

Programmiertechnik (C++)

Prof. Dr. Stefan Enderle

NTA Isny

2. Dynamische Speicherverwaltung

Zeiger

- Zeiger (Pointer) zeigen auf eine Speicherstelle.
- Zeiger haben trotzdem einen Typ!
- Der Typ gibt an, wie die entsprechende Speicherzelle interpretiert wird.
- Ein Zeiger, der auf Nichts zeigt besitzt den Wert `NULL`.

Zeiger - Deklaration

- Deklaration eines Zeigers:

```
int* i;  
char* c;  
double* d;  
string* s;
```

- „Traditionelle Schreibweise“:

```
int *i;
```

Zeiger - Vorsicht

- Bei „traditioneller Schreibweise“ möglich:

```
int *i, *j, *k;
```

- Achtung:

```
int* i, j, k;
```

Hier sind `j` und `k` keine Pointer sondern `int`!!

- Sauberste Lösung:

```
int* i;  
int* j;  
int* k;
```

new

- Die Deklaration eines Zeigers erzeugt noch keinen Speicherplatz!
- Der Operator `new` reserviert Speicherplatz und liefert die Adresse davon zurück.
- Aufruf: `new <Typ>`
- Die Größe des benötigten Speicherplatzes entnimmt `new` dem übergebenen Typ.
- Dies funktioniert sowohl für elementare Datentypen als auch für Klassen.

new - Beispiele

- Anlegen von Speicherplatz für einen `int`:

```
int* i;           // Zeiger deklarieren  
i = new int;     // Speicher anlegen
```

- Anlegen eines Objektes einer Klasse:

```
string* s;  
s = new string;
```

- Beim Anlegen eines Objektes wird automatisch der Default-Konstruktor der Klasse aufgerufen, damit das Objekt ordnungsgemäß initialisiert ist.

new - Initialisierung

- Elementare Datentypen sind nach Aufruf von new nicht initialisiert.
- Zur Initialisierung kann ein passender Anfangswert in Klammern angegeben werden:

```
i = new int(0);  
d = new double(17.5);
```
- Zur Initialisierung von Objekten einer Klasse kann der gewünschte Konstruktor aufgerufen werden:

```
s = new string(„Hallo“);
```

new []

- Mit dem Operator `new []` wird Speicherplatz für ein Array von Objekten reserviert.
- Anlegen von Speicherplatz für 100 `ints`:

```
int *i;           // Zeiger deklar.  
i = new int[100]; // Speicher anlegen
```
- Anlegen von 100 Strings:

```
string* s;  
s = new string[100];
```
- Beim dynamischen Anlegen von Arrays mit `new []` wird stets der Default-Konstruktor aufgerufen. Initialisierungswerte sind nicht möglich.

Zugriff auf dynamischen Speicher

- Auf Speicherplatz, der mit `new []` reserviert wurde, wird über den entsprechenden Zeiger zugegriffen.
- Beispiel für elementaren Typ:

```
int *i;           // Zeiger deklar.  
i = new int;     // Speicher anlegen  
*i = 15;         // Wert zuweisen  
cout << *i << endl; // Wert ausgeben  
cout << i << endl;  // Adresse ausgeben!
```

Zugriff auf dynamische Objekte

- Beispiel für Objekt einer Klasse:

```
string *s;           // Zeiger deklar.  
s = new string;     // Speicher anlegen  
*s = „Hallo“;       // Wert zuweisen  
s->append(„ Welt“); // Methode aufrufen  
cout << *s << endl; // String ausgeben  
cout << s << endl;  // Adresse ausgeben!
```

- Bemerkung: `s->` entspricht `(*s)`.

Zugriff auf dynamische Arrays

- Beispiel für Array eines elementaren Typ:

```
int *i;           // Zeiger deklar.  
i = new int[100]; // Speicher anlegen  
i[3] = 15;       // Wert zuweisen  
cout << i[3] << endl; // Wert ausgeb.  
cout << i << endl; // Adresse ausgeben!
```

- **Bemerkung:** `s[n]` entspricht `*(s+n)`
wobei für `n` die korrekte Größe des Objektes eingesetzt wird.

Zugriff auf dynamische Arrays

- Beispiel für Array von Objekten einer Klasse:

```
string *s;           // Zeiger deklar.  
s = new string[100]; // Speicher anlegen  
s[3] = „Hallo“;     // Wert zuweisen  
s[3].append(„ Welt“); // Methode auf.  
cout << s[3] << endl; // String ausg.  
cout << s << endl;  // Adresse ausgeben!
```

delete

- Nach Benutzen des dynamischen Speichers muss dieser wieder freigegeben werden.
- Nicht freigegebener Speicher bleibt zum Ende des Programms reserviert und wird dann durch das BS wieder freigegeben.
- Der Operator `delete` gibt reservierten Speicherplatz wieder frei.
- Aufruf: `delete <Zeiger>`

delete - Beispiele

- Anlegen und Freigeben eines `double`:

```
double* d;        // Zeiger deklarieren
d = new double;  // Speicher anlegen
delete d;        // Speicher freigeben
```

- Anlegen und Freigeben eines Objektes einer Klasse:

```
string* s;
s = new string;
delete s;
```

- Beim Freigeben eines Objektes wird automatisch der Destruktor der Klasse aufgerufen, damit das Objekt ordnungsgemäß beendet wird.

delete []

- Mit dem Operator `delete[]` wird Speicherplatz für ein Array von Objekten wieder freigegeben.
- Anlegen und Freigeben von 100 `ints`:

```
int *i;           // Zeiger deklar.  
i = new int[100]; // Speicher anlegen  
delete[] i;      // Speicher freig.
```
- Anlegen und Freigeben von 100 Strings:

```
string* s;  
s = new string[100];  
delete[] s;
```
- Beim Freigeben von Objekt-Arrays wird für jedes Objekt der Destruktor aufgerufen.