

Datatypes

std::string

int

float

etc.

Arrays and List

int ar[size]

int arLenght = sizeof(ar) / sizeof(int);

Class Body File Paradigm // Compare Header

```
#include "Zaehlwerk.h"
Zaehlwerk::Zaehlwerk(int ma){
    zahl = 0;
    max = ma;
}
void Zaehlwerk::Inkrementieren(){
    if (zahl < max)
        zahl++;
}
std::string Zaehlwerk::Anzeigen(){
    return "der Zaehler ist auf der " + std::to_string(zahl);
}
```

Extensions / CPP File

```
#include
"Ringwerkzaehler.h"
Ringwerkzaehler::Ringwerkzaehler()
{Zaehlwerk(){}}
```

Extensions / CPP File (cont)

```
> Ringwerkzaehler::Ringwerkzaehler(int ma) : Zaehlwerk(ma){}
void Ringwerkzaehler::Inkrementieren(){
    if (zahl < max)
        zahl++;
    else
        zahl = 0;
}
```

bubbleSort Algorithmus

```
void bubblesort(int *array, int length){
    int j;
    for (j = 0; j < length - 1; ++j) {
        if (array[j] > array[j + 1]) {
            int tmp = array[j];
            array[j] = array[j + 1];
            array[j + 1] = tmp;
        }
    }
}
```

Compare Classes

```
bool operator < (classobj) {
    return (cVar < obj.cVar); } //In Constructor
```

Class header File Paradigm

```
class Zaehlwerk
{
protected:
    int zahl;
    int max;
public:
```

Class header File Paradigm (cont)

```
> Zaehlwerk();
Zaehlwerk(int max);
~Zaehlwerk();
virtual void Inkrementieren();
virtual void Dekrementieren();
virtual std::string Anzeigen();
virtual void Zuruecksetzen();
private:
};
```

Extentions / Header File

```
class Ringwerkzaehler :
    Zaehlwerk
{
public:
    Ringwerkzaehler();
    Ringwerkzaehler(int ma);
    ~Ringwerkzaehler();
    void Inkrementieren() override;
    void Dekrementieren() override;
    std::string Anzeigen() override;
    void Zuruecksetzen() override;
};
```

Thread Operations

```
std::this_thread::sleep_for(std::chrono::seconds(1));
```

String functions

```
printf("%s\n", text.c_str());
std::string s1(" ", 20);
s1.append(s2);
int equ = s1.compare(2, 4, s2);
s1.clear();
s2.erase(1, 3);
s1.find_last_of(s2);
s.insert(2, "einString ", 0, 3);
s.length();
s.replace(2, 5, "anoString", 2, 2);
std::string s3=s1.substr(2,3);
```

Data processing / Read file / char

```
std::ifstream f;
f.open (filename,
std::ios_base::out);
int counter = 0;
char t;
while (f.get(t))
    if (t == c)
        counter++;
f.close();
```

Struct Example

```
struct kontakt{
    std::string name;
    int nummer;
    std::string toString()
    {
        std::string temp = name + " ";
        temp += std::to_string(nummer);
        return temp;
    }
};
```

Data processing / Write file

```
std::ofstream f;
f.open (filename,
std::ios_base::out);
std::string temp =
" ";
temp = "joe ";
temp += std::string(
120);
f << temp << std::endl;
f.close();
```

Debugger / cout shorter

```
//Debug Messenger
#define DEBUG 1
#if DEBUG
#define LOG(x) std::cout
<< x << std::endl
#else
#define LOG(x)
#endif
system ("PAUSE "); //
Just here to remember
```

Data processing / Read file / getline

```
std::vector<kontakt>
kontaktListe(readFile(filename));
std::ifstream f;
f.open (filename, std::ios_
base::out);
std::string t="";
int i = 0;
while (getline(f, t))
{
for each (char var in t)
{
if (var == ' ')
{
//Operation
}
else
i++;
}
i = 0;
}
f.close();
```

