- **Schreiben Sie ein Programm, welches folgende Bestandteile besitzt**
  - **Anlegen eines Feldes mit z.B. 10 ganzzahligen Elementen und Abfrage dieser Werte von der Tastatur**
  - **Funktion void print(int Feld[ ])**
    die alle Feldelemente z.B. in einer Zeile ausgibt
  - **Funktion int minIndex(int Feld[ ], int Ind1, int Ind2),**
    welche den Index (!) des Feldelements mit dem kleinsten Wert im Bereich der Indices Ind1… Ind2 liefert
  - **Funktion void swap(int Feld[ ], int Ind1, int Ind2),**
    welche zwei durch Indices angegebene Feldelemente miteinander vertauscht

- **Verwenden Sie diese Elemente geeignet, um eine Sortierung des Feldes zu erzielen!**

- **Tip: Eine Skizze dieses Sortierverfahrens ist sicher hilfreich**

Sortieren

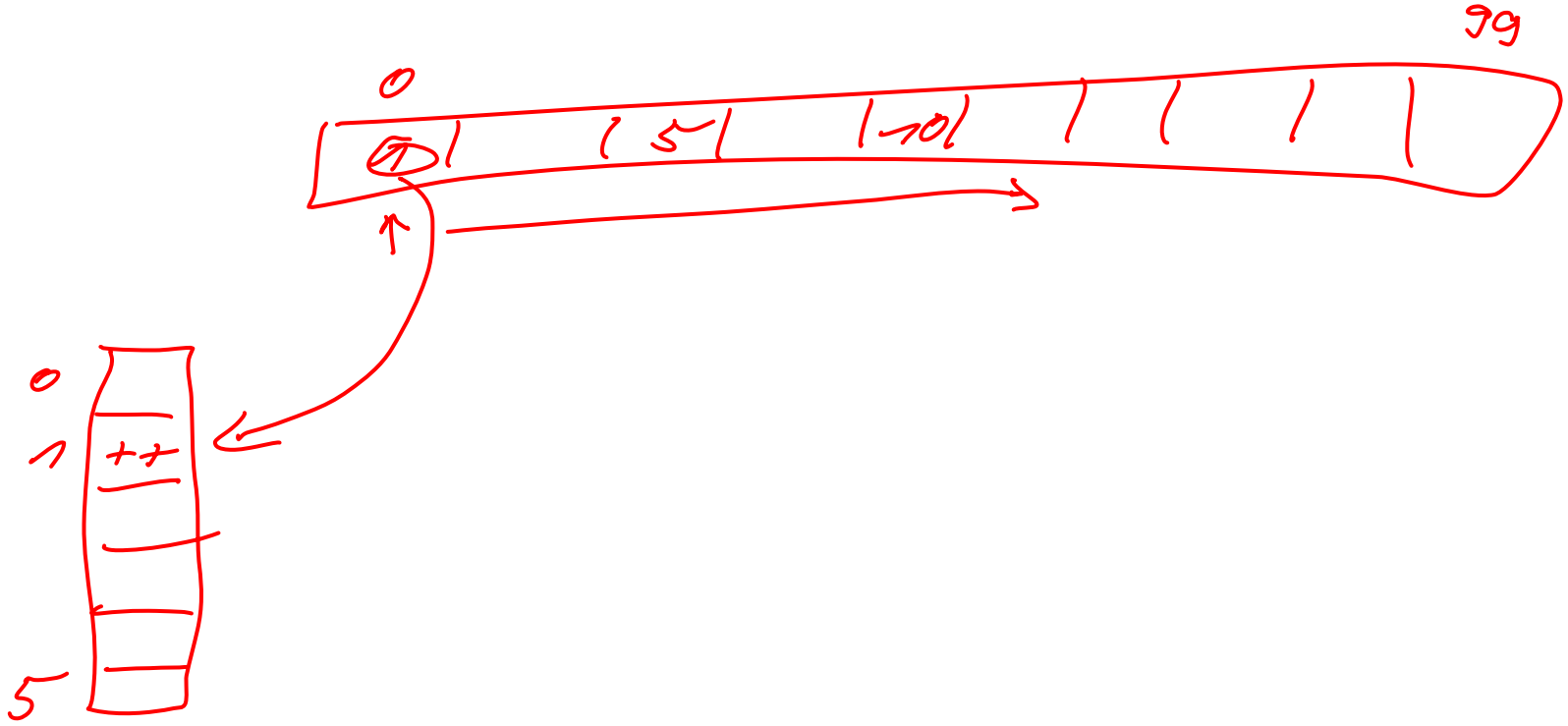
```
#include "stdafx.h"
#include <stdlib.h>
#include <iostream>
using namespace std;

const int FeldGr = 100000;


void print(int a[])
{
     for (int i=0; i<FeldGr; i++)
         cout << a[i] << " ";

     cout << endl;
}


int minIndex(int a[], int Ind1, int Ind2)
{
     int mI=Ind1;

     for (int i=Ind1+1; i<Ind2; i++)
     {
         if (a[i] < a[mI])
                  mI = i;
     }
     return mI;
}


void swap(int a[], int Ind1, int Ind2)
{
     int t = a[Ind1];
     a[Ind1] = a[Ind2];
     a[Ind2] = t;
}
```

```
int _tmain(int argc, _TCHAR* argv[])
{
     int meinFeld[FeldGr]; // = { 9, 4, 6, 1,
     3, 2, 8, 7, 5, 0 };
     int mI;

     srand(4711);

     for (int i=0; i<FeldGr; i++)
         meinFeld[i] = rand() % 1000;

     //print(meinFeld);

     for (int i=0; i<FeldGr-1; i++)
     {
         mI = minIndex(meinFeld, i, FeldGr);
         swap(meinFeld, i, mI);
         //print(meinFeld);
     }

     return 0;
}
```

- **Schreiben Sie ein Programm, welches folgende Bestandteile besitzt**
  - **Anlegen eines Feldes mit z.B. 100 ganzzahligen Elementen und Auffüllen dieser Werte durch Zufallszahlen { srand(), rand() } im Wertebereich von 0..10**
  - **Funktion void Feld_scannen(int ZahlenFeld[], int ScanFeld[]), welche das Zahlenfeld hinsichtlich der Häufigkeit der generierten Zufallszahlen untersucht**
  - **Funktion void ScanFeldAusgeben(int ScanFeld[ ]) die alle Feldelemente, d.h. die Häufigkeiten z.B. in einer Zeile ausgibt**

- **Anregung: Sie können die Ausgabe auch als horizontale Balkendiagramme durchführen**

  0: ******
  1: ***
  2: *******


99
0
5
10
0
1
++
5

Zufallsgütetest

```
#include "stdafx.h"
#include <iostream>
#include <stdlib.h>
using namespace std;

const int zmax = 100000;
const int smax = 20;

void FeldScannen(int z[], int s[])
{
     for (int i=0; i<zmax; i++)
     {
        //switch(z[i])
        //{
        //case 0: null=null+1; break;
        //case 0: s[0] = s[0]+1;  // s[0]++;

        s[ z[i] ] ++;
     }
}


void ScanFeldAusgeben(int s[])
{
     for (int i=0; i<smax; i++)
     {
        cout << i << ": " << s[i] << endl;
     }
}
```

```
int _tmain(int argc, _TCHAR* argv[])
{
     int zahlen[zmax], scan[smax];
     int i;

     srand(4711);
     for (i=0; i<zmax; i++)
        zahlen[i] = rand() % smax;
     for (i=0; i<smax; i++)
        scan[i]=0;

     FeldScannen(zahlen, scan);
     ScanFeldAusgeben(scan);

     return 0;
}
```

- **Schreiben Sie ein Programm, welches folgende Aufgaben löst**
  - **Abfragen einer Zeile von der Tastatur { cin.getline() , siehe Hilfe }**
  - **Umkopieren der Eingabe in ein anderes Feld gleicher Länge, hierbei sollen alle Kleinbuchstaben in Großbuchstaben konvertiert werden**
    void CopyToUpper(char *ziel, char *quelle)
  - **Analyse dieser Großbuchstabenzeile wie in der vorherigen Übungsaufgabe hinsichtlich der Häufigkeit der vorkommenden Zeichen**
    void Analyse(char *string, int scan[ ])
  - **Wenn als Text "stop" eingegeben wird, soll das Sammeln der Daten aufhören und die Häufigkeiten der Zeichen ausgegeben werden**

Hinweis: Verwenden Sie beim Umkopieren und der Analyse Zeiger für den Felddurchlauf!

Strings, Zeichenzählen, Zeiger

```
#include "stdafx.h"
#include <iostream>
using namespace std;

void CopyToUpper(char *ziel, char *quelle)
{
    while (*quelle != '\0')
    {
       if (*quelle >= 'a' && *quelle<='z')
           *ziel = *quelle-'a'+'A';
       else
           *ziel = *quelle;
       ziel++;
       quelle++;
    }
    *ziel = '\0';
}

void Analyse(char *string, int scan[])
{
    while (*string != '\0')
    {
       if (*string >= 'A' && *string <= 'Z')
           scan[ *string -'A' ] ++;
       string++;
    }
}
```

```
int _tmain(int argc, _TCHAR* argv[])
{
    char kFeld[100], gFeld[100];
    int  scanFeld['Z'-'A'+1];

    for (int i=0; i<26; i++)
       scanFeld[i] = 0;

    while (1)
    {
       cin.getline(kFeld, 100);
       if (strcmp(kFeld,"stop") == 0)
           break;
       CopyToUpper(gFeld, kFeld);
       Analyse(gFeld, scanFeld);
       cout << "Zeile: " << gFeld << endl;
    }
    for (char c='A'; c<= 'Z'; c++)
       cout << c << ": " << scanFeld[c-'A'] <<
    endl;

    return 0;
}
```

Spiel "Hangman"

Realisieren Sie das bekannte Wortratespiel auf dem Computer. Zeigen Sie zunächst von einem zu ratenden Wort für die einzelnen Zeichen jeweils nur Striche an. Anschließend fragen Sie nach Buchstaben. Sollten die Buchstaben im zu ratenden Wort vorkommen, so werden die entsprechenden Positionen aufgedeckt, d.h. an dieser Stelle werden die tatsächlichen Zeichen angezeigt. Durch Eingabe eines Zeichens (z.B. '#' – welches dann in keinem Wort vorkommen darf) kann das Raten abgebrochen werden und die Eingabe des vollständigen Worts ist nun möglich. Stimmt das eingegebene Wort mit dem verdeckten Wort überein, ist das Spiel zu Ende, ansonsten wird wieder in den obigen Buchstabenratemodus zurückgekehrt.

- Beispiel: Zu ratendes Wort: "Hangman" (verdeckt)

  - - - - - - - - Eingabe (a)
  - a - - - a - Eingabe (g)
  - a - g - a - Eingabe (e)
  - a - g - a - Eingabe (n)
  - a n g - a n Eingabe (#) ~ Ende Buchstabenraten

  Eingabe: Hangman → richtig geraten / bzw. Zurückkehren zum Buchstabenraten

Tipp: Legen Sie sich eine Tabelle von zu ratenden Wörtern an und wählen Sie aus dieser Tabelle zufällig einen Eintrag für das Ratewort:

```
char *ratewoerter[ ]={"Wort1", "Wort2", "Wort3", …};
```

Hinweis: Verwenden Sie für das Durchlaufen der Zeichenkette(n) wieder Zeiger, wo immer dies sinnvoll ist!

Hangman (Ausschnitt)

```
#include "stdafx.h"
#include <iostream>
#include <stdlib.h>
using namespace std;


void InitGeraten(char *gw, char *w)
{
      //int len=strlen(w);
      //for (int i=0; i<len; i++)
      //   gw[i]='-';
      while (*w != '\0')
      {
          *gw='-';
          gw++;
          w++;
      }
      *gw='\0';
}


bool ZeichenEintragen(char *gw, char *w, char ch, bool &ez)
{
      bool fertig=true;
      ez = false;
      //ch = toupper(ch);

      while (*w != '\0')
      {
          if (toupper(*w) == toupper(ch))
          {
                    *gw=ch;
                    ez = true;
          }
          else if (*gw == '-')
                    fertig = false;
          w++;
          gw++;
      }
      return fertig;
}
```

```
int _tmain(int argc, _TCHAR* argv[])
{
      char *ratewoerter[] = {"Test", "Computer", "Hangman",
      "Otto"};
      char geraten[20];
      char *wort, ch;
      bool ExZeichen, fertig;
      int fehlversuche=0;

      srand(4711);
      wort = ratewoerter[rand() % 4];

      InitGeraten(geraten, wort);

      do
      {
          cout << "Wort: " << geraten << "   Zeichen? ";
          cin >> ch;
          fertig = ZeichenEintragen(geraten, wort, ch, ExZeichen);
          if (!ExZeichen)
                    fehlversuche++;
          cout << fehlversuche << "  | ";
      } while (!fertig);

      return 0;
}
```

Übungsaufgabe Schiffe versenken

- In dieser Übungsaufgabe soll das bekannte "Schiffe versenken"-Spiel auf einem 10x10 Kästchen großen Spielfeld realisiert werden. Hierbei darf der Computer auf einem Spielfeld Schiffe platzieren (ein Schiff, welches 4 Kästchen groß ist, zwei 3er-Schiffe und drei 2er-Schiffe). Die Schiffe müssen von dem Verfahren zufällig platziert werden, jedoch natürlich so, dass sich zwei Schiffe nicht berühren, es muss also mindestens ein Kästchen zwischen zwei Schiffen frei bleiben).

- Im folgenden können Sie raten, wo sich die Schiffe befinden könnten. Hierzu können Sie Koordinaten eingeben und der Computer hat die Aufgabe, diese Position folgendermaßen zu bewerten:
  - Wasser - knapp daneben – Treffer - Treffer, versenkt
  - Neben dieser Information soll nach jedem Raten ein aktualisiertes Spielfeld angezeigt werden, in dem schon geratene Positionen, getroffene Schiffe, etc. geeignet dargestellt werden.
- Hinweise:
  - Überlegen Sie sich bitte zuerst genau (!), welche Informationen Sie für die einzelnen Stadien des Spiels (Initialisierung, Raten) benötigen und wie Sie diese am besten zur Verfügung stellen können, d.h. welche Informationen werden für ein Schiff benötigt, um alle Zustände beschreiben zu können, z.B. belegte Felder, vollständig versenkt?, ...