Fachhochschule Bingen

Programmieren

Prozeduren und Funktionen: Prototypen / Call by Reference

Prof. Dr. Maximilian Mengel,
Professur Programmiermethodik,
Grundlagen der Informatik und Multimedia
Gebäude 1, Raum 212
Tel.: 06721-409 152
E-Mail: mengel@fh-bingen.de

Deklaration einer Funktion

- Bevor eine Funktion genutzt wird muß diese „bekannt“ sein
  - Möglichkeit 1: „Richtige“ Reihenfolge
  - Möglichkeit 2: Deklaration per „Prototyping“
    - Die Schnittstelle der Funktion wird vereinbart ohne jedoch den Funktionsblock zu implementieren
    - Eine Funktion wird deklariert indem der Funktionskopf gefolgt von einem Semikolon aufgeschrieben wird:

      ```
      double kehrw_fak(int n);
      ```

    - Die Implementierung erfolgt an anderer Stelle im Programm

Beispiel

```
#include <stdio.h>
double dmax(double a, double b);

main()
{
printf(“Maximum(3.5,7.2) = %f\n“,dmax(3.5,7.2));
}

double dmax(double a, double b)
{
if (a>b)
  return a;
else
  return b;
}
```


Geltungsbereich von Variablen

- Variablen die innerhalb einer Funktion deklariert werden sind nur innerhalb der Funktion gültig
- Variablen die außerhalb von Funktionen deklariert sind sind sogenannte „globale Variablen“
  - Auf globale Variablen kann immer zugegriffen werden (in jeder Funktion)
- **Achtung**: lokale Variablen überdecken globale Variablen

Beispiel

```
#include <stdio.h>

double erg;

void dmax(double a, double b);

main()
{
dmax(3.5,7.2);
printf("Maximum(3.5,7.2) = %f\n",erg);
}

void dmax(double a, double b)
{
if (a>b)
  erg = a;
else
  erg = b;
}
```


Übergabeverfahren von Parametern

- Parameter können prinzipiell auf zwei verschiedene Arten übergeben werden
  - Call-by-Value
  - Call-by-Reference

Call-by-Value

- Die Zuordnung zwischen einer übergebenen Variablen erschöpft sich darin, daß vor Beginn der Ausführung des Funktions- bzw. Prozedurblocks der Wert der Variable dem entsprechenden formalen Parameter zugewiesen und damit dieser initialisiert wird
- Es ist somit **nicht** möglich, eine Wirkung der Ausführung (also eine Wertänderung von der aufgerufenen Variable oder des Ausdruckes) nach außen zu geben.

Call-by-Value

- Vorteile:
  - Der Parameterwert wird vor Funktions-/Prozedurausführung bestimmt
  - Das Übergabeverfahren kann durch einige wenige Maschinenbefehle erledigt werden
  - Veränderungen des Wertes des Parameter haben keine Auswirkung über die Ausführungszeit der Funktion/Prozedur
  - Im Funktionskörper kann der formale Parameter wie eine lokale Variable benutzt werden

Call-by-Value

- Nachteile:
  - Der formale Parameter belegt separaten Speicherplatz; dies ist relevant, wenn der Speicherbedarf groß ist
  - Der Wert des Parameters wird auf jeden Fall ermittelt, auch wenn er in einem bestimmten Ausführungsfall wegen einer gegebenen Datensituation nicht benötigt wird
  - Das Kopieren des Wertes kann aufwendig sein
  - Resultate können nicht nach außen gelangen

Call-by-Reference

- Die übergebenen Variablen verschmelzen förmlich mit den formalen Parametern und werden statt ihrer referenziert. Dies ist nur möglich, wenn es sich beim Aufruf um Variablen handelt.
- Dementsprechend bleiben sämtliche Veränderungen im aufrufenden Programm nach Ausführung der Funktion/Prozedur sichtbar.

Call-by-Reference

- Vorteile:
  - Die Parameterübergabe erfordert keine zusätzlichen Maschinenoperationen
  - Es wird kein zusätzlicher Speicherplatz benötigt
  - Veränderungen können nach außen gelangen
  - Wert des aktuellen Ausdrucks für den Parameter wird nur ermittelt, wenn er benötigt wird

Call-by-Reference

- Nachteile:
  - Zugriff ist u.U. schwieriger, da es keine lokale Variable der Funktion/Prozedur darstellt
  - Veränderungen während der Ausführung des Funktions-/Prozedurkörpers wirken sich nach außen aus, d.h. das aktuelle Objekt hat nach der Ausführung möglicherweise einen anderen Wert als vorher; es ist vom Anwendungsfall abhängig, ob solche Seiteneffekte erwünscht sind
  - Der Wert des aktuellen Parameters wird ggf. mehrfach ermittelt

C: Call-by-Value

- In C werden Parameter normalerweise immer als Call-by-Value Parameter übergeben:
  - Beispiel:

```
double dmax(double a, double b)
{
  double temp;
  if (a>b)
    {
      temp = a;
      a =b;
      b = temp;
    }
  return b;
}
```


C: Call-by-Reference

- Um eine Variable als Call-by-Reference zu übergeben muß in der diese in der Funktionsvereinbarung mit dem * Operator versehen werden
  - Beispiel

```
double dmax(double *a, double *b);
```

- An der aufrufenden Stelle muß der & Operator benutzt werden
  - Beispiel

```
erg = dmax(&x, &y);
```


C: Call-by-Reference

- Innerhalb der Funktion muß ebenfalls immer der * Operator vor dem by-Reference übergebenen Parameter stehen:

```
double dmax(double *a, double *b)
{
  double temp;
  if (*a>*b)
    {
      temp = *a;
      *a = *b;
      *b = temp;
    }
  return *b;
}
```


Call-by-Reference: Beispiel

```
#include <stdio.h>

void swap(double *a, double *b)
{
  double temp;
  temp = *a;
  *a = *b;
  *b = temp;
}

main()
{
double x=0.5,y=1.5;
swap(&x,&y);
printf("x=%f, y=%f",x,y);
}
```

Datenfelder als Parameter

- Wenn ein Feld als Parameter übergeben werden soll existieren einige Besonderheiten:
  - Felder werden **immer on-Reference** übergeben
  - In dem Funktionskopf sollte die Übergabe eines Feldes analog zur Variablenvereinbarung eines Feldes folgendermaßen erfolgen*:
    - `int mittelwert(int Werte[10]);`
  - Beim Aufruf der Funktion wird einfach nur der Feldname ohne [ ] oder & genannt
    - ```
      int erg, meineWerte[10] = {1,2,3,4,5,5,6,7,8,9};
      erg = mittelwert(meineWerte)
      ```

*Es existieren noch weitere Möglichkeiten
      ```