Fachhochschule Bingen

Programmieren

Prozeduren und Funktionen

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

Prozeduren und Funktionen

- Ziel:
  - Zerlegen eines (komplexen) Problems in mehrere einfache Teilprobleme
  - Mehrfache Verwendung einer (Teil-)Lösung
    - Innerhalb eines Programmes
    - Innerhalb mehrerer Programme
- Grundlage zur modularen Programmierung

Vorgehen

- An einer Stelle des Programms wird die Prozedur bzw. Funktion deklariert und das Verhalten definiert (bzw. implementiert)
  - Eine Prozedur bzw. Funktion muß einen eindeutigen Namen besitzen
  - Eine Prozedur bzw. Funktion muß genau eine Implementierung besitzen
- An anderer/anderen Stellen wird die Prozedur bzw. Funktion aufgerufen
  - Der Aufruf muß exakt so erfolgen wie er bei der Deklaration festgelegt wurde

Prozeduren versus Funktionen

- Funktionen
  - liefern ein Ergebnis
  - können in Ausdrücken aufgerufen werden
    - Beispiel: sin(x) > 0.5
- Prozeduren
  - liefern kein Ergebnis
  - können als Anweisungen aufgerufen werden
    - Beispiel: Ausgabe(Matrix);

Sowohl Prozeduren als auch Funktionen bekommen im allgemeinen Parameter übergeben

Prozeduren und Funktionen in C

- In C wird nicht explizit zwischen Prozeduren und Funktionen unterschieden:
  - In C existieren nur Funktionen
  - Um auch Prozeduren implementieren zu können benutzt man den C-Datentyp void (deutsch „leer“). Von diesem Datentyp können **keine Variablen** instanziiert werden. Er kann jedoch als Rückgabetyp genutzt werden um Prozeduren zu deklarieren.

Parameter

- An Prozeduren und Funktionen können Werte in der Form von Parametern übergeben werden
- Hierbei müssen ...
  - die Anzahl der Parameter,
  - der Datentyp der Parameter und
  - die Reihenfolge der Parameter
- ... genau mit den Angaben der Deklaration übereinstimmen.

Deklaration

- Eine Funktion* besteht aus dem:
  - Funktionskopf
  - Funktionsblock
- Der Funktionskopf definiert die **Schnittstelle** der Funktion:
  - Was ist der Typ des Resultats der Funktion?
  - Welche Parameter werden an die Funktion übergeben?

*** bzw. eine Prozedur**

Der Funktionskopf

- Der Funktionskopf besitzt folgende Syntax*:
  - **<funktionskopf> ::= <Typ> <F-Name>(<Param-liste>)**
    <Param-liste> ::= [<Typ> <P-Name>{, <Typ> <P-Name>}*] | void
  - Beispiele:
    - int fakultaet (int n)
    - void ausgabe(int matrix[3][3])
    - double e_hoch-1(void)
    - double e_hoch-1(double epsilon)

*** I. Allg. sind C-Compiler wesentlich „nachlässiger“ und akzeptieren auch nicht dieser Syntax entsprechende Deklarationen**

Der Funktionsblock

- Der Funktionsblock ist ein „normaler“ Block aus Anweisung, der mit einer öffnenden Klammer { beginnt und einer schließenden Klammer } endet
  - Beispiel (Funktionskopf und Funktionsblock):

```
void matrix_ausgabe(double m[3][3])
{
int i;
for (i=0; i<3; ++i)
  printf(“%.5lf %.5lf %.5lf\n“, m[0][i],m[1][i], m[2][i]);
}
```


Beenden der Funktion

- Eine Funktion kann jederzeit mit der Anweisung **`return`** beendet werden
  - Bespiel:

```
void ausgabe_div(int dividend, int divisor)
{
  if (!divisor)  // entspricht (divisor == 0)
  {
    printf(“Fehler Division durch Null!“);
    return;
  }
  else
  .....
```


Beenden der Funktion / Rückgabewert

- Soll eine Funktion ein Resultat liefern, so muß dieses der Anweisung **`return`** folgen
  - Beispiel:

```
double dabs(double x)
{
    if (x >= 0.0)
       return x;
    else
      return -x;
}
```


Lokale Variablen

- Variablen die innerhalb des Funktionsblocks vereinbart werden (sogenannte lokale Variablen) sind nur innerhalb des Funktionsblocks gültig
- Übergebene Parameter sind wie lokale Variablen benutzbar
- !!! Änderungen der Werte der Parameter haben **keine** Auswirkung an der Aufrufenden Stelle!!!

Beispiele

- Lokale Variablen

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

c = dmax(x,y);
```

```
int sum(int a, int b[])
{
  int sum = 0;
  while (a > 0)
  {
    sum +=b[a-1];
    --a;
  }
  return sum;
}
...
int feld[] = {1,4,2,7,3,9};
int erg, anz=6;
...
erg = sum(anz,feld);
   // Welchen Wert besitzt anz ???
```


Typische Fehler

```
double irgendwas(double a,b) /* Fehler:je 1xTyp/1xParam.*/
{
  double temp,a;     /* Fehler: Namenskonflikt a */
  temp = c;          /* Fehler: Variable c ist unbekannt */
  return;            /* Fehler: kein Rückgabewert */
  printf("Fertig!"); /* Fehler: printf() wird nie erreicht */
}

/* Besser so:*/
double irgendwas2(double a, double b)
{
  double temp;
  temp = a*b;
  printf("Fertig!");
  return temp;
}
```


Die Funktion main()

- Auch main() ist eine Funktion wie alle anderen
- Die korrekte Schnittstelle von main() ist:
  - int main(int argc, char * argv[])
  - Die Funktion main() kann also einen ganzzahligen Rückgabewert besitzen. Dieser wird i.Allg. mittels der Funktion exit(<Wert>) gesetzt
  - Die Funktion besitzt weiterhin folgende Parameter:
    - argc: Anzahl der in argv vorhandenen Strings; mindestens 1, da der Programmname auch dem ersten String entspricht
    - argv: Array von Strings:
      - argv[0] enthält den Programmnamen;
      - argv[1] einen ersten Aufrufparameter,
      - ...

Prozedur ohne / mit Parameter / main()

- Schreiben Sie eine Prozedur Autor(), die Sie in der folgenden Art als Autor benennt:
  - Dieses Programm wurde von <IHR NAME> erstellt!
  - Die Prozedur soll keine Parameter besitzen
- Erweitern Sie die Prozedur Autor() um die Parameter „Autor“, „Programmname“und „Semester“. Die Ausgabe sollte folgendermaßen aussehen:
  - Dieses Programm (<Programmname>) wurde von <Autorenname> im (<Semester>) erstellt.
- Erweitern Sie main() so, dass die für die Prozedur Autor() notwendigen Parameter als Kommando-zeilenparameter übergeben werden. Geben Sie Fehlermeldungen aus wenn die Parameter fehlen!

Funktionen

- Das **geometrische** Mittel zweier positiver Zahlen a, b wird gemäß folgender Formel berechnet:

$$m_g = \sqrt{a * b}$$

- Schreiben Sie eine Funktion die das geometrische Mittel von zwei Zahlen berechnet. Wenn die Zahl a oder b nicht positiv ist soll die Funktion –1.0 als Fehlerwert zurückgeben.

Funktionen

- Um die Länge der Hypotenuse eines rechtwinkligen Dreiecks aus den beiden Katheten zu bestimmen nutzt man den Satz des Pythagoras (a, b = Katheten, c = Hypotenuse):
  - $a^2 + b^2 = c^2$
- Schreiben Sie eine Funktion Pythagoras(a, b) die aus den gegebenen Katheten die Hypotenuse berechnet.

Unterfunktionen

- Lösen Sie folgende (schwere) Aufgabe:
- Schreiben Sie ein Programm zur Ausgabe von Tannenbäumen. Jeder Tannenbaum besteht aus mehreren Tannenebenen (im Beispiel 3). Jede Tannenebene aus mehreren (im Beispiel 3) Reihen Zeichen (im Beispiel Sterne: *). Innerhalb einer Tannenebene enthält jede Reihe zwei Zeichen mehr als die vorhergehende Reihe. Von einer Tannenebene zur nächsten reduziert sich die Anzahl der Zeichen um zwei. Programmieren Sie folgende Funktionen in Ihrem Programm:

```
Reihe(Einrueckung, Zeichen, Zeichenanzahl)
Ebene(Einrueckung, Zeichen, Startbreite, Reihenanzahl)
```

```
    *
   ***
  *****
   ***
  *****
 *******
  *****
 *******
*********
   ###
```