Der Zeigertyp (Pointer)

Ein **Zeiger** (Pointer) ist eine Variable, deren Wert eine Maschinenadresse ist. An dieser Adresse kann dann der Wert der zugehörigen **dynamischen Variablen** abgelegt werden. Die Deklaration einer Zeigervariablen erfolgt durch einen vorangestellten Stern *. Zu jedem Typ kann eine Zeigervariable definiert werden.

Beispiel:
```
int * z1, * z2;        //Deklaration von Zeigern
char * cp;
int i;                 //Deklaration einer einfachen Variablen
```

Zeiger werden u. a. zur **dynamischen Speicherverwaltung** oder zur **Definition von strukturierten Datentypen** (Listen, Warteschlangen, Bäumen) genutzt.

Aufgabe 1:
Vergleichen Sie die Eigenschaften (Typ, Name, Vereinbarung, Adresse usw.) von statischen und dynamischen Variablen.

Zum Arbeiten mit Zeigern gibt es den **Adressoperator &**, den **Inhaltsoperator *** und die **Operatoren *new*** und ***delete***. Wird einem Zeiger die Konstante ***NULL*** (= 0) zugewiesen, so zeigt er auf keine Variable.

Beispiel:
```
z1 = NULL;    //Zeiger z1 bekommt den NULL-Zeiger zugewiesen
z1 = &i;      //Zeiger z1 erhält als Wert die Adresse der Variablen i
*cp = 'a';    //die char-Variable *cp, auf die der Zeiger cp ‚zeigt', erhält den Wert 'a'
z2 = new int; //Erzeugung  einer dynam. int-Variablen, z2 enthält die Adr. dieser Variabl.
delete z2;    //Freigabe des durch z2 bereitgestellten Speicherplatzes
              //nur auf mit new erzeugte Variablen anwenden!
```

Aufgabe 2:
Interpretieren und testen Sie folgendes Programm. Was wird ausgegeben?

```
#include <iostream.h>
int main()
{       int * z1, * z2;
        z1 = new int;   z2 = new int;
        *z1 = 10;       *z2 = 20;
        cout << z1 << endl << z2 << endl << endl;
        cout << *z1 << endl << *z2 << endl << endl;
        *z1 = *z2;
        cout << z1 << endl << z2 << endl << endl;
        cout << *z1 << endl << *z2 << endl << endl;
        *z1 = 30;
        cout << *z1 << endl << *z2 << endl << endl;
        z2 = z1;
        cout << z1 << endl << z2 << endl << endl;
        cout << *z1 << endl << *z2 << endl << endl;
        *z1 = 50;
        cout << *z1 << endl << *z2 << endl << endl;
        delete z1, z2;
        return 0;
}
```

Zeiger auf Funktionen und Felder

Zeiger auf Funktionen ermöglichen es, erst zur Laufzeit des Programms zu bestimmen, welche Funktion ausgeführt werden soll (**dynamic** oder **late binding**).

Aufgabe 3:
Gegeben sei eine beliebige Funktion *f* auf einem abgeschlossenen Intervall [*a*, *b*]. Gesucht ist der Flächeninhalt des Trapezes, das begrenzt ist durch die *x*-Achse, die Geraden $x = a$ und $x = b$ und die Sekante durch die Punkte (a, $f(a)$) und (b, $f(b)$).
Folgendes Programm liefert eine Lösung:

```
//ZeigerFkt.cpp
#include <iostream.h>
#include <math.h>
const double PI = 3.141592;
double trapez(double a, double b, double (*f)(double))
{       return ((b - a)/2) * (f(a) + f(b));       }
double meine(double x)
{       return x+1;    }
int main()
{       cout << trapez(0, PI/2, sin)  << "\t";
        cout << trapez(1, 3, log)     << "\t";
        cout << trapez(0, 2, meine)   << "\t";
        cout << endl;
        return 0;
}
```

Aufgabe 4:
a) Versehen Sie obiges Programm mit Kommentaren (insbesondere zu den fett hervorgehobenen Zeilen).
b) Wandeln Sie das Programm so ab, dass beliebige Intervallgrenzen über Tastatur eingegeben werden können.
c) Definieren Sie die Funktion $g(x) = 17\,x^3 - 12\,x + 4$ und rufen Sie diese im Programm auf.
d) Definieren Sie weitere Funktionen Ihrer Wahl und rufen Sie diese auf.

Zeiger auf Felder ermöglichen es (siehe folgendes Programm), die Feldgröße erst zur Laufzeit des Programms (dynamisch) festzulegen.

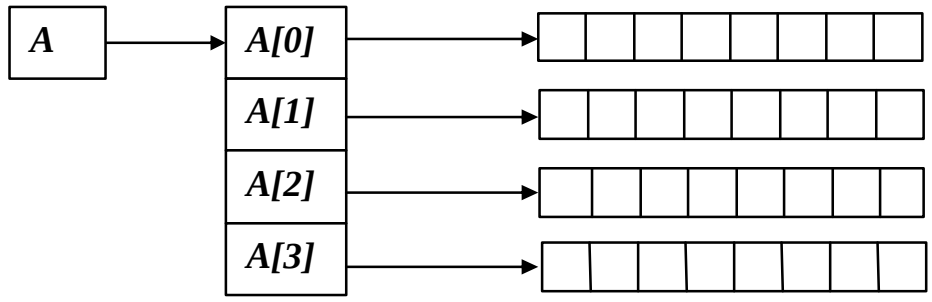
```
#include <iostream.h>
int main()
{
        int i, n;
        cout << "Dynamische Speicherzuordnung fuer einen Vektor\n";
        cout << "Laenge des Vektors eingeben: ";
        cin >> n;
        int * a = new int[n];
        for (i = 0; i < n; i++)
        {       a[i] = i;
                cout << a[i] * a[i] << ' ';
        }
        return 0;
}
```

Um ein **zweidimensionales Feld (**Matrix *A*) **mit dynamischer Feldgröße** (Typ (*m*, *n*)) zu erzeugen geht man wie folgt in zwei Schritten vor:

1. Feld von Zeigern auf eindimensionale Felder (*m* Zeilen) definieren:
   int ** A = new int*[m];

2. Jedem dieser *m* Zeiger eine Zeile (eindimensionales Feld mit *n* Komponenten) zuordnen:
   for (int i = 0; i < m; i++)
   A[i] = new int[n];

Grafische Veranschaulichung:
(für $m = 4$)



Aufgabe 5:
Schreiben Sie ein Programm, das eine Matrix *A* vom Typ (*m*, *n*) dynamisch erzeugt, die erste Zeile mit lauter Einsen, die zweite mit Zweien usw. belegt und sowohl *A* als auch αA (α soll über Tastatur eingegeben werden) auf dem Bildschirm ausgibt.