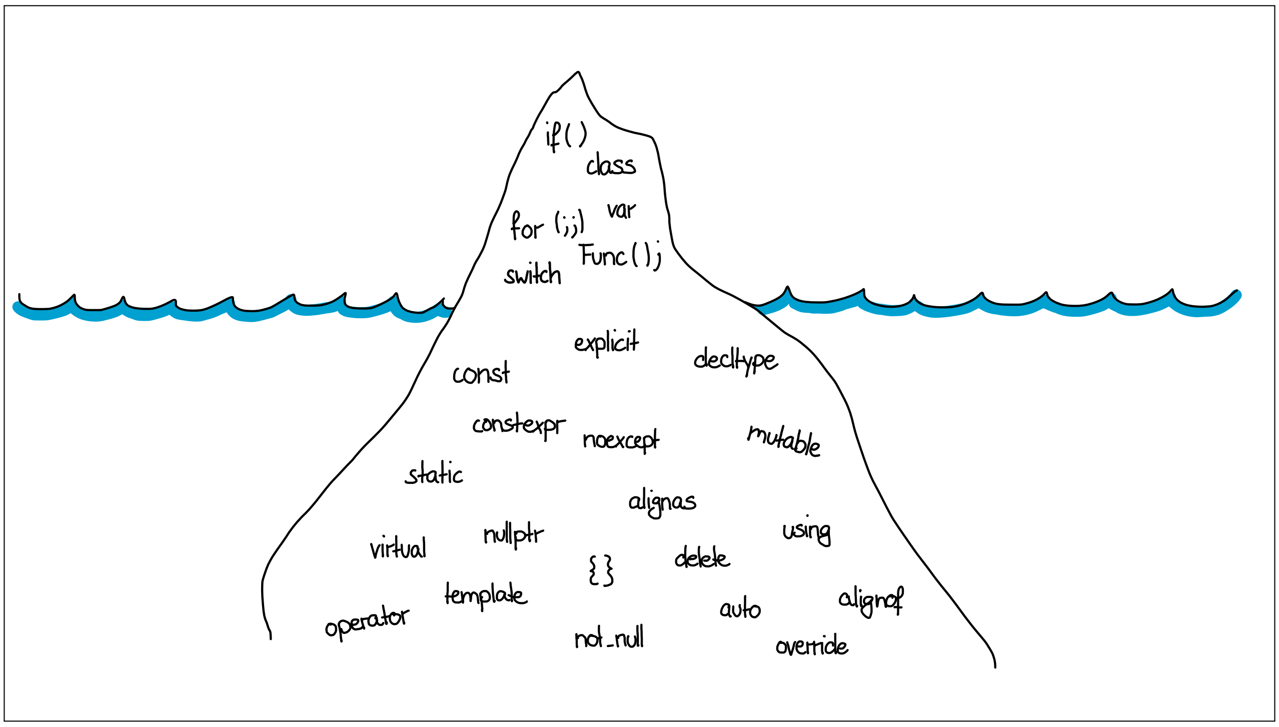
Use the power of the language

Programmieren einmal anders

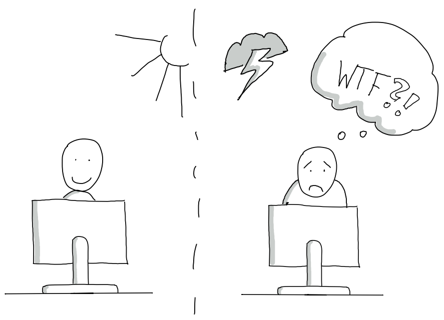
Andreas Fertig
https://www.AndreasFertig.Info
post@AndreasFertig.Info



The Good, the Bad and the Ugly

```c
#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);
  }
}
```

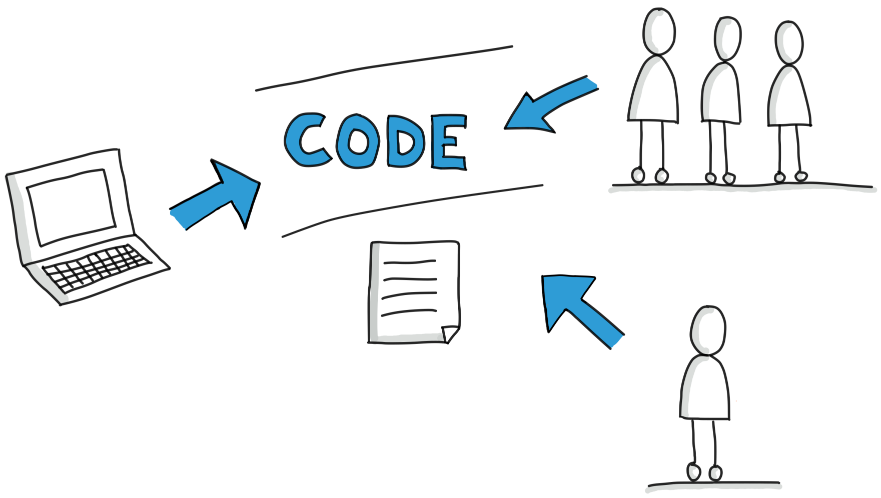
Quelle: [1]



Programmier**sprache**

> 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]



> Programming is the art of telling another human being what one wants the computer to do.
>
> — Knuth [4]

“ ..., programs must be written for people to read, and only incidentally for machines to execute.

— Knuth [4]

“ 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 [5]

/* 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: [6]

“ If a program is incorrect, it matters little what the documentation says.

— Kernighan und Plauger [7]

> Good code needs fewer comments than bad code does.
>
> — Kernighan und Pike [8]

> Code never lies, comments sometimes do.
>
> — Jeffries [9]

> If a program is incorrect, it matters little what the documentation says.
>
> — Kernighan und Plauger [7]

> Good code needs fewer comments than bad code does.
>
> — Kernighan und Pike [8]

> Code never lies, comments sometimes do.
>
> — Jeffries [9]

Viele Leute kommentieren Kommentare.

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

```
if ((employee.flags & HOURLY_FLAG) && (employee.age > 65))
```

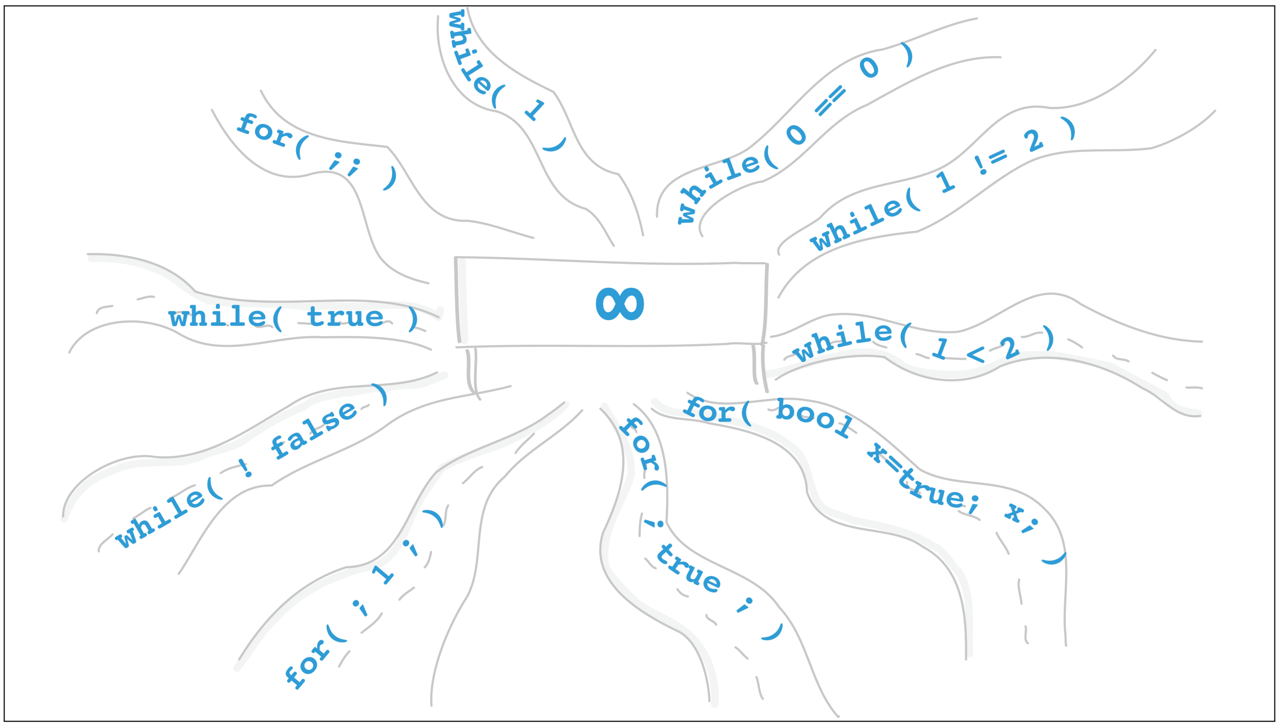
```
static constexpr unsigned int HOURLY_FLAG{ 0x02 };

class Employee
{
public:

    Employee()
    : flags{ 0 }
    , age  { 0 }
    {}

    unsigned int flags;
    int age;
};
```

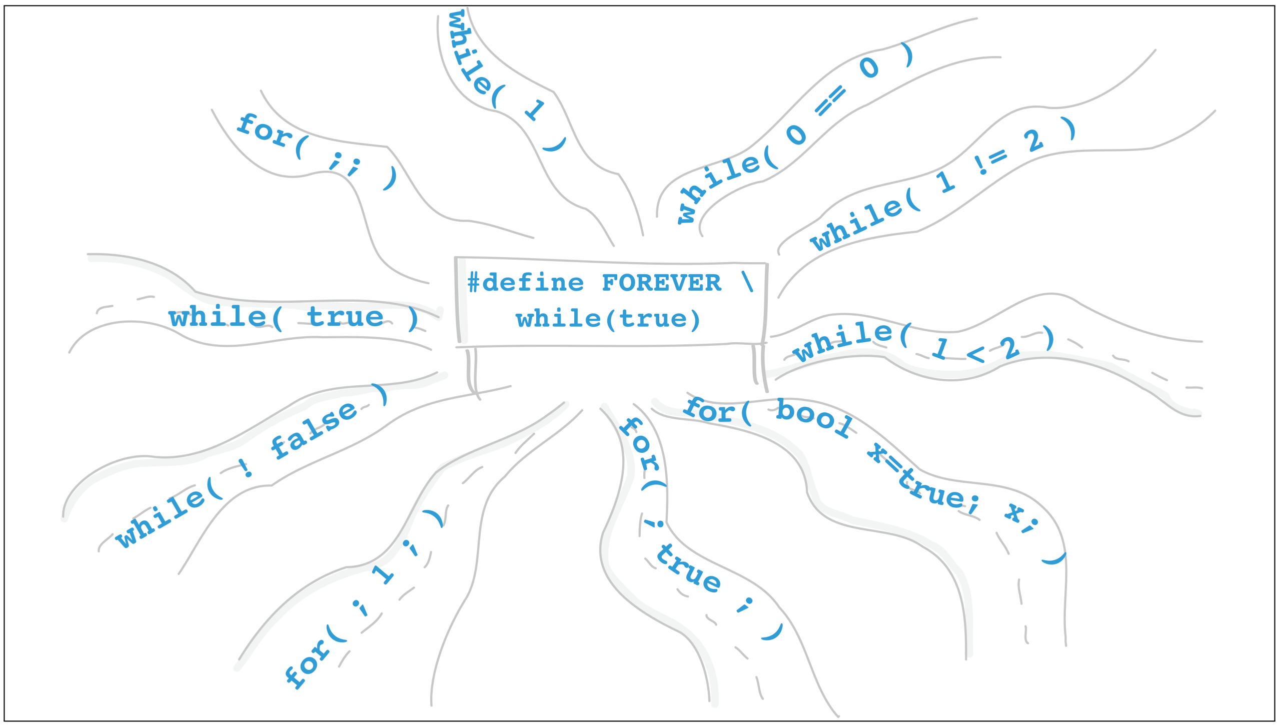
```cpp
if (employee.HasFlagHourly() && employee.HasRetired())
```

```cpp
class Employee
{
public:
    Employee()
    : flags{ 0 }
    , age  { 0 }
    {}

    bool HasFlagHourly() const { return flags & HOURLY_FLAG; }
    bool HasRetired()    const { return age > 65; }

private:
    static constexpr unsigned int HOURLY_FLAG{ 0x02 };

    unsigned int flags;
    int age;
};
```

“ Broadly speaking, the short words are best, and old words when short are best of all.

— Churchill [10]


while(1)
for(;;)
while(0 == 0)
while(1 != 2)
∞
while(true)
while(1 < 2)
while(! false)
for(bool x=true; x;)
for(; 1 ;)
for(; true ;)


while(1)
for(;;)
while(0 == 0)
while(1 != 2)
#define FOREVER \
while(true)
while(true)
while(1 < 2)
while(! false)
for(bool x=true; x;)
for(; 1 ;)
for(; true ;)

for each

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

for(int i=0; i < numbers.size(); ++i)
{
    // use v[i]
}
```

for each

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

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

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

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

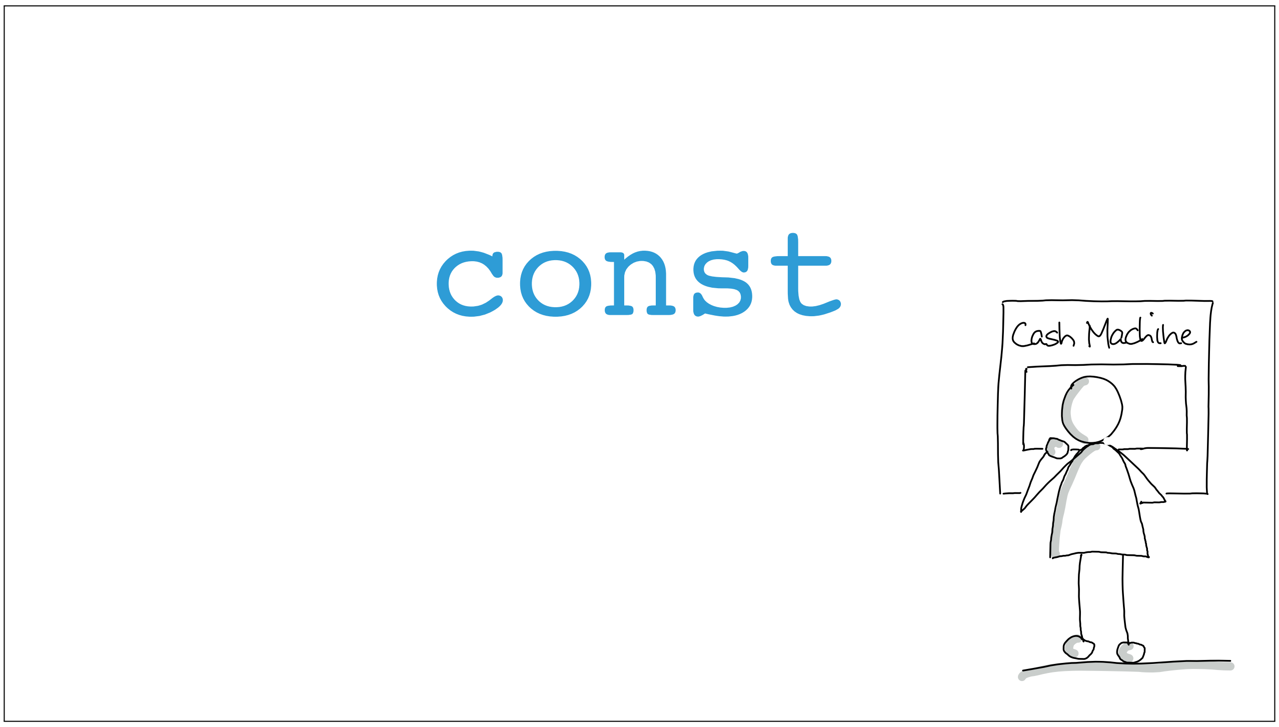
```
char* strncpy(not_null<char*> dst,
              not_null<const char*> src,
              size_t n)
{
    char* s1 = dst;
    const char* s2 = src;
    for( ; (0 < n) && ('\0' != *s2); --n ) {
      *s1++ = *s2++;
    }

    return dst;
}
```

Mehr zur GSL: [11]

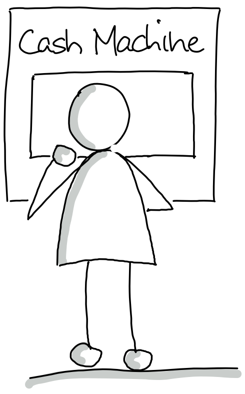
not null

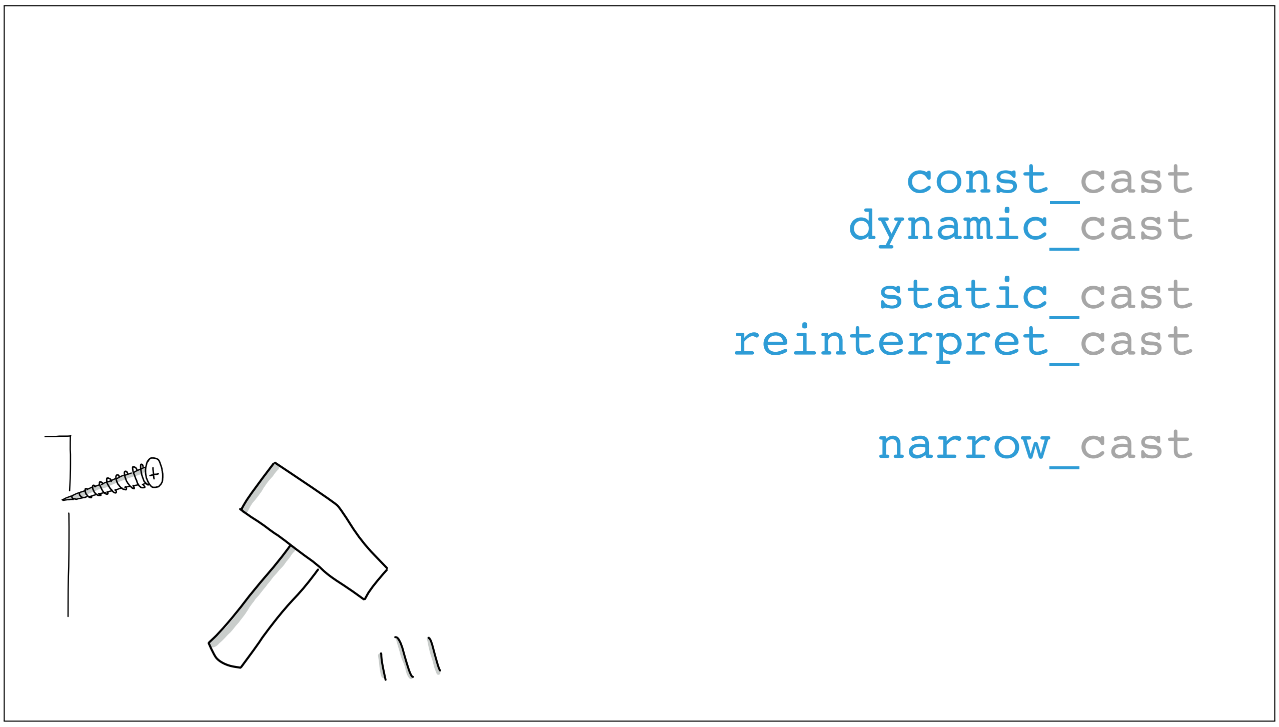
```
int sstrcpy( span<char>  dst,
             span<const char>  src)
{
    const int n = MIN( dst.length(), src.length() );

    copy( src, dst );

    return n;
}
```

Mehr zur GSL: [11]

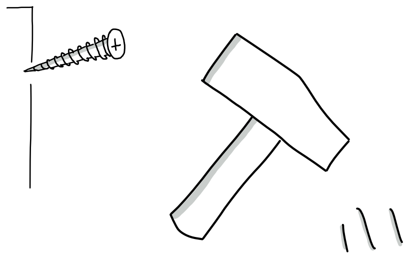


const





```
const_cast
dynamic_cast

static_cast
reinterpret_cast

narrow_cast
```



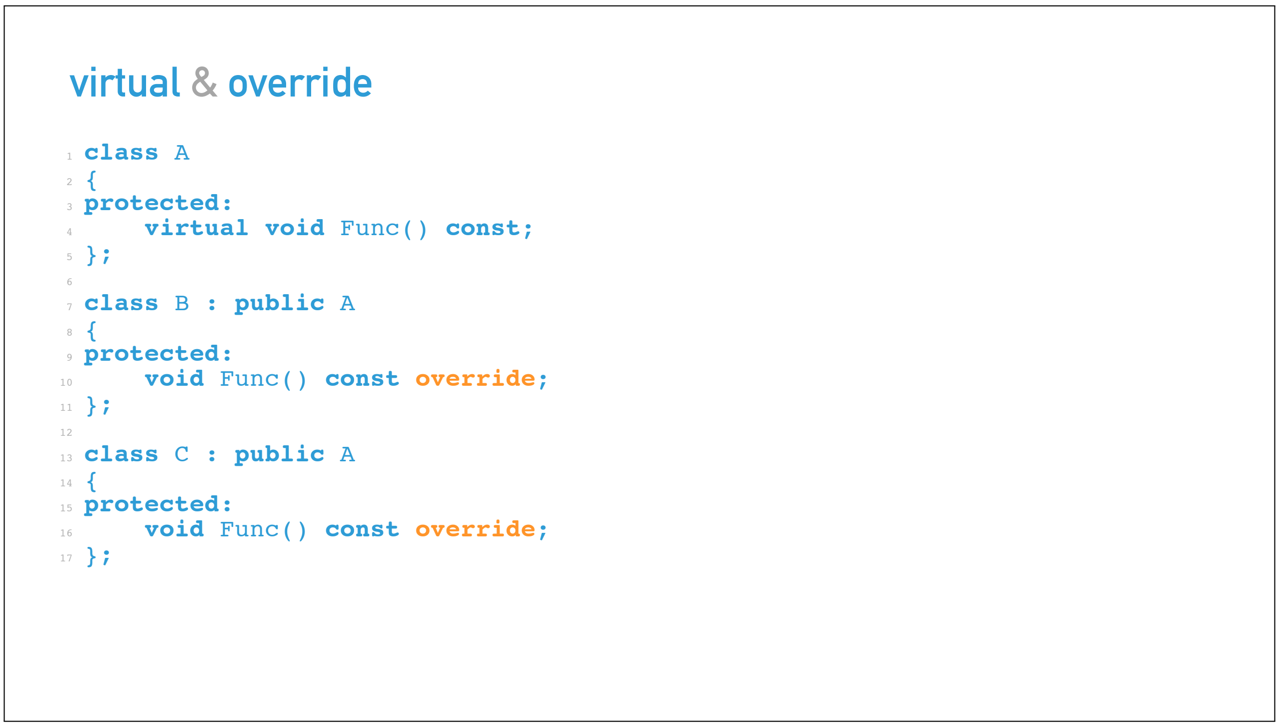
> If you use object-oriented technology, you can take any class someone else wrote, and by using it as a base class, refine it to do a similar task.
>
> — Murray [12]

```
class Base
{
};

class Derived : public Base
{
};
```

```cpp
class Base
{
};

class Derived  final  : public Base
{
};
```

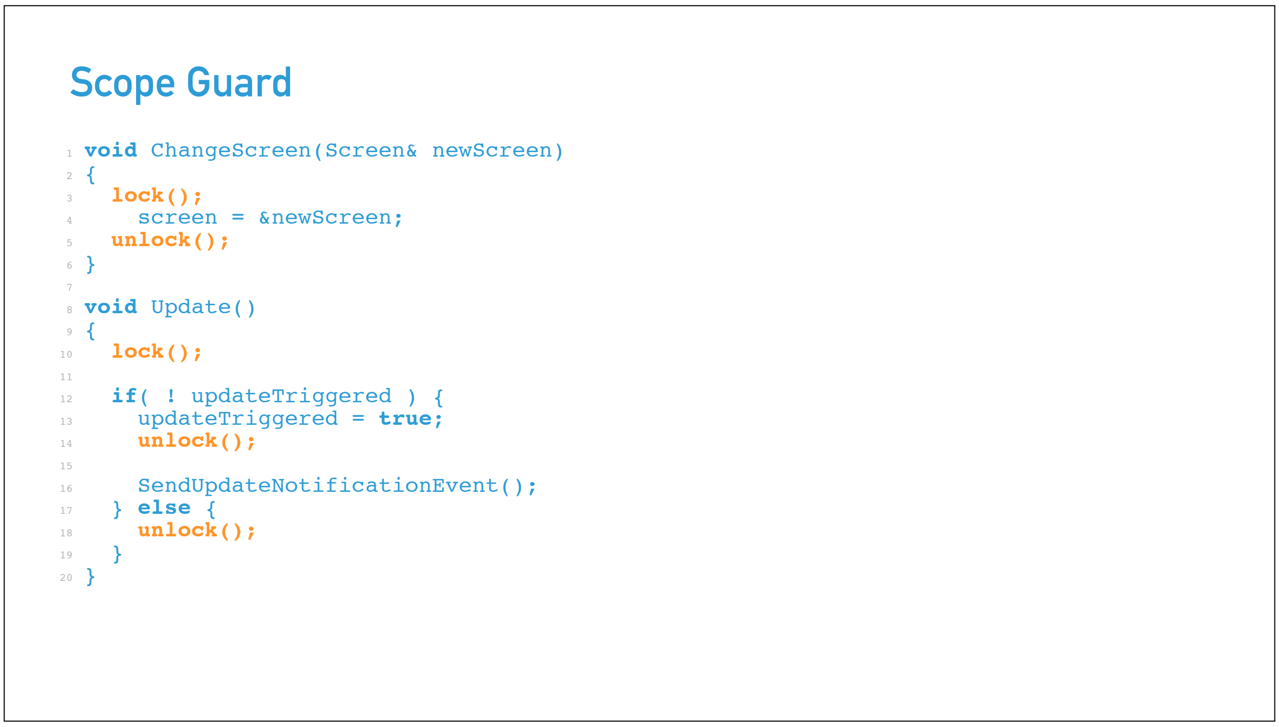
virtual & override

```cpp
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

```cpp
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;
};
```



Scope Guard

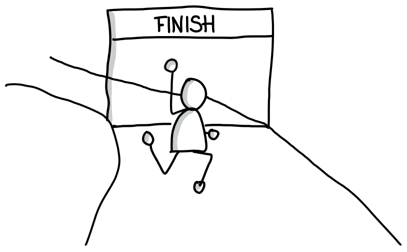
```cpp
void ChangeScreen(Screen& newScreen)
{
  lock();
    screen = &newScreen;
  unlock();
}

void Update()
{
  lock();

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

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

Scope Guard

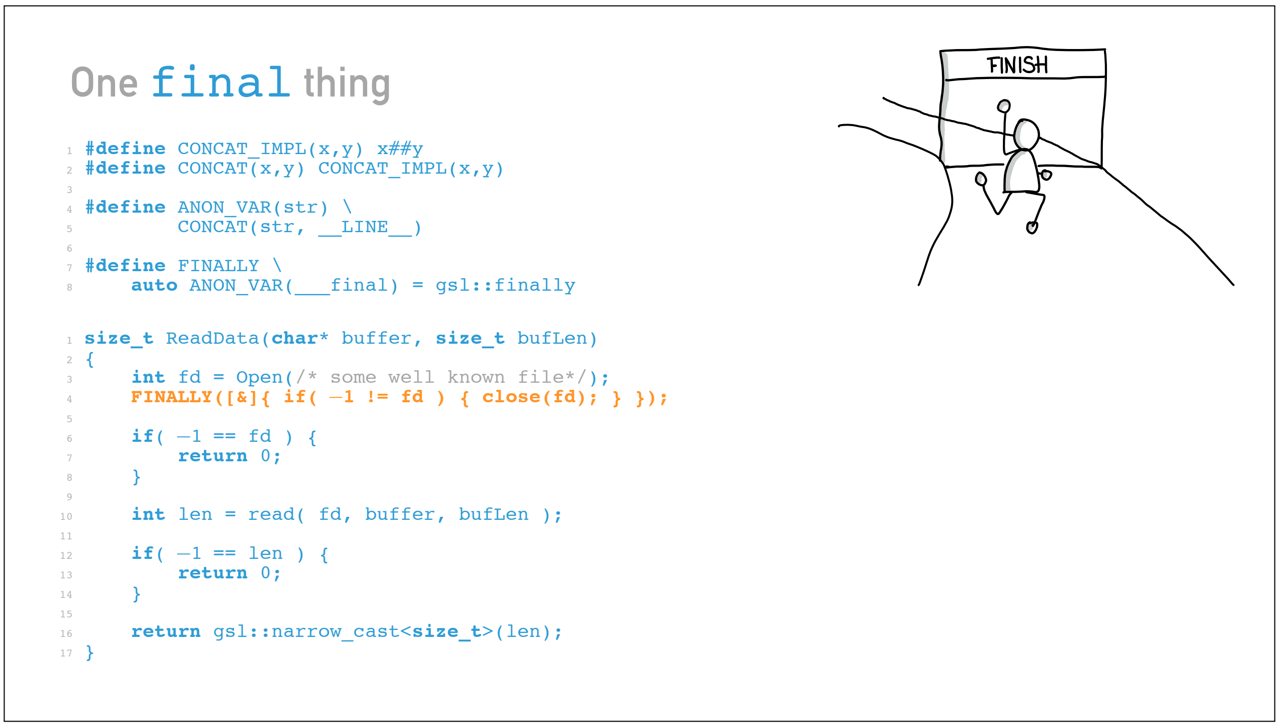
```cpp
void ChangeScreen(Screen& newScreen)
{
  Lock lock;
  screen = &newScreen;
}

void Update()
{
  Lock lock;

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

    SendUpdateNotificationEvent();
  }
}
```

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

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

private:
  bool mLocked;
};
```

One `final` thing



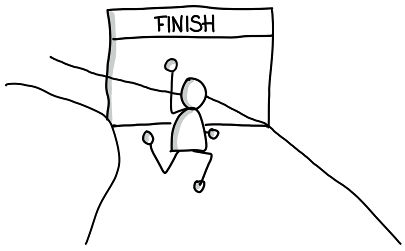
```cpp
size_t ReadData(char* buffer, size_t bufLen)
{
    int fd = Open(/* some well known file*/);


    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);
}
```



}

Ich bin Fertig.

Available online:

https://www.AndreasFertig.Info

Images by Panther Concepts:

https://panther-concepts.de

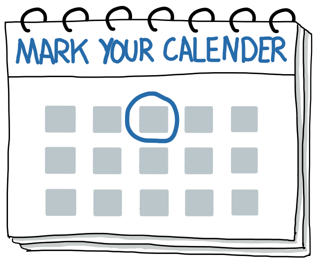
Quellen

[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] Knuth D. E., Structure and Interpretation of Computer Programs.
[5] Winters T., “The Philosophy of Googles Coding Guidlines”, in cppcon, 2014. andreasfertig.info/lnk/witi14
[6] Martin R. C., Clean Code: A Handbook of Agile Software Craftsmanship. Pearson Education, 2008.
[7] Kernighan B. W. und Plauger P., The Elements of Programming Style. McGraw-Hill, 1978.
[8] Kernighan B. und Pike R., The Practice of Programming, Serie: Addison-Wesley professional computing series. Addison-Wesley, 1999.
[9] Jeffries R. (2017, Jun). http://www.azquotes.com/quote/878654
[10] Churchill W. https://simple.wikiquote.org/wiki/Winston_Churchill
[11] Microsoft , “GSL: Guideline Support Library”. andreasfertig.info/lnk/gsl16
[12] Murray R. B., C++ Strategies and Tactics, Serie: Addison-Wesley professional computing series. Addison-Wesley, 1993.

Nächste Events

- 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.