Marc Mutz <marc@kdab.com>
KDAB (DE), Qt Widget Maintenance

Speed Up Your Qt 5 Programs Using C++11

Overview


```
#include <c++11>

int main() {
    if (programmer.lovesCxx11())
        if (!management.isConvinced())
            provideArguments();
        else
            reaffirm();
    else
        convince();
    return EXIT_SUCCESS;
}
```

C++11 @ Qt 5.0

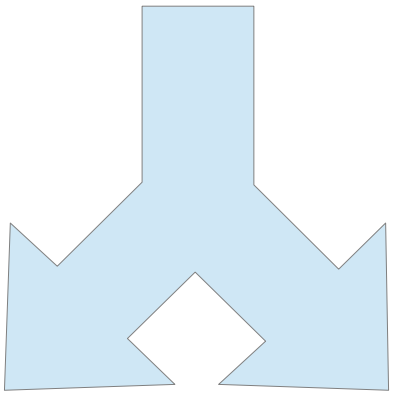
- constexpr added to many types
- move semantics added to a few types
- initializer_list added to most types
- very few N-ary ctors marked explicit, N ≥ 2
- = delete used almost ubiquitously
- noexcept added in a few central places

- other features not very usable in APIs

A Simple Example


```
static const QPoint triangle[] = {
    QPoint( -50,   0 ),
    QPoint( +50,   0 ),
    QPoint(   0, 100 ),
};

static const QPointF box[] = {
    QPointF( -1.0,  1.0 ),
    QPointF( -1.0, -1.0 ),
    QPointF(  1.0, -1.0 ),
    QPointF(  1.0,  1.0 ),
};

int main() { printf("%p%p", (void*)triangle, (void*)box); }
```

Prevents optimisations...



Results

Less!

Less!

```
text    data     bss     dec     hex filename
1576     552      16    2144     860 ex-qpoint.o2.c++11
1664     560     120    2344     928 ex-qpoint.o2.c++98
```

Less!

The C++11 Free Lunch

```
text    data     bss     dec     hex filename
1343     544      16    1903     76f ex-empty.o2.c++11
1343     544      16    1903     76f ex-empty.o2.c++98
```

Baseline
(`int main() {}`)

(GCC 4.8-20120823 / AMD64 / Linux)

Whodunnit?

The Qt-Project did...

```
class Q_CORE_EXPORT QPoint
{
public:
    Q_DECL_CONSTEXPR QPoint();
    Q_DECL_CONSTEXPR QPoint(int xpos, int ypos);

    Q_DECL_CONSTEXPR inline bool isNull() const;

    Q_DECL_CONSTEXPR inline int x() const;
    Q_DECL_CONSTEXPR inline int y() const;
    inline void setX(int x
    inline void setY(int y);
```

This is magic...

Whodunnit?

C++98

```
class Q_CORE_EXPORT QPoint
{
public:
                        QPoint();
                        QPoint(int xpos, int ypos);

                        inline bool isNull() const;

                        inline int x() const;
                        inline int y() const;
    inline void setX(int x);
    inline void setY(int y);
```

Whodunnit?

C++11

```
class Q_CORE_EXPORT QPoint
{
public:
    constexpr           QPoint();
    constexpr           QPoint(int xpos, int ypos);

    constexpr           inline bool isNull() const;

    constexpr           inline int x() const;
    constexpr           inline int y() const;
    inline void setX(int x);
    inline void setY(int y);
```

constexpr?

constexpr

- New keyword in C++11
- Can be applied to
  - (Free and Member) Functions
  - Variables
  - Constructors
- Enables Evaluation at Compile-Time

constexpr

- New keyword
- Can be applie
  - (Free and Mei
  - Variables
  - Constructors
- Enables Evalu

constexpr constructor

(also requires a trivial destructor)

“Literal Type”

const Type == compile-time constant

constexpr


```
static const QPoint triangle[] = {
    QPoint( -50,   0 ),
    QPoint( +50,   0 ),
    QPoint(   0, 100 ),
};

static const QPointF box[] = {
    QPointF( -1.0,  1.0 ),
    QPointF( -1.0, -1.0 ),
    QPointF(  1.0, -1.0 ),
    QPointF(  1.0,  1.0 ),
};

int main() { printf("%p%p", (void*)triangle, (void*)box); }
```

```
        .section        .rodata
        .align 16
        .type    _ZL8triangle, @object
        .size    _ZL8triangle, 24
_ZL8triangle:
        .long    -50
        .long    0
        .long    50
        .long    0
        .long    0
        .long    100
```

“read-only data”

C++11

And that is new...
...how?

Well...

~~"read-only data"~~

```
[...]
.section  .ctors,"aw",@progbits
.align 8
.quad   _GLOBAL__sub_I_main
.local  _ZL8triangle
.comm   _ZL8triangle,24,16
```

C++98

```
.type   _GLOBAL__sub_I_main, @function
_GLOBAL__sub_I_main:
movl $50, _ZL8triangle(%rip)
movl $0, 4+_ZL8triangle(%rip)
movl $50, 8+_ZL8triangle(%rip)
movl $0, 12+_ZL8triangle(%rip)
movl $0, 16+_ZL8triangle(%rip)
movl $100, 20+_ZL8triangle(%rip)
ret
[...]
.section  .ctors,"aw",@progbits
.align 8
.quad   _GLOBAL__sub_I_main
.local  _ZL8triangle
.comm   _ZL8triangle,24,16
```

“Dynamic Initialisation”

~~“read-only data”~~

C++98

constexpr

```
.type   _GLOBAL__sub_I_main, @function
_GLOBAL__sub_I_main:
movl $ 50, _ZL8triangle(%rip)
movl $0   _ZL8triangle(%rip)
movl $50,         triangle(%
movl $0, 12+_ZL
movl $0, 16+_ZL
movl $100
ret
                       gbits
                  _I_main
.     _ZL8triangl
.         ZL8triangle, , ,16
```

“Dynamic Initiali

Startup Costs?

Seriously???

Multithreading: Data Race on Initialisation?

Order of Initialisation?

</grey's anatomy>

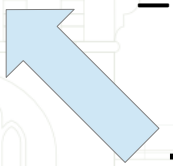
C++98

constexpr


```
extern int checkIndex(const QModelIndex&);

static const QModelIndex root;

int one() {
    return checkIndex(QModelIndex());
}

int two() {
    return checkIndex(root);
}
```

```
_Z3onev:
    subq    $40, %rsp
    movq    %rsp, %rdi
    movl    $-1, (%rsp)
    movl    $-1, 4(%rsp)
    movq    $0, 8(%rsp)
    movq    $0, 16(%rsp)
    call    _Z10checkIndexRK11QModelIndex@PLT
    addq    $40, %rsp
    ret
```

```
_Z3twov:
    leaq    _ZL4root(%rip), %rdi
    jmp     _Z10checkIndexRK11QModelIndex@PLT
```



Tail Call Optimisation

constexpr

- compilers don't fold instances of literal types
- standard doesn't permit it
- addresses must be unique
- you need to do the folding yourself
- learn to love static const variables :)

virtual functions

Talking about embarrssments...

```
void timerEvnet(QTimerEvent*) Q_DECL_OVERRIDE;
```

Error: `timerEvnet(QTimerEvent*)` doesn't override anyth

Doesn't speed up runtime...
...but development :)

Another Simple Example

C++98: QString::fromUtf8()
C++11: copy of a pointer

```
int main() {
    QString s(QStringLiteral("Hello,"));
    qDebug("%s", qPrintable(s));
    return 0;
}
```

C++11: won't throw

Guidelines

- add Q_DECL_OVERRIDE to virtual overrides
- use static const type in favour of temporaries
  - in apps, not DLLs/SOs/DYLIBs
  - even when just default-constructed
- prefer unnamed over named temporaries
- use QStringLiteral
- qDebug() doesn't throw anymore

C++11 @ Qt 5.1 (planned)

- add more constexpr
- add more noexcept
- add all missing move ctors
- experiment with rvalue refs on *this
- experiment with extern templates
- mark N-arg ctors explicit, N ≥ 2
- add Q_DECL_OVERRIDE everywhere
- implement C++11 API on our containers

Q_DECL_NOEXCEPT

Q_DECL_EQ_DELETE

Q_DECL_CONSTEXPR

Questions?

Q_DECL_NOTHROW

```
QString(QString &&other)
    : d(other.d) { other.d = 0; }
```

Suggestions?

```
QStringList sl;
sl.push_back("Hello, World!");
```

Q_DECL_OVERRIDE