Better Data Analysis with Modern C++
Know Your Tools

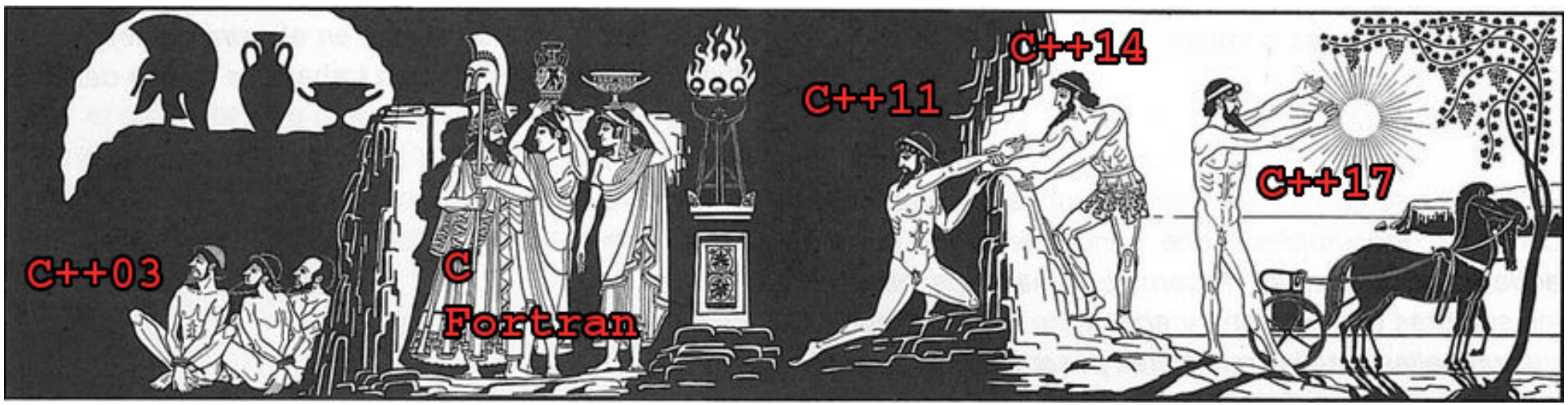
Sam Marinelli, Ivan Pogrebnyak

April 6, 2017

Modern C++

MICHIGAN STATE UNIVERSITY

- More expressive
- Highly abstract
- Less verbose
- Runs faster



- Working code from examples on GitHub:
  - https://github.com/ivankp/pgo_cxx

Get a new compiler

MICHIGAN STATE UNIVERSITY

- `wget http://mirrors.concertpass.com/gcc/releases/gcc-6.3.0/gcc-6.3.0.tar.bz2`
- `tar xvjf gcc-6.3.0.tar.bz2`
- `cd gcc-6.3.0`
- `./contrib/download_prerequisites`
- `cd ..`
- `mkdir build-gcc-6.3.0`
- `cd build-gcc-6.3.0`
- `$PWD/../gcc-6.3.0/configure --prefix=$HOME/gcc-6.3.0 --enable-languages=c,c++,fortran --with-default-libstdcxx-abi=gcc4-compatible --enable-bootstrap --enable-threads=posix --with-long-double-128 --enable-long-long --enable-lto --enable-gnu-unique-object --enable-gold --with-system-zlib --disable-nls`
- `make -j8`
- `make -j8 install`

Get a new compiler

MICHIGAN STATE UNIVERSITY

- After installation is complete, you can remove `build-gcc-6.3.0`.
- If you want to use the new compiler by default, only two environmental variables are needed
  - `export PATH=$HOME/gcc-6.3.0/bin:$PATH`
  - `export LD_LIBRARY_PATH=$HOME/gcc-6.3.0/lib64:$LD_LIBRARY_PATH`
- Or you can just call the g++ executable by the whole path.
- New headers and libraries are automatically looked up first during compilation and linking. No extra environmental variables needed.
- Setting `LD_LIBRARY_PATH` can be avoided by compiling with `-Wl,-rpath=$HOME/gcc-6.3.0/lib64`

Range-based for loop

MICHIGAN STATE UNIVERSITY

Example 1:

```
for (int i : {1,3,5,7,9,11,13,17})
  cout << i << endl;
```

Example 2:

```
std::vector<std::vector<TH1*>> hists;
. . .
for (auto& hh : hists)
  for (auto* h : hh)
    cout << h->GetName() << endl;
```

Range-based for loop

MICHIGAN STATE UNIVERSITY

How it works

```
std::vector<int> vec;
for (auto x : vec) { . . . }
```

is equivalent to

```
for (std::vector<int>::iterator it  = vec.begin,
                                end = vec.end();
     it!=end; ++it)
{
  int x = *it;
  . . .
}
```

λ functions

MICHIGAN STATE UNIVERSITY

- Write functions inline!
- Great with standard algorithms like `std::sort`, `std::accumulate`, etc.

Example: find distribution with highest mean

```
std::vector<TH1D*> hists;
. . .
auto highest_mean = *std::max_element(
  hists.begin(), hists.end(), [](auto h, auto i) {
    return h->GetMean() < i->GetMean();
  }
);
```

λ functions

MICHIGAN STATE UNIVERSITY

- Capture variables from the environment in the `[]`.

Example: sort functions by their maxima or minima

```
std::vector<TF1*> funcs;
bool ascending;
. . .
std::sort(
  funcs.begin(), funcs.end(), [ascending](auto f, auto g) {
    double f_max = f->GetMaximum(),
           g_max = g->GetMaximum();
    return ascending ^ g_max < f_max;
  }
);
```

Variadic templates

MICHIGAN STATE UNIVERSITY

$$\sum_i x_i^2$$

```
template <typename T>
constexpr T sq(T x) noexcept { return x*x; }

template <typename T, typename... TT>
constexpr T sq(T x, TT... xx) noexcept { return sq(x)+sq(xx...); }
```

Usage example:

```
double hypotenuse = std::sqrt(sq(3.,4.)); // 5
```

Variadic templates

MICHIGAN STATE UNIVERSITY

```cpp
#include <sstream>

template <typename S, typename T>
inline void cat_impl(S& ss, const T& x) {
  ss << x;
}

template <typename S, typename T, typename... TT>
inline void cat_impl(S& ss, const T& x, const TT&... xx) {
  ss << x;
  cat_impl(ss,xx...);
}

template <typename... TT>
inline std::string cat(const TT&... xx) {
  std::stringstream ss;
  cat_impl(ss,xx...);
  return ss.str();
}
```

Variadic templates

MICHIGAN STATE UNIVERSITY

Example 1:

```
cout << cat("char_array",' ',5,' ',std::fixed,std::setprecision(2),4.2)
     << endl; // char_array 5 4.20
```

Example 2:

```
TFile file("file.root");

for (int j : {1,2,3}) {
  const auto name = cat("jet",j,"_pT");
  TH1 *hist = dynamic_cast<TH1*>(file->Get(name.c_str()));

  if (!hist) throw std::runtime_error(cat(
    name," histogram does not exists in file ",file.GetName()));

  if (hist->GetEntries()==0) throw std::runtime_error(cat(
    hist->GetName()," histogram is empty"));

  . . .
}
```

C++17 fold expressions

MICHIGAN STATE UNIVERSITY

```
template <typename... Args>
inline std::string cat(const Args&... args) {
  std::stringstream ss;
  (ss << ... << args); // fold expression
  return ss.str();
}
```

Random-number generators

MICHIGAN STATE UNIVERSITY

- New `random` header provides several pseudorandom-number generators and 20 commonly used distributions, including many not implemented in ROOT, such as Bernoulli, negative-binomial, γ, etc.

Smart pointers

MICHIGAN STATE UNIVERSITY

- These own dynamically allocated memory and clean it up automatically (`delete` called on destruction of pointer).

Example: distribution of a sample mean

```
std::random_device device;
std::mt19937 engine(device());
std::uniform_real_distribution<double> dist;

TH1::AddDirectory(false);
auto sample_mean = std::make_unique<TH1D>("sample_mean", "", 100, 0.0,
                                          1.0);
for (std::size_t i = 0; i < 1000000; ++i) {
  auto sample = std::make_unique<TH1D>("sample", "", 100, 0.0, 1.0);
  for (std::size_t j = 0; j < 10; ++j) sample->Fill(dist(engine));
  sample_mean->Fill(sample->GetMean());
}
```

- See also `std::shared_ptr` for shared ownership.

Move semantics

MICHIGAN STATE UNIVERSITY

- Prevents unnecessary copying of dynamically allocated memory.
- Compiling with `-std=c++11` or later turns these on automatically in the standard library.

Example: return a `std::vector` from a function

```
std::vector<double> sample_dist(std::size_t n_samples) {
  . . .
}

int main() {
  auto samples = sample_dist(1000000);
  . . .
}
```

Move semantics

MICHIGAN STATE UNIVERSITY

- Combine with `std::unique_ptr` for safe, portable dynamic memory.

Example: function that generates distributions

```
std::unique_ptr<TH1D> sample_dist(std::size_t n_samples) {
  auto hist = std::make_unique<TH1D>("hist", "", 100, 0.0, 1.0);
  . . .
  return hist;
}

int main() {
  auto hist = sample_dist(1000000);
  . . .
}
```

Uniform initialization

MICHIGAN STATE UNIVERSITY

```
struct point { double x, y; }; // local class
std::vector<point> points {
  {0.1, 0.3},
  {3.1, 4,7},
  {55., 34.}
};

for (const auto& p : points)
  cout << '(' << p.x << ',' << p.y << ')' << endl;
```

auto and decltype

MICHIGAN STATE UNIVERSITY

```
std::vector<some_very_useful_but_very_long_type> vec;

auto it = vec.begin();

decltype(*it) x;
```

- The type of `it` is `std::vector<some_very_useful_but_very_long_type>::iterator`;
- The type of `x` is `some_very_useful_but_very_long_type`;

Regular expressions

MICHIGAN STATE UNIVERSITY

Suppose you have histograms with name like

- `jet1_pT`
- `jet2_pT`
- `jet1_rapidity`
- `lepton2_rapidity`
- `higgs_pT`
- `higgs_mass`

but you want to do something for only jet and Higgs pT histograms.

```
std::regex re("(jet[0-9]|higgs)_pT");
for (TH1* h : hists) {
  if (std::regex_match( h->GetName(), re )) {
    . . . // Do something
  }
}
```

<array>

MICHIGAN STATE UNIVERSITY

`std::array` is like `std::vector`, but statically allocated and with the size a part of its type.

```
using p4 = std::array<double,4>; // can use it to represent 4-momentum
std::vector<p4> particles; // will be contiguously allocated
```

Other similar aggregate templates (statically allocated containers):

- `std::pair<T1,T2>` – 2 heterogeneous types
- `std::tuple<Ts...>` – n heterogeneous types
- `std::array<T,N>` – n homogeneous types

An analysis example

MICHIGAN STATE UNIVERSITY

- An example of analysis code, reading a `TTree` from a ROOT file, making histograms of particle's transverse momentum for different types of events.
- Example of using `std::unordered_map` and a timed counter based on `C++11 <chrono>`.
- Also uses other previously discussed features.

Link to the code

A fitting example

MICHIGAN STATE UNIVERSITY

- An example of fitting code, using lambda functions with `TMinuit`.
- The wrapper for `TMinuit` also demonstrates other modern `C++` features.

Link to the code