}U info I 22.11.17

Nachbesprechung

- Short circuiting:
  - So bitte nicht machen

```
bool GetInput(int% n, doulbe& a, doulbe& sig, doulbe& eps){
        cout << "# Enter the number of atoms n" << endl;
        cin >> n;

        cout << "# Enter the lattice parameter a" << endl;
        cin >> a;

        cout << "# Enter sigma" << endl;
        cin >> sig;

        cout << "# Enter epsilon" << endl;
        cin >> eps;

        if(n < 0) {
                cout << n << "is false" << endl;
                return 0;
        }
        if(a < 0) {
                cout << a << "is false" << endl;
                return 0;
        }
        if(sig < 0) {
                cout << sig << "is false" << endl;
                return 0;
                }
        if(eps < 0) {
                cout << eps << "is false" << endl;
                return 0
        }
        return 1;
}
```

- Wie geht es richtig? So:

```
bool GetInput(int& n, double& a, double& sig, double& eps) {

        cout << "# Enter the number of atoms n" << endl;
        cin >> n ;

        cout << "# Enter the lattice parameter a [nm]" << endl;
        cin >> a ;

        cout << "# Enter the LJ collision diameter sigma [nm]" << endl;
        cin >> sigma ;

        cout << "# Enter the LJ well-depth parameter epsilon [kJ/mol]" << endl;
        cin >> epsilon ;

        if (n <= 0 || a <= 0.0 || sigma <= 0.0 || epsilon <= 0.0) return false;
        return** **true**;
}
```

- Warum?
  - Durch die Verknüpfung der OR wird das gesamte Statement true, sobald einer der Vergleiche true ergibt. Also sobald ein Input falsch ist, wird der gesamte Input als falsch erkannt und false wird zurückgegeben.

Theorie zu ex 4.

- Arrays:

```
int num[length];
```

  - erzeugt einen array mit namen num und länge length. length muss ein int sein, zb 3. Dann wäre unser Array num 3 lang
  - um auf die Element in num zuzugreifen benutzt man [a]. Wobei a 0,1,2 sein muss. Dass erste Element ist an der 0. position.
  - wenn a grösser ist als 2 versucht ihr auf ein Element was nicht im array ist zuzugrifen, dass gibt ein segmentation fault oder out of bounds.
    - Merkt euch das für debugging!
  - Matrix
    `c++ int num[len1][len2];`
    - gibt euch eine matrix mit dimension len1 auf len2. Also höhe len1 und breite len2.
    - Zugriff mit $num[a][b]$
    - Könnt das num auch vom type double machen, len1 und len2 müssen aber ints sein
  - Gefüllte Arrays:
    `c++ int num[2][3] = {1,2,3,4,5,6};`
    - macht:
    1|2|3
    4|5|6
  - können wie variablen auch an Funktionen weiter gegeben werde. Werden immer per Referenz übergeben
- const:
  - mach die Variable oder den Array schreibgeschützt
- strings
  - sind Variablen die ein wort speichern
    `c++ string str_test = "test";`
  - können auch zu arrays werden, also mehrere Worte speichern, wenn ihr die Länge in eckigen klammern angebt wie bei arrays.
  - functions for strings:

Function Type	Functions	Comment
Capacity	**size** or **length**	return length of string
	resize	resize string
	clear	clear string
	empty	test if string is empty
Element Access	**[]** (operator) or **at**	get character of string
	back	access last character
	front	access first character
Modifiers	**=** (operator) or **assign**	assign content to string
	+= (operator) or **append**	append to string
	push_back	append character to string
	pop_back	delete last character
	insert	insert into string
	erase	erase characters from string
	replace	replace portion of string
	swap	swap string values
String operations	**+** (operator)	concatenate two strings into a new one
	== (operator)	compare strings
	find	find content in string
	substr	generate substring

- ifstream & -ofstream
  - skipped Tabs, Leerzeichen und Umbrüche
  - liest wort für wort aus einem Input .txt file
  - muss geöffnet werden und importiert werden:

```
#include <fstream>
ifstream inp("path/inp.txt");
ofstream out("out.txt");
```

  - und geschlossen:

```
inp.close();
out.close();
```

- Funktionen:

Function Type	Functions	Comment
General	`open`	open file
	`is_open`	check if a file is open
	`close`	close file
	`eof`	check whether eofbit is set
For Input Streams	`>>` (operator)	extract formatted input
	`gcount`	get character count
	`get`	get character
	`getline`	get line
	`ignore`	extract and discard characters
	`peek`	peek next character
	`putback`	put character back
	`unget`	unget character
	`tellg`	get position in input sequence
	`seekg`	set position in input sequence
For Output Streams	`<<` (operator)	insert formatted output
	`put`	put character
	`write`	write block of data
	`tellp`	get position in output sequence
	`seekp`	set position in output sequence
	`flush`	flush output stream buffer

4 Algorithms for sorting (S2017.4)

Betrachten Sie die folgende C++ Funktion `sort`, welche benutzt wird, um einen Array `a[1..N]` des Typs `int` (mit unbenutztem `a[0]`) zu sortieren, so dass am Ausgang `a[1]` $\leq$ `a[2]` $\leq$ .. $\leq$ `a[N]`. Die Funktionsdefinition lautet

```
void sort (int a[], int N) {
    int i, j, t;
    for (i = N; i > 1; i--) {
        for (j = 1; j < i; j++) {
            if (a[j+1] <= a[j]) { // comparison
                t = a[j]; a[j] = a[j+1]; a[j+1] = t; // swap
            }
        }
    }
    return;
}
```

Jedes Mal, wenn die Statements, die mit "`comparison`" oder "`swap`" kommentiert sind, ausgeführt werden, zählt man einen Vergleich beziehungsweise einen Austausch.

a. Geben Sie die **Anzahl der Vergleiche** und die **Anzahl der Austausche** an, die ausgeführt werden, wenn `N=4` und `a[1]=4, a[2]=3, a[3]=2, a[4]=1`.

b. Geben Sie die **Anzahl der Vergleiche** und die **Anzahl der Austausche** an, die ausgeführt werden, wenn `N=4` und `a[1]=1, a[2]=2, a[3]=3, a[4]=4`.

c. Für grosse N ist die Anzahl der Vergleiche durch αN^β gegeben. Geben Sie **Werte von α und β** im schlechtesten Fall, im besten Fall und im Durchschnitt an.

d. Für grosse N ist die Anzahl der Austausche durch γN^δ gegeben. Geben Sie **Werte von γ und δ** im schlechtesten Fall, im besten Fall und im Durchschnitt an.

e. Ist der Algorithmus **stabil**? (Ihre Antwort "ja" oder "nein" muss kurz begründet werden!). Wenn Ihre Antwort "nein" ist, schlagen Sie einen Weg vor, um den Algorithmus mit einer einfachen Veränderung stabil zu machen.

4 Algorithms for sorting (S2017.4)

a. The following table lists the successive values of `i` and `j`, the content of the array `a[1..N]` at the start of the inner loop, and the requriement for a comparison or/and swap (involving the elements in boldface)

i	j	a[1]	a[2]	a[3]	a[4]	comp.	swap
4	1	**4**	**3**	2	1	×	×
	2	3	**4**	**2**	1	×	×
	3	3	2	**4**	1	×	×
3	1	**3**	**2**	1	4	×	×
	2	2	**3**	**1**	4	×	×
2	1	**2**	**1**	3	4	×	×
end		1	2	3	4		

So, we need 6 comparisons and 6 swaps.

b. The following table lists the successive values of `i` and `j`, the content of the array `a[1..N]` at the start of the inner loop, and the requriement for a comparison and/or swap (involving the elements in boldface)

i	j	a[1]	a[2]	a[3]	a[4]	comp.	swap
4	1	**1**	**2**	3	4	×	-
	2	1	**2**	**3**	4	×	-
	3	1	2	**3**	**4**	×	-
3	1	**1**	**2**	3	4	×	-
	2	1	**2**	**3**	4	×	-
2	1	**1**	**2**	3	4	×	-
end		1	2	3	4		

So, we need 6 comparisons and 0 swaps.

c. The i-loop goes backward from `N` to `2`. The j-loop goes forward from `1` to `i-1`. Each iteration of the inner loop requires one comparison, so

$$N_{\text{comp}} = \sum_{i=2}^{N} \sum_{j=1}^{i-1} 1 = \frac{N(N-1)}{2}$$

For large N, this gives $\alpha = \frac{1}{2}$ and $\beta = 2$, valid for best, worst and average cases (as it does not depend on the content of the array).

d. In the best case, the array is already sorted and

$$N_{\text{swap}} = 0.$$

In the worst case, the array is inversly sorted and

$$N_{\text{swap}} = N_{\text{comp}} = \frac{N(N-1)}{2}.$$

Thus, on average

$$N_{\text{swap}} = \frac{N(N-1)}{4}.$$

For large N, this gives $\gamma = 0$ and δ undefined (best case), $\gamma = \frac{1}{2}$ and $\delta = 2$ (worst case) and $\gamma = \frac{1}{4}$ and $\delta = 2$ (on average).

e. A sorting algorithm is stable if two elements with the same value are kept in the same order after the sorting. The algorithm is not stable. A simple counter example is `N = 2, a[1] = a[2] = 1`. At the single iteration of the inner loop (`i = 2, j = 1`), the two elements will be swapped. If we replaced the condition `a[j+1] <= a[j]` by `a[j+1] < a[j]`, it would become stable.