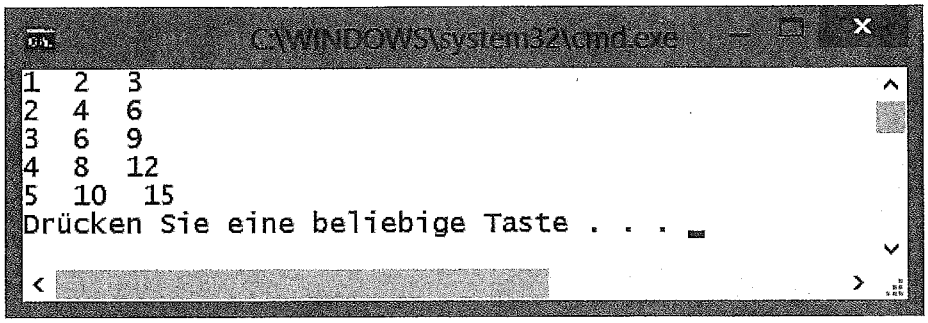
Die Bildschirmausgabe sieht dann so aus:



ACHTUNG:

Das obige Programm füllt mithilfe einer Funktion das Array mit Werten. Im Hauptprogramm ist das Array damit verändert. Dabei fällt auf, dass es sich bei der Übergabe des Arrays nicht um einen call by value handeln kann.

In der Tat ist die Übergabe eines Arrays ein call by reference. Das bedeutet, dass die übergebene Variable eine nachhaltige Änderung erfahren kann und nicht nur ihren Wert gibt.

Das Prinzip des call by reference wird im Kapitel Zeiger genauer betrachtet. Dann wird auch vollständig klar sein, warum eine Array-Übergabe keine Wertübergabe ist.

7.2 Zeichenketten in C++

Eine Zeichenkette ist nichts anderes als eine endliche Folge von Zeichen. In vielen Beispielprogrammen wurden Zeichenketten auch schon benutzt – zur Ausgabe von Text auf dem Bildschirm.

Beispiel:

```
cout << "Das ist ein Test.";
```

Das Objekt cout gibt die Zeichenkette „Das ist ein Test." auf dem Bildschirm aus.

Zeichenketten können ganz verschieden aussehen. Folgende Beispiele sind ebenfalls Zeichenketten.

- `"das32jd8kjcncioeo2  dew90wek238)(jioj"`
- `"171287321712821389723782378239"`
- `",:;:_'*`?=)(/&%$§!"`

Viele Programmiersprachen bieten für die Speicherung von Zeichenketten einen eingebauten Datentyp an. Beispielsweise gibt es in PASCAL und BASIC den Datentyp String.

In der Sprache C++ gibt es keinen elementaren Datentyp für Zeichenketten. Sie werden deshalb durch eindimensionale Arrays vom Typ char konstruiert.

7.2.1 Arrays vom Typ char

Das Speichern einer Zeichenkette in C++ erfolgt über ein Array vom Typ char. Für jedes Zeichen wird dann ein Element des Arrays reserviert.

Allerdings gibt es in C++ eine Besonderheit: Jede Zeichenkette muss durch eine Art Enderkennung für beendet erklärt werden. Diese Enderkennung nennt man **Nullterminierung**, das Zeichen dafür ist `'\0'`.

Beispiel:

```
char Kette [ 5 + 1 ] = "HALLO";
```

Es wird ein Array mit 6 (= 5 + 1) Elementen vereinbart, obwohl nur fünf Zeichen nötig sind. Die Reservierung eines weiteren Zeichens ist aber wichtig, da der Compiler automatisch die Nullerkennung an die Zeichenkette anhängt, und dafür ist ein Element nötig.

Deshalb ist die Schreibweise 5 + 1 eine Art Gedächtnisstütze dafür, dass Platz für die Nullerkennung reserviert werden muss.

Im Speicher sieht das Array dann wie folgt aus:

Index i	0	1	2	3	4	5
Inhalt: Kette[i]	H	A	L	L	O	\0

Hinweis:

Das Zeichen '\0' besteht zwar auf den ersten Blick aus zwei Zeichen, allerdings ist der Backslash \ nur die Einleitung für ein Sonderzeichen und deshalb ist '\0' auch nur ein einzelnes Zeichen.

Besonderheiten bei Arrays vom Typ char:

Besonderheit 1:

Die Zuweisung einer Zeichenkette an ein char-Array kann nur bei der Definition geschehen. Eine Zuweisung im Programm ist nicht möglich.

```
char Kette [ 5 + 1 ] = "HALLO";
```

Hier kann die Zeichenkette dem Array zugewiesen werden. Der Compiler sorgt für die Nullerkennung.

```
Kette = "OLLAH";
```

Eine solche Zuweisung im Programm ist **nicht** möglich. Hier müssten Funktionen benutzt werden.

Alternativ könnten die Elemente einzeln zugewiesen werden:

```
Kette [0] = 'H';
Kette [1] = 'A';
Kette [2] = 'L';
Kette [3] = 'L';
Kette [4] = 'O';
Kette [5] = '\0';
```

Besonderheit 2:

Die automatische Zuweisung der Nullerkennung durch den Compiler erfolgt nur bei der Definition eines char-Arrays, wie in Besonderheit 1 dargestellt.

An allen anderen Stellen ist der Programmierer oder eine entsprechende Funktion dafür verantwortlich, die Nullerkennung korrekt zu setzen.

Wird die Nullerkennung bewusst überschrieben, so kann das nicht vorhersehbare Folgen haben, wie im Folgenden dargestellt wird.

```
char Kette [ 5 + 1 ] = "HALLO";
cout << Kette;
Kette [ 5 ] = 'X';
cout << Kette;
```

Das Objekt cout gibt alle Zeichen des Arrays aus – bis zur Nullerkennung.

Die Nullerkennung wird bewusst überschrieben.

Das Objekt cout gibt nun so lange Zeichen aus, bis es zufällig auf eine Nullerkennung im Speicher stößt. Das kann einen fatalen Fehler verursachen.

7.2.2 Funktionen zur Zeichenkettenbearbeitung

Die Behandlung von Zeichenketten in C++ ist mühsam. Deshalb ist es sinnvoll, Funktionen zu schreiben, die die Arbeit erleichtern.

Ganz besonders wichtig ist, dass die Funktionen auf die korrekte Verwendung der Nullterminierung achten.

Ein elementares Beispiel für eine Zeichenkettenfunktion ist die Funktion `Laenge`, die die Länge einer Zeichenkette ermittelt und zurückgibt:

```
int Laenge( char K [ ] )
{
    int i;
    for (i=0; K[i]!='\0'; i++);
    return i;
}
int main()
{
    char Kette[5+1] = "Hallo";
    cout << Laenge(Kette); // gibt 5 auf dem Bildschirm aus
    return 0;
}
```

Solange die Nullerkennung nicht erreicht ist, wird die Schleifenvariable i inkrementiert.

Ein weiteres Beispiel ist die Funktion `Kopie`, die eine Zeichenkette auf eine andere kopiert. Dadurch sind auch Zuweisungen von Zeichenketten im Programm möglich.

```
int Kopie( char Ziel [ ] , char Quelle [ ] )
{
    int i;
    for (i=0; Quelle[i]!='\0'; i++) Ziel[i] = Quelle[i];
    Ziel[i]='\0';
}

int main()
{
    char Quelle[5+1] = "Hallo";
    char Ziel[5+1];
    Kopie (Ziel,Quelle);
    cout << Ziel ; // gibt "Hallo" auf dem Bildschirm aus
    Kopie (Ziel, "ollaH");
    cout << Ziel ; // gibt "ollaH" auf dem Bildschirm aus
    return 0;
}
```

WICHTIG: Nullerkennung nicht vergessen!

Hinweis:

Die Standardbibliothek bietet selbstverständlich eine Vielfalt von Funktionen zur Zeichenkettenbearbeitung. Die Benutzung der Funktionen setzt die Kenntnisse der Zeiger aus dem nächsten Kapitel voraus. Einen weiteren Überblick wird das Kapitel „Die C++-Standardbibliothek" verschaffen.

7.3 Sortieren von Arrays

In der Praxis werden Arrays oft zur Speicherung von Messwerten benutzt. Bei der Auswertung der Messwerte ist es oftmals sinnvoll, die Messwerte in sortierter Reihenfolge zu haben. Die Ermittlung des Medians (eines speziellen Mittelwertes) kann beispielsweise nur in einem sortierten Array durchgeführt werden.

Es gibt sehr viele Sortieralgorithmen, die unterschiedlich arbeiten und je nach Voraussetzung langsamer oder schneller sind. Beispielsweise sind die Art und die Vorsortierung der Messwerte wichtig, um den besten Algorithmus aussuchen zu können.

Solange es nicht auf Geschwindigkeit ankommt (was eher selten ist), könnte man mit einem Algorithmus auskommen.

In den folgenden Abschnitten werden zwei elementare Algorithmen vorgestellt, um eine Vorstellung von den Sortieralgorithmen zu erhalten.

Aufgabe 7.4

In der Mathematik werden zwei Matrizen nach folgender Vorschrift multipliziert:

Die erste Zeile der ersten Matrix wird mit der ersten Spalte der zweiten Matrix elementeweise multipliziert und anschließend werden die Produkte addiert.

Das Endergebnis ist das erste Element der Multiplikationsmatrix.

Danach wird die erste Zeile der ersten Matrix mit der zweiten Spalte der zweiten Matrix elementeweise multipliziert und anschließend werden die Produkte addiert.

Das Endergebnis ist das zweite Element der Multiplikationsmatrix usw.

Am anschaulichsten ist es mit einem einfachen Beispiel:

$$\begin{pmatrix} 1 & 3 \\ 2 & 6 \end{pmatrix} \bullet \begin{pmatrix} 4 & 2 \\ 3 & 5 \end{pmatrix} = \begin{pmatrix} 1 \cdot 4 + 3 \cdot 3 & 1 \cdot 2 + 3 \cdot 5 \\ 2 \cdot 4 + 6 \cdot 3 & 2 \cdot 2 + 6 \cdot 5 \end{pmatrix} = \begin{pmatrix} 13 & 17 \\ 26 & 34 \end{pmatrix}$$

Schreiben Sie ein C++-Programm, das die Multiplikation von beliebigen 2 × 2-Matrizen erlaubt.

Zusatz: Erweitern Sie das Programm auf die Multiplikation von 3 × 3-Matrizen.

Aufgabe 7.5

Schreiben Sie die folgenden Funktionen zur Zeichenkettenbearbeitung.

- Schreiben Sie eine Funktion `int` **`AnzahlZeichen`** `( char K[] , char c )`, die überprüft, wie oft ein Zeichen c in einer übergebenen Zeichenkette enthalten ist, und die diese Anzahl zurückgibt.

Beispiel:

```
char Kette[20 + 1] = "Uebungen sind schoen";
cout << AnzahlZeichen ( Kette , 'e' );    // gibt 3 aus.
```

- Schreiben Sie eine Funktion `bool` **`Palindrom`** `( char K [ ] )`, die überprüft, ob eine übergebene Zeichenkette ein Palindrom ist. Der Rückgabewert ist entweder `true` oder `false`.

  Ein Wort heißt Palindrom, wenn es rückwärts gelesen immer noch dasselbe Wort ist – beispielsweise das Wort „otto".

Beispiel:

```
char Kette[4 + 1] = "otto";
if  ( Palindrom ( Kette ) == true )
cout << Kette << " ist ein Palindrom";
```

- Schreiben Sie eine Funktion `void` **`Umdrehen`** `(char K [ ] )`, die eine übergebene Zeichenkette umdreht.

Beispiel:

```
char  Kette[5 + 1] = "Hallo";
Umdrehen ( Kette );
cout << Kette; // gibt "ollaH" auf dem Bildschirm aus
```

- Schreiben Sie eine Funktion `long` **`Char2Int`** `(char K [ ] )`, die eine übergebene Zeichenkette in einen Long-Integer-Wert umwandelt und diesen zurückgibt. Die Funktion soll Nichtziffern ignorieren.

Beispiele:

```
long  x;
x  =  Char2Int ( "1234" );          // x hat den Wert 1234
x  =  Char2Int ( "01234" );         // x hat den Wert 1234
x  =  Char2Int ( "xyz123" );        // x hat den Wert 123
x  =  Char2Int ( "123xyz" );        // x hat den Wert 123
x  =  Char2Int ( "1:v;2##3+4" );    // x hat den Wert 1234
```