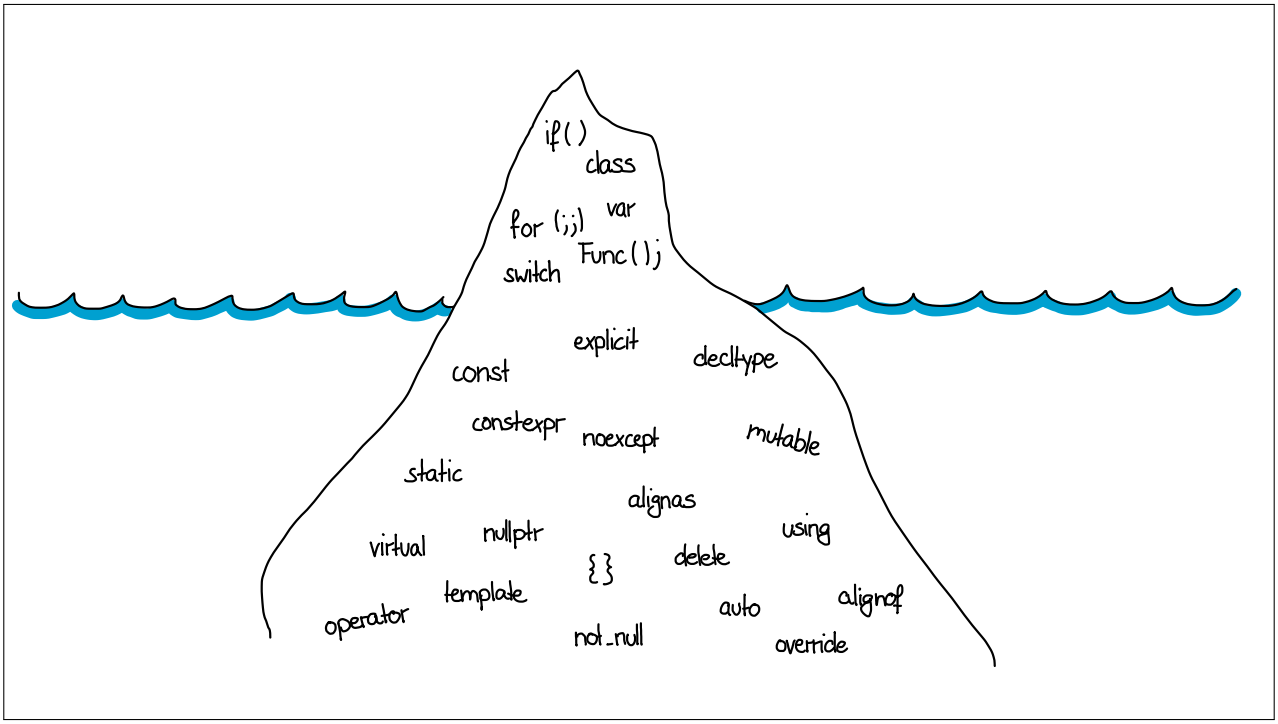
Nutzen Sie die Macht der Programmiersprache

Programmieren einmal anders

Andreas Fertig
https://www.AndreasFertig.Info
post@AndreasFertig.Info



The Good, the Bad and the Ugly

```
#include "stdio.h"
#define e 3
#define g (e/e)
#define h ((g+e)/2)
#define f (e-g-h)
#define j (e*e-g)
#define k (j-h)
#define l(x) tab2[x]/h
#define m(n,a) ((n&(a))==(a))

long tab1[]={ 989L,5L,26L,0L,88319L,123L,0L,9367L };
int tab2[]={ 4,6,10,14,22,26,34,38,46,58,62,74,82,86 };

main(m1,s) char *s; {
    int a,b,c,d,o[k],n=(int)s;
    if(m1==1){ char b[2*j+f-g]; main(l(h+e)+h+e,b); printf(b); }
    else switch(m1-=h){
  case f:
      a=(b=(c=(d=g)<<g)<<g)<<g;
      return(m(n,a|c)|m(n,b)|m(n,a|d)|m(n,c|d));
  case h:
      for(a=f;a<j;++a)if(tab1[a]&&!(tab1[a]%((long)l(n))))return(a);
  case g:
      if(n<h)return(g);
      if(n<j){n-=g;c='D';o[f]=h;o[g]=f;}
      else{c='\r'-'\b';n-=j-g;o[f]=o[g]=g;}
      if((b=n)>=e)for(b=g<<g;b<n;++b)o[b]=o[b-h]+o[b-g]+c;
      return(o[b-g]%n+k-h);
  default:
      if(m1-=e) main(m1-g+e+h,s+g); else *(s+g)=f;
      for(*s=a=f;a<e;) *s=(*s<<e)|main(h+a++,(char *)m1);
  }
}
```

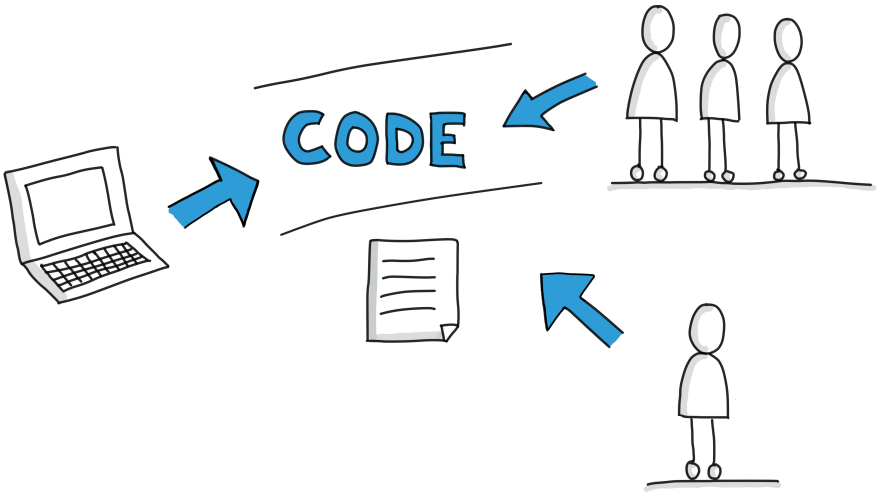
Quelle: [1]

Programmiersprache

> komplexes Regelsystem als zentrales menschliches Verständigungsmittel.
>
> — Pons [2]

> (historisch entstandenes und sich entwickelndes) System von Zeichen und Regeln, das einer Sprachgemeinschaft als Verständigungsmittel dient;
>
> — Duden [3]

https://www.AndreasFertig.Info
post@AndreasFertig.Info



> Code is going to live a long time, and be read many times. We choose explicitly to optimize for the reader, not the writer.
>
> — Winters [4]

https://www.AndreasFertig.Info
post@AndreasFertig.Info

/* Kommentare */

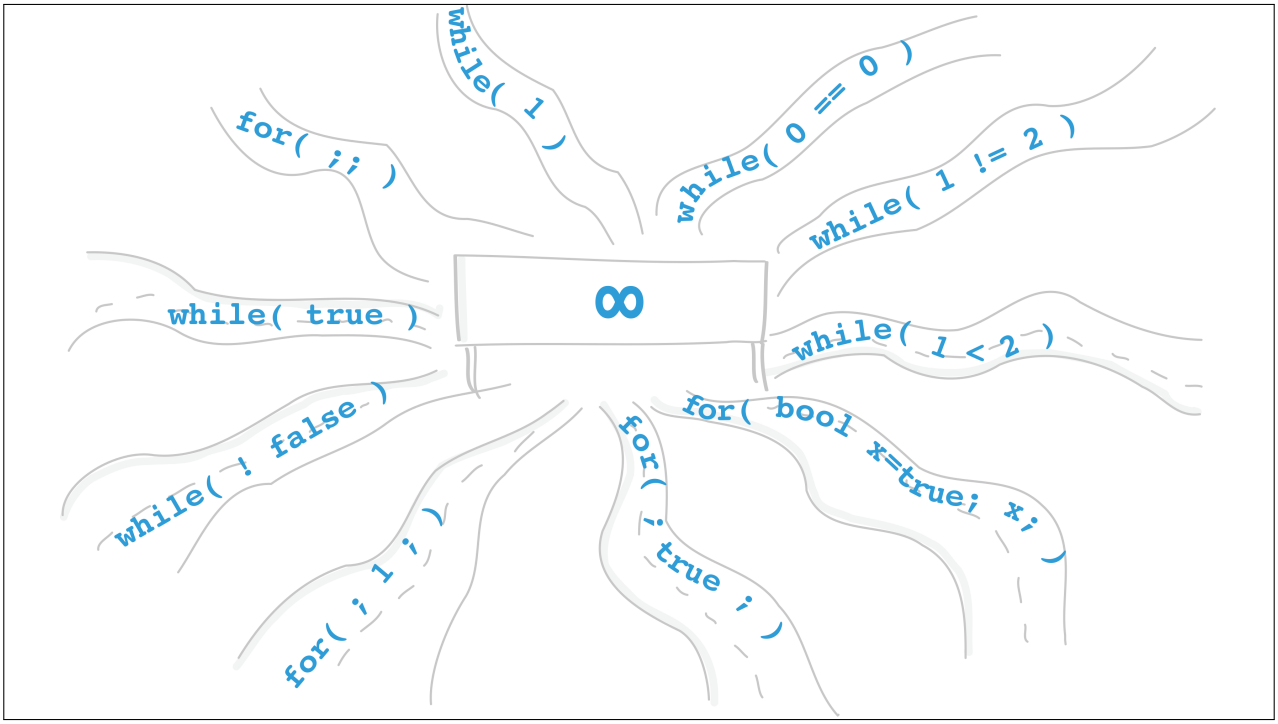
Variante A:

```
// Check to see if the employee is eligible for full benefits
if ((employee.flags & HOURLY_FLAG) && (employee.age > 65))
```

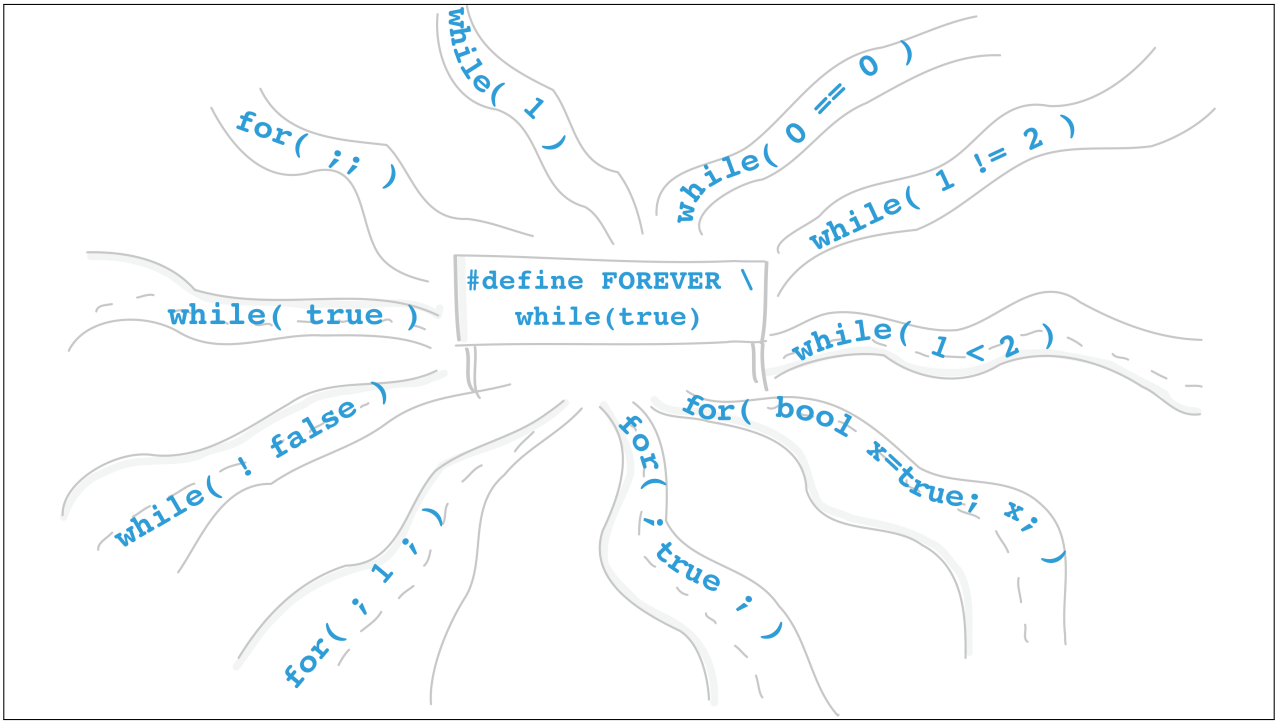
Variante B:

```
if (employee.isEligibleForFullBenefits())
```

Quelle: [5]



https://www.AndreasFertig.Info
post@AndreasFertig.Info



for each

```
1 std::vector<int> numbers{1, 2, 3, 5};
2
3 for(int i=0; i < numbers.size(); ++i)
4 {
5     // use v[i]
6 }
```

for each

```
std::vector<int> numbers{1, 2, 3, 5};

for(auto& it : numbers)
{
  // use it
}
```

```
bool Drive(
    const License& l,
    const SafetyTraining* t
          );
```


https://www.AndreasFertig.Info
post@AndreasFertig.Info

not null

```
char* strncpy(char* dst,
              const char* src,
              size_t n)
{
    if( !src || !dst ) { return nullptr; }

    char* s1 = dst;
    for( ; (0 < n) && ('\0' != *src); --n ) {
      *s1++ = *src++;
    }

    return dst;
}
```

not null

```
char* strncpy(not_null<char*> dst,
              not_null<const char*> src,
              size_t n)
{
    char* s1 = dst;
    const char* s2 = src;
    for( ; (0 < n) && ('\0' != *s2); --n ) {
      *s1++ = *s2++;
    }

    return dst;
}
```

Mehr zur GSL: [6]

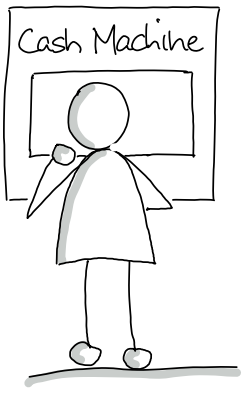
https://www.AndreasFertig.Info
post@AndreasFertig.Info

not null

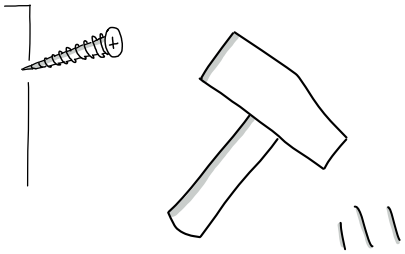
```
1 int sstrcpy( span<char>  dst,
2               span<const char>  src)
3 {
4     const int n = MIN( dst.length(), src.length() );
5
6     copy( src, dst );
7
8     return n;
9 }
```

Mehr zur GSL: [6]

const



```
const_cast
dynamic_cast

static_cast
reinterpret_cast

narrow_cast
```



> If you use object-oriented technology, you can take any class someone else wrote, and by using it as a base class, refine it to do a similar task.
>
> — Murray [7]

https://www.AndreasFertig.Info
post@AndreasFertig.Info

```
1 class Base
2 {
3 };
4
5 class Derived : public Base
6 {
7 };
```

```
1 class Base
2 {
3 };
4
5 class Derived  final  : public Base
6 {
7 };
```

virtual & override

```
class A
{
protected:
    virtual void Func() const;
};

class B : public A
{
protected:
    void Func() const;
};

class C : public A
{
protected:
    virtual void Func() const;
};
```

virtual & override

```
class A
{
protected:
    virtual void Func() const;
};

class B : public A
{
protected:
    void Func() const override;
};

class C : public A
{
protected:
    void Func() const override;
};
```

https://www.AndreasFertig.Info
post@AndreasFertig.Info

Scope Guard

```
void ChangeScreen(Screen& newScreen)
{
  lock();
    screen = &newScreen;
  unlock();
}

void Update()
{
  lock();

  if( ! updateTriggered ) {
    updateTriggered = true;
    unlock();

    SendUpdateNotificationEvent();
  } else {
    unlock();
  }
}
```

Scope Guard

```
void ChangeScreen(Screen& newScreen)
{
  Lock lock;
  screen = &newScreen;
}

void Update()
{
  Lock lock;

  if( ! updateTriggered ) {
    updateTriggered = true;
    lock.Unlock();

    SendUpdateNotificationEvent();
  }
}
```

```
class Lock {
public:
  Lock() : mLocked(true)
  { lock(); }
  ~Lock() { Unlock(); }

  void Unlock() {
    if( mLocked ) {
      mLocked = false;
      unlock();
    }
  }

private:
  bool mLocked;
};
```


https://www.AndreasFertig.Info
post@AndreasFertig.Info

Scope Guard

```
void ChangeScreen(Screen& newScreen)
{
  SCOPE_GUARD(Lock);
  screen = &newScreen;
}

void Update()
{
  SCOPE_GUARD(Lock, lock);

  if( ! updateTriggered ) {
    updateTriggered = true;
    lock.Unlock();

    SendUpdateNotificationEvent();
  }
}
```

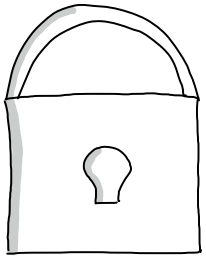
Scope Guard

```
#define CONCATX(x,y) x##y
#define CONCAT(x, y) CONCATX(x,y)

#define ANON_VAR(name) CONCAT(name, __LINE__)

#define RESOLVE_NAME(_0, _1, _2, NAME, ...) NAME

#define SCOPE_GUARD(clsName,...) \
    RESOLVE_NAME(_0, ##__VA_ARGS__, INVALID_SCOPE_GUARD, NAMED_SCOPE_GUARD, /
         ANON_SCOPE_GUARD_WRAPPER)(clsName, __VA_ARGS__)


#define INVALID_SCOPE_GUARD(clsName, ...) \
    static_assert(false, "Error: Invalid use of macro")

#define NAMED_SCOPE_GUARD(clsName, name) \
    clsName name

#define ANON_SCOPE_GUARD_WRAPPER(clsName, _0) \
    ANON_SCOPE_GUARD(clsName)

#define ANON_SCOPE_GUARD(clsName) \
    clsName ANON_VAR(___scopeGuard)
```

Idee: [8]

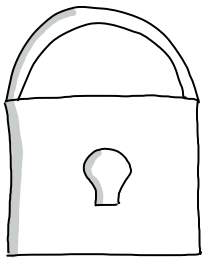
Lock

```
1
2
3
4
5
6  void ChangeScreen(Screen& newScreen)
7  {
8    _lock()
9    {
10      if( screen == &newScreen ) {
11          return;
12      }
13
14      screen = &newScreen;
15   }
16
17   SendUpdateNotificationEvent();
18 }
```



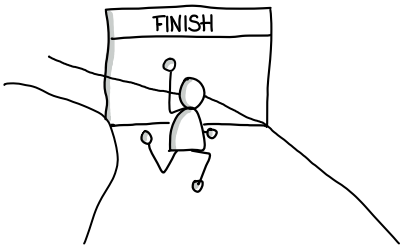
Lock

```
1  #define _lock()          \
2    if( Lock lock; false ) \
3    {}                     \
4    else
5
6  void ChangeScreen(Screen& newScreen)
7  {
8    _lock()
9    {
10      if( screen == &newScreen ) {
11          return;
12      }
13
14      screen = &newScreen;
15   }
16
17   SendUpdateNotificationEvent();
18 }
```



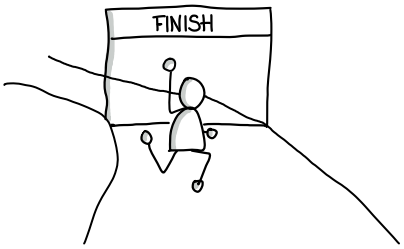
https://www.AndreasFertig.Info
post@AndreasFertig.Info

One `final` thing




```
size_t ReadData(char* buffer, size_t bufLen)
{
    int fd = Open(/* some well known file*/);


    if( -1 == fd ) {
        return 0;
    }

    int len = read( fd, buffer, bufLen );

    if( -1 == len ) {
        return 0; // urg: missing close of fd
    }

    close(fd);

    return gsl::narrow_cast<size_t>(len);
}
```

One `final` thing




```
#define CONCAT_IMPL(x,y) x##y
#define CONCAT(x,y) CONCAT_IMPL(x,y)

#define ANON_VAR(str) \
        CONCAT(str, __LINE__)

#define FINALLY \
    auto ANON_VAR(___final) = gsl::finally
```

```
size_t ReadData(char* buffer, size_t bufLen)
{
    int fd = Open(/* some well known file*/);
    FINALLY([&]{ if( -1 != fd ) { close(fd); } });

    if( -1 == fd ) {
        return 0;
    }

    int len = read( fd, buffer, bufLen );

    if( -1 == len ) {
        return 0;
    }

    return gsl::narrow_cast<size_t>(len);
}
```


https://www.AndreasFertig.Info
post@AndreasFertig.Info

```
}
```

Ich bin Fertig.

Available online:

https://www.AndreasFertig.Info

Images by Panther Concepts:

https://panther-concepts.de

Quellen

[1] Holloway B., “Obfuscated "Hello World"”, 1986. andreasfertig.info/lnk/hobr86

[2] Pons , “Sprache, die”, Nov. 2016. andreasfertig.info/lnk/posp16

[3] Duden , “Sprache, die”, Nov. 2016. andreasfertig.info/lnk/dusp16

[4] Winters T., “The Philosophy of Googles Coding Guidlines”, in cppcon, 2014. andreasfertig.info/lnk/witi14

[5] Martin R. C., Clean Code: A Handbook of Agile Software Craftsmanship. Pearson Education, 2008.

[6] Microsoft , “GSL: Guideline Support Library”. andreasfertig.info/lnk/gsl16

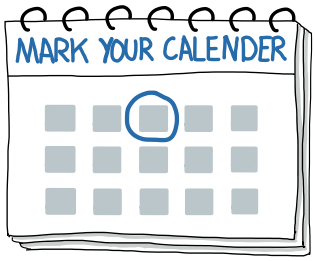
[7] Murray R. B., C++ Strategies and Tactics, Serie: Addison-Wesley professional computing series. Addison-Wesley, 1993.

[8] stackoverflow - netcoder , “Overloading macro on number of arguments”. andreasfertig.info/lnk/sonc16

Nächste Events

- Fast and Small - What are the Costs of Language Features, NDC { Oslo }, June 14 2017
- Use the power of the language, Keynote Address, Clean Code Days, June 21 2017
- C++1x für eingebettete Systeme kompakt, Seminar QA Systems, November 21 2017

Aktuelle Informationen unter:
https://andreasfertig.info/talks.html



Über Andreas Fertig

Andreas arbeitet seit 2010 bei Philips Medizin Systeme als Softwareentwickler mit Schwerpunkt eingebettete Systeme.

Sein Fachgebiet ist der Entwurf und die Implementierung von C++ Softwaresystemen.

Freiberuflich arbeitet er als Dozent und Trainer. Zudem entwickelt er verschiedene Mac OS X Anwendungen.