Cheat Sheet C++

Preprocessor

```
#include <stdio.h>          // Insert standard header file
#include “my file.h“        // Insert file in current directory
#define add(a,b) a+b        // define a Makro
#ifndef file_name           // conditional compilation
#define file_name           //    for header files
#endif                      //
using namespace std;        // kein std:: mehr
```

Variable Types

```
bool                        // 1 Bit
char                        // 1 Byte
short                       // 2 Byte
Int                         // 4 Byte
float                       // 4 Byte float
long                        // 8 Byte int
double                      // 8 Byte float
string                      // string object
auto                        // keyword for generic use of loops
```

Qualifier

```
const type                  // read only
static type                 // function keeps values between
invocation
Inline func()               // speed optimization
```

Statements

```
if else                          //
while()                          //
for(a ; a<b ; c++)               // for loop
for(int &a : arr)                // range based for loop
switch()                         //
Try{ a; }                        // versuche statement a
catch(T t){b;}                   // catch exception
catch{c;}                        // keine exception
throw x                          // throw exception
```

Struct

```
struct name{                     // struct with member a
    type a
};
struct structName varName // initialize
varName.a                        // accessing
ptrName->a                       // accessing if pointer
```

Enum

```
enum name {                      // creating own datatype
    name,name2
};
enum name varName;               // accessing
```

Pointer

```
type *ptr                        // creating
ptr                              // memory address
*ptr                             // value
&varName                         // memory address normal variable
```

Array

```
type name = new type[size]       // set array length
name[pos]                        // accessing arr at pos
*(name+pos)                      // accessing arr at pos
sizeof(array) / sizeof(array[0]) // return size
delete [] arrName                // deallocation
```

Strings

```
c[4] = “Ash“;                    //strings are char arrays
c[4] = {`A`,`s`,`h` };           // equivalent
\0                               //Null-terminated == false
```

Functions

```
type funcName(type var){    //declaration

}
by Value                         // passing Value
by reference                     // passing memAddress
```

Main

```
Int main(int argc, char *argv[]){  //main method call
      return 0;
}
argc                             // Number of command line args
*argv[pos]                       // actual arguments
*c = *(arg+pos)                  // get input at pos
```

Classes

```
class name{
      private:                  // ac only in class
      protected:                //
      public:                   // ac to all
            name()              // constructor
            ~name()             // destructor
}
void f() const                  // no var modification
className operator+()           // adding classMembers
ClassName(const & ptr){}        // copy Constructor
friend class className          // friend class can ac private
members
class className:public className // vererbung
virtual void classname()        // late binding, soon override
template<class T> void funcName() //creating a generic type
```

Consol I/O

```
cin >> var;                     // reading input
cout << var;                    // writing input
cerr << “Text“                  // writing to error
```

Fstream

```
ifstream f(“filename“);         // open file to read
f.getline(s,n)                  // read line into s
ofstream f(“filename“);         // open file to write
f << s                          // write to file s
```

String

```
s.size();              // length
s = c_str(…)           // char to string
s.insert(pos,s)        // concatenate string s at n
s.erase(pos,n)         // erase at pos n chars
s.begin()              // iterator to first elem
s.end()                // iterator to last elem
```

Vector

```
vector<int> v(10);     // vector with 10 pos
v.size()               // return size
v.push_back(5);        // increase size by 1 and add 5
v.back()=4             // last elem = 4
v.front()              // first elem
v.at(pos)              // return param of pos
```

deque

```
deque<int> d           //queue like vector
```

map

```
map<string,int> m;     // a map with string associated int's
```