

---

# Advanced practical Programming for Scientists

**Thorsten Koch**

Zuse Institute Berlin

TU Berlin

SS2017

---

- Use descriptive names for globals, short names for locals
- Be consistent
- Use active names for functions
- Be accurate
- Indent to show structure
- Use the natural form for expressions
- Parenthesize to resolve ambiguity
- Break up complex expressions
- Be clear
- Be careful with side effects
- Use a consistent indentation and brace style
- Use idioms for consistency
- Give names to magic numbers

Integer (8, 16, 32, 64 bit)

Float (single 32 bit, double 64 bit, quad 128 bit) precision

Characters? (UTF-8)

Strings?

Vectors

Compound data types

## Imperative

loc= xxx

val = yyy

If is\_ok(loc) and is\_valid(val) then

add(container, val)

## OO

If loc.is\_ok() and val.is\_valid() then

container.add(val)

container.size() vs. size(container)

^ask object

^apply on data

Structured:

While not end of file do

Process line

Functional:

Process(forall lines in file)

Loops vs. Recursion

Add(5, 7)

Add(12.3, 34.7)

Add("hallo", "wie")

a **type signature** or **type annotation** defines the inputs and outputs for a [function](#), [subroutine](#) or [method](#). A type signature includes the number of arguments, the types of arguments and the order of the arguments contained by a function. A type signature is typically used during overload resolution for choosing the correct definition of a function to be called among many overloaded forms.

```
class Foo {  
    public:  
        Foo (int foo) : m_foo (foo) { }  
        int GetFoo () { return m_foo; }  
    private:  
        int m_foo;  
};  
void DoBar (Foo foo)  
{  
    int i = foo.GetFoo ();  
}  
int main ()  
{  
    DoBar (42);  
}
```

You have a

```
MyString(int size)
```

class with a constructor that constructs a string of the given size.

You have a function

```
print(const MyString&),
```

and you call it with `print(3)`.

You expect it to print "3", but it prints an empty string of length 3 instead.

<http://stackoverflow.com/questions/121162/what-does-the-explicit-keyword-mean-in-c>



```
if ( (country == SING) || (country == BRNI) ||  
    (country == POL) || (country == ITALY) )  
{  
/*  
* If the country is Singapore, Brunei or Poland  
* then the current time is the answer time  
* rather than the off hook time.  
* Reset answer time and set day of week.  
* /  
...  

```

```
for (theElementIndex = 0;  
    theElementIndex < numberOfElements;  
    theElementIndex++)  
    elementArray[theElementIndex] = theElementIndex;
```

```
for (i = 0; i < nelems; i++)  
    elem[i] = i;
```

```
enum { DANGER, CAUTION, CLEAR} the_signal;
```

```
If (CLEAR == the_signal)
{
    open_gates();
    start_train();
}
; = = 4
```

```
#define isupper(c) ((c) >= 'A' && (c) <= 'Z')
```

parameter `c` occurs twice in the body of the macro.

If `isupper` is called in a context like this,

```
while (isupper(c = getchar()))
```

```
int main()
{
    int const    fixed = 20;
    int*         var;
    int const**  constptr;

    constptr     = &var;
    *constptr    = &fixed;
    *var         = 30;

    printf("x=%d, y=%d\n", fixed, *var);
}
```

```

1  double  s[2048];
2  double  e = 1;
3  int      n = 0;
4
5  do { n = n + 1;    e = e / 2;    s[n] = 1 + e; }
6
7  while (    e > 0);           /* Alternative 1 */
8  while (1 + e > 1);           /* Alternative 2 */
9  while (s[n] > 1);           /* Alternative 3 */

```

Does the loop terminate?

Will the program crash?

If it terminates will  $n$  have the same value in all alternatives? (Lines 7,8,9)

```

1  double  s[2048];
2  double  e = 1;
3  int      n = 0;
4
5  do { n = n + 1;    e = e / 2;    s[n] = 1 + e; }
6
7  while (    e > 0);          /* n = 1075          */
8  while (1 + e > 1);          /* n = 64      (Intel-P4) */
9  while (s[n] > 1);           /* n = 53          */

```

The loop will terminate, the program will not crash and  $n$  is different in most cases, depending on the architecture, the compiler, and the switches.

```
a /**  
/**/ b
```

In old C: a / b

In C++ : a



Combinatorial Optimization at Work II took place at ZIB from September 21 to October 9, 2009 with 105 participants from 23 countries.

We wanted to compute the seat allocation for the lecture hall.  
To do this we required every participant to state their preferences.  
Everyone should send an email with a data file.  
Let's see how long it took...

ASCII text with only a LF (ASCII 10) as line separator.

Fields are separated by a single space (ASCII 32)

Line 1: **ParticipantNo** **HasLaptop** **EmailAddress**

e.g.     **67 1 koch@zib.de**

0 = has no Laptop, 1 = has a Laptop

Lines 2-???: **SeatNumber** **PreferenceValue**

- Seat numbers start down at the low entrance, left to right, row by row.
- The highest numbered seat is at the window side at the top.
- Count only seats that are physically there.
- The seat numbers in the file should be monotonically increasing.
- The preference values should be between 0 and 100.

e.g.     **12 55**  
          **13 40**  
          **14 35 ...**

Allowed values are between 0 and 100

Only seats which are not available for the participants are allowed to get a value of 0

All numbers 1-100 have to be used at least once

The average has to be between 40-60

The difference to an adjacent seat has to be  $< 40$

The difference to a neighboring seat has to be  $< 20$

The data should not be randomly generated

---

Lines ???-???: **ParticipantNo PreferenceOffset**

List indicating persons which you would like or not like to be your seat neighbor.  
(You have to know the ParticipantNo of the person.)

- A ParticipantNo of 0 indicates an empty seat.
- The PreferenceOffset is between -20 and 20 and will be added to your PreferenceValue if the person with the given ParticipantNo is your neighbor.  
e.g. **55 17**  
**27 -5**  
**72 8**  
**0 -10 ...**
- This list can have as many entries as you like, but there should be at least 2 entries, and the occurring participant numbers have to be unique and valid.

Submission of this file is required for the course

The name of the file has to be *ParticipantNo . txt*

It should be **attached** to an email

Send the email to [koch@zib.de](mailto:koch@zib.de)

The subject of the email should be

*CO@Work: SeatData for ParticipantNo*

*Please, as soon as possible.*

---

Mails received : 13

Different Subjects : 4 (10 1 1 1)

Wrong field spacing : 4

Seat counts : 2 (12 1)

Missing data : 1

Too much data : 1

Ok, from first view : 5 out of 13

---

Mails received : 23

Different Subjects : 6 (17 2 1 1 1 1)

Wrong field spacing : 4

Seat counts : 4 (19 1 1)

Missing data : 2

Too much data : 0

Ok, from first view : 10

Corrected : 1

[Add to the specification:](#)

A seat without a desk is not allowed for the participants

Seats with a 0 preference value are not relevant for the adjacency/neighboring difference rules.

---

Mails received : 37

Wrong subject : 11

Wrong field spacing : 8

Strange seat counts : 5

Missing data : 2

Corrected : 3

---



Mails received	: 47
Data sets	: 41 (6 corrections)
Wrong subject	: 12
Wrong attachment name	: 2
Wrong line separator	: 29
Wrong field separator	: 10
Pref. value not used	: 11
Other Errors	: 1
Number of seats	: 153 - 181
No complains so far	: 4

---

Mails received	: 79
Data sets	: 64
Wrong subject	: 16
Wrong attachment name	: 2
Wrong line separator	: 45
Wrong field separator	: 11
Pref.value not used	: 22
Other Errors	: 2
Number of seats	: 153 - 181
No complains so far	: 8

Mails received	: 104
Data sets	: 76
Wrong subject	: 18
Wrong attachment name	: 2
Pref. value not used	: 19
Neighbor difference	: 21
Wrong no/seq. seats:	: 10
Wrong 0 seats	: 20
No complains so far	: 10

---

	E7	E10	E11	E12	E13	E14	E16
5							X
6							X
12					X		X
13							X
16							X
18					X		X
19						X	X
20						X	
23					X		
24						X	
26							X
27						X	
36						X	
42					X		
45			X	X	X	X	X
47					X		
53					X		
59						X	
63			X		X	X	
64			X		X	X	X
71					X	X	

E7 bad seatno

E10 bad offset

E11 wrong seatno

E12 bad average

E13 prefval missing

E14 neighbour diff

E16 seat not 0

	E7	E10	E11	E12	E13	E14	E16
77							X
78	X		X			X	X
81				X	X		X
98					X		
99	X		X			X	
103					X	X	
107			X			X	X
108			X			X	X
111							X
121							X
128			X			X	X
129		X					
134			X	X	X	X	
135					X		
137		X			X	X	X
139					X		X
145	X		X		X	X	
160						X	
166					X	X	

**Please correct and resubmit**

Mails received : 144

Wrong subject : ~23

Wrong attachment name : 4

Data sets : 92

To be corrected : 28

Missing : 6

Pref. value not used : 14

Neighbor difference : 18

Wrong no/seq. seats : 2

---

	E7	E10	E11	E12	E13	E14
12					X	
18					X	
23	X		X			X
24						X
27						X
45					X	X
47					X	
63			X		X	X
71					X	X
78	X		X			X
79		X	X	X	X	
103						X
107			X			X
108			X			X
110		X				
114						X
118					X	X
128			X			X
134			X	X	X	X
135					X	
136						X
137		X			X	X
138					X	
139					X	
160						X
166					X	X

E7 bad seatno

E10 bad offset

E11 wrong seatno

E12 bad average

E13 prefval missing

E14 neighbour diff

**Please correct and resubmit**

Mails received : 159

Wrong subject : ~26

Wrong attachment name : 4

Data sets : 94

To be corrected : 18

Missing : 4

Preference value not used : 9

Neighbor difference : 14

Wrong no/sequence seats : 3

---

	E7	E10	E11	E12	E13	E14
18					X	
24						X
27						X
45					X	X
63					X	
71					X	X
78	X		X			X
79		X	X	X	X	
103						X
107			X			X
108			X			X
114						X
118					X	X
128			X			X
134			X	X	X	X
136						X
137		X			X	X
138					X	

E7 bad seatno

E10 bad offset

E11 wrong seatno

E12 bad average

E13 prefval missing

E14 neighbour diff

**Please correct and resubmit**



Mails received : 166  
Wrong subject : ~28  
Wrong attachment name : 4

Data sets : 95  
To be corrected : 18  
Missing : 3

Preference value not used : 7  
Neighbor difference : 14  
Wrong no/sequence seats : 3

---

	E7	E10	E11	E12	E13	E14
24						X
27						X
45					X	X
71					X	X
78	X		X			X
79		X	X	X	X	
92					X	X
107			X			X
108			X			X
114						X
118					X	X
128			X			X
134			X	X	X	X
136						X
137		X			X	X

E7 bad seatno

E10 bad offset

E11 wrong seatno

E12 bad average

E13 prefval missing

E14 neighbour diff

**Please correct and resubmit**

Mails received : 172  
Wrong subject : ~31  
Wrong attachment name : 4

Data sets : 95  
To be corrected : 13

Preference value not used : 5  
Neighbor difference : 13  
Wrong no/sequence seats : 2

The subject of the email should be  
**CO@Work: SeatData for *ParticipantNo***

CO@Work: SeatData for 022  
CO@Work:SeatData for 222  
CO@Work:SeatDatafor222  
CO@work: SeatData for 222  
CO@Work: Seat Data for 222  
Co@Work: SeatData for 222  
CO@Work: SeatData for Participant222  
CO@Work: SeatData for ParticipantNo  
Co@Work: SeatData for Participan222  
CO@WORK: seatdata for 222  
COatWork: SeatData for 222  
COatWork for 222  
SeatData for 222  
SeatData for ParticipantNo 222  
set data for participant number 222  
data set participant number 222  
Sitting assignment  
Seats assignment

---

	E7	E10	E11	E12	E13	E14
24						X
27						X
45					X	X
71					X	X
78	X		X			X
92					X	X
107			X			X
108			X			X
114						X
128			X			X
134			X	X	X	X
136						X
137		X			X	X

E7 bad seatno

E10 bad offset

E11 wrong seatno

E12 bad average

E13 prefval missing

E14 neighbour diff

**Sorry,  
too late to correct!**

Wrong line 1: 81, 129

You would think a ...

- ▶ ... cellular network operator knows where its base stations are located?
- ▶ ... fixed network operator can tell where the parts of its network are connected?
- ▶ ... chemical company knows how many plants they have?
- ▶ ... 5 m long pipeline cannot have a height difference from end-to-end of 100 m?
  
- ▶ Many companies have their data in Excel.  
There is no formal validation or referential integrity check.
- ▶ If they did formal validation, usually they found there was information they needed which they could not input and they started to “reuse” some data fields.
- ▶ If there is not at least 1 error per 100 data sets you are not looking hard enough.
- ▶ Usually the data changes all the time.
- ▶ They might not want to give it to you.
- ▶ The data might just not exist.

**The first result of an optimization project is usually to improve the quality of planning data available at the company.**

---

File: ex1.dat with 100001235 lines

File: ex1.dat with 100001235 lines

File: ../../ex1.dat with 100001235 lines.

File: ex1.dat with 99999947 lines

File: ex1.dat with 100001235 lines

File ex1.dat with 239341319 lines

('File: ', '../../exercise\_1/ex1.dat', 'with ', 100001235, ' lines (containing data)')

File: ../../exercise\_1/ex1.dat with 100001235 lines

('File:', '../../exercise\_1/ex1.dat', 'with', 100001235, 'lines')

File: ex1.dat with 100001235 lines

File: ex1.dat with 100001236 lines

File: ../../exercise\_1/ex1.dat with 100001233 lines

File ex1.dat with 100001235 lines

File: ../../../../exercise\_1/ex1.dat with 100001236 lines

File: ex1.dat with 100001235 lines

Valid values Loc1: 49994581 with GeoMean: 36.782583  
Valid values Loc1: 49994581 with GeoMean: 36.782583  
Valid values Loc1: 50004466 with GeoMean: 36.781736117270  
Valid values Loc1: 50004682 with GeoMean: 36.781681  
Valid values Loc1: 50004706 with GeoMean: 36.781671  
Valid values Loc1: 50004332 with GeoMean: 36.782305  
Valid values Loc1: 50004555 with GeoMean: 36.7817  
Valid Values Loc1: 50004616 with gemetric mean: ', 1.0)  
Valid values Loc1: 50004851 with GeoMean: 0.000000  
Valid values Loc1: 50004616 with GeoMean: 36.781761336223234  
Valid values Loc1: 49504741 with GeoMean: [37.6172](#)  
Valid Values Loc1: 50004616 with GeoMean: 36.78176133614294  
Valid values Loc1: 50005278 with GeoMean: 36.7817  
Valid values Loc1: 50004571 with GeoMean: 50.00307664753011  
Valid values Loc1: 50004620 with GeoMean: 36.78175072922403  
Valid values Loc1: 50004798 with GeoMean: 36.7817  
Valid values Loc1: 49994892 with GeoMean: 36.7825  
Valid values Loc1: 50004777 with GeoMean: 36.7817



Valid values Loc2: 49994581 with GeoMean: 36.782583200332

Valid values Loc2: 49994783 with GeoMean: 36.782547

Valid values Loc2: 49994820 with GeoMean: 36.782505

Valid values Loc2: 49994439 with Geomean: 36.783359

Valid values Loc2: 49994670 with GeoMean: 36.7826

Valid Values Loc2: 49994716 with gemetric mean: ', 1.0)

Valid values Loc2: 49994951 with GeoMean: 0.000000

Valid values Loc2: 49994716 with GeoMean: 36.782573922686403)

Valid values Loc2: 49494352 with GeoMean: 37.6204

Valid Values Loc2: 49994716 with GeoMean: 36.782573922484126

Valid values Loc2: 49995371 with GeoMean: 36.7825

Valid values Loc2: 49994669 with GeoMean: 50.00238276443099

Valid values Loc2: 49994723 with GeoMean: 36.78255535635074

Valid values Loc2: 49994865 with GeoMean: 36.7825

Baumann	C++	75
wyczisk	Py3	173
Dinse1	Py	
Fleischer	C	30
Gühring	Py2	203
Hark	Py2	224
Jeney	Py	
Julia	C++	124
Jung	C++	
Lewandowski	C++	68
Luetzke	Py2	124
Morgenroth	Java	50
Nuernberger	C	21
ongini	C++	116
RayChew	C++	
Rettkowski	C++	55
Richter	Java	
Sanny	C++	53
Schmidt	C	41
Seliverstov	Awk	384
Sterling	C	
Koch:	C	17 / 15 / 9

We provided some documentation and an example file.

Write a program **ex2** that reads in the

`measured-1.0.0.2017-02-03.b0050c5c8deb1db59c7b2644414b079d.xml`

And writes CSV data in format:

`YYYY-MM-DD; HH; amountOfPower-Value`

You will need the reading part later again.

Try to validate the XML file against the provided schema

The filename to be converted should be taken from the command line.

The output should be to stdout, any errors to stderr.

You may use whatever library to parse the XML file.