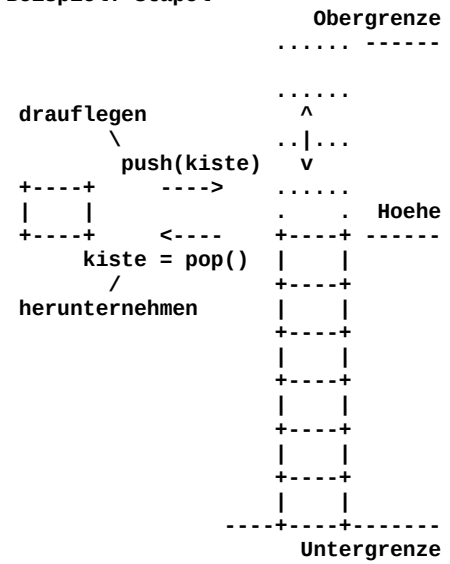
Programmierstile in C++

Beispiel: Stapel



Lit.:
B. Stroustrup: 16 Ways to Stack a Cat.
http://public.research.att.com/~bs/stack_cat.pdf

strukturiert

```
#include <iostream>            // prog.cpp

int main()
{
  const int MAX = 10;
  int hoehe = 0;
  float daten[MAX];
  float kiste;

  while (hoehe < MAX && std::cin >> kiste)
  {
    daten[hoehe++] = kiste;
  }

  while (hoehe > 0)
  {
    std::cout << daten[--hoehe] << '\n';
  }
  return 0;
}
```

---

stile.pdf (c) R.Richter 2006-03-04

prozedural

```
const int MAX = 10;            // prog.cpp
int hoehe = 0;
float daten[MAX];

void push(float kiste)
{
  daten[hoehe++] = kiste;
}

float pop()
{
  return daten[--hoehe];
}

bool leer()
{
  return hoehe == 0;
}

bool voll()
{
  return hoehe == MAX;
}

#include <iostream>

int main()
{
  float kiste;

  while (!voll() && std::cin >> kiste)
  {
    push(kiste);
  }

  while (!leer())
  {
    std::cout << pop() << '\n';
  }
  return 0;
}
```

modular, Datenbündelung

```
#ifndef STAPEL_H              // stapel.h
#define STAPEL_H

const int MAX = 10;
struct Stapel
{
  int hoehe;
  float daten[MAX];
};
void init(Stapel& s);
void push(Stapel& s, float kiste);
float pop(Stapel& s);
bool leer(const Stapel& s);
bool voll(const Stapel& s);

#endif // STAPEL_H
---------------------------------------->8---
#include "stapel.h"           // stapel.cpp

void init(Stapel& s)
{ s.hoehe = 0;
}

void push(Stapel& s, float kiste)
{ s.daten[s.hoehe++] = kiste;
}

float pop(Stapel& s)
{ return s.daten[--s.hoehe];
}

bool leer(const Stapel& s)
{ return s.hoehe == 0;
}

bool voll(const Stapel& s)
{ return s.hoehe == MAX;
}
---------------------------------------->8---
#include <iostream>            // prog.cpp
#include "stapel.h"

int main()
{
  Stapel stapel;
  init(stapel);
  float kiste;

  while (!voll(stapel) && std::cin >> kiste)
  { push(stapel, kiste);
  }
  while (!leer(stapel))
  { std::cout << pop(stapel) << '\n';
  }
  return 0;
}
```

datenabstrahiert (opaker Typ)

```
#ifndef STAPEL_H              // stapel.h
#define STAPEL_H
struct  StapelImpl;
typedef StapelImpl* Stapel;

Stapel create();
void push(Stapel s, float kiste);
float pop(Stapel s);
bool leer(const Stapel s);
bool voll(const Stapel s);
void destroy(Stapel s);
#endif // STAPEL_H
------------------------------------>8---
#include "stapel.h"           // stapel.cpp

const int MAX = 10;
struct StapelImpl
{ int hoehe;
  float daten[MAX];
};

Stapel create()
{ Stapel s = new StapelImpl();
  s.hoehe = 0;
  return s;
}
void destroy(Stapel s)
{ delete s;
}
void push(Stapel s, float kiste)
{ s->daten[s->hoehe++] = kiste;
}
float pop(Stapel s)
{ return s->daten[--s->hoehe];
}
bool leer(const Stapel s)
{ return s->hoehe == 0;
}
bool voll(const Stapel s)
{ return s->hoehe == MAX;
}
------------------------------------>8---
#include <iostream>           // prog.cpp
#include "stapel.h"

int main()
{ Stapel stapel = create();
  float kiste;
  while (!voll(stapel) && std::cin >> kiste)
  { push(stapel, kiste);
  }
  while (!leer(stapel))
  { std::cout << pop(stapel) << '\n';
  }
  destroy(stapel);
  return 0;
}
```

objektorientiert

```
#ifndef STAPEL_H              // stapel.h
#define STAPEL_H

class Stapel
{
public:
  Stapel();
  void push(float kiste);
  float pop();
  bool leer() const { return hoehe == 0;   }
  bool voll() const { return hoehe == MAX; }
private:
  static const int MAX = 10;
  int hoehe;
  float daten[MAX];
};

#endif // STAPEL_H
------------------------------------>8---
#include "stapel.h"           // stapel.cpp

Stapel::Stapel()
: hoehe(0)
{
}

void Stapel::push(float kiste)
{
  daten[hoehe++] = kiste;
}

float Stapel::pop()
{
  return daten[--hoehe];
}
------------------------------------>8---
#include <iostream>           // prog.cpp
#include "stapel.h"

int main()
{
  Stapel stapel;
  float kiste;

  while (!stapel.voll() && std::cin >> kiste)
  {
    stapel.push(kiste);
  }

  while (!stapel.leer())
  {
    std::cout << stapel.pop() << '\n';
  }
  return 0;
}
```

generisch (typparametrisiert)

```
#ifndef STAPEL_H              // stapel.h
#define STAPEL_H

template <class Kiste, int MAX>
class Stapel
{
public:
  Stapel() : hoehe(0) {}
  void push(Kiste kiste);
  Kiste pop();
  bool leer() const { return hoehe == 0;   }
  bool voll() const { return hoehe == MAX; }
private:
  int hoehe;
  Kiste daten[MAX];
};

template <class Kiste, int MAX>
void Stapel<Kiste,MAX>::push(Kiste kiste)
{
  daten[hoehe++] = kiste;
}

template <class Kiste, int MAX>
Kiste Stapel<Kiste,MAX>::pop()
{
  return daten[--hoehe];
}

#endif // STAPEL_H
------------------------------------>8---
#include <iostream>           // prog.cpp
#include "stapel.h"

int main()
{
  Stapel<float, 10> stapel;
  float kiste;

  while (!stapel.voll() && std::cin >> kiste)
  {
    stapel.push(kiste);
  }

  while (!stapel.leer())
  {
    std::cout << stapel.pop() << '\n';
  }
  return 0;
}
```