[OGIS] ®
OatGrain InnovationS

2021-05-19

Advanced Developers Conference ++
Development for Professionals!

R“--(The Compound-Group „LOOP“)--“_@de

Vereinfachte Iterationen für C/C++

Frank Haferkorn

Senior Software-Entwickler, Physiker, Erfinder

mailto:info@OatGrain-InnovationS.de

https://github.com/F-Haferkorn/ogis-modern-cxx-future-cpp

[OGIS] ®
OatGrain
InnovationS

Advanced Developers Conference C++
Development for Professionals!

Der AUTHOR

- Dipl.-Phys.
  Frank Haferkorn **(*1968)**
  Senior Software-Entwickler
- **Firma**
  [OGIS] OatGrain-InnovationS
  Ottobrunn, bei München
- Modern C++17, C++20 , HPC, Algorithmik, STL
  Physik (QM, Elektrodynamik, Theoretische & Computer Physik)
  Erfinder

[OGIS] ®
OatGrain
InnovationS

Advanced Developers Conference
Development for Professionals!

Inhaltsverzeichnis

Die Compound-Gruppe "LOOP"

- **Einführung**
- Syntax
- Implementation
- Beispiele
- Diskussion: Fakten / Pro / Contra


[OGIS] ® OATGRAIN INNOVATIONS

Advanced Developers Conference C++ Development for Professionals!

Einführung

Was sind Compounds in C/C++?

- **Kontroll-Fluss Befehlen** wie
  - **If**() {} **if**(){} **else**{} **switch**(){}
  - **for**(;;) {} **while**(){} **do**{} **while**();
  - **try** {} **catch**(){}

- Alle beinhalten ein anschließendes **„Statement“** in Form von
  - **expression-statement** **sin(x)/x;**
  - oder **compound-statement** **{** x*=M_PI; sin(x)/x;**}**
  - oder weiteren **Kontroll-Fluss Befehlen** // siehe oben

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Einführung

Die for(){} Iteration

- (reguläre) **for-loop** **for(int i=0; i<N;, ++i) {}** -
  mit **explitiziter** Initialisierung, expliziter Abbruch-Bedingung und epliziten, optionalen Post-Expressions.
- range-based for loop **for (auto element : container) {}**
  ist eine Iteration über einen Container (meist aus der STL)

```
for(int i=0; i<4; ++i)          // iteriere über 4 Zeilen
    for(int j=0; j<10; ++j)     // iteriere über 10 Spalten
        *tgt++ =  *src++ ;      // kopiere Inhalt von source zu target
```

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Einführung

Was ist : Die Compound-Gruppe "LOOP" ?

- ist eine „**Spracherweiterung**“ für die Programmiersprache **C++**, mit Einschränkungen auch für C
- In Form von Einfachen *Iterationen*
  - basierend auf dem regulären C*ompound* ***for(;;){}***
  - Und ist bereits implementiert mit dem *CPP-Preprozessor*
- NICHT gedacht als Konkurrenz für **for-range-loops** wie

  for(int &element : container) do_something(element);

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Spoiling:

Die Neue Compound-Gruppe „LOOP“

- **loop**(repetitions [, expression …])statement
- **typed_loop**(type, repetitions [, expression..]) statement
- **named_loop_up(**id, repetitions, [, expression...]**)**statement
- **named_loop_down**(id, repetitions, [, expression]) statement

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Einführung

Die Idee

Um *simple Formen von Iterationen* zu ermöglichen.

Reduziert das Neues **loop(){}** Compound

den *FREIHEITSGRAD* der Iterationen des **for(;;){}** Iteration-Compounds

–

```
loop(4)                          // iteriere über 4 Zeilen
    loop(10)                     // iteriere über 10 Spalten
        *tgt++ =  *src++ ;       // kopiere  Inhalt von source zu target
```

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Motivation

Vorteile der Compound-Block „LOOP“

- **READABILity**
- **TEACHABLILity**
- **Reduzierter Freiheitsgrad** für die Iteration
- **Performance**
  - auf einigen Architekturen (wie DSP) Hardware Beschleunigung
  - Reduzierung von „**cache misses**“ durch geringere Codegröße
- **ALGORITHMics**

[OGIS] ®
OatGrain
InnovationS

Advanced Developers Conference
Development for Professionals!

Inhaltsverzeichnis

Die Compound-Gruppe ”LOOP“

- Einführung
- **Syntax**
- Implementation
- Beispiele
- Diskussion: Fakten / Pro / Contra

[OGIS] ®
OatGrain
InnovationS

Advanced Developers Conference C++
Development for Professionals!

Syntax

loop(<rep> [, <postexpr>]...**){}**

<rep> **wiederhole** den compound-body **{}** **<rep>-mal**

<postexpr> **optionale** Liste **von “post-expressions“** (Komma separiert)

Der **versteckte index** ist vom selben Typ wie **<rep>**.

```
loop(4) loop(10, tgt++, src++) { *tgt = *src; }
```


[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Syntax

typed_loop(<type>, <rep> [, <postexpr>]...**){}**

<type> **Typ der (versteckten) Index-Variable**
<rep> wiederhole die Iteration <rep>-mals.
<postexpr> optionale Komma separierte Liste von „post-expressions“.

```
typed_loop(char, 40) *tgt++ = *src++;
```


[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference
Development for Professionals!

Syntax

named_loop_up(<id>**,** <rep> [, <postexpr>]...**){}**
named_loop_down(<id>**,** <rep> [, <postexpr>]...**){}**

<id> **bekannter Symbolname der Index Variable.**
<rep> wiederhole den Body {} <rep>-mal.
<postexpr> optionale Komma separierte Liste von „post-expressions“.

```
named_loop_up(index, noRepetitions)  value+=func(index);
```


[OGIS] ®
OatGrain
InnovationS

Advanced Developers Conference
Development for Professionals!

Inhaltsverzeichnis

- Einführung
- Syntax
- **Implementation**
- Beispiele
- Diskussion: Fakten / Pro / Contra

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference
Development for Professionals!

Implementation

```
/// CPP Makro
//      generiert eine eindeutige SymbolId

#ifndef CPPMACRO_UNIQUE_ID
#define CPPMACRO_UNIQUE_ID() \
        CPPMACRO_UNIQUE_ID_LINE_##__LINE__##_##__COUNTER__
#endif
```

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference ++
Development for Professionals!

Implementation CPPMACRO_NTIMES_UP()

```
// CPP Basis-Makro
//      iteriert   von 0 bis zu nbrOfRepetitions-1    aufwärts.

#define CPPMACRO_NTIMES_UP( indexType,                  \
                            indexName,                  \
                            nbrOfRepetitions,           \
                            ... )                       \
        for(    indexType  indexName = 0          ;     \
                IndexName < nbrOfRepetitions      ;     \
                indexName++                             \
                ,   ##__VA_ARGS__)
```

Implementation CPPMACRO_NTIMES_DOWN()

```
// CPP Makro
//       iteriert   von nbrOfRepetitions-1 bis 0    abwärts.

#define CPPMACRO_NTIMES_DOWN(indexType,        \
                             indexVarName,     \
                             nbrOfRepetitions, \
                             ... )             \
        for(    indexType  indexName = nbrOfRepetitions  ;   \
                indexVarName--  >0          ;               \
                __VA_ARGS__)
```

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference
Development for Professionals!

Implementation CPPMACRO_DECLTYPE()

```
// CPP Makro
//       index-variable verwendet den selben Typ wie nbrOfRepetitions


#if defined __cplusplus
#include <type_traits>
#define CPPMACRO_DECLTYPE(varname)                                    \
    typename std::remove_const<decltype(varname)>::type
#else
// as "C" cannot detect the type of a varname.
// This can lead  to problems with  signed/unsigned comparison
#define CPP_DECLTYPE(varname)          int
#endif
```

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Implementation loop(){}

```
// CPP-MAKRO: loop(){}
//  Iteriere nachfolgendes  block-statement
// <nbrOfRepetitions> mal.

#define loop(nbrOfRepetitions, ... )                    \
        CPPMACRO_NTIMES_UP(                             \
                CPPMACRO_DECLTYPE(nbrOfRepetitions) ,   \
                CPPMACRO_UNIQUE_ID() ,                  \
                nbrOfRepetitions        ,               \
                ##__VA_ARGS__   )
```

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Implementation typed_loop(){}

```
// typed_loop(){} :
//      Verwende type für versteckte Index Variable und iteriere
//      nachfolgenden block-statement  <nbrOfRepetitions> mal.

#define typed_loop(indexType, nbrOfRepetitions, ...  ) \
            CPPMACRO_NTIMES_UP(                        \
                    indexType                  ,       \
                    CPPMACRO_UNIQUE_ID()       ,       \
                    nbrOfRepetitions           ,       \
                    ##__VA_ARGS__        )
```

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference
Development for Professionals!

Implementation named_loop_up(){}

```
// CPP-MACRO: named_loop_up(){}
//       iteriere aufwärts  mit einer benannten
//       (, nicht versteckten )  Index-Variable.

#define named_loop_up(  indexVarName, nbrOfRepetitions, ...)  \
          CPPMACRO_NTIMES_UP(                                  \
                CPPMACRO_DECLTYPE(nbrOfRepetitions),   \
                indexVarName    ,                              \
                nbrOfRepetitions ,                             \
                ##__VA_ARGS__)
```

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference
Development for Professionals!

Implementation named_loop_down(){}

```
// CPP-MACRO: named_loop_down(){}
//       iteriere abwärts  mit einer benannten
//       (, nicht versteckte)  Index-Variable.

#define named_loop_down(indexVarName, nbrOfRepetitions, ...) \
          CPPMACRO_NTIMES_DOWN(                              \
                CPPMACRO_DECLTYPE(nbrOfRepetitions),         \
                indexVarName    ,                            \
                nbrOfRepetitions ,                           \
                ##__VA_ARGS__)
```


[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference ++
Development for Professionals!

Inhaltsverzeichnis

- Einführung
- Syntax
- Implementation
- **Beispiele**
- Diskussion: Fakten / Pro / Contra

[OGIS] ®
OatGrain
InnovationS

Advanced Developers Conference
Development for Professionals!

Beispiel 1

Reguläres **for(;;){}** compound

```
#include “ascii_print.h”
// Drucke ein Quadrat (aus Sternen)
// mit regulärem   for(;;){}

void square(short nRows, short nColumns)
{
    for(short row=0; row<nRows; row++)
    {
        for(short col=0; col<nColumns; col++)
        {
            star();
        }
        newline();
    }
}
```

```
// @file: ascii_print.h

#include <iostream>
void star()    { std::cout.put('*'); }
void space()   { std::cout.put(' '); }
void newline() { std::cout.put('\n'); }
```

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference ++
Development for Professionals!

Beispiel 1b
Reguläres **for(;;){}** compound

```
#include “ascii_print.h”
// Drucke ein Quadrat (aus Sternen)
// mit regulärem   for(;;){}

void square(short nRows, short nColumns)

{
    for(short row=0; row<nRows; row++)
    {
        for(short col=0;col<nColumns; col++)
        {
            star();
        }
        newline();
    }
}
```

Beispiel 1c loop(){}
Reduzierung for(;;){} zu while(){}

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

```
#include “ascii_print.h”
// Drucke ein Quadrat (aus Sternen)
// mit regulärem   for(;;){}

void square(short nRows, short nColumns)
{
    for(short row=0; row<nRows; row++)
    {
        for(short col=0;col<nColumns; col++)
        {
            star();
        }
        newline();
    }
}
```

```
#include “ascii_print.h”
// Drucke  Quadrat  reduziert auf while()


void square(short nRows, short nColumns)
{
    while(nRows--)
    {
        while(nColumns--)
        {
            star();
        }
        newline();
    }
}
```

Beispiel 1d loop(){}

[OGIS] ® OATGRAIN INNOVATIONS

Advanced Developers Conference C++ Development for Professionals!

```
#include “ascii_print.h”
// Drucke ein Quadrat (aus Sternen)
// mit regulärem   for(;;){}

void square(short nRows, short nColumns)
{
    for(short row=0; row<nRows; row++)
    {
        for(short col=0;col<nColumns; col++)
        {
            star();
        }
        newline();
    }
}
```

```
// print square using loop(){}
#include “ascii_print.h”
#include <loop>

void square(short  nRows, short nColumns)
{
    loop(nRows,  newline())
        loop(nColumns,  star())
            ;
}
```

[OGIS] ®
OatGrain
InnovationS

Advanced Developers Conference C++
Development for Professionals!

Inhaltsverzeichnis

[OGIS] ® OatGrain InnovationS

Advanced Developers Conference C++ Development for Professionals!

Beispiel 2

named_loop_up(){}

named_loop_down(){}

```
*
**
***
****
*****
******

******
*****
****
***
**
*
```

```
#include “ascii_print.h”
#include <loop>
void triangular_upwards(short nRows) {
   named_loop_up(row, nRows, newline())
        loop(row + 1,  star() )
             ;
}
void triangular_downwards(short nRows){
   named_loop_down(row, nRows, newline())
        loop(row + 1,  star() )
             ;
}
main(){
     triangular_upwards(6);      newline();
     triangular_downwards(6);
}
```

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Inhaltsverzeichnis

Beispiel 3 typed_loop(){}

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference
Development for Professionals!

```
// prints out
**********
**********
**********
**********
**********
**********
**********
**********
**********
**********
```

```
///  erzwinge versteckte index variabkle
// / vom Typ byte („unsigned char“)

#include ”ascii_print.h”

#include <loop>
#include <cstdint>
main()
{
      typed_loop(char,     10,      newline() )
                typed_loop(char,      20)
                     star();
}
```

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Inhaltsverzeichnis

Beispiel 4
matrix_copy_with_stride()

```
#include <loop>
template<typename TPtr,
                typename TRowSize,     typename TColSize,       typename TStrideSize>
void matrix_copy_with_stride(     TPtr  tgt, TPtr src,
                                  TRowSize nRows,
                                  TColSize nColumns,
                                  TStrideSize stride)
{
    loop(nRows,  tgt+=stride, src+=stride)      // Addiere Stride-Offsets nach jeder Zeile.
        loop(nColumns, tgt++, src++)          // Inkrementiere Adressen nach jedem Zugriff.
              *tgt = *src ;                   //  Kopiere Quelle-Daten  zum Ziel.
     return ;
}
```


[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Inhaltsverzeichnis

Beispiel 5
FooTest_StressTestCall1MillionTimes()

[OGIS] ®
OatGrain
innovationS

Advanced Developers Conference
Development for Professionals!

```
#include <chrono>
#include <thread>
#include <loop>

class Foo{
    test(){/*...*/}
};
TEST_F(FooTest, StressTestCall1MillionTimes) {
    Foo foo;
    loop(1000000) {
        EXPECT_TRUE(foo.test());
        std::this_thread::sleep_for(std::chrono::microseconds(10));
    }
}
```

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Inhaltsverzeichnis

- Einführung
- Syntax
- Implementation
- Beispiele
- **Diskussion: Fakten / Vorteile / Nachteile**

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Diskussion: Fakten

*Die LOOP- **Compound Iterationen***

- Sind nur eine kleine Erweiterungen
- *Vergleich mit* ***for(;;){}***
  - *Reduzierter Freiheitsgrad*
  - *Bessere Maintenance*
  - *Genauso Performant*
  - *Auf Speziellen CPU-Architekturen mgl. Performance-Gewinn*

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Diskussion: Fakten

Die LOOP- ***Compound Iterationen***

- sind bereits fertig implementiert
  - basierend alleinig auf dem ***CPP-Preprozessor***
  - Durch „Mapping“ zu dem **for(;;){}** Compound.
- mit einzelnem Compilation „Caveat“
- z.B.: mit der Kombination von Kommas bei Templates.

[OGIS] ®
OatGrain
InnovationS

Advanced Developers Conference C++
Development for Professionals!

Diskussion: Lesbarbeit

- Lesbarkeit / READABILITY:
  - Die C++ Source Code Größe wird verringert
  - die Code-Lesbarkeit wird deutlich verbessert

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Diskussion: Teachability

- **ALGORITHMen**:
  - Ermöglicht es Einfacheren Code zu schreiben
  - UNABHÄNGIGKEIT vom Iterations Index
  - Reduzierter Freiheitsgrad der Iteration

[OGIS] ® OatGrain InnovationS

Advanced Developers Conference C++ Development for Professionals!

Diskussion: Teachability

- **TEACHABILITY (aktuelle Methodik):**
  - Im Moment wird beim Erlernen der Programmiersprache C/C++ das **for(;;){}** compound in einer der ersten Lektionen unterrichtet.
  - Ein einfaches **for(int i=0; i<n; ++i){}** erfordert die Prinzipien
    - Das **Prinzip Typ,**
    - das **Prinzip Variable**, und das **Prinzip Zuweisung,**
    - das **Inkrementieren um 1**
    - die Verwendung des **Prinzips Bedingung** in Form v. **Boolschen Ausdrücken** inklusive **Vergleichen (<, ==, >, >=, <=)**

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Diskussion: Teachability

- **TEACHABILITY: !!! Die Raspberry-PI Generation !!!**
- **loop(){}**
  - Verbessert die Art C++ zu Unterrichten
    speziell für jüngeres Publikum
  - so hat sich die Britische-Regierung entschieden verpflichtend Kinder ab dem Alter von 4-Jahren Programmieren zu lernen.
  - 5. Klässler **(11-12 Jahre alte) Schüler können das Konzept der Iteration/Looping verstehen.**

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Diskussion: Teachability

- **TEACHABILITY: !!! sehr einfache Iterationen !!!**
  Loop(){} , named_loop_up(), named_loop_down()
- Um ein Rechtecke mit fester Kantenlänge auszugeben, benötigt man
  - ✔ außer dem Konzept von Zahlen,
  - ✔ und dem Konzept von Kommandos
  - ✔ keine weiteren Programmierkonzepten,
    - Für weiteren Schwierigkeitsstufen kann man Konzepte wie Variablen und Inkrement hinzunehmen.

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Diskussion: Freiheitsgrad

- **Reduzierter FREIHEITSGRAD**:
  - Offensichtlich
  - Reduzieren die ***Compounds der* LOOP Group** die Freiheitsgrade der **for(;;){}** Iteration and erlauben es Code in einfacherer Weise zu strukturieren.
  - Es kann verwendet werden um
  - **einfacheren / sicheren / besser wartbaren Code**
  - zu schreiben.

[OGIS] ® OATGRAIN INNOVATIONS

Advanced Developers Conference C++ Development for Professionals!

Diskussion: Performance

- **LOOP öffnet die Tür zu weiteren OPTIMIERUNGEN**
  - Loop(){} **mindestens genauso performant** wie eine *reguläre* ***for(;;){}*** *Schleife*,
  - Bietet aber eine höhere Flexibilität für den Compiler Zählrichtung steht nicht fest.
  -
  - ✔ int N=10000, M=2000;
    loop(N)*loop(M) count++;
  - ✔ loop(N*M) count++;

[OGIS] ® OATGRAIN INNOVATIONS

Advanced Developers Conference C++ Development for Professionals!

Diskussion: Performance

- **Geringere Code Size:**
  - ✔ Verringert Auftreten von INSTRUCTION-CACHE-MISSES
    - https://en.wikipedia.org/wiki/CPU_cache

      Cache read misses from an **instruction cache** generally **cause the largest delay**,
      because the processor, or at least the thread of execution, has to wait (stall) until the instruction is fetched from main memory.

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference
Development for Professionals!

Diskussion: Performance

- **OPTIMIEREN via Hardware Beschleunigug**
- **Einige (speziell DSP) Architekturen erlauben**
  - ➔ ***Hardware "accelerated" Loops :***
    - ➔ e.g.: DSP TMS320: *Software Pipelined Loop: (*"**SPLOOP")**
  - ➔ **Fast Register Post-Operations :**
    - ➔ e.g.: DSP ADSP218x "Data Address Generators" (DAG1/2)

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference ++
Development for Professionals!

Diskussion: Enums

- Wie erwartet
  - **Compiliert Looping** über **enum-typen** NICHT.
    - enum {RED, GREEN, BLUE} rgb=BLUE;
      loop(rgb) /// !peng compiler error.
      do_something();

[OGIS] ® OATGRAIN INNOVATIONS

Advanced Developers Conference C++ Development for Professionals!

Diskussion: Nachteile

Compilierungs-Fallstrick

- Die (derzeitige) Preprozessor Implementation bekommt **Compilierungsprobleme** bei **Argumenten die Kommas enthalten** (wie einige **templates**).
-

```
loop(std::integral_constant<int, 10>::value)    /// compiler error
                                    do_something();

/// workaround: umklammern with “()”
 loop( (std::integral_constant<int, 10>::value) )   /// works:
                                    do_something();
```

Live Demo

Godbolt-Presentation

[godbolt.org-CompoundGroup-LOOP-DEMO]

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Zusammenfassung

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Zusammenfassung

Neue Compound-Befehle

- **loop**(rep [, expression …])statement

- **typed_loop**(type, rep [, expression..]) statement
- **named_loop_up(**id, rep, [, expression...]**)**statement
- **named_loop_down**(id, rep, [, expression]) statement

[OGIS] ®
OatGrain
InnovationS

Advanced Developers Conference ++
Development for Professionals!

Fragen?

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference C++
Development for Professionals!

Antworten

Zum LOOP Compound Block

Sie finden

- Details
- Vollständige Implementierung
- Code Beispiele

unter

https://github.com/F-Haferkorn/ogis-modern-cxx-future-cpp

[OGIS] ®
OatGrain
InnovationS

Advanced Developers Conference
Development for Professionals!

Vorteile

Compound-Block „LOOP“

- READABILity
- ALGORITHMics
- TEACHABLILity
- Reduzierter Freiheitsgrad für Iterationen
- Performance
  - Auf DSP Architekturen
  - d. kleinere Code Größe Reduzierung von Cachemisses)

Antwort:

Was ist mit break/continue?

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference
Development for Professionals!

```
//Verhalten wie in jedem Iterations-Compound
loop(N){
    break;          //          unterbricht  die Iteration
    continue;       //     setzt die Iteration fort.
}
```

Konkurrenz <range> Library? Teachability & Performance Vorteil

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference
Development for Professionals!

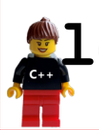
```
int N=1000, M=200;

//   hat hohe Flexibilität, erkauft durch overhead.
for (int n : std::view::iota{0, N} )
   for (int m : std::view::iota{0, M} )
      do_something();


/// Einfacher Code,  Zero-Overhead mit mögl. Speedup
loop(N) loop(M)   do_something();
```

Mögliche Optimierung

```
loop(N*M) do_something();
```



Frank Haferkorn

[OGIS] ®
OatGrain
InnovationS

Advanced Developers Conference C++
Development for Professionals!

Vielen Dank!

Ich freue mich auf Feedback!

info@oatgrain-innovations.de

OGIS OatGrain-InnovationS
Kontakt-Data & Einige Links

Dipl.-Phys. Frank Haferkorn
[OGIS] - OatGrain-InnovationS
D-85521 Ottobrunn // bei München
E-Mail : OGIS.eu@gmail.com
www : www.oatgrain-innovations.de
Handy: : +49/176/70311275
Skype : live:F.Haferkorn
GITHUB: : https://github.com/F-Haferkorn
Twitter : https://twitter.com/FrankHaferkorn
LinkedIn : https://de.linkedin.com/in/frank-haferkorn-48ba568
Xing : https://www.xing.com/profile/Frank_Haferkorn

Dipl.-Phys. Frank.Haferkorn
Über dieses Dokument

[OGIS] ®
OATGRAIN
INNOVATIONS

Advanced Developers Conference
Development for Professionals!

https://adcpp.de/21/sessions

In seinem deutschsprachigem Vortrag stellt Frank Haferkorn die Compound-Gruppe "LOOP" als eine C/C ++ Core-Language Extension vor. Die ursprünglichen C Kontrollfluss-Befehle haben sich mindestens seit 1978(!), seit der Veröffentlichung von Kernighan&Ritchies „K&R C“ wenig bis gar nicht verändert, Sie heißen „Compound(s)“ und sind in die wohlbekannten if-else, while, do-while, for und switch. Nur ein Compound in Form des try/catch Blocks hat sich in C++ hinzugesellt. Seit C++ 17 gibt es ein kleinere weitere Entwicklungen.
Ist es ein physikalisches Gesetz, dass niemals weiteren Compounds hinzukommen dürfen?
Die hier vorgestellten neuen Compounds loop(){}, typed_loop(){}, named_loop_up(){} und named_loop_down(){} ermöglichen eine simple Codierung einfacher Iterationen. Auch wenn dies kein neues Major-Feature werden wird, habe diese Vorteile. Sie reduzieren die Komplexität, verbessern die Lesbarkeit von C/C ++ und werden zu anderen/einfacheren Notationen (auch bestehender) Algorithmen führen. Compiler können aufgrund reduzierter Komplexität performanteren Code erzeugen. Eine Verbesserung der Teachability von C/C++ ist zu erwarten und damit sinkt die Einstiegsschwelle in C für zukünftige C/C++ Entwickler aus der heutigen Raspberry Generation.
Frank Haferkorn zeigt die Syntax, erklärt die grundlegende Verwendung erläutert die Anwendung Anhand von Beispielen, diskutiert die Vor- und Nachteile und präsentiert zuerst eine reine C-Implementierung alleinig basierend auf dem C-Preprocessor mittels Variadischen Makros. Für eine elaboriertere C++ Implementierung sind noch einige weitere C++ Kniffe nötig…
Auch bekannte Probleme mit der derzeitigen Implementierung dürfen nicht fehlen. Die LOOP Compounds sind implementiert als einzelne header-only include Datei.

Dipl.-Phys.
Frank.Haferkorn
Eine Kurze Biographie

[OGIS] ®
OatGrain
InnovationS

Advanced Developers Conference ++
Development for Professionals!

R”---(
Frank.Haferkorn ist 53a, Diplom-Physiker, Senior-Software Entwickler und Gründer bzw. Head-Of-Science des Erfinderbüros [OGIS] OatGrain-InnovationS. Er gehört zu der Generation die noch die ganze Entwicklung der Tischrechner (ab dem CBM-PET) miterlebt hat und arbeitete seit seinem Abschluss and der TU-München 1995 bis heute als professioneller Software-Entwickler in der Industrie. Seine Spezialgebiete sind unter anderem Modern C++ (ab C+ +17). Er macht sich neben kleineren Publikationen „Eigene Gedanken" u.a. über die Weiterentwicklung von C++.
Die Verwendung von elaborierten Werkzeugen aller Art ist sein Steckenpferd und deckt das ganze Spektrum von Visual Studio über Qt und Linux ab. Als Ausgleich für die IT kann man Frank als Künstler beim Zeichnen, beim Komponieren und als Sound-Designer von räumlichem Audio antreffen.
mailto:info@OatGrain-InnovationS.de
)--”_@en