

# Nutzen Sie die Macht der Programmiersprache

Programmieren einmal anders



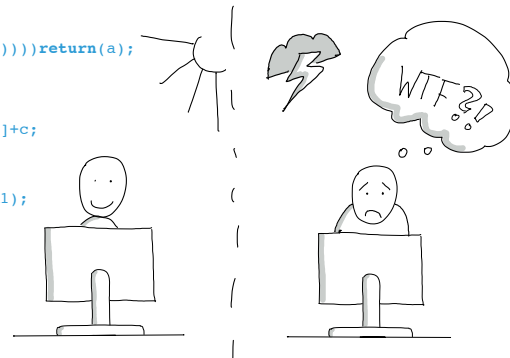
Andreas Fertig

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

## The Good, the Bad and the Ugly

```
1 #include "stdio.h"
2 #define e 3
3 #define g (e/e)
4 #define h ((g+e)/2)
5 #define f (e-g-h)
6 #define j (e*e-g)
7 #define k (j-h)
8 #define l(x) tab2[x]/h
9 #define m(n,a) ((n&(a))==a)
10
11 long tab1[]={ 989L,5L,26L,0L,88319L,123L,0L,9367L };
12 int tab2[]={ 4,6,10,14,22,26,34,38,46,58,62,74,82,86 };
13
14 main(m1,s) char *s; {
15     int a,b,c,d,o[k],n=(int)s;
16     if(m1=1){ char b[2*j+f-g]; main(l(h+e)+h+e,b); printf(b); }
17     else switch(m1-h){
18     case f:
19         a=(b=(c=(d=g)<<g)<<g)<<g);
20         return(m(n,a|c)|m(n,b)|m(n,a|d)|m(n,c|d));
21     case h:
22         for(a=f;a<j;++a)if(tab1[a]&&! (tab1[a]%((long)l(n))))return(a);
23     case g:
24         if(n<h)return(g);
25         if(n<j){n-=g;c='D';o[f]=h;o[g]=f;}
26         else{c='r'-'b';n-=j-g;o[f]=o[g]=g;}
27         if((b=n)>=e)for(b=g<<g;b<n;++b)o[b]=o[b-h]+o[b-g]+c;
28         return(o[b-g]%n+k-h);
29     default:
30         if(m1==e) main(m1-g+e+h,s+g); else *(s+g)=f;
31         for(*s=a=f;a<e; *s=( *s<<e)|main(h+a++,(char *)m1);
32         }
33 }
```

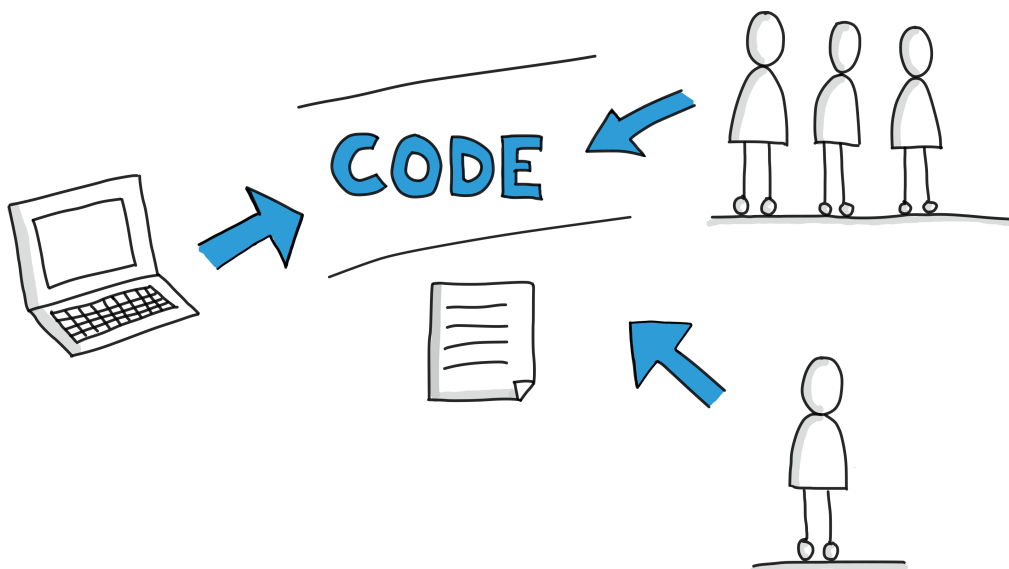
Quelle: [1]



# 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]



“ 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]

## **/\* Kommentare \*/**

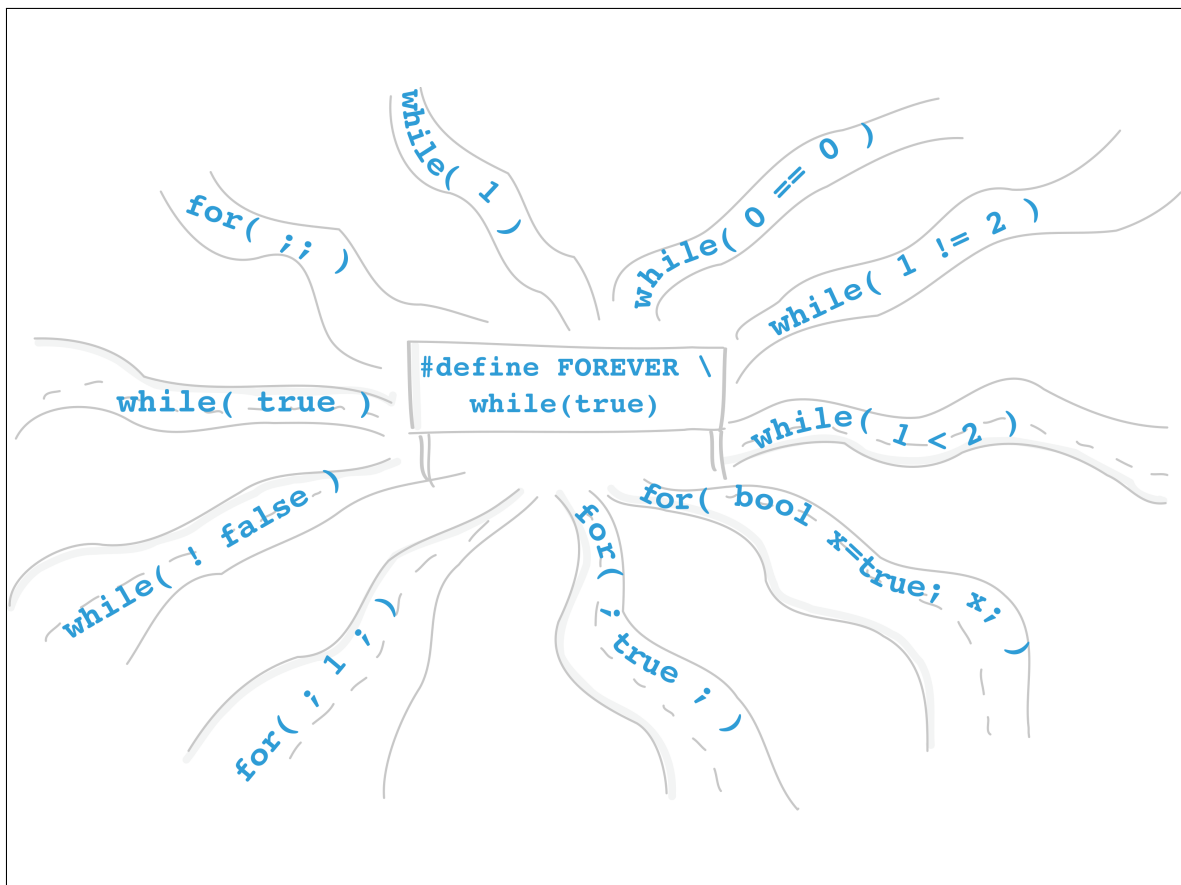
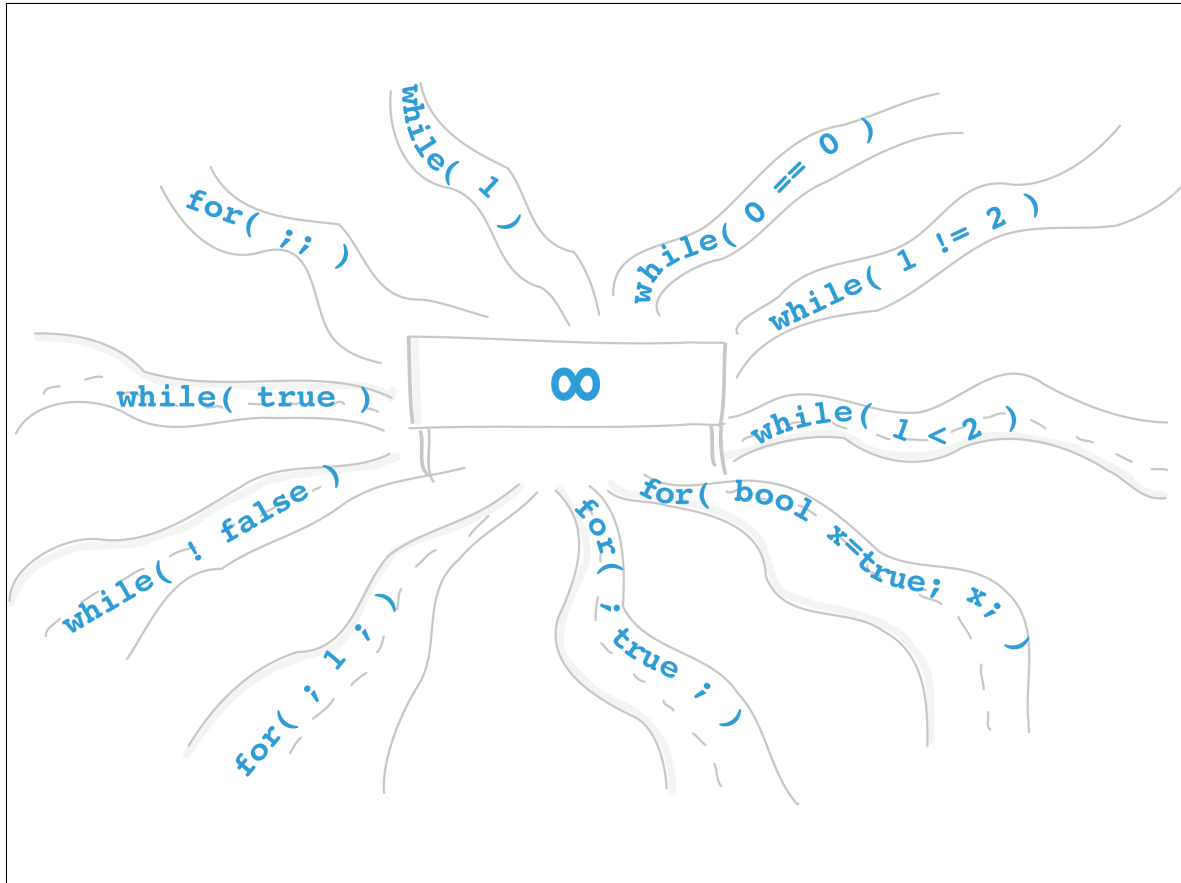
### Variante A:

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

### Variante B:

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

Quelle: [5]



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

## not null

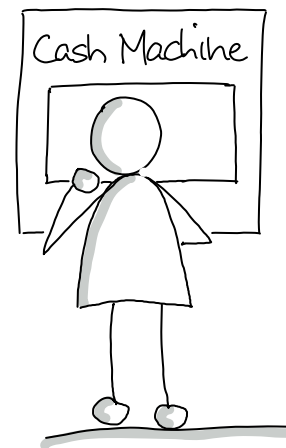
```
1 char* strncpy(char* dst,  
2             const char* src,  
3             size_t n)  
4 {  
5     if( !src || !dst ) { return nullptr; }  
6  
7     char* s1 = dst;  
8     for( ; (0 < n) && ('\0' != *src); --n ) {  
9         *s1++ = *src++;  
10    }  
11  
12    return dst;  
13 }
```

## not null

```
1 char* strncpy(not_null<char*> dst,  
2             not_null<const char*> src,  
3             size_t n)  
4 {  
5     char* s1 = dst;  
6     const char* s2 = src;  
7     for( ; (0 < n) && ('\0' != *s2); --n ) {  
8         *s1++ = *s2++;  
9     }  
10  
11     return dst;  
12 }
```

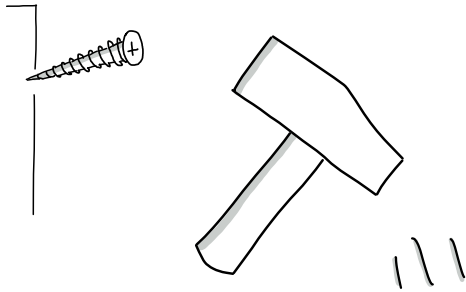
Mehr zur GSL: [6]

## const



`const_cast`  
`dynamic_cast`

`static_cast`  
`reinterpret_cast`



`narrow_cast`

## virtual & override

```
1 class A
2 {
3     protected:
4         virtual void Func() const;
5 };
6
7 class B : public A
8 {
9     protected:
10        void Func() const;
11 };
12
13 class C : public A
14 {
15     protected:
16        virtual void Func() const;
17 };
```

## virtual & override

```
1 class A
2 {
3     protected:
4         virtual void Func() const;
5 };
6
7 class B : public A
8 {
9     protected:
10        void Func() const override;
11 };
12
13 class C : public A
14 {
15     protected:
16        void Func() const override;
17 };
```

## Scope Guard

```
1 void ChangeScreen(Screen& newScreen)
2 {
3     lock();
4     screen = &newScreen;
5     unlock();
6 }
7
8 void Update()
9 {
10    lock();
11
12    if( ! updateTriggered ) {
13        updateTriggered = true;
14        unlock();
15
16        SendUpdateNotificationEvent();
17    } else {
18        unlock();
19    }
20 }
```



# Scope Guard

```

1 void ChangeScreen(Screen& newScreen)
2 {
3     Lock lock;
4     screen = &newScreen;
5 }
6
7 void Update()
8 {
9     Lock lock;
10
11     if( ! updateTriggered ) {
12         updateTriggered = true;
13         lock.Unlock();
14
15         SendUpdateNotificationEvent();
16     }
17 }

```

```

1 class Lock {
2 public:
3     Lock() : mLocked(true)
4         { lock(); }
5     ~Lock() { Unlock(); }
6
7     void Unlock() {
8         if( mLocked ) {
9             mLocked = false;
10            unlock();
11        }
12    }
13
14 private:
15     bool mLocked;
16 };

```

# Scope Guard

```

1 void ChangeScreen(Screen& newScreen)
2 {
3     SCOPE_GUARD(Lock);
4     screen = &newScreen;
5 }
6
7 void Update()
8 {
9     SCOPE_GUARD(Lock, lock);
10
11     if( ! updateTriggered ) {
12         updateTriggered = true;
13         lock.Unlock();
14
15         SendUpdateNotificationEvent();
16     }
17 }

```

# Scope Guard

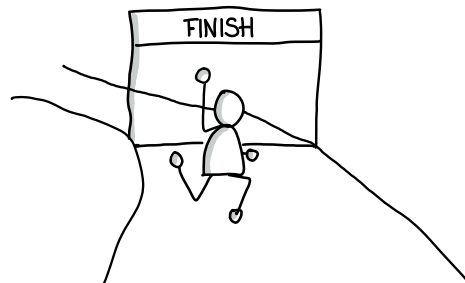
```

1 #define CONCATX(x,y) x##y
2 #define CONCAT(x, y) CONCATX(x,y)
3
4 #define ANON_VAR(name) CONCAT(name, __LINE__)
5
6 #define RESOLVE_NAME(_0, _1, _2, NAME, ...) NAME
7
8 #define SCOPE_GUARD(clsName,...) \
9     RESOLVE_NAME(_0, ## __VA_ARGS__, INVALID_SCOPE_GUARD, /
10     NAMED_SCOPE_GUARD, ANON_SCOPE_GUARD_WRAPPER)(clsName, /
11     __VA_ARGS__)
12
13 #define INVALID_SCOPE_GUARD(clsName, ...) \
14     static_assert(false, "Error: Invalid use of macro")
15
16 #define NAMED_SCOPE_GUARD(clsName, name) \
17     clsName name
18
19 #define ANON_SCOPE_GUARD_WRAPPER(clsName, _0) \
20     ANON_SCOPE_GUARD(clsName)
21
22 #define ANON_SCOPE_GUARD(clsName) \
23     clsName ANON_VAR(__scopeGuard)

```

Idee: [7]

## One final thing

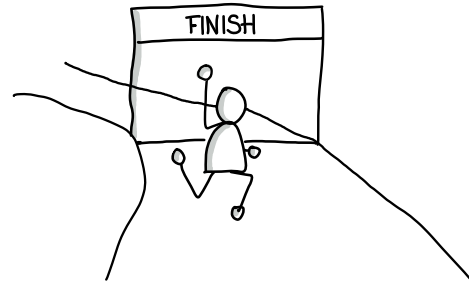


```

1 size_t ReadData(char* buffer, size_t bufLen)
2 {
3     int fd = Open(/* some well known file*/);
4
5     if( -1 == fd ) {
6         return 0;
7     }
8
9     int len = read( fd, buffer, bufLen );
10
11     if( -1 == len ) {
12         return 0; // urg: missing close of fd
13     }
14
15     close(fd);
16
17     return gsl::narrow_cast<size_t>(len);
18 }
19

```

# One final thing



```
1 #define CONCAT_IMPL(x,y) x##y
2 #define CONCAT(x,y) CONCAT_IMPL(x,y)
3
4 #define ANON_VAR(str) \
5     CONCAT(str, __LINE__)
6
7 #define FINALLY \
8     auto ANON_VAR(__final) = gsl::finally
9
10
11 size_t ReadData(char* buffer, size_t bufLen)
12 {
13     int fd = Open(/* some well known file*/);
14     FINALLY([&]{ if( -1 != fd ) { close(fd); } });
15
16     if( -1 == fd ) {
17         return 0;
18     }
19
20     int len = read( fd, buffer, bufLen );
21
22     if( -1 == len ) {
23         return 0;
24     }
25
26     return gsl::narrow_cast<size_t>(len);
27 }
```

```
}
```

Ich bin Fertig.

# Danke!

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](http://andreasfertig.info/lnk/hobr86)
- [2] Pons , "Sprache, die", Nov. 2016. [andreasfertig.info/lnk/posp16](http://andreasfertig.info/lnk/posp16)
- [3] Duden , "Sprache, die", Nov. 2016. [andreasfertig.info/lnk/dusp16](http://andreasfertig.info/lnk/dusp16)
- [4] Winters T., "The Philosophy of Googles Coding Guidelines", in cppcon, 2014. [andreasfertig.info/lnk/witi14](http://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](http://andreasfertig.info/lnk/gsl16)
- [7] stackoverflow - netcoder , "Overloading macro on number of arguments". [andreasfertig.info/lnk/sonc16](http://andreasfertig.info/lnk/sonc16)

## Über Andreas Fertig



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

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

Er ist auch Dozent und Trainer.