

Übung für alle mit Papier und Bleistift oder Computer

Implementieren Sie eine C++-Funktion `mit30_40Belegen`, sodass dessen Aufruf im folgenden Programmausschnitt zu der Ausgabe `vz1: 30, vz2: 40` führt.

```
struct Vorgang {  
    double dauer;  
    double fruehanf;  
    double spaetend;  
};
```

```
void funk() {  
    Vorgang* vz1 = nullptr;  Vorgang* vz2 = nullptr;  
    mit30_40Belegen(vz1, vz2);  
    cout << "vz1: " << vz1->dauer << ", vz2: " << vz2->dauer << "\n";  
    delete vz1; vz1 = nullptr; delete vz2; vz2 = nullptr;  
}
```

Übung für alle mit Papier und Bleistift oder Computer

```
void main() {
    Vorgang* vz1 = nullptr;  Vorgang* vz2 = nullptr;
#ifdef SCHON_FERTIG
    vz1 = new Vorgang();    vz2 = new Vorgang();
    vz1->dauer = 30;    vz2->dauer = 40;
#else
    // Implementieren Sie nun Schnittstelle und Code der folgenden Funktion
    // sodass sich die gleiche Ausgabe wie beim direkten Setzen ergibt
    mit30_40Belegen(vz1, vz2);
#endif
    cout << "vz1: " << vz1->dauer << ", vz2: " << vz2->dauer << "\n";
    delete vz1; vz1 = nullptr; delete vz2; vz2 = nullptr;
}
```

```
struct Vorgang {
    double dauer;
    double fruehanf;
    double spaetend;
};
```

Ausgangssituation

```
Vorgang* vz1 = nullptr;  
Vorgang* vz2 = nullptr;
```

```
1000 vz1 0
```

```
1004 vz2 0
```

```
struct Vorgang {  
    double dauer;  
    double fruehanf;  
    double spaetend;  
};
```

Zielsituation:

```
Vorgang* vz1 = nullptr;  
Vorgang* vz2 = nullptr;  
mit30_40Belegen(vz1, vz2);
```

```
1000 vz1 7000
```

```
1004 vz2 4488
```

7000

```
dauer: 30  
fruehanf: xxx  
spaetend: yyy
```

4488

```
dauer: 40  
fruehanf: kk  
spaetend: mm
```

Vereinfachung; ohne Funktion

```
void funk() {  
    Vorgang* vz1 = nullptr;  Vorgang* vz2 = nullptr;  
    vz1 = new Vorgang();    vz2 = new Vorgang();  
    vz1->dauer = 30;  vz2->dauer = 40;  
    cout << "vz1: " << vz1->dauer << ", vz2: " << vz2->dauer << "\n";  
    delete vz1; vz1 = nullptr; delete vz2; vz2 = nullptr;  
}
```

```
struct Vorgang {  
    double dauer;  
    double fruehanf;  
    double spaetend;  
};
```

Es sind somit drei Schritte von der Funktion zu leisten:

1. 2 Vorgänge erzeugen
2. In die Vorgänge die Dauern 30 und 40 eintragen
3. vz1 und vz2 auf die beiden erzeugten Vorgänge zeigen lassen

Nur Schritt 2 wird von der Funktion geleistet

```
struct Vorgang {  
    double dauer;  
    double fruehanf;  
    double spaetend;  
};
```

Es sind somit drei Schritte von der Funktion zu leisten:

1. 2 Vorgänge erzeugen
2. In die Vorgänge die Dauern 30 und 40 eintragen
3. vz1 und vz2 auf die beiden erzeugten Vorgänge zeigen lassen

```
void Mit30_40Belegen(Vorgang* v1, Vorgang* v2){  
    v1->dauer = 30; v2->dauer = 40;  
}
```

```
void funk() {  
    Vorgang* vz1 = nullptr; Vorgang* vz2 = nullptr;  
    vz1 = new Vorgang(); vz2 = new Vorgang();  
    Mit30_40Belegen();  
    cout << "vz1: " << vz1->dauer << ", vz2: " << vz2->dauer << "\n";  
    delete vz1; vz1 = nullptr; delete vz2; vz2 = nullptr;  
}
```



Auch Schritt 1 und 3 in die Funktion verlagert

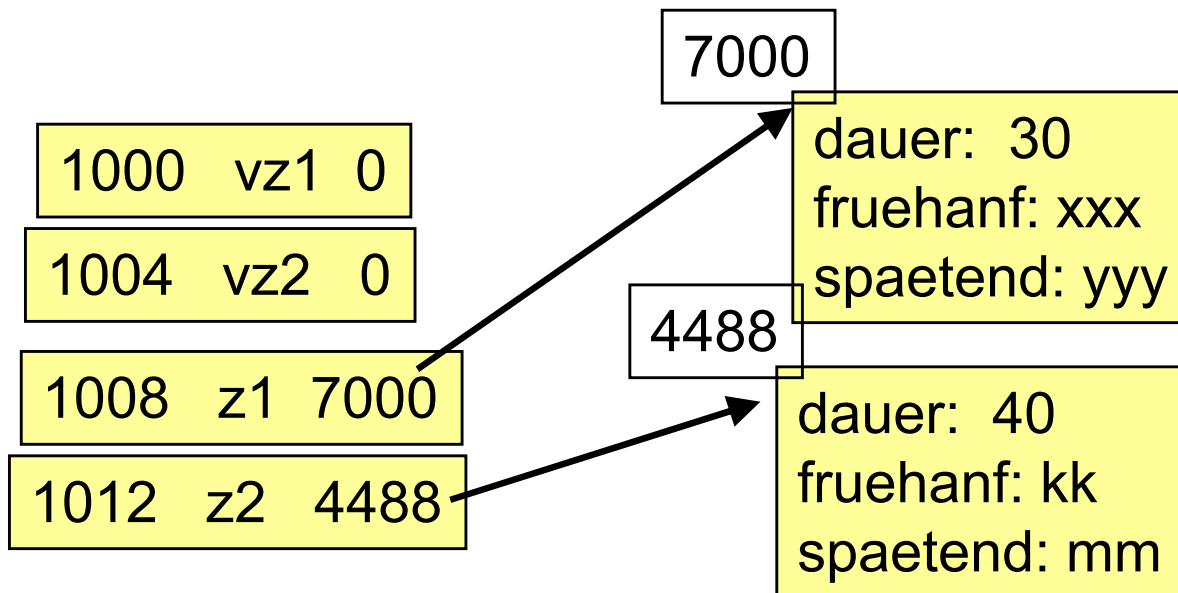
```
/** Die Funktion fordert Speicher auf dem Heap für z1 und z2 an und belegt
    die Vorgangsdauern dann mit 30 bzw. 40. */
void mit30_40Belegen(Vorgang* z1, Vorgang* z2) {
    z1 = new Vorgang(); z1->dauer=30;
    z2 = new Vorgang(); z2->dauer=40;
}
```

Die Änderung von z1 und z2
kommt in main gar nicht an.
z1 und z2 sind keine Ausgabeparameter.

```
int main() {
    Vorgang* vz1 = nullptr;  Vorgang* vz2 = nullptr;
    mit30_40Belegen(vz1, vz2);
    cout << "vz1: " << vz1->dauer << ", vz2: " << vz2->dauer << "\n";
    delete vz1; vz1 = nullptr; delete vz2; vz2 = nullptr;
}
```

Falsche Lösung

```
void mit30_40Belegen(Vorgang* z1, Vorgang* z2) {  
    z1 = new Vorgang(); z1->dauer=30;  
    z2 = new Vorgang(); z2->dauer=40; }  
int main() {  
    Vorgang* vz1 = nullptr;  Vorgang* vz2 = nullptr;  
    mit30_40Belegen(vz1, vz2); .... }
```



Die Verbindung zwischen vz1 und z1 bzw. vz2 und z2 fehlt.



Übung 4.3 -- Lösung

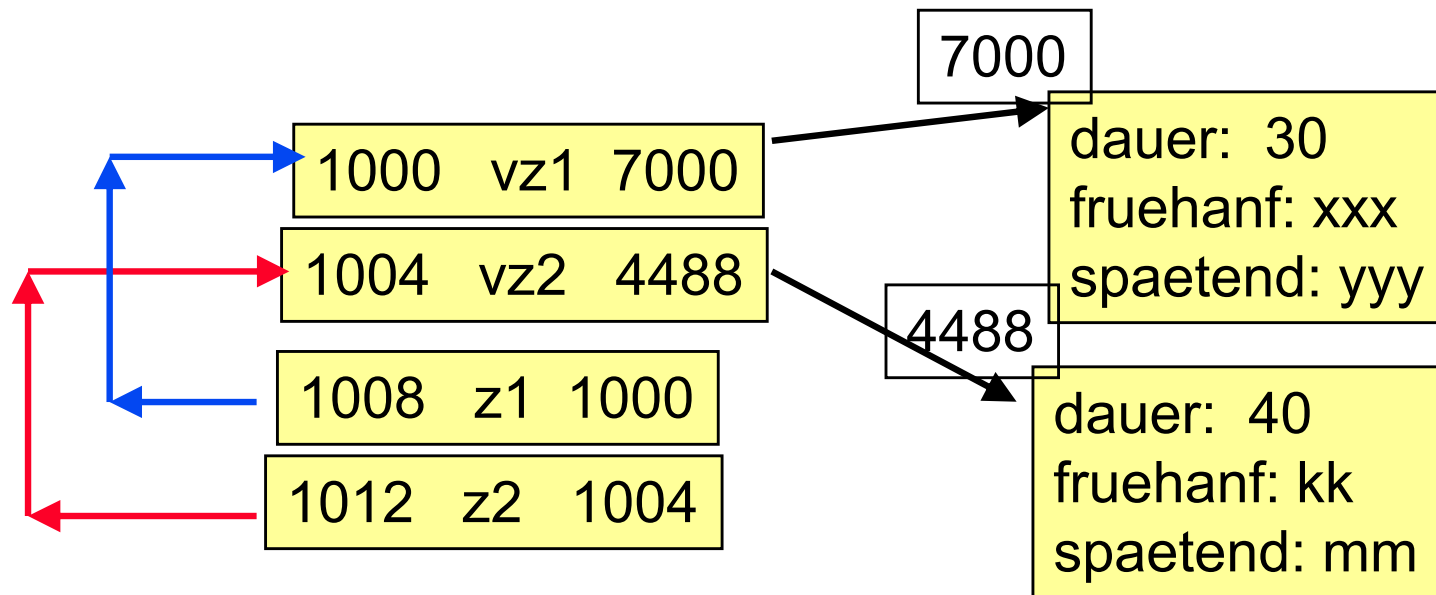
```
/** Die Funktion fordert Speicher auf dem Heap für z1 und z2 an und belegt  
    die Vorgangsdauern dann mit 30 bzw. 40. */
```

```
void mit30_40Belegen(Vorgang*& z1, Vorgang*& z2) {  
    z1 = new Vorgang; z1->dauer=30;  
    z2 = new Vorgang; z2->dauer=40;  
}
```

```
int main() {  
    Vorgang* vz1 = nullptr;  Vorgang* vz2 = nullptr;  
    mit30_40Belegen(vz1, vz2);  
    cout << "vz1: " << vz1->dauer << ", vz2: " << vz2->dauer << "\n";  
    delete vz1; vz1 = nullptr; delete vz2; vz2 = nullptr;  
}
```

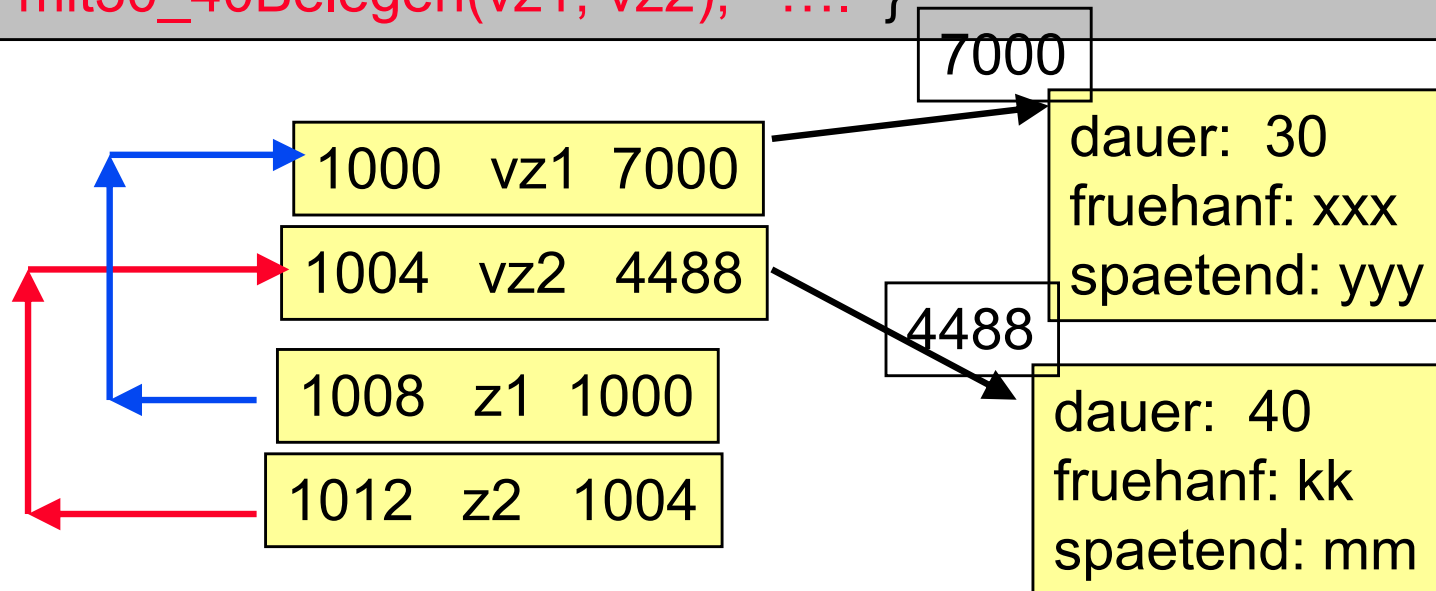
Richtige Lösung

```
void mit30_40Belegen(Vorgang*& z1, Vorgang*& z2) {  
    z1 = new Vorgang(); z1->dauer=30;  
    z2 = new Vorgang(); z2->dauer=40; }  
int main() {  
    Vorgang* vz1 = nullptr;  Vorgang* vz2 = nullptr;  
    mit30_40Belegen(vz1, vz2); .... }
```



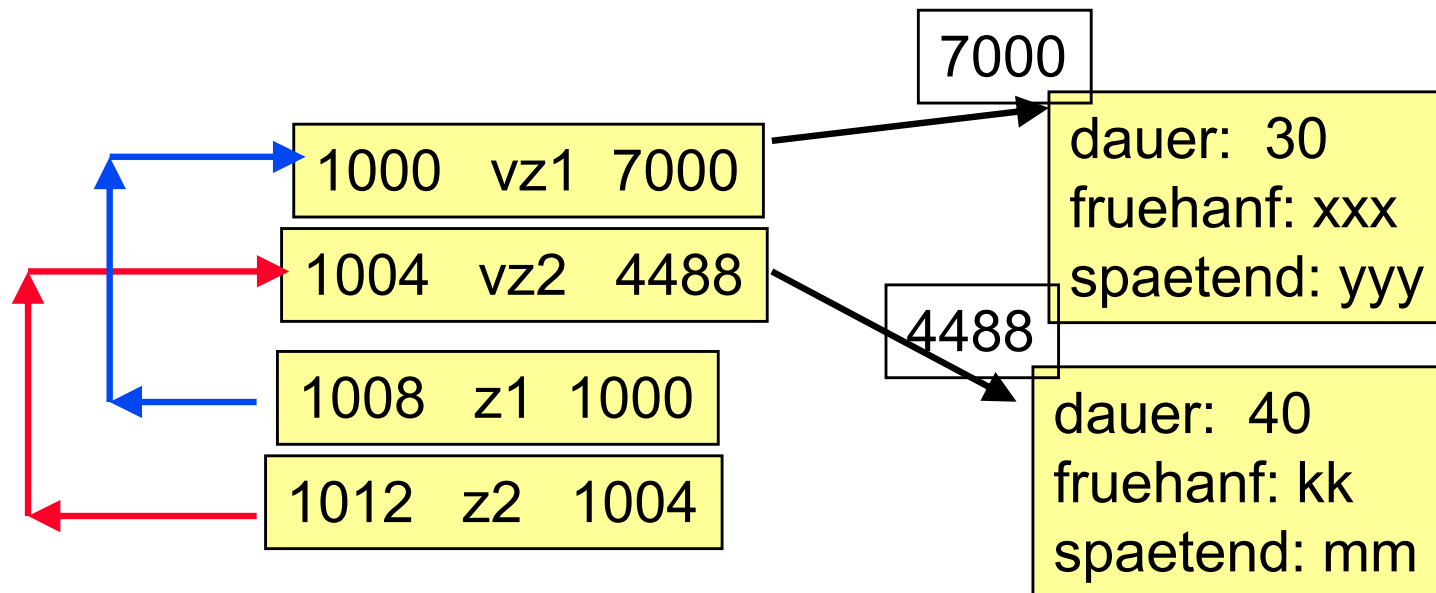
Lösung unter Verwendung von typedef

```
typedef Vorgang *MeinZeiger;  
void mit30_40Belegen(MeinZeiger& z1, MeinZeiger& z2) {  
    z1 = new Vorgang(); z1->dauer=30;  
    z2 = new Vorgang(); z2->dauer=40; }  
int main() {  
    MeinZeiger vz1 = nullptr; MeinZeiger vz2 = nullptr;  
    mit30_40Belegen(vz1, vz2); .... }
```



Lösung mit Zeiger auf Zeiger

```
void mit30_40Belegen(Vorgang** z1, Vorgang** z2) {  
    *z1 = new Vorgang(); (*z1)->dauer=30;  
    *z2 = new Vorgang(); (*z2)->dauer=40; }  
int main() {  
    Vorgang* vz1 = nullptr;  Vorgang* vz2 = nullptr;  
    mit30_40Belegen(&vz1, &vz2);  .... }  
}
```



1. Habe was anderes gemacht
2. Habe viel versucht, aber irgendwie geht es nicht
3. Hatte das Problem, das ich nur Zeiger übergeben habe und konnte es **nicht** lösen
4. Hatte erst das Problem, das ich nur Zeiger übergeben habe und konnte es dann lösen
5. Habe das Problem gleich gelöst.

Noch eine Anwendung von Zeiger auf Zeiger



Sinnvolle Anwendung von Zeiger auf Zeiger (char**) :

```
char* pch = 0;
char name[100];
cin >> name;
LosGibSpeicherHer( &pch,  strlen(name)+1  );
strcpy(pch, name);
```

```
/* Implementierung (Definition): */
void LosGibSpeicherHer(char** ppch, int nBytes)
{
    *ppch = new char [nBytes];           // C++
    /*ppch = (char*) malloc(nBytes);    // C
}
```



Sinnvolle Anwendung von Zeiger auf Zeiger (char**) :

```
char* pch = 0;  
char name[100];  
cin >> name;  
LosGibSpeicherHer( pch,  strlen(name)+1  );  
strcpy(pch, name);
```

```
/* Implementierung (Definition): */  
void LosGibSpeicherHer(char*& ppch, int nBytes)  
{  
    ppch = new char [nBytes];           // C++  
}
```