C++ vs. C

the embedded perspective

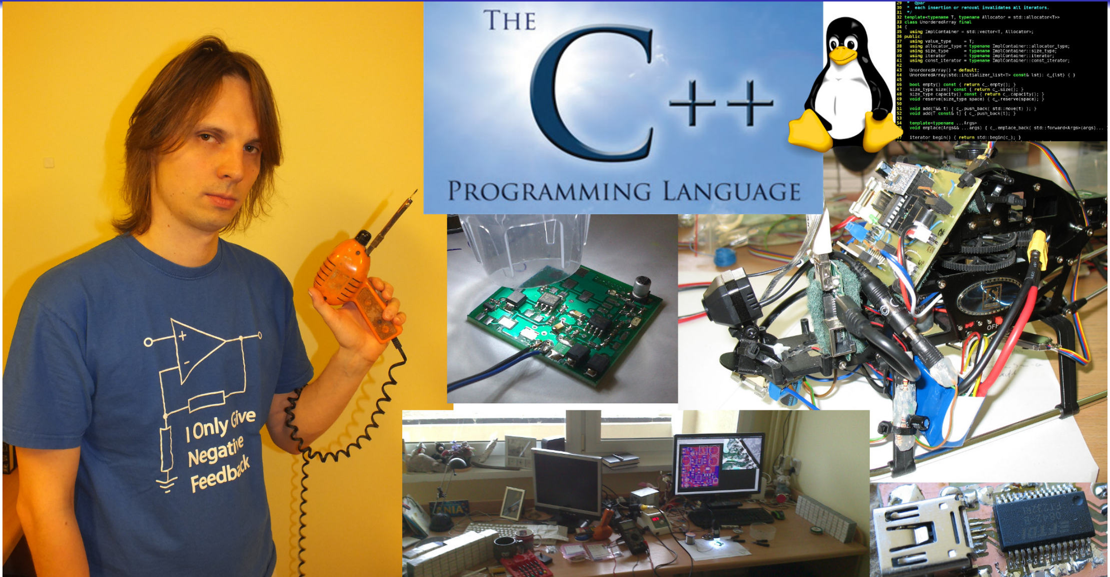
Bartek 'BaSz' Szurgot

http://www.baszerr.eu

2015-11-05

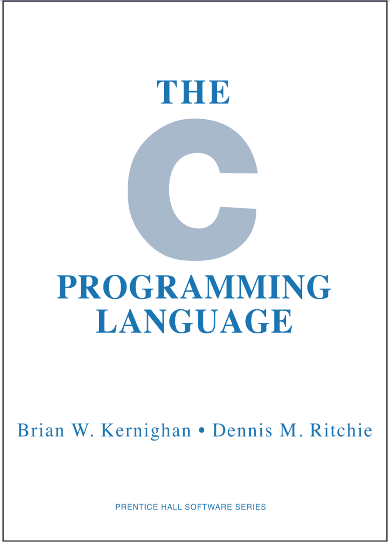
Code for food

Professional geek



```
 * @par
 * each insertion or removal invalidates all iterators.
 */
template<typename T, typename Allocator = std::allocator<T>>
class UnorderedArray final
{
  using ImplContainer = std::vector<T, Allocator>;
public:
  using value_type     = T;
  using allocator_type = typename ImplContainer::allocator_type;
  using size_type      = typename ImplContainer::size_type;
  using iterator       = typename ImplContainer::iterator;
  using const_iterator = typename ImplContainer::const_iterator;

  UnorderedArray() = default;
  UnorderedArray(std::initializer_list<T> const& lst): c_{lst} { }

  bool empty() const { return c_.empty(); }
  size_type size() const { return c_.size(); }
  size_type capacity() const { return c_.capacity(); }
  void reserve(size_type space) { c_.reserve(space); }

  void add(T&& t) { c_.push_back( std::move(t) ); }
  void add(T const& t) { c_.push_back(t); }

  template<typename ...Args>
  void emplace(Args&& ...args) { c_.emplace_back( std::forward<Args>(args)...

  iterator begin() { return std::begin(c_); }
```

Historically. . .



THE C PROGRAMMING LANGUAGE

Brian W. Kernighan • Dennis M. Ritchie

PRENTICE HALL SOFTWARE SERIES

https://upload.wikimedia.org/wikipedia/commons/9/95/The_C_Programming_Language%2C_First_Edition_Cover_%282%29.svg

Part 2

1. The experiment
2. Proving ground
3. Baseline
4. Flow control
5. Generic programming
6. Conclusions

Rules

- Year: 2005
- Software: GCC-3.x
- Hardware: x86

Rules

- Year: 2005
- Software: GCC-3.x
- Hardware: x86
- Bet:
  - *C is faster than C++*
  - *No it's not*

Rules

- Year: 2005
- Software: GCC-3.x
- Hardware: x86
- Bet:
  - *C is faster than C++*
  - *No it’s not*
- Task:
  - Iterating array
  - Summing elements
  - Displaying result

Experimental code in C

```c
#include <stdint.h>

int main(void)
{
   const unsigned size = 2*1000*1000*1000;
   uint8_t* tab = (uint8_t*)malloc(size*sizeof(uint8_t));
   for(unsigned i=0; i<size; ++i)
       tab[i] = 3;
   unsigned out = 42;
   for(unsigned i=0; i<size; ++i)
       out += tab[i];
   free(tab);
   return out % 256;
}
```

Experimental code in C

```c
#include <stdint.h>

int main(void)
{
    const unsigned size = 2*1000*1000*1000;
    uint8_t* tab = (uint8_t*)malloc(size*sizeof(uint8_t));
    for(unsigned i=0; i<size; ++i)
        tab[i] = 3;
    unsigned out = 42;
    for(unsigned i=0; i<size; ++i)
        out += tab[i];
    free(tab);
    return out % 256;
}
```

Experimental code in C

```c
#include <stdint.h>

int main(void)
{
  const unsigned size = 2*1000*1000*1000;
  uint8_t* tab = (uint8_t*)malloc(size*sizeof(uint8_t));
  for(unsigned i=0; i<size; ++i)
      tab[i] = 3;
  unsigned out = 42;
  for(unsigned i=0; i<size; ++i)
      out += tab[i];
  free(tab);
  return out % 256;
}
```

Experimental code in C

```
#include <stdint.h>

int main(void)
{
  const unsigned size = 2*1000*1000*1000;
  uint8_t* tab = (uint8_t*)malloc(size*sizeof(uint8_t));
  for(unsigned i=0; i<size; ++i)
      tab[i] = 3;
  unsigned out = 42;
  for(unsigned i=0; i<size; ++i)
      out += tab[i];
  free(tab);
  return out % 256;
}
```

Experimental code in C

```c
#include <stdint.h>

int main(void)
{
  const unsigned size = 2*1000*1000*1000;
  uint8_t* tab = (uint8_t*)malloc(size*sizeof(uint8_t));
  for(unsigned i=0; i<size; ++i)
      tab[i] = 3;
  unsigned out = 42;
  for(unsigned i=0; i<size; ++i)
      out += tab[i];
  free(tab);
  return out % 256;
}
```

Experimental code in C++

```cpp
#include <algorithm>
#include <cstdint>
#include <boost/scoped_array.hpp>

int main(void)
{
  const unsigned size = 2*1000*1000*1000;
  boost::scoped_array<uint8_t> tab(new uint8_t[size]);
  std::fill(tab.get(), tab.get()+size, 3);
  unsigned out = std::accumulate(tab.get(), tab.get()+size, 42u);
  return out % 256;
}
```

Experimental code in C++

```
#include <algorithm>
#include <cstdint>
#include <boost/scoped_array.hpp>

int main(void)
{
  const unsigned size = 2*1000*1000*1000;
  boost::scoped_array<uint8_t> tab(new uint8_t[size]);
  std::fill(tab.get(), tab.get()+size, 3);
  unsigned out = std::accumulate(tab.get(), tab.get()+size, 42u);
  return out % 256;
}
```

Experimental code in C++

```
#include <algorithm>
#include <cstdint>
#include <boost/scoped_array.hpp>

int main(void)
{
  const unsigned size = 2*1000*1000*1000;
  boost::scoped_array<uint8_t> tab(new uint8_t[size]);
  std::fill(tab.get(), tab.get()+size, 3);
  unsigned out = std::accumulate(tab.get(), tab.get()+size, 42u);
  return out % 256;
}
```

Experimental code in C++

```
#include <algorithm>
#include <cstdint>
#include <boost/scoped_array.hpp>

int main(void)
{
  const unsigned size = 2*1000*1000*1000;
  boost::scoped_array<uint8_t> tab(new uint8_t[size]);
  std::fill(tab.get(), tab.get()+size, 3);
  unsigned out = std::accumulate(tab.get(), tab.get()+size, 42u);
  return out % 256;
}
```

Experimental code in C++

```cpp
#include <algorithm>
#include <cstdint>
#include <boost/scoped_array.hpp>

int main(void)
{
  const unsigned size = 2*1000*1000*1000;
  boost::scoped_array<uint8_t> tab(new uint8_t[size]);
  std::fill(tab.get(), tab.get()+size, 3);
  unsigned out = std::accumulate(tab.get(), tab.get()+size, 42u);
  return out % 256;
}
```

Outcome?

Outcome?

C++ over

3% **faster**

Surprise!

http://dailypicksandflicks.com/wp-content/uploads/2012/09/Day-33-they-still-suspect-nothing-Bobtail-Sheepdog-and-sheep-.jpg

What happened?

```
const unsigned size = 2*1000*1000*1000;
boost::scoped_array<uint8_t> tab(new uint8_t[size]);
std::fill(tab.get(), tab.get()+size, 3);
unsigned out = std::accumulate(tab.get(), tab.get()+size, 42u);
```

- Abstract approach:

What happened?

```
const unsigned size = 2*1000*1000*1000;
boost::scoped_array<uint8_t> tab(new uint8_t[size]);
std::fill(tab.get(), tab.get()+size, 3);
unsigned out = std::accumulate(tab.get(), tab.get()+size, 42u);
```

- Abstract approach:

What happened?

```
const unsigned size = 2*1000*1000*1000;
boost::scoped_array<uint8_t> tab(new uint8_t[size]);
std::fill(tab.get(), tab.get()+size, 3);
unsigned out = std::accumulate(tab.get(), tab.get()+size, 42u);
```

- Abstract approach:

What happened?

```
const unsigned size = 2*1000*1000*1000;
boost::scoped_array<uint8_t> tab(new uint8_t[size]);
std::fill(tab.get(), tab.get()+size, 3);
unsigned out = std::accumulate(tab.get(), tab.get()+size, 42u);
```

- Abstract approach:
  - STL
  - Opportunities!

What happened?

```
const unsigned size = 2*1000*1000*1000;
boost::scoped_array<uint8_t> tab(new uint8_t[size]);
std::fill(tab.get(), tab.get()+size, 3);
unsigned out = std::accumulate(tab.get(), tab.get()+size, 42u);
```

- Abstract approach:
  - STL
  - Opportunities!
- std::vector<uint8_t> too!
- Clang and GCC!
- Measurements, measurements. . .

Part 3

1. The experiment
2. Proving ground
3. Baseline
4. Flow control
5. Generic programming
6. Conclusions

Compilers’ settings

- GCC with custom flags

Compilers' settings

- GCC with custom flags
- Common flags:
  - `-DNDEBUG`
  - `-s`
  - `-ffunction-sections`
  - `-fdata-sections`
  - `-Wl,-gc-sections`
  - `-flto` (!!)

Compilers' settings

- GCC with custom flags
- Common flags:
  - `-DNDEBUG`
  - `-s`
  - `-ffunction-sections`
  - `-fdata-sections`
  - `-Wl,-gc-sections`
  - `-flto` (!!)
- Standards:
  - `-std=c++11`
  - `-std=gnu99`

Compilers' settings

- GCC with custom flags
- Common flags:
  - `-DNDEBUG`
  - `-s`
  - `-ffunction-sections`
  - `-fdata-sections`
  - `-Wl,-gc-sections`
  - `-flto` (!!)
- Standards:
  - `-std=c++11`
  - `-std=gnu99`
- Flags for µCs:
  - `-fno-rtti`
  - `-fno-exceptions`

(a local wariant of) `https://upload.wikimedia.org/wikipedia/commons/d/dd/Buzz_salutes_the_U.S._Flag.jpg`

Optimization flags

- Optimizing for speed:
  - `-O3`
  - `-finiline-limit=150`

Optimization flags

- Optimizing for speed:
  - `-O3`
  - `-finiline-limit=150`
- Optimizing for size:
  - `-Os`

http://www.cashmanequipment.com/blog/wp-content/uploads/2012/02/Checkered-Flag4.jpg

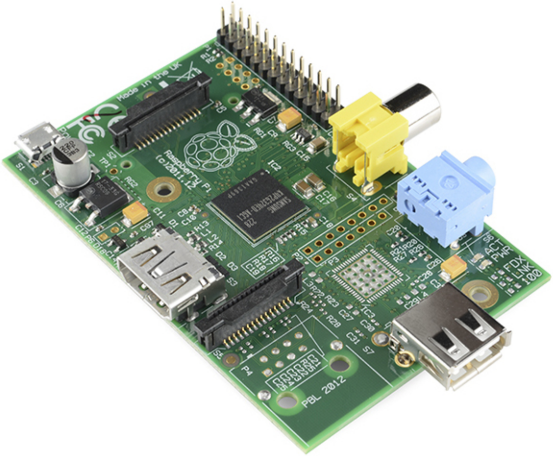
AMD64 (x86_64)

- "Modern PC"
- AMD FX8350
  - 64-bit
  - 4GHz
  - 32GB RAM
- Linux
- GCC-5.2.1
- Extra flags:
  - `-march=native`

http://www.hartware.de/media/reviews/1544/intro_hi.jpg

ARM-11

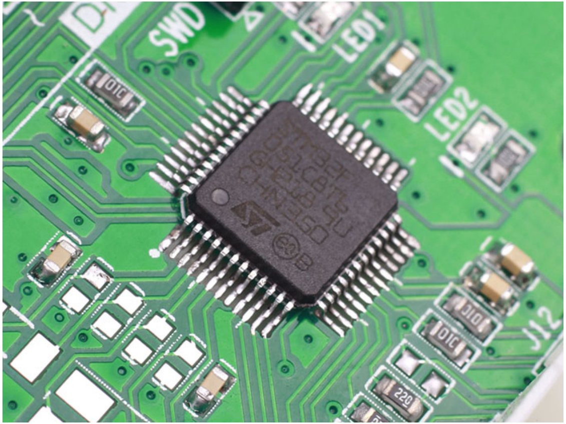
- Raspberry Pi
- ARM1176JZF-S (Broadcom)
  - 32-bit
  - 700MHz
  - 1GB RAM
- Linux
- GCC-5.2.1
- Extra flags:
  - `-march=native`



https://upload.wikimedia.org/wikipedia/commons/4/45/Raspberry_Pi_-_Model_A.jpg

ARM-M0

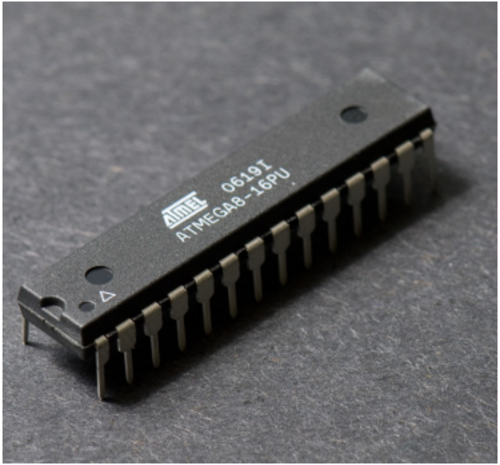
- "Mid-class" µC
- STM32F051 – Cortex M0 (STM)
  - 32-bit
  - 48MHz
  - 64kB flash
  - 8kB RAM
- No operating system
- GCC-4.9.3
- Extra flags:
  - `-mthumb`
  - `-mcpu=cortex-m0`



http://www.seeedstudio.com/depot/images/1029901720_02.jpg

AVR

- "Small" µC
- ATmega8 (Atmel)
  - 8-bit
  - 16MHz
  - 8kB flash
  - 1kB RAM
- No operating system
- GCC-4.8.1
- Extra flags:
  - `-mmcu=atmega8`



https://upload.wikimedia.org/wikipedia/commons/a/a9/ATmega8_01_Pengo.jpg

Runtime measurments

```
int main()
{
  // [...] - preparing test data
  measureStart();
  // [...] - actual code being measured
  measurePrint();
  measureSink(/* computed value */);
}
```

Runtime measurments

```
int main()
{
  // [...] - preparing test data
  measureStart();
  // [...] — actual code being measured
  measurePrint();
  measureSink(/* computed value */);
}
```

Runtime measurments

```
int main()
{
  // [...] — preparing test data
  measureStart();
  // [...] — actual code being measured
  measurePrint();
  measureSink(/* computed value */);
}
```

Runtime measurments

```
int main()
{
  // [...] — preparing test data
  measureStart();
  // [...] - actual code being measured
  measurePrint();
  measureSink(/* computed value */);
}
```

Runtime measurments

```
int main()
{
  // [...] – preparing test data
  measureStart();
  // [...] – actual code being measured
  measurePrint();
  measureSink(/* computed value */);
}
```

Runtime measurments

```
int main()
{
  // [...] – preparing test data
  measureStart();
  // [...] – actual code being measured
  measurePrint();
  measureSink(/* computed value */);
}
```

Runtime measurments

```
int main()
{
  // [...] - preparing test data
  measureStart();
  // [...] - actual code being measured
  measurePrint();
  measureSink(/* computed value */);
}
```

- Time overhead?
  - Few CPU cycles
  - Negligible. . .

Runtime measurments

```
int main()
{
  // [...] - preparing test data
  measureStart();
  // [...] - actual code being measured
  measurePrint();
  measureSink(/* computed value */);
}
```

- Time overhead?
  - Few CPU cycles
  - Negligible. . .

- Size overhead?

Platform	Size [B]	LOC
AMD64	536	50
ARM-11	400	50
ARM-M0	1184	300
AVR	324	100

Part 4

1. The experiment
2. Proving ground
3. **Baseline**
4. Flow control
5. Generic programming
6. Conclusions

Let’s start simple!

- C:

```c
int main()
{
}
```

Let’s start simple!

- C:

```c
int main()
{
}
```

- C++:

```cpp
int main()
{
}
```

AMD64 - attempt 1

- Sizes:
  - 4352[B] – C++
  - 4112[B] – C
  - i.e. 240[B] overhead

AMD64 - attempt 1

- Sizes:
  - 4352[B] – C++
  - 4112[B] – C
  - i.e. 240[B] overhead
- `ldd`:
  - C++:
    - linux-vdso.so.1 . . .
    - **libstdc++.so.6** . . .
    - **libm.so.6** . . .
    - **libgcc_s.so.1** . . .
    - libc.so.6 . . .
    - /lib64/ld-linux-x86-64.so.2 . . .
  - C:
    - linux-vdso.so.1 . . .
    - libc.so.6 . . .
    - /lib64/ld-linux-x86-64.so.2 . . .

AMD64 - attempt 1

- Sizes:
  - 4352[B] – C++
  - 4112[B] – C
  - i.e. 240[B] overhead
- `ldd`:
  - C++:
    - linux-vdso.so.1 . . .
    - **libstdc++.so.6** . . .
    - **libm.so.6** . . .
    - **libgcc_s.so.1** . . .
    - libc.so.6 . . .
    - /lib64/ld-linux-x86-64.so.2 . . .
  - C:
    - linux-vdso.so.1 . . .
    - libc.so.6 . . .
    - /lib64/ld-linux-x86-64.so.2 . . .

✗

AMD64 - attempt 2, Mk I

- Extra flags:
  - `-static`

AMD64 - attempt 2, Mk I

- Extra flags:
    - `-static`
- Sizes:
    - 717856[B] – C++
    - 717856[B] – C

AMD64 - attempt 2, Mk I

- Extra flags:
  - `-static`
- Sizes:
  - 717856[B] – C++
  - 717856[B] – C
- `md5sum`:
  - 72572f4e0006d21516e4377faa50b752 – C++
  - 72572f4e0006d21516e4377faa50b752 – C

AMD64 - attempt 2, Mk I

- Extra flags:
  - `-static`
- Sizes:
  - 717856[B] – C++
  - 717856[B] – C
- `md5sum`:
  - 72572f4e0006d21516e4377faa50b752 – C++
  - 72572f4e0006d21516e4377faa50b752 – C

✓

AMD64 - attempt 2, Mk II

- Alternatively:
  - Disable extra linking
  - Just link basic libs
  - A bit more fuss. . . ;)

AMD64 - attempt 2, Mk II

- Alternatively:
  - Disable extra linking
  - Just link basic libs
  - A bit more fuss. . . ;)
- Sizes:
  - 4112[B] – C++
  - 4112[B] – C

AMD64 - attempt 2, Mk II

- Alternatively:
  - Disable extra linking
  - Just link basic libs
  - A bit more fuss. . . ;)
- Sizes:
  - 4112[B] – C++
  - 4112[B] – C
- `md5sum`:
  - fbb1c3d0acfa62f72f6bdce02a764e4a – C++
  - fbb1c3d0acfa62f72f6bdce02a764e4a – C

AMD64 - attempt 2, Mk II

- Alternatively:
  - Disable extra linking
  - Just link basic libs
  - A bit more fuss. . . ;)
- Sizes:
  - 4112[B] – C++
  - 4112[B] – C
- `md5sum`:
  - fbb1c3d0acfa62f72f6bdce02a764e4a – C++
  - fbb1c3d0acfa62f72f6bdce02a764e4a – C

✓

Getting "more embedded"

- ARM-11 – static:
  - 476272[B] – C++
  - 476272[B] – C
  - MD5-identical

Getting "more embedded"

- ARM-11 – static:
  - 476272[B] – C++
  - 476272[B] – C
  - MD5-identical
- ARM-11 – no standard libs:
  - 2780[B] – C++
  - 2780[B] – C

Getting "more embedded"

- ARM-11 – static:
  - 476272[B] – C++
  - 476272[B] – C
  - MD5-identical
- ARM-11 – no standard libs:
  - 2780[B] – C++
  - 2780[B] – C
- ARM-M0:
  - 592[B] – C++ (!!)
  - 596[B] – C

Getting "more embedded"

- ARM-11 – static:
  - 476272[B] – C++
  - 476272[B] – C
  - MD5-identical
- ARM-11 – no standard libs:
  - 2780[B] – C++
  - 2780[B] – C
- ARM-M0:
  - 592[B] – C++ (!!)
  - 596[B] – C
- AVR:
  - 128[B] – C++
  - 128[B] – C
  - MD5-identical

Getting "more embedded"

- ARM-11 – static:
  - 476272[B] – C++
  - 476272[B] – C
  - MD5-identical
- ARM-11 – no standard libs:
  - 2780[B] – C++
  - 2780[B] – C
- ARM-M0:
  - 592[B] – C++ (!!)
  - 596[B] – C
- AVR:
  - 128[B] – C++
  - 128[B] – C
  - MD5-identical

✓

Bottom line

$$C \subset C{+}{+}$$

Bottom line

$$C \subset C{+}{+}$$

- Well – almost. . .
- Key takeaways:
  - C++ – much bigger
  - Almost C-compatible
  - Very few differences
  - A lot of "C-legacy"

Bottom line

$$C \subset C{+}{+}$$

- Well – almost. . .
- Key takeaways:
  - C++ – much bigger
  - Almost C-compatible
  - Very few differences
  - A lot of "C-legacy"
- "Over-linking" on PC. . . ;)
- No differences for µCs

C++ motto

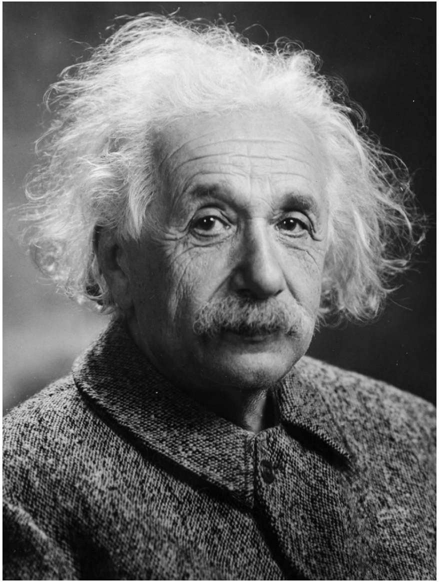
You do not pay for what you do not use.

Part 5

1. The experiment
2. Proving ground
3. Baseline
4. Flow control
5. Generic programming
6. Conclusions

And now. . .

if()

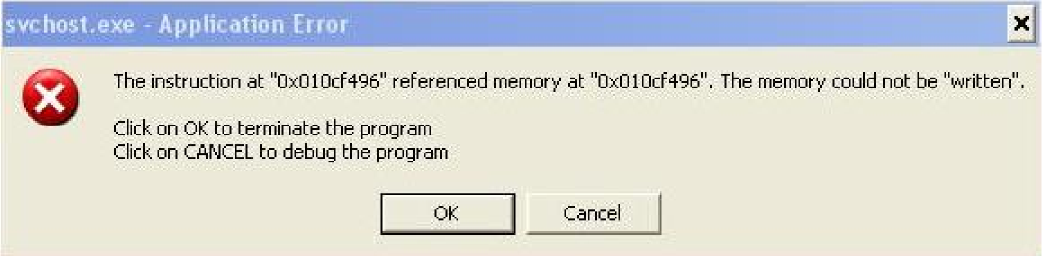


`https://upload.wikimedia.org/wikipedia/commons/d/d3/Albert_Einstein_Head.jpg`

http://cdn.androidpolice.com/wp-content/uploads/2014/12/nexus2cee_50190-Have-you-tried-turning-it-off-HfqE_thumb.jpg

Errare humanum est. . .

Errare humanum est. . .



svchost.exe - Application Error

The instruction at "0x010cf496" referenced memory at "0x010cf496". The memory could not be "written".

Click on OK to terminate the program
Click on CANCEL to debug the program

http://1.bp.blogspot.com/-HzwyoFKILFQ/TZ0LlhQqcdI/AAAAAAAAAFE/qR201ehh2xg/s1600/svchost.exe+error.JPG

Error checking example – main.c(pp)

```
Data d;
d.param_    = magic;
d.hello_[0] = 'f';
d.hello_[1] = 'o';
d.hello_[2] = 'o';
Data* ptr = usePtr ? &d : 0;
int   out = magic;

for(int i=0; i<1000; ++i)
{
  if( magic != 1 )
    out = func1(ptr, out);
  if( magic != 2 )
    out = func2(ptr, out);
  if( magic != 3 )
    out = func3(ptr, out);
}
```

Error checking example – main.c(pp)

```
Data d;
d.param_     = magic;
d.hello_[0] = ’f’;
d.hello_[1] = ’o’;
d.hello_[2] = ’o’;
Data∗ ptr = usePtr ? &d : 0;
int   out = magic;

for(int i=0; i<1000; ++i)
{
  if( magic != 1 )
    out = func1(ptr, out);
  if( magic != 2 )
    out = func2(ptr, out);
  if( magic != 3 )
    out = func3(ptr, out);
}
```

- Data – parameter
- Random initialization
- Runtime arguments:
  - `magic`

Error checking example – main.c(pp)

```
Data d;
d.param_    = magic;
d.hello_[0] = ’f’;
d.hello_[1] = ’o’;
d.hello_[2] = ’o’;
Data* ptr = usePtr ? &d : 0;
int   out = magic;

for(int i=0; i<1000; ++i)
{
  if( magic != 1 )
    out = func1(ptr, out);
  if( magic != 2 )
    out = func2(ptr, out);
  if( magic != 3 )
    out = func3(ptr, out);
}
```

- Data – parameter
- Random initialization
- Runtime arguments:
  - magic
  - usePtr

Error checking example – main.c(pp)

```
Data d;
d.param_     = magic;
d.hello_[0] = ’f’;
d.hello_[1] = ’o’;
d.hello_[2] = ’o’;
Data* ptr = usePtr ? &d : 0;
int   out = magic;

for(int i=0; i<1000; ++i)
{
  if( magic != 1 )
    out = func1(ptr, out);
  if( magic != 2 )
    out = func2(ptr, out);
  if( magic != 3 )
    out = func3(ptr, out);
}
```

- Data – parameter
- Random initialization
- Runtime arguments:
  - `magic`
  - `usePtr`
- Watch out for optimizations! ;)
  - LTO
  - Constant statements
  - Identical functions
  - ...

Error checking example – main.c(pp)

```
Data d;
d.param_     = magic;
d.hello_[0] = ’f’;
d.hello_[1] = ’o’;
d.hello_[2] = ’o’;
Data* ptr = usePtr ? &d : 0;
int   out = magic;

for(int i=0; i<1000; ++i)
{
  if( magic != 1 )
    out = func1(ptr, out);
  if( magic != 2 )
    out = func2(ptr, out);
  if( magic != 3 )
    out = func3(ptr, out);
}
```

- Data – parameter
- Random initialization
- Runtime arguments:
  - `magic`
  - `usePtr`
- Watch out for optimizations! ;)
  - LTO
  - Constant statements
  - Identical functions
  - ...
- C-compatible code:
  - Common implementation
  - Minimal differences

Error checking – simple case

- Functions' implementations:

```
int func1(Data* d, int in)
{
  if(!d)
    abort();
  if( d->param_ % 10 )
    d->param_ += 13;
  if( in > d->param_ )
    in -= 4;
  return (d->param_*in+1)
         % 1235;
}
```

```
int func2(Data* d, int in)
{
  if(!d)
    abort();
  d->param_ = (d->param_+12)
               % 75;
  return (d->param_+in/3*41)
         % 1206;
}
```

```
int func3(Data* d, int in)
{
  if(!d)
    abort();
  if(in>1000)
    in %= 999;
  for(int i=0; i<in; ++i)
  {
    d->param_ += in/2 - i;
    ++in;
    in *= 2;
  }
  return in + d->param_;
}
```

Error checking – simple case

- Functions’ implementations:

```
int func1(Data* d, int in)
{
  if(!d)
    abort();
  if( d->param_ % 10 )
    d->param_ += 13;
  if( in > d->param_ )
    in -= 4;
  return (d->param_*in+1)
         % 1235;
}
```

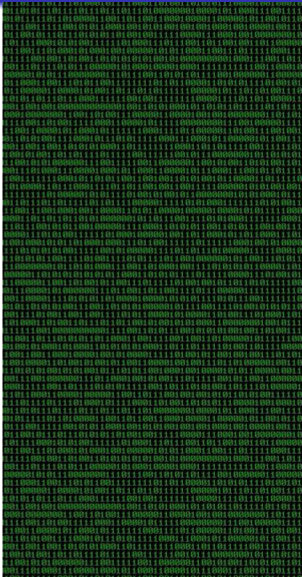
```
int func2(Data* d, int in)
{
  if(!d)
    abort();
  d->param_ = (d->param_+12)
              % 75;
  return (d->param_+in/3*41)
         % 1206;
}
```

```
int func3(Data* d, int in)
{
  if(!d)
    abort();
  if(in>1000)
    in %= 999;
  for(int i=0; i<in; ++i)
  {
    d->param_ += in/2 - i;
    ++in;
    in *= 2;
  }
  return in + d->param_;
}
```

- Simplified view for measuring
- Error means `abort()`

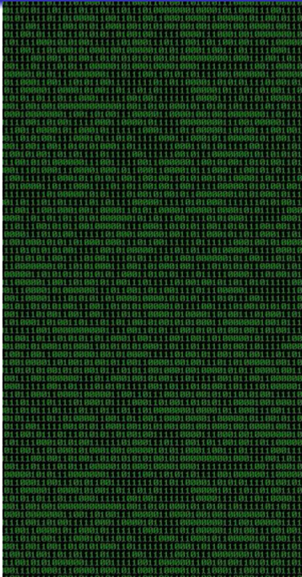
Code size results

Platform	C [B]	C++ [B]
AMD64/nostdlib	5072	5080



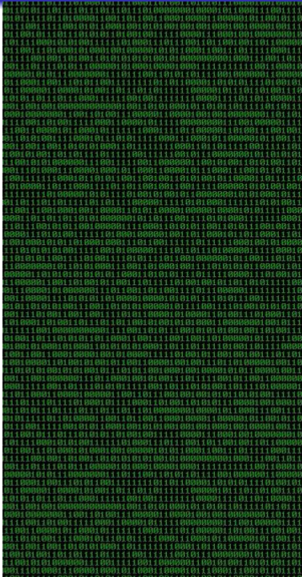
Code size results

Platform	C [B]	C++ [B]
AMD64/nostdlib	5072	5080
ARM-11	4248	3996



Code size results

Platform	C [B]	C++ [B]
AMD64/nostdlib	5072	5080
ARM-11	4248	3996
ARM-M0	4268	4264



Code size results

Platform	C [B]	C++ [B]
AMD64/nostdlib	5072	5080
ARM-11	4248	3996
ARM-M0	4268	4264
AVR	958	958

http://upload.wikimedia.org/wikipedia/commons/thumb/c/cf/Binary_Code.jpg/1024px-Binary_Code.jpg

Can we do more with C++?

- So far:
  - Similar sizes
  - Similar code
  - Just another example. . .

Can we do more with C++?

- So far:
  - Similar sizes
  - Similar code
  - Just another example. . .
- Can we do more?
- With C++?
- Remember `if(!pointer)` problem?

```
int funcN(Data* d, int in)
{
  if(!d)          // hmm....
    abort();      // ...
  // bla..
  // bla..
  // bla..
  return 42;
}
```

Property type wrapper – non-nullptr pointer

```
template<typename P>
class NotNull final
{
public:
  explicit NotNull(P* p):
    p_{p}
  {
    if(not p_)
      abort();
  }

  P* operator->() const { return p_; }

private:
  P* p_;
};
```

Property type wrapper – non-nullptr pointer

```
template<typename P>
class NotNull final
{
public:
  explicit NotNull(P* p):
    p_{p}
  {
    if(not p_)
      abort();
  }

  P* operator->() const { return p_; }

private:
  P* p_;
};
```

- Helper wrapper:
  - Represents a pointer
  - Pointer is never `nullptr`
  - Arrow operator for syntax
  - Drop-in replacement

Property type wrapper – non-nullptr pointer

```cpp
template<typename P>
class NotNull final
{
public:
  explicit NotNull(P* p):
    p_{p}
  {
    if(not p_)
      abort();
  }

  P* operator->() const { return p_; }

private:
  P* p_;
};
```

- Helper wrapper:
  - Represents a pointer
  - Pointer is never `nullptr`
  - Arrow operator for syntax
  - Drop-in replacement

Property type wrapper – non-nullptr pointer

```
template<typename P>
class NotNull final
{
public:
  explicit NotNull(P* p):
    p_{p}
  {
    if(not p_)
      abort();
  }

  P* operator->() const { return p_; }

private:
  P* p_;
};
```

- Helper wrapper:
  - Represents a pointer
  - Pointer is never `nullptr`
  - Arrow operator for syntax
  - Drop-in replacement
- Pointer checked in c-tor:
  - Just once – RAII-style
  - Type-guaranteed safely
  - No need for re-checking

Property type wrapper – non-nullptr pointer

```
template<typename P>
class NotNull final
{
public:
  explicit NotNull(P* p):
    p_{p}
  {
    if(not p_)
      abort();
  }

  P* operator->() const { return p_; }

private:
  P* p_;
};
```

- Helper wrapper:
  - Represents a pointer
  - Pointer is never `nullptr`
  - Arrow operator for syntax
  - Drop-in replacement
- Pointer checked in c-tor:
  - Just once – RAII-style
  - Type-guaranteed safely
  - No need for re-checking
- Simplistic
- Enough for tests

Applying the change

```
Data d;
d.param_     = magic;
d.hello_[0] = 'f';
d.hello_[1] = 'o';
d.hello_[2] = 'o';
NotNull<Data> ptr{ usePtr ? &d : 0 };
int    out = magic;

for(int i=0; i<1000; ++i)
{
  if( magic != 1 )
    out = func1(ptr, out);
  if( magic != 2 )
    out = func2(ptr, out);
  if( magic != 3 )
    out = func3(ptr, out);
}
```

Applying the change

```
Data d;
d.param_    = magic;
d.hello_[0] = 'f';
d.hello_[1] = 'o';
d.hello_[2] = 'o';
NotNull<Data> ptr{ usePtr ? &d : 0 };
int   out = magic;

for(int i=0; i<1000; ++i)
{
  if( magic != 1 )
    out = func1(ptr, out);
  if( magic != 2 )
    out = func2(ptr, out);
  if( magic != 3 )
    out = func3(ptr, out);
}
```

```
int funcN(NotNull<Data> d, int in)
{
  // if(!d)     // not needed...
  //   abort(); // ...any more! :D
  // bla..
  // bla..
  // bla..
  return 42;
}
```

Applying the change

```
Data d;
d.param_     = magic;
d.hello_[0] = ’f’;
d.hello_[1] = ’o’;
d.hello_[2] = ’o’;
NotNull<Data> ptr{ usePtr ? &d : 0 };
int    out = magic;

for(int  i=0;  i<1000; ++i)
{
  if(  magic != 1  )
    out = func1(ptr,  out);
  if(  magic != 2  )
    out = func2(ptr,  out);
  if(  magic != 3  )
    out = func3(ptr,  out);
}
```

```
int funcN(NotNull<Data> d, int in)
{
  // if(!d)     // not needed...
  //   abort(); // ...any more! :D
  // bla..
  // bla..
  // bla..
  return 42;
}
```

- Simple refactoring
- Compiler supported
- Less code! :D

Less code. . .

http://memecrunch.com/meme/62H3/happy-cat/image.png

Results – sizes

- Old results copied for reference

Platform	C [B]	C++ [B]	C++/NotNull [B]
AMD64/nostdlib	5072	5080	5056

Results – sizes

- Old results copied for reference

Platform	C [B]	C++ [B]	C++/NotNull [B]
AMD64/nostdlib	5072	5080	5056
ARM-11	4248	3996	3964

Results – sizes

- Old results copied for reference

Platform	C [B]	C++ [B]	C++/NotNull [B]
AMD64/nostdlib	5072	5080	5056
ARM-11	4248	3996	3964
ARM-M0	4268	4264	4256

Results – sizes

- Old results copied for reference

Platform	C [B]	C++ [B]	C++/NotNull [B]
AMD64/nostdlib	5072	5080	5056
ARM-11	4248	3996	3964
ARM-M0	4268	4264	4256
AVR	958	958	934

Results – sizes

- Old results copied for reference

Platform	C [B]	C++ [B]	C++/NotNull [B]
AMD64/nostdlib	5072	5080	5056
ARM-11	4248	3996	3964
ARM-M0	4268	4264	4256
AVR	958	958	934

- New type – decreased size!
- Templates ≠ footprint. . .

Results – speed

Platform	C	C++	C++/NotNull
AMD64 [μs]	31 ± 6	29 ± 5	30 ± 7

Results – speed

Platform	C	C++	C++/NotNull
AMD64 [μs]	31 ± 6	29 ± 5	30 ± 7
ARM-11 [μs]	140 ± 30	160 ± 40	140 ± 40

Results – speed

Platform	C	C++	C++/NotNull
AMD64 [μs]	31 ± 6	29 ± 5	30 ± 7
ARM-11 [μs]	140 ± 30	160 ± 40	140 ± 40
ARM-M0 [*cycles*]	308,594	308,594	302,017

Results – speed

Platform	C	C++	C++/NotNull
AMD64 [*μs*]	31 ± 6	29 ± 5	30 ± 7
ARM-11 [*μs*]	140 ± 30	160 ± 40	140 ± 40
ARM-M0 [*cycles*]	308,594	308,594	302,017
AVR [*cycles*]	1,493,926	1,493,926	1,471,556

Results – speed

Platform	C	C++	C++/NotNull
AMD64 [*μs*]	31 ± 6	29 ± 5	30 ± 7
ARM-11 [*μs*]	140 ± 30	160 ± 40	140 ± 40
ARM-M0 [*cycles*]	308,594	308,594	302,017
AVR [*cycles*]	1,493,926	1,493,926	1,471,556

- C++ and C – equal
- `NotNull`:
  - No/small difference on Linux
  - 2% gain on µCs
- Templates ≠ footprint. . .

Faster by encapsulation

- **Faster by encapsulation** approach:
  - Faster execution time
  - Faster development
  - Faster testing

Faster by encapsulation

- **Faster by encapsulation** approach:
  - Faster execution time
  - Faster development
  - Faster testing
- Smaller binary – free lunch!

Faster by encapsulation

- **Faster by encapsulation** approach:
  - Faster execution time
  - Faster development
  - Faster testing
- Smaller binary – free lunch!
- Mixture of techniques
- General approach
- Good practice to avoid errors

Faster by encapsulation

- **Faster by encapsulation** approach:
  - Faster execution time
  - Faster development
  - Faster testing
- Smaller binary – free lunch!
- Mixture of techniques
- General approach
- Good practice to avoid errors
- Field-tested
- `NotNull<>` available as OS:
- `https://github.com/el-bart/but/blob/master/But/NotNull.hpp`

http://www.blowfieldbikers.be/wp-content/uploads/2011/06/hellyeah.jpg

Part 6

1. The experiment
2. Proving ground
3. Baseline
4. Flow control
5. Generic programming
6. Conclusions

Example task

- Computing 4 medians
- 4 input arrays:
  - 2 of `ints`
  - 2 of `doubles`
  - 10 elements each

Example task

- Computing 4 medians
- 4 input arrays:
  - 2 of `ints`
  - 2 of `doubles`
  - 10 elements each
- Approach:
  - Sort input array
  - Select middle element

Example task

- Computing 4 medians
- 4 input arrays:
  - 2 of `int`s
  - 2 of `double`s
  - 10 elements each
- Approach:
  - Sort input array
  - Select middle element
- Bubble sort

https://upload.wikimedia.org/wikipedia/commons/9/94/Reflection_in_a_soap_bubble_edit.jpg

C++ (templates)

```
template<typename T>
T arrayMedian(T* tab, int size)
{
  for(int i=0; i<size; ++i)
  {
    auto changed = false;
    for(int j=1; j<size; ++j)
    {
      if(tab[j-1] > tab[j])
      {
        swap(tab[j-1], tab[j]);
        changed = true;
      }
    }
    if(not changed)
      break;
  }
  return tab[size/2];
}
```

C++ (templates)

```
template<typename T>
T arrayMedian(T* tab, int size)
{
  for(int i=0; i<size; ++i)
  {
    auto changed = false;
    for(int j=1; j<size; ++j)
    {
      if(tab[j-1] > tab[j])
      {
        swap(tab[j-1], tab[j]);
        changed = true;
      }
    }
    if(not changed)
      break;
  }
  return tab[size/2];
}
```

C++ (templates)

```cpp
template<typename T>
T arrayMedian(T* tab, int size)
{
  for(int i=0; i<size; ++i)
  {
    auto changed = false;
    for(int j=1; j<size; ++j)
    {
      if(tab[j-1] > tab[j])
      {
        swap(tab[j-1], tab[j]);
        changed = true;
      }
    }
    if(not changed)
      break;
  }
  return tab[size/2];
}
```

C++ (templates)

```
template<typename T>
T arrayMedian(T* tab, int size)
{
  for(int i=0; i<size; ++i)
  {
    auto changed = false;
    for(int j=1; j<size; ++j)
    {
      if(tab[j-1] > tab[j])
      {
        swap(tab[j-1], tab[j]);
        changed = true;
      }
    }
    if(not changed)
      break;
  }
  return tab[size/2];
}
```

C++ (templates)

```
template<typename T>
T arrayMedian(T* tab, int size)
{
  for(int i=0; i<size; ++i)
  {
    auto changed = false;
    for(int j=1; j<size; ++j)
    {
      if(tab[j-1] > tab[j])
      {
        swap(tab[j-1], tab[j]);
        changed = true;
      }
    }
    if(not changed)
      break;
  }
  return tab[size/2];
}
```

C++ (templates)

```cpp
template<typename T>
T arrayMedian(T* tab, int size)
{
  for(int i=0; i<size; ++i)
  {
    auto changed = false;
    for(int j=1; j<size; ++j)
    {
      if(tab[j-1] > tab[j])
      {
        swap(tab[j-1], tab[j]);
        changed = true;
      }
    }
    if(not changed)
      break;
  }
  return tab[size/2];
}
```

- Swapping:

```cpp
template<typename T>
void swap(T& a, T& b)
{
  T tmp = a;
  a = b;
  b = tmp;
}
```

C++ (templates)

```
template<typename T>
T arrayMedian(T* tab, int size)
{
  for(int i=0; i<size; ++i)
  {
    auto changed = false;
    for(int j=1; j<size; ++j)
    {
      if(tab[j-1] > tab[j])
      {
        swap(tab[j-1], tab[j]);
        changed = true;
      }
    }
    if(not changed)
      break;
  }
  return tab[size/2];
}
```

- Swapping:

```
template<typename T>
void swap(T& a, T& b)
{
  T tmp = a;
  a = b;
  b = tmp;
}
```

- Automatic array size deduction:

```
template<typename T, int N>
T arrayMedian(T (&tab)[N])
{
  return arrayMedian(tab, N);
}
```

C (macros)

```
#define ARRAY_MEDIAN(tab, out, tmp) \
  do \
  { int size=sizeof(tab)/sizeof(tab[0]); \
    for(int i=0; i<size; ++i) \
    { \
      int changed = 0; \
      for(int j=1; j<size; ++j) \
      { \
        if(tab[j-1] > tab[j]) \
        { \
          tmp      = tab[j-1]; \
          tab[j-1] = tab[j]; \
          tab[j]   = tmp; \
          changed  = 1; \
        } \
      } \
      if(!changed) \
        break; \
    } \
    out = tab[size/2]; \
  } while(0)
```

C (macros)

```
#define ARRAY_MEDIAN(tab, out, tmp) \
  do \
  { int size=sizeof(tab)/sizeof(tab[0]); \
    for(int i=0; i<size; ++i) \
    { \
      int changed = 0; \
      for(int j=1; j<size; ++j) \
      { \
        if(tab[j-1] > tab[j]) \
        { \
          tmp      = tab[j-1]; \
          tab[j-1] = tab[j]; \
          tab[j]   = tmp; \
          changed  = 1; \
        } \
      } \
      if(!changed) \
        break; \
    } \
    out = tab[size/2]; \
  } while(0)
```

https://pugnaciousirishman.files.wordpress.com/2008/12/vogons22.jpg

C (macros)

```
#define ARRAY_MEDIAN(tab, out, tmp) \
  do \
  { int size=sizeof(tab)/sizeof(tab[0]); \
    for(int i=0; i<size; ++i) \
    { \
      int changed = 0; \
      for(int j=1; j<size; ++j) \
      { \
        if(tab[j-1] > tab[j]) \
        { \
          tmp      = tab[j-1]; \
          tab[j-1] = tab[j]; \
          tab[j]   = tmp; \
          changed  = 1; \
        } \
      } \
      if(!changed) \
        break; \
    } \
    out = tab[size/2]; \
  } while(0)
```

https://pugnaciousirishman.files.wordpress.com/2008/12/vogons22.jpg

C (macros)

```
#define ARRAY_MEDIAN(tab, out, tmp) \
  do \
  { int size=sizeof(tab)/sizeof(tab[0]); \
    for(int i=0; i<size; ++i) \
    { \
      int changed = 0; \
      for(int j=1; j<size; ++j) \
      { \
        if(tab[j-1] > tab[j]) \
        { \
          tmp      = tab[j-1]; \
          tab[j-1] = tab[j]; \
          tab[j]   = tmp; \
          changed  = 1; \
        } \
      } \
      if(!changed) \
        break; \
    } \
    out = tab[size/2]; \
  } while(0)
```

https://pugnaciousirishman.files.wordpress.com/2008/12/vogons22.jpg

C (macros)

```
#define ARRAY_MEDIAN(tab, out, tmp) \
  do \
  { int size=sizeof(tab)/sizeof(tab[0]); \
    for(int i=0; i<size; ++i) \
    { \
      int changed = 0; \
      for(int j=1; j<size; ++j) \
      { \
        if(tab[j-1] > tab[j]) \
        { \
          tmp      = tab[j-1]; \
          tab[j-1] = tab[j]; \
          tab[j]   = tmp; \
          changed  = 1; \
        } \
      } \
      if(!changed) \
        break; \
    } \
    out = tab[size/2]; \
  } while(0)
```

https://pugnaciousirishman.files.wordpress.com/2008/12/vogons22.jpg

C (macros)

```c
#define ARRAY_MEDIAN(tab, out, tmp) \
  do \
  { int size=sizeof(tab)/sizeof(tab[0]); \
    for(int i=0; i<size; ++i) \
    { \
      int changed = 0; \
      for(int j=1; j<size; ++j) \
      { \
        if(tab[j-1] > tab[j]) \
        { \
          tmp      = tab[j-1]; \
          tab[j-1] = tab[j]; \
          tab[j]   = tmp; \
          changed  = 1; \
        } \
      } \
      if(!changed) \
        break; \
    } \
    out = tab[size/2]; \
  } while(0)
```

https://pugnaciousirishman.files.wordpress.com/2008/12/vogons22.jpg

C (macros)

```
#define ARRAY_MEDIAN(tab, out, tmp) \
  do \
  { int size=sizeof(tab)/sizeof(tab[0]); \
    for(int i=0; i<size; ++i) \
    { \
      int changed = 0; \
      for(int j=1; j<size; ++j) \
      { \
        if(tab[j-1] > tab[j]) \
        { \
          tmp      = tab[j-1]; \
          tab[j-1] = tab[j]; \
          tab[j]   = tmp; \
          changed  = 1; \
        } \
      } \
      if(!changed) \
        break; \
    } \
    out = tab[size/2]; \
  } while(0)
```

https://pugnaciousirishman.files.wordpress.com/2008/12/vogons22.jpg

main()s

```
// C implementation:
int    n1[10] = { /*...*/ };
int    n2[10] = { /*...*/ };
double d1[10] = { /*...*/ };
double d2[10] = { /*...*/ };

int tmpInt;
double tmpDouble;

int out1;
ARRAY_MEDIAN(n1, out1, tmpInt);
int out2;
ARRAY_MEDIAN(n2, out2, tmpInt);
double out3;
ARRAY_MEDIAN(d1, out3, tmpDouble);
double out4;
ARRAY_MEDIAN(d2, out4, tmpDouble);
```

main()s

```
// C implementation:
int    n1[10] = { /*...*/ };
int    n2[10] = { /*...*/ };
double d1[10] = { /*...*/ };
double d2[10] = { /*...*/ };

int tmpInt;
double tmpDouble;

int out1;
ARRAY_MEDIAN(n1, out1, tmpInt);
int out2;
ARRAY_MEDIAN(n2, out2, tmpInt);
double out3;
ARRAY_MEDIAN(d1, out3, tmpDouble);
double out4;
ARRAY_MEDIAN(d2, out4, tmpDouble);
```

main()s

```
// C implementation:
int    n1[10] = { /*...*/ };
int    n2[10] = { /*...*/ };
double d1[10] = { /*...*/ };
double d2[10] = { /*...*/ };

int tmpInt;
double tmpDouble;

int out1;
ARRAY_MEDIAN(n1, out1, tmpInt);
int out2;
ARRAY_MEDIAN(n2, out2, tmpInt);
double out3;
ARRAY_MEDIAN(d1, out3, tmpDouble);
double out4;
ARRAY_MEDIAN(d2, out4, tmpDouble);
```

main()s

```
// C implementation:
int    n1[10] = { /*...*/ };
int    n2[10] = { /*...*/ };
double d1[10] = { /*...*/ };
double d2[10] = { /*...*/ };

int tmpInt;
double tmpDouble;

int out1;
ARRAY_MEDIAN(n1, out1, tmpInt);
int out2;
ARRAY_MEDIAN(n2, out2, tmpInt);
double out3;
ARRAY_MEDIAN(d1, out3, tmpDouble);
double out4;
ARRAY_MEDIAN(d2, out4, tmpDouble);
```

main()s

```
// C implementation:
int    n1[10] = { /*...*/ };
int    n2[10] = { /*...*/ };
double d1[10] = { /*...*/ };
double d2[10] = { /*...*/ };

int tmpInt;
double tmpDouble;

int out1;
ARRAY_MEDIAN(n1, out1, tmpInt);
int out2;
ARRAY_MEDIAN(n2, out2, tmpInt);
double out3;
ARRAY_MEDIAN(d1, out3, tmpDouble);
double out4;
ARRAY_MEDIAN(d2, out4, tmpDouble);
```

main()s

```
// C implementation:
int    n1[10] = { /*...*/ };
int    n2[10] = { /*...*/ };
double d1[10] = { /*...*/ };
double d2[10] = { /*...*/ };

int tmpInt;
double tmpDouble;

int out1;
ARRAY_MEDIAN(n1, out1, tmpInt);
int out2;
ARRAY_MEDIAN(n2, out2, tmpInt);
double out3;
ARRAY_MEDIAN(d1, out3, tmpDouble);
double out4;
ARRAY_MEDIAN(d2, out4, tmpDouble);
```

main()s

```c
// C implementation:
int    n1[10] = { /*...*/ };
int    n2[10] = { /*...*/ };
double d1[10] = { /*...*/ };
double d2[10] = { /*...*/ };

int tmpInt;
double tmpDouble;

int out1;
ARRAY_MEDIAN(n1, out1, tmpInt);
int out2;
ARRAY_MEDIAN(n2, out2, tmpInt);
double out3;
ARRAY_MEDIAN(d1, out3, tmpDouble);
double out4;
ARRAY_MEDIAN(d2, out4, tmpDouble);
```

main()s

```
// C implementation:
int    n1[10] = { /*...*/ };
int    n2[10] = { /*...*/ };
double d1[10] = { /*...*/ };
double d2[10] = { /*...*/ };

int tmpInt;
double tmpDouble;

int out1;
ARRAY_MEDIAN(n1, out1, tmpInt);
int out2;
ARRAY_MEDIAN(n2, out2, tmpInt);
double out3;
ARRAY_MEDIAN(d1, out3, tmpDouble);
double out4;
ARRAY_MEDIAN(d2, out4, tmpDouble);
```

```
// C++ implementation:
int    n1[10] = { /*...*/ };
int    n2[10] = { /*...*/ };
double d1[10] = { /*...*/ };
double d2[10] = { /*...*/ };

const auto out1 = arrayMedian(n1);
const auto out2 = arrayMedian(n2);
const auto out3 = arrayMedian(d1);
const auto out4 = arrayMedian(d2);
```

main()s

```c
// C implementation:
int    n1[10] = { /*...*/ };
int    n2[10] = { /*...*/ };
double d1[10] = { /*...*/ };
double d2[10] = { /*...*/ };

int tmpInt;
double tmpDouble;

int out1;
ARRAY_MEDIAN(n1, out1, tmpInt);
int out2;
ARRAY_MEDIAN(n2, out2, tmpInt);
double out3;
ARRAY_MEDIAN(d1, out3, tmpDouble);
double out4;
ARRAY_MEDIAN(d2, out4, tmpDouble);
```

```cpp
// C++ implementation:
int    n1[10] = { /*...*/ };
int    n2[10] = { /*...*/ };
double d1[10] = { /*...*/ };
double d2[10] = { /*...*/ };

const auto out1 = arrayMedian(n1);
const auto out2 = arrayMedian(n2);
const auto out3 = arrayMedian(d1);
const auto out4 = arrayMedian(d2);
```

main()s

```
// C implementation:
int    n1[10] = { /*...*/ };
int    n2[10] = { /*...*/ };
double d1[10] = { /*...*/ };
double d2[10] = { /*...*/ };

int tmpInt;
double tmpDouble;

int out1;
ARRAY_MEDIAN(n1, out1, tmpInt);
int out2;
ARRAY_MEDIAN(n2, out2, tmpInt);
double out3;
ARRAY_MEDIAN(d1, out3, tmpDouble);
double out4;
ARRAY_MEDIAN(d2, out4, tmpDouble);
```

```
// C++ implementation:
int    n1[10] = { /*...*/ };
int    n2[10] = { /*...*/ };
double d1[10] = { /*...*/ };
double d2[10] = { /*...*/ };

const auto out1 = arrayMedian(n1);
const auto out2 = arrayMedian(n2);
const auto out3 = arrayMedian(d1);
const auto out4 = arrayMedian(d2);
```

main()s

```c
// C implementation:
int    n1[10] = { /*...*/ };
int    n2[10] = { /*...*/ };
double d1[10] = { /*...*/ };
double d2[10] = { /*...*/ };

int tmpInt;
double tmpDouble;

int out1;
ARRAY_MEDIAN(n1, out1, tmpInt);
int out2;
ARRAY_MEDIAN(n2, out2, tmpInt);
double out3;
ARRAY_MEDIAN(d1, out3, tmpDouble);
double out4;
ARRAY_MEDIAN(d2, out4, tmpDouble);
```

```cpp
// C++ implementation:
int    n1[10] = { /*...*/ };
int    n2[10] = { /*...*/ };
double d1[10] = { /*...*/ };
double d2[10] = { /*...*/ };

const auto out1 = arrayMedian(n1);
const auto out2 = arrayMedian(n2);
const auto out3 = arrayMedian(d1);
const auto out4 = arrayMedian(d2);
```

main()s

```c
// C implementation:
int    n1[10] = { /*...*/ };
int    n2[10] = { /*...*/ };
double d1[10] = { /*...*/ };
double d2[10] = { /*...*/ };

int tmpInt;
double tmpDouble;

int out1;
ARRAY_MEDIAN(n1, out1, tmpInt);
int out2;
ARRAY_MEDIAN(n2, out2, tmpInt);
double out3;
ARRAY_MEDIAN(d1, out3, tmpDouble);
double out4;
ARRAY_MEDIAN(d2, out4, tmpDouble);
```

```cpp
// C++ implementation:
int    n1[10] = { /*...*/ };
int    n2[10] = { /*...*/ };
double d1[10] = { /*...*/ };
double d2[10] = { /*...*/ };

const auto out1 = arrayMedian(n1);
const auto out2 = arrayMedian(n2);
const auto out3 = arrayMedian(d1);
const auto out4 = arrayMedian(d2);
```

main()s

```
// C implementation:
int    n1[10] = { /*...*/ };
int    n2[10] = { /*...*/ };
double d1[10] = { /*...*/ };
double d2[10] = { /*...*/ };

int tmpInt;
double tmpDouble;

int out1;
ARRAY_MEDIAN(n1, out1, tmpInt);
int out2;
ARRAY_MEDIAN(n2, out2, tmpInt);
double out3;
ARRAY_MEDIAN(d1, out3, tmpDouble);
double out4;
ARRAY_MEDIAN(d2, out4, tmpDouble);
```

```
// C++ implementation:
int    n1[10] = { /*...*/ };
int    n2[10] = { /*...*/ };
double d1[10] = { /*...*/ };
double d2[10] = { /*...*/ };

const auto out1 = arrayMedian(n1);
const auto out2 = arrayMedian(n2);
const auto out3 = arrayMedian(d1);
const auto out4 = arrayMedian(d2);
```

Measurements – size

Platform	C [B]	C++ [B]
AMD64/nostdlib	6672	6048

Measurements – size

Platform	C [B]	C++ [B]
AMD64/nostdlib	6672	6048
ARM-11	5808	5164

Measurements – size

Platform	C [B]	C++ [B]
AMD64/nostdlib	6672	6048
ARM-11	5808	5164
ARM-M0	11404	11296

Measurements – size

Platform	C [B]	C++ [B]
AMD64/nostdlib	6672	6048
ARM-11	5808	5164
ARM-M0	11404	11296
AVR	3126	2922

Measurements – size

Platform	C [B]	C++ [B]
AMD64/nostdlib	6672	6048
ARM-11	5808	5164
ARM-M0	11404	11296
AVR	3126	2922

- C’s binaries are larger:
  - All the "embedded" platforms
  - Spectacular loss on "big" platforms

Measurements – size

Platform	C [B]	C++ [B]
AMD64/nostdlib	6672	6048
ARM-11	5808	5164
ARM-M0	11404	11296
AVR	3126	2922

- C’s binaries are larger:
  - All the "embedded" platforms
  - Spectacular loss on "big" platforms
- Surprised? :)
- . . . How about speed?

https://upload.wikimedia.org/wikipedia/commons/3/37/African_Bush_Elephant.jpg

Measurements – speed

Platform	C	C++
AMD64 [μs]	2.3 ± 0.5	2.3 ± 0.7

Measurements – speed

Platform	C	C++
AMD64 [μs]	2.3 ± 0.5	2.3 ± 0.7
ARM-11 [μs]	16 ± 10	14 ± 12

Measurements – speed

Platform	C	C++
AMD64 [μs]	2.3 ± 0.5	2.3 ± 0.7
ARM-11 [μs]	16 ± 10	14 ± 12
ARM-M0 [*cycles*]	11123	9868

Measurements – speed

Platform	C	C++
AMD64 [μs]	2.3 ± 0.5	2.3 ± 0.7
ARM-11 [μs]	16 ± 10	14 ± 12
ARM-M0 [*cycles*]	11123	9868
AVR [*cycles*]	14198	12640

Measurements – speed

Platform	C	C++
AMD64 [μs]	2.3 ± 0.5	2.3 ± 0.7
ARM-11 [μs]	16 ± 10	14 ± 12
ARM-M0 [*cycles*]	11123	9868
AVR [*cycles*]	14198	12640

- C’s binaries are slower:
  - Non-measurable on "bigger" CPUs
  - All the "embedded" platforms
  - Almost 13% gain on µCs

Measurements – speed

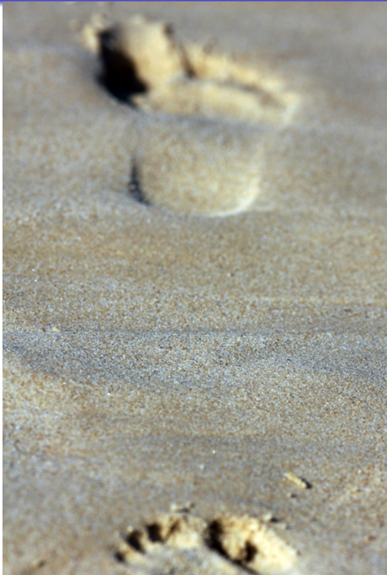
Platform	C	C++
AMD64 [μs]	2.3 ± 0.5	2.3 ± 0.7
ARM-11 [μs]	16 ± 10	14 ± 12
ARM-M0 [*cycles*]	11123	9868
AVR [*cycles*]	14198	12640

- C's binaries are slower:
  - Non-measurable on "bigger" CPUs
  - All the "embedded" platforms
  - Almost 13% gain on µCs
- Surprised again? :)
- ...

https://upload.wikimedia.org/wikipedia/commons/1/1d/Turtle3m.JPG

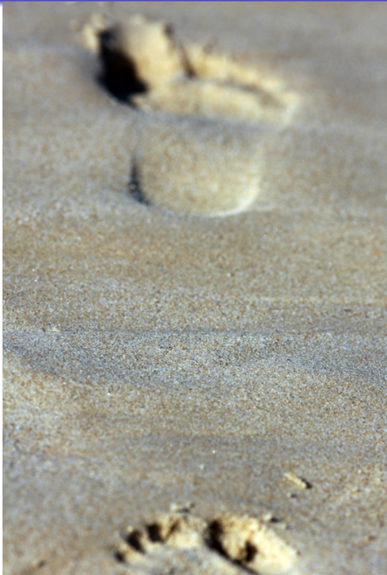
Observations

- C++ templates outperformed C macros
- C++ templates smaller than C macros



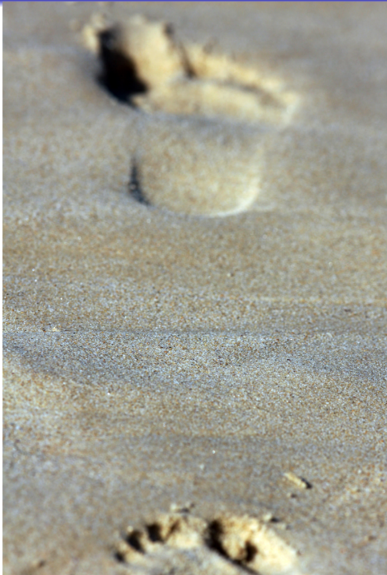
Observations

- C++ templates outperformed C macros
- C++ templates smaller than C macros
- Why templates won?
    - "Named hints"
    - Symbols
    - Inlining (opt-in)



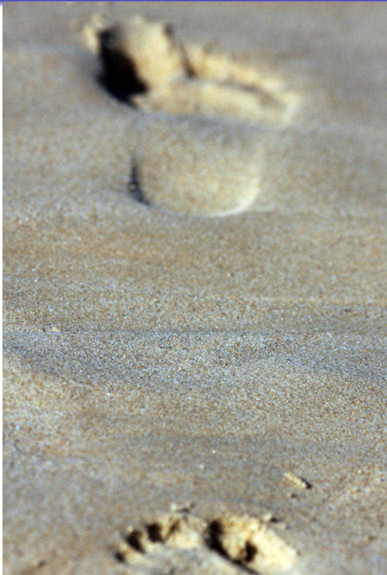
Observations

- C++ templates outperformed C macros
- C++ templates smaller than C macros
- Why templates won?
  - "Named hints"
  - Symbols
  - Inlining (opt-in)
- Why macros lost?
  - No "hints"...
  - No symbols
  - Preprocessor's "copy & paste"...
  - Code bloat



Observations

- C++ templates outperformed C macros
- C++ templates smaller than C macros
- Why templates won?
  - "Named hints"
  - Symbols
  - Inlining (opt-in)
- Why macros lost?
  - No "hints"...
  - No symbols
  - Preprocessor's "copy & paste"...
  - Code bloat
- C lacks good support for generic code:
  - Macros *vs.* language rules
  - `void*` API?
  - LTO often helps



https://upload.wikimedia.org/wikipedia/commons/2/24/Big_Foot_%286225546731%29.jpg

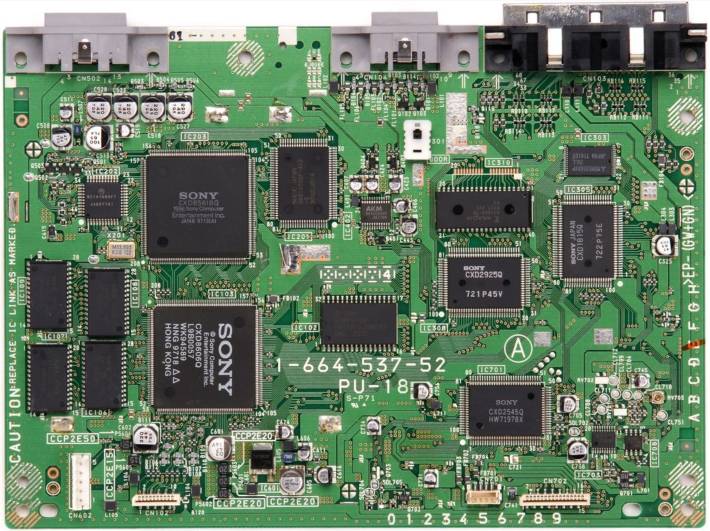
Part 7

1. The experiment
2. Proving ground
3. Baseline
4. Flow control
5. Generic programming
6. Conclusions

Software optimizations

https://upload.wikimedia.org/wikipedia/commons/a/af/GNU_Compiler_Collection_logo.svg
https://upload.wikimedia.org/wikipedia/en/4/4c/LLVM_Logo.svg

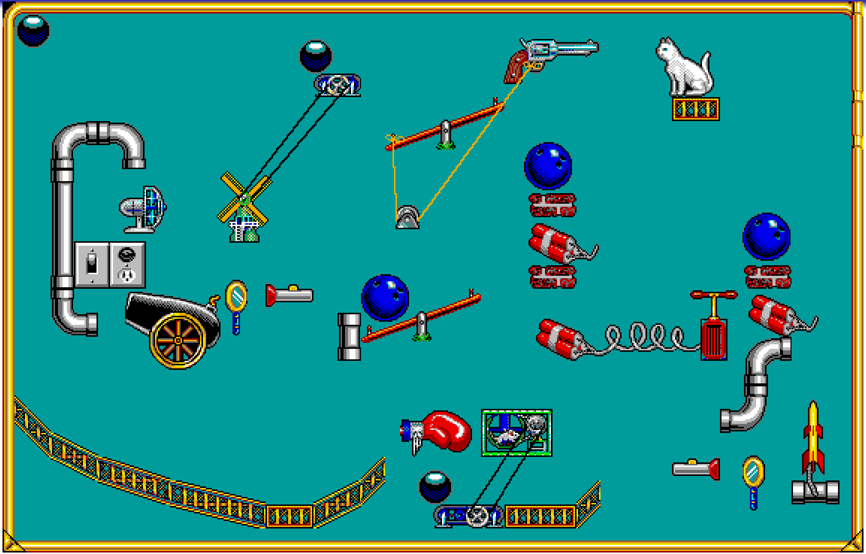
Meet the hardware!



https://upload.wikimedia.org/wikipedia/commons/d/d7/PSX-SCPH-5001-Motherboard.jpg

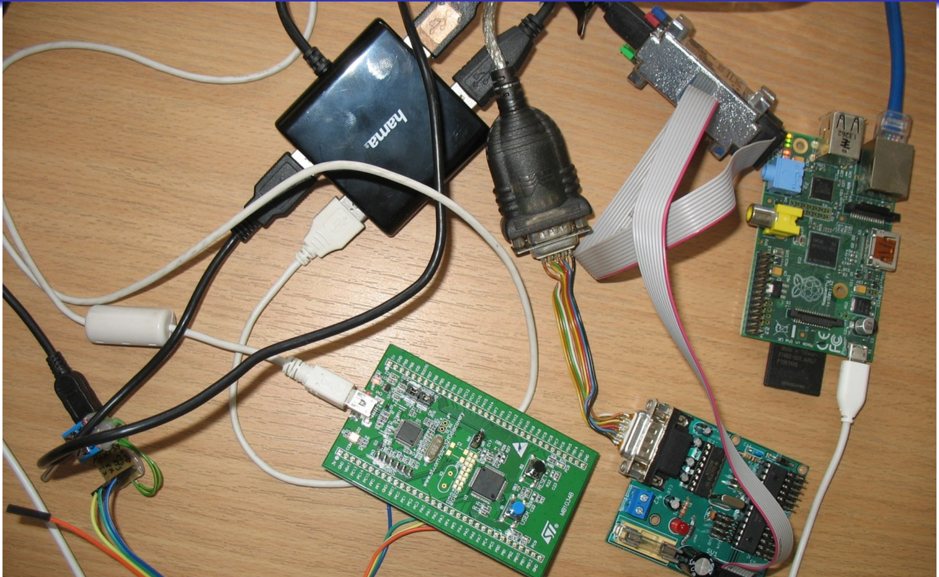
Working with optimizations

Working with optimizations



http://sfprod.shikadi.net/old/pic/timss.png

Testing time!



C++ or C?

- Avoid C:
  - Binary not smaller
  - Binary not faster
  - Lacks abstractions

C++ or C?

- Avoid C:
  - Binary not smaller
  - Binary not faster
  - Lacks abstractions
- Consider C if:
  - No other option
  - C-only team
  - Common ABI

C++ or C?

- Avoid C:
  - Binary not smaller
  - Binary not faster
  - Lacks abstractions
- Consider C if:
  - No other option
  - C-only team
  - Common ABI
- C++ as default:
  - Can do what C can
  - Offers much more
  - Lightweight abstractions

C++ or C?

- Avoid C:
  - Binary not smaller
  - Binary not faster
  - Lacks abstractions
- Consider C if:
  - No other option
  - C-only team
  - Common ABI
- C++ as default:
  - Can do what C can
  - Offers much more
  - Lightweight abstractions
- C++ is powerful
- Many libraries! (REST)

C++ or C?

- Avoid C:
  - Binary not smaller
  - Binary not faster
  - Lacks abstractions
- Consider C if:
  - No other option
  - C-only team
  - Common ABI
- C++ as default:
  - Can do what C can
  - Offers much more
  - Lightweight abstractions
- C++ is powerful
- Many libraries! (REST)

- HTTP request in 10 LOCs!

```cpp
#include <iostream>
#include <boost/network/protocol/http/client.hpp>
using boost::network::http::client;
int main()
{
  client          client;
  client::request request("http://duckduckgo.com");
  const auto response = client.get(request);
  std::cout << body(response) << std::endl;
}
```

C++ or C?

- Avoid C:
  - Binary not smaller
  - Binary not faster
  - Lacks abstractions
- Consider C if:
  - No other option
  - C-only team
  - Common ABI
- C++ as default:
  - Can do what C can
  - Offers much more
  - Lightweight abstractions
- C++ is powerful
- Many libraries! (REST)

- HTTP request in 10 LOCs!

```cpp
#include <iostream>
#include <boost/network/protocol/http/client.hpp>
using boost::network::http::client;
int main()
{
  client          client;
  client::request request("http://duckduckgo.com");
  const auto response = client.get(request);
  std::cout << body(response) << std::endl;
}
```

- Standard library grows
- "Do not pay for what you do not use"
- Embedded-friendly. . .

"Door light" project

"Door light" project

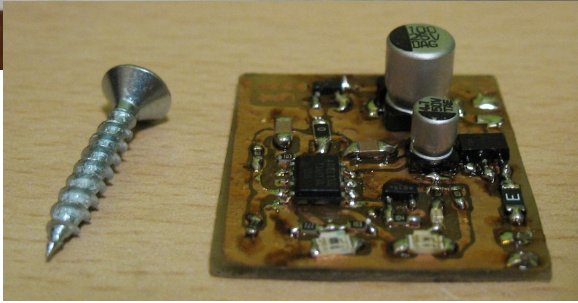
- Features:
  - Proximity sensor
  - Adaptive noise filtering
  - Dim-in/dim-out
  - Sleep modes
  - Watchdog
  - Power monitoring
  - 2 hardware variants

"Door light" project

- Features:
  - Proximity sensor
  - Adaptive noise filtering
  - Dim-in/dim-out
  - Sleep modes
  - Watchdog
  - Power monitoring
  - 2 hardware variants
- Written in C++11:
  - 900 LOC
  - 796 [B] binary

"Door light" project

- Features:
  - Proximity sensor
  - Adaptive noise filtering
  - Dim-in/dim-out
  - Sleep modes
  - Watchdog
  - Power monitoring
  - 2 hardware variants
- Written in C++11:
  - 900 LOC
  - 796 [B] binary

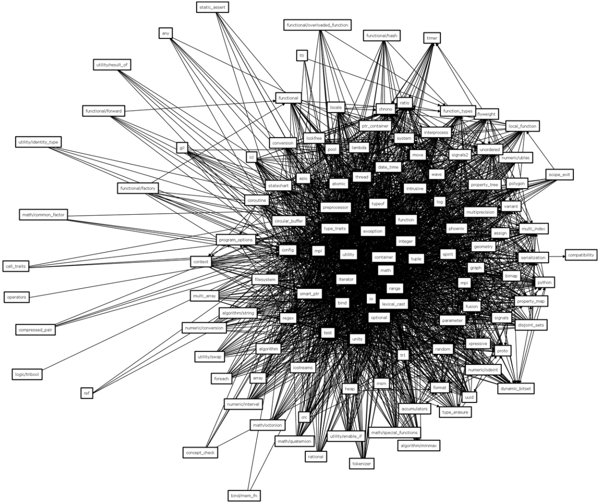


- AVR ATtiny13:
  - 1024 [B] ROM
  - 64 [B] RAM
  - 1 [MHz] (RC oscillator)

Other materials

- "*Real-Time C++: Efficient Object-Oriented and Template Microcontroller Programming*", 2013
  Christopher Kormanyos
- "*The C++ Programming Language*", 2013
  Bjarne Stroustrup
- "*Technical Report on C++ Performance - Open Standards*", 2004
  `http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2004/n1666.pdf`
- "*Hello Houston, czyli rzecz o błędów zgłaszaniu*", 2013
  `http://baszerr.eu/lib/exe/fetch.php/docs/hello_houston.pdf`
- "*Życie bez #ifdefów*", 2013
  `http://baszerr.eu/lib/exe/fetch.php/docs/zycie_bez_ifdefow.pdf`
- "*Effective modern C++*", 2015
  Scott Meyers
- Door light project, 2013
  `http://baszerr.eu/doku.php/prjs/door_light/door_light`

Questions?



http://www.meetingcpp.com/tl_files/blog/bda/boost154_libxml.png

Coming up next year!

VS

https://upload.wikimedia.org/wikipedia/commons/9/9f/Vimlogo.svg
https://upload.wikimedia.org/wikipedia/commons/5/5f/Emacs-logo.svg