

Übersicht über die grundlegenden C++-Elemente

Daniel Stein

Software-Praktikum "Muster- und Bilderkennung" – SS 2007

**Human Language Technology and Pattern Recognition
Lehrstuhl für Informatik VI
Computer Science Department
RWTH Aachen University, Germany**

Beispiel:

```
#include <iostream>
#include <string>

// hello world
int main(int argc, char **argv) {
    std::string s("Hello World");
    std::cout << s << "\n";
    return 0;
}
```

Kompilieren: `g++ main.cc -o main`

namespaces `std::`

Bemerkungen: **output operator** `<<`

comments `//` **oder** `/* ... */`

```
short year;  
int i;  
unsigned int counter;  
size_t length;  
float distance;  
double probability;  
char letter;  
bool flag;  
  
enum Bier { Alt, Koelsch, Pils, Weizen };  
  
char vektor[100];           vektor[ 0]='a';  
                           vektor[ 99]='z';  
                           vektor[100]='b'; // ERROR!!  
  
struct verbund {  
    int member;  
    char c;  
};
```

```
if (Bedingung) {  
    ...  
}  
else {  
    ...  
}
```

```
switch (Variable) {  
    case Wert1:  
        ...  
        break;  
    case Wert2:  
        ...  
        break;  
    default:  
        ...  
        break;  
}
```

```
for(int i=1; i<=N; i+=10) {
```

```
    ...
```

```
}
```

```
while (Bedingung) {
```

```
    ...
```

```
}
```

Bedingungen:

and	& &
or	
not	!
equal	==
not equal	!=
leq	<=
geq	>=

```
void printSquares();           // Prototypen
double square(const int& i);
double square(const float& f);
double square(const double& d);
```

```
void printSquares() {
    for(int i=0; i<10; ++i) {
        printf("%8.4f\n", square(i));
    }
}
```

```
double square(const int& i) {
    return double(i*i);
}
```

```
double square(const double& d) {
    return d*d;
}
```

Zeiger und Adressen

Zugriffsoperator: * **Adressoperator: &**

```
double square(double d) { return d*d; } // call by value (1)

double square(double *d) { return *d=(*d) * (*d); } // call by reference (2)

double power(const double& d, const double& n) { // call by reference (3)
    return exp(n*log(d));
}

void addOne(double& d) { d+=1.0; } // call by reference (4)

void main() {
    double d=10.0, n=2.0;
    square(d); // (1) d= 10.0
    square(&d); // (2) d= 100.0
    power(d,n); // (3) d=10000.0
    addOne(d); // (4) d=10001.0
}
```

Zeiger und Adressen

```
int gehalt;
int *int_ptr;

struct Person {
    std::string name_;
    size_t age_;
    Person(const std::string& name, const size_t& age) :
        name_(name), age_(age) {}
}

void main() {
    int_ptr = &gehalt;
    *int_ptr = 8000;
    Person *p = new Person("Klaus Macherey", 31);

    printf("%d \n", gehalt); // 8000
    printf("%d \n", p->age_); // 31
}
```

```
class fraction {
private:
    double numerator;
    double denominator;

public:
    fraction(const double& num,                // constructor
            const double& denom) : numerator(num),
                                   denominator(denom) {}

    ~fraction() {}                            // destructor

    void reduceLCD(const double& l) {         // member (Methode)
        numerator*=l;
        denominator*=l;
    }

    void print() const {                     // member
        std::cout << numerator << "/" << denominator;
    }
};

void main() {
    fraction b(10.0, 5.0);
    b.print();                               // 10.0/5.0
    b.reduceLCD(2.0);
    b.print();                               // 20.0/10.0
}
```

```
template <class T> bool isLower(const T& t1, const T& t2) {  
    return t1<t2;  
}
```

```
void main() {  
    int i=10, j=20;  
    isLower(i, j);           // true  
  
    double d=20.0, f=30.0;  
    isLower(d, f);          // true  
}
```

Klassen-Templates

```
template <class T> class stack {
private:
    T *s;
    size_t max;    // max Anzahl von Elementen im Stack
    size_t num;    // Anzahl der Elemente im Stack

public:
    stack(size_t n) {s=new T[max=n]; num=0;}
    ~stack() {delete [] s;}
    void push(T elem) {s[num++]=elem;}
    T pop() {return s[num--];}
};

void main() {
    stack<int> s(100);    // stack mit 100 int-Elementen
    s.push(10);
    s.push(20);
    s.pop();            // liefert 20
}
```

Programme über mehrere Dateien

Hello_World.h

```
class helloWorld {
private:
;
public:
helloWorld();
~helloWorld();
void print();
};
```

Hello_World.cc

```
#include <iostream>
#include "Hello_World.h"

helloWorld::helloWorld() {}

helloWorld::~~helloWorld() {}

void helloWorld::print() {
std::cout << "hello world" << "\n"
}

}
```

Makefile

```
#!/usr/bin/gmake
CC      :=      g++
LN      :=      g++

OBJ     =      main.o helloWorld.o

hello:  $(OBJ)
        $(LN) -o helloWorld $(OBJ)

%.o:   %.cc
        $(CC) -c $(CFLAGS) $< -o $@
```

main.cc

```
#include "Hello_World.h"

void main() {
helloWorld hw;
hw.print();
}
```

```
std::vector<int> v(100, 7); // Vektor mit 100 int-Elementen
v[40]=2;

for(std::vector<int>::iterator i=v.begin(); i!=v.end(); ++i) {
    std::cout << *i << "\n";
}

for(std::vector<int>::const_iterator ci=v.begin(); ci!=v.end(); ++ci) {
    std::cout << *ci << "\n";
}

std::map<string, int> m;
m["Klaus"]=31;
m["Peter"]=33;
for(std::map<string, int>::const_iterator ci=m.begin(); ci!=m.end(); ++ci) {
    std::cout << ci->first << ":" << ci->second << "\n";
}
```

Andere: list, deque, set, multiset, multimap, ...

Bücher:

B. Stroustrup **Die C++ Programmiersprache, Addison-Wesley**

N. M. Josuttis **The C++ Standard Library, Addison-Wesley**

B. Eckel **Thinking in C++, Web-Download:**

<http://www.mindview.net/Books/TICPP/ThinkingInCPP2e.html>

Kernighan,Ritchie **Programmieren in C, Hanser Fachbuch**

Web:

http://www.sgi.com/tech/stl/table_of_contents.html