsblock2>; break;
  ...
  case Wertn : <Anweisungsblockn>; break;
  default : <Anweisungsblockd> ; break;
}
```

  - Wert nach switch wird nacheinander mit $Wert_1$, $Wert_2$, ..., $Wert_n$ verglichen. Bei Gleichheit wird der folgende Code, beginnend beim entsprechenden Anweisungsblock ausgeführt.
  - Trifft keiner der Werte $Wert_1$, $Wert_2$, ..., $Wert_n$ mit Wert überein, so wird der Anweisungsblock nach default ausgeführt.

Kontrollstrukturen

switch ... case ... Verzweigung

- Syntax:

```
switch (Wert)
{ case Wert_1 : <Anweisungsblock_1>; break;
   ...
  case Wert_n : <Anweisungsblock_n>; break;
  default : <Anweisungsblock_d>; break;
}
```

  - Achtung: Bei Gleichheit von Wert mit eine $Wert_i$ werden die Anweisungsblöcke i, i+1, ..., n und d ausgeführt.

  - Ist diese nicht gewünscht, so muss ein Anweisungsblock mit break abgeschlossen werden. (Fortsetzung des Ablaufs nach der switch ... case ... Verzweigung)

Kontrollstrukturen

Hochschule Deggendorf

switch ... case ... Verzweigung

```
#include <stdio.h>

int main(void)
{       char k;

        printf("Bitte eins der Kommando 'A', 'B', 'C' eingeben:\n");
        scanf("%c", &k);

        switch (k)

        {  case 'A': printf("Komamndo A wurde eingegeben\n"); break;

           case 'B': printf("Komamndo B wurde eingegeben\n"); break;

           case 'C': printf("Komamndo C wurde eingegeben\n"); break;

           default: printf("ein falsches Komamndo wurde eingegeben\n");
        }:
};
```

Kontrollstrukturen

Hochschule Deggendorf

switch ... case ... Verzweigung

- soll ein Anweisungsblock bei mehreren Werten ausgeführt werden, so können nacheinander mit case mehrere Werte angegeben werden, z.B

```
switch (k)
{
  case ‘A‘:
  case ‘a‘: printf(“Kommando A wurde eingegeben\n“); break;
  ...
}
```

- Der Anweisungsblock nach : darf beliebige weitere Kontrollstrukturen enthalten.

Kontrollstrukturen

switch ... case ... Verzweigung

```
#include <stdio.h>

int main(void)
{       char k;

        printf("Bitte eins der Kommando 'A', 'B', 'C' eingeben:\n");
        scanf("%c", &k);

        switch (k)

        {  case 'A':
           case 'a': printf("Komamndo A wurde eingegeben\n"); break;

           case 'B':
           case 'b': printf("Komamndo B wurde eingegeben\n"); break;

           ...
        }:
};
```

Kontrollstrukturen

Hochschule Deggendorf

switch ... case ... Verzweigung

- Programmierstil:
  - switch case nur für wenige diskrete Werte verwenden
  - alle im Normallfall vorkommenden Werte durch eigenen case abdecken
  - default: ist „nur“ eine Art „Notausgang“ für außergewöhnliche Fälle oder falls ein Fall vergessen wurde
  - default kann, sollte aber nicht weggelassen werden

Kontrollstrukturen

Hochschule Deggendorf

Zum Schluss dieses Abschnitts ...

Noch Fragen ??

Kontrollstrukturen

while ... Schleife

- Motivation: Ein bestimmter Programmteil soll abhängig von einer zur Laufzeit ausgewerteten Bedingung immer wieder ausgeführt werden (Bedingung vor erster Ausführung überprüfen)

Kontrollstrukturen

Hochschule Deggendorf

while ... Schleife

- Syntax: while (Bedingung) <Anweisungsblock>
  wobei
  - <Anweisungsblock> eine einzelne Anweisung oder eine durch { ... } geklammerte Folge von Anweisungen ist
  - Bedingung muss immer in (...) geklammert sein
  - Der Anweisungsblock nach (Bedingung) darf beliebige weitere Kontrollstrukturen enthalten.

Kontrollstrukturen

Hochschule Deggendorf

while ... Schleife

- Syntax: while (Bedingung) <Anweisungsblock>
  wobei
  - Der Anweisungsblock wird immer wieder ausgeführt, so lange die Bedingung wahr ist (ungleich 0). Vor jeder Ausführung, auch vor der ersten wird die Bedingung ausgewertet.
  - Ist Bedingung vor der ersten Ausführung bereits falsch (gleich 0) wird der Anweisungsblock gar nicht ausgeführt.

Kontrollstrukturen

Hochschule Deggendorf

while ... Schleife

```
#include <stdio.h>
/* Multiplikation zweier Zahlen > 0 */
int main(void)
{      unsigned int a, b, c, erg;

       printf("Bitte erste Zahl (>0) eingeben:\n");;
       scanf("%d", &a);
       printf("Bitte zweite Zahl (>0) eingeben:\n");;
       scanf("%d", &b);

       erg = 0; c = a;
       while (a>0)
       {  erg = erg + b;
          a = a – 1;
       };

       printf("%d mal %d ergibt %d\n", c, b, erg);
};
```

Kontrollstrukturen

Hochschule Deggendorf

Zum Schluss dieses Abschnitts ...

Noch Fragen ??

Kontrollstrukturen

do ... while Schleife

- Motivation: Ein bestimmter Programmteil soll abhängig von einer zur Laufzeit ausgewerteten Bedingung immer wieder ausgeführt werden (Bedingung erstmalig nach erster Ausführung überprüfen)

Kontrollstrukturen

Hochschule Deggendorf

do … while Schleife

```
#include <stdio.h>
/* Multiplikation zweier Zahlen > 0 */
int main(void)
{      unsigned int a, b; int erg;

       printf("Bitte erste Zahl (>0) eingeben:\n");;
       scanf("%d", &a);
       printf("Bitte zweite Zahl (>0) eingeben:\n");;
       scanf("%d", &b);

       erg = – b;
       do
       {  erg = erg + b;
          a = a – 1;
       }
       while (a>=0);
       printf("%d mal %d ergibt %d\n", a, b, erg);
};
```

Kontrollstrukturen

Hochschule Deggendorf

do ... while Schleife

- Syntax: do <Anweisungsblock> while (Bedingung)
  wobei
  - Der Anweisungsblock wird immer wieder ausgeführt, so lange die Bedingung wahr ist (ungleich 0). Nach jeder Ausführung, auch nach der ersten wird die Bedingung ausgewertet.
  - Der Anweisungsblock wird mindestens einmal ausgeführt, auch wenn die Bedingung immer falsch ist.

Kontrollstrukturen

Hochschule Deggendorf

do ... while Schleife

falscher Algorithmus

```
#include <stdio.h>

int main(void)
{      unsigned int a, b, erg;

       printf("Bitte erste Zahl (>0) eingeben:\n");;
       scanf("%d", &a);
       printf("Bitte zweite Zahl (>0) eingeben:\n");;
       scanf("%d", &b);

       erg = 0;
       do
       {  erg = erg + b;
          a = a – 1;
       }
       while (a>0)
       printf("%d mal %d ergibt %d\n", a, b, erg);
};
```

Kontrollstrukturen

Hochschule Deggendorf

do ... while Schleife

- Unterschied zu while-Schleife beachten (while-Schleife prüft vor erster Ausführung, ob Bedingung für Ausführung gültig)

  ➔While Schleife ist etwas „sicherer“

  ➔While Schleife bevorzugen (Programmierstil)

Kontrollstrukturen

Hochschule Deggendorf

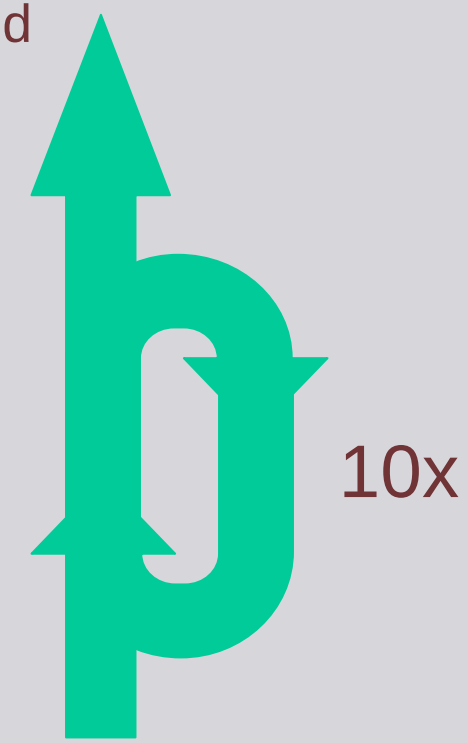
Zum Schluss dieses Abschnitts ...

Noch Fragen ??

Kontrollstrukturen

for ... Schleife

- Motivation: Ein bestimmter Programmteil soll eine bestimmte Anzahl von gezählten Durchläufen ausführen. Dabei steht die Anzahl der Durchläufe vor dem ersten Durchlauf bereits fest und wird während der Durchläufe nicht verändert.
  Zum Zählen der Durchläufe wird dabei eine eigene Variable (Schleifenzähler) verwendet



Kontrollstrukturen

Hochschule Deggendorf

for ... Schleife

- Syntax: for (<Start>;
  <Abbruchbedingung>;
  <Änderung>)
  <Anweisungsblock> wobei
  - <Start> eine Anweisung ist, die den Schleifenzähler initialisiert wird
  - <Abbruchbedingung> eine Bedingung ist, die vor jedem Durchlauf geprüft wird. Der Durchlauf findet nur statt, falls die Bedingung wahr (ungleich 0) ist
  - <Änderung> eine Anweisung ist, die den Schleifenzähler bei jedem Durchlauf verändert

Kontrollstrukturen

for ... Schleife

- Beispiel:

```
int i;
int j = 5;
int erg;

for (i = 1; i<=10; i++)

{ erg = j*i;
  printf("%d mal %d ist %d\n", j, i, erg);
};
```

Startwert für i ist 1

Schleife wird so lange ausgeführt bis i>10

Nach jedem Schleifendurchlauf wird i um 1 erhöht

Kontrollstrukturen

Hochschule Deggendorf

for ... Schleife

```
#include <stdio.h>

int main(void)
{      unsigned int a, b, erg;

       printf("Bitte erste Zahl (>0) eingeben:\n");;
       scanf("%d", &a);
       printf("Bitte zweite Zahl (>0) eingeben:\n");;
       scanf("%d", &b);

       erg = 0;
       int i;
       for (i = a; i>0; i--)
         erg = erg + b;

       printf("%d mal %d ergibt %d\n", a, b, erg);
};
```

Kontrollstrukturen

Hochschule Deggendorf

for ... Schleife

- Verboten ist es

  den Schleifenzähler innerhalb des Rumpfs der for-Schleife zu verändern!

  z.B.

```
for (int i=0; i<10; i++)
{
  ...
  i = i -2;
  ...
};
```

Kontrollstrukturen

Hochschule Deggendorf

for ... Schleife

- Verboten ist es

  den Schleifenzähler innerhalb des Rumpfs der for-Schleife zu verändern!

  Sollte das „notwendig“ sein, andere Schleifenform (while, do while) wählen.

Kontrollstrukturen

Hochschule Deggendorf

Zum Schluss dieses Abschnitts ...

Noch Fragen ??

Kontrollstrukturen

Hochschule Deggendorf

Vorzeitiges Beenden von Schleifen oder Verlassen von Kontrollstrukturen

- break; Beendet Schleifenbearbeitung, Fortsetzung nach der Schleife (bzw. Kontrollstruktur)
- continue; Beendet aktuelle Schleifenausführung, es wird mit der Überprüfung der Bedingung fortgesetzt
- return; im main(), beendet Programmausführung
- exit; beendet Programmausführung

Kontrollstrukturen

Hochschule Deggendorf

Zum Schluss dieses Abschnitts ...

Noch Fragen ??

Kontrollstrukturen

Hochschule Deggendorf

goto

- Syntax: goto Sprungmarke; wobei
  - Sprungmarke eine Bezeichner ist, bei dem die Ausführung des Programms fortgeführt werden soll.

Kontrollstrukturen

Hochschule Deggendorf

while ... Schleife

```
#include <stdio.h>
/* Multiplikation zweier Zahlen > 0 */
int main(void)
{     unsigned int a, b, c, erg;

      printf("Bitte erste Zahl (>0) eingeben:\n");;
      scanf("%d", &a);
      printf("Bitte zweite Zahl (>0) eingeben:\n");;
      scanf("%d", &b);

      erg = 0;
      Anfang: if (a<=0) goto Ende;
      erg = erg + b;
      a = a – 1;
      goto Anfang;

      Ende: printf("%d mal %d ergibt %d\n", c, b, erg);
};
```

Kontrollstrukturen

goto

- Anmerkungen zum goto
  - „considered harmful“ (Dykstra, ca. 1970)
  - Verwendung führt zu unübersichtlichem, schlecht verständlichem und schlecht wartbarem Code
  - eigentlich verboten (!)

Kontrollstrukturen

Hochschule Deggendorf

Zum Schluss dieses Abschnitts ...

Noch Fragen ??