strukturierte Datentypen

Feldtyp (Array)

Deklaration:

```
<ele_type> <ident>[<ele_anzahl>];
```

- `<ele_type>` : Typ **aller** Elemente
- `<ident>` : Bezeichner der Array-Variablen
- `<ele_anzahl>` : Anzahl der Elemente

Feldtyp (Array)

Beispiel:

```
int i, Zahl, Feld[20];
```

Zugriff:

```
Feld[7] = 85;
zahl = Feld[i];
```

strukturierte Datentypen

Strukturtyp (Record)

Deklaration:

struct [<tag_name>] { **...** };

Beispiel:

```
struct person { char name[20];
                char vname[15];
                short alter;
              };
    /* Strukturdefinition */
....
[struct] person single, gruppe[10], *p_ptr;
    /* Variablendefinition */
```

dabei gilt:

- `single` ist einfache Variable vom Typ `struct person`
- `gruppe` ist Array mit 10 Elementen vom Typ `struct person`
- `*p_ptr` ist ein Pointer auf eine Variable vom Typ `struct person`

Zugriff:

```
single.alter = 25;

gruppe[i].alter = 47;


p_ptr = &single;
p_ptr->alter = 32;
```

unbenannte Struktur: ohne `<tag_name>`

Beispiel:

```
struct { char name[20];
         char vname[15];
         short alter;
       } single, gruppe[10], *p_ptr;
```

Variablen wie vorher, es gibt jedoch keinen Struktur-Bezeichner

Funktionen

- Übersichtlichkeit des Programms
- Mehrfachaufruf innerhalb eines Programms (auch mit anderen Parametern)
- Wiederverwendung in anderen Programmen (Bibliotheken)
- nach Definition der Schnittstelle Bearbeitung durch andere
- sollten universell sein, d. h. möglichst keine Operationen enthalten, die von der Programmumgebung abhängen oder diese verändern

Referenzierung und Dereferenzierung

& Referenzoperator (Adress-Operator)

* Dereferenzierungsoperator

Referenzparameter in Funktionen

```
int x = 7, y = 12;
void swap(int a, int b)
 { int h;
   h = a;
   a = b;
   b = h;
  }
```

Aufruf: `swap(x, y);`

Referenzparameter in Funktionen

```
int x = 7, y = 12;
void swap(int *a, int *b)
 { int h;
   h = *a;
   *a = *b;
   *b = h;
  }
```

Aufruf: `swap(&x, &y);`

Array-Parameter in Funktionen

Übergabe von Arrays standardmäßig als Referenzparameter (offenes Array)

```
int max1, max2, feld1[10], feld2[20];
int max(int array[], int anzahl)
 { int i;
   ...
   ...
   return array[i];
  }

  max1 = max(feld1, 10);
  max2 = max(feld2, 20);
```